From Entropy to Epiplexity: Rethinking Information for Computationally Bounded Intelligence

Marc Finzi[*1]   **Shikai Qiu**[*2]   **Yiding Jiang**[*1]   **Pavel Izmailov**[2]   **J. Zico Kolter**[1]
Andrew Gordon Wilson[2]

[1]Carnegie Mellon University   [2]New York University

Abstract

Can we learn more from data than existed in the generating process itself? Can new and useful information be constructed from merely applying deterministic transformations to existing data? Can the learnable content in data be evaluated without considering a downstream task? On these questions, Shannon information and Kolmogorov complexity come up nearly empty-handed, in part because they assume observers with unlimited computational capacity and fail to target the useful information content. In this work, we identify and exemplify three seeming paradoxes in information theory: (1) information cannot be increased by deterministic transformations; (2) information is independent of the order of data; (3) likelihood modeling is merely distribution matching. To shed light on the tension between these results and modern practice, and to quantify the value of data, we introduce *epiplexity*, a formalization of information capturing what computationally bounded observers can learn from data. Epiplexity captures the structural content in data while excluding time-bounded entropy, the random unpredictable content exemplified by pseudorandom number generators and chaotic dynamical systems. With these concepts, we demonstrate how information can be created with computation, how it depends on the ordering of the data, and how likelihood modeling can produce more complex programs than present in the data generating process itself. We also present practical procedures to estimate epiplexity which we show capture differences across data sources, track with downstream performance, and highlight dataset interventions that improve out-of-distribution generalization. In contrast to principles of model selection, epiplexity provides a theoretical foundation for *data selection*, guiding how to select, generate, or transform data for learning systems.

1 Introduction

As AI research progresses towards more general-purpose intelligent systems, cracks are beginning to show in mechanisms for grounding mathematical intuitions. Much of learning theory is built around controlling generalization error with respect to a given distribution, treating the training distribution as fixed and focusing optimization effort on the choice of model. Yet modern systems are expected to transfer across tasks, domains, and objectives that were not specified at training time, often after large-scale pretraining on diverse and heterogeneous data. In this regime, success or failure frequently hinges less on architectural choices than on what data the model was exposed to in the first place. Pursuing broad generalization to out-of-distribution tasks forces a shift in perspective: instead of treating data as given and optimizing for in-distribution performance, we need to choose and curate data to facilitate generalization to unseen tasks. This shift makes the value of data itself a central question—how much usable, transferable information can a model acquire from training? In other words, instead of model selection, how do we perform *data selection*? On this question, existing theory offers little guidance and often naively contradicts empirical observations.
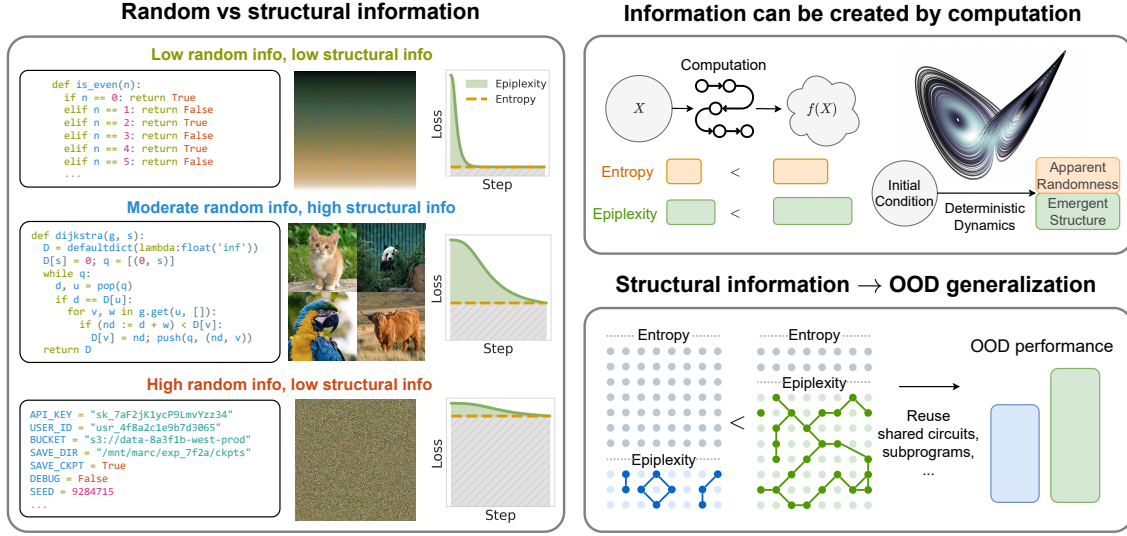
---

[*]Equal contribution.

Figure 1: **Illustration of random vs structural information.** (**Left**) Illustration of random vs structural information of different data for computationally-bounded observers, which we formalize with time-bounded entropy and epiplexity (Section 3) and can be estimated from loss curves of neural networks trained on that data (Section 4). (**Top Right**) Unlike other forms of information, time-bounded entropy and epiplexity can be increased through computational processes, such as simulating dynamical systems (cellular automation, Lorenz equations) and interventions like changing the data ordering, which can produce apparent randomness but also learnable, emergent structures like gliders and the Lorenz attractor invariant measure (Section 5). (**Bottom Right**) Whereas time-bounded entropy captures the in-distribution randomness and unpredictability, epiplexity measures the amount of structural information the model extracts from the data to its weights, which can be useful for OOD tasks such as by reusing learned circuits shared between the in-distribution and OOD tasks.

Consider synthetic data, crucial for further developing model capabilities (Abdin et al., 2024; Maini et al., 2024) when existing natural data are exhausted. Existing concepts in information theory like the data processing inequality appear to suggest that synthetic data adds no additional value. Questions about what information is transferred to a given model seem naturally within the purview of information theory, yet, quantifying this information with existing tools proves to be elusive. Even basic questions, such as the source of the information in the weights of an AlphaZero game-playing model (Silver et al., 2018), are surprisingly tricky to answer. AlphaZero takes in zero human data, learning merely from the deterministic rules of the game and the AlphaZero RL algorithm, both of which are simple to describe. Yet the resulting models achieve superhuman performance and are large in size. To assert that AlphaZero has learned little to no information in this process is clearly missing the mark, and yet both Shannon and algorithmic information theory appear to say so.

In this paper, we argue that the amount of structural information a *computationally bounded* observer can extract from a dataset is a fundamental concept that underlies many observed empirical phenomena. As we will show, existing notions from Shannon and algorithmic information theory are inadequate when forced to quantify this type of information. These frameworks often lend intuitive or mathematical support to beliefs that, in fact, obscure important aspects of empirical phenomena. To highlight the limitations of classical frameworks and motivate the role of computational constraints in quantifying information, we identify and demonstrate three *apparent paradoxes*: statements which can be justified mathematically by Shannon and algorithmic information theory, and yet are in tension with intuitions and empirical phenomena.

Paradox 1: **Information cannot be increased by deterministic processes.** For both Shannon entropy and Kolmogorov complexity, deterministic transformations cannot meaningfully increase the information content of an object. And yet, we use pseudorandom number generators to produce randomness, synthetic data improves model capabilities, mathematicians can derive new knowledge by reasoning from axioms without external information, dynamical systems produce emergent phenomena, and self-play loops like AlphaZero learn sophisticated strategies from games (Silver et al., 2018).

Paradox 2: **Information is independent of factorization order.** A property of both Shannon entropy and Kolmogorov complexity is that total information content is invariant to factorization: the information from observing first X and then Y is the same as observing Y followed by X. On the other hand, LLMs learn better on English text ordered left-to-right than reverse ordered text, picking out an "*arrow of time*" (Papadopoulos et al., 2024; Bengio et al., 2019), and we have cryptography built on the existence of functions that are computationally hard to predict in one direction and easy in another.

Paradox 3: **Likelihood modeling is merely distribution matching.** Maximizing the likelihood is often equated with matching the training data generating process: the true data-generating process is a perfect model of itself, and no model can achieve a higher expected likelihood. As a consequence, it is often assumed that a model trained on a dataset cannot extract more structure or learn useful features that were not used in generating the data. However, we show that a computationally-limited observer can in fact uncover much more structure than is in the data generating process. For example, in Conway's game of life the data are generated via simple programmatic rules that operate on two-dimensional arrays of bits. Applying these simple rules sequentially, we see emergent structures, such as different species of objects that move and interact in a predictable way. While an unbounded observer can simply simulate the evolution of the environment exactly, a computationally-bounded observer would make use of the emergent structures and learn the different types of objects and their behaviors.

The tension between these theoretical statements and empirical phenomena can be resolved by imposing computational constraints on the observer and separating the random content from the structural content. Drawing on ideas from cryptography, algorithmic information theory, and these unexplained empirical phenomena, we define a new information measure, **epiplexity** (epistemic complexity), which formally defines the amount of structural information that a computationally-bounded observer can extract from the data (Section 3, Definition 8). Briefly, epilexity is the information in the model that minimizes the description length of data under computational constraints. A simple heuristic measurement is the area under the loss curve above the final loss, while a more rigorous approach uses the cumulative KL divergence between a teacher and student model (Section 4, Figure 2).

Our definitions capture the intuition that an object contains both random, inherently unpredictable information (entropy), and predictable structured information that enables observers to generalize by identifying patterns (epiplexity). In Figure 1 (left) we illustrate this divide. In the top row, we have highly redundant and repetitive code and simple color gradients, which have little information content, be it structural or random. In the middle row, we have the inner workings of an algorithm and pictures of animals, showing complex, long-range interdependencies between the elements from which a model can learn complex features and subcircuits that are helpful even for different tasks. In contrast, on the bottom, we have random data with little structure: configuration files with randomly generated API keys, file paths, hashes, arbitrary boolean flags have negligible learnable content and no long-range dependencies or complex circuits that result from learning on this task. Similarly, uniformly shuffled pixels from the animal pictures have high entropy but are fundamentally unpredictable, and no complex features or circuits arise from training on these data.

An essential property of our formulation is that information is *observer dependent*: the same object may appear random or structured depending on the computational resources of the observer. For instance, the output of a strong pseudorandom generator appears indistinguishable from true randomness to any polynomial-time observer lacking the secret key (seed), regardless of the algorithm or function class. In other situations, such as chaotic dynamical systems, both apparently random behavior is produced along with structure: the state of the system cannot be predicted precisely over long time-scales, but such observers may still learn meaningful predictive distributions, as shown by the invariant measure in Figure 1 (top right).

Models trained to represent these distributions are computer programs, and substructures within these programs, like circuits for performing specific tasks, or induction heads (Olsson et al., 2022), can be reused even for seemingly unrelated data. This view motivates selecting high epiplexity data that induces more structural information in the model, since these structures can then be reused for unseen out-of-distribution (OOD) tasks, as illustrated in Figure 1 (bottom right). We emphasize, however, that epiplexity is a measure of information, *not* a guarantee of OOD generalization to specific tasks. Epiplexity quantifies the amount of structural information a model extracts, while being agnostic to whether these structures are relevant to a *specific* downstream task.

To build intuition, we explore a range of phenomena and provide experimental evidence for behaviours that are poorly accounted for by existing information-theoretic tools, yet naturally accommodated by epiplexity. We show that information *can* be created purely through computation, giving insights into synthetic data (subsection 5.1). We examine how certain factorizations of the same data can increase structural information and downstream OOD performance—even as they result in worse training loss (subsection 5.2). We show why likelihood modeling is more than distribution matching, identifying induction and emergence as two settings where the observer can learn more information than was present in the data generating process (subsection 5.3). By measuring epiplexity, we can better understand why pre-training on text data transfers more broadly than image data, and why certain data selection strategies for LLMs are empirically successful (Section 6). Together, our results provide clarity on the motivating questions: the information content of data can be compared independently of a specific task, new information can be created by computation, and models can learn more information than their generating processes contain.

In short, we identify a disparity between existing concepts in information theory and modern practice, embodied by three apparent paradoxes, and introduce epiplexity as a measurement of structural information acquired by a computationally-bounded observer to help resolve them. We formally define epiplexity in Section 3 (Definition 8) and present measurement procedures in Section 4. In Section 5, we show how epiplexity and time-bounded entropy shed light on these paradoxes, including induction and emergent phenomena. Finally, in Section 6, we demonstrate that epiplexity correlates with OOD generalization, helping explain why certain data enable broader generalization than others.

2 Background

In order to define the interesting, structural, and predictive component of information, we must separate it out from random information—that which is fundamentally unpredictable given the computational constraints of the observer. Along the way, we will review algorithmic randomness as developed in algorithmic information theory as well as notions of pseudo-randomness used in cryptography, and how these concepts crucially depend on the observer.

2.1 What Does it Mean for An Object to Be Random?

Random Variables and Shannon Information. Many common intuitions about randomness start from random variables and Shannon information. A random variable defines a map from a given measurable probability space to different outcomes, with probabilities corresponding to the measure of the space that lead to a certain outcome. Shannon information assigns to each outcome x a self-information (or surprisal) $\log 1/P(x)$ based on the probability P, and an entropy for the random variable $\mathrm{H}(X) = \mathbb{E}[\log 1/P(X)]$, which provides a lower bound on the average code length needed to *communicate* samples to another party (Shannon, 1948). In Shannon's theory, information comes only from distributions and random variables. Objects which are not random must contain no information. As a result, non-random information is seemly contradictory, and thus we must draw from a broader mathematical perspective to describe such concepts.

In the mid 1900s, mathematicians were interested in formalizing precisely what it means for a given sample to be a random draw from a given distribution, to ground the theory of probability and random variables (Shafer and Vovk, 2006). A central consideration involves a uniformly sampled binary sequence $u_{1:\infty}$ from which other distributions of interest can be constructed. This sequence can also be interpreted as the binary expression of a number $[0, 1)$. Intuitively, one might think that all sequences should be regarded as equally random, as they are all equally likely according to the probability distribution: $1111111\ldots$ has the same probability mass as $10011101\ldots$ and also the same self-information. However, looking at statistics on these sequences reveals something missing from this perspective; from the law of large numbers, for example, it must be that $\lim_{N \to \infty} \frac{1}{N} \sum_{i=1}^{N} u_i = 0.5$, which is clearly not satisfied by the first sequence of 1s.

Martin-Löf Randomness: No algorithm exists to predict the sequence. Initial attempts were made to formalize randomness as sequences which pass all statistical tests for randomness, such as the law of large numbers for selected substrings. However, under such definitions all sequences fail to be random since tests like $u_{1:\infty} \neq y_{1:\infty}$ for any particular sequence y must also be included (Downey and Hirschfeldt, 2019). The solution to these issues was found by defining random sequences not as those that pass all tests of randomness, but those that pass all *computable* tests of randomness, in a formalization known as Martin-Löf randomness (Martin-Löf, 1966). As it turned out, this definition is equivalent to a number of seemingly distinct definitions, such as the inability for any gambler to exploit properties of the sequence to make a profit, or that all prefixes of the random sequence should be nearly incompressible (Terwijn, 2016). For this last definition, we must invoke Kolmogorov complexity, a notion of compressibility and a key concept in this paper.

Definition 1 (Prefix Kolmogorov complexity (Kolmogorov, 1968; Chaitin, 1975)) *Fix a universal prefix-free Turing machine $\mathcal{U}$. The (prefix) Kolmogorov complexity of a finite binary string x is $K(x) = \min\{|p|: \mathcal{U}(p) = x\}$. That is, $K(x)$ is the length of the shortest self-delimiting program (a program which also encodes its length) that outputs x and halts.*

Due to the universality of Turing machines, the Kolmogorov complexity for two Turing machines (or programming languages) $\mathcal{U}_1$ and $\mathcal{U}_2$ differ by at most a constant, $|K_{\mathcal{U}_1}(x) - K_{\mathcal{U}_2}(x)| \leq C$, where the constant C depends only on $\mathcal{U}_1, \mathcal{U}_2$, but not on x (Li et al., 2008).

Definition 2 (Martin–Löf random sequence (Martin-Löf, 1966)) *An infinite sequence $x_{1:\infty} \in \{0,1\}^{\mathbb{N}}$ is Martin–Löf random iff there exists a constant c such that for all n, $K(x_{1:n}) \geq n - c$. Using this criterion, all computable randomness tests are condensed into a single incomputable randomness test concerning Kolmogorov complexity.*

One can extend Martin-Löf randomness to finite sequences. We say that a sequence $x \in \{0,1\}^n$ is c-random if $K(x) > n - c$. Equivalently, *randomness discrepancy* is defined as $\delta(x) = n - K(x)$,

which measures how far away x is from having maximum Kolmogorov complexity. A sequence x is c-random if $\delta(x) < c$. High Kolmogorov complexity, low randomness discrepancy, sequences are overwhelmingly likely when sampled from uniform randomly sampled random variables. From Kraft's inequality (Kraft, 1949; McMillan, 1956), there are at most 2^{n-c} (prefix-free) programs of length $L \leq n - c$, therefore in the 2^n possibilities in uniformly sampling $X \sim U_n$, the probability that $K(X)$ is size L or smaller is $P(K(X) \leq n - c) = P(\delta(X) \geq c) < 2^{-c}$. The randomness discrepancy of the sequence can thus be considered as a test statistic for which one would reject the null hypothesis that the given object X is indeed sampled uniformly at random (Grünwald et al., 2008). In order for a sequence to have low randomness discrepancy there must be no discernible pattern, and thus there is an objective sense in which 1001011100 is more random than 0101010101.

Given the Martin-Löf definition of infinite random sequences, every random sequence is incomputable; in other words, there is no program that can implement the function $\mathbb{N} \to \{0, 1\}$ which produces the bits of the sequence. One should contrast such random numbers from those like $\pi/4$ or $e/3$, which though transcendental, are computable, as there exist programs that can compute the bits of their binary expressions. While the computable numbers in $[0, 1)$ form a countable set, algorithmically random numbers in $[0, 1)$ are uncountably large in number. With the incomputability of random sequences in mind we can appreciate the Von Neumann quote

> "Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin." (Von Neumann, 1951)

which anticipates the Martin–Löf formalization that came later. But this viewpoint also misses something essential, as evidenced by the success of pseudorandom number generation, derandomization, and cryptography.

Cryptographic Randomness: No polynomial time algorithm exists to predict the sequence. An important practical and theoretical development of random numbers has come from the cryptography community, by once again limiting the computational model of the observer.

Rather than passing all computable tests as with Martin-Löf randomness, cryptographically secure pseudorandom number generators (CSPRNG or PRG) are defined as functions which produce sequences which pass all *polynomial time* tests of randomness.

Definition 3 (Non-uniform CSPRNG (Blum and Micali, 1982; Goldreich, 2006)) *A function G stretching k input bits into n output bits is a CSPRNG iff over the randomness of the seed s, no D with polynomial-size advice can distinguish the generation from a random sequence more than a negligible fraction of the time. More precisely, G is a (non-uniform) CSPRNG iff for every non-uniform probabilistic polynomial time algorithm $D_k : \{0, 1\}^n \to \{0, 1\}$ (making use of advice strings $\{a_k\}_{k \in \mathbb{N}}$ of length $\mathrm{poly}(k)$) has at most negligible advantage $\epsilon(k)$ distinguishing outputs of G from uniformly random sequences $u \sim U_n$:*

$$\epsilon(k) = \left| \Pr_{s \sim U_k}[D_n(G(s)) = 1] - \Pr_{u \sim U_n}[D_n(u) = 1] \right| < \mathrm{negl}(k) .^1 \tag{1}$$

The definition of indistinguishability via polynomial time tests is equivalent to a definition on the failure to predict the next element of a sequence given the previous elements: no polynomial time predictor can predict the next bit of the sequence with probability negligibly better than random guessing (Yao, 1982).

---

1. Here $\mathrm{negl}(k)$ means that the function decays faster than the reciprocal of any polynomial ($\mathrm{negl}(k) < \dfrac{1}{k^c}$ for all integers $c > 0$ and sufficiently large k).

Following from the indistinguishability definition, randomness of this kind can be substituted for Martin-Löf randomness in the vast majority of practical circumstances.[2] For example, if a use-case of randomness that runs in polynomial time like quicksort, and takes more iterations to run with CSPRNG sequences than with truly random sequences, and this difference could be determined within polynomial time such as by measuring the quicksort runtime, then this construction could be used as a polynomial time distinguisher, which by the definition of CSPRNG does not exist. If CSPRNGs exist, then quicksort must run nearly as fast using pseudorandom number generation as it does with truly random sequences.

The existence of CSPRNGs hinges on the existence of *one way functions*, from which CSPRNGs and other cryptographic primitives are constructed, forming the basis of modern cryptography. For example, the backbone algorithm for parallel random number generation in Jax (Bradbury et al., 2018), works to create random numbers $u_1, u_2, \ldots u_N$ by simply encrypting the numbers $1, 2, \ldots, N$: $u_k = E(k, s)$ where the encryption key s is the random seed and E is the threefish block cypher (Salmon et al., 2011). Block ciphers, like other primitives, are constructed using one way functions.

Definition 4 (Non-uniform one-way function, OWF (Yao, 1982; Goldreich, 2006)) *Let* $f :$ $\{0,1\}^n \to \{0,1\}^m$ *(with* $m > n$*) be computable in time* $\text{poly}(n)$ *where* $n = |x|$*. We say* f *is one-way against non-uniform adversaries if for every non-uniform PPT algorithm* A_n *(i.e., a polynomial-time algorithm* A *with advice strings* $\{a_n\}_{n \in \mathbb{N}}$ *of length* $\text{poly}(n)$*),*

$$\Pr_{x \sim U_n} \left[A_n(f(x)) \in f^{-1}(f(x)) \right] < \text{negl}(n),$$

where the probability is over the uniform choice of x *(and the internal randomness of* A*).*

While cryptographers are most interested in the polynomial versus nonpolynomial compute separations for security, one way functions with respect to less extreme compute separations have been constructed and are believed to exist, for example for quadratic time (Merkle, 1978), quasipolynomial time (Liu and Pass, 2024), and even constraints on circuit depth (Applebaum, 2016). While the results we prove in this paper are based on the polynomial vs nonpolynomial separation in cryptographic primitives, it seems likely that a much wider array of compute separations are relevant for information in the machine learning context even if not as important for cryptography. For example, the separations between quadratic or cubic time and higher order polynomials may be relevant to transformer self attention, or gaps between fixed circuit depth and variable depth as made possible with chain of thought or other mechanisms.

2.2 Random vs Structural Information

With these notions of randomness in hand, we can use what is random to define what is not random. In algorithmic information theory, there is a lesser known concept that captures exactly this idea, known as *sophistication* (Koppel, 1988), which has no direct analog in Shannon information theory. While several variants of the definition exist, the most straightforward is perhaps the following:

Definition 5 (Naive Sophistication (Mota et al., 2013)) *Sophistication, like Kolmogorov complexity, is defined on individual bitstrings, and it uses the compressibility criterion from Martin-Löf randomness to carve out the random content of the bitstring. Sophistication is defined as the smallest Kolmogorov complexity of a set* S *such that* x *is a random element from that set (at randomness discrepancy of* c*).*

$$\text{nsoph}_c(x) = \min_S : \{K(S) : K(x \mid S) > \log|S| - c\} \tag{2}$$

---

2. Specifically, when the difference between outcomes can be measured in polynomial time.

Informally, sophistication describes the structural component of an object; however, it is surprisingly difficult to give concrete examples of high sophistication objects. The difficulty of finding high sophistication objects is a consequence of Chaitin's incompleteness theorem (Chaitin, 1974). This theorem states that in a given formal system there is a constant L for which there are no proofs that any specific string x has $K(x) > L$, even though nearly all strings have nearly maximal complexity. Since $\text{nsoph}_c(x) > L$ implies $K(x) > L - O(1)$, there can be no proofs that the sophistication of a particular string exceeds a certain constant either. It is known that high sophistication strings exist by a diagonalization argument (Antunes et al., 2005), but we cannot pinpoint any specific strings which have high sophistication. On typical Turing machines, L is often not more than a few thousand (Chaitin, 1998), far from the terabytes of information that frontier AI models have encoded.

We look towards complex systems and behaviors as likely examples of high sophistication objects; however in many of these cases the objects could conceivably be produced by simpler descriptions given tremendous amounts of computation. The mixing of two fluids for example can produce extremely complex transient behavior due to the complexities of fluid dynamics; however, with access to unlimited computation and some appropriately chosen random initial data one should be able to reproduce the exact dynamics (Aaronson et al., 2014). Owing to the unbounded compute available for the programs in sophistication, many complex objects lose their complexity. Additionally, for strings that *do* have high sophistication, the steps of computation required for the optimal program grow faster than any computable function with the sophistication content (Ay et al., 2010). For a computationally-bounded observer, an encrypted message or a *cryptographically secure pseudo-random number generator* (CSPRNG) output *is* random, and measurements that do not recognize this randomness do not reflect the circumstances of this observer. These limitations of sophistication leads to a disconnect with real systems with observers that have limited computation, and it is our contention that this disconnect is an essential one, central to phenomena such as emergence, induction, chaos, and cryptography.

2.3 The Minimum Description Length Principle

Finally, we review the minimum description length principle (MDL), used as a theoretical criterion for model selection, which we will use in defining epiplexity. The principle states that among models for the data, the best explanation minimizes the total description length of the data, including both the description of the data using the model and the description of the model itself (Rissanen, 2004). The most common instantiation of this idea is via the statistical two-part code MDL.

Definition 6 (Two-part MDL (Rissanen, 2004; Grünwald, 2007)) *Let $x \in \{0,1\}^{n \times d}$ be the data and $\mathcal{H}$ be a set of candidate models. The two-part MDL is:*

$$L(x) = \min_{H \in \mathcal{H}} L(H) - \log P(x \mid H),$$

where $L(H)$ specifies the number of bits required to encode the model H, and $-\log P(x \mid H)$ is the number of bits required to encode the data given the model.

This formulation provides an intuitive implementation of Occam's Razor: complex models (large $L(H)$) are penalized unless they provide a reduction in the data's description length (large $P(x \mid H)$). If there are repeating patterns in the data, they can be stored in the model H rather than being duplicated in the code for the data. We review the modern developments of MDL in Appendix H. While MDL is a criterion for model selection given a fixed dataset, epiplexity, which we introduce next, can be viewed as its dual: a criterion for data selection given a fixed computation budget.

3 Epiplexity: Structural Information Extractable by a Computationally Bounded Observer

Keeping in mind the distinction between structural and random information in the unbounded compute setting, and the computational nature of pseudorandomness in cryptography, we now introduce epiplexity. *Epiplexity* captures the structural information present to a computationally bounded observer. As the computational constraints of this observer change, so too does the division between random and structured content. After introducing epiplexity here, we present ways of measuring epiplexity in Section 4. In Sections 5 and 6 we show how epiplexity can shed light on seeming paradoxes in information theory around the value of data, and OOD generalization.

First we will define what it means for a probability distribution to have an efficient implementation, requiring that it be implemented on a prefix-free universal Turing machine (UTM) and halt in a fixed number of steps.

