DO LARGE LANGUAGE MODELS LEARN ORDER-DEPENDENT RULES?

Anonymous authors
Paper under double-blind review

ABSTRACT

Can large language models (LLMs) capture sequential structure when outcomes depend on order? Theoretically and experimentally, we prove that ordered-compliance inherently leaks information through frequency patterns, creating predictive shortcuts that allow models to succeed without truly capturing causal sequential structure. To isolate true order learning, we introduce a shortcut-auditing protocol that quantifies the predictive power of non-sequential features (e.g., k-grams, pair counts) under label noise. We develop a controlled, semantic-free sequence framework with three mechanisms over a hidden set of key letters: ordered compliance, driven by key order at a fixed lag; global-property compliance, driven by the parity of the total key count; and key-inversion parity, driven by the parity of key-order inversions. Empirically, encoder and recurrent models reach the theoretical optimum (measured by AUC and F1) on ordered-compliance more efficiently than fine-tuned decoder-only LLMs. On global-property compliance, the tested sequence models remain near chance under standard training unless explicitly guided on feature relevance. On key-inversion parity, recurrent and transformer models can solve the task with four keys, while all tested models remain near chance with six. Our results demonstrate that high-quality performance on order-sensitive tasks often masks shortcut exploitation.

1 INTRODUCTION

Consider a clinician reviewing a patient's medical history to predict disease progression. The order of events matters: for predicting *myocardial infarction*, *diabetes mellitus* followed by *visual disturbance* tells a different story than the same two events in reverse order (Edelman et al., 2026). A patient who experiences the transition from cancer stage 2 to cancer stage 3 is worsening, while the opposite order would indicate an improvement. Recent studies of electronic health records (EHR) produce comparator cohorts that keep the same events but change their temporal order, and find that sequence-based disease representations consistently outperform bag-of-events models because the order of disease progression carries prognostic information (Edelman et al., 2026; Beaney et al., 2024; Placido et al., 2023; Pollington et al., 2026). Similarly, in recommendation systems, the same items can reflect different user preferences depending on the order in which they were selected (Kim et al., 2025). In these settings, predictions depend not only on *which* events occurred but also on the *order* in which they occurred. Yet, if a sequential model outperforms a bag-of-events model on order-dependent processes, this does not necessarily mean that it has learned the causal power of the order (Geirhos et al., 2020; McCoy et al., 2019). Sequential data contain many overlapping signals: which events occur, how often, when, and which tend to appear together. A model can use any of these statistics to predict the outcome, if useful, without ever recovering the order-based rule that actually generates the outcome. We call any such signal a *predictive shortcut*: a set of statistics computed from the input sequence that permits above-chance prediction of the label without being sufficient to recover the intended label-generating mechanism. The difficulty is that real-world event trajectories rarely come with a known underlying mechanism, and the space of possible shortcuts is open-ended. LLMs and other transformer-based sequential models are increasingly used for clinical event prediction, sequential recommendation, and temporal reasoning (Savcisens et al., 2023; Shmatko et al., 2025; Li et al., 2020; Rasmy et al., 2021; Fatemi et al., 2024). We therefore focus on evaluating LLMs and other transformer-based models on sequential tasks and comparing their performance against recurrent neural networks, gradient-boosted trees such as XGBoost, and other statistical baselines.

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

In general, does the performance of LLMs on sequential data reveals anything about how and whether the LLM learns the output-generating mechanism? Recent evidence suggests it may not: LLM-based recommenders often retain most of their performance even when the item order is shuffled (Kim et al., 2025), and transformer predictions can often be approximated by simple k-gram statistics[1] (Nguyen, 2024). This motivates us to design a controlled framework based on sequences of letters that deliberately removes meaning and world knowledge. We introduce *ordered-compliance* (OC): a semantic-free abstraction in which a hidden subset of letters (keys) must satisfy a secret ordering rule at a fixed lag to produce a favorable outcome. While label prediction is genuinely order-sensitive, predictive shortcuts may remain available. Therefore, high performance on OC tasks alone cannot distinguish capturing of the intended rule from exploitation of a shortcut.

To complement OC, we introduce two further classes of tests, *key-inversion-parity* (KIP) and *global-property-compliance* (GPC) as stress-test controls. All three classes of tasks share two ingredients: only a hidden subset of letters is relevant, and labels depend on their hidden structure. They progress from an ordering rule which can be substituted with specific shortcuts (OC) to a counting rule that, with balanced sampling, removes predictive shortcuts completely (GPC). KIP is intermediate: order matters, but the label depends on whether the number of key pairs violating the order is even or odd. Neither letter counts nor small subsets reveal the label. GPC removes order dependence by counting key-letter occurrences and assigning label 1 if an even number of key letters have even counts, otherwise 0. In order to justify our three classes of tasks theoretically, under the assumptions in Section 3, we prove that the three tasks form a progression in how much order matters and how many shortcuts remain. OC uses an explicit order rule, but allows local shortcuts: even a single position carries information about the label, though not necessarily enough to determine it (Theorem 3.4). KIP still depends on order, but only through the number of order violations, and it removes several shortcuts: neither letter counts nor any fixed set of at most $m - 2$ positions carries information about the label, where m is the number of key letters (Theorem 3.6). GPC completely eliminates the order and eliminates all local shortcuts: anything shorter than the full sequence provides no information about the label (Theorem 3.8).

We compare LLMs, encoder-only transformers, and recurrent networks across model sizes and training strategies. Using OC, KIP, and GPC, we assess whether these models outperform simpler predictors and baselines that rely just on simple statistics that express predictive shortcuts. We examine both predictive performance and data efficiency, tracking how performance varies with model size and training data quantity. Our comparisons use standard training on the input sequences, without revealing the hidden key letters or providing additional information about the underlying rule. Several models perform well on OC,but simple statistical baselines can match the best neural models. Success is less consistent on KIP: some models solve the four-key task, but all tested models remain near chance with six keys. On GPC, all the tested models remain near chance. Additional experiments show that providing additional task-relevant information can enable models to solve otherwise unsuccessful KIP and GPC settings. Finally, we apply our approach to a real-world clinical case.

Our findings suggest that the tested models fail to recognize order-dependence as the cause of an outcome, relying instead on predictive shortcuts to achieve high performance. Future theoretical and experimental work is required to verify this hypothesis further.

Contributions.

- **Theory.** We define three label-generating mechanisms over letter sequences (OC, KIP, and GPC) and establish their theoretical properties concerning order dependence and the availability of predictive shortcuts.
- **Shortcut-auditing protocol.** We measure how the predictive performance can be explained by different kinds of shortcuts. We also test how changing event order affects predictions and compare model performance with the theoretical optimum under label noise.
- **Empirical evaluation.** We compare LLMs, encoder-only transformers, and recurrent models across model sizes, training strategies, and training data budgets. We also apply the audits to a real

[1] k-gram statistics are the frequency distributions of contiguous sequences of length k (characters, words, or tokens) extracted from text. Unigrams (k = 1), bigrams (k = 2), trigrams (k = 3), etc., capture local sequential patterns and are fundamental in many NLP applications (Suen, 1979).

clinical case to assess how much performance can be explained by simpler statistical patterns in real-world sequential tasks.

2 Related work

A growing body of work in domains such as healthcare and recommendation systems suggests that temporal order can add predictive value beyond unordered event identity (Beaney et al., 2024; Placido et al., 2023; Pollington et al., 2026; Kim et al., 2025), and transformer-based models are now routinely applied to longitudinal event histories (Li et al., 2020; Rasmy et al., 2021; Savcisens et al., 2023; Shmatko et al., 2025). Of particular relevance, Edelman et al. (2026) study interpretable $A \to B \to C$ clinical trajectories using comparator cohorts that preserve event content while changing temporal structure, including B without prior A, reverse-order, and unordered comparisons. Such settings, together with event-log domains such as predictive process monitoring and system-log anomaly detection (Teinemaa et al., 2019; Du et al., 2017), motivate our question but cannot answer it directly: sequence-model gains may reflect temporal order, event semantics, co-occurrence, recency, or local event-pattern shortcuts rather than global trajectory reasoning.

Several recent benchmarks probe temporal or sequential ability (Fatemi et al., 2024; Cui et al., 2025), but Klenitskiy et al. (2024) argue that many datasets marketed as sequential do not in fact require strong sequential modeling. Consistent with this, transformer behavior is often well approximated by local statistics or k-gram rules (Nguyen, 2024; Kim et al., 2025; Bedi et al., 2025), connecting to the broader shortcut-learning literature (McCoy et al., 2019; Geirhos et al., 2020; Mirzadeh et al., 2024). At the same time, sequential models can in principle realize dependencies beyond fixed k-gram rules (Gu & Dao, 2024). Recent work has introduced tunable synthetic benchmarks for reasoning and sequence processing (Kaiser et al., 2025; Ramezanali et al., 2025). Our framework differs by stripping away semantics while enabling a theoretical analysis of the local-information structure of each mechanism. OC suffers unavoidable local leakage; KIP still depends on order but in a less direct manner and restricts the available shortcuts. Balanced GPC removes order dependence entirely, and anything less than the full sequence carries no information about the label.

3 Methods and theory

We study three label-generating tasks: ordered compliance (OC), global-property compliance (GPC), and key-inversion parity (KIP). We use the term *task* for the combination of a distribution over input sequences, a mechanism that classifies them, and a label-noise probability. The mechanism assigns each sequence a binary label, which is then flipped with the label-noise probability to give the observed label.

3.1 General setup

Let $\mathcal{A}$ be a set of symbols of size $\ell \geq 2$, let $n \geq 2$ be the sequence length, and write $[h] = \{1, \ldots, h\}$. For each task instance, sample a hidden key set $\mathcal{S} \subseteq \mathcal{A}$ uniformly among subsets of size m, where $1 \leq m \leq \ell$. For OC and KIP, also sample a uniform bijection $\kappa : \mathcal{S} \to [m]$. These parameters remain fixed across all splits; all probabilities condition on their realized values. GPC uses only $\mathcal{S}$.

For OC and GPC, sequences $X = (X_1, \ldots, X_n)$ are generated from the distribution P_0, defined by

$$X_t \overset{\text{iid}}{\sim} \text{Uniform}(\mathcal{A}), \qquad t = 1, \ldots, n.$$

so each letter is drawn independently and uniformly from the alphabet $\mathcal{A}$. For KIP, sequences are generated from the conditional distribution Q_0 obtained from P_0 by conditioning on each key letter occurring exactly once. An explicit sampling procedure is given in Definition 3.5.

Each sequence is assigned a latent, noise-free binary label $Y^* = f(X) \in \{0, 1\}$, where f is the deterministic labeling rule of the chosen mechanism, corresponding to ordered compliance (OC), global-property compliance (GPC), and key-inversion parity (KIP), defined below.

To allow for label noise, the observed label Y is generated as

$$Y = Y^* \oplus E, \qquad E \sim \text{Bernoulli}(\pi), \qquad E \perp\!\!\!\perp X, \qquad 0 \leq \pi < \tfrac{1}{2}, \tag{1}$$

where $\oplus$ denotes addition modulo two: $E = 0$ leaves the latent label unchanged, whereas $E = 1$ flips it. Thus, each label is flipped with probability π, independently of the sequence and independently across examples. Setting $\pi = 0$ gives deterministic labels.

Let P denote the joint distribution of (X, Y^*, Y) induced by the corresponding input distribution, labeling rule, and noise model. Write $\rho = P(Y^* = 1)$ for the latent positive-class prevalence. For $x \in \mathcal{A}^n$ and $a \in \mathcal{A}$, define

$$N_a(x) := \sum_{t=1}^{n} \mathbf{1}\{x_t = a\}, \qquad C(x) := (N_a(x))_{a \in \mathcal{A}}, \tag{2}$$

the symbol counts and their count vector. For each position $t \in [n]$, define the membership bit

$$B_t := \mathbf{1}\{X_t \in \mathcal{S}\}. \tag{3}$$

Definition 3.1 (Order sensitivity)**.** A task with deterministic labeling rule f is *order-sensitive* if there exist two sequences x, x' in the input support that contain the same number of occurrences of each symbol, but satisfy

$$f(x) \neq f(x').$$

Definition 3.2 (Local purity)**.** For a fixed $J \subseteq [n]$, let $X_J = (X_j)_{j \in J}$, ordered by index. A task is *J-locally pure* under P if $I_P(Y; X_J) = 0$, equivalently $P(Y = 1 \mid X_J = x_J) = P(Y = 1)$ for every positive-probability view x_J. For $k \in [n]$, it is *k-locally pure* if this holds for every J with $1 \leq |J| \leq k$. The same definitions apply to Y^*.

Here I_P denotes mutual information[2]. Positions in J need not be adjacent. The set J is fixed in advance and does not depend on the realized sequence. Local purity forms a hierarchy: k-local purity implies j-local purity for every $1 \leq j \leq k$, with larger k imposing a stronger requirement. We call a task *maximally locally pure* if it is $(n-1)$-locally pure, meaning that no fixed proper subset of the sequence positions provides information about the label. For a non-degenerate latent label, $0 < \rho < 1$, and the independent label-noise model above with $\pi < 1/2$, this is the strongest possible level: the full sequence remains informative about the label.

3.2 Ordered compliance mechanism

Definition 3.3 (Ordered-Compliance Mechanism)**.** Under the setup of Section 3.1, fix an integer lag $\lambda \in [n-1]$. A *λ-run of key symbols* in a sequence $X = (X_1, \ldots, X_n)$ is a tuple of positions

$$r = (t, t+\lambda, \ldots, t+(q-1)\lambda),$$

where t, q are positive integers satisfying $t + (q-1)\lambda \leq n$, and

$$X_{t+i\lambda} \in \mathcal{S} \qquad \text{for every } i = 0, \ldots, q-1.$$

The run is *maximal* if it cannot be extended to the left or to the right by another key symbol at distance λ; that is, both of the following conditions hold:

$$t - \lambda < 1 \quad \text{or} \quad X_{t-\lambda} \notin \mathcal{S},$$
$$t + q\lambda > n \quad \text{or} \quad X_{t+q\lambda} \notin \mathcal{S}.$$

A maximal λ-run $r = (t, t+\lambda, \ldots, t+(q-1)\lambda)$ is *ordered* if $q \geq 2$ and

$$\kappa(X_t) \leq \kappa(X_{t+\lambda}) \leq \cdots \leq \kappa(X_{t+(q-1)\lambda}).$$

We say that X is *ordered-compliant* (OC) if it contains at least one ordered maximal λ-run.