Definition 7 (Time-bounded probabilistic model) *Let $T : \mathbb{N} \to \mathbb{N}$ be non-decreasing time-constructible function and let $\mathcal{U}$ be a fixed prefix-free universal Turing machine. A (prefix-free) program P is a T-time probabilistic model over $\{0,1\}^n$ if it supports both sampling and probability evaluation in time $T(n)$:*

Evaluation. *On input $(0, x)$ with $x \in \{0,1\}^n$, $\mathcal{U}(\mathrm{P}, (0, x))$ halts within $T(n)$ steps and outputs an element in $[0,1]$ (with a finite binary expansion), denoted*

$$\mathrm{Prob}_{\mathrm{P}}(x) := \mathcal{U}(\mathrm{P}, (0, x)).$$

Sampling. *On input $(1, u)$ where $u \in \{0,1\}^\infty$ is an infinite random tape, $\mathcal{U}(\mathrm{P}, (1, u))$ halts within $T(n)$ steps and outputs an element of $\{0,1\}^n$, denoted*

$$\mathrm{Sample}_{\mathrm{P}}(u) := \mathcal{U}(\mathrm{P}, (1, u)).$$

These outputs must define a normalized distribution matching the sampler:

$$\sum_{x \in \{0,1\}^n} \mathrm{Prob}_{\mathrm{P}}(x) = 1 \quad and \quad \Pr_{u \sim U_\infty} [\mathrm{Sample}_{\mathrm{P}}(u) = x] = \mathrm{Prob}_{\mathrm{P}}(x) \quad \forall x \in \{0,1\}^n.$$

Let $\mathcal{P}_T$ be the set of all such programs. To simplify the notation, we will use italicized P to denote the probability mass function $\mathrm{Prob}_{\mathrm{P}}$ in contrast with the non-italicized P, which denotes the program.

Here, n denotes the dimension of the underlying sample space (e.g., the length of the binary string.) This definition allows us to constrain the amount of computation the function class can use. Such a model class enforces that the functions of interest are both efficiently sampleable and evaluable, which include most sequence models. While in this work we focus primarily on computational constraints which we consider most fundamental, other constraints such as memory or within a given function class $\mathcal{F}$ can be accommodated by replacing $\mathcal{P}_T$ with $\mathcal{P}_\mathcal{F}$, and may be important for understanding particular phenomena.[3] With these preliminaries in place, we can now separate the random and structural components of information.

We define epiplexity and time-bounded entropy in terms of the program which achieves the best expected compression of the random variable X, minimizing the two-part code length (model and data given model bits) under the given runtime constraint.

---

3. One such possibility is to constrain the function class to all models reachable by a given optimization procedure with a given neural network architecture.

Definition 8 (Epiplexity and Time-Bounded Entropy) *Consider a random variable X on $\{0,1\}^n$. Let*

$$\mathrm{P}^\star = \arg\min_{\mathrm{P}\in\mathcal{P}_T}\{|\mathrm{P}|+\mathbb{E}[\log 1/P(X)]\} \tag{3}$$

be the program that minimizes the time bounded MDL with ties broken by the smallest program, and expectations taken over X. $|\mathrm{P}|$ denotes the length of the program P in bits, and logarithms are in base 2. We define the T-bounded epiplexity S_T and entropy H_T of the random variable X as

$$\mathrm{S}_T(X) := |\mathrm{P}^\star|, \quad and \quad \mathrm{H}_T(X) := \mathbb{E}[\log 1/P^\star(X)]. \tag{4}$$

The time-bounded entropy H_T captures the amount of information in the random variable that is random and unpredictable, whereas the epiplexity S_T captures the amount of structure and regularity visible within the object at the given level of compute. Uniform random variables have trivial epiplexity because a model as simple as the uniform distribution achieves a small two part code length, despite having large time bounded entropy. Explicitly, for a uniform random variable U_n on $\{0,1\}^n$, and even a constant time bound $T(n) \geq c_1$, $\mathrm{S}_T(U_n)+\mathrm{H}_T(U_n) \leq n+c_2$ where c_2 is the length of a program for the uniform distribution running in time c_1, and since $\mathrm{H}_T(U_n) \geq \mathrm{H}(U_n) = n$, it must be that $\mathrm{S}_T(U_n) \leq c_2$. Random variables with very simple patterns, like 0101010101... with probability 1/2 and 1010101010... with probability 1/2, also have low epiplexity because the time bounded MDL minimal model is simple. In this case with linear time $T(n) = \Theta(n)$, both $\mathrm{S}_T(X) = O(1)$ and $\mathrm{H}_T(X) = O(1)$. Henceforth, we will abbreviate $\mathrm{MDL}_T(X) := \mathrm{S}_T(X) + \mathrm{H}_T(X)$, which is the total time-bounded information content. We will now enumerate a few basic consequences of these definitions.

Basic Properties

(1) $\mathrm{S}_T(X) \geq 0, \quad \mathrm{H}_T(X) \geq 0,$

(2) $\mathrm{H}(X) \leq \mathrm{S}_T(X) + \mathrm{H}_T(X) \leq n + c_1,$

(3) $\mathrm{MDL}_{T'}(X) \leq \mathrm{MDL}_T(X) \quad$ whenever $\; T'(n) \geq T(n),$

(4) $\mathrm{MDL}_{T'}(f^{-1}(X)) \leq \mathrm{MDL}_T(X) + |\mathsf{f}|+c_2, \text{with } T'(n) = T(n) + \mathsf{Time(f)}.$

Statement 4 (defined for programs f that run in a fixed time implementing a bijection) is an analog of the information non-increase property $K(f(x)) \leq K(x) + K(f) + c$. However, note that while the Kolmogorov complexity for $K(f)$ and $K(f^{-1})$ are the same to within an additive constant, in our setting of a fixed computational budget having a short program for f^{-1} does not imply one for f, and vice versa. This gap between a function and its inverse has important consequences for the three paradoxes as we will see in Section 5.

Pseudorandom number sequences have high random content and little structure. Unlike Shannon entropy, Kolmogorov complexity, or even resource bounded forms of Kolmogorov complexity (Allender et al., 2011), we show that CSPRNGs have nearly maximal time-bounded entropy for polynomial time observers. Additionally, while CSPRNGs produce random content, they do not produce structured content as the epiplexity is negligibly larger than constant. Formally, let U_k be the uniform distribution on k bits.

Theorem 9 *For any $T \in \mathrm{Poly}(n)$ and $G \in \mathrm{CSPRNG}$ that stretches the input to $n = \mathrm{poly}(k)$ bits and allowing for an advantage of at most $\varepsilon(k)$, the time bounded entropy is nearly maximal:*

$$n - 2 - n\varepsilon(k) < \mathrm{H}_T(G(U_k)) \leq n + c,$$

and the epiplexity is nearly constant

$$\mathrm{S}_T(G(U_k)) \leq c + n\varepsilon(k).$$

Proof: see Appendix A.1.

In contrast, the Shannon entropy is $\mathrm{H}(G(U_k)) = k$, polynomial time bounded Kolmogorov complexity will be at most $k + c$ (assuming n is fixed or specified ahead of time) as there is a short and efficiently runnable program G which produces the output, and similarly with other notions such as Levin complexity (Li and Vitányi, 2008) or time bounded Kolmogorov complexity (Allender et al., 2011). Taken together, these results show that epiplexity appropriately characterizes pseudorandom numbers as carrying a large amount of time-bounded randomness but essentially no learnable structure, exactly as intuition suggests.

Existence of Random Variables with High Epiplexity. One may wonder whether any high epiplexity random variables exist at all, and indeed under the existence of one way functions we can show via a counting argument that there exists a sequence of random variables whose epiplexity grows at least logarithmically with the dimension.

Theorem 10 *Assuming the existence of one-way functions secure against non-uniform probabilistic polynomial time adversaries, there exists a sequence of random variables $\{X_n\}_{n=1}^{\infty}$ over $\{0,1\}^n$ such that*

$$\mathrm{S}_{\mathrm{Poly}}(X_n) = \Omega(\log n).$$

Proof: see Appendix A.4.

From this result we know at least that random variables with arbitrarily large epiplexities indeed exist; however, logarithmically growing information content only admits a very modest amount of information, still far from the power law scaling we see with some natural data.

Conditional Entropy and Epiplexity. To describe situations like image classification, where we are only interested in a function which predicts the label from the image, and not the information in generating the images, we define *conditional* time-bounded entropy and epiplexity.

Definition 11 (Conditional epiplexity and time-bounded entropy) *For a pair of random variables X and Y, define $\mathcal{P}_{T(n)}^X$ as the set of probabilistic models P such that for each fixed x, the conditional model $\mathrm{P}_{Y|x}$ is in $\mathcal{P}_{T(n)}$. The optimal conditional model with access to X is:*

$$\mathrm{P}_{Y|X}^{\star} = \arg\min_{\mathrm{P} \in \mathcal{P}_T^X} \left\{ |\mathrm{P}| + \mathbb{E}_{(X,Y)}\left[-\log P(Y \mid X)\right] \right\}. \tag{5}$$

The conditional epiplexity *and* time-bounded entropy *are defined as:*

$$\mathrm{S}_T(Y \mid X) := \left|\mathrm{P}_{Y|X}^{\star}\right|, \quad \mathrm{H}_T(Y \mid X) := \mathbb{E}_{(X,Y)}\left[-\log P_{Y|X}^{\star}(y \mid x)\right]. \tag{6}$$

11

These quantities are defined with respect to the time bounded MDL over programs which take as input X, Y and output the probabilities over Y (conditioned on X), and with expectations taken over both X and Y. We note that in general this definition is not equivalent to the difference of the joint and individual entropies, $\mathrm{H}_T(Y, X) - \mathrm{H}_T(X) \neq \mathrm{H}_T(Y|X)$. Unlike Shannon entropy, we can also condition on deterministic strings, which will change the values on account of not needing such a large program P. For example, we may be interested in the conditional epiplexity $\mathrm{S}_T(X|m)$ or entropy $\mathrm{H}_T(X|m)$ given a model m. For a deterministic string $d \in \{0,1\}^*$ we define the conditional epiplexity via

$$\mathrm{P}^{\star}_{Y|d} = \min_{\mathrm{P} \in \mathcal{P}_T^{\{0,1\}^*}} \left\{ |\mathrm{P}| + \mathbb{E}_Y \left[-\log P(Y \mid d) \right] \right\}, \tag{7}$$

where the minimization is over time bounded functions $P(\cdot \mid \cdot)$ that take in the string d as the second argument (which we refer to as $\mathcal{P}_T^{\{0,1\}^*}$).

For the machine learning setting, we take the random variable X to refer to the *entire dataset* of interest, i.e. typically a collection $X = [X_1, X_2, \dots]$ of many iid samples from a given distribution, rather than a lone sample from, and $\mathbb{E}[\log 1/P(X)]$ scales with the dataset size. Epiplexity typically grows with the size of the dataset (see detailed arguments for why this is the case in Section B.4) as larger datasets allow identifying and extracting more intricate structure and patterns, mirroring the practice of ML training. Moreover, as we will see later, the epiplexity of a typical dataset is orders of magnitudes smaller than the random information content. While not a focus of this paper, conditioning on deterministic strings opens up the possibility to understand what additional data is most useful for a specific machine learning model, such as for post-training a pre-trained LLM.

4 Measuring Epiplexity and Time-Bounded Entropy

We have now introduced epiplexity and time-bounded entropy as measures of structural and random information of the data. In this section, we present practical procedures to estimate upper bounds and empirical proxies for these quantities. Intuitively, we want to find a probabilistic model $P(\cdot)$ of the data X that achieves low expected loss $\mathbb{E}[\log 1/P(X)]$, is described by a short program P, and evaluating $P(X)$ takes time at most $T(|X|)$, which we will abbreviate as T. Using this model, we thereby decompose the information of the data into its structural and random components, namely, (1) epiplexity $\mathrm{S}_T(X)$: the length of the program $|\mathrm{P}|$, accounting for the bits required to model the data distribution, and (2) time-bounded entropy $\mathrm{H}_T(X)$: the expected length for entropy coding the data using this model, which accounts for the bits required to specify the particular realization of X within that distribution. We estimate conditional epiplexity analogously, providing random variable conditioning as input into the model.

Since directly searching over the space of programs is intractable, we restrict attention to probabilistic models parameterized by neural networks, as they achieve strong empirical compression across data modalities (MacKay, 2003; Goldblum et al., 2023; Delétang et al., 2023; Ballé et al., 2018) and capture the most relevant ML phenomenology. While a naive approach is to let P be a program that directly stores the architecture and weights of a neural network and evaluates it on the given data, this approach can significantly overestimate the information content in the weights, particularly for large models trained on relatively little data. Instead, we will use a more efficient approach that encodes the training process that produces the weights. We will discuss two approaches for encoding neural network training processes, based on *prequential coding* (Dawid, 1984) and *requential coding* (Finzi et al., 2026), respectively. The former is more straightforward to understand and evaluate, but relies on a heuristic argument to separate structure bits from noise bits, while the latter is rigorous at the cost of being more difficult to evaluate. Fortunately, both approaches often yield comparable rankings of epiplexity across datasets (Section 4.3).
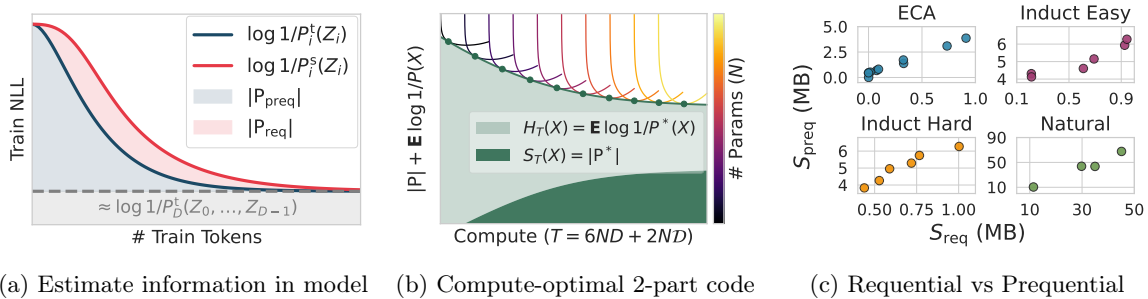
(a) Estimate information in model (b) Compute-optimal 2-part code (c) Requential vs Prequential

Figure 2: **How to estimate epiplexity.** (**a**) We consider two approaches for efficiently coding trained neural networks. Prequential estimation estimates information content as the area under the loss curve of a model above its final loss, with the training set matching the test data distribution. Requential coding, which provides an explicit code for P^s with expected length as the cumulative KL between a student model P^s and the teacher P^t that generates its *synthetic* training data, visualized approximately by their loss gap. We typically choose P^t to be a model trained on the *real* training set, as in prequential coding. (**b**) Using either approach, we optimize hyperparameters (model size N, training tokens D, etc.) to find the shortest two-part code for each compute budget, which decomposes into the estimated epiplexity and time-bounded entropy. (**c**) Comparing prequential and requential coding on four groups of datsets used in this work. The prequential estimate is typically larger, but the two correlate well, particularly within each group.

Moving forward, we will measure time by the number of floating-point operations (FLOPs) and dataset size by number of tokens, so that training a model with N parameters on D tokens takes time approximately $6ND$ (Kaplan et al., 2020), while evaluating it on X takes time $2N\mathcal{D}$ with $\mathcal{D} = |X|$ the number of tokens in X. To distinguish X from the training dataset, which we are free to choose, we will refer to X as the test dataset, as it is the data we need to perform inference on.

4.1 Approximating Model Description Length with Prequential Coding

Prequential coding provides a classic approach for compressing the training process of a neural network. We assume a batch size of one for simplicity, but generalizing to batch sizes larger than one is straightforward. Starting with a randomly initialized network P_0 (where the subscript indicates timestep), we proceed iteratively: at each step i, we entropy encode the current training token Z_i using $\log 1/P_i(Z_i)$ bits, then train the model on this token to produce P_{i+1}. Typically Z_i's are drawn i.i.d. from the same distribution as X. On the side of the decoder, a synchronized model is maintained; the model decodes Z_i using P_i and then trains on it to produce the identical P_{i+1}. Omitting small constant overheads for specifying the random initialization, architecture, and training algorithm, a total of $L(Z_{:M}, P_M) = \sum_{i=0}^{M-1} \log 1/P_i(Z_i)$ bits yields an explicit code for both the training data $Z_{:M} = \{Z_0, \ldots, Z_{M-1}\}$ and the final model weights P_M, which can be decoded in time $6ND$ for a model with N parameters trained on D tokens (typically $D > M$ as each example contains multiple tokens). Despite having an explicit code for Z, P_M, we cannot easily separate this into a code for P_M alone for estimating epiplexity.

To isolate the description length of P_M alone, we adopt the heuristic in Zhang et al. (2020) and Finzi et al. (2025): we first estimate the description length of the training data given P_M as its entropy code length under the final model, $L(Z_{:M}|P_M) = \sum_{i=0}^{M-1} \log 1/P_M(Z_i)$. Then, appealing to the symmetry of information, which states $K(P_M) = K(Z_{:M}, P_M) - K(Z_{:M}|P_M)$ up to constant terms, we estimate the description length of P_M as the difference $L(Z_{:M}, P_M) - L(Z_{:M}|P_M)$:

$$|\mathrm{P}_{\mathrm{preq}}| \approx \sum_{i=0}^{M-1} (\log 1/P_i(Z_i) - \log 1/P_M(Z_i)). \tag{8}$$

13

If Z_i is sampled i.i.d., as is typically the case, then the code length for the model *can be visualized as the area under the loss curve above the final loss* in Figure 2a. Intuitively, the model absorbs a significant amount of information from the data if training yields a sustained and substantial reduction in loss. For random data, $\log 1/P_i(Z_i)$ never decreases, while for simple data, $\log 1/P_i(Z_i)$ drops rapidly and stabilizes, both leading to small $|\mathrm{P_{preq}}|$.

Encoding the test dataset X (not to be confused with the training data) using this model, we obtain a two-part code of expected length $|\mathrm{P_{preq}}|+\mathbb{E}[\log 1/P_M(X)]$ that runs in time $6ND + 2N\mathcal{D}$. We optimize the training hyperparameters (e.g., learning rate) and the trade-off between N and D subject to the time bound $6ND + 2N\mathcal{D} \leq T$ to find the optimal $P^\star$ that minimizes the two-part code within this family, and estimate epiplexity and time-bounded entropy as $\mathrm{S}_T(X) = |\mathrm{P^\star_{preq}}|$ and $\mathrm{H}_T(X) = \mathbb{E}[\log 1/P^\star(X)]$. The better these hyperparameters are optimized, the more accurate our estimates become. We use the Maximal Update Parameterization (μP) (Yang et al., 2022) to ensure the optimal learning rate and initialization are consistent across model sizes, simplifying tuning. We estimate the expectation $\mathbb{E}[\log 1/P_M(X)]$ by its empirical value on held-out validation data, i.e., the validation loss scaled by the size of X. We detail the full procedure in Section B, such as how we choose the hyperparameters and estimate the Pareto frontier of MDL vs compute.

While conceptually simple, practically useful, and easy to evaluate, this prequential approach to approximating epiplexity is not rigorous for two reasons. First, both $L(Z_{:M}, P_M)$ and $L(Z_{:M}|P_M)$ can only upper-bound the respective Kolmogorov complexities, and thus their difference does not yield an upper bound for $K(P_M)$.[4] Second, even setting this issue aside, the argument only establishes the existence of a program that encodes P_M with length $|\mathrm{P_{preq}}|$, but does not guarantee that its runtime falls within $6ND$, since the symmetry of information does not extend to time-bounded Kolmogorov complexity. Nevertheless, prequential coding can serve as a useful starting point for crudely estimating epiplexity, particularly convenient when one already has access to the loss curve from an existing training run.

4.2 Explicitly Coding the Model with Requential Coding

To address the shortcomings of the previous approach based on prequential coding, we adopt requential coding (Finzi et al., 2026) for constructing an explicit code of the model with a known runtime. Rather than trying to code a particular training dataset, with requential coding one can use the insensitivity to the exact data points sampled to code for *a* sampled dataset that leads to a performant model but without paying for the entropy of the data. Specifically, it encodes a training run where at step i a student model P_i^{s} is trained on a synthetic token sampled randomly from a teacher model P_i^{t}, where the sequence $P_0^{\mathrm{t}}, \ldots, P_{M-1}^{\mathrm{t}}$ are arbitrary teacher model checkpoints. We typically choose P_i^{t} to be the checkpoints from training on the original *real* training set, as in prequential coding. Using relative entropy coding (Theis and Ahmed, 2022), the synthetic tokens $\widetilde{Z}_i \sim P_i^{\mathrm{t}}$ can be coded given only the student P_i^{s} (synchronized between encoder and decoder) using $\mathrm{KL}(P_i^{\mathrm{t}}\|P_i^{\mathrm{s}}) + \log(1 + \mathrm{KL}(P_i^{\mathrm{t}}\|P_i^{\mathrm{s}})) + 4$ bits in expectation. Summing over all steps gives the requential code length for P_M^{s}:

$$|\mathrm{P_{req}}| = \sum_{i=0}^{M-1} \mathrm{KL}(P_i^{\mathrm{t}}\|P_i^{\mathrm{s}}) + \log(1 + \mathrm{KL}(P_i^{\mathrm{t}}\|P_i^{\mathrm{s}})) + 4 + O(1) \approx \sum_{i=0}^{M-1} \mathrm{KL}(P_i^{\mathrm{t}}\|P_i^{\mathrm{s}}), \tag{9}$$

where the logarithmic and constant overheads are typically negligible due to large sequence length and batch size, and as before we omit the small constant cost of specifying the random initialization, architecture, and training algorithm. In addition to providing an explicit code, a key advantage of requential coding is its flexibility in choosing the teacher sequence: by selecting teachers P_i^{t} that

---

4. We have $L(Z_{:M}, P_M) + O(1) \geq K(Z_{:M}, P_M)$, but not that $L(Z_{:M}|P_M) + O(1) \leq K(Z_{:M}|P_M)$.

remain close to the student P_i^s while still pointing toward the target distribution, we keep the per-step coding cost $\mathrm{KL}(P_i^t \| P_i^s)$ small while effectively guiding the student's learning.

Figure 2a connects requential coding to the student's and teacher's loss curves: suppose we take as teachers the checkpoints $P_0^t, \ldots, P_{M-1}^t$ from a model trained on real data $Z_0, \ldots, Z_{M-2} \sim P_X$. For visualization, we can then estimate $\mathrm{KL}(P_i^t \| P_i^s)$ by the loss gap $\log 1/P_i^s(Z_i) - \log 1/P_i^t(Z_i)$, which is accurate when $P_i^t \approx P_X$. We can thus visualize the code length for the student as approximately the area between the teacher's and student's loss curves on real data, as shown in Figure 2a.

The two-part code has expected length $|\mathrm{P_{req}}| + \mathbb{E}[\log 1/P_M^s(X)]$, consisting of first decoding P_M^s by replaying the training process, which takes time $6ND$ for a total of D requential training tokens, and then evaluating P_M^s on the test dataset X, taking an additional time $2N\mathcal{D}$, for a total runtime of $6ND + 2N\mathcal{D}$. We optimize the training hyperparameters, teacher choices, and the trade-off between N and D subject to the specified time bound T to find the optimal model $P^\star$ minimizing the two-part code, and estimate $\mathrm{S}_T(X) = |\mathrm{P_{req}^\star}|$ and $\mathrm{H}_T(X) = \mathbb{E}[\log 1/P^\star(X)]$. See details in Section B.1.

4.3 Comparison Between the Two Approaches and Practical Recommendations

Figure 2c compares the estimated epiplexity obtained by the two approaches across four groups of datasets used in this work: ECA (Section 5.1), easy and hard induction (Section 5.3.1), and natural datasets (Section 6.2). While the prequential estimate is typically several times larger than the requential estimate, the two estimates correlate well, particularly within each group where the datasets yield similar learning dynamics. We detail the datasets and time bounds used in Section C.7. This general agreement is expected since the prequential estimate can be viewed as an approximation of requential coding with a static teacher (Section B.2). In general, however, the discrepancy between the two estimates will depend on particular datasets and training configurations, and a good correlation between the two is not guaranteed.

While requential coding is the more rigorous approach, it is typically $2\times$ to $10\times$ slower than prequential coding, which requires only standard training. The overhead depends on batch size, sequence length, and inference implementation (smaller overhead for large batches and short sequences), as requential coding requires repeatedly sampling from the teacher, though it is possible that the overhead can be reduced with more efficient algorithms. Therefore, we recommend using prequential coding for crudely estimating epiplexity and ranking the epiplexity of different datasets, particularly when one has access to the loss curve from an existing expensive training run (e.g., see an application in Section 6.2), and requential coding for obtaining the most accurate estimates otherwise.

4.4 How Epiplexity and Time-Bounded Entropy Scale with Compute and Data

Under natural assumptions about neural network training—namely, that larger models are more sample-efficient and that there are diminishing returns to scaling model size or data alone—we expect epiplexity and time-bounded entropy to exhibit certain generic scaling behavior as a function of the compute budget T and dataset size $\mathcal{D}$. In Section B.4, we show that, under these assumptions, the compute-optimal model size $N^\star(T)$ and training data size $D^\star(T)$ are generally increasing in the compute budget T, which implies that epiplexity $\mathrm{S}_T(X)$ typically grows with T while time-bounded entropy $\mathrm{H}_T(X)$ decreases. In the infinite-compute limit, epiplexity $\mathrm{S}_\infty(X)$ typically grows with the test set size $\mathcal{D} = |X|$, while the per-token time-bounded entropy $\mathrm{H}_\infty(X)/\mathcal{D}$ decreases. These results align with our intuition that larger compute budgets and more data allow the model to extract more structural information from the dataset and reduce the apparent randomness remaining in each sample. However, they should be understood only as typical trends, with a counterexample shown in Section 5.3.2 relating to the phenomenon of emergence.

5 Three Apparent Paradoxes of Information

To illustrate the lacunae in existing information theory perspectives, we highlight three *apparent paradoxes* of information: (1) information cannot be created by deterministic transformations; (2) total information content of an object is the same regardless of the factorization; and (3) likelihood modeling can only learn to match the data-generating process. Each statement captures some existing sentiment within the machine learning community, can be justified mathematically by Shannon and algorithmic information theory, and yet seems to be in conflict with intuitions and experimental observations. In this section, we will show with both theoretical results and empirical evidence that time bounding and epiplexity help resolve these apparent paradoxes.

5.1 Paradox 1: Information Cannot be Created by Deterministic Transformations

Both Shannon and algorithmic information theory state in some form that the total information cannot be increased by applying deterministic transformations on existing data. The data processing inequality (DPI) states that if some information source W produces natural data X that are collected, then no deterministic *or stochastic* transformations used to produce Y from X can increase the mutual information with the variable of interest W: $I(Y; W) \leq I(X; W)$. Similarly, information non-increase states that a deterministic transformation f can only preserve or decrease the Shannon information, a property that holds pointwise $-\log P_Y(f(x)) \leq -\log P_X(x)$ and in expectation: $\mathrm{H}(f(X)) \leq \mathrm{H}(X)$ (we note X here is a discrete random variable). In algorithmic information theory, there is a corresponding property: $K(f(x)) \leq K(x) + K(f) + c$ for a fixed constant c. These inequalities appear to rule out creating new information with deterministic computational processes.

How can we reconcile this fact with algorithms like AlphaZero (Silver et al., 2018) that can be run in a closed environment from a small deterministic program on the game of chess, extracting insights about the game, different openings, the relative values of pieces in different positions, tactics and high level strategy, and requiring megabytes of information stored in the weights? Similarly we have dynamical systems with simple descriptions of the underlying laws that produce rich and unexpected structures, from which we can learn new things about them and mathematics.

We also have evidence that synthetic data is helpful (Liu et al., 2024; Gerstgrasser et al., 2024; Maini et al., 2024; OpenAI, 2025) is helpful for model capabilities and moreover, if we believe that the processes that create natural data could in principle have been simulated to sufficient precision on a large computer, then all data could have been equivalently replaced with synthetic data. For practical synthetic data produced from transformations of samples from a given model and prompt, this sampling is performed with pseudorandom number generators, making the entire transformation deterministic. If we consider f as the transformations we use to produce synthetic data and x was the limited real data we started with, these inequalities appear to state very concretely that our synthetic data adds no additional information beyond the model and training data.

Whatever information it is that we mean when we say that AlphaZero has produced new and unexpected insights in chess, or new theoretical results in mathematics, or with synthetic data, it is not Shannon or algorithmic information. We argue that these unintuitive properties of information theory are a consequence of assuming unlimited computation for the observer. With limited computation, a description of the AlphaZero algorithm and the result of running AlphaZero for thousands of TPU hours are distinct. To build intuition, we start with the humble CSPRNG which also creates time-bounded information through computation (albeit random information).
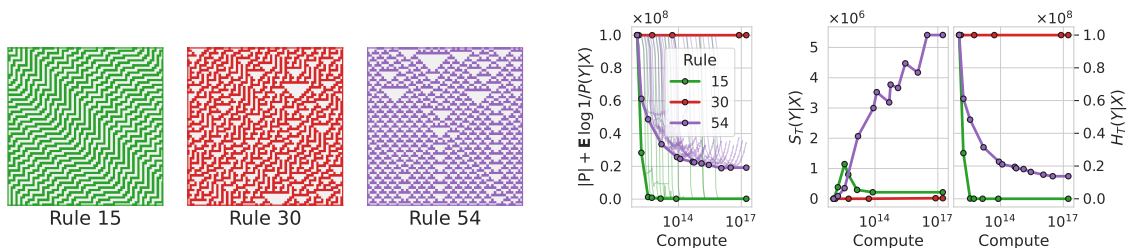
Figure 3: **Information created with cellular automata.** (**Left**) Example rollouts from random initial conditions of the class II rule 15, class III rule 30, and class IV rule 54. Time flows from up to down. (**Right**) Measuring epiplexity on data produced by these transformations, we see that rule 15 produces little information (low H_T, low S_T), rule 30 produces lots of unpredictable random information (high H_T, low S_T), and rule 54 produces both random and structural information (medium H_T, high S_T). These observations are reflected in the training loss curve of LLMs, which saturates quickly for rule 15, makes no progress for rule 30, and makes continued progress with compute for rule 54.

Theorem 12 *Let* $G : \{0,1\}^k \to \{0,1\}^n$ *be a* CSPRNG *which admits advantage* $\varepsilon(k)$ *and* U_k *be the uniform distribution.* $H_{\mathrm{Poly}}(G(U_k)) > H_{\mathrm{Poly}}(U_k) + n - n\varepsilon(k) - k - O(1)$.
Proof: see Appendix A.2.

Notably, we have a deterministic function which dramatically increases the time-bounded information content of the input. It is worth contrasting this result with Equation 3, where the time-bounded information content increase from a deterministic function *can* be bounded if the inverse function has a short program which can run efficiently. The statement highlights an important asymmetry between the function G and its inverse with fixed computation that does not hold with unlimited computation (e.g. $K(G^{-1}) = K(G) + O(1)$). Simultaneously, it provides some useful guidance for synthetic data: if we want to produce interesting information, we should make sure the functions we use do not have simple and efficiently computable inverses.

As an illustrative example, consider the iterated dynamics of elementary cellular automata (Wolfram and Gad-el Hak, 2003; Zhang et al., 2024). An elementary cellular automaton (ECA) is a one-dimensional array of binary cells that evolves in discrete time steps according to a *fixed* rule mapping each cell's current state and the states of its two immediate neighbors to its next state. Despite their simple formulation – only 256 possible rules—these systems can produce a rich variety of behaviors, from stable and periodic patterns to chaotic and computationally universal dynamics. We setup the problem of predicting $Y_i = F(X_i)$ from random initial data X_i for F being an ECA iterated 48 times on a grid of size 64, and assemble these pairs into a dataset $X = [X_1, \ldots, X_K]$ and $Y = [Y_1, \ldots, Y_K]$ for a total dataset of $\mathcal{D} = 100M$ tokens. We measure the conditional information content $Y|X$ (epiplexity and entropy) for ECA rules 15, 30, and 54 by training LLMs on this dataset. We provide a visualization of these dynamics in Figure 3 (left). For the class II rule 15 in the Wolfram hierarchy (Wolfram and Gad-el Hak, 2003), the produced behavior is periodic and has a simple inverse. Consequently, in Figure 3 (right), we see that training dynamics that rapidly converge to optimal predictions and with little epiplexity or time-bounded entropy. With the class III rule 30, the computation produces outputs that are inherently intractable to predict with limited computation, and as a result we see that there is maximal time-bounded entropy that is produced but no epiplexity. For the class IV rule 54, we see that the dynamics are complex but also partly understandable: the loss decreases slowly and much epiplexity is produced. These results highlight the sensitivity of epiplexity to the generating process. With the same compute spent and with a very similar program

17

we can have drastically different outcomes, producing simple objects, producing only random content, and producing a mix of random and structured content.

5.2 Paradox 2: Information Content is Independent of Factorization

An important property of Shannon's information is the symmetry of information, which states that the amount of information content does not change with factorization. The information we acquire when predicting x and then y is exactly equal to when predicting y and then x: $\mathrm{H}(Y \mid X) + \mathrm{H}(X) = \mathrm{H}(X, Y) = \mathrm{H}(X \mid Y) + \mathrm{H}(Y)$. An analogous property also holds for Kolmogorov complexity, known as the symmetry of information identity: $K(Y \mid X) + K(X) = K(X \mid Y) + K(Y) + O(1)$.

On the other hand, multiple works have observed that natural text is better compressed (with final model achieving higher likelihoods) when modeled in the left-to-right order (for English) than when modeled in reverse order (Papadopoulos et al., 2024; Bengio et al., 2019), picking out an *arrow of time* in LLMs where one direction of modeling is preferred over the other. It seems likely that for many documents, other orderings may lead to more information extracted by LLMs. Similarly, as we will show later, small rearrangements of the data can lead to substantially different losses and downstream performance. Cryptographic primitives like one way functions and block cyphers also provide examples where the order of conditioning can make all the difference to how entropic the data appears, for example considering autoregressive modeling of two prime numbers followed by their product vs the reverse ordering. These experimental results and cryptographic ideas indicate what can be learned is dependent on the ordering of the data, which in turn suggests that different amounts of "information" are extracted from these different orderings.

Our time-bounded definitions capture this discrepancy. Under the existence of one way permutations, we can prove that a gap in prediction exists over different factorizations for time bounded entropy.

> **Theorem 13** *Let f be a one-way permutation and let $X = U_n$ be uniform and $Y = f(X)$.*
> $\mathrm{H}_{\mathrm{Poly}}(X \mid Y) + \mathrm{H}_{\mathrm{Poly}}(Y) > \mathrm{H}_{\mathrm{Poly}}(Y \mid X) + \mathrm{H}_{\mathrm{Poly}}(X) + \omega(\log n)$.
> *Proof: see Appendix A.5.*

As a corollary, we show no polynomial time probability model which can fit a one way function's forward direction can satisfy Bayes theorem (see Theorem 26). Adding to these theoretical results, we look empirically at the gap in time-bounded entropy for one way functions, and the gap in both entropy and epiplexity over two orderings of predicting chess data.

In Figure 4(a), we choose f to be given by the 8 steps of evolution of the ECA rule 30 with state size n and periodic boundary conditions (Wolfram and Gad-el Hak, 2003). Though distinct from the one way functions used in cryptography, rule 30 is believed to be one way (Wolfram and Gad-el Hak, 2003) and unlike typical one way functions, the forward pass of rule 30 can be modeled by an autoregressive transformer, which we demonstrate by constructing an explicit RASP-L (Zhou et al., 2023; Weiss et al., 2021) program in Appendix D. As shown in Figure 4(a), the model achieves the Shannon entropy (gray) in the forward direction, but has a consistent gap in the reverse direction.

Beyond just how the random information can vary with orderings, the structural information can also differ as we will show next. We demonstrate this fact by training autoregressive transformer models on the Lichess dataset, a large collection of chess games where the moves are recorded in algebraic chess notation. We consider two variants of this dataset: (1) formatting each game as the move sequence followed by final board state in FEN notation, and (2) formatting each game as the final board state followed by the move sequence, as illustrated in Figure 4b. We provide full experiment details in Section C.4. While there is no clear polynomial vs non-polynomial time separation in this setup, the first ordering is analogous to the forward direction as the final board
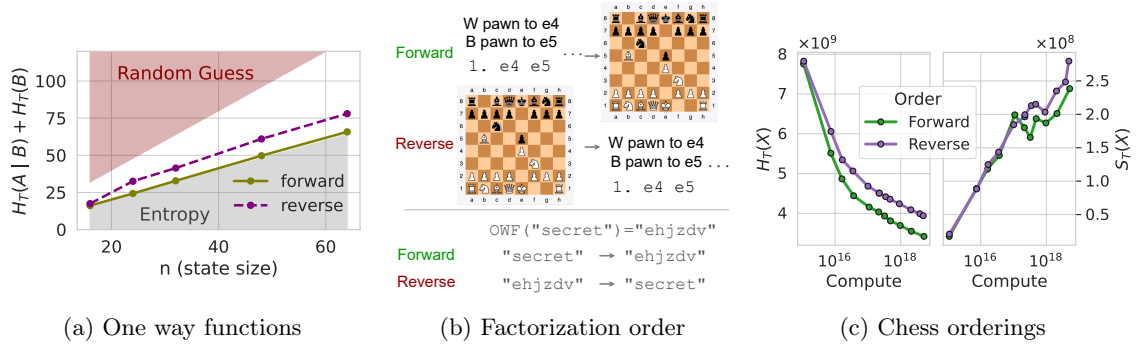
(a) One way functions (b) Factorization order (c) Chess orderings

Figure 4: **Factorization matters.** (**a**) We compare the losses from modeling a conjectured one way function in forward and reverse as the state size n is increased. The model reaches Shannon entropy in the forward direction, but with a persistent gap in the reverse direction. (**b**) The two orderings produce different outcomes. Analogous to the OWF, predicting the moves followed by the final board state is the direction that can be predicted with a straightfoward computation. Predicting the board first and then the moves requires more complex behaviors. (**c**) As compute increases, the same chess data presented in the reverse order leads to higher time-bounded entropy and epiplexity, showing it becomes more difficult to predict but allows more structure to be learned.

state can be straightforwardly mapped from the moves with a simple function, while the latter ordering is analogous to the reverse direction, where recovering the moves from the final board state requires the inverse function that infers the intermediate moves from the final state. We hypothesize the reverse direction is a more complex task and will lead the model to acquire more structural information, such as a deeper understanding of the board state. Figure 4c confirms this hypothesis, showing that the reverse order has both time-bounded higher entropy and epiplexity. This gap vanishes at small compute budgets where the model likely learns only surface statistics common to both orderings before the additional complexity of the reverse task forces it to develop richer board-state representations.

5.3 Paradox 3: Likelihood Modeling is Merely Distribution Matching

There is a prevailing view that from a particular training distribution, we can at best hope to match the data generating process. If there is a property or function that is not present in the data-generating process, then we should not expect to learn it in our models. As an extension, if the generating process is simple, then so are models that attempt to match it. This viewpoint can be supported by considering the likelihood maximization process abstractly, $\arg\min_P \mathbb{E}_{X \sim Q}[-\log P(X)] = Q$; the test NLL is minimized when the two distributions match. The extent to which the distributions differ is regarded as a failure either from too limited a function class or insufficient data for generalization. From these arguments we could reasonably believe that AI models cannot surpass human intelligence when pretraining on human data. Here we provide two classes of phenomena that seem to contradict this viewpoint: induction, and emergence. In both cases, restricting the compute available to AI models leads them to extract more structural information than what is required for implementing the generating process itself.

5.3.1 INDUCTION

The generative modeling community is often challenged with simultaneously wanting a tractable sampling process and tractable likelihood evaluation, with autoregressors, diffusion models, VAEs,

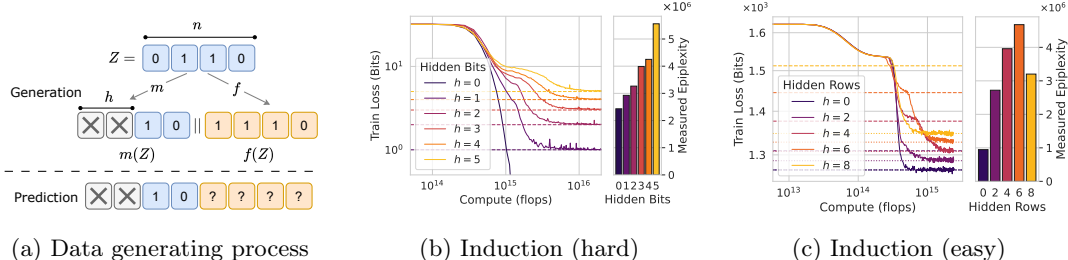(a) Data generating process   (b) Induction (hard)   (c) Induction (easy)

Figure 5: **Studying induction through epiplexity.** (a) Our setup for creating induction problems. (b) Predicting Rule 30 ECA with hidden inputs. The LLM must induct on the h bits missing from the input, paying a cost exponential in h. For h small enough but > 0, epiplexity is increased. (c) Predicting Markov chain samples with hidden transition probabilities. Models that need to both use the provided probabilities and induct on the missing ones acquire the most epiplexity.

GANs, and normalizing flows each providing different approaches. For natural generative processes, it is often the case that one direction may be much more straightforward than the other. Here we investigate generative processes which can be constructed by transforming latent variables such that computing likelihoods requires inducting on the values of those latents.

A window into the phenomenon can be appreciated through this quote from Ilya Sutskever:

> "*You're reading a murder mystery and at some point the text reveals the identity of the criminal. ... If the model can predict [the name] then it must have figured out [who perpetrated the murder from the evidence provided].*" (Sutskever, 2019)

The author of the book on the other hand, need not have made that same induction. Instead, they may have chosen the murderer first and then painted a compelling story of their actions. This example highlights a gap between the generating process and the requirements of a predictive model, a gap which we explore with the following more mathematical setup.

As we illustrate in Figure 5(a), consider a simple to model random variable Z over $\{0,1\}^n$ which we transform with two functions m and f, which are both short in length and efficient to compute, and produce the data $Y = (m(Z), f(Z))$. We choose $m : \{0,1\}^n \to \{0,1\}^{n-h}$ as a masking function which removes the bits at a total of h fixed locations in the input, leaving the rest unchanged. The generating process is simple to implement and can be executed efficiently. Now consider a likelihood generative model learning to model Y, under any given factorization. With appropriate properties of the function f, in producing the likelihoods the model must learn to induct on the missing information in the state Z, and then apply the transformation given by the data generating process. We consider cases both where the function f is hard to invert and those where f is not especially hard to invert. In both cases, predictive circuits must be learned that were not present in the data generating process, but with hard f these circuits only appear at exponentially high compute.

Induction Hard: Rule 30 ECA. For the first setting we use uniform $Z = U_n$ and f as 4 steps of the rule 30 ECA on state size $n = 32$, m simply removes the first h bits, and we also compute the loss only on $f(Z)$ (conditioned on $m(Z)$) as the bits in $m(Z)$ are uniform and only add noise. We use an LLM, and the loss curves and measured epiplexities are shown in Figure 5b. The loss converges to the number of hidden bits $-\log_2 P(f(Z) \mid m(Z)) = h$, representing the 2^h possible inductions on the hidden state. However, the total compute required for this loss to converge grows exponentially with h, an overall behavior consistent with a strategy of passing all 2^h candidates through f and then eliminating inconsistent candidates as values of $f(Z)_i$ are observed with the autoregressive factorization. This complex learned function stands in contrast with the mere $f(Z)$

20

and simple postprocessing removing bits with masking. This picture is mirrored by the measured epiplexity: as the model is forced to induct on the missing bits, the epiplexity grows.

Induction Easy: Random Markov Chains. In the second setting, we leverage the statistical induction heads setup (Edelman et al., 2024) with a few modifications. Z is given by a random Markov chain transition matrix with $V = 8$ symbols, and m removes h columns of the matrix at fixed random locations. The function $f(Z)$ computes a sampled sequence from the Markov chain of length $n = 512$. When $h > 0$, the optimal solution involves 1) using the provided rows Z to perfectly predict next-token probabilities on $V - h$ of the symbols, and 2) inducting on the missing rows of Z in-context based on the empirically observed transitions to improve remaining predictions. For $h = 0$, the first is sufficient, and for $h = 8$ the second is sufficient. In Figure 5c, we find evidence that both strategies are employed whenever $0 < h < 8$ as the final loss achieved matches the theoretical loss of both (the lower of the two dotted lines). The higher horizontal line marks the loss achievable using 1) along with a simple unigram strategy (Edelman et al., 2024), showing that the transformer learns 1) first and later the induction strategy 2). While the data generating program only only involves strategy one followed by the postprocessing masking step, the model must learn both strategies to reach these values. Measured epiplexity matches this picture, with values $0 < h < 8$ having higher epiplexity than $h = 0$ or $h = 8$. We emphasize that the induction strategy was never present in the data-generating process, yet it is learned by a generative model trained on that same data distribution. In Section G, we argue the induction phenomena are not specific to autoregressive models, but occur more generally for models trained via Maximum Likelihood Estimation as they need to be able to evaluate the likelihood $P(x)$ for an arbitrary data point x rather than merely sample random x from P. VAEs (Kingma et al., 2013) provide a clear example of explicitly performing induction in non-autoregressive models: the encoder is trained specifically to approximate the posterior $P_{Z|X}$, enabling tractable likelihood estimation, yet this encoder is entirely unnecessary if the goal is merely to sample from the model.

In both of the hard and easy induction examples, the size of the program needed to perform the induction strategy is greater than the size of the program needed generate the data. We can expect that with limited computational constraints, it will not be generically possible to invert the generation process using brute force, and thus, in cases where alternative inverse strategies exist (like the easy induction example with the statistical induction heads), those additional strategies increase the epiplexity. Given that there is likely no single generally applicable strategy for these computationally efficient inverses across problems, it is likely to be possible as a source of epiplexity.

To make these statements more precise, it seems likely that there are *no* constants c_1 and c_2 for which the following property holds:

> **Limited Epiplexity Increase Property:** Given any program G $: \{0,1\}^k \to \{0,1\}^n$ running in time at most T_1 on random variable Z, the epiplexity of $G(Z)$ is increased by at most a constant more than the size of G: $S_{T_2}(G(U_k)) \leq |G| + c_1$ for $T_2(n) > T_1(k) + c_2$.

In other words, there is no bound on how much larger the MDL optimal probability model will be than the generating program even when the model is allowed more compute than the generating program. We present this phenomenon in contrast to Shannon information or Kolmogorov complexity, where a function and its inverse can differ in complexity by at most a fixed constant: $K(F^{-1}) = K(F) + O(1)$. When the computational constraints are lifted, the brute force inverse is possible, and there is no essential gap between deduction and induction, or between sampling and likelihood computation.

One of the most striking counterexamples to the "distribution matching" viewpoint is *emergence*. Even when a system's underlying dynamics admit a simple description, an observer with limited computation may need to learn a richer, and seemingly unrelated, set of concepts to predict or explain its behavior. As articulated by Anderson (1972), reductionism—that a complex object's behavior follows from its parts—does not guarantee that knowing those parts lets us predict the whole. Across biology and physics, many-body interactions give rise to behaviors (e.g. bird flocking, Conway's Game of Life patterns, molecular chemistry, superconductivity) that are not apparent from the microscopic laws alone. Here we sketch how emergence critically relates to the computational constraints of the observer, demonstrating how observers predicting future states may be required to learn *more* than their unbounded counterparts who can execute the full generating process.

Consider Type-Ib emergence in the Carroll and Parola (2024) classification, in which higher-level patterns arise from local rules yet resist prediction from those rules. A canonical example is Conway's Game of Life (see Appendix E for definition), where iterating a simple computational rule Φ on a 2D grid leads to complex emergent behavior. For observers that lack the computational resources to directly compute the iterated evolution Φ^k, an alternate description must be found. In the state evolution, one can identify localized "species" (static blocks, oscillators, gliders, guns) which propagate through space and time. By classifying these species, learning their velocities, and how they are altered under collisions with other species, as well as the ability to identify their presence in the initial state, computationally more limited observers can make predictions about the future state of the system. Doing so, however, requires a more complex program in the sense of description length, and the epiplexity will be higher. We can formalize this intuition into the following definition of emergence.

> **Definition 14 (Epiplexity Emergent)** *Let $\{\Phi_n\}_{n \geq 1}$ be a computable family $\Phi_n : \{0,1\}^n \to \{0,1\}^n$ and let $\{X_n\}_{n \geq 1}$ be random variables over $\{0,1\}^n$. We say (Φ, X) is epiplexity-emergent if there exist time bounds T_1, T_2 with $T_1(n) = o(T_2(n))$ and an iteration schedule $k(n)$ such that as $n \to \infty$,*
>
> $$S_{T_1}(\Phi(X) \mid X, n) - S_{T_2}(\Phi(X) \mid X, n) = \Theta(1), \tag{10}$$
> $$S_{T_1}(\Phi^k(X) \mid X, n, k) - S_{T_2}(\Phi^k(X) \mid X, n, k) = \omega(1),$$
>
> *where we have suppressed the dependence of X_n and Φ_n on n for clarity.*

In words, Φ, X displays emergent phenomena if two observers see equivalent structural complexity in the one step map, but asymptotically more structural complexity in the multistep map for the observer with fewer computational resources.

Considering Φ from the Game of Life as an example, $P(\Phi(X) \mid X, n)$ could be well estimated by both T_1 and T_2-bounded observers using the exact time evolution rule, using constant bits for both. $P(\Phi^k(X) \mid X, n, k)$ could be estimated by T_2 using the iterated rule, but not by T_1. Using knowledge of the different pattern species improves predictions of $\Phi^k(X) \mid X$, so they would need to be learned; however, the number of patterns that needs to be considered in the time-bounded optimal solution is unbounded, and grows with the size of the board n, and thus the gap in epiplexity for the two time bounds grows with n. We have not proven that the Game of Life satisfies this definition, which is likely difficult as small changes to the evolution rule can destroy the emergent behavior; however, we provide empirical evidence for this set being non-empty with the example below.

In Figure 6, we empirically demonstrate the emergence phenomenon by training a transformer to predict the iterated dynamics of ECA rule 54, a class IV rule that produces complex patterns. As in Conway's Game of Life, a model with sufficient computation can exactly simulate the dynamics by directly iterating the per-step rule—a brute-force solution with a short description length. However, a compute-limited model cannot afford this approach and must instead learn emergent patterns (e.g., gliders and their collision rules) that approximately shortcut the infeasible exact simulation. The brute-force solution can be naturally implemented by learning to autoregressively unroll intermediate ECA states rather than directly predicting the final state, resembling the use of chain-of-thought (Wei et al., 2022) or looped transformers (Dehghani et al., 2018; Giannou
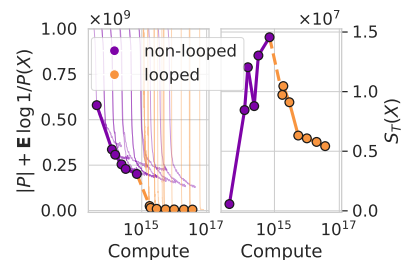

Figure 6: **Emergence in ECA.** Compute-constrained models extract high epiplexity from data generated by simple rules, trading increased program length for reduced computation.

et al., 2023; Saunshi et al., 2025). We provide experiment details in Section C.8. While initially the non-looped model (directly predicting final state) gradually achieves better MDL and higher epiplexity as compute increases, we a identify critical compute threshold beyond which the looped model suddenly becomes favorable, causing an abrupt drop in MDL and epiplexity, likely by learning the simple, brute-force solution. Below this threshold, the looped model underperforms likely because it lacks the compute to fully unroll the dynamics. The non-looped model, unable to rely on brute-force simulation, must instead learn increasingly sophisticated emergent rules, recognizing more species and their interactions, thus causing epiplexity to initially rise with compute before eventually falling.

While this experiment cleanly demonstrates how compute-limited models can learn richer structure from data, it is a rare example where the brute-force solution is experimentally accessible, and where training with more compute can therefore be counterproductive. In most practical settings, such as AlphaZero or modeling natural data, the corresponding simple, brute-force solutions are computationally infeasible, such as performing exhaustive search over an astronomical game tree and predicting natural data by simulating the laws of physics. Therefore, for most settings, models trained with more compute do continue to learn more structure, as the regime where brute-force becomes viable remains out of reach.

We explore other kinds of emergence, such as in chaotic dynamical systems or in the optimal strategies of game playing agents in Appendix F. Each of these examples presents clear evidence that in pursuit of the best probability distribution to explain the data, observers with limited compute will require models with greater description length than the minimal data generating process in order to achieve comparable predictive performance (Martínez et al., 2006; Redeker, 2010). Epiplexity provides a general tool for understanding and quantifying these phenomena of emergence, and how simple rules can create meaningful, complex structures that AI models can learn from, as recently demonstrated empirically by Zhang et al. (2024).

6 Epiplexity, Pre-Training, and OOD Generalization

Pre-training on internet-scale data has led to remarkable OOD generalization, yet a thorough understanding of this phenomenon remains elusive. What kinds of data provide the best signal for enabling broad generalization? Why does pre-training on text yield capabilities that transfer across domains while image data does not? As high-quality internet data becomes exhausted, what metric should guide the selection or synthesis of new pre-training data? In this section, we show how epiplexity helps answer these foundational questions.

OOD generalization is fundamentally about how much reusable structure the model acquires, not how well it predicts in-distribution. Two models trained on different corpora can achieve the same

in-distribution loss, yet differ dramatically in their ability to transfer to OOD tasks. This happens because loss captures only the residual unpredictability, corresponding to the time-bounded entropy, not how much reusable structure the model has internalized to achieve that loss. Epiplexity measures exactly this missing component: the amount of information in the learned program. Intuitively, loss indicates how random the data looks to the model, while epiplexity indicates how much structure the model must acquire to explain away the non-random part. If OOD generalization depends on reusing learned mechanisms rather than memorizing superficial statistics, then epiplexity is a natural lens through which to understand the relationship between pre-training data and OOD transfer. As a motivating toy example, Zhang et al. (2024) observed that type IV ECA rules tend to benefit downstream tasks, which correlates with Figure 3 where we showed that rule 54 (a type IV rule) induces much larger epiplexity compared to other types of ECAs.