For this mechanism, the labeling rule is $Y^* = f(X) = \mathbf{1}\{X \text{ is ordered-compliant}\}$.

Maximality excludes increasing pairs inside longer unordered runs: for ranks $A < B < C$, distractor d, and $\lambda = 1$, (d, A, B, C, d) is compliant but (d, A, C, B, d) is not. Repeated keys are allowed.

[2] $$I_P(Y; X_J) = \sum_{y \in \{0,1\}} \sum_{x_J \in \mathcal{A}^J} P(Y = y, X_J = x_J) \log \frac{P(Y = y, X_J = x_J)}{P(Y = y) P(X_J = x_J)}.$$

Theorem 3.4 (OC contains single-position label information). *Fix $(\mathcal{S}, \kappa)$, an integer $\lambda \geq 1$, and $n \geq \lambda + 1$. More generally than the uniform setting above, suppose X has independent coordinates with full support on $\mathcal{A}$. Let P denote the joint distribution induced by this input distribution, the OC mechanism in Definition 3.3, and the noise model in (1). Write $\rho = P(Y^* = 1)$. If $0 < \rho < 1$, there exist $a, b \in \mathcal{A}$ such that*

$$P(Y^* = 1 \mid X_1 = a) \neq P(Y^* = 1 \mid X_1 = b).$$

Consequently, $I_P(Y^*; X_1) > 0$ and $I_P(Y; X_1) > 0$.

The proof is in Appendix A.2. Under the assumptions of the theorem, OC fails every positive level of local purity: it is not k-locally pure for any $1 \leq k \leq n$, since the singleton $J = \{1\}$ already contains information about the label.

The OC mechanism has outcomes that depend on whether some events satisfy a hidden ordering rule. Theorem 3.4 shows that this structure also leaks predictive information through single positions: under the theorem's assumptions, even the first position carries information about the label. A predictor, therefore, can exploit shortcuts without reading the whole sequence and hence without identifying the hidden causal mechanism.

3.3 Key-inversion parity mechanism

We introduce key-inversion parity (KIP), an order-sensitive mechanism that is $(m-2)$-locally pure.

Definition 3.5 (Key-inversion parity). Under the setup of Section 3.1, take $3 \leq m \leq n$ and $m < \ell$. Define Q_0 as the probability law on $\mathcal{A}^n$ generated by the following procedure: choose m key positions uniformly from the m-subsets of $[n]$, assign the keys in $\mathcal{S}$ to those positions by a uniform random bijection, and fill remaining positions independently and uniformly from $\mathcal{A} \setminus \mathcal{S}$, independently of these choices. Each key occurs exactly once. For key positions $T_1 < \cdots < T_m$, let $R_i = \kappa(X_{T_i})$, write $R = (R_1, \ldots, R_m)$ and define

$$\mathrm{Inv}(R) = \sum_{1 \leq i < j \leq m} \mathbf{1}\{R_i > R_j\}, \qquad Y^* = \mathrm{Inv}(R) \bmod 2. \tag{4}$$

The exactly-once constraint makes coordinates dependent: KIP differs from OC in both input law and label rule, so the OC theorem does not apply.

Theorem 3.6 (Order sensitivity and local purity of KIP). *Under Definition 3.5, let $C(X) = (N_a(X))_{a \in \mathcal{A}}$. Then $P(Y^* = 1) = 1/2$, and swapping any two keys flips the label while preserving $C(X)$. Moreover,*

$$I_P(Y^*; C(X)) = 0, \qquad I_P(Y^*; X_J) = 0 \quad \textit{for every } J \subseteq [n] \textit{ with } |J| \leq m - 2.$$

KIP is not $(m-1)$-locally pure, so its threshold is exactly $m-2$. Both independence and failure of $(m-1)$-local purity persist for Y under (1).

The proof is in Appendix A.3. Our $n = 20$, $m \in \{4, 6\}$, $\pi = 0$ settings have exact purity through two and four positions. In KIP, symbol counts alone provide no information about the label.

3.4 Global-property compliance mechanism

We finally introduce global-property compliance (GPC), a mechanism that is not order-sensitive and satisfies $(n-1)$ local purity.

Definition 3.7 (Global-property compliance). Under P_0 and the general setup defined of Section 3.1, take $1 \leq m < \ell$. GPC sets $Y^* = 1$ when an even number of keys have even counts, including zero occurrences; equivalently,

$$Y^* = \mathbf{1}\left\{ \sum_{t=1}^{n} B_t \equiv m \pmod 2 \right\}. \tag{5}$$

To see why the two formulations agree, let o be the number of keys with odd counts. Then $m - o$ keys have even counts, and $\sum_t B_t \equiv o \pmod 2$, since only odd counts contribute to the sum modulo two. Therefore, $m - o$ is even exactly when $\sum_t B_t \equiv m \pmod 2$.

Theorem 3.8 (Local information in GPC). *Under Definition 3.7 and P_0, let $p = m/\ell \in (0, 1)$. For every nonempty $J \subsetneq [n]$,*

$$I_P(Y^*; X_J) = 0 \text{ if } p = \tfrac{1}{2}, \qquad I_P(Y^*; X_J) > 0 \text{ otherwise.}$$

For fixed $p \neq 1/2$ and a finite nonempty index set J, this information tends to zero as $n \to \infty$. The same holds for Y under (1). Hence balanced GPC ($p = 1/2$) is k-locally pure for every $1 \leq k < n$.

The proof is in Appendix A.4.

In essence, our theoretical measure how much a sequential task is shortcut-dominated. OC shows this directly. Its label depends on order, even a single position carries information about it, so a model can perform well on OC without using the order rule. This motivated KIP and GPC as comparison tests that theoretically allow fewer shortcuts. KIP keeps the label order-dependent and removes two of them: neither letter counts nor any fixed set of at most $m - 2$ positions carries information about the label. GPC removes all local shortcuts: anything less than the full sequence carries no information. Comparing a model across the three tasks, and against baselines restricted to the statistics each task allows, is what lets us tell rule learning from shortcut use.

4 Experiments

We evaluate how well different architectures perform on the tasks built from the three mechanisms defined above and whether they outperform baselines that rely on predictive shortcuts. These tasks vary in whether order matters and in how much information individual positions or small groups of positions provide about the label. We apply the same evaluation framework to a real clinical task in MIMIC-IV (Johnson et al., 2023), a widely used database of electronic health records. Code is available in an anonymous repository.[3]

4.1 Controlled synthetic dataset and task variants

We instantiate the three mechanisms on sequences of length $n = 20$ over an alphabet of $\ell = 26$ symbols, yielding five experimental variants.

Ordered compliance. OC-Deterministic and OC-Noisy use $m = 6$ keys and lag $\lambda = 7$, with label-noise probabilities $\pi = 0$ and $\pi = .3$, respectively. Both have latent positive-class prevalence $\rho \approx .293$. The lag makes the relevant ordering non-adjacent, while the two noise levels allow us to assess the effect of label noise on model performance. Parameter choices and sensitivity analyses are discussed in Appendix D.

Key-inversion parity. KIP-4 and KIP-6 use four and six keys, respectively, with no label noise and latent positive class prevalence $\rho = .5$. Each key occurs exactly once. By Theorem 3.6, these variants are exactly 2- and 4-locally pure, allowing us to compare model performance as both the number of keys and the degree of local purity increase.

Global-property compliance. GPC-Deterministic uses $m = 6$ keys and no label noise, matching the hidden-subset size of the OC variants, with latent positive-class prevalence $\rho \approx .499$. Its label depends on global parity and is invariant to sequence order.

4.2 Models and baselines

We fine-tune pretrained decoder models, Llama-3.2-1B, Llama-3.1-8B, and Llama-3.1-70B (Touvron et al., 2023; Grattafiori et al., 2024), Qwen3-4B-Think (Qwen Team, 2025), Qwen2.5-14B (Qwen Team, 2024; Yang et al., 2024), and Qwen3.8-27B (Qwen Team, 2026) and encoder models BERT-base (110M) (Devlin et al., 2019) and RoBERTa-large (355M) (Liu et al., 2019) for binary classification. To isolate architecture from pretrained knowledge, we also train randomly initialized Llama-style decoders (1M parameters), LSTMs (Hochreiter & Schmidhuber, 1997), transformer encoders (Vaswani et al., 2017), and a hybrid BiLSTM-Transformer (Graves & Schmidhuber, 2005) (called here RNNTransformer). Our baselines include XGBoost (Chen & Guestrin, 2016) with integer-encoded letters (XGBoost-basic) or, following Do et al. (2023); Alshraideh et al. (2025),

[3] https://anonymous.4open.science/r/repro_iclr-77DD/

mean-pooled Llama-3.2-1B embeddings (XGBoost-LLM), contiguous k-gram scores, and classifiers based on letter counts or ordered pair counts. Pair-count features record how often each ordered pair of letters occurs at a specified distance or across all possible distances. On OC, we compare a baseline given the true lag λ with baselines using all $n-1$ possible lags. Regularized logistic regression and boosted trees assess the predictive value of these features. Input encodings and feature definitions are detailed in Appendices B.2.

4.3 Training and evaluation

For the controlled synthetic tasks, all neural models are trained for binary classification with early stopping on validation performance with AdamW (Loshchilov & Hutter, 2019). The classification loss is cross-entropy; for causal decoders, the training loss is the sum of cross-entropy loss and a next-token prediction loss computed over the input prompt. We finetuned pretrained models using LoRA (Hu et al., 2021) and for big models (27B+ parameters) we use QLoRA (Dettmers et al., 2023) with 4-bit precision to reduce memory requirements. The synthetic datasets use an 80/10/10 train/validation/test split. For each task and each model we use 4K, 40K, 120K, 200K, 300K, and 400K training examples in order to check data-efficiency. Experiments run on one NVIDIA H200 GPU (141GB), with three seeds unless otherwise stated. We report test AUC, F1, precision, and recall, using AUC as the primary metric. Model-specific settings are detailed in Appendix B. We compare performance with the label-noise oracle (Theorem A.1). AUC^* is the optimal AUC; precision*, recall*, and F1* refer to the classifier $\hat{Y} = Y^*$. All four values equal 1 in noise-free settings. For OC-Noisy ($\rho = .293$, $\pi = .3$), the corresponding values are .670, .700, .491, and .577, respectively.

4.4 Real-world case study: chronic kidney disease progression

We focus on predicting progression from chronic kidney disease (CKD) to end-stage renal disease (ESRD) using patient histories. This choice follows previous studies treating the task as sequential prediction (Zhu et al., 2023; Kadatz et al., 2016; Tangri et al., 2017; Echouffo-Tcheugui & Kengne, 2012). We use MIMIC-IV (Johnson et al., 2024; 2023) and we assess whether the order of recorded diagnoses adds predictive value beyond their identity and frequency. The cohort contains 14 813 patients with CKD, including 1 146 positives whose subsequent admissions contain an ESRD-related code. For positive cases, inputs stop before the first ESRD-related admission; patients with CKD and ESRD first recorded in the same admission are excluded. Diagnosis codes are mapped to 293 Clinical Classifications Software (CCS) categories (Agency for Healthcare Research and Quality, 2025). Admissions are ordered chronologically, while codes within each admission follow administrative billing order. We compare different models using original and shuffled diagnosis histories, with AUC as the main evaluation metric. Details are in Appendix F.

5 Results

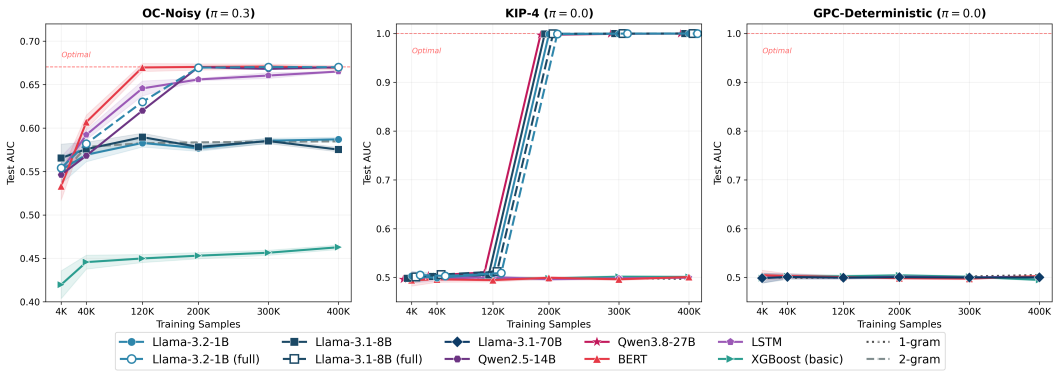

Figure 1: Test AUC versus training-set size on OC-Noisy, KIP-4, and GCP. For readability, we show one representative model per family (full results in Appendix C). Dashed curves labeled *full* denote full fine-tuning; other LLMs use LoRA/QLoRA.

This section summarizes the main results; the complete ones are in Appendix C and F. Figure 1 shows how test AUC changes with training-set size on the three most informative settings, OC-Noisy, KIP-4, and GPC, for one representative model per family. In appendix G we inspect attention layer by layer for encoder and decoder-only models on all the tasks. Notably, in KIP-4, only decoder-only models solve the task.

Ordered compliance. On **OC-Deterministic**, pretrained bidirectional encoders (BERT, RoBERTa) and small randomly initialized sequential models (Transformer, LSTM, RNNTransformer; 100K–400K parameters) reach AUC $\geq .997$. Decoder-only Llama-3.2-1B plateaus at AUC $\approx .85$, close to the contiguous 2-gram baseline of 0.82, while Llama-3.1-8B reaches AUC $= .995$. XGBoost on frozen Llama-3.2-1B mean-pooled embeddings also fails; this likely reflects a mismatch between pretrained representations and our synthetic letter task rather than a property of the embedding architecture per se. We tested a 676-dimensional logistic regression on the λ-aware pair feature $g_{ab}^{(\lambda=7)}$ (Appendix C) that reaches AUC $= .997$, and that a λ-agnostic 12,844-dimensional logistic regression recovers $\lambda = 7$ on its own. We therefore read the encoder–decoder gap on this task as an efficiency gap in discovering the right lagged-pair feature from raw-token supervision, rather than as evidence for global sequence integration. On **OC-Noisy**, BERT, LSTM, the small Transformer, and RNNTransformer reach AUC between .665 and .671, close to the oracle reference of .670. BERT approaches this value with 120K examples. The number of trained parameters also affects performance: Llama-3.2-1B and Llama-3.1-8B reach .587 and .575 with rank-8 LoRA, but .670 and .668 with full fine-tuning. BERT reaches .669 using LoRA, and Qwen2.5-14B also reaches .670 with LoRA. When the lag λ is given , an L_2 logistic regression on $g_{ab}^{(\lambda=7)}$ reaches AUC $= .671$; without it, XGBoost on pair counts at all lags reaches .667. An L_1 logistic regression on the same all-lag counts reaches only .632: under label noise it fails to identify to identify the features of the correct lag, and its largest weights land on $\lambda = 19$ instead of 7 (Appendix D.2). The results suggest that both OC variants appear shortcut-dominated: lagged-pair statistics suffice for near-optimal performance, while label noise and restricted fine-tuning can hinder feature discovery.

Key-inversion parity. On KIP-4, LSTM reaches AUC 1.000 across three seeds. Several pretrained decoders also achieve near-perfect performance, including Llama-1B and Llama-8B with full fine-tuning, Llama-8B with LoRA, Qwen3.8-27B with QLoRA, and Llama-70B. The successful decoders remain near chance through 120K training examples and reach near-perfect AUC at 200K. On KIP-6, all tested neural models remain near chance under the same training procedures. Shuffle controls show that successful KIP-4 predictions depend on order. For LSTM, retraining on shuffled sequences while retaining the original labels yields AUC .500, and test-only shuffling yields .502. Test-only shuffling also reduces successful models' AUC from 1.000 to .505. For each ordered pair of letters, we count how often the second letter appears one position after the first, two positions after it, and so on up to distance $n - 1$, where n is the sequence length, keeping a separate count for each pair and each distance. A classifier trained on these counts reaches AUC 1.000 on KIP-4. In a separate experiment, we use the known key set to build an input that says, for each pair of keys, which one occurs first in the sequence. A small MLP trained on this input reaches AUC 1.000 on both KIP-4 and KIP-6. Thus, once the relative order of the keys is supplied, even KIP-6 is solvable. Here, extracting key-order information is the major bottleneck, since the task is solvable once this information is explicit. KIP-4 highlights the ability of large decoders to solve this task where shortcuts appear hard to exploit, achieving near-perfect performance with sufficient training data.

Global-property compliance. On GPC, all tested models remain near chance (AUC ≈ 0.50), including 70B-parameter LLMs, encoder-only transformers, and recurrent networks. The k-gram baselines remain near chance as well, consistent with Fu et al. (2024). This holds both in the main $|\mathcal{S}| = 6$ regime and, for the core architectures, in the exactly balanced $|\mathcal{S}| = 13$ regime. Once key membership is provided, the parity rule becomes learnable (Appendix E). A 64-unit MLP given the membership vector $(B_1, \ldots, B_n)$, where $B_t = \mathbf{1}\{X_t \in \mathcal{S}\}$, reaches AUC .997 in under a minute; this n-dimensional binary input says which positions hold a key, but not which key. Given the m-dimensional per-key parity vector $(N_a(X) \bmod 2)_{a \in \mathcal{S}}$ instead, the same MLP reaches AUC 1.000. Results also depend on sequence length: some models that succeed on shorter sequences remain near chance when trained directly on longer ones (Appendix E.1). Hahn (2020) identifies length-dependent difficulties for transformer parity learning, consistent with what we observe: small encoders learn parity at $n = 10$ but collapse to chance at $n = 20$ (Appendix E.1). We exploit this

gap with a two-stage curriculum: train from scratch on $n = 10$, then warm-start on $n = 20$ with positional embeddings transferred where applicable. This recovers global-property-compliance for an LSTM and for a small randomly initialized Transformer, but not for pretrained BERT or Llama under our tested recipes (Appendix E.5). These results highlight the importance of training strategies when shortcuts appear unavailable: task-helper inputs or a length-based curriculum enable small models to learn global parity despite widespread failure under standard training.

Chronic kidney disease progression For CKD-to-ESRD prediction, unigrams achieve an AUC of .823 outperforming contiguous bigrams that reach .801 across five patient splits. On a separate split used for bag-of-code and neural models, count-based logistic regression and XGBoost attain AUCs of .809 and .803, respectively. The Transformer model scores .860 with ordered histories and .844 when the histories are shuffled; BERT and BiLSTM do not show any positive difference between ordered and shuffled histories. Predictors that only use the presence and frequency of diagnosis codes already achieve about .810 AUC. The Transformer performs better with ordered histories, but BERT and BiLSTM show no improvement over the shuffled data. So, there can be significant shortcuts in sequential medical tasks and, as Klenitskiy et al. (2024) has pointed out, phenomena that should depend on the temporal ordering of events might be heavily shortcut-dominated, so that good performance on it reveals nothing about whether a model uses order.

6 Discussion and Limitations

This work introduces a controlled, semantic-free framework to examine whether a model has learned the underlying causal order-dependent rule whenever its predictive performance is good. Our three frameworks form a progression: from OC, where order takes its simplest form, whether A comes before or after B, and for that very reason leaves shortcuts available, to KIP and GPC, where order becomes less and less decisive until it disappears, and fewer or no shortcuts remain available.

On OC, simple classifiers given lagged-pair counts match the best neural models, so what really matters is how efficiently they discover such features from raw tokens. KIP-4 shows the other side: an LSTM and fine-tuned Llama and Qwen models solve it from raw sequences, where count-based baselines cannot, and the only baseline that matches them is given pair counts over all lags as input. On KIP-6 and GPC, every model trained on raw sequences stays near chance under standard training, while small MLPs given key membership or the keys' relative order succeed: so the difficulty lies in discovering the hidden structure and learning the rule at the same time. The clinical audit shows why this matters in practice: which diagnoses occur, and how often, already supports strong prediction, and temporal order adds at most a modest gain. This could indicate that detecting the decisive role of an order of events to predict a clinical outcome might not always be necessary for good predictions, even when causal, because of predictive shortcuts. As recent work highlights (Edelman et al., 2026), while true temporal dependencies certainly exist in clinical trajectories, they are not trivial to isolate. Our work demonstrates that causal sequentiality in real data does not exclude the presence of predictive shortcuts.

From a theoretical perspective, the OC theorem is specific to the OC mechanism under independent full-support sampling and does not extend to arbitrary order-sensitive labels. The amount of OC label information depends on the sampling distribution. KIP uses dependent inputs and falls outside these assumptions. The local-purity guarantees concern fixed sets of positions and do not exclude informative features computed from the whole sequence. These results describe where label information is available; they do not determine whether a model will learn to use it. Likewise, high AUC and sensitivity to noise do not identify a model's internal reasoning. Our experimental setting is not a general benchmark for long-range sequential learning. Empirically, we focus on one main regime per task, with sensitivity sweeps and ablations in the appendix; most models are tested up to 14B parameters, with the 70B model evaluated only on the GPC mechanism. The theoretical claims also rely on independent full-support sampling and should not be read as impossibility results for arbitrary real-world sequence distributions, transformers, or parity-like tasks in general.

Our work leads to the hypothesis that models are unable to detect order-dependence as the actual reason for an outcome, when this is the case, as their predictive performance seems to rely on the presence of predictive shortcuts.

7 AI USE STATEMENT

LLMs were used to assist with code debugging and visualization scripts, improve writing, draft selected manuscript paragraphs, and discuss the interpretation and limitations of the results. The authors reviewed and revised the AI-assisted text and take full ownership and responsibility for the final content of this work.

REFERENCES

Agency for Healthcare Research and Quality. HCUP clinical classifications software (CCS) for ICD-9-CM, 9 2025. URL https://hcup-us.ahrq.gov/toolssoftware/ccs/ccs.jsp. Accessed 2026-07-31.

Abedalrahman Alshraideh, Bayan Al Fayoumi, Bahaaldeen M. Alshraideh, and Mohammad Alshraideh. Hybrid AI framework for the early detection of heart failure: Integrating traditional machine learning and generative language models with clinical data. *Cureus*, 17(6):e85638, 2025.

Thomas Beaney, Sneha Jha, Asem Alaa, Alexander Smith, Jonathan Clarke, Thomas Woodcock, Azeem Majeed, Paul Aylin, and Mauricio Barahona. Comparing natural language processing representations of coded disease sequences for prediction in electronic health records. *Journal of the American Medical Informatics Association*, 31(7):1451–1462, 2024. ISSN 1067-5027. doi: 10.1093/jamia/ocae091. URL https://academic.oup.com/jamia/article/31/7/1451/7667337.

Suhana Bedi, Yixing Jiang, Philip Chung, Sanmi Koyejo, and Nigam Shah. Fidelity of medical reasoning in large language models. *JAMA Network Open*, 8(8):e2526021, 8 2025. ISSN 2574-3805. doi: 10.1001/jamanetworkopen.2025.26021.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of KDD*, pp. 785–794, 2016.

Hejie Cui, Alyssa Unell, Bowen Chen, Jason Alan Fries, Emily Alsentzer, Sanmi Koyejo, and Nigam Shah. Timer: Temporal instruction modeling and evaluation for longitudinal clinical records, 2025. URL https://arxiv.org/abs/2503.04176.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL*, pp. 4171–4186, 2019.

Tuyen Do, Bichar Dip Shrestha Gurung, Shiva Aryal, Anup Khanal, Sandeep Chataut, Venkataramana Gadhamshetty, Carol Lushbough, and Etienne Z. Gnimpieba. Utilizing XGBoost for the prediction of material corrosion rates from embedded tabular data using large language model. In *2023 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pp. 4497–4499, 2023. doi: 10.1109/BIBM58861.2023.10385544.

Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. DeepLog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17)*, pp. 1285–1298. ACM, 2017. doi: 10.1145/3133956.3134015. URL https://doi.org/10.1145/3133956.3134015.

Justin B. Echouffo-Tcheugui and Andre P. Kengne. Risk models to predict chronic kidney disease and its progression: A systematic review. *PLoS Medicine*, 9(11):e1001344, 2012. doi: 10.1371/journal.pmed.1001344.

Brice Edelman, Hannah Kim, and Jeffrey Skolnick. Scalable discovery and validation of order-specific electronic health record event trajectories for interpretable adverse-outcome risk estimation. *Journal of Biomedical Informatics*, 177:105008, 5 2026. ISSN 1532-0464. doi: 10.1016/j.jbi.2026.105008.

Bahare Fatemi, Mehran Kazemi, Anton Tsitsulin, Karishma Malkan, Jinyeong Yim, John Palowitch, Sungyong Seo, Jonathan Halcrow, and Bryan Perozzi. Test of time: A benchmark for evaluating LLMs on temporal reasoning, 2024. URL https://arxiv.org/abs/2406.09170.

Tairan Fu, Raquel Ferrando, Javier Conde, Carlos Arriaga, and Pedro Reviriego. Why do large language models (LLMs) struggle to count letters?, 2024. URL https://arxiv.org/abs/2412.18626.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.

Aaron Grattafiori, Abhimanyu Dubey, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5-6):602–610, 2005.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2024. URL http://arxiv.org/abs/2312.00752.

Michael Hahn. Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8:156–171, 2020. doi: 10.1162/tacl_a_00306. URL https://aclanthology.org/2020.tacl-1.11/.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Alistair Johnson, Lucas Bulgarelli, Tom Pollard, Brian Gow, Benjamin Moody, Steven Horng, Leo Anthony Celi, and Roger Mark. MIMIC-IV (version 3.1). PhysioNet, 10 2024. URL https://doi.org/10.13026/kpb9-mt58.

Alistair E. W. Johnson, Lucas Bulgarelli, Lu Shen, Alvin Gayles, Ayad Shammout, Steven Horng, Tom J. Pollard, Sicheng Hao, Benjamin Moody, Brian Gow, Li-wei H. Lehman, Leo A. Celi, and Roger G. Mark. MIMIC-IV, a freely accessible electronic health record dataset. *Scientific Data*, 10(1):1, 2023. doi: 10.1038/s41597-022-01899-x.

Matthew James Kadatz, Elizabeth Sunmin Lee, and Adeera Levin. Predicting progression in CKD: Perspectives and precautions. *American Journal of Kidney Diseases*, 67(5):779–786, 2016. doi: 10.1053/j.ajkd.2015.11.007.

Daniel Kaiser, Arnoldo Frigessi, Ali Ramezani-Kebrya, and Benjamin Ricaud. CogniLoad: A synthetic natural language reasoning benchmark with tunable length, intrinsic difficulty, and distractor density. *arXiv preprint arXiv:2509.18458*, 2025.

Sein Kim, Hongseok Kang, Kibum Kim, Jiwan Kim, Donghyun Kim, Minchul Yang, Kwangjin Oh, Julian McAuley, and Chanyoung Park. Lost in sequence: Do large language models understand sequential recommendation? In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, pp. 1160–1171. ACM, 8 2025. ISBN 979-8-4007-1454-2. doi: 10.1145/3711896.3737035.

Anton Klenitskiy, Anna Volodkevich, Anton Pembek, and Alexey Vasilev. Does it look sequential? an analysis of datasets for evaluation of sequential recommendations. In *18th ACM Conference on Recommender Systems*, pp. 1067–1072, 10 2024. doi: 10.1145/3640457.3688195.

Yikuan Li, Shishir Rao, Jose Roberto Ayala Solares, et al. BEHRT: Transformer for electronic health records. *Scientific Reports*, 10:7155, 2020.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach, 2019. URL https://arxiv.org/abs/1907.11692.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

R. Thomas McCoy, Ellie Pavlick, and Tal Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of ACL*, pp. 3428–3448, 2019.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. GSM-Symbolic: Understanding the limitations of mathematical reasoning in large language models. *arXiv preprint arXiv:2410.05229*, 2024.

Timothy Nguyen. Understanding transformers via n-gram statistics, 2024. URL https://arxiv.org/abs/2407.12034.