6.1 Epiplexity Correlates with OOD Generalization in Chess

We finetune models trained on either ordering from Section 5.2 on two downstream tasks: (1) solving chess puzzles, where the model must predict the *optimal* next move given a board state (Burns et al., 2023), and (2) predicting centipawn evaluation, where the model evaluates positional advantage from FEN notation—a more substantial distribution shift from next-move prediction learned in pre-training. Experiment details are in Section C.4. As shown in Figure 7, the reverse (board-then-moves) ordering yields higher epiplexity and better downstream performance: matching accuracy on chess puzzles but significantly higher accuracy on the centipawn task. This result supports our hypothesis: the reverse order forces the model to develop richer board-state representations needed to infer the intermediate moves, and these representations transfer to OOD tasks like centipawn evaluation that similarly require understanding

Figure 7: **Epiplexity and OOD performance in chess.** Models trained on the higher epiplexity reverse order performs better in OOD tasks.

the board state. This example reflects a more general principle: epiplexity measures the learnable structural information a model extracts from data to its weights, which is precisely the source of the information transferable to novel tasks, making epiplexity a plausible indicator for the potential of OOD generalization. However, we emphasize that higher epiplexity does not guarantee better generalization to any specific task: epiplexity measures the amount of structural information, irrespective of its content. A model trained on high epiplexity data can learn a lot of structures, but these structures may or may not be relevant to the particular downstream task of interest.
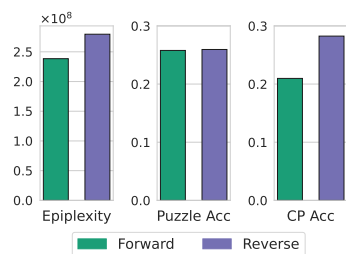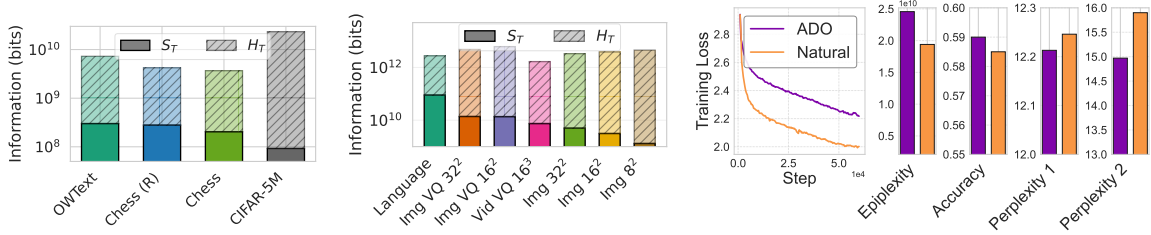
6.2 Measuring Structural Information in Natural Data

Among different modalities of natural data, language has proven uniquely fruitful for pre-training, not only for improving in-distribution performance such as language understanding (Radford et al., 2019), but also for out-of-distribution tasks such as robotics control (Ahn et al., 2022), formal theorem proving (Song et al., 2024), and time-series forecasting (Gruver et al., 2023). While equally abundant information (as measured by raw bytes) is available in other modalities, such as images and videos, pre-training on those data sources typically does not confer a similarly broad increase in capabilities. We now show that epiplexity helps explain this asymmetry by revealing differences in structural information across datasets. In Figure 8a, we show the estimated decomposition of the information in 5B tokens of data from OpenWebText, Lichess, and CIFAR-5M (Nakkiran et al., 2020) into epiplexity (structural) and time-bounded entropy (random) with a time-bound of 6×10^{18} FLOPs, by training models of up to 160M parameters on at most 5B tokens using requential coding. In all cases, epiplexity accounts for only a tiny fraction of the total information, with the OpenWebText data carrying the

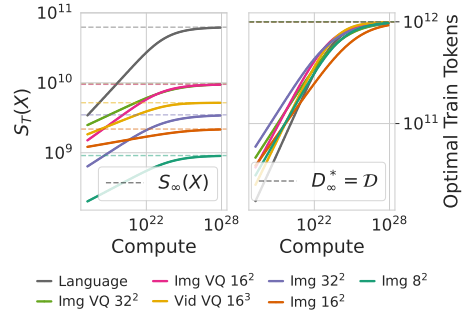(a) Epiplexity in natural data (b) Estimation via scaling laws (c) ADO: epiplexity and downstream metrics

Figure 8: **Epiplexity reveals differences in the structural information across data modalities and can guide pre-training data selection.** (**a**) Estimated epiplexity and time-bounded entropy using requential coding for 1B OpenWebText, Chess, and CIFAR-5M tokens at 6×10^{18} FLOPs. (**b**) Estimated values based on scaling laws and prequential coding for 1T language, image, and video tokens at 10^{25} FLOPs. (**c**) Selecting pre-training data using ADO (Jiang et al., 2025) leads to different loss curves than standard sampling (natural). Our measurement shows ADO selects data with higher epiplexity, in line with the improved downstream performance and OOD perplexity on different text corpora.

most epiplexity, followed by chess data. Despite carrying the most total information, CIFAR-5M data has the least epiplexity, as over 99% of its information is random (e.g., unpredictability of the exact pixels).

6.3 Estimating Epiplexity from Scaling Laws

We can estimate the epiplexities of larger datasets at higher compute budgets using reported scaling laws, which describe the loss achieved by an N-parameter model trained on D tokens as $\mathcal{L}(N, D) = E + (N/N_0)^{-\alpha} + (D/D_0)^{-\beta}$, for some dataset-specific constants $\alpha, \beta, N_0, D_0, E$ (Hoffmann et al., 2022; Kaplan et al., 2020; Henighan et al., 2020). By estimating the model's description length via the prequential coding approach (Section 4.3), we obtain estimates for the epiplexity and time-bounded entropy for language, image, and video datasets, with varying resolutions and tokenizations of size $\mathcal{D} = 10^{12}$ (1T) tokens under a compute budget of 10^{25} FLOPs (equivalent to the training compute of Llama3 70B), illustrated in Figure 8b (see details in Section C.9). Consistent with our smaller-scale experiments, we find that language data has the highest

Figure 9: **Epiplexity and optimal training tokens for each fixed dataset converge to predictable limits as compute increases.**

epiplexity, while image data has the least. For image data, applying VQ tokenization leads to a significant increase in epiplexity, likely as a result of allowing the model to focus on higher-level semantic structures. Video data has less time-bounded entropy and epiplexity than image data with the same resolution, likely due to significant redundancy across the temporal dimension.

Using this approach, we can also gain some analytical insights about epiplexity for data admitting scaling laws of this form. As we derive in Section B.3, for a fixed dataset X with $\mathcal{D}$ tokens, the optimal split of the compute budget between training and inference (evaluating the trained model on X) approaches a fixed ratio as compute increases, with the optimal asymptotic training tokens $D_\infty^\star = \mathcal{D}$ and asymptotic epiplexity $\mathrm{S}_\infty(X) = \frac{\beta}{1-\beta} D_0^\beta \mathcal{D}^{1-\beta}$, both illustrated in Figure 9. As expected, the maximum amount of extractable structural information is ultimately capped by the dataset size $\mathcal{D}$ when compute is not the bottleneck, and epiplexity can increase further if we also grow the dataset

size. For large $\mathcal{D}$, the scale of the asymptotic epiplexity is primarily determined by β and D_0, with smaller β and larger D_0 leading to higher epiplexity, corresponding to slower improvement in loss and thus more (estimated) information absorbed per token. In line with our discussion on emergence in Section 5.3.2, it is possible that with significantly more compute much simpler programs can model these natural datasets, such as by directly simulating the basic laws of physics from which the natural world emerges, but the amount of required computation is likely so high that such programs remain inaccessible to any physically realizable observer and we must treat natural data as having high epiplexity for all practical purposes.

6.4 Pre-Training Data Selection and Curriculum for Language Models

A crucial step in pretraining a language model is designing the composition of the pretraining data, but there lack clear guidelines for this step. Existing data mixtures are designed through extensive trial-and-error and rely on heuristic guidelines such as "diversity" or "high-quality". More importantly, the primary way of comparing different training data is via perplexity metrics of held-out datasets and downstream performance. These procedures are highly susceptible to data contamination, overfitting to a narrow set of downstream evaluations, and Goodhart's law. After all, no suite of downstream evaluations is extensive enough to faithfully capture the range of tasks that a general-purpose language model will encounter in the real world.

As we argued above, epiplexity measures the structural information learned by the model, which could be affected by data selection strategies. Jiang et al. (2025) demonstrated that models of the loss curves for different data subsets can be used to dynamically adjust the data distribution online to favor data subsets whose training losses are *decreasing faster*. Intuitively, this objective coincides with increasing the prequential approximation of epiplexity described in subsection 4.3 as it leads to a higher area under the loss curve. We hypothesize that the proposed algorithm, Adaptive Data Optimization (ADO), inadvertently achieves higher epiplexity. Experiments of Jiang et al. (2025) are conducted on decoder-only transformers with 1.3B parameters trained on 125B tokens from the Pile dataset (Gao et al., 2020). The models are evaluated on a suite of 7 zero-shot downstream tasks and two OOD validation datasets, SlimPajama (Soboleva et al., 2023) and FineWeb (Penedo et al., 2024).

In Figure 8c(c), we show the estimated epiplexity and the downstream performance as well as perplexity on two OOD datasets, adapted from Jiang et al. (2025). As shown in Jiang et al. (2025), ADO achieves higher downstream performance than a standard data sampling strategy that uniformly samples from the entire dataset (denoted by *Natural* in Figure 8c), despite not being optimized for any of these metrics. Interestingly, we see that ADO indeed achieves higher epiplexity measured by prequential coding. While these downstream evaluations do not capture everything about a pretrained model, they do offer evidence that epiplexity is a potentially useful concept for understanding the intrinsic value of pretraining data without particular downstream evaluations.

7 Additional Related Work

Epiplexity builds on a number of related ideas in algorithmic information theory and complexity science that attempt to theoretically characterize *meaningful information*. A group of closely related concepts are sophistication (subsection 2.2), effective complexity, and logical depth. Similar to sophistication, effective complexity aims to separate random from structural content (Gell-Mann and Lloyd, 1996). From a different starting point, Bennett (1988) introduced logical depth, measuring the number of time steps required by a nearly optimal program to produce a given string, and which was later shown to be equivalent to sophistication through the busy beaver function (Antunes et al., 2005; Ay et al., 2010). Several other formal measures have been developed to quantify structured

or meaningful complexity. Algorithmic statistics offers a principled decomposition of data into regular versus random components by introducing the notion of an algorithmic sufficient statistic (Vereshchagin and Vitányi, 2004), a concept closely tied to sophistication. Relatedly, statistical complexity in computational mechanics (Shalizi and Crutchfield, 2001) measures the entropy of causal states in an optimally predictive model, capturing structure in time-series data. As we argued above, these existing notions of complexity fail to account for the limited computation available to the observer, which is essential for understanding machine learning algorithms. Being oblivious to computational limits means that they cannot characterize CSPRNGs or encrypted objects as being random. One might think that these failures are surface-level; for example, a plausible strategy would be to upgrade sophistication by replacing Kolmogorov complexity with time-bounded Kolmogorov complexity in (Definition 5). However, this approach does not work for several reasons, the most obvious being that CSPRNG outputs do have short and efficiently runnable generating programs and thus their time-bounded Kolmogorov complexities are small. A more subtle reason is that doing so results in trivial sophistication for all strings, which we discuss in more detail in Appendix A.6.

Our work is also closely related to several lines of work trying to characterize observer-dependent notions of information. In cryptography, Barak et al. (2003) and Hsiao et al. (2007) discuss several possible definitions for *computational pseudoentropy*, an observer-dependent analogue of entropy. HILL-pseudoentropy (Håstad et al., 1999) is defined relative to a class of tests: a source is considered random if no test within the class can distinguish it from a high-entropy distribution with nontrivial advantage, and Yao-pseudoentropy is defined via compressing and decompressing an object for example. Both definitions are closely related to time-bounded entropy, which measures the random content to a given computationally bounded observer; however, our formulation directly maps on to machine learning practice and it allows for separating out the structural information content, a key contribution of our work. More recently, Xu et al. (2020) propose $\mathcal{V}$-entropy, a generalization of Shannon entropy to the minimum expected negative log probability over a given family of probability models, such as those with given computational constraints. With $\mathcal{V}$-entropy, the symmetry of information can be violated, and so too can the data processing inequality, though neither is explicitly proven in the paper. Unlike time-bounded entropy, the computational constraint in $\mathcal{V}$-entropy only limits the inference time, and does not account for the time to find such a model. Hence, the minimizer can be far away from the regime that is practically evaluated (such as models that are *trained* on infinite data or with infinite compute). While these undesirable behaviors can be overcome by imposing further data constraints, we believe our formulation of imposing a single bound on both training and inference time leads to fewer complications. More importantly, both pseudoentropy and $\mathcal{V}$-entropy, much like time-bounded entropy, capture only the random component of information since it still measures the unpredictability of the random variable under the best feasible model. For understanding what useful information a model has learned, we are more interested in the non-random component of information as measured by epiplexity. Using existing measures of data complexity, such as the Lempel-Ziv complexity and Wolfram classification, Zhang et al. (2024) showed that models trained on complex data like Class IV ECA rules tend to perform better on downstream tasks.

Several works have also explored how to quantify data complexity. Dziugaite and Roy (2025) suggests that the complexity of a minimal near-optimal reference model can be viewed as a measure of data complexity under the PAC-Bayes framework and how such data complexity gives rise to empirical scaling laws. This perspective is related to epiplexity in that both associate data complexity with the size of compact models that explain the data well. However, the two notions differ in important ways. In particular, the PAC-Bayes formulation is concerned with the existence of some small reference model achieving good in-distribution performance, whereas epiplexity characterizes the amount of structural information extractable by a computationally bounded observer, formalized through a two-part code that explicitly accounts for the cost of obtaining such a model. Further, our primary interest is not in characterizing in-distribution generalization, but in using epiplexity to measure the

intrinsic value of data in settings that extend beyond supervised learning. Relatedly, Hutter (2021) shows that power-law learning curves can emerge under specific assumptions on the data-generating distribution, illustrating how properties of the data itself can shape empirical scaling behavior. While this line of work focuses on explaining observed learning dynamics rather than defining a complexity measure, it similarly emphasizes the role of data structure in determining learning outcomes. These perspectives on data complexity can be viewed as instances of *coarse graining*, where one seeks a compressed representation that preserves some notion of "relevant" structure. A canonical example is the information bottleneck framework, which formalizes coarse graining as a trade-off between compression and retained information about a relevant variable (Tishby et al., 2000). Epiplexity is aligned with this perspective, but rather than defining relevance through a task variable or through distinguishability to tests, it measures the amount of structural information extractable by a computationally bounded learner, while explicitly accounting for the cost of obtaining the model.

More broadly, our work is related to several lines of work on how resource constraints fundamentally alter the notion of simplicity and learnability. In algorithmic information theory, Schmidhuber (2002) proposes the speed prior, which replaces Solomonoff's universal prior with a *computable* semimeasure that favors both shorter program length and smaller computation time, thereby incorporating computational resources directly into the definition of simplicity. In learning theory, a related line of work shows that computational limitations can directly affect what can be learned from data. For instance, in the problem of sparse PCA detection, Berthet and Rigollet (2013) show that although there exist procedures that succeed with an information-theoretically minimal number of samples, any algorithm that runs in polynomial time necessarily requires more data under widely used average-case hardness assumptions. Memory and space constraints alone can also qualitatively change learnability. Steinhardt et al. (2016) show that restricting a learner's memory can dramatically increase the amount of data required to learn, even when the target concept itself has a very concise description. They identify parity functions as a canonical example where this tension is conjectured to be sharp. Raz (2018) later resolves this conjecture by proving that any learner with sub-quadratic memory requires exponentially many samples to learn parity from random examples.

8 Discussion

Much of classical information theory is concerned with the representation and transmission of information, and abstracts away key aspects of the computational processes by which information is extracted and used. While complexity theory and cryptography treat computation as fundamental, machine learning theory typically does not. Yet learning, whether biological or artificial, is an inherently computational process. What can be learned from data depends not only on statistical feasibility, but on the available resources. This perspective calls for more theoretical tools that place computation on an equal footing with information.

This work reframes information as a property of data relative to a computationally-bounded observer, and demonstrates that information can be decomposed into time-bounded entropy and epiplexity, a formalization of structural information. It also sheds light on how perceived information can be changed through computation. This perspective resolves several tensions between information theory and empirical machine learning—including the usefulness of synthetic data, the dependence of learning on factorization and ordering, and the emergence of structure beyond the data-generating process itself. Technically, epiplexity connects ideas from algorithmic statistics, cryptography, and learning theory, showing that standard assumptions (i.e., existence of one-way functions) suffice to produce distributions with high structural complexity for efficient learners.

Our framework opens several exciting directions for future work. On the theoretical side, it invites a systematic and more fine-grained understanding of how structural information changes with

computational budget, model class, and data transformations, potentially yielding new lower bounds and impossibility results for representation learning and transfer. Taking information and computation as the fundamental resources may offer new explanations for the relative universality observed in large-scale training, including why scaling law exponents depend only weakly on architectural and optimizer details. There is also a possibility of a compute-aware analogue of classical notions such as sufficient statistics and information bottlenecks. More broadly, framing emergence, induction, and generalization through the lens of computationally bounded observers may offer a unifying language across learning theory, algorithmic information theory, cryptography, and complexity theory.

On the empirical side, epiplexity provides a way to reason about why some data sources, formatting, and transformations can lead to more transferable models than others, even when they do not improve training loss. The framework suggests that pretraining data should be evaluated not only by held-out perplexity, but by how much reusable structural information it induces in a computationally bounded model. This perspective helps explain empirical successes of curriculum design, data ordering, augmentation strategies, and even synthetic data that appear counterintuitive from a purely statistical viewpoint. Our empirical estimator offers a concrete starting point for comparing datasets and interventions in data centric research. In the long run, we believe epiplexity could provide guidance on how to generate new synthetic data from existing data.

Finally, representation learning can be understood as the gradual accumulation of epiplexity: the construction of increasingly rich internal programs that approximate a data distribution within a fixed time budget. While epiplexity in isolation is not a measure of generalization, or a complete theory of learning, this perspective raises the possibility of new notions of hardness for learning and transfer that are orthogonal to classical PAC-style measures, capturing not sample complexity but the size of the structure that must be extracted. Such notions may help explain why certain tasks appear to require disproportionately large models or long training horizons despite admitting simple generative descriptions, and why improvements in generalization sometimes correlate more strongly with training dynamics or data structure than with likelihood alone.

References

Scott Aaronson, Sean M Carroll, and Lauren Ouellette. Quantifying the rise and fall of complexity in closed systems: the coffee automaton. *arXiv preprint arXiv:1405.6903*, 2014.

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, et al. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*, 2024.

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

Eric Allender, Michal Koucký, Detlef Ronneburger, and Sambuddha Roy. The pervasive reach of resource-bounded kolmogorov complexity in computational complexity theory. *Journal of Computer and System Sciences*, 77(1):14–40, 2011.

Philip W Anderson. More is different: Broken symmetry and the nature of the hierarchical structure of science. *Science*, 177(4047):393–396, 1972.

Luis Antunes, Lance Fortnow, Dieter van Melkebeek, and N. V. Vinodchandran. Sophistication revisited. *Theory of Computing Systems*, 38(4):535–555, 2005.

Benny Applebaum. Cryptographic hardness of random local functions: Survey. *Computational complexity*, 25(3):667–722, 2016.

Nihat Ay, Markus Müller, and Arleta Szkola. Effective complexity and its relation to logical depth. *IEEE transactions on information theory*, 56(9):4593–4607, 2010.

Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. *arXiv preprint arXiv:1802.01436*, 2018.

Boaz Barak, Ronen Shaltiel, and Avi Wigderson. Computational analogues of entropy. In *International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 200–215. Springer, 2003.

Yoshua Bengio, Tristan Deleu, Nasim Rahaman, Rosemary Ke, Sébastien Lachapelle, Olexa Bilaniuk, Anirudh Goyal, and Christopher Pal. A meta-transfer objective for learning to disentangle causal mechanisms. *arXiv preprint arXiv:1901.10912*, 2019.

Charles H Bennett. Logical depth and physical complexity. *The Universal Turing Machine: A Half-Century Survey*, 1:227–257, 1988.

Quentin Berthet and Philippe Rigollet. Computational lower bounds for sparse pca. *arXiv preprint arXiv:1304.0828*, 2013.

Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo random bits. In *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, pages 112–117, 1982. doi: 10.1109/SFCS.1982.72.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/jax-ml/jax`.

Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, et al. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. *arXiv preprint arXiv:2312.09390*, 2023.

Sean M Carroll and Achyuth Parola. What emergence can possibly mean. *arXiv preprint arXiv:2410.15468*, 2024.

Gregory J Chaitin. Information-theoretic limitations of formal systems. *Journal of the ACM (JACM)*, 21(3):403–424, 1974.

Gregory J Chaitin. A theory of program size formally identical to information theory. *Journal of the ACM (JACM)*, 22(3):329–340, 1975.

Gregory J Chaitin. *The limits of mathematics: A course on information theory and the limits of formal reasoning.* Springer, 1998.

A Philip Dawid. Present position and potential developments: Some personal views statistical theory the prequential approach. *Journal of the Royal Statistical Society: Series A (General)*, 147(2): 278–290, 1984.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

Grégoire Delétang, Anian Ruoss, Paul-Ambroise Duquenne, Elliot Catt, Tim Genewein, Christopher Mattern, Jordi Grau-Moya, Li Kevin Wenliang, Matthew Aitchison, Laurent Orseau, et al. Language modeling is compression. *arXiv preprint arXiv:2309.10668*, 2023.

Nolan Dey, Bin Claire Zhang, Lorenzo Noci, Mufan Li, Blake Bordelon, Shane Bergsma, Cengiz Pehlevan, Boris Hanin, and Joel Hestness. Don't be lazy: Completep enables compute-efficient deep transformers. *arXiv preprint arXiv:2505.01618*, 2025.

Rod Downey and Denis R Hirschfeldt. Algorithmic randomness. *Communications of the ACM*, 62 (5):70–80, 2019.

Gintare Karolina Dziugaite and Daniel M Roy. The size of teachers as a measure of data complexity: Pac-bayes excess risk bounds and scaling laws. In *The 28th International Conference on Artificial Intelligence and Statistics*, 2025.

Benjamin L Edelman, Ezra Edelman, Surbhi Goel, Eran Malach, and Nikolaos Tsilivis. The evolution of statistical induction heads: In-context learning markov chains. *arXiv preprint arXiv:2402.11004*, 2024.

Marc Finzi, Sanyam Kapoor, Diego Granziol, Anming Gu, Christopher De Sa, J Zico Kolter, and Andrew Gordon Wilson. Compute-optimal llms provably generalize better with scale. *arXiv preprint arXiv:2504.15208*, 2025.

Marc Finzi, et, and al. Requential coding. Forthcoming, 2026.

Aviezri S Fraenkel and David Lichtenstein. Computing a perfect strategy for n× n chess requires time exponential in n. In *International Colloquium on Automata, Languages, and Programming*, pages 278–293. Springer, 1981.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Martin Gardner. Mathematical games. *Scientific american*, 222(6):132–140, 1970.

Murray Gell-Mann and Seth Lloyd. Information measures, effective complexity, and total information. *Complexity*, 2(1):44–52, 1996.

Matthias Gerstgrasser, Rylan Schaeffer, Apratim Dey, Rafael Rafailov, Henry Sleight, John Hughes, Tomasz Korbak, Rajashree Agrawal, Dhruv Pai, Andrey Gromov, et al. Is model collapse inevitable? breaking the curse of recursion by accumulating real and synthetic data. *arXiv preprint arXiv:2404.01413*, 2024.

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

Micah Goldblum, Marc Finzi, Keefer Rowan, and Andrew Gordon Wilson. The no free lunch theorem, kolmogorov complexity, and the role of inductive biases in machine learning. *arXiv preprint arXiv:2304.05366*, 2023.

Oded Goldreich. *Foundations of Cryptography: Volume 1, Basic Tools*. Cambridge University Press, 2006.

Peter D Grünwald. *The minimum description length principle*. MIT press, 2007.

Peter D Grünwald, PM Vitányi, et al. Algorithmic information theory, 2008.

Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew G Wilson. Large language models are zero-shot time series forecasters. *Advances in Neural Information Processing Systems*, 36:19622–19635, 2023.

Alex Hägele, Elie Bakouch, Atli Kosson, Leandro Von Werra, Martin Jaggi, et al. Scaling laws and compute-optimal training beyond fixed training durations. *Advances in Neural Information Processing Systems*, 37:76232–76264, 2024.

Johan Håstad, Russell Impagliazzo, Leonid A Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B Brown, Prafulla Dhariwal, Scott Gray, et al. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Chun-Yuan Hsiao, Chi-Jen Lu, and Leonid Reyzin. Conditional computational entropy, or toward separating pseudoentropy from compressibility. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 169–186. Springer, 2007.

Marcus Hutter. Learning curve theory. *arXiv preprint arXiv:2102.04074*, 2021.

Yiding Jiang, Allan Zhou, Zhili Feng, Sadhika Malladi, and J Zico Kolter. Adaptive data optimization: Dynamic sample selection with scaling laws. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=aqok1UX7Z1`.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Diederik P Kingma, Max Welling, et al. Auto-encoding variational bayes, 2013.

A. N. Kolmogorov. Three approaches to the quantitative definition of information *. *International Journal of Computer Mathematics*, 2(1-4):157–168, 1968. doi: 10.1080/00207166808803030. URL `https://doi.org/10.1080/00207166808803030`.

Moshe Koppel. Structure. In Rolf Herken, editor, *The Universal Turing Machine: A Half-Century Survey*, pages 435–452. Oxford University Press, 1988.

Leon G. Kraft. A device for quantizing, grouping, and coding amplitude-modulated pulses. S.m. thesis, Massachusetts Institute of Technology, Cambridge, MA, 1949. URL `https://hdl.handle.net/1721.1/12390`.

Ming Li and Paul Vitányi. *An introduction to Kolmogorov complexity and its applications*. Springer, New York, NY, 2008.

Ming Li, Paul Vitányi, et al. *An introduction to Kolmogorov complexity and its applications*, volume 3. Springer, 2008.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Yanyi Liu and Rafael Pass. A direct prf construction from kolmogorov complexity. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 375–406. Springer, 2024.

David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

Pratyush Maini, Skyler Seto, He Bai, David Grangier, Yizhe Zhang, and Navdeep Jaitly. Rephrasing the web: A recipe for compute and data-efficient language modeling. *arXiv preprint arXiv:2401.16380*, 2024.

Per Martin-Löf. The definition of random sequences. *Information and control*, 9(6):602–619, 1966.