Davide Placido, Bo Yuan, Jessica X. Hjaltelin, Chunlei Zheng, Amalie D. Haue, Piotr J. Chmura, Chen Yuan, Jihye Kim, Renato Umeton, Gregory Antell, Alexander Chowdhury, Alexandra Franz, Lauren Brais, Elizabeth Andrews, Debora S. Marks, Aviv Regev, Siamack Ayandeh, Mary T. Brophy, Nhan V. Do, Peter Kraft, Brian M. Wolpin, Michael H. Rosenthal, Nathanael R. Fillmore, Søren Brunak, and Chris Sander. A deep learning algorithm to predict risk of pancreatic cancer from disease trajectories. *Nature Medicine*, 29(5):1113–1122, 2023. ISSN 1546-170X. doi: 10.1038/s41591-023-02332-5. URL https://www.nature.com/articles/s41591-023-02332-5.

Freya Pollington, Spiros C. Denaxas, Kezhi Li, Johan H. Thygesen, Georgios Lyratzopoulos, and Becky White. Evaluation of trajectory analysis for disease risk assessment: A scoping review. *Journal of the American Medical Informatics Association*, 33(2):521–535, 2026. ISSN 1527-974X. doi: 10.1093/jamia/ocaf208. URL https://doi.org/10.1093/jamia/ocaf208.

Qwen Team. Qwen2.5: A party of foundation models, 9 2024. URL https://qwenlm.github.io/blog/qwen2.5/.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Qwen Team. Qwen3.8-Max: A new bar for coding and cowork, August 2026. URL https://qwen.ai/blog?id=qwen3.8.

Mohammad Ramezanali, Mo Vazifeh, and Paolo Santi. seqBench: A tunable benchmark to quantify sequential reasoning limits of LLMs, 9 2025.

Laila Rasmy, Yang Xiang, Ziqian Xie, Cui Tao, and Degui Zhi. Med-BERT: Pretrained contextualized embeddings on large-scale structured electronic health records for disease prediction. *npj Digital Medicine*, 4:86, 2021.

Germans Savcisens, Tina Eliassi-Rad, Lars Kai Hansen, Laust Hvas Mortensen, Lau Lilleholt, Anna Rogers, Ingo Zettler, and Sune Lehmann. Using sequences of life-events to predict human lives. *Nature Computational Science*, 4(1):43–56, 12 2023. ISSN 2662-8457. doi: 10.1038/s43588-023-00573-5. URL http://dx.doi.org/10.1038/s43588-023-00573-5.

Artem Shmatko, Alexander Wolfgang Jung, Kumar Gaurav, Søren Brunak, Laust Hvas Mortensen, Ewan Birney, Tom Fitzgerald, and Moritz Gerstung. Learning the natural history of human disease with generative transformers. *Nature*, 647(8088):248–256, 11 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09529-3.

Ching Y. Suen. n-gram statistics for natural language understanding and text processing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):164–172, 1979. doi: 10.1109/TPAMI.1979.4766902.

Navdeep Tangri, Lesley A. Inker, Brett Hiebert, Jenna Wong, David Naimark, David Kent, and Andrew S. Levey. A dynamic predictive model for progression of CKD. *American Journal of Kidney Diseases*, 69(4):514–520, 2017. doi: 10.1053/j.ajkd.2016.07.030.

Irene Teinemaa, Marlon Dumas, Marcello La Rosa, and Fabrizio Maria Maggi. Outcome-oriented predictive process monitoring: Review and benchmark. *ACM Transactions on Knowledge Discovery from Data*, 13(2):17:1–17:57, 2019. ISSN 1556-4681. doi: 10.1145/3301300. URL https://doi.org/10.1145/3301300.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, et al. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

Yitan Zhu, Dehua Bi, Milda Saunders, and Yuan Ji. Prediction of chronic kidney disease progression using recurrent neural network and electronic health records. *Scientific Reports*, 13(1):22091, 2023. doi: 10.1038/s41598-023-49271-2.

Appendix roadmap. Proofs: appendix A; training: appendix B; synthetic results: appendix C; OC analysis appendix D; GPC analysis appendix E; clinical case study appendix F; attention analysis appendix G.

A PROOFS

All probabilities condition on the realized hidden parameters. The OC theorem assumes independent coordinates with full support; its quantitative bound uses the uniform proposal. GPC uses the i.i.d. uniform proposal P_0, whereas KIP uses the dependent-input law Q_0 of Definition 3.5. For any statistic V of X, independent symmetric label noise gives

$$P(Y = 1 \mid V) = \pi + (1 - 2\pi)P(Y^* = 1 \mid V). \tag{6}$$

For $\pi < 1/2$, this affine map preserves constancy and nonconstancy of the posterior. We use this observation throughout instead of repeating the noise calculation in each proof.

A.1 ORACLE PERFORMANCE UNDER LABEL NOISE

Theorem A.1 (Oracle AUC and hard-oracle metrics)**.** *Suppose Y^* is a deterministic function of X, $0 < \rho = P(Y^* = 1) < 1$, and labels are independently flipped with probability $\pi \in [0, 1/2)$. Set $q = P(Y = 1) = \pi + (1 - 2\pi)\rho$. The Bayes-optimal AUC is*

$$\mathrm{AUC}^* = \frac{1}{2} + \frac{1}{2}\left(\frac{(1-\pi)\rho}{q} - \frac{\pi\rho}{1-q}\right) = \frac{1}{2} + \frac{\rho(1-\rho)(1-2\pi)}{2q(1-q)}. \tag{7}$$

For the hard oracle $\widehat{Y} = Y^*$, precision, recall, and F1 are $\mathrm{Prec}_{\mathrm{oracle}} = 1-\pi$, $\mathrm{Rec}_{\mathrm{oracle}} = (1-\pi)\rho/q$, and $F_{1,\mathrm{oracle}} = 2(1-\pi)\rho/[\pi + 2(1-\pi)\rho]$. The latter metrics describe this particular thresholded classifier and are not asserted to be optimal over thresholds.

Proof. The posterior $P(Y = 1 \mid X) = \pi + (1 - 2\pi)Y^*$ orders the two latent classes correctly, so $s = Y^*$ attains Bayes-optimal AUC. Let $a = P(s = 1 \mid Y = 1) = (1 - \pi)\rho/q$ and $c = P(s = 1 \mid Y = 0) = \pi\rho/(1 - q)$. Counting ties with weight $1/2$ gives

$$\mathrm{AUC}(s) = a(1 - c) + \tfrac{1}{2}\{ac + (1 - a)(1 - c)\} = \tfrac{1}{2} + \tfrac{1}{2}(a - c).$$

For the hard oracle, $(\mathrm{TP}, \mathrm{FP}, \mathrm{FN}, \mathrm{TN}) = ((1 - \pi)\rho, \pi\rho, \pi(1 - \rho), (1 - \pi)(1 - \rho))$; substitution gives the remaining formulas. □

For $\pi = 0$, the oracle metrics equal one; for $\rho = 1/2$, they equal $1 - \pi$.

A.2 PROOF OF THEOREM 3.4

Proof. Let $s_{\min}$ and $s_{\max}$ be the lowest- and highest-ranked keys. We show that $P(Y^* = 1 \mid X_1 = a)$ is nonconstant in a.

Case $\mathcal{S} \subsetneq \mathcal{A}$. Choose $a_0 \notin \mathcal{S}$ and couple $\widetilde{X} = (s_{\min}, Z_2, \ldots, Z_n)$ and $\widetilde{X}' = (a_0, Z_2, \ldots, Z_n)$, where the Z_t are independent with the respective laws of X_t. Independence makes these the conditional laws given $X_1 = s_{\min}$ and $X_1 = a_0$. Write C and C' for their compliance events. Any ordered maximal λ-run in $\widetilde{X}'$ either remains unchanged in $\widetilde{X}$ or starts at $1 + \lambda$ and extends left by the minimum-ranked key. The extended run remains ordered and maximal. Thus $C' \subseteq C$. The event

$$W = \{Z_{1+\lambda} \in \mathcal{S}\} \cap \bigcap_{t \in \{2,\ldots,n\} \setminus \{1+\lambda\}} \{Z_t \notin \mathcal{S}\}$$

has positive probability by full support. On W, $\widetilde{X}$ has the ordered maximal run $(1, 1 + \lambda)$, while $\widetilde{X}'$ has only one key and is not compliant. Consequently,

$$P(Y^* = 1 \mid X_1 = s_{\min}) - P(Y^* = 1 \mid X_1 = a_0) = P(C \setminus C') \geq P(W) > .$$

Case $\mathcal{S} = \mathcal{A}$. Maximal runs are the residue classes modulo λ. Let $R_1 = (1, 1+\lambda, \ldots, 1+(L-1)\lambda)$, $L = 1 + \lfloor (n-1)/\lambda \rfloor \geq 2$. Let O_1 denote that R_1 is ordered, and B that a different maximal run is ordered. These events depend on disjoint coordinates, so

$$P(Y^* = 1 \mid X_1 = a) = P(B) + (1 - P(B))P(O_1 \mid X_1 = a).$$

Here $P(B) < 1$, since otherwise $\rho = 1$. Also $|\mathcal{A}| \geq 2$ by nondegeneracy. Given $X_1 = s_{\min}$, O_1 requires only that the remaining entries of R_1 be nondecreasing. Given $X_1 = s_{\max}$, it requires that all those entries equal $s_{\max}$. The latter event is contained in the former; containment is strict with positive probability because all remaining entries can instead equal $s_{\min}$. Thus the two compliance posteriors differ.

In both cases, Y^* and X_1 are dependent, hence $I(Y^*; X_1) > 0$. Equation (6) gives $I(Y; X_1) > 0$ as well. □

Quantitative bound and sampling scope.

Corollary A.2 (Uniform witness lower bound)**.** *Under the OC theorem with $\mathcal{S} \subsetneq \mathcal{A}$ and i.i.d. uniform inputs, let $m = |\mathcal{S}|$, $\ell = |\mathcal{A}|$, and $\delta(a, b) = P(Y^* = 1 \mid X_1 = a) - P(Y^* = 1 \mid X_1 = b)$. Then*

$$\delta(s_{\min}, a_0) \geq \frac{m}{\ell} \left(1 - \frac{m}{\ell}\right)^{n-2}. \tag{8}$$

The noisy-label bound is $(1 - 2\pi)$ times this bound by (6).

Proof. Evaluate $P(W)$ in the preceding proof using independence and uniformity, then apply (6). □

Remark A.3 (Scope of the bound). For independent non-uniform inputs, put $\alpha_t = P(X_t \in \mathcal{S})$. The same proof gives the posterior-gap bound $\alpha_{1+\lambda} \prod_{t \in \{2,\ldots,n\} \setminus \{1+\lambda\}} (1 - \alpha_t)$. There is no positive bound depending only on (m, ℓ, n) under full support alone: the total key probability can approach zero. These are bounds on posterior gaps, not direct lower bounds on mutual information.

Remark A.4 (Label-dependent sampling). If a population is reweighted by a binary label $L \in \{Y^*, Y\}$, $Q(x, l) \propto r_l P(x, l)$ with $r_0, r_1 > 0$, then $Q(X \mid L) = P(X \mid L)$ and $Q(L = 1 \mid V) = h_\beta(P(L = 1 \mid V))$, where $h_\beta(u) = \beta u / [1 + (\beta - 1)u]$ and $\beta = r_1/r_0$. Strict monotonicity preserves independence or dependence on V, but does not preserve numerical posterior gaps. AUC against the reweighted label L is unchanged for a fixed score. In our OC/GPC procedure, stratification uses Y^* before noise is added; any changed latent prevalence must therefore be used in the oracle formula. The uniform witness bound describes the proposal law and does not automatically transfer numerically to a resampled dataset. Finite-sample fluctuations do not alter these population statements.

A.3 Proof of Theorem 3.6

Proof. Every allowed sequence under Q_0 has probability $\binom{n}{m}^{-1} (m!)^{-1} (\ell - m)^{-(n-m)}$. For fixed distinct keys a, b, swapping their positions defines a probability-preserving involution $\tau_{a,b}$ without fixed points. A transposition reverses permutation sign, so it flips inversion parity while preserving the full count vector $C(X)$. Pairing completions within each count class yields $P(Y^* = 1 \mid C(X) = c) = 1/2$. Hence the label is balanced and $I(Y^*; C(X)) = 0$.

Now fix J with $|J| \leq m - 2$ and a possible view x_J. At least two key identities are absent from this view. Choose two such identities as a function of x_J, independently of the completion. They occur outside J in every completion, so their transposition preserves x_J and pairs equally probable completions with opposite labels. Therefore $P(Y^* = 1 \mid X_J = x_J) = 1/2$ and $I(Y^*; X_J) = 0$.

For tightness, set $J = [m - 1]$ and condition on $H = \{X_i = \kappa^{-1}(i),\ i \in [m - 1]\}$. This event has probability $(n - m + 1)/(\binom{n}{m} m!) > 0$. The remaining key has rank m and occurs later, so the rank permutation is increasing and $Y^* = 0$ on H. Its unconditional distribution is balanced; thus $I(Y^*; X_{[m-1]}) > 0$. Equation (6) preserves all independence statements and the strict dependence for $\pi < 1/2$. □

Ranking identifiability. Replacing κ by $\sigma \circ \kappa$ multiplies permutation sign by $\operatorname{sgn}(\sigma)$. Every even σ preserves all labels and every odd σ flips them. Hence labels cannot identify the full ranking beyond this equivalence, even with the key set fixed.

A.4 Proof of Theorem 3.8

Proof. Write K for the GPC key set, $B_t = \mathbf{1}\{X_t \in K\}$, and $p = |K|/|\mathcal{A}|$. Under P_0, the B_t are i.i.d. Bernoulli(p). The number of even key counts has the same parity as $|K| + \sum_t B_t$, so

$$Y^* = 1 \oplus (|K| \bmod 2) \oplus T, \qquad T = \bigoplus_{t=1}^{n} B_t.$$

For nonempty $J \subsetneq [n]$, let $U = \bigoplus_{j \in J} B_j$, $V = \bigoplus_{j \notin J} B_j$, and $s = n - |J| \geq 1$. Then $T = U \oplus V$, U is determined by X_J, and V is independent of X_J. Independence gives

$$1 - 2P(V = 1) = \mathbb{E}[(-1)^V] = (1 - 2p)^s, \qquad q_s := P(V = 1) = \frac{1 - (1 - 2p)^s}{2}.$$

Thus $P(T = 1 \mid X_J)$ equals q_s when $U = 0$ and $1 - q_s$ when $U = 1$. If $p = 1/2$, both values equal $1/2$, proving independence. If $p \neq 1/2$, they differ for every finite s, and both values of U have positive probability because $p \in (0, 1)$ and J is nonempty. This proves strict dependence. Since Y^* is a fixed bit flip of T, the same conclusions hold for Y^*.

For fixed $p \in (0, 1)$ and fixed finite J, $q_s \to 1/2$ as $n \to \infty$. The conditional probabilities converge uniformly over the finite support of X_J; continuity of finite-alphabet entropy gives $I(Y^*; X_J) \to 0$. Equation (6) preserves the finite-n independence/dependence statements, and data processing gives $I(Y; X_J) \leq I(Y^*; X_J) \to 0$. □

B Training and Reproducibility

B.1 Data, optimization, and model configurations

The main OC/GPC/KIP experiments use $n = 20$, $\ell = 26$, six hidden keys, and 400K training sequences. The training, validation, and test proportions are $80/10/10$. OC uses $\lambda = 7$ and latent prevalence .293, with $\pi = 0$ or .3; the core GPC and KIP tasks use $\pi = 0$. Tables 3 4 5 reports means and standard deviations over three runs with three different seeds.

Table 1: Shared settings for the core neural experiments. All rows use AdamW.

Setting	Pretrained decoders	BERT-family	Random initialization
Learning rate	2×10^{-5}	2×10^{-5}	Table 2
Batch size / max epochs	16/20	16/20	64/30
Stopping statistic	Validation loss	Validation loss	Validation F1
Patience	3	3	5
Schedule	Linear, 6% warmup	Linear, 6% warmup	—
LoRA $(r, \alpha, \text{dropout})$	$(8, 16, 0.1)$	$(8, 16, 0.1)$	Not used
QLoRA		Not used	Not used
Token limit	200	200	Task length

LoRA (Hu et al., 2021) targets `q_proj`, `k_proj`, `v_proj`, and `o_proj` in causal decoders, and `query`, `key`, and `value` in BERT. QLoRA (Dettmers et al., 2023) with 4-bit NF4 weight quantization, double quantization, BF16 computation, and FP32 trainable adapters and classification heads. The classification head is trained outside LoRA. Core randomly initialized models use cross-entropy loss. Transformer uses four heads, GELU, learned positional embeddings, and masked mean pooling; the LSTM uses the final hidden state. RNNTransformer denotes a BiLSTM whose outputs are projected into a Transformer encoder.

Small decoder. The randomly initialized Llama-1M uses hidden width 128, four layers, four attention heads, two key/value heads, and feed-forward width 512.

Tree baselines. XGBoost-basic uses the length-n vector of integer-mapped letters. XGBoost-LLM uses 2048-dimensional mean-pooled last-layer embeddings from Llama-3.2-1B. Both use learning rate .05, maximum depth 6, row/column subsampling .8, L_1/L_2 penalties $.5/1.5$, minimum child

Table 2: Default (D) and optimized (O) configurations for the core randomly initialized models. FFN denotes feed-forward width; hybrid layers are recurrent plus Transformer layers.

Model	Variant	Embed.	Hidden / FFN	Layers	Dropout	Learning rate
Transformer	D	64	256	2	.3	5×10^{-4}
	O	64	128	3	.3	5×10^{-4}
LSTM	D	32	64	2	.3	10^{-3}
	O	32	128	2	.2	2×10^{-3}
RNNTransformer	D	64	64 / 256	$1+2$	.3	5×10^{-4}
	O	128	32 / 256	$1+3$	.3	10^{-4}

weight 2, and split penalty .1, with histogram tree construction. Training allows 200 boosting rounds with validation-log-loss patience 10 and positive class weight $N_{\text{negative}}/N_{\text{positive}}$.

k-gram and lag-aware baselines. Three local-pattern references: (i) a contiguous k-gram classifier that estimates $\hat{\mathbb{P}}(Y=1 \mid g) = \#(g, Y=1)/\#(g)$ and averages over the test sequence's k-grams; (ii) a λ-aware pair-count classifier, logistic regression with L_2 regularization on the 26×26-dimensional feature $g_{ab}^{(\lambda)}$ with λ given, which controls for whether the task can be solved by a single linear readout once the hidden offset is known – where

$$g_{ab}^{(d)}(X) = \sum_{t=1}^{n-d} \mathbf{1}\{X_t = a,\ X_{t+d} = b\}, \qquad a, b \in \mathcal{A}, \quad d \in \{1, \ldots, n-1\};$$

and (iii) a λ-agnostic baseline that stacks $g_{ab}^{(\lambda')}$ across all candidate offsets $\lambda' \in \{1, \ldots, n-1\}$ (dimension $(n-1)\times676 = 12\,844$) and lets an L_1-regularized logistic regression or a boosted tree rediscover λ from data. Baseline (iii) is the fair "is the encoder's advantage about discovering λ?" test.

B.2 Sequence representations and objectives

Pretrained models receive `Sequential events:` $x_1\ x_2\ \cdots\ x_n$, tokenized by their native tokenizer. Causal decoders append the literal sentinel `Outcome (0 or 1):`; the true label is never appended. Their last non-padding hidden state feeds a two-layer classification head $d_{\text{model}} \to d_{\text{model}}/2 \to 2$, with Tanh and dropout .1. The decoder objective is

$$\mathcal{L} = \mathcal{L}_{\text{LM}}(\text{prompt}) + \mathcal{L}_{\text{CE}}(\widehat{y}_{\text{head}}, y),$$

with equal weights. Predictions use the head softmax, not next-token logits. Encoder classifiers use the classification loss.

Randomly initialized models (for cold start) embed letter indices $1, \ldots, 26$. Transformer models add learned positional embeddings, initialized from a standard normal distribution. Padding is zero for indexed sequences; pretrained inputs are padded/truncated at the token limit in Table 1. When reporting threshold-dependent metrics, the threshold is selected on validation data from $\{.10, .11, \ldots, .89\}$ to maximize F1; AUC uses unthresholded scores.

Compute. The core experiments use an NVIDIA H200 (141GB), with bfloat16 and gradient checkpointing for large models.

C Extended synthetic results

This appendix provides the numerical results of our synthetic framework.

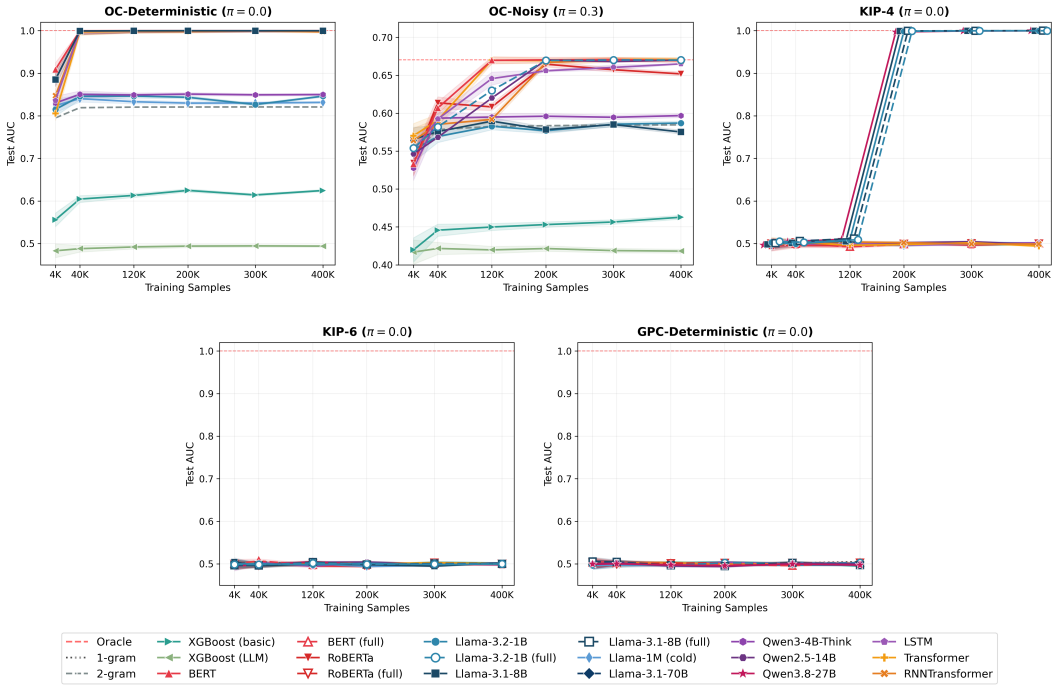

Figure 2: AUC learning curves for all evaluated models. Top: OC-Deterministic ($\pi = 0$), OC-Noisy ($\pi = 0.3$), KIP-4 ($\pi = 0$); bottom: KIP-6 ($\pi = 0$), hidden-subset GPC-Deterministic ($\pi = 0$). Threshold-dependent metrics are in Tables 3, 4, 5. Red dashed lines mark the oracle AUC (1.0, or $\approx .67$ under label noise); gray curves are the k-gram baselines (2-gram on OC, 1-gram on KIP and GPC). Dashed curves with hollow markers denote full fine-tuning, solid curves LoRA/QLoRA. Encoders and recurrent models reach the ceiling on OC while several decoders plateau below it; on KIP-4 only decoder-only models (Qwen3.8-27B, Llama-3.1-8B, Llama-3.2-1B) reach the oracle, from 200K examples, with all other models at chance (coinciding curves are slightly shifted horizontally). On KIP-6 and GPC-Deterministic, every architecture remains at chance. Shading denotes variability across seeds; single-run results are unshaded.

Table 3: Full classification performance on OC datasets (100% training data). Standard deviations in parentheses.

Model	AUC	F1	Prec	Rec
OC-Deterministic				
Oracle	1.000	1.000	1.000	1.000
2-gram	.821	.672	.593	.775
XGBoost (basic)	.625 (.012)	.457 (.015)	.516 (.010)	.409 (.018)
XGBoost (LLM)	.494 (.010)	.399 (.012)	.353 (.008)	.459 (.018)
BERT	.999 (.001)	.999 (.001)	.999 (.001)	1.000 (.001)
RoBERTa	.999 (.001)	.999 (.001)	.999 (.001)	1.000 (.001)
Llama-3.2-1B	.846 (.015)	.695 (.018)	.623 (.020)	.785 (.022)
Llama-3.2-1B‡	.993 (.001)	.972 (.010)	.992 (.000)	.953 (.019)
Llama-3.1-8B	.995 (.003)	.984 (.008)	.976 (.010)	.993 (.007)
Llama-1M (cold)	.832 (.012)	.686 (.015)	.605 (.018)	.791 (.020)
Qwen3-4B-Think	.850 (.014)	.701 (.016)	.620 (.017)	.807 (.020)
Qwen3.8-27B (xhigh)	.999 (.000)	.999 (.001)	.999 (.001)	1.000 (.000)
LSTM	.999 (.001)	.994 (.003)	.995 (.003)	.994 (.003)
Transformer	.999 (.001)	.992 (.002)	.985 (.005)	1.000 (.001)
RNNTransformer†	.997 (.002)	.992 (.003)	.985 (.005)	1.000 (.000)
OC-Noisy				
Oracle	.670	.577	.700	.491
2-gram	.585	.592	.420	1.000
XGBoost (basic)	.463 (.010)	.418 (.012)	.437 (.008)	.400 (.015)
XGBoost (LLM)	.418 (.011)	.453 (.014)	.437 (.008)	.470 (.021)
BERT	.669 (.006)	.592 (.004)	.420 (.004)	1.000 (.000)
RoBERTa	.652 (.008)	.592 (.003)	.420 (.003)	1.000 (.000)
Llama-3.2-1B	.587 (.008)	.592 (.005)	.420 (.005)	1.000 (.000)
Llama-3.2-1B‡	.670 (.003)	.593 (.002)	.468 (.021)	.814 (.056)
Llama-3.1-8B	.575 (.008)	.591 (.005)	.425 (.006)	.968 (.010)
Llama-3.1-8B‡	.668 (.005)	.548 (.005)	.690 (.006)	.455 (.010)
Qwen3-4B-Think	.597 (.007)	.591 (.004)	.423 (.005)	.979 (.008)
Qwen2.5-14B	.670 (.003)	.586 (.007)	.476 (.005)	.763 (.010)
Qwen3.8-27B (xhigh)	.670 (.000)	.590 (.001)	.421 (.000)	.986 (.002)
LSTM	.665 (.003)	.562 (.020)	.660 (.002)	.489 (.031)
Transformer	.671 (.001)	.579 (.003)	.701 (.015)	.492 (.012)
RNNTransformer†	.671 (.005)	.578 (.005)	.683 (.010)	.501 (.005)

† RNNTransformer denotes the BiLSTM–Transformer.
‡ Fully fine-tunded.

Table 4: Full classification performance on KIP datasets (100% training data). Standard deviations in parentheses.