Genaro Juárez Martínez, Andrew Adamatzky, and Harold V McIntosh. Phenomenology of glider collisions in cellular automaton rule 54 and associated logical gates. *Chaos, Solitons & Fractals*, 28 (1):100–111, 2006.

Sean McLeish, John Kirchenbauer, David Yu Miller, Siddharth Singh, Abhinav Bhatele, Micah Goldblum, Ashwinee Panda, and Tom Goldstein. Gemstones: A model suite for multi-faceted scaling laws. *arXiv preprint arXiv:2502.06857*, 2025.

Brockway McMillan. Two inequalities implied by unique decipherability. *IRE Transactions on Information Theory*, 2(4):115–116, December 1956. doi: 10.1109/TIT.1956.1056818.

Ralph C Merkle. Secure communications over insecure channels. *Communications of the ACM*, 21 (4):294–299, 1978.

Roger J. Metzger. Sinai-ruelle-bowen measures for contracting Lorenz maps and flows. *Annales de l'I.H.P. Analyse non linéaire*, 17(2):247–276, 2000. URL https://www.numdam.org/item/AIHPC_2000__17_2_247_0/.

Francisco Mota, Scott Aaronson, Luís Antunes, and André Souto. Sophistication as randomness deficiency. In *Descriptional Complexity of Formal Systems: 15th International Workshop, DCFS 2013, London, ON, Canada, July 22-25, 2013. Proceedings 15*, pages 172–181. Springer, 2013.

Preetum Nakkiran, Behnam Neyshabur, and Hanie Sedghi. The deep bootstrap framework: Good online learners are good offline generalizers. *arXiv preprint arXiv:2010.08127*, 2020.

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

OpenAI. GPT-5 System Card. https://cdn.openai.com/gpt-5-system-card.pdf, August 2025. Version dated August 13, 2025. Accessed: 2026-01-05.

Vassilis Papadopoulos, Jérémie Wenger, and Clément Hongler. Arrows of time for large language models. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=UpSe7ag34v.

Tim Pearce and Jinyeop Song. Reconciling kaplan and chinchilla scaling laws. *arXiv preprint arXiv:2406.12907*, 2024.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin A Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024.

Ya B Pesin. Characteristic lyapunov exponents and smooth ergodic theory. *Russian Mathematical Surveys*, 32(4):55, 1977.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Ran Raz. Fast learning requires good memory: A time-space lower bound for parity learning. *Journal of the ACM (JACM)*, 66(1):1–18, 2018.

Markus Redeker. A language for particle interactions in one-dimensional cellular automata. *arXiv preprint arXiv:1012.0158*, 2010.

Jorma Rissanen. Minimum description length principle. *Encyclopedia of statistical sciences*, 7, 2004.

John K Salmon, Mark A Moraes, Ron O Dror, and David E Shaw. Parallel random numbers: as easy as 1, 2, 3. In *Proceedings of 2011 international conference for high performance computing, networking, storage and analysis*, pages 1–12, 2011.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J Reddi. Reasoning with latent thoughts: On the power of looped transformers. *arXiv preprint arXiv:2502.17416*, 2025.

Jürgen Schmidhuber. The speed prior: a new simplicity measure yielding near-optimal computable predictions. In *International conference on computational learning theory*, pages 216–228. Springer, 2002.

Glenn Shafer and Vladimir Vovk. The sources of kolmogorov's grundbegriffe. 2006.

Cosma Rohilla Shalizi and James P Crutchfield. Computational mechanics: Pattern and prediction, structure and simplicity. *Journal of Statistical Physics*, 104(3–4):817–879, 2001.

Claude E Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27 (3):379–423, 1948.

Claude E. Shannon. Programming a computer for playing chess. *Philosophical Magazine*, 41(314): 256–275, 1950.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama. `https://cerebras.ai/blog/slimpajama-a-627b-token-cleaned-and-deduplicated-version-of-redpajama`, 2023. URL `https://huggingface.co/datasets/cerebras/SlimPajama-627B`.

Peiyang Song, Kaiyu Yang, and Anima Anandkumar. Towards large language models as copilots for theorem proving in lean. *arXiv preprint arXiv:2404.12534*, 2024.

Jacob Steinhardt, Gregory Valiant, and Stefan Wager. Memory, communication, and statistical queries. In *Conference on Learning Theory*, pages 1490–1516. PMLR, 2016.

Ilya Sutskever. Gpt-2. Presented at the Scaled Machine Learning Conference 2019, Computer History Museum, 2019. `https://www.youtube.com/watch?v=T0I88NhR_9M`.

Sebastiaan A Terwijn. The mathematical foundations of randomness. In *The Challenge of Chance: A Multidisciplinary Approach from Science and the Humanities*, pages 49–66. Springer International Publishing Cham, 2016.

Lucas Theis and Noureldin Y Ahmed. Algorithms for the communication of samples. In *International Conference on Machine Learning*, pages 21308–21328. PMLR, 2022.

Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

Nikolay Vereshchagin and Paul M.B. Vitányi. Kolmogorov's structure functions and model selection. *IEEE Transactions on Information Theory*, 50(12):3265–3290, 2004.

John von Neumann. Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, 100(1):295–320, 1928.

John Von Neumann. Various techniques used in connection with random digits. *Appl. Math Ser*, 12 (36-38):3, 1951.

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

Gail Weiss, Yoav Goldberg, and Eran Yahav. Thinking like transformers. In *International Conference on Machine Learning*, pages 11080–11090. PMLR, 2021.

Stephen Wolfram and M Gad-el Hak. A new kind of science. *Appl. Mech. Rev.*, 56(2):B18–B19, 2003.

Yilun Xu, Shengjia Zhao, Jiaming Song, Russell Stewart, and Stefano Ermon. A theory of usable information under computational constraints. *arXiv preprint arXiv:2002.10689*, 2020.

Greg Yang and Etai Littwin. Tensor programs ivb: Adaptive optimization in the infinite-width limit. *arXiv preprint arXiv:2308.01814*, 2023.

Greg Yang, Edward J Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer. *arXiv preprint arXiv:2203.03466*, 2022.

Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 80–91. IEEE Computer Society, 1982. doi: 10.1109/SFCS.1982.95.

Shiyang Zhang, Aakash Patel, Syed A Rizvi, Nianchen Liu, Sizhuang He, Amin Karbasi, Emanuele Zappala, and David van Dijk. Intelligence at the edge of chaos. *arXiv preprint arXiv:2410.02536*, 2024.

Xiao Zhang, Xingjian Li, Dejing Dou, and Ji Wu. Measuring information transfer in neural networks. *arXiv preprint arXiv:2009.07624*, 2020.

Hattie Zhou, Arwen Bradley, Etai Littwin, Noam Razin, Omid Saremi, Josh Susskind, Samy Bengio, and Preetum Nakkiran. What algorithms can transformers learn? a study in length generalization. *arXiv preprint arXiv:2310.16028*, 2023.

Appendix Outline

This appendix provides the technical details, proofs, and experimental specifications supporting the main text.

Appendix A presents rigorous proofs of all theoretical results, including properties of cryptographically secure pseudorandom number generators under time-bounded entropy and epiplexity (Theorem 9), creation of information through deterministic transformations (Theorem 12), the existence of high-epiplexity random variables (Theorem 10), the factorization dependence of information content (Theorem 13).

Appendix B details the practical methodology for estimating epiplexity, covering both prequential and requential coding implementations, hyperparameter optimization procedures for compute-optimal two-part codes, the connection between prequential and requential estimates under a static teacher assumption, and a solvable analytical model combining neural scaling laws with prequential coding. We also establish general properties showing that optimal model size and training tokens increase monotonically with compute budget, that optimal training tokens for prequential coding generally saturate at the test set size for large compute budgets, and that epiplexity and per-token entropy exhibit predictable monotonicity with respect to dataset size.

Appendix C provides comprehensive experimental specifications for all empirical results, including architectural choices, hyperparameters, and dataset details for elementary cellular automata experiments, easy and hard variants of induction tasks, chess experiments (with both pre-training data formatting and downstream evaluation tasks), natural data experiments on OpenWebText and CIFAR-5M, comparisons between prequential and requential coding estimates, and scaling law estimation procedures.

Appendix D presents executable RASP-L code demonstrating that elementary cellular automaton evolution rules can be implemented within the transformer computational model, providing constructive evidence that autoregressive transformers are capable of solving these tasks.

Appendix E contains definitions of elementary cellular automata and Conway's Game of Life, emergence examples referenced in the paper.

Appendix F explores additional examples illustrating the relationship between emergence and epiplexity, including the Lorenz system as a case study in chaotic dynamics where entropy is created at a rate determined by Lyapunov exponents, and chess strategy as exemplified by the contrast between AlphaZero's multi-million parameter networks solution at moderate compute and the simple minimax algorithm available at very high compute.

Appendix G argues that induction phenomena occur not merely in autoregressive models; instead, the key requirement is maximum likelihood estimation rather than autoregressive factorization specifically.

Appendix H provides a more comprehensive review of MDL, in particular on two-part code, one-part code and the notion of regret which can be interpreted as a generalization of epiplexity for all coding schemes.

Compute Resources. A cluster of 6 2080Ti was used for many of the smaller scale experiments. A cluster of 6 Titan RTX and 32 TPUv4 provided by the Google TPU Research Cloud was used for the more computationally expensive natural data experiments. We refer the reader to Jiang et al. (2025) for computational resources required in evaluating ADO.

Licenses. The Chess data used in Section 5.2 is released under Creative Commons CC0 license (`database.lichess.org/`). The OpenWebText dataset used in Section 6.2 is released under Creative Commons CC0 license.

Appendix A. Proofs

First, we prove two short lemmas about the basic properties of epiplexity and time-bounded entropy.

> **Lemma 15 (Maximum expected description length)** *For any random variable X on $\{0,1\}^n$ there exists constants c_1, c_2, c_3 such that:*
>
> $$\mathrm{S}_T(X) + \mathrm{H}_T(X) \leq n + c_1 \tag{11}$$
>
> *for time bounds $T(n) \geq c_2 n + c_3$.*

Proof Let U_n be the uniform distribution $Q_{\mathrm{unif}}(x) = 2^{-n}$. Q_{unif} can be computed in linear time (just by outputting 2^{-n} for each input) and with a program of constant size c_1 and in time $c_2 n + c_3$ with constants depending on the Turing machine..

$$|Q_X^\star| + \mathbb{E}[-\log Q_X^\star(x)] \leq |Q_{\mathrm{unif}}| + \mathbb{E}[-\log Q_{\mathrm{unif}}(x)] \leq c + n.$$

■

> **Lemma 16 (Time-bounded entropy of uniform distribution)** *Let $X = U_n$ be the uniform distribution on $\{0,1\}^n$. The time-bounded entropy of U_n for $T(n) \geq c_2 n + c_3$ is:*
>
> $$n \leq \mathrm{H}_T(X) \leq n + c_1. \tag{12}$$

Proof

For the lower bound, we have

$$\mathbb{E}_X[-\log Q(X)] = \mathrm{H}(X) + D_{\mathrm{KL}}(P_X \| Q) \geq \mathrm{H}(X) = n$$

given that the KL is always positive. For the upper bound, we have that

$$\mathrm{H}_T(X) \leq \mathrm{MDL}_T(X) \leq n + c$$

.

■

A.1 CSPRNGs have (nearly) maximal time-bounded Entropy and low epiplexity

> **Theorem 17** *Let $X = U_k$ and $n = \ell(k)$ for a non-uniform PRG G that admits advantage $\varepsilon(n)$. Then, for every polynomial time bound $T(n)$,*
>
> $$\mathrm{H}_T(G(U_k)) \ \geq \ n - 2 - n\,\varepsilon(k). \tag{13}$$

Proof Fix $\mathrm{P} \in \mathcal{P}_T$ and let $L(x) = -\log P(x)$. For each precision level $t \in \{1, 2, \ldots, n\}$, we define the following distinguisher:

$$D_t(x) = \mathbb{1}\{L(x) \leq n - t\} = \mathbb{1}\{P(x) \geq 2^{-(n-t)}\}.$$

For any solution P for MDL_T, we have that $\text{MDL}_T(X) = |P| + \mathbb{E}[-\log P(X)] \leq n + c$. Since both quantities are positive, it must be the case that $|P| \leq n$, which means that $|P| \in \text{poly}(n)$. Since P belongs in $\mathcal{P}_T$ and cannot be longer than n, each D_t is a non-uniform PPT algorithm with polysized advice (i.e., P) that CSPRNGs are secure against.

Uniform threshold bound. Let U_n be uniform on $\{0,1\}^n$ and set $A_t := \{x : D_t(x) = 1\}$.

$$1 \geq \sum_x P(x) \geq \sum_{x \in A_t} P(x) \geq |A_t| 2^{-(n-t)} \Rightarrow |A_t| \leq 2^{n-t}.$$

Hence ,

$$\Pr[D_t(U_n) = 1] = \frac{|A_t|}{2^n} \leq \frac{2^{n-t}}{2^n} = 2^{-t}.$$

CSPRNG transfers bound to $X := G(U_k)$. By the security of G, for each t,

$$\Pr[D_t(X) = 1] \ \leq \ \Pr[D_t(U_n) = 1] + \varepsilon(k) \ \leq \ 2^{-t} + \varepsilon(k),$$

From threshold probabilities to an entropy lower bound. For any non-negative random variable Z, we have the layercake representation:

$$\mathbb{E}[Z] = \sum_{u=0}^{\infty}(1 - P(Z \leq u)) \tag{14}$$

$$n - \mathbb{E}[Z] = \sum_{u=0}^{n-1} 1 - \sum_{u=0}^{\infty}(1 - P(Z \leq u)) \tag{15}$$

$$= \sum_{u=0}^{n-1} 1 - \sum_{u=0}^{n-1}(1 - P(Z \leq u)) - \sum_{u=n}^{\infty}(1 - P(Z \leq u)) \tag{16}$$

$$= \sum_{u=0}^{n-1} P(Z \leq u) - \sum_{u=n}^{\infty}(1 - P(Z \leq u)) \tag{17}$$

$$\leq \sum_{u=0}^{n-1} P(Z \leq u). \tag{18}$$

Now we change the bounds to be in terms of t with $t = n - u$. The lower bound becomes $t = n$. The upper bound becomes $t = 1$, which yields

$$n - \mathbb{E}[Z] \leq \sum_{u=0}^{n-1} P(Z \leq u) = \sum_{t=1}^{n} P(Z \leq n - t).$$

Let $Z = L(X) = -\log P(X)$:

$$n - \mathbb{E}[Z] \leq \sum_{t=1}^{n} P(Z \leq n - t) = \sum_{t=1}^{n} P(D_t(X) = 1) \leq \sum_{t=1}^{n} 2^{-t} + \varepsilon(k) \leq 1 + n\varepsilon(k).$$

The last two steps come from the fact that X is a CSPRNG. This means that:

$$n - \mathbb{E}[L(X)] \leq 1 + n\varepsilon(k) \Rightarrow \mathbb{E}[-\log P(X)] \geq n - n\varepsilon(k) - 1.$$

Since this is true for any $P \in \mathcal{P}_T$, taking the minimum over yields:

$$\text{H}_T(X) = \text{H}_T(G(U_n)) = \min_{P \in \mathcal{P}_T} \mathbb{E}[-\log P(X)] \geq n - n\varepsilon(k) - 1.$$

$\blacksquare$

A.2 Deterministic transformation can increase time bounded entropy and epiplexity

> **Theorem 18** *Let $G : \{0,1\}^k \to \{0,1\}^n$ be a* CSPRNG *which admits advantage $\varepsilon(k)$ and U_k be the uniform distribution.* $\mathrm{H_{Poly}}(G(U_k)) > \mathrm{H_{Poly}}(U_k) + n - n\varepsilon(k) - k - O(1)$. *Proof: see Appendix A.1.*

Proof By Lemma 15 applied to the uniform distribution on $\{0,1\}^k$, there is an absolute constant c such that

$$\mathrm{H_{poly}}(U_k) \leq k + c.$$

Rearranging gives $k \geq \mathrm{H_{poly}}(U_k) - O(1)$. Combining this with the assumed CSPRNG lower bound (Lemma 17),

$$\mathrm{H_{poly}}(G(U_k)) \geq n - 2 - n\varepsilon(k),$$

we obtain,

$$\mathrm{H_{poly}}(G(U_k)) - \mathrm{H_{poly}}(U_k) \geq n - 2 - n\varepsilon(k) - (k + c)$$
$$\Rightarrow \mathrm{H_{Poly}}(G(U_k)) > \mathrm{H_{Poly}}(U_k) + n - n\varepsilon(k) - k - O(1).$$

∎

A.3 CSPRNGs have low epiplexity

> **Theorem 19** *Let $X = U_k$ and $n = \ell(k)$ for CSPRNG G that admits advantange $\varepsilon(n)$. Then, for every polynomial time bound $T(n)$, the epiplexity of $Y = G(X)$ is,*
>
> $$\mathrm{S}_T(Y) \leq c + n\varepsilon(k). \tag{19}$$

Proof We know from Theorem 17 that $\mathrm{H}_T(G(U_k)) \geq n - n\varepsilon(k) - 2$, which means:

$$\mathrm{S}_T(Y) + \mathrm{H}_T(Y) \geq \mathrm{S}_T(Y) + n - n\varepsilon(k) - 2. \tag{20}$$

We also have from Lemma 15 that $\mathrm{S}_T(Y) + \mathrm{H}_T(Y) \leq n + c$. Combining these two results yields:

$$\mathrm{S}_T(Y) + n - n\varepsilon(k) - 1 \leq n + c \Rightarrow \mathrm{S}_T(Y) \leq c + n\varepsilon(k). \tag{21}$$

∎

A.4 Existence of High Epiplexity random variables

Definition 20 (Pseudorandom functions (PRF)) *Let* PRF *be the class of keyed functions $F :$ $\{0,1\}^k \times \{0,1\}^n \to \{0,1\}^m$ that are computable in polynomial time and satisfy the following property: For any probabilistic polynomial-time distinguisher D with oracle access to the provided function,*

$$\left| \Pr_{K \sim U_k}[D^{F_K(\cdot)}] - \Pr_{f \sim \mathcal{F}_n}[D^{f(\cdot)}] \right| < \frac{1}{n^c}, \tag{22}$$

for all integers $c > 0$ and sufficiently large n. Here, $F_K(\cdot)$ denotes the function $F(K, \cdot)$ with the key K fixed, and $\mathcal{F}_n$ is the set of all functions mapping $\{0,1\}^n$ to $\{0,1\}^m$.

Cryptographic assumptions. Assume one-way functions exist (secure against non-uniform PPT adversaries with inversion probability at most $\varepsilon(n)$). By standard constructions (Håstad et al., 1999), this implies the existence of PRFs secure against non-uniform PPT distinguishers with advantage $\text{poly}(\varepsilon(n))$ (and in particular negligible if $\varepsilon(n)$ is negligible).

Definition 21 (Heavy set) *For a distribution Q on $\{0,1\}^n$, $m < n$, and a fixed threshold $t \geq 0$, the (Q,t)-heavy set is:*

$$A_{Q,t} := \{z : Q(z) \geq 2^{-2(m+t)}\}. \tag{23}$$

> **Lemma 22** *Let P be a distribution on $\{0,1\}^n$ with entropy $\mathrm{H}(P) = m$. If $\mathrm{KL}(P,Q) \leq t$, then $P(A_{Q,t}) \geq \frac{1}{2}$.*

Proof First, observe the standard inequality:

$$\mathbb{E}_{z \sim P}\left[\log \frac{1}{Q(z)}\right] = \mathrm{H}(P) + \mathrm{KL}(P\|Q) \leq m + t.$$

Applying Markov's inequality, we get:

$$\Pr_{z \sim P}\left[\log \frac{1}{Q(z)} \geq 2(m+t)\right] \leq \frac{\mathbb{E}[-\log Q(z)]}{2(m+t)} \leq \frac{1}{2}. \tag{24}$$

Taking the complement gives:

$$\Pr_{z \sim P}\left[\log \frac{1}{Q(z)} \leq 2(m+t)\right] = \Pr_{z \sim P}\left[Q(z) \geq 2^{-2(m+t)}\right] = P(A_{Q,t}) \geq \frac{1}{2}. \tag{25}$$

∎

> **Lemma 23** *Let U_n be the uniform distribution over $\{0,1\}^n$, the weights of $A_{Q,t}$ under U_n is $U_n(A_{Q,t}) \leq 2^{-(n-2(m+t))}$*

Proof For $z \sim U_n$, we have $\mathbb{E}_{z \sim U_n}[Q(z)] = \sum_z 2^{-n} Q(z) = 2^{-n}$. Applying Markov' inequaltiy:

$$\Pr_{z \sim U_n}\left[Q(z) \geq 2^{-2(m+t)}\right] \leq \frac{\mathbb{E}_{z \sim U_n}[Q(z)]}{2^{-2(m+t)}} \leq 2^{-n+2(m+t)} = 2^{-(n-2(m+t))}. \tag{26}$$

∎

> **Theorem 24** *If there exists a PRF family $F_K : \{0,1\}^m \to \{0,1\}^k$ that is indexed by $K \in \{0,1\}^m$ and secure against a non-uniform PPT distinguisher D_m allowing for an advantage of at most $\varepsilon(m)$, there exists n_0 such that for all $n = m + k \geq n_0$, there exists a sequence of random variables $\{X_k\}_{k=1}^n$ over $\{0,1\}^n$ such that $\mathrm{S}_{\text{Poly}}(X_n) = \Omega(\log n)$.*

Proof We will prove the existence of such P via a counting argument. First, we define the family of distributions of interest. Concretely, we draw a sample P_K as follows:

1. Sample $x \sim U_m$

2. Output $z = (x, F_K(x)) \in \{0,1\}^n$

Since F_K is a deterministic function, $\mathrm{H}(P_K) = m$.

We also defined a *keyed model* Q_K that models P_K by directly storing the key K and the program for generating PRF from K inside its program:

$$Q_K(x,y) = 2^{-m}\mathbb{1}\{y = F_K(x)\}.$$

This model matches the density of P_K so $\mathrm{KL}(P_K\|Q_K) = 0$, and:

$$L(Q_K, P_K) = |Q_K|+\mathrm{H}(P_K) \leq m + c_1 + m = 2m + c_1.$$

c_1 is the constant overhead to implement the PRF evaluation and sampling wrapper under a fixed encoding (i.e., a UTM).

Constructing distinguisher from Q. Given a model Q and its heavy set $A_{Q,t}$ (Definition 21), we can turn Q into a *single-query* distinguisher D^O:

1. Sample $x \sim U_m$ and query the oracle $y = O(x)$ and set $z = (x, y)$.

2. Output 1 if $z \in A_{Q,t}$ i.e., $Q(z) \geq 2^{-2(m+t)}$ else 0.

If O is a truly random function R, then $(x, R(x))$ follows U_n and by Lemma 23:

$$\Pr[D^R = 1] = \Pr_{z \sim U_n}[z \in A_{Q,t}] \leq 2^{-(n-2(m+t))} \tag{27}$$

If O is the PRF F_K for a K that satisfies $\mathrm{KL}(P_K\|Q) \leq t$, Lemma 22 gives:

$$\Pr\left[D^{F_K} = 1 \mid \mathrm{KL}(P_K\|Q) \leq t\right] \geq \frac{1}{2}. \tag{28}$$

Let $p_{Q,t} = \Pr_K[\mathrm{KL}(P_K\|Q) \leq t]$. We can average over all possible K and obtain the following bound:

$$\Pr\left[D^{F_K} = 1\right] \geq \Pr_K[\mathrm{KL}(P_K\|Q) \leq t]\Pr\left[D^{F_K} = 1 \mid \mathrm{KL}(P_K\|Q) \leq t\right] \geq \frac{1}{2}p_{Q,t}. \tag{29}$$

Therefore, the distinguishing advantage of D^O is:

$$\mathsf{Adv}(D^O) = \Pr\left[D^{F_K} = 1\right] - \Pr\left[D^R = 1\right] \geq \frac{1}{2}p_{Q,t} - 2^{-(n-2(m+t))}. \tag{30}$$

Rearranging:

$$p_{Q,t} \leq 2\mathsf{Adv}(D^O) + 2 \cdot 2^{-(n-2(m+t))}. \tag{31}$$

Since F_K is a PRF and D_O is a PPT distinguisher, the advantage is upperbounded by $\varepsilon(m)$:

$$p_{Q,t} \leq 2\varepsilon(m) + 2 \cdot 2^{-(n-2(m+t))}. \tag{32}$$

Union bound over short models. Given a maximum program length s, there are at most 2^{s+1} candidate programs Q with $|Q|\leq s$. Applying union bound on all such Q's:

$$\Pr_K\left[\exists Q : |Q|\leq s \wedge \mathrm{KL}(P_K\|Q) \leq t\right] \leq 2^{s+1}p_{Q,t} \leq 2^{s+1}\left(2\varepsilon(m) + 2 \cdot 2^{-(n-2(m+t))}\right). \tag{33}$$

Now, it suffices to choose parameters such that the RHS of equation 33 is smaller than 1, which implies there exists a hard key $K^\star$ such that:

$$\mathrm{KL}(P_{K^\star}\|Q) > t, \ \forall Q \text{ satisfying } |Q|\leq s. \tag{34}$$

MDL lower bound from $K^\star$. For $K^\star$, every $|Q| \leq s$ satisfies:

$$L(Q, P_{K^\star}) = |Q| + H(P^\star) + KL(P_{K^\star} \| Q) \geq H(P^\star) + KL(P_{K^\star} \| Q) \geq m + t.$$

Meanwhile, the keyed model $Q_{K^\star}$ satisfies: $L(Q_{K^\star}, P_{K^\star}) \leq 2m + c_1$. If we set:

$$t = m + c_1 + \Delta,$$

we get a margin of Δ:

$$L(Q, P_{K^\star}) \geq m + m + c_1 + \Delta > 2m + c_1 \geq L(Q_{K^\star}, P_{K^\star}). \tag{35}$$

This implies that there exists at least one model that achieves a lower description length than any Q with $|Q| \leq s$ and the MDL minimizer must have $|Q^\star| > s$.

Choosing parameters. Set:

- $s = \log m$

- $\Delta = \log m$

- $t = m + c_1 + \Delta = m + c_1 + \log m$

- $k = 4m + 4\Delta + 2c_1$

We now plug these values into Equation 33. First, $2^{s+1} = \text{poly}(m)$ and $\lim_{m \to \infty} 2^{s+1} \cdot 2\varepsilon(m) = 0$. For the second term:

$$2^{s+1} \cdot 2 \cdot 2^{-(n-2(m+t))}$$
$$= 2^{\log m + 1} \cdot 2 \cdot 2^{-(m+4m+4\Delta+2c_1 - 2(m+m+c_1+\log m))}$$
$$= 2^{\log m + 2} \cdot 2^{-(5m + 4\log m + 2c_1 - 2(2m+c_1+\log m))}$$
$$= 2^{\log m + 2} \cdot 2^{-(m + 2\log m)}$$
$$= 2^{-m - \log m + 2}.$$

This term also approaches 0 as m increases. So for sufficiently large m the RHS of Equation 33 is less than 1 as desired.