Model	AUC	F1	Prec	Rec
KIP-4: 400K training examples				
Oracle	1.000	1.000	1.000	1.000
2-gram	.501	.472	.498	.449
XGBoost (basic)	.507 (.004)	.494 (.004)	.502 (.003)	.486 (.010)
XGBoost (LLM)	.505 (.002)	.515 (.004)	.501 (.001)	.531 (.009)
BERT	.501 (.003)	.664 (.002)	.498 (.001)	.996 (.003)
BERT‡	.500 (.002)	.665 (.001)	.498 (.001)	.998 (.002)
RoBERTa	.504 (.004)	.664 (.002)	.498 (.002)	.997 (.003)
RoBERTa‡	.500 (.003)	.665 (.001)	.498 (.001)	.998 (.001)
Llama-3.2-1B	.505 (.004)	.664 (.002)	.498 (.002)	.996 (.004)
Llama-3.2-1B‡	.999 (.001)	.998 (.002)	.998 (.002)	.998 (.002)
Llama-3.1-8B	1.000 (.001)	.999 (.001)	.999 (.001)	.999 (.001)
Llama-3.1-8B‡	.999 (.001)	.999 (.001)	.999 (.001)	.999 (.001)
Qwen2.5-14B	.999 (.001)	.998 (.001)	.999 (.001)	.998 (.001)
Qwen3.8-27B (xhigh)	1.000 (.001)	1.000 (.001)	1.000 (.001)	1.000 (.001)
LSTM	.999 (.001)	.999 (.001)	.999 (.001)	.999 (.001)
Transformer	.500 (.003)	.664 (.001)	.498 (.001)	.998 (.002)
RNNTransformer†	.500 (.002)	.665 (.000)	.498 (.000)	1.000 (.000)
KIP-6: 400K training examples				
Oracle	1.000	1.000	1.000	1.000
2-gram	.499	.474	.501	.450
XGBoost (basic)	.504 (.000)	.486 (.055)	.504 (.001)	.476 (.098)
XGBoost (LLM)	.501 (.004)	.492 (.015)	.501 (.004)	.483 (.025)
BERT	.498 (.003)	.666 (.002)	.501 (.001)	.996 (.003)
BERT‡	.500 (.002)	.667 (.001)	.501 (.001)	.998 (.002)
RoBERTa	.501 (.004)	.667 (.002)	.501 (.002)	.997 (.003)
RoBERTa‡	.500 (.003)	.667 (.001)	.501 (.001)	.998 (.001)
Llama-3.2-1B	.500 (.004)	.667 (.002)	.501 (.002)	.996 (.003)
Llama-3.2-1B‡	.500 (.003)	.667 (.001)	.501 (.001)	.998 (.002)
Llama-3.1-8B	.502 (.003)	.667 (.001)	.501 (.001)	.998 (.002)
Llama-3.1-8B‡	.499 (.004)	.667 (.002)	.501 (.001)	.997 (.003)
Qwen2.5-14B	.501 (.004)	.667 (.002)	.501 (.001)	.997 (.002)
Qwen3.8-27B (xhigh)	.500 (.002)	.667 (.001)	.501 (.001)	.999 (.001)
LSTM	.500 (.002)	.644 (.022)	.501 (.001)	.904 (.083)
Transformer	.502 (.003)	.667 (.001)	.501 (.001)	.998 (.002)
RNNTransformer†	.500 (.002)	.667 (.002)	.501 (.003)	1.000 (.001)

† RNNTransformer denotes the BiLSTM–Transformer.
‡ Fully fine-tunded.

Table 5: Full classification performance on the GPC dataset (100% training data). Standard deviations in parentheses.

Model	AUC	F1	Prec	Rec
GPC: reported values and explicitly simulated completions				
Oracle	1.000	1.000	1.000	1.000
1-gram	.505	.668	.501	1.000
2-gram	.498	.476	.501	.453
XGBoost (basic)	.495 (.004)	.490 (.005)	.497 (.003)	.484 (.006)
XGBoost (LLM)	.501 (.003)	.639 (.005)	.502 (.002)	.881 (.008)
BERT	.503 (.002)	.668 (.002)	.501 (.002)	1.000 (.001)
BERT‡	.501 (.003)	.666 (.002)	.502 (.001)	.991 (.004)
RoBERTa	.501 (.003)	.668 (.002)	.501 (.002)	1.000 (.001)
RoBERTa‡	.500 (.002)	.665 (.002)	.500 (.001)	.993 (.004)
Llama-3.2-1B	.497 (.005)	.668 (.001)	.501 (.002)	1.000 (.001)
Llama-3.2-1B‡	.503 (.004)	.668 (.002)	.503 (.002)	.995 (.003)
Llama-3.1-8B	.500 (.003)	.668 (.002)	.501 (.002)	1.000 (.001)
Llama-3.1-8B‡	.501 (.003)	.666 (.002)	.502 (.001)	.989 (.005)
Llama-1M (cold)	.503 (.003)	.668 (.002)	.501 (.003)	1.000 (.001)
Llama-3.1-70B	.502 (.004)	.668 (.001)	.501 (.001)	1.000 (.001)
Qwen3-4B-Think	.500 (.002)	.668 (.001)	.501 (.000)	1.000 (.001)
Qwen2.5-14B	.500 (.018)	.666 (.003)	.501 (.002)	.991 (.004)
Qwen3.8-27B	.500 (.003)	.668 (.001)	.502 (.001)	.997 (.002)
LSTM	.502 (.003)	.668 (.002)	.501 (.002)	1.000 (.001)
Transformer	.501 (.003)	.668 (.002)	.501 (.002)	1.000 (.001)
RNNTransformer†	.501 (.002)	.668 (.002)	.501 (.003)	1.000 (.001)

† RNNTransformer denotes the BiLSTM–Transformer.
‡ Fully fine-tunded.

D OC ANALYSIS

D.1 SENSITIVITY SWEEP

We run the stochastic sweep over $m \in \{3, 6, 10\}$ and $\lambda \in \{1, 3, 5, 7, 9, 10\}$, yielding 18 configurations. For each configuration we train BERT-base, a single-layer LSTM, a two-layer Transformer encoder, and Llama-3.2-1B with LoRA from scratch on 400K training sequences, using the same hyperparameters as Appendix B.1; we report mean AUC over three seeds and divide by the theoretical oracle AUC^* from Theorem A.1 to obtain the $\mathrm{AUC}/\mathrm{AUC}^*$ ratio used in Table 6.

BERT and LSTM stay within 0.05 of the ceiling across the sweep.

Three patterns emerge from Table 6. First, BERT and LSTM are essentially at the oracle ceiling throughout the stochastic sweep. Second, Llama-3.2-1B is consistently below the encoder/recurrent models, with the largest gaps in the $m = 10$ regime where AUC^* is highest. Third, the small Transformer is sensitive to dense-key regimes: it degrades at $m = 10$ for small and medium lags but recovers at larger lags. The main-paper setting $(m = 6, \lambda = 7)$ is therefore a medium-difficulty point rather than a cherry-picked worst case; harder $m = 10$ regimes amplify the encoder/recurrent advantage.

D.2 BASELINE LADDER ON STANDARD EVALUATION

We report test AUC across seven feature families and two classifiers (L_2-regularized logistic regression with the `lbfgs` solver, and XGBoost with histogram tree construction) on OC-Deterministic and OC-Noisy, with $200,000$ training and $50,000$ evaluation sequences (single run). The families are the full 26-dimensional letter counts, the 6-dimensional key-letter counts, residue-class counts modulo λ, the aggregated λ-aware pair counts $g_{ab}^{(\lambda=7)}$ of Appendix B, 6×6 key-only lag-pair counts, position-aware lag-pair indicators (sparse, $(n - \lambda) \times 2 = 8,788$ dimensions), and 6^3 key-only lag-trigram counts. Table 7 reports the results. The λ-aware pair feature saturates both datasets with a linear classifier (.997 on OC-Deterministic and .671 on OC-Noisy, against the oracle .670). Position-aware indicators add nothing over aggregated counts, and restricting the pair counts to key letters hurts, because part of the signal lives in correlations with non-key letters. Residue-class counts are the only other family on which XGBoost improves substantially over logistic regression (.938 versus .818 on OC-Deterministic).

D.3 λ-AGNOSTIC LADDER

To test whether the encoder's advantage comes from knowing λ, we stack pair counts across all offsets $\lambda' \in \{1, \ldots, n-1\}$, giving a 12 844-dimensional feature. L_1 logistic regression on this feature space reaches AUC $= 0.997$ on OC-Deterministic (recovering $\lambda = 7$ from data; the per-λ weight norm is 303 at $\lambda = 7$ versus ≤ 48 elsewhere) and AUC $= 0.632$ on OC-Noisy (where label noise prevents clean lag identification, with the top-weight lag drifting to $\lambda = 19$). XGBoost on the same features closes the gap on OC-Noisy, reaching AUC $= 0.667$, essentially matching the best encoder.

D.4 HELD-OUT-RULE GENERALIZATION

To check whether the λ-aware pair feature carries a transferable algorithm or a per-rule linear weighting, we generate 6 synthetic datasets with random $(\mathcal{S}, \kappa, \lambda)$ drawn from the same family as the main OC-Noisy dataset. Leave-one-rule-out: train logistic regression on 5 rules, evaluate on the held-out rule and on a held-out split of the same rule. The λ-aware pair-count feature reaches AUC $= 1.00$ within-rule and AUC $= 0.50$ across-rule; full letter-count features reach AUC $= 0.77$ within-rule and AUC $= 0.49$ across-rule. The linear ceiling is therefore carried by a per-rule weighting, not by a transferable procedure.

E GPC ANALYSIS

For the GPC-Deterministic case, we decompose its failure by revealing successive levels of information about the hidden subset $\mathcal{S}$. Given raw tokens with $\mathcal{S}$ hidden, both logistic regression and a 64-unit

Table 6: Sensitivity analysis in the stochastic setting ($\pi = 0.3$). Each cell reports AUC/AUC*, where AUC* is the per-configuration theoretical ceiling. A ratio of 1.00 indicates the model reaches the Bayes-optimal bound.

m	λ	AUC*	BERT	LSTM	Transformer	Llama-3.2-1B
3	1	.602	1.00	1.00	0.94	0.96
3	3	.595	1.00	1.00	0.94	0.93
3	5	.589	1.00	1.00	0.94	0.95
3	7	.581	1.00	0.98	0.96	0.95
3	9	.574	1.00	1.00	0.97	0.96
3	10	.570	1.00	0.99	1.00	0.95
6	1	.683	1.00	1.00	0.97	0.88
6	3	.679	1.00	1.00	0.99	0.86
6	5	.675	1.00	1.00	1.00	0.86
6	**7**	**.670**	**1.00**	**0.99**	**1.00**	**0.88**
6	9	.666	1.00	0.99	1.00	0.88
6	10	.664	1.00	0.99	1.00	0.89
10	1	.699	1.00	1.00	0.76	0.78
10	3	.699	1.00	1.00	0.80	0.79
10	5	.698	1.00	0.99	0.83	0.81
10	7	.698	1.00	0.96	0.87	0.85
10	9	.697	1.00	0.96	0.98	0.85
10	10	.696	1.00	0.98	0.99	0.87

Bold: baseline configuration (m=6, λ=7) from the main paper.

Table 7: OC baseline ladder: test AUC of L_2 logistic regression (LR) and XGBoost (XGB) on seven feature families, $200,000$ training and $50,000$ evaluation sequences, single run. The λ-aware pair row is the .997/.671 quoted in Section 5. The last row is the λ-agnostic all-lag stacking of Appendix D.3 (L_1 logistic regression and XGBoost; sample sizes given there).

		OC-Deterministic		OC-Noisy	
Feature family	dim	LR	XGB	LR	XGB
Letter counts	26	.816	.815	.584	.579
Key-letter counts	6	.773	.772	.575	.573
Residue-class counts mod λ	182	.818	.938	.582	.608
λ-aware pair counts $g_{ab}^{(\lambda=7)}$	676	.997	.997	.671	.670
Key-only lag-pair counts	36	.846	.846	.629	.630
Position-aware lag-pair indicators	8,788	.996	.996	.638	.665
Key-only lag-trigram counts	216	.563	.559	.513	.513
All-lag stacked pair counts (L_1 LR / XGB)	12,844	.997	.998	.632	.667

MLP remain at chance (AUC $= 0.50$). Revealing the per-position membership bit $b_t = \mathbf{1}\{X_t \in \mathcal{S}\}$ (a 20-dimensional binary input) allows the same MLP to reach AUC $= 0.997$ in under one minute of training while logistic regression remains at chance. Revealing the 6-dimensional per-key count vector or per-key parity bits reaches AUC $= 0.997$ and 1.000 respectively with the MLP. Revealing only the scalar total key count yields AUC $= 0.981$ with the MLP.

E.1 Reproduction of a binary-parity reference protocol

As defined in Definition 3.7, the GPC mechanism is based on counting and checking whether the count is even. A known setting in which small transformer encoders learn parity is binary-vocabulary parity on short sequences (Hahn, 2020). To locate our main-paper regime relative to this boundary, we reproduce that setting exactly, a 2-layer Transformer encoder ($d_{\text{model}} = 64$, 8 heads, vocabulary size 2, $n = 10$) trained for 1000 steps with a `[CLS]` readout, and then sweep sequence length and data regime to characterize the transition.

Experimental grid. We vary (i) sequence length $n \in \{10, 20\}$ and (ii) training-data regime: *streaming* (a fresh uniformly random batch of 1024 sequences at every gradient step) or *fixed* (a training pool of 100,000 sequences sampled once and drawn from with replacement at every step). All runs use identical architecture, optimizer (Adam, $\mathrm{lr} = 10^{-3}$), batch size (1024), step count (1000), and are repeated for seeds $\in \{0, 1, \ldots, 9\}$. Test AUC is measured on a held-out set of 10,000 fresh sequences. Because outcomes at $n = 10$ are bimodal (a run either converges to AUC ≈ 1.0 or remains at chance), we report both the mean $\pm$ std and the success rate, defined as the fraction of seeds reaching AUC > 0.9.

Table 8: Test AUC for binary parity ($\ell = 2$, $|\mathcal{S}| = 1$) across sequence length and training-data regime. Values are mean (std) over 10 seeds, following the format of Table 5. Success rate counts seeds reaching AUC > 0.9. At $n = 10$, the same three seeds ($\{2, 4, 8\}$) fail in both regimes, confirming that the outcome is controlled by model initialization rather than the data-sampling regime.

Regime	n	AUC	Success rate
streaming	10	.850 (.242)	7/10
fixed	10	.850 (.241)	7/10
streaming	20	.503 (.004)	0/10
fixed	20	.499 (.005)	0/10

Three observations from Table 8. (i) Streaming versus fixed-size training makes no material difference at either sequence length. A fixed pool of 10^5 binary sequences of length 10 contains the full $2^{10} = 1024$-element input space many times over, and even at $n = 20$ the fixed-size and streaming curves coincide at chance; data-regime effects are therefore not the load-bearing variable. (ii) At $n = 10$, 7 of 10 seeds reach AUC ≈ 1.0 and 3 remain at chance, and the identity of the failing seeds ($\{2, 4, 8\}$) is identical across the two regimes. Since our seed controls model initialization and only the data pipeline differs across regimes, this is strong evidence that the failure mode is initialization-driven, consistent with the known sensitivity of parity learning to initialization in small transformers. (iii) At $n = 20$, all 20 runs across both regimes remain at AUC ≈ 0.50, matching the main-paper result in the vocabulary sweep at $\ell = 2$, $|\mathcal{S}| = 1$.