■

A.5 Information Content is not Independent of Factorization

> **Theorem 25 (OWP induces entropy asymmetry)** *Let* $f : \{0,1\}^n \to \{0,1\}^n$ *be a polynomial-time computable one-way permutation secure against non-uniform PPT inverters with negligible success probability. Let* $X = U_n$ *and* $Y = f(X)$. *Let* $H_{\text{poly}}(\cdot)$ *and* $H_{\text{poly}}(\cdot \mid \cdot)$ *be defined as in Definition 8. Then for every constant* $c > 0$ *there exists* N *such that for all* $n \geq N$,
> $$H_{\text{poly}}(X \mid Y) + H_{\text{poly}}(Y) > H_{\text{poly}}(Y \mid X) + H_{\text{poly}}(X) + c \log n.$$

Proof We prove bounds on each term.

Unconditional terms $H_{\mathrm{poly}}(X)$ and $H_{\mathrm{poly}}(Y)$. Since $X = U_n$ and f is a permutation, $Y = f(X)$ is also uniform on $\{0,1\}^n$. By Lemma 15 (time-bounded entropy of the uniform distribution), there is a constant c_0 such that

$$n \leq H_{\mathrm{poly}}(X) \leq n + c_0, \qquad n \leq H_{\mathrm{poly}}(Y) \leq n + c_0.$$

In particular, $c_0 \leq H_{\mathrm{poly}}(Y) - H_{\mathrm{poly}}(X) \leq c_0$, so $H_{\mathrm{poly}}(Y) - H_{\mathrm{poly}}(X) = O(1)$.

Forward conditional term $H_{\mathrm{poly}}(Y \mid X)$. There is a deterministic conditional sampler that on input x outputs $f(x)$. For this sampler, $P(Y \mid X) = 1$ almost surely, hence $\log(1/P(Y \mid X)) = 0$ almost surely. Since $H_{\mathrm{poly}}(Y \mid X)$ is the expected log-loss of the MDL-optimal conditional sampler, we obtain

$$H_{\mathrm{poly}}(Y \mid X) = O(1).$$

Hard conditional term $H_{\mathrm{poly}}(X \mid Y)$. Let $P^\star := P^\star_{X|Y}$ be the MDL-optimal conditional probabilistic model for $X \mid Y$ over the class of non-uniform PPT model, and define

$$\phi(y) \ := \ \Pr_{u \sim U_\infty} \left[\mathrm{Sample}_{P^\star_{X|y}}(u) = f^{-1}(y) \right].$$

Because $Y = f(X)$ and f is a permutation, we have $X = f^{-1}(Y)$ almost surely, and thus

$$P^\star(X \mid Y) = P^\star(f^{-1}(Y) \mid Y) = \phi(Y) \qquad \text{a.s.}$$

Therefore

$$H_{\mathrm{poly}}(X \mid Y) = \mathbb{E}\left[\log \frac{1}{P^\star(X \mid Y)} \right] = \mathbb{E}\left[\log \frac{1}{\phi(Y)} \right].$$

By Jensen's inequality for the convex function $\log(1/t)$,

$$\mathbb{E}\left[\log \frac{1}{\phi(Y)} \right] \ \geq \ \log \frac{1}{\mathbb{E}[\phi(Y)]}.$$

Now consider the inverter $\mathcal{I}$ that on input y runs the sampler $P^\star(X \mid Y)$ once and outputs the resulting x. Since $P^\star$ is a non-uniform PPT sampler, $\mathcal{I}$ is a non-uniform PPT inverter. Moreover, its inversion success probability is exactly

$$\Pr\left[\mathcal{I}(Y) = f^{-1}(Y) \right] = \mathbb{E}[\phi(Y)].$$

Equivalently (since $Y = f(X)$),

$$\Pr_{X \sim U_n} \left[\mathcal{I}(f(X)) = X \right] = \mathbb{E}[\phi(Y)].$$

By one-wayness, this success probability is negligible. In particular, for every constant $c > 0$ there exists N such that for all $n \geq N$,

$$\mathbb{E}[\phi(Y)] \leq n^{-c}.$$

Plugging into the Jensen bound yields, for all $n \geq N$,

$$H_{\mathrm{poly}}(X \mid Y) \ \geq \ \log \frac{1}{\mathbb{E}[\phi(Y)]} \ \geq \ c \log n.$$

Combine. For $n \geq N$, we have

$$H_{\mathrm{poly}}(X \mid Y) + H_{\mathrm{poly}}(Y) \geq c \log n + H_{\mathrm{poly}}(Y) \tag{36}$$

$$\geq c \log n + H_{\mathrm{poly}}(X) - O(1) \tag{37}$$

$$= H_{\mathrm{poly}}(Y \mid X) + H_{\mathrm{poly}}(X) + c \log n - O(1), \tag{38}$$

where we used $H_{\mathrm{poly}}(Y \mid X) = O(1)$ and $H_{\mathrm{poly}}(Y) - H_{\mathrm{poly}}(X) \geq -c_0$. ∎

Corollary 26 *Let f be a one-way permutation and lef $X = \text{Unif}(\{0,1\}^n), Y = f(X)$. Define $\mathcal{P}$ as a family of probabilistic generative model that allows for multiple factorizations of the data, ie $P \in \mathcal{P}$ it can make predictions $P_{1 \to 2}(X, Y) = P_1(X)P_2(Y; X)$ and $P_{2 \to 1}(X, Y) = P_2(Y)P_1(X; Y)$ for the functions $P_1(\cdot), P_1(\cdot\,;\cdot), P_2(\cdot), P_2(\cdot\,;\cdot)$ that are normalized probability distributions over the first variable.*

Suppose that P fits the forward direction of f (and the input uniform distributions)

$$\mathbb{E}[-\log P_1(X)] \leq n + \varepsilon$$
$$\mathbb{E}[-\log P_2(f(X) \mid X)] \leq \varepsilon$$

then it must violate Bayes theorem $P_{1 \to 2} = P_{2 \to 1}$ by a margin growing with n. Specifically, for any value of c there exists N such that for all $n > N$, there exists at least one $x \in \{0,1\}^n$ such that

$$P_1(x)P_2(f(x); x) > n^c 2^{-2\varepsilon} P_2(f(x))P_1(x; f(x)) \tag{39}$$

Proof From Theorem 25 which applies also for each P, we have

$$\mathbb{E}\left[-\log P_2(X; Y)\right] > c \log n.$$

The minimim value of $\mathbb{E}\left[-\log P_2(f(X))\right]$ is n since f is a bijection. Assembling these components,

$$\mathbb{E}\left[\log \frac{P_1(X)P_2(f(X); X)}{P_2(f(X))P_1(X; f(X))}\right] > c \log n - 2\varepsilon. \tag{40}$$

Since the inequality holds in expectation, it also must hold for at least one value of X. Exponentiating provides the final result. ∎

A.6 Problems with time-bounded sophistication

Epiplexity can be seen as a time-bounded and distributional generalization of sophistication. A natural question is whether we can directly define a time-bounded version of sophistication for individual strings. We show below that a naive time-bounded generalization degenerates: it makes the "model" part essentially constant for *every* string.

Preliminaries. Fix a reference universal (prefix-free or plain) Turing machine U. For a program p and auxiliary input d, we write $U(p, d)$ for the output of running p on input d. The length of a binary string p is denoted $|p|$. A program p is *total* if $U(p, d)$ halts for every input d (i.e., p computes a total function).

We write $K(x)$ for Kolmogorov complexity (plain or prefix; the choice only changes values by $O(1)$). For a time bound $t(\cdot)$, the time-bounded Kolmogorov complexity is

$$K^t(x) := \min\{ |q| : U(q) \text{ outputs } x \text{ within } t(|x|) \text{ steps} \}.$$

(Any standard time-constructible t suffices for the discussion.)

We adopt the definition of sophistication from Koppel (1988) and Antunes et al. (2005), phrased for finite strings as in later expositions. For a significance level $c \geq 0$, the sophistication of x is

Definition 27 (Sophistication at significance c)

$$\text{soph}_c(x) := \min_p \left\{ |p| : p \text{ is total and } \exists d \text{ such that } U(p,d) = x \text{ and } |p|+|d| \leq K(x) + c \right\}.$$

Intuitively, (p,d) is a near-optimal two-part description of x. The requirement that p be *total* is crucial: it prevents taking p to be a tiny universal interpreter and pushing all information into d (since a universal interpreter is not total). One of the most intuitive attempts at "time-bounded sophistication" is to simply replace $K(x)$ by the time-bounded complexity $K^t(x)$ in Definition 27.

Definition 28 (Naive time-bounded sophistication) *Fix a time bound $t(\cdot)$ and significance level $c \geq 0$. Define*

$$\text{soph}_c^t(x) := \min_p \left\{ |p| : p \text{ is total and } \exists d \text{ such that } U(p,d) = x \text{ and } |p|+|d| \leq K^t(x) + c \right\}.$$

The definition above *collapses*, essentially because time bounds make it easy to "totalize" a universal interpreter by adding a timeout.

> **Lemma 29 (Naive time-bounded sophistication is $O(1)$)** *For every time bound $t(\cdot)$ and every $c \geq 0$, there exists a constant C_t (depending only on t and the choice of U) such that for every string x,*
> $$\text{soph}_c^t(x) \leq C_t.$$
> *In particular, $\text{soph}_c^t(x)$ does not meaningfully distinguish structured strings from random-looking strings.*

Proof [sketch] Fix t. Let p_{tl} be a constant-size program that, on input d, simulates $U(d)$ for at most $t(|x|)$ steps (or more generally for the same time budget used in the definition of $K^t(x)$), and: (i) if the simulation halts within the budget, output the same result; otherwise (ii) output a fixed default string (say 0). By construction, p_{tl} is *total* (it always halts, because it enforces a timeout).

Now let $d^\star$ be a shortest program witnessing $K^t(x)$, i.e., $|d^\star| = K^t(x)$ and $U(d^\star)$ outputs x within the allowed time. Then $U(p_{\text{tl}}, d^\star) = x$. Moreover,

$$|p_{\text{tl}}|+|d^\star| = |p_{\text{tl}}|+K^t(x) \leq K^t(x) + c \quad \text{for all } c \geq |p_{\text{tl}}|.$$

Thus p_{tl} is feasible in Definition 28, giving $\text{soph}_c^t(x) \leq |p_{\text{tl}}| = C_t$ for all x. ∎

In the original (unbounded-time) Definition 27, totality prevents a universal interpreter from being used as the "model" part, because such an interpreter cannot halt on inputs that encode non-halting computations. However, once we commit to a time bound in the *optimality criterion* (i.e., we compare against $K^t(x)$), the data part d can be chosen to be a short program that is *guaranteed to halt quickly*. A constant-size *clocked interpreter* p_{tl} is then total and suffices for every x, pushing all of the description length into d. This is precisely the sense in which the naive time-bounded generalization becomes degenerate.

Appendix B. Measuring Epiplexity

B.1 Further details on estimating epiplexity

Here we provide further details on measuring epiplexity.

Evaluating code lengths and time bounds. As described in Section 4, evaluating the code length for the model boils down to tracking the training losses (prequential) or teacher-student KL (requential) at each step i :

$$|\mathrm{P}_{\mathrm{preq}}| \approx \sum_{i=0}^{M-1} (\log 1/P_i(Z_i) - \log 1/P_M(Z_i)), \tag{41}$$

$$|\mathrm{P}_{\mathrm{req}}| \approx \sum_{i=0}^{M-1} \mathrm{KL}(P_i^{\mathrm{t}} \| P_i^{\mathrm{s}}). \tag{42}$$

For prequential coding, we need to compute the loss of the final model summed over the entire training dataset, $\sum_{i=0}^{M-1} \log 1/P_M(Z_i)$, which is time-consuming if done exactly. Since all of our experiments are in the one-epoch training regime without data repeat and training data Z_i are drawn i.i.d. (except for the ADO experiment Section 6.4), we make the assumption that the generalization gap is small and estimate $\sum_{i=0}^{M-1} \log 1/P_M(Z_i)$ as $M \log 1/P_M(Z_M)$, where the latter is a rescaled loss for P_M on unseen data Z_M. The i.i.d. assumption breaks down for the ADO experiment Section 6.4, where we instead compute $\sum_{i=0}^{M-1} \log 1/P_M(Z_i)$ exactly.

For requential coding, we need to evaluate the teacher-student KL, $\mathrm{KL}(P^{\mathrm{t}} \| P^{\mathrm{s}})$, at each training step. The KL divergence over sequences decomposes as a sum over token positions and is estimated as:

$$\mathrm{KL}(P^{\mathrm{t}} \| P^{\mathrm{s}}) = \sum_{j=1}^{L} \mathbb{E}_{Z_{<j} \sim P^{\mathrm{t}}} \left[\sum_{Z_j \in \mathcal{V}} P^{\mathrm{t}}(Z_j | Z_{<j}) \log \frac{P^{\mathrm{t}}(Z_j | Z_{<j})}{P^{\mathrm{s}}(Z_j | Z_{<j})} \right] \tag{43}$$

$$\approx \sum_{j=1}^{L} \sum_{Z_j' \in \mathcal{V}} P^{\mathrm{t}}(Z_j' | Z_{<j}) \log \frac{P^{\mathrm{t}}(Z_j' | Z_{<j})}{P^{\mathrm{s}}(Z_j' | Z_{<j})}, \tag{44}$$

where $Z \sim P^{\mathrm{t}}$ is a sample from the teacher, L is the sequence length, and $\mathcal{V}$ is the vocabulary. We evaluate this estimator using the sample Z generated by the teacher to train the student, along with their next-token-prediction logits $\{P^{\mathrm{t}}(Z_j | Z_{<j}), P^{\mathrm{s}}(Z_j | Z_{<j})\}_j$ recorded on the generated sequence

Finally, to estimate the expected entropy code length for the test data $\mathbb{E}[\log 1/P(X)]$ under the trained model P, we use an appropriately scaled empirical entropy code length of a heldout test set $\hat{X}$. Let K and $\hat{K}$ denote the number of examples in each dataset. Then:

$$\mathbb{E}[\log 1/P(X)] = \mathbb{E}\left[\log \frac{1}{\prod_i P(X_i)} \right] \tag{45}$$

$$= \sum_i \mathbb{E}[\log 1/P(X_i)] \tag{46}$$

$$= K \mathbb{E}[\log 1/P(X_1)] \tag{47}$$

$$\approx \frac{K}{\hat{K}} \sum_{i=1}^{\hat{K}} \log 1/P(\hat{X}_i) \tag{48}$$

where we assumed the datasets X and $\hat{X}$ consist of i.i.d. draws from the same distribution. This estimator is simply a scaled version of the standard empirical test loss, and it converges to the true expectation as $\hat{K}$ becomes large. To speedup evaluation, we typically choose $\hat{K} \ll K$, but this choice does not affect our time-bound calculation: for both prequential and requential coding, the total decoding time of the two-part code for the test dataset X is estimated as $6ND + 2N\mathcal{D}$ where N is the number of parameters of the (student) model, D is the number of (student) training tokens,
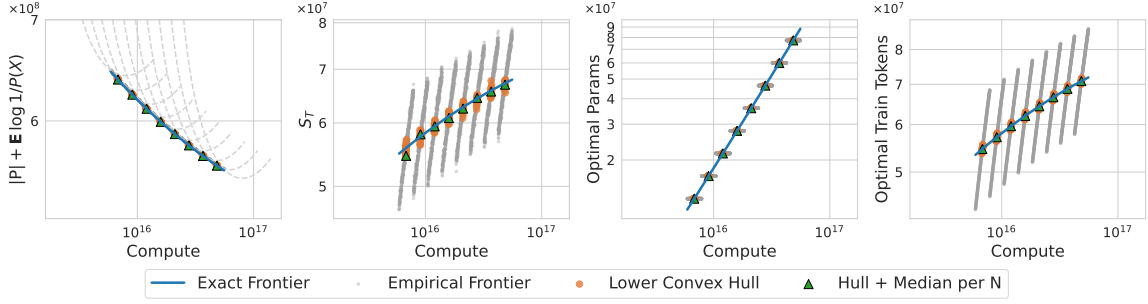
Figure 10: **Estimating the Pareto frontier from a finite number of training runs.** While the exact Pareto frontier is smooth and the optimal model size and training tokens increase smoothly with compute, the empirical frontier is jagged and includes many spurious points due to selecting over only a finite number of hyperparameter combinations. Replacing the empirical Pareto frontier with the lower convex hull and retaining only the median point (ordered by compute) belong to a single training run with a fixed model size results in a much more accurate estimate of the true Pareto frontier. The example training curves are generated using the scaling laws in Hoffmann et al. (2022) and prequential coding. The exact frontier is found via root finding for Equation (56).

and $\mathcal{D}$ is the number of tokens in the test dataset. When evaluating conditional epiplexity $S_T(Y|X)$, decoding time takes into account both the input (X) and label (Y) tokens, but code length only needs to be computed for the label tokens (tokens contributing to the training loss).

Finding Hyperparameters for Compute-Optimal Two-Part Code. To identify models that lead to compute-optimal two-part code, we need to optimize several key hyperparameters, including model size (N), training tokens (D), width-depth ratio, learning rate, etc. Through our early experiments, we found two interventions that reduce the model code length under requential coding: (1) distilling from an exponential moving average (EMA) of teacher checkpoints rather than instantaneous checkpoints, which reduces noise in the distillation signal, and (2) imposing a maximum KL threshold between teacher and student—when exceeded, the teacher is frozen while the student catches up, preventing divergence that would otherwise inflate the code length. The EMA time scale and the maximum KL threshold are additional hyperparameters for requential coding.

In each experiment, we first identify a good learning rate for a small model size and use the Maximum Update Parameterization (Yang et al., 2022) and CompleteP (Dey et al., 2025) to transfer the found learning rate to larger models. We also optimize the EMA time scale and maximum KL threshold for the small model when using requential coding. We then train models of various depths and widths to simultaneously sweep over model size and width-depth ratios, for a total number of training tokens chosen to be larger than the test dataset size $\mathcal{D}$, motivated by the observation that the optimal training tokens typically grows with the model size but do not exceed $\mathcal{D}$ (see Section B.4). To avoid separately training a model for intermediate training token budgets, we record an EMA of the iterates (for requential coding, this is done for the student) under a constant learning rate schedule, rather than using a decaying learning rate schedule, following Hägele et al. (2024). Each training run traces a curve in the $|P|+\mathbb{E}[1/\log P(X)]$ vs T plane as more training tokens are seen. The Pareto frontier of all such curves yields the optimal hyperparameters (N, D, width, depth, etc.) as a function of the compute budget.

Estimating the Pareto Frontier. Due to computational constraints, we can only sweep over a limited set of hyperparameter combinations, which makes the empirical Pareto frontier noisy and jagged; we therefore use the lower convex hull of the resulting curves as a smoother approximation to the true Pareto frontier, a strategy often used in the compute-optimal scaling law literature

(Henighan et al., 2020; McLeish et al., 2025) to overcome similar issues. After applying this strategy, we still often observe that multiple checkpoints from a single training run appear on the Pareto frontier. This is an artifact of finite hyperparameter sweeps: we expect both the optimal training tokens D and model size N to vary smoothly with compute budget, precluding multiple values of D at the same N from lying on the true Pareto frontier. These spurious points cause noisy, oscillatory trends in the estimated epiplexity, as shown in Figure 10. As a simple workaround, we retain only the median point (ordered by compute) per training run (which has a fixed model size) on the lower convex hull.

Sources of errors. In addition to the artifacts produced by finite (N, D) combinations, our estimated epiplexity may differ from the true value for a few reasons: 1) potential systematic errors introduced by using the lower convex hull and taking the median point, 2) using a fixed architecture (e.g., the transformer) and learning algorithm (e.g., requential training with Adam) rather than considering all possible programs, and 3) suboptimality of other hyperparameters, such as the learning rate, Adam (β_1, β_2), etc. In most cases, we believe these sources of errors only contribute sub-leading corrections to the estimated epiplexity that do not impact the result qualitatively. For example, they are unlikely to alter the ordering between datasets if the estimated epiplexity gap is already significant or there is a clear trend along some axis of variation (e.g., number of hidden bits in the induction experiment in Section 5.3.1)

B.2 Prequential Coding Approximates Requential Coding with a Static Teacher

In this section, we show that the prequential coding estimate in Equation (8) can be viewed as an approximation to requential coding with a static teacher, providing an alternative justification for its use beyond the symmetry of information argument.

Consider requential coding with a fixed teacher across all time steps, i.e., $P_i^{\mathrm{t}} = P^{\mathrm{t}}$ for all $i \in \{0, \ldots, M-1\}$. The requential code length becomes

$$|\mathrm{P}_{\mathrm{req}}| \approx \sum_{i=0}^{M-1} \mathrm{KL}(P^{\mathrm{t}} \| P_i^{\mathrm{s}}) = \sum_{i=0}^{M-1} \mathbb{E}_{P^{\mathrm{t}}} \left[\log \frac{1}{P_i^{\mathrm{s}}(X)} - \log \frac{1}{P^{\mathrm{t}}(X)} \right]. \tag{49}$$

Now suppose the static teacher closely matches the true data distribution, i.e., $P^{\mathrm{t}} \approx P_{X_1}$ (we use P_{X_1} in order to refer to the distribution of a single example, not the dataset). Under this assumption, we can make three simplifying approximations:

1. The expectation under the teacher can be replaced by the expectation under the data distribution: $\mathbb{E}_{P^{\mathrm{t}}}[\cdot] \approx \mathbb{E}_{P_{X_1}}[\cdot]$.

2. Training the student on synthetic samples from P^{t} yields similar dynamics to training on real data samples from P_{X_1}.

3. If the student converges to the teacher, then $P_M^{\mathrm{s}} \approx P^{\mathrm{t}}$, allowing us to estimate the teacher's loss $\mathbb{E}_{P_{X_1}}[\log 1/P^{\mathrm{t}}(X)]$ by the final student's loss $\mathbb{E}_{P_{X_1}}[\log 1/P_M^{\mathrm{s}}(X)]$.

Applying these approximations, the requential code length with a static teacher becomes

$$|\mathrm{P}_{\mathrm{req}}| \approx \sum_{i=0}^{M-1} \mathbb{E}_{P_{X_1}} \left[\log \frac{1}{P_i^{\mathrm{s}}(X)} - \log \frac{1}{P_M^{\mathrm{s}}(X)} \right], \tag{50}$$

which, when estimated empirically on real training data $Z_0, \ldots, Z_{M-1} \sim P_{X_1}$, recovers precisely the prequential estimate from Equation (8):

$$|\mathrm{P}_{\mathrm{preq}}| \approx \sum_{i=0}^{M-1} \left(\log \frac{1}{P_i(Z_i)} - \log \frac{1}{P_M(Z_i)} \right). \tag{51}$$

This connection also lends some justification to treating $6ND$ as the decoding time for the model in prequential coding, as it relates to a requential scheme that achieves this runtime. Since a static teacher is generally suboptimal compared to the time-varying teachers used in full requential coding, which can remain close to the student throughout training while still guiding it toward the target distribution, we expect the prequential estimate to be an overestimate of the requential code length. This is consistent with the empirical observations in Figure 2c, where the prequential estimate is typically several times larger than the requential estimate.

B.3 A Solvable Model Using Scaling Laws

In this section, we present a simplified analytical model from combining neural scaling laws with prequential coding to gain insight into how epiplexity and compute-optimal hyperparameters typically vary with compute and dataset size, along with their asymptotic behaviors.

We adopt a standard scaling law for the loss as a function of model size N and training tokens D:

$$\mathcal{L}(N, D) = E + \left(\frac{N_0}{N} \right)^{\alpha} + \left(\frac{D_0}{D} \right)^{\beta}, \tag{52}$$

where E is the irreducible loss, N_0 and D_0 are scaling constants, and $0 < \alpha, \beta < 1$ are the scaling exponents. The total compute for training and evaluating on $\mathcal{D}$ test tokens is $T = 6ND + 2N\mathcal{D} = 2N(3D + \mathcal{D})$.

To simplify the analysis, we work in natural units: $n = N/N_0$, $d = D/D_0$, $\delta = \mathcal{D}/D_0$, and $t = T/(2N_0 D_0)$. The loss becomes $\mathcal{L}(n, d) = E + n^{-\alpha} + d^{-\beta}$, and the compute constraint simplifies to $t = n(3d + \delta)$.

Two-part code length. The two-part code $\mathrm{P}_{\mathrm{tot}}$ consists of the model description and the data encoded using the model. The data code length on the test set is $\delta D_0 \cdot \mathcal{L}(n, d)$.

For the model description length, we use the prequential estimate from Equation (8), which corresponds to the area under the loss curve above the final loss[5]:

$$
\begin{aligned}
|\mathrm{P}_{\mathrm{preq}}| &= \sum_{i=1}^{D} \left[\left(\frac{i}{D_0} \right)^{-\beta} - \left(\frac{D}{D_0} \right)^{-\beta} \right] \\
&= \int_1^D \left[\left(\frac{u}{D_0} \right)^{-\beta} - \left(\frac{D}{D_0} \right)^{-\beta} \right] du + O(1),
\end{aligned} \tag{53}
$$

where the $O(1)$ term remains bounded as $D \to \infty$. Evaluating the integral and dropping $O(1)$ terms, we obtain the expression valid for large D:

$$|\mathrm{P}_{\mathrm{preq}}| = \frac{\beta}{1 - \beta} D_0 \, d^{1-\beta}. \tag{54}$$

---

5. We start the sum and integral at 1 to avoid the singularity at 0, which is an artifact of the scaling law as it typically only holds for large D.

Optimality condition. Dropping the constant term $\delta D_0 E$ from the two-part code length and dividing by D_0, we seek to minimize

$$f(n, d) = \frac{\beta}{1 - \beta} d^{1-\beta} + \delta(n^{-\alpha} + d^{-\beta}) \tag{55}$$

subject to $t = n(3d + \delta)$.

Solution. Eliminating n using the constraint $n = t/(3d + \delta)$, we obtain a one-dimensional optimization problem in d. Setting the derivative to zero and simplifying, the optimal $d^\star(t)$ satisfies

$$\beta d^{-\beta-1}(\delta - d) = 3\alpha\delta\, t^{-\alpha}(3d + \delta)^{\alpha-1}, \tag{56}$$

with the corresponding optimal model size given by

$$n^\star(t) = \frac{t}{3d^\star(t) + \delta}. \tag{57}$$

While Equation (56) does not admit a simple closed-form solution in general, we can extract the asymptotic behavior in the large- and small-compute regimes.