Takeaway. The protocol demonstrates that the balanced ablation at $n = 20, |\mathcal{S}| = \ell/2$ lies on the failing side of a length-driven phase boundary, while the $n = 10$ reference point lies on the learnable side. This suggests that sequence length and class balance are important contributors, not a methodological artifact of our training pipeline, consistent with length-dependent parity difficulty in standard-depth self-attention architectures (Hahn, 2020).

E.2 Vocabulary-size sweep at the balanced regime

Setup. We sweep the alphabet size $\ell \in \{2, 4, 8, 16, 26\}$ at the balanced parity regime $|\mathcal{S}| = \max(1, \lfloor \ell/2 \rfloor)$, so that $\ell/|\mathcal{S}| = 1/2$ at every ℓ and Theorem 3.8 gives exact k-local purity for every $k < n$. Sequence length is fixed at $n = 20$. For each ℓ we generate 100K train, 20K validation, and 20K test sequences with i.i.d. uniform sampling over the ℓ-letter alphabet.

Models and training. Three architectures: BERT-base (110M, fine-tuned with LoRA, 20 epochs, batch 32); a 2-layer Transformer encoder ($d_{\text{model}} = 64$, 8 heads, $\sim$170K parameters); and a single-layer LSTM. The two from-scratch architectures are trained for 30 epochs with batch 64, AdamW, learning rate 10^{-3}. Three seeds per cell.

Table 9: Test AUC on parity at the balanced regime $|\mathcal{S}| = \max(1, \lfloor \ell/2 \rfloor)$, $n = 20$. Mean (std) over 3 seeds.

Arch	ℓ=2	ℓ=4	ℓ=8	ℓ=16	ℓ=26
BERT	.508 (.008)	.497 (.001)	.501 (.004)	.499 (.003)	.498 (.008)
LSTM	.501 (.003)	.499 (.002)	.499 (.003)	.503 (.001)	.505 (.002)
Transformer	.505 (.000)	.501 (.003)	.500 (.002)	.504 (.003)	.501 (.005)

Takeaway. Every architecture sits at chance for every alphabet size in the balanced regime, including the binary case $\ell = 2$, $|\mathcal{S}| = 1$. This is consistent with Theorem 3.8: at $\ell/|\mathcal{S}| = 1/2$ the task is k-locally pure for every $k < n$, so none of our tested architectures trained end-to-end without curriculum at n = 20 finds a useful learning signal. The length-driven phase boundary documented for binary parity in Appendix E.1 (success at $n = 10$, failure at $n = 20$) is therefore the relevant explanation for the main-paper failure, not a vocabulary effect. Results are reported in Table 9.

E.3 Subset-size sweep at $\ell = 26$

Setup. We fix $\ell = 26$ and $n = 20$ and sweep $|\mathcal{S}| \in \{2, 4, 6, 8, 10, 13\}$, so that $p = |\mathcal{S}|/|\mathcal{A}|$ ranges from $2/26 \approx 0.077$ to $13/26 = 0.500$ (the exactly balanced point). Per cell: 100K train, 20K validation, 20K test. Same architectures and training schedule as Appendix E.2, three seeds per cell.

Table 10: Test AUC on parity at $\ell = 26$, $n = 20$ as $|\mathcal{S}|$ varies. Mean (std) over 3 seeds. $|\mathcal{S}| = 13$ is the exactly balanced point $p = 1/2$ at which Theorem 3.8 gives exact k-local purity.

Arch	$\|\mathcal{S}\|$=2	$\|\mathcal{S}\|$=4	$\|\mathcal{S}\|$=6	$\|\mathcal{S}\|$=8	$\|\mathcal{S}\|$=10	$\|\mathcal{S}\|$=13
BERT	1.000 (.000)	.497 (.003)	.501 (.008)	.499 (.003)	.505 (.002)	.498 (.008)
LSTM	1.000 (.000)	.501 (.001)	.505 (.005)	.504 (.003)	.502 (.001)	.505 (.002)
Transformer	1.000 (.000)	.504 (.001)	.501 (.001)	.500 (.002)	.505 (.003)	.501 (.005)

Takeaway. The transition is sharp and located at $|\mathcal{S}| = 2$, not at the balanced point. At $|\mathcal{S}| = 2$ all three architectures reach AUC ≈ 1.000 ($p = 2/26 \approx 0.077$ provides large finite-n local leakage, which bounded-attention models pick up in full); from $|\mathcal{S}| = 4$ onward every architecture drops to chance and stays there through $|\mathcal{S}| = 13$. The main-paper regime ($|\mathcal{S}| = 6$, $p \approx 0.231$) sits on the failing side of this transition, and approaching the balanced point does not change the outcome. The result is consistent with the bounded-local statement of Theorem 3.8: for $p \neq 1/2$ finite-n leakage is positive but vanishes for fixed local view as n grows; here $n = 20$ is already large enough that any $|\mathcal{S}| \geq 4$ regime is no longer learnable end-to-end at our training budget. Results are reported in Table 10.

E.4 Sequence-length sweep at the balanced regime

Setup. We fix the balanced regime $\ell = 26$, $|\mathcal{S}| = 13$ ($p = 1/2$, exact k-local purity by Theorem 3.8) and sweep $n \in \{10, 15, 20, 30\}$. The $n = 20$ column reuses the anchor runs from the $|\mathcal{S}| = 13$ cell of Appendix E.3; for $n \in \{10, 15, 30\}$ we generate fresh 100K train / 20K validation / 20K test splits at $\ell = 26$, $|\mathcal{S}| = 13$. Same architectures and training schedule as Appendix E.3, three seeds per cell.

Table 11: Test AUC on parity at the balanced regime $\ell = 26$, $|\mathcal{S}| = 13$ as n varies. Mean (std) over 3 seeds.

Arch	n=10	n=15	n=20	n=30
BERT	.501 (.001)	.501 (.007)	.498 (.008)	.503 (.004)
LSTM	.498 (.004)	.499 (.001)	.505 (.002)	.503 (.003)
Transformer	.500 (.002)	.498 (.001)	.501 (.005)	.497 (.004)

Takeaway. At the balanced large-alphabet regime every architecture sits at chance across all four lengths, including $n = 10$. This is the key contrast with the binary regime ($\ell = 2, |\mathcal{S}| = 1$) of Appendix E.1, where the same 2-layer Transformer reaches AUC ≈ 1.0 for $7/10$ seeds at $n = 10$ and decays to chance only at $n = 20$. The balanced-and-large-alphabet task appears harder under our tested training protocol: at $\ell = 26$, $|\mathcal{S}| = 13$ even the short-sequence regime is not learned end-to-end at our training budget. Theorem 3.8 removes the local shortcuts captured by our bounded-view analysis: reducing n from 20 to 10 is not enough to compensate when the task is exactly k-locally pure. Results are reported in Table 11.

E.5 Length-based curriculum protocol and full per-seed results

Setup. We test a two-stage length curriculum on main-paper parity ($\ell = 26$, $K = \mathcal{S}_{\text{parity}}$, $|\mathcal{A}| = 6$). Stage 1 trains a model from random initialization on $n = 10$ sequences. Stage 2 loads the matched-seed Stage 1 checkpoint and fine-tunes on $n = 20$. The shape mismatch in the positional embedding ($[1, 10, d] \to [1, 20, d]$) is handled by copying the first 10 positions from the Stage 1 checkpoint and randomly initializing the remaining 10; all other parameters are copied directly. Each stage uses a freshly generated dataset of 100K train / 20K validation / 20K test sequences.

Architectures and hyperparameters.

- **Transformer:** 2 layers, 8 heads, $d_{\text{model}} = 64$, $d_{\text{ff}} = 512$, dropout 0.3 (168 642 parameters at $n = 10$, 169 282 at $n = 20$). Adam optimizer, learning rate 10^{-3}, batch size 64, 30 epochs, early-stopping patience 3 on validation loss.
- **LSTM:** same training schedule as the Transformer.
- **BERT-base:** 110M parameters, fine-tuned with LoRA; 20 epochs, batch size 32.

We run 3 seeds per architecture per stage.

Per-seed results. Table 12 reports test AUC for every (architecture, stage, seed) cell. Three patterns are visible. First, the LSTM is fully insensitive to initialization on this task: all 6 runs reach AUC ≥ 0.999. Second, the Transformer's Stage 2 outcome is perfectly determined by its Stage 1 outcome: seed 1, which fails to learn $n = 10$ parity (AUC $= 0.502$), also fails after curriculum at $n = 20$ (AUC $= 0.497$); seeds 0 and 2, which succeed at $n = 10$, also succeed at $n = 20$ (AUC ≈ 1.0 in both stages). The init-driven bimodality documented for $n = 10$ binary parity in Appendix E.1 is therefore inherited unchanged by the curriculum. Third, BERT-base does not benefit from the protocol: all 6 BERT runs sit at AUC ≈ 0.50, and in particular Stage 1 itself is unsolved (no seed clears 0.51 at $n = 10$). The curriculum cannot transfer a representation that the warm-start checkpoint does not contain, so the Stage 2 outcome is at chance for every seed.

Table 12: Length-based curriculum learning per-seed test AUC. Stage 1: cold-start training on $n = 10$. Stage 2: warm-start fine-tuning on $n = 20$ from the matched-seed Stage 1 checkpoint.

Architecture	Seed	Stage 1 (n=10)	Stage 2 (n=20)
Transformer	0	.999	.999
	1	.502	.497
	2	.999	.999
LSTM	0	1.000	1.000
	1	1.000	1.000
	2	1.000	1.000
BERT	0	.507	.493
	1	.504	.502
	2	.508	.500

F Clinical case study: chronic kidney progression

Cohort definition. We use MIMIC-IV v3.1 diagnosis records (Johnson et al., 2024; 2023). ICD diagnosis codes are mapped to Clinical Classifications Software (CCS) categories (Agency for Healthcare Research and Quality, 2025). CKD is defined using ICD-10 N18.1–N18.5 and ICD-9 585.1–585.5. ESRD-related endpoint codes include ICD-10 N18.6, Z99.2, Z49.* and ICD-9 585.6, V45.1, V56.*. Positives are patients whose first ESRD-related code occurs in an admission strictly after the first CKD-coded admission; their sequences are truncated before the ESRD-coded admission. Negatives are CKD patients with no ESRD-related code in the observed record. Patients whose first CKD and ESRD-related codes occur in the same admission are excluded to avoid same-visit label leakage. Patients with fewer than 10 diagnosis codes after preprocessing are also excluded.

Table 13: Cohort summary and sanity checks for the MIMIC-IV CKD→ESRD audit.

Category	Quantity	Value
Cohort	Patients	14,813
	Progressors, $Y = 1$	1,146
	Non-progressors, $Y = 0$	13,667
	Progressor prevalence	7.7%
	CCS vocabulary size	293
Sequence length	Median / mean / max codes	49 / 82 / 2,396
	Median length, $Y = 1$ / $Y = 0$	54 / 49
Admissions	Median / mean admissions	3 / 4.9
	Patients with ≥ 2 admissions	76%
	Patients with ≥ 3 admissions	58%
Overlap checks	Shared CCS categories	265 / 293
	Vocabulary Jaccard overlap	0.90
	CCS categories exclusive to $Y = 1$	0
	Cosine similarity of code frequencies	0.956
	Spearman correlation of code frequencies	0.968

Across admissions, diagnoses are ordered by admission time; within admissions, they are ordered by diagnosis sequence number, which reflects billing/discharge order rather than true event time. See Table 13 for an overview.

Shortcut controls. We apply three shortcut controls. First, for each contiguous k-gram g, we estimate $\hat{P}(Y = 1 \mid g)$ on the training set and score each test patient by averaging over the patient's observed k-grams. Second, we train order-invariant bag-of-codes Logistic Regression and XGBoost models over CCS counts. Third, for sequence models, we compare ordered training/evaluation against a shuffled condition that randomly permutes each patient's diagnosis sequence while preserving the patient-level multiset of codes. See Table 14 for the results.

Discriminative unigram content. In the MIMIC-IV audit, each token X_t is a CCS diagnosis category. A k-gram is therefore a contiguous block of k CCS tokens, and the $k = 1$ case corresponds to a single diagnosis category. To understand why unigram and bag-of-codes baselines are strong, we inspect patient-level CCS prevalence differences between progressors and non-progressors. Table 15 reports selected enriched unigrams. These are descriptive content checks, not validated clinical trajectories.

The cohort-defining CKD category itself, CCS_158, is highly prevalent in both groups and therefore only weakly enriched. In contrast, the more enriched unigrams are shared diagnosis categories that occur more frequently among progressors. This supports the interpretation that the task is not driven by progressor-exclusive codes, but by frequency differences among shared clinical content.

Interpretation. The audit shows that CKD→ESRD prediction, while is clinically temporal, is not strongly order-informational under this diagnosis-code representation. Unigrams and bag-of-codes baselines are already strong, and sequential-model gains largely survive order destruction. This should not be interpreted as evidence that CKD progression is clinically non-temporal, nor as evidence that sequence models are inadequate. Rather, it indicates that this representation contains strong non-temporal shortcuts: diagnosis identity, frequency, and co-occurrence carry enough signal that temporal order is not decisive for prediction.

Limitations and data access. MIMIC-IV is an ICU-centered EHR dataset and may not reflect outpatient CKD progression trajectories. The audit uses diagnosis codes only; richer longitudinal variables such as eGFR, laboratory values, medications, dialysis timing, visit intervals, and outpatient follow-up may contain decisive temporal information. Within-admission order reflects billing/discharge order rather than clinical event time. All MIMIC-IV analyses are performed under credentialed access. We release cohort-construction and preprocessing code, but not raw MIMIC-IV data or patient-level derived sequences.

Table 14: Compact audit summary for MIMIC-IV CKD→ESRD. k-gram rows report mean $\pm$ standard deviation over five 80/20 stratified splits. Bag-of-codes and neural rows use the 60/20/20 neural-model split.

Audit	Model/control	Ordered AUC	Shuffled AUC	Auxiliary	Interpretation
k-gram	1-gram	0.823 ± 0.015	–	F1 0.356	Code identity/frequency carries strong signal.
	2-gram	0.801 ± 0.016	–	F1 0.341	Adding contiguous order does not improve over unigrams.
	3-gram	0.576 ± 0.015	–	F1 0.165	Longer local patterns become sparse.
	4–7 gram	0.479–0.500	–	F1 0.144–0.146	Higher-order k-grams collapse to chance.
Bag-of-codes	Logistic Reg.	0.809	0.809	$\Delta = 0$	Strong order-invariant baseline.
	XGBoost	0.803	0.803	$\Delta = 0$	Strong order-invariant baseline.
Sequence models	Transformer	0.860	0.844	$\Delta = +0.016$	Most of the gain survives order destruction.
	BiLSTM	0.820	0.823	$\Delta = -0.003$	No positive ordered-vs-shuffled gain.
	BERT-base	0.826	0.843	$\Delta = -0.017$	No positive ordered-vs-shuffled gain.
	LSTM	0.500	0.467	failed	Did not converge reliably on long sequences.