Large-compute regime $(t \to \infty)$. As t grows, the right-hand side of Equation (56) scales as $t^{-\alpha} \to 0$. For the equation to remain balanced, we require $\delta - d \to 0$, i.e., $d^\star(t) \to \delta$. The leading-order scaling is therefore:

$$d^\star(t) = \delta - \Theta(t^{-\alpha}), \qquad n^\star(t) \sim \frac{t}{4\delta}. \tag{58}$$

In this regime, the optimal training set size saturates at the test set size δ, while the model size grows linearly with compute. Correspondingly, the epiplexity saturates to

$$S_\infty(X) = \frac{\beta}{1 - \beta} D_0\, \delta^{1-\beta} = \frac{\beta}{1 - \beta} D_0^\beta \mathcal{D}^{1-\beta}. \tag{59}$$

For the entropy, we have $(n^\star)^{-\alpha} \to 0$ while $(d^\star)^{-\beta} \to \delta^{-\beta}$, so

$$H_\infty(X) = \mathcal{D}(E + \delta^{-\beta}) = \mathcal{D}E + D_0^\beta \mathcal{D}^{1-\beta}. \tag{60}$$

The entropy approaches the irreducible entropy $\mathcal{D}E$ plus a residual term from finite training data that scales sublinearly with the test set size, meaning that the achieved per-token loss is $E + O(\mathcal{D}^{-\beta})$.

Small-compute regime $(d^\star \ll \delta)$. When compute is limited such that $d \ll \delta$, we approximate $\delta - d \approx \delta$ and $(3d + \delta)^{\alpha-1} \approx \delta^{\alpha-1}$. Substituting into Equation (56) and solving for d gives

$$d^\star(t) = \left(\frac{\beta}{3\alpha}\right)^{\frac{1}{\beta+1}} t^{\frac{\alpha}{\beta+1}} \delta^{\frac{1-\alpha}{\beta+1}}. \tag{61}$$

Since $3d^\star \ll \delta$ in this regime, the optimal model size is

$$n^\star(t) \approx \frac{t}{\delta}. \tag{62}$$

Here, the model size is constrained by the need to evaluate on δ tokens, and the optimal training set size grows sublinearly with compute as $d^\star \propto t^{\alpha/(\beta+1)}$. The epiplexity in this regime scales as

$$S_T(X) = \frac{\beta}{1 - \beta} D_0\, (d^\star)^{1-\beta} \propto T^{\frac{\alpha(1-\beta)}{\beta+1}}, \tag{63}$$

growing sublinearly with compute.

For the entropy, both the model and data contributions are significant. The model contribution scales as

$$\mathcal{D}(n^\star)^{-\alpha} = \mathcal{D}\left(\frac{\delta}{t}\right)^\alpha \propto T^{-\alpha}, \tag{64}$$

while the data contribution scales as

$$\mathcal{D}(d^\star)^{-\beta} \propto T^{-\frac{\alpha\beta}{\beta+1}}. \tag{65}$$

Since $\alpha\beta/(\beta+1) < \alpha$, the data term decays more slowly and dominates for larger t within this regime. The entropy above the irreducible level is thus

$$\mathrm{H}_T(X) - \mathcal{D}E \propto T^{-\frac{\alpha\beta}{\beta+1}}, \tag{66}$$

decaying as a power law with compute.

For typical scaling exponents (e.g., $\alpha \approx 0.34$ and $\beta \approx 0.28$ from Hoffmann et al. (2022)), the epiplexity grows as $\mathrm{S}_T \propto T^{0.19}$ and the entropy decays as $\mathrm{H}_T - \mathcal{D}E \propto T^{-0.07}$ in the small-compute regime.

B.4 How Epiplexity and Time-Bounded Entropy Scale with Compute and Dataset Size

In this section, we analyze how epiplexity and time-bounded entropy scale with compute budget and dataset size under natural assumptions about neural network training, without relying on specific functional forms for scaling laws. The goal is to provide some general intuitions for how these quantities are expected to vary as a function of the compute budget and dataset size. Section B.3 explicitly demonstrates using scaling laws and prequential coding that (1) epiplexity grows with both compute and dataset size, and (2) for a fixed X, epiplexity saturates to a finite value in the limit of infinite compute—specifically, to the amount of information acquired by an arbitrarily large model trained on a training set of the same size as the test set X, while time-bounded entropy decays to the loss achievable by an infinitely large model on this training set. Here, we show that similar or weaker statements hold more generally, requiring only a few natural assumptions about the effect of increasing model size N and training data size D. These assumptions capture typically observed regularities in deep learning, such as the smoothly diminishing returns in scaling only model size while holding training set size fixed, but they may fail to capture rare exceptions like grokking and sudden improvement in performance above certain compute thresholds (as in Section 5.3.2).

Denote the code length for an N-parameter model trained on D tokens as $|\mathrm{P}|(N, D)$, the per-token loss it achieves as $\mathcal{L}(N, D)$, the compute-optimal model size as $N^\star(T)$ and training data size as $D^\star(T)$, so that $\mathrm{S}_T(X) = |\mathrm{P}|(N^\star(T), D^\star(T))$ and $\mathrm{H}_T(X) = \mathcal{D}\mathcal{L}(N^\star(T), D^\star(T))$. We establish the following results as we vary T and $\mathcal{D} = |X|$, fixing the distribution of X_i (only the dataset size changes):

- **Monotonicity of $N^\star(T)$, $D^\star(T)$, $\mathrm{S}_T(X)$, and $\mathrm{H}_T(X)$ (Section B.4.1):** Under natural assumptions on the effect of increasing N and D, the compute-optimal model size $N^\star(T)$ and training data size $D^\star(T)$ are both increasing in the compute budget T. As a result, epiplexity typically grows with T while time-bounded entropy typically decreases with T.

- **Monotonicity of $\mathrm{S}_\infty(X)$ and $\mathrm{H}_\infty(X)$ in $\mathcal{D}$ (Section B.4.2):** In the infinite-compute limit, epiplexity $\mathrm{S}_\infty(X)$ is nondecreasing in $\mathcal{D} = |X|$, while the per-token time-bounded entropy $h_\infty(X_\mathcal{D}) := \mathrm{H}_\infty(X_\mathcal{D})/\mathcal{D}$ is nonincreasing in $\mathcal{D}$.

- $D^\star(T)$ **generally approaches** $\mathcal{D}$ **in prequential coding** (Section B.4.3): For prequential coding, the compute-optimal training set size satisfies $D^\star(T) \to \mathcal{D}$ as $T \to \infty$, where $\mathcal{D}$ is the test set size, without assuming the scaling law form. Combined with monotonicity of $D^\star(T)$, this implies $D^\star(T) \uparrow \mathcal{D}$ from below.

B.4.1 MONOTONICITY OF $N^*(T)$, $D^*(T)$, $\mathrm{S}_T(X)$, AND $\mathrm{H}_T(X)$

The following theorem shows that the compute-optimal model size and training data size are both monotonically increasing in the compute budget under natural assumptions.

Theorem 30 (Monotone growth of compute-optimal N and D) *Define the effective data $\widetilde{D} = 6D + 2\mathcal{D}$, so that the compute constraint becomes $T = N\widetilde{D}$. Let $J(N, \widetilde{D})$ denote the two-part code length as a function of model size N and effective data $\widetilde{D}$, and assume J is twice continuously differentiable. Consider the constrained MDL problem*

$$\min_{N>0,\, \widetilde{D} \geq 2\mathcal{D}} J(N, \widetilde{D}) \qquad s.t. \qquad N\widetilde{D} = T. \tag{67}$$

Assume that for each T in the regime of interest there is a unique interior optimizer $(N^\star(T), \widetilde{D}^\star(T))$ with $\widetilde{D}^\star(T) > 2\mathcal{D}$ and $N^\star(T)\widetilde{D}^\star(T) = T$.
Work in logarithmic coordinates $\mu := \log N$ and $\nu := \log \widetilde{D}$, and by slight abuse of notation write $J(\mu, \nu) = J(e^\mu, e^\nu)$. Assume that for all such T, the following conditions hold at the corresponding optimum $(\mu^\star(T), \nu^\star(T))$:

1. ***Complementarity (larger models are more sample-efficient):***

$$\frac{\partial^2 J}{\partial \mu \partial \nu} \leq 0. \tag{68}$$

2. ***Diminishing returns in model size (in log coordinates):***

$$\frac{\partial^2 J}{\partial \mu^2} > 0. \tag{69}$$

3. ***Diminishing returns in effective data (in log coordinates):***

$$\frac{\partial^2 J}{\partial \nu^2} > 0. \tag{70}$$

Then both compute-optimal choices are strictly increasing functions of T:

$$T_2 > T_1 \quad \Longrightarrow \quad N^\star(T_2) > N^\star(T_1) \quad and \quad \widetilde{D}^\star(T_2) > \widetilde{D}^\star(T_1). \tag{71}$$

Proof Work in logarithmic coordinates

$$\mu := \log N, \qquad \nu := \log \widetilde{D}, \qquad \tau := \log T. \tag{72}$$

The compute constraint $N\widetilde{D} = T$ becomes the affine constraint

$$\mu + \nu = \tau \qquad \Longleftrightarrow \qquad \nu = \tau - \mu. \tag{73}$$

By slight abuse of notation, write $J(\mu, \nu) := J(e^\mu, e^\nu)$ and denote its partial derivatives by $J_\mu, J_\nu, J_{\mu\mu}, J_{\nu\nu}, J_{\mu\nu}$, etc., all taken with respect to the log-coordinates (μ, ν).

Define the *restricted objective* along the compute frontier by

$$f(\mu, \tau) := J(\mu, \tau - \mu). \tag{74}$$

For each τ in the regime of interest, let $\mu^\star(\tau)$ denote the unique interior minimizer of $f(\cdot, \tau)$, and set $\nu^\star(\tau) := \tau - \mu^\star(\tau)$.

Holding τ fixed and differentiating f with respect to μ gives

$$\begin{aligned}
f_\mu(\mu, \tau) &= \frac{\partial}{\partial \mu} J(\mu, \tau - \mu) \\
&= J_\mu(\mu, \nu) + J_\nu(\mu, \nu) \frac{\partial}{\partial \mu}(\tau - \mu) \\
&= J_\mu(\mu, \nu) - J_\nu(\mu, \nu),
\end{aligned} \tag{75}$$

where $\nu = \tau - \mu$. The optimality condition for $\mu^\star(\tau)$ is therefore

$$f_\mu(\mu^\star(\tau), \tau) = 0 \qquad \Longleftrightarrow \qquad J_\mu(\mu^\star(\tau), \nu^\star(\tau)) = J_\nu(\mu^\star(\tau), \nu^\star(\tau)). \tag{76}$$

Differentiating the identity $f_\mu(\mu^\star(\tau), \tau) = 0$ with respect to τ yields

$$0 = \frac{d}{d\tau} f_\mu(\mu^\star(\tau), \tau) = f_{\mu\mu}(\mu^\star(\tau), \tau) \frac{d\mu^\star}{d\tau} + f_{\mu\tau}(\mu^\star(\tau), \tau). \tag{77}$$

Assuming $f_{\mu\mu}(\mu^\star(\tau), \tau) \neq 0$ (verified below), we obtain

$$\frac{d\mu^\star}{d\tau} = -\frac{f_{\mu\tau}}{f_{\mu\mu}} \quad \text{evaluated at } (\mu, \tau) = (\mu^\star(\tau), \tau). \tag{78}$$

We now express $f_{\mu\tau}$ and $f_{\mu\mu}$ in terms of second partial derivatives of J. From (75) and the chain rule, using $\partial_\tau(\tau - \mu) = 1$,

$$\begin{aligned}
f_{\mu\tau}(\mu, \tau) &= \frac{\partial}{\partial \tau}(J_\mu(\mu, \nu) - J_\nu(\mu, \nu)) \\
&= J_{\mu\nu}(\mu, \nu) \frac{\partial \nu}{\partial \tau} - J_{\nu\nu}(\mu, \nu) \frac{\partial \nu}{\partial \tau} \\
&= J_{\mu\nu}(\mu, \nu) - J_{\nu\nu}(\mu, \nu),
\end{aligned} \tag{79}$$

with $\nu = \tau - \mu$. Similarly, differentiating (75) with respect to μ while holding τ fixed, and using $\partial_\mu(\tau - \mu) = -1$ together with symmetry $J_{\nu\mu} = J_{\mu\nu}$, yields

$$\begin{aligned}
f_{\mu\mu}(\mu, \tau) &= \frac{\partial}{\partial \mu}(J_\mu(\mu, \nu) - J_\nu(\mu, \nu)) \\
&= \left(J_{\mu\mu}(\mu, \nu) + J_{\mu\nu}(\mu, \nu) \frac{\partial \nu}{\partial \mu} \right) - \left(J_{\nu\mu}(\mu, \nu) + J_{\nu\nu}(\mu, \nu) \frac{\partial \nu}{\partial \mu} \right) \\
&= (J_{\mu\mu} - J_{\mu\nu}) - (J_{\mu\nu} - J_{\nu\nu}) \\
&= J_{\mu\mu}(\mu, \nu) + J_{\nu\nu}(\mu, \nu) - 2J_{\mu\nu}(\mu, \nu).
\end{aligned} \tag{80}$$

Substituting (79)–(80) into (78) gives

$$\frac{d\mu^\star}{d\tau} = -\frac{J_{\mu\nu} - J_{\nu\nu}}{J_{\mu\mu} + J_{\nu\nu} - 2J_{\mu\nu}} = \frac{J_{\nu\nu} - J_{\mu\nu}}{J_{\mu\mu} + J_{\nu\nu} - 2J_{\mu\nu}}, \tag{81}$$

with all second partial derivatives of J evaluated at $(\mu, \nu) = (\mu^\star(\tau), \nu^\star(\tau))$.

By the assumptions $J_{\nu\nu} > 0$ and $J_{\mu\nu} \leq 0$ at the optimum, the numerator in (81) satisfies $J_{\nu\nu} - J_{\mu\nu} > 0$. By the assumptions $J_{\mu\mu} > 0$, $J_{\nu\nu} > 0$, and $J_{\mu\nu} \leq 0$, the denominator satisfies $J_{\mu\mu} + J_{\nu\nu} - 2J_{\mu\nu} > 0$. Hence

$$\frac{d\mu^\star}{d\tau} > 0. \tag{82}$$

Since $\nu^\star(\tau) = \tau - \mu^\star(\tau)$, we also have

$$\frac{d\nu^\star}{d\tau} = 1 - \frac{d\mu^\star}{d\tau} = \frac{J_{\mu\mu} - J_{\mu\nu}}{J_{\mu\mu} + J_{\nu\nu} - 2J_{\mu\nu}} > 0, \tag{83}$$

where positivity follows from $J_{\mu\mu} > 0$ and $J_{\mu\nu} \leq 0$ together with the same positive denominator.

Finally, $N^\star(T) = \exp(\mu^\star(\log T))$ and $\widetilde{D}^\star(T) = \exp(\nu^\star(\log T))$, so $d\mu^\star/d\tau > 0$ and $d\nu^\star/d\tau > 0$ imply that both $N^\star(T)$ and $\widetilde{D}^\star(T)$ are strictly increasing functions of T. ∎

Empirical plausibility of the assumptions. The three conditions in Theorem 30 reflect well-documented empirical phenomena in deep learning. The complementarity condition $\partial^2 J / \partial\mu\partial\nu \leq 0$ captures the observation that larger models are more sample-efficient: increasing model size leads to faster learning (Kaplan et al., 2020; Yang et al., 2022), which leads to a faster decrease in both the model description length and data code length (final loss), and thus $\partial J / \partial\nu$ should decrease with μ. The diminishing returns conditions $\partial^2 J / \partial\mu^2 > 0$ and $\partial^2 J / \partial\nu^2 > 0$ simply state that there is diminishing return in successive doubling of the model size or training data size, holding the other quantity fixed.

Asymptotic growth of S_T and monotone decay of H_T. The monotone growth of the compute-optimal $N^\star(T)$ and $D^\star(T)$ does not by itself imply that $\mathrm{S}_T(X) := |\mathrm{P}|(N^\star(T), D^\star(T))$ is monotone for all T. Intuitively, while we expect the model description length $|\mathrm{P}|(N, D)$ to grow with D, it need not increase with N: larger models can be more sample-efficient, which may reduce the effective complexity of the learned predictor under some coding schemes. However, one should still expect $\mathrm{S}_T(X)$ to grow with T, at least asymptotically, if we assume (1) the compute-optimal model size diverges while the optimal training horizon converges, as in the scaling-law model of Section B.3, and (2) the existence of infinite-model-size limits of the training dynamics.

That is, assume that along the compute-optimal path,

$$N^\star(T) \to \infty \qquad \text{and} \qquad D^\star(T) \to D_\infty < \infty \qquad \text{as } T \to \infty. \tag{84}$$

Assume moreover that for bounded training horizons, the description length admits a well-defined infinite-model-size limit: there exists a function $P_\infty(D)$ such that for each fixed D in a neighborhood of $[0, D_\infty]$,

$$|\mathrm{P}|(N, D) \to P_\infty(D) \qquad \text{as } N \to \infty. \tag{85}$$

This assumption is motivated by the existence of infinite-width and depth limits of neural networks under appropriate parameterizations (Yang and Littwin, 2023; Dey et al., 2025), where scalar quantities such as loss and teacher–student KL divergence that determine $|\mathrm{P}|(N, D)$ admit stable large-model limits for bounded training durations. Under these conditions, any non-monotonic dependence of $|\mathrm{P}|$ on N is a finite-model effect; once $N^\star(T)$ is large enough, $|\mathrm{P}|(N^\star(T), D^\star(T))$ is well-approximated by the limiting curve $P_\infty(D^\star(T))$. Since $D^\star(T)$ is monotone increasing and

54

convergent under our earlier assumptions, the large-T behavior of $S_T(X)$ is therefore governed primarily by the behavior of $D^\star(T)$ alone, which we have shown is increasing with T, so one expects $S_T(X)$ to increase at large T as $P_\infty(D)$ should increase with D.

For the entropy term $H_T(X) := \mathcal{D}\,\mathcal{L}(N^\star(T), D^\star(T))$, the conclusion is simpler and does not require taking $N \to \infty$. Assume only that the loss $\mathcal{L}(N, D)$ is nonincreasing in both N and D (more data and parameters cannot make the loss worse). Since $N^\star(T)$ and $D^\star(T)$ are increasing in T, we have

$$T_2 > T_1 \quad \implies \quad \mathcal{L}(N^\star(T_2), D^\star(T_2)) \leq \mathcal{L}(N^\star(T_1), D^\star(T_1)), \tag{86}$$

and therefore $H_T(X)$ is nonincreasing in T. In particular, whenever $H_T(X)$ has a finite large-compute limit $H_\infty(X)$, it approaches this limit from above.

B.4.2 MONOTONICITY OF $S_\infty(X)$ AND $H_\infty(X)$ IN $\mathcal{D}$

We now show that epiplexity and time-bounded entropy (after appropriate normalization) in the infinite-compute limit are monotonic in the test set size $\mathcal{D} = |X|$, regardless of the coding scheme.

Fix a dataset $X_\mathcal{D}$ of length $\mathcal{D}$ tokens. For a two-part code of the form

$$J(N, D; \mathcal{D}) = |\mathrm{P}|(N, D) + \mathcal{D}\,\mathcal{L}(N, D), \tag{87}$$

let $(N_T^\star(\mathcal{D}), D_T^\star(\mathcal{D}))$ denote the compute-optimal choices. We write

$$S_T(X_\mathcal{D}) := |\mathrm{P}|(N_T^\star(\mathcal{D}), D_T^\star(\mathcal{D})), \tag{88}$$
$$H_T(X_\mathcal{D}) := \mathcal{D}\,\mathcal{L}(N_T^\star(\mathcal{D}), D_T^\star(\mathcal{D})), \tag{89}$$
$$h_T(\mathcal{D}) := \frac{H_T(X_\mathcal{D})}{\mathcal{D}} = \mathcal{L}(N_T^\star(\mathcal{D}), D_T^\star(\mathcal{D})). \tag{90}$$

In the infinite-compute limit $T \to \infty$, the compute constraint becomes irrelevant, so the limiting quantities coincide with the optimum of the unconstrained problem

$$(N_\infty^\star(\mathcal{D}), D_\infty^\star(\mathcal{D})) = \arg \min_{N > 0,\, D \geq 0} |\mathrm{P}|(N, D) + \mathcal{D}\,\mathcal{L}(N, D). \tag{91}$$

Thus

$$S_\infty(X_\mathcal{D}) = |\mathrm{P}|(N_\infty^\star, D_\infty^\star), \qquad h_\infty(X_\mathcal{D}) = \mathcal{L}(N_\infty^\star, D_\infty^\star). \tag{92}$$

We claim that $S_\infty(X_\mathcal{D})$ is nondecreasing in $\mathcal{D}$, and $H_\infty(X_\mathcal{D})$ is nonincreasing in $\mathcal{D}$, assuming that for each $\mathcal{D} > 0$ the unconstrained problem (91) admits at least one minimizer.

To see this, fix $\mathcal{D}_2 > \mathcal{D}_1$ and let (N_i, D_i) be minimizers of (91) at $\mathcal{D} = \mathcal{D}_i$. Write $P_i := |\mathrm{P}|(N_i, D_i)$ and $L_i := \mathcal{L}(N_i, D_i)$. Optimality of (N_2, D_2) at $\mathcal{D}_2$ implies

$$P_2 + \mathcal{D}_2 L_2 \leq P_1 + \mathcal{D}_2 L_1. \tag{93}$$

Optimality of (N_1, D_1) at $\mathcal{D}_1$ implies

$$P_1 + \mathcal{D}_1 L_1 \leq P_2 + \mathcal{D}_1 L_2. \tag{94}$$

Adding (93) and (94) gives

$$(P_2 + \mathcal{D}_2 L_2) + (P_1 + \mathcal{D}_1 L_1) \leq (P_1 + \mathcal{D}_2 L_1) + (P_2 + \mathcal{D}_1 L_2)$$
$$\mathcal{D}_2 L_2 + \mathcal{D}_1 L_1 \leq \mathcal{D}_2 L_1 + \mathcal{D}_1 L_2$$
$$(\mathcal{D}_2 - \mathcal{D}_1)(L_2 - L_1) \leq 0, \tag{95}$$

hence $L_2 \leq L_1$ since $\mathcal{D}_2 > \mathcal{D}_1$, i.e., the achieved loss $h_\infty(X_\mathcal{D})$ is nonincreasing in $\mathcal{D}$. Substituting $L_2 \leq L_1$ back into (94) yields $P_2 \geq P_1$, i.e., $S_\infty(X_\mathcal{D})$ is nondecreasing in $\mathcal{D}$.

B.4.3 $D^\star(T)$ Generally Approaches $\mathcal{D}$ in Prequential Coding

We now show that the compute-optimal training set size for prequential coding generically saturates at $D = \mathcal{D}$ as $T \to \infty$, without assuming specific scaling laws.

In continuous time, the prequential model description length is the area above the final loss:

$$|\mathrm{P}_{\mathrm{preq}}(N, D)| := \int_0^D \left(\mathcal{L}(N, u) - \mathcal{L}(N, D) \right) du. \tag{96}$$

The corresponding two-part code length for a test set of size $\mathcal{D}$ is

$$
\begin{aligned}
J_{\mathrm{preq}}(N, D; \mathcal{D}) &= |\mathrm{P}_{\mathrm{preq}}(N, D)| + \mathcal{D}\, \mathcal{L}(N, D) \\
&= \int_0^D \mathcal{L}(N, u)\, du + (\mathcal{D} - D)\, \mathcal{L}(N, D).
\end{aligned}
\tag{97}
$$

We express N in terms of D for fixed T using the constraint $6ND + 2N\mathcal{D} = T$:

$$N_T(D) = \frac{T}{6D + 2\mathcal{D}}. \tag{98}$$

Large-compute limit. Assume: (i) $\mathcal{L}(N, D)$ is nonincreasing in N and admits a pointwise infinite-model-size limit $\mathcal{L}_\infty(D) := \lim_{N \to \infty} \mathcal{L}(N, D)$;[6] (ii) $\mathcal{L}_\infty$ is continuously differentiable and strictly decreasing, i.e., $\mathcal{L}'_\infty(D) < 0$. Along the compute frontier (98), for any fixed D we have $N_T(D) \to \infty$ as $T \to \infty$, hence

$$J_{\mathrm{preq}}(N_T(D), D; \mathcal{D}) \to J_\infty(D) := \int_0^D \mathcal{L}_\infty(u)\, du + (\mathcal{D} - D)\, \mathcal{L}_\infty(D). \tag{99}$$

Differentiating gives

$$J'_\infty(D) = (\mathcal{D} - D)\, \mathcal{L}'_\infty(D). \tag{100}$$

Since $\mathcal{L}'_\infty(D) < 0$, we have $J'_\infty(D) < 0$ for $D < \mathcal{D}$ and $J'_\infty(D) > 0$ for $D > \mathcal{D}$. Thus J_∞ is uniquely minimized at $D = \mathcal{D}$, implying

$$D^\star(T) \to \mathcal{D} \qquad \text{as } T \to \infty. \tag{101}$$

Approach from below and linear growth of $N^\star(T)$. By Theorem 30, under the complementarity and diminishing-returns assumptions, the compute-optimal training set size $D^\star(T)$ is strictly increasing in T. Combined with the convergence $D^\star(T) \to \mathcal{D}$, this yields $D^\star(T) \uparrow \mathcal{D}$, i.e., $D^\star(T)$ approaches $\mathcal{D}$ from below. Finally, since $N^\star(T) = N_T(D^\star(T))$,

$$N^\star(T) = \frac{T}{6D^\star(T) + 2\mathcal{D}} \sim \frac{T}{8\mathcal{D}}, \tag{102}$$

so the compute-optimal model size grows linearly with T in the large-compute regime.

---

6. This limit provably exists under μP, but is a reasonable assumption in general as it simply asserts diminishing returns in scaling model size without increasing data.

Appendix C. Experiment Details

Unless otherwise stated, we use the GPT-2 (Radford et al., 2019) transformer architecture trained with Adam optimizer. In experiments where we vary the model size, we tune the base learning rate on a small model and transfer it to larger models using using μP (Yang et al., 2022) and CompleteP (Dey et al., 2025). In μP, the per-layer learning rate is base learning rate divided by the input dimension, so our reported base learning rate is larger than typical learning rates used for Adam. The hyperparameters presented below are shared between the teacher and the student for requential coding (width, depth, learning rate, EMA time scale, etc.). As described in Section B.1, the EMA for the teacher is used only for producing the distillation target and does not alter the raw teacher training dynamics, while the EMA for the student model does alter its training dynamics and is used to replace a decaying learning rate schedule.

C.1 ECA

In Figure 3, we train the transformer to predict Y given X where X is the initial state with a state size of 64 cells and Y is obtained by evolving X for 48 steps. We apply a burnin period of 1000 steps for sampling the initial state X to eliminate the less uninteresting transient dynamics from random initialization. That is X is obtained by evolving the ECA on Z for 1000 steps where Z is a uniform random initial state. For each rule, we train models with width (embedding dimension) $\in \{16, 32, 64, 128, 256, 512\}$ and depth (number of transformer blocks) $\in \{1, 2, 4, 6, 9\}$. We train both teacher and student using batches of 1536 sequences (each an (X, Y) pair), a base learning rate of 0.03 with 100 warmup steps, and an EMA time scale of 50 steps (half-life divided by $\ln(2)$). We did not set a max teacher-student KL as the student smoothly tracks the teacher throughout training. The epiplexity and time-bounded entropy is estimated for a test set of size $\mathcal{D} = 100\text{M}$ tokens (counting Y only).