Table 15: Selected CCS unigrams enriched among CKD→ESRD progressors. Frequencies are patient-level prevalences. The log ratio is $\log_2(\text{freq}_{Y=1}/\text{freq}_{Y=0})$.

Code	Description	$Y = 1$	$Y = 0$	$\log_2$ ratio
CCS_156	Nephritis, nephrosis, renal sclerosis	30.3%	9.2%	+1.71
CCS_87	Retinal detachments, retinopathy	18.1%	7.4%	+1.29
CCS_215	Genitourinary congenital anomalies	5.5%	2.0%	+1.47
CCS_158	Chronic kidney disease	99.7%	89.9%	+0.15

G EXPLORATORY ATTENTION INSPECTION

We inspect attention as a descriptive diagnostic across ordered compliance (OC), key-inversion parity (KIP), and CKD→ESRD prediction. Attention allocation does not establish the causal contribution of a head or identify a unique internal algorithm. Our mechanism claims therefore rely on the feature-space and input audits, with attention providing complementary observations.

OC: models and measurement. We analyze BERT-base and RoBERTa-base with full fine-tuning, Llama-3.1-8B with LoRA, and Qwen3.8-27B with QLoRA on OC-Deterministic, plus a separate Qwen3.8-27B QLoRA fit on OC-Noisy. Each task-specific fit starts from the pretrained base, without KIP fine-tuning. The encoders and Llama use the same 200K training subset; Qwen uses 400K. Both Qwen fits were capped at six completed epochs, with validation loss selecting epoch 5 for deterministic OC and epoch 6 for noisy OC.

All five audits use the same 1,000 held-out sequences, sampled proportionally by the latent clean label (seed 9950): 292 compliant and 708 non-compliant. The corrected OC rule uses key letters $\{W, D, Q, J, X, U\}$ and lag $\lambda = 7$. For each key-source event, we measure attention to key events at distance λ, other key events, and non-key events. Lag membership is positional; it does not require that the pair satisfy the secret rank ordering. Self-attention and prompt/special-token attention are

recorded separately. Target-token masses are summed within events and query tokens are averaged within source events. We respect each model's visibility mask: bidirectional for encoders and causal for decoders.

For each example, layer, and head, we divide the mean attention mass for an eligible target category by its expected mass under uniform attention over visible tokens. We then average these ratios over eligible examples and heads. Thus enrichment above one denotes allocation above that visibility-matched baseline. Queries without visible targets in a category are omitted; the lag category has support in 465 of the 1,000 sequences. Categories consequently have different eligibility sets and their averaged curves are not an additive partition. Primary summaries include both outcomes, without selecting examples by prediction correctness. For hybrid Qwen3.8-27B, we inspect only its 16 standard attention blocks; recurrent blocks are excluded.

OC: localized lag enrichment and input controls. Figure 3 shows localized enrichment for lag-connected keys. Using one-based physical layer numbers, head-averaged lag enrichment peaks at 3.18 in BERT layer 7, 1.42 in RoBERTa layer 12, and 2.24 in Llama layer 12. Both Qwen fits peak at physical layer 28: 6.16 for deterministic OC and 4.18 for noisy OC. These are descriptive maxima. Enrichment varies across depth and heads; it is not a consistently elevated signal throughout the network. Normalization accounts for target availability, but does not control differences in architecture, tokenization, training size, or adaptation method.

We additionally perturb the same 200 fixed sequences while preserving letter counts. Oracle-verified label-changing permutations were found for 192 sequences within 64 proposals each. All four deterministic-OC models correctly change their predictions on all 192 permutations and preserve their predictions on all 200 distractor-only swaps. Noisy Qwen correctly follows 187 of the 192 label changes and likewise has no prediction flips on distractor swaps. These diagnostics use threshold 0.5 and latent clean labels, including for noisy OC. The conditional 192-example subset is not an unbiased performance estimate. The controls establish sensitivity to order beyond letter counts on the tested inputs; they do not establish the causal necessity of the displayed attention patterns. All four deterministic fits classify the 1,000 sampled sequences correctly at this threshold, so these audits provide no missed-compliant group for a correctness comparison.

KIP: key-order sensitivity without a unique attention signature. We inspect 22 saved checkpoints on 1,000 fixed held-out examples per task: BERT-base and RoBERTa-base with full fine-tuning and LoRA on 4/6; Llama-3.2-1B with full fine-tuning on KIP-4/6; and Llama-3.1-8B LoRA, Qwen2.5-14B QLoRA, and Qwen3.8-27B QLoRA on 4 at 4K, 40K, 120K, and 200K training examples. Their attention profiles show model-dependent key routing rather than a common preference for inversion pairs. Figure 4 shows the four successful M4 checkpoints

MIMIC-IV CKD attention analysis. For BERT-base, RoBERTa-base, Llama-3.1-8B, and Qwen3.8-27B, we compare separately trained ordered and shuffled models on the same 1,000 held-out patients, including 77 progressors. We inspect the classifier readout query (CLS for encoders, the last unmasked token for decoders); Qwen again contributes only its 16 standard attention blocks. We aggregate repeated CCS-code occurrences, average code masses over heads and audited layers, then normalize within each patient to sum to one over CCS codes, excluding prompt mass. The mean paired overlap, defined as one minus total-variation distance between these code distributions, is 0.850 for BERT, 0.848 for RoBERTa, 0.703 for Llama, and 0.674 for Qwen. This describes shared allocation to clinical content, without identifying causal risk factors or proving order invariance. The comparison changes both model weights and input order. Moreover, encoders truncate at 512 tokens, affecting 191 BERT and 188 RoBERTa sampled patients, whereas both decoders retain full histories. Shuffling can change which events remain visible after truncation, precluding an equal-context architecture comparison. Figure 5 visualizes the layer-wise allocation to diagnosis events and the dispersion of readout attention for both conditions.

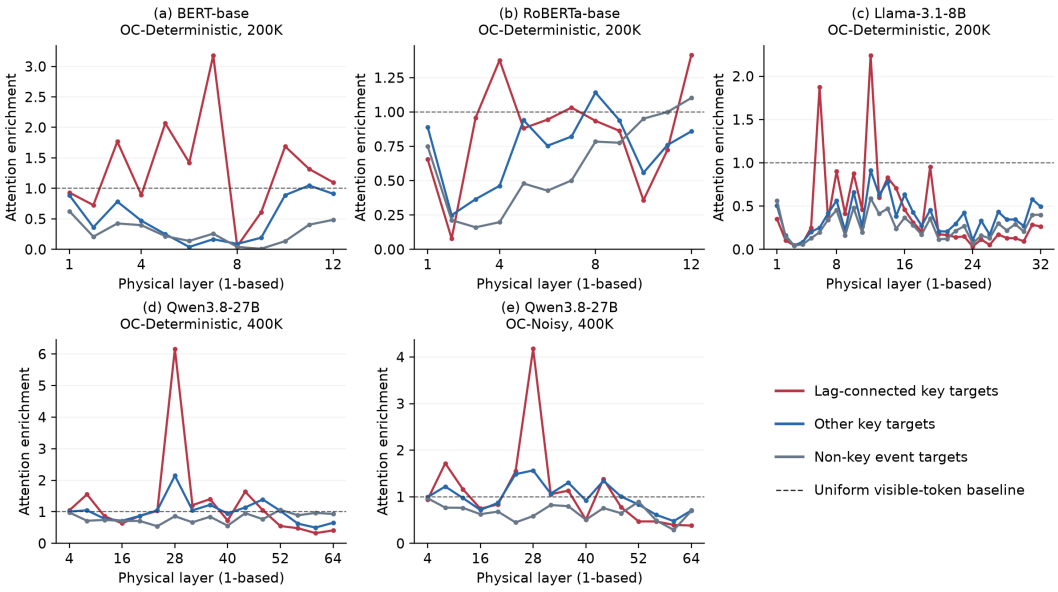

Figure 3: OC attention enrichment across physical layers for the five completed audits. Red: lag-connected key targets; blue: other key targets; gray: non-key event targets. Dashed lines indicate uniform attention over visible tokens. Curves average per-example enrichment over eligible examples and heads, using all sampled outcomes. Target categories have different eligibility sets. Layer numbering is one-based and panels use separate y-axis ranges. Qwen panels include standard attention blocks only. Localized lag enrichment is visible, but the profiles are descriptive and do not constitute a controlled architecture comparison or a causal attribution.

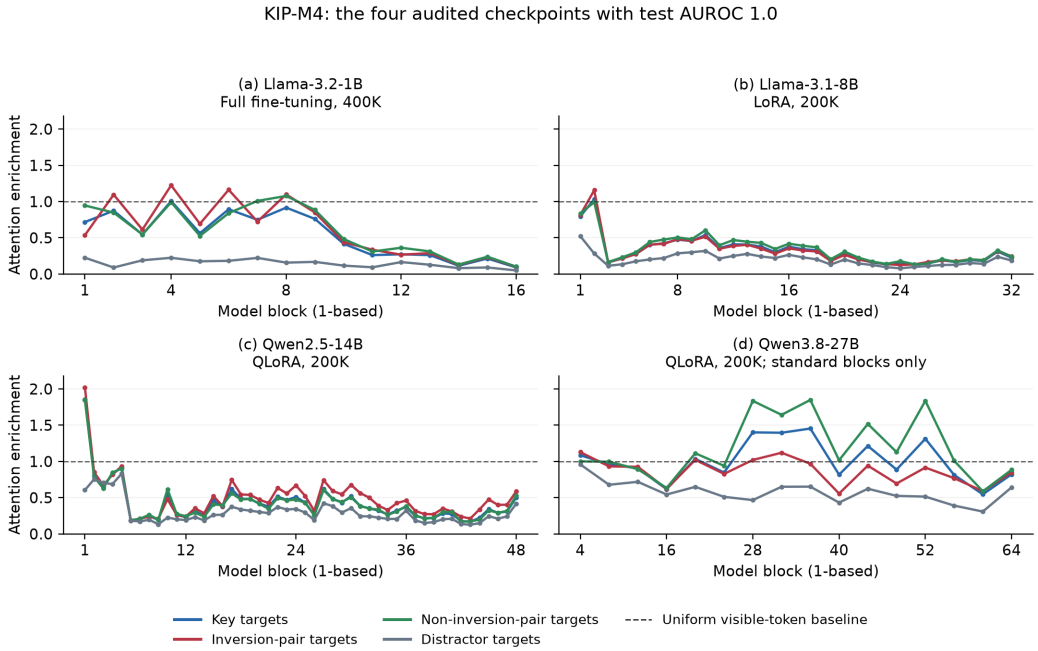

Figure 4: Attention enrichment for the four audited KIP-M4 checkpoints with test AUROC 1.0. Llama-3.2-1B uses full fine-tuning on 400K examples; the larger decoders use LoRA or QLoRA on 200K. Blue: key targets excluding the source event; red: inversion-pair targets; green: non-inversion-pair targets; gray: distractors. Dashed lines denote uniform attention over all visible tokens, including self and prompt/special tokens. Curves average per-example mass-to-baseline ratios over eligible examples and then heads. Key and distractor curves use all 1,000 sampled examples; inversion and non-inversion curves use 971 and 961 eligible examples, respectively. Categories overlap and conditional averaging differs, so the curves are not additive. Model-block indices are one-based. Only standard attention blocks are shown for Qwen3.8-27B; connecting lines do not measure its intervening recurrent blocks. These selected-checkpoint profiles describe model-dependent routing and do not identify an internal parity algorithm or provide a controlled architecture comparison.

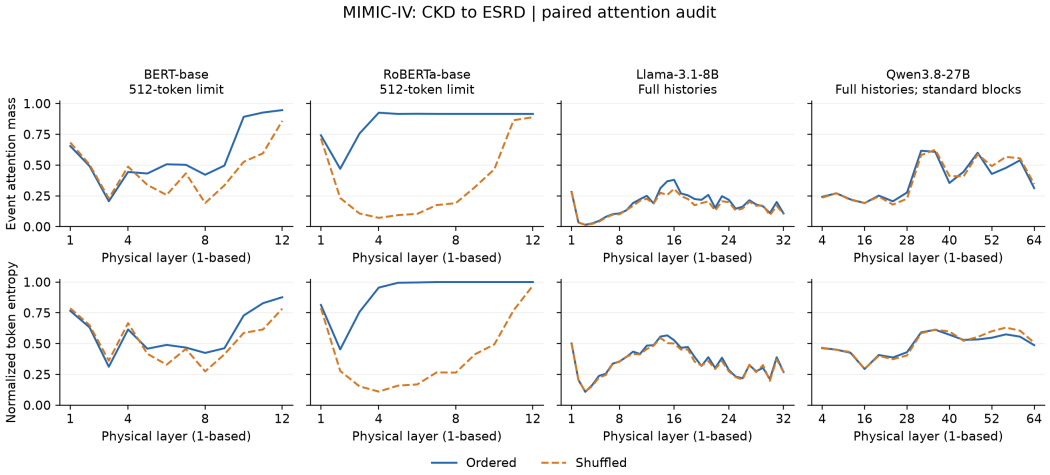

Figure 5: MIMIC-IV CKD→ESRD readout attention in separately trained ordered (solid blue) and shuffled (dashed orange) models on the same 1,000 held-out patients. Top: attention mass assigned to complete visible diagnosis events. Bottom: token-attention entropy normalized by the logarithm of the visible token count; higher values indicate more diffuse allocation. Both quantities are averaged over heads within each patient and then over patients at each physical layer. Encoders use the CLS query and a 512-token window; decoders use the last unmasked query and full histories. Qwen includes only its 16 standard attention blocks. Layers are numbered from one. Conditions differ in model weights and input order, and encoder truncation can change visible content. The profiles are descriptive, not a controlled comparison of architectures or causal effects of shuffling.