C.2 Easy induction

For this task, we use a sequence length of $n = 512$ (as described in Section 5.3.1). The model has 3 layers and a width of 128, and is trained with a learning rate of 0.03 and a batch size of 384 sequences for 3000 steps with 15 warmup steps and an EMA time scale of 50 steps. We found further increasing the model size led to negligible improvement in the loss, and Figure 5c shows that the model has nearly converged by the end of training to the theoretical minimum loss, so there is no need to further increase the training data. As a result, we expect the epiplexity $S_T(X)$ to stabilize as T and $\mathcal{D} = |X|$ increases (in the relevant regime where T is still much less than what is required for implementing the brute-force solution that enumerates all possible combinations of hidden entries in the transition matrix), and our estimated epiplexity approximates this stabilized value.

C.3 Hard induction

We modify the ECA experiment in Section C.1 to remove the first $h \in \{0, 1, \ldots, 5\}$ bits in X when fed to the model as input. We use a state size of 32, batch size of 1536 sequences, learning of 0.03, EMA time scale of 100 steps. We set the max KL threshold between the teacher and student as 0.03 (nats per token). To construct a forward function that is hard to invert, we use rule 30 iterated for 4 steps. We train models with 3 layers and width 256 for 20000. Further increasing model size or training data led to no improvement in the loss. As Figure 5b shows, the models converge by the end of training (the loss curves shown are for the student models, but the teacher models also converge) to the theoretical minimum values. Therefore, like the case for Section C.2, we expect the epiplexity

$S_T(X)$ to stabilize as T and $\mathcal{D} = |X|$ increases, at least in the relevant regime where T is still much less than what is required for implementing the brute-force solution that enumerates all possible combinations of hidden bits, and our estimated epiplexity approximates this stabilized value.

C.4 Chess

We train models of varying sizes from 1M to 160M parameters with depth between 3 and 24 layers. The base learning rate is set to 2 and the batch size is 256, with a sequence length of 512. We set the EMA time scale to 50 steps and max KL to 0.1 nats per token. We use character-level tokenization. The teacher models are trained for 5B tokens in total, and the student models are trained for slightly more due to hitting the max KL threshold during training. The test set size is set to 5B tokens.

Pre-Training Data. We use the Lichess dataset available on Hugging Face at `https://huggingface.co/datasets/Lichess/standard-chess-games` as pre-training data, formatted as either "<board>|<moves>" or "<moves>|<board>", where moves are in algebraic chess notation and board is the final board state in FEN notation. We use a slightly more concise version of the algebraic notation to further compress the move sequence. An example input where the board appears last is:

```
e4,e5;Nf3,Nc6;Bb5,a6;Ba4,Nf6;O-O,Be7;Re1,b5;Bb3,d6;c3,O-O;h3,Nb8;d4,Nbd7;
|r1bq1rk1/2pnbppp/p2p1n2/1p2p3/3PP3/1BP2N1P/PP3PP1/RNBQR1K1 w - - 0 10
```

For downstream evaluation, we evaluate performance on the following two datasets after fine-tuning on 50k examples for a 10M-parameter model with depth 24. We report accuracy under greedy decoding at zero temperature.

Chess Puzzles. We use puzzles from the Lichess puzzle database available at `https://huggingface.co/datasets/EleutherAI/lichess-puzzles`, filtering for puzzles with difficulty rating above 2000. The task is to predict the correct move sequence given the game context. Puzzles are formatted as move sequences where the model must predict the next optimal move, following (Burns et al., 2023), with only the target moves included in the loss computation via masking. This tests the model's ability to recognize tactical patterns and calculate forced sequences.

Centipawn Evaluation. We evaluate position understanding using the Lichess chess position evaluations dataset at `https://huggingface.co/datasets/Lichess/chess-position-evaluations`, where models classify positions into 9 evaluation buckets based on Stockfish centipawn (cp) scores: class 0 (≤ -800cp), class 1 (-800 to -400cp), class 2 (-400 to -200cp), class 3 (-200 to -50cp), class 4 (-50 to $+50$cp), class 5 ($+50$ to $+200$cp), class 6 ($+200$ to $+400$cp), class 7 ($+400$ to $+800$cp), and class 8 ($\geq +800$cp). Examples are formatted as "<board>|<class>" where the model predicts the evaluation class, with mate positions assigned to the extreme classes (0 or 8). Loss during fine-tuning is computed only for predicting the class.

C.5 OpenWebText

We use the OpenWebText dataset at `https://huggingface.co/datasets/Skylion007/openwebtext`, keeping only documents containing only 96 common alphanumeric symbols, and apply character-level tokenization. The setup is otherwise identical to the chess experiment (Section C.4).

C.6 CIFAR-5M

We use the CIFAR-5M dataset at `https://github.com/preetum/cifar5m`. We convert the $32{\times}32{\times}3$ images to greyscale and flatten to a 1D sequence of 1024 in raster-scan order. The vocabulary is the set of pixel intensities $\{0,\ldots,255\}$. The setup is otherwise identical to the chess experiment (Section C.4).

C.7 Prequential vs Requential Comparison

ECA. The ECA experiment include rules $\{0, 32, 4, 15, 22, 30, 41, 54, 106, 110\}$, covering all 4 classes. We train models with width $\in \{16, 32, 64, 128\}$ and depth $\in \{1, 2, 3\}$ up to 10000 steps. We use a base learning rate of 0.03 and batch size of 384. Other hyperparameters are identical to Section C.1. We set $\mathcal{D} = 250\text{M}$ tokens. For each rule, we report the maximum epiplexity over the resulting compute range.

Induction. Both the easy and hard induction results directly come from the experiments in Section 5.3.1. As explained in Section C.2 and Section C.3, the compute budget T and test set size $\mathcal{D}$ need not be precisely specified for these two tasks as the epiplexity stabilizes as T and $\mathcal{D}$ increase due to the convergent training dynamics.

Natural data. We report the estimated epiplexity on each dataset at the maximum tested compute budget as described in Section C.4, Section C.5, and Section C.6.

C.8 ECA Emergence

We modify the setup in Section C.1 to include models that predict intermediate states and the final state rather than the final state directly. Let $X^{(0)}$ denote the initial ECA state, and $X^{(s)}$ denote it evolved for s steps. For an ℓ-loop model, we train the model to predict $(X^{(\Delta)}, X^{(2\Delta)}, \ldots, X^{(t)})$ instead of $X^{(t)}$ only, where $\Delta = t/\ell$. Its marginal probability on the final state is lower-bounded by its joint probability on the ground truth trajectory:

$$P(X^{(t)}) = \sum_{X'^{(\Delta)},\ldots,X'^{(t-\Delta)}} P(X'^{(\Delta)},\ldots,X'^{(t-\Delta)},X^{(t)}) \tag{103}$$

So we upper-bound its NLL as

$$\log \frac{1}{P(X^{(t)})} \leq \log \frac{1}{P(X^{(\Delta)},\ldots,X^{(t)})}$$

$$= \sum_{k=1}^{\ell} \log \frac{1}{P(X^{(k\Delta)} \mid X^{((k-1)\Delta)},\ldots,X^{(\Delta)})}, \tag{104}$$

We account for the intermediate tokens when computing the time bound and the code length (they contribute to the model code length as well as the data entropy code length). In the experiment, we set the ECA steps to $t = 64$. We train models with width $\{16, 32, 64, 128\}$, depth $\in \{1, 2, 4, 8, 16, 32\}$, and number of loops $\ell \in \{1, 2, 4, 8, 16\}$. We found $\ell \in \{2, 4, 8\}$ has no advantage over the non-looped model ($\ell = 1$) in terms of the two-part code, only $\ell = 16$ does. We therefore refer to $\ell = 1$ as non-looped and $\ell = 16$ as looped. The fact that a small $\ell > 1$ is not helpful is likely because the overhead of encoding and generating intermediate states exceeds the savings from only slightly simplifying each prediction step, as the per-step prediction horizon is still significant. We train all models with a base learning rate of 0.06, batch size of 147456 tokens, warmup of 100 steps, and EMA time scale of 50 steps. We did not set a max teacher-student KL. The test set size is set to $\mathcal{D} = 100\text{M}$ final state tokens.

C.9 Scaling Laws

We estimate epiplexity and time-bounded entropy using the expressions derived in Section B.3 for prequential coding using existing scaling laws for $\mathcal{L}(N, D)$. We solve for the optimal training tokens $D^\star(T)$ as a function of compute using root finding for Equation (56). For language, we use the Chinchilla scaling laws from Hoffmann et al. (2022), which were fit to total parameter counts. For all other modalities (images and video), we use the scaling laws from Henighan et al. (2020), which follow the methodology of Kaplan et al. (2020) and report non-embedding parameter counts. We correct these to use total parameters following Pearce and Song (2024), as described below.

Correcting for embedding parameters. The scaling laws in Kaplan et al. (2020) and Henighan et al. (2020) are reported in terms of non-embedding parameters $N_{\backslash E}$ and non-embedding compute $C_{\backslash E}$, excluding embedding and unembedding parameters. As shown by Pearce and Song (2024), this choice—combined with smaller model scales—accounts for much of the discrepancy between the Kaplan and Chinchilla scaling laws. Following their approach, we relate total parameters N to non-embedding parameters via

$$N = N_{\backslash E} + \omega N_{\backslash E}^{1/3}, \qquad \omega = (V + L_{\text{ctx}})\left(\frac{A}{12}\right)^{1/3}, \tag{105}$$

where V is the vocabulary size, L_{ctx} is the context length, and A is the aspect ratio (width/depth). We use $A = 5$ as Henighan et al. (2020) showed the optimal aspect ratio is around this value for non-language datasets. We generate points $(C_{\backslash E}, N_{\backslash E}, L)$ from the original scaling laws, convert to $(C, N, \mathcal{L})$ using this relation (with total compute as $C = C_{\backslash E} \cdot N/N_{\backslash E}$), and refit the power-law exponents and the irreducible loss.

Parameterization conversion. The scaling laws in Henighan et al. (2020) are reported in compute-centric form, expressing the optimal loss $L^\star(C) = (C/C_0)^{-\gamma} + E$ and optimal model size $N^\star(C) = (C/\hat{C})^\delta$ as functions of compute budget C. We convert these to the (N, D) parameterization used in this work:

$$\mathcal{L}(N, D) = \left(\frac{N}{N_0}\right)^{-\alpha} + \left(\frac{D}{D_0}\right)^{-\beta} + E, \tag{106}$$

where the exponents transform as $\alpha = \gamma/\delta$ and $\beta = \gamma/(1 - \delta)$, and the token scale is given by $D_0 = \frac{\hat{C}}{6} N_0^{\alpha/\beta} (\beta/\alpha)^{-1/\beta}$.

Corrected parameters. Table 1 presents the corrected scaling law parameters used in our final calculations.

Table 1: Final scaling law parameters used. Image and video domains from Henighan et al. (2020) are corrected for embedding parameters using aspect ratio $A = 5$ following (Pearce and Song, 2024); Chinchilla (language) from Hoffmann et al. (2022) was originally fit to total parameter counts and requires no correction. D_0 is measured in tokens and E is measured in nats.

Domain	α	β	N_0	D_0	E
Image 8×8	0.331	0.566	8.0×10^1	2.66×10^6	3.14
Image 16×16	0.307	0.820	2.8×10^2	8.94×10^7	2.68
Image 32×32	0.258	0.399	6.3×10^1	1.95×10^6	2.30
Image VQ 16×16	0.322	0.441	2.7×10^4	4.44×10^7	4.23
Image VQ 32×32	0.287	0.560	1.9×10^4	1.63×10^8	3.32
Video VQ 16^3	0.428	0.718	3.7×10^4	1.79×10^8	1.15
Language (Chinchilla)	0.339	0.285	4.91×10^7	1.49×10^9	1.69

Appendix D. RASP-L for Elementary Cellular Automata

Below we provide RASP-L code (Zhou et al., 2023) demonstrating how the evolution rule of an ECA can be implemented, providing evidence that the solution can be expressed within an autoregressive transformer model.

Listing 1: RASPL implementation of a cellular automaton evolution step

```python
from np_rasp import *

def int2bits(x, bits=8): # returns LSB first
    """ Helper function to generate fixed bitstring representing a number.
    Not RASP-L, can be assumed constant."""
    bits_str = bin(x)[2:].zfill(bits)
    return np.array(list(map(int,bits_str[::-1])),dtype=np.uint8)

sep = -1
sep2 = -2
def evolve_ca(x, rule):
    """ Function to autoregressively output produce the output of one step of the ECA
        rule. Problem encoded as x= --input state--,sep,sep2,--output state--.
    Rule: int (specifying the ECA)"""
    lookup = int2bits(rule, 8)
    in_input = 1 - has_seen(x, full(x, sep))
    in_input2 = 1 - has_seen(x, full(x, sep2))
    width = cumsum(in_input)   # only valid after sep
    idx = indices(x)
    circ_x = where(in_input, x, index_select(x, idx - width))
    prev = shift_right(x, 1)
    cprev = where(in_input2, prev, index_select(prev, idx - width))
    prev2 = shift_right(x, 2)
    nbhd = (prev2 << 2) + (cprev << 1) + circ_x
    shifted_nextstate = lookup[nbhd]
    to_select_idx = idx - width
    to_select_idx = where(to_select_idx < 3, idx, to_select_idx)
    outstate = index_select(shifted_nextstate, to_select_idx)
    return outstate
```

Appendix E. Cellular Automata and Game of Life

Elementary cellular automata Elementary cellular automata (ECA) (Wolfram and Gad-el Hak, 2003) are one-dimensional cellular automata defined on a finite or infinite line of cells, each in one of two states: 0 or 1. The system evolves in discrete time steps according to local rules: a cell's next state depends only on its current state and those of its two immediate neighbors, yielding $2^3 = 8$ possible neighborhood configurations. Since each configuration can map to either 0 or 1, there are $2^8 = 256$ possible rules, conventionally numbered 0–255 using Wolfram's notation, where the rule number's binary representation specifies the output for each neighborhood. Despite their simplicity, ECAs exhibit diverse behaviors ranging from trivial (e.g., Rule 0) to complex and chaotic (e.g., Rule 30), with Rule 54 proven to be Turing-complete. These systems serve as minimal models for studying emergence, computation, and the relationship between local rules and global behavior.

Conways Game of Life Conway's Game of Life (Gardner, 1970) is a cellular automaton defined on an infinite two-dimensional grid of cells, each in one of two states: alive (1) or dead (0). The system evolves in discrete time steps according to deterministic local rules: a cell's next state depends only on its current state and those of its eight neighbors. Specifically, a live cell survives if it has exactly 2 or 3 live neighbors (otherwise it dies), while a dead cell becomes alive if it has exactly 3 live neighbors (otherwise it remains dead). Despite the simplicity of these rules, the Game of Life exhibits
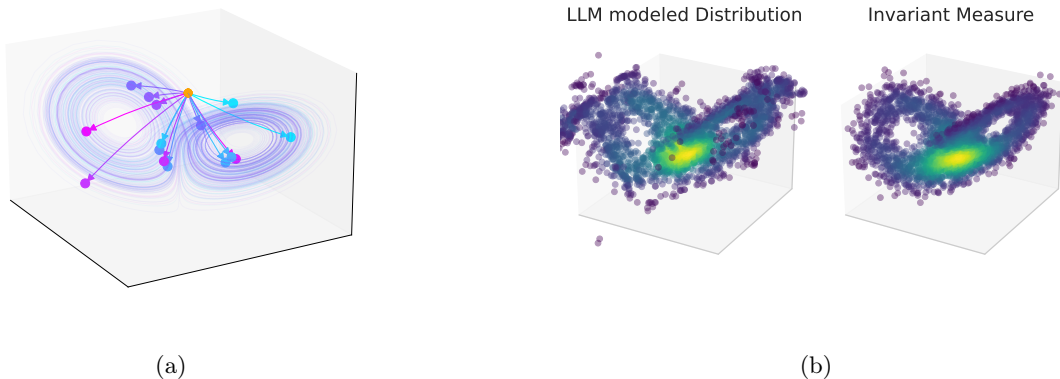
Figure 11: **LLMs can learn the invariant measure of chaotic systems despite unpredictable trajectories.** (**a**) Chaotic systems like the Lorenz equations display sensitive dependence on initial conditions. Tiny perturbations to the initial conditions (orange) diverge exponentially, making long-term predictions impossible when simulating with limited computation and precision on a computer. (**b**) 3000 sampled points from the distribution modeled by the LLM (left) and from the invariant measure of the Lorenz system (right). Color denotes kernel density estimation of each density.

remarkably complex emergent behavior, including stable structures (blocks), periodic oscillators (blinkers), mobile patterns (gliders), and structures that generate infinite streams of gliders (glider guns). The system also happens to be Turing-complete, with a specific initial configuration specifying the program, it is capable of universal computation.

Appendix F. Emergence

Lorenz System and Chaotic Dynamics For the Lorenz system, a canonical example of a chaotic ODE, we can observe a different kind of emergence (Type-0 in Carroll and Parola (2024)). There exists a canonical invariant measure in dynamical systems (under some regularity conditions) known as the SRB measure(Metzger, 2000). States evolved for a long time in the Lorenz system will converge this measure. As the Lorenz system is chaotic, tiny perturbations are exponentially amplified through time at a rate related to the largest Lyapunov exponent $\lambda_1 \approx 0.9$. There is a precise sense in which entropy is created in this system at a rate of $\lambda_1 \log_2(e)$ bits per second, formalized through Pesin's theorem (Pesin, 1977), despite the fact that it is a purely deterministic process. Intuitively one can see this picture when simulating the system using fixed precision numbers, and seeing $\log_2(e)$ bits of that description replaced with unpredictable random content after every Lyapunov time $1/\lambda_1$. On the one hand randomness is produced, but it is not uniformly random. Rather, there is a stationary measure in the shape of a butterfly, and an observer who has lost track of all previous bits due to chaos can still learn the shape of the butterfly. Moreover, the shape of the stationary measure is not immediately obvious from the ODE, it is emergent and cannot easily be understood without intensive numerical simulation of the system (hence why most of chaos theory was developed after computers).

To demonstrate this interplay, we train a language model to predict the first $B = 10$ bits of the future state $\Phi_t(X)$ from an initial state sampled uniformly from the box $X \sim U[-20, 20]^3 + 20[0, 0, 1]$ quantized to B bits, in comparison to directly modeling $\Phi_t(X)$. For both we set the time t to be 30 Lyapunov times into the future, $t = 30/\lambda_1$. The resulting model has a nearly identical loss and estimated epiplexity in the two settings. Despite being unable to distinguish the initial conditions, the LLM learns the invariant (SRB) measure to a reasonable approximation as shown in Figure 11b.

With very limited compute the stationary measure is not predictable apriori from the dynamics, but with more compute it is merely a consequence. The epiplexity of the attractor for limited compute may be larger than a description of the dynamics $S_T(\Phi_t(X)) > S_T(\Phi, t)$.

Chess: AlphaZero and Minimax A qualitatively different kind of example can be had by considering the models produced by AlphaZero (Silver et al., 2018) and the theoretically optimal minimax solution for these two player zero sum perfect information games (von Neumann, 1928; Shannon, 1950). The minimax strategy can be implemented by a short program, and with sufficient compute (exponential in the size of the board (Fraenkel and Lichtenstein, 1981)) the optimal strategy can be found. On the other hand the CNN policy and value network produced by AlphaZero contain 10s of millions of parameters. Given that the rules of chess can be encoded in just a few hundred bytes, and the algorithm used to train the model can be simply described and also implemented by a short program, one may wonder *where does this information come from?* With the other examples of emergent phenomena in mind, we can make sense of this information being produced by the computational process of the AlphaZero system. In contrast, with unbounded compute, the best strategy contains little information.

To summarize, due to the existence of emergent phenomena, even systems that have simple generating processes or simple descriptions can lead to large amounts of structural information to be learned by computationally constrained observers.

Appendix G. Induction is Not Specific to Autoregressive Factorization

One might get the impression that key constraint that leads to this induction phenomenon is the autoregressive factorization, as it is intuitive to see how such a model needs to perform induction in-context to achieve minimum loss. However, we argue this phenomenon takes place with other classes of generative models trained as long as they are trained with Maximum Likelihood Estimation (MLE) or its approximations.

In MLE, a generative model allowing explicit likelihood evaluation is trained to maximize the likelihood of the data. Computing the likelihood can be significantly more computationally challenging than sampling from the distribution P. This distinction is particularly clear in the examples we gave where the ground-truth P is a mixture distribution represented by a latent variable model with the CA initial state or Markov chain transition matrix acting as the latent variable Z. Given access to $P_{X|Z}$ (equivalent to some easy to implement forward function F), sampling is easy as long as P_Z is a simple, but computing $P_X(x)$ for some input x requires evaluating an intractable integral $P_X(x) = \int P_{X|Z}(x|z)P_Z(z)\,dz$ due to the high-dimensionality of Z. As such, a model given a limited compute-budget is forced to learn a cheaper but more sophisticated algorithm for computing $P_X(x)$, often involving approximating the inverse $P_{Z|X}$ either explicitly as done in expectation–maximization-type algorithms and Variational Autoencoders (Kingma et al., 2013), or implicitly as we illustrated for the autoregressive transformer.

Appendix H. Minimum Description Legnth

Intuitively, $L(H)$ can be interpreted as the structural information, and $-\log P(x \mid H)$ can be understood as the remaining random information that cannot be predicted by the best model in $\mathcal{H}$. A main problem with the crude two-part code is that it does not prescribe how one should design the code for H (i.e., a procedure for describing H within $\mathcal{H}$). The description of a particular H can be short under one code but very large under another, which could require additional knowledge to resolve. To circumvent this issue, one can use a more refined one-part code that describes the data

with the entire model class $\mathcal{H}$ rather than any single model H. One of the most important one-part codes is the normalized maximum likelihood code.

Definition 31 (Normalized maximum likelhood code (Grünwald, 2007)) *The NML distribution $P_{\mathcal{H}}^{\mathrm{NML}} : \{0,1\}^{n \times d} \to [0,1]$ of a probablistic model class $\mathcal{H}$ is:*

$$P_{\mathcal{H}}^{\mathrm{NML}}(x) = \frac{P(x \mid \widehat{H}(x))}{\sum_{y \in \{0,1\}^{n \times d}} P(y \mid \widehat{H}(y))},$$

where $\widehat{H}(x) = \arg\max_{H \in \mathcal{H}} P(x \mid H)$ is the maximum likelihood estimator for x.

Crucially, notice that the NML code only depends on $\mathcal{H}$ rather than any particular $H \in \mathcal{H}$, so we do not have to design a particular code for H. Unfortunately, the NML code requires integrating over the maximum likelihood estimator for all possible data, which is intractable for most practical models such as deep neural networks. We can instead use a more tractable variant of one-part code based on sequential prediction called prequential coding.

Definition 32 (Prequential code (Grünwald, 2007)) *The prequential distribution $P_{\mathcal{H}}^{\mathrm{PREQ}} : \{0,1\}^{n \times d} \to [0,1]$ of a probabilistic model class $\mathcal{H}$ is:*

$$P_{\mathcal{H}}^{\mathrm{PREQ}}(x) = \prod_{k=1}^{n} P(x_k \mid \widehat{H}(x_{1:k})),$$

where $\widehat{H}(x_{1:k}) = \arg\max_{H \in \mathcal{H}} P(x_{1:k} \mid H)$ is the MLE for the first k elements of x.

This definition above uses the MLE for updating $\widehat{H}$ but there are in fact no constraints on how the update is performed. We may use any update method of our choice to produce the next model in the sequence, so long as it only depends on the previous data. This means that we can naturally adapt it for deep learning, where we use stochastic gradient descent to update the model sequentially.

A code cannot be optimal simultaneously for all possible data x unless it has knowledge of the particular x. Therefore, it is useful to characterize how close a given code is to the optimal model, which can be formalized via the notion of *regret*.

Definition 33 (Regret (Grünwald, 2007)) *The regret of a code Q relative to $\mathcal{H}$ for x is the additional number of bits needed to encode x using Q compared to the best model in hindsight,*

$$\mathsf{Reg}(Q, \mathcal{H}, x) = -\log Q(x) - \min_{H \in \mathcal{H}} \{-\log P(x \mid H)\}.$$

Under this notion of penalty, the NML is optimal in the sense that it achieves the minimax regret. The regret provides a way to compare different codes. Consider the two-part regret of the crude two-part code $P^{2\mathrm{P}}(\cdot)$ with minimizer $H^\star$ and associated predictive distribution $P(\cdot \mid H^\star)$,

$$\mathsf{Reg}(P^{2\mathrm{P}}, \mathcal{H}, x) = L(H^\star) + \log \frac{1}{P(x \mid H^\star)} - \log \frac{1}{P(x \mid \widehat{H})}.$$

This means that for a two-part code, the regret is an upper bound on the description length of the model. For sufficiently large n, the last two terms become close to each other and $\mathsf{Reg}(P^{2\mathrm{P}}, \mathcal{H}, x) \approx L(H^\star)$. In the case of NML, the regret is the minimax regret that $\mathsf{Reg}(P_{\mathcal{H}}^{\mathrm{NML}}, \mathcal{H}, x) = \log \sum_{y \in \{0,1\}^n} P(y \mid \widehat{H}(y))$. This quantity is independent of x, which is also called *parametric complexity* of $\mathcal{H}$, because

it measures how expressive the entire *model class* is by counting the total amount of possible data sequences the model class can model well.

As shown above, the regret of a code is fundamentally related to the size of the model. Similarly, the regret of the prequential code can also be interpreted as a measurement of model size even though it does not provide a separate description of the model. While prequential code is not minimax optimal like NML, it is shown that prequential Bayes code can have a small additive $O(\log n)$ regret penalty compared to NML. Furthermore, any non-prequential code (e.g., NML or two-part code) can be converted into a prequential code with only $O(\log n)$ overhead. As such, we can interpret $\mathsf{Reg}(P_{\mathcal{H}}^{\mathrm{PREQ}}, \mathcal{H}, x) = \log 1. P_{\mathcal{H}}^{\mathrm{PREQ}}(x) - \log 1/P(x \mid \widehat{H})$ as a measurement of the effective model size conditioned on the data x. The crucial advantage of prequential code is that it is data-dependent, it is fully compatible with sequential model optimization, and it avoids the need for estimating individual model size $L(H)$, making it an ideal tool for studying deep learning and properties of the data.

In section 4.3, we introduced $\mathsf{Reg}(P_{\mathcal{H}}^{\mathrm{PREQ}}, \mathcal{H}, x)$ as a proxy for $L(H^\star)$. Indeed, since prequential coding does not emit a valid code for the model, only the code for both the data and model, it can only be interpreted as an approximation of epiplexity. However, if we forgo the two-part code formulation and use the regret as a measurement of the effective model size, then prequential epiplexity is no longer an approximation. We will refer to this regret-based formulation as *generalized epiplexity*. The main benefit of generalized epiplexity is that it applies to a much wider range of coding schemes (i.e., both one-part and two-part codes), as it only requires the total length of the data and model, and a reference probability model. Unfortunately, the theoretical results based on two-part epiplexity do not immediately transfer to generalized epiplexity. It remains an open question whether similar results can be obtained.