TOKEN-AXIS PREDICTIVE DECOMPOSITION: WHAT INFORMATION DOES EACH TOKEN ADD TO THE HIDDEN STATES OF A LANGUAGE MODEL?

Anonymous authors
Paper under double-blind review

ABSTRACT

Interpretability research on large language models commonly relies on the decomposition along the layer axis provided by the architecture, $h_t^l = h_t^{l-1} + \Delta h_t^l$. Along the token axis, however, the architecture provides no such decomposition. We decompose the hidden state h_t at position t into its expectation conditioned only on the past hidden states, $\hat{h}_t = \mathbb{E}[h_t \mid h_{<t}]$, and the residual $\varepsilon_t = h_t - \hat{h}_t$. Since the true conditional expectation cannot be computed, we approximate it with several predictors of different architectures and also examine how much the observed properties depend on the choice of predictor. At five layers of Pythia-410m-deduped, we evaluate this decomposition from three perspectives: temporal structure, linguistic information, and effect on the output. First, using a Transformer as the predictor reduces the squared error only slightly in the middle layers, but when the predictor aggregates the entire past, the autocorrelation of $\hat{h}_t$ exceeds that of h_t, while that of ε_t is nearly zero. Second, when linear probes trained on h_t are applied unchanged to both components, attributes determined by the context (e.g., the document source) can be read out from $\hat{h}_t$ and token-local attributes (e.g., noun number) from ε_t; in deeper layers, the component that carries the attributes shifts toward $\hat{h}_t$. Third, the KL divergence of the next-token distribution when h_t is replaced with $\hat{h}_t$ is generally smaller for better predictors, and the output is more sensitive to reducing ε_t than to increasing it by the same amount. These results suggest that the predictive decomposition along the token axis, like the decomposition along the layer axis, can serve as a unit for analyzing internal representations.

1 INTRODUCTION

To understand the behavior of large language models (LLMs), researchers have been analyzing their internal representations (Ferrando et al., 2024). Such analyses often divide hidden states into smaller units and examine what each unit represents and how it contributes to the output. Among these, the decomposition along the layer axis is widely used as a unit of analysis (Elhage et al., 2021; Geva et al., 2022; Csordás et al., 2025). In the residual stream, $h_t^l = h_t^{l-1} + \Delta h_t^l$ holds by construction (vertical arrows in Figure 1). One can therefore examine the component Δh_t^l added by layer l, or ablate it to measure its effect.

Along the token axis, in contrast, the architecture provides no such decomposition (horizontal arrows in Figure 1). The hidden state h_t at position t (hereafter we fix the layer and omit l) mixes a component already predictable from the context with a component added by the token x_t. Therefore, even if some information is found in h_t, one cannot tell whether it was added by that token or was already predictable from the preceding context.

We therefore ask whether, as along the layer axis, h_t can be split into "the component already predictable" and "the component added at that position." The expectation conditioned only on the past hidden states, $\hat{h}_t = \mathbb{E}[h_t \mid h_{<t}]$, does not depend on x_t and corresponds to the average value that the hidden state at position t takes given the context $h_{<t}$. The residual from the hidden state actually observed at position t, $\varepsilon_t = h_t - \hat{h}_t$, can then be regarded as the component added by that

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.

token (§3). The true conditional expectation cannot be computed, but since it equals the prediction that minimizes the squared error, it can be approximated with a trained predictor.

Using this approximation, we study the decomposition at five layers of Pythia-410m-deduped (Biderman et al., 2023) and show that the predictive decomposition along the token axis, like the decomposition along the layer axis, could serve as a unit for analyzing internal representations. Our specific contributions are as follows.

We examine whether this decomposition is meaningful from three perspectives: temporal structure, linguistic information, and functional effect on the output.

- **Temporal structure**: The autocorrelation of ε_t is nearly zero, and the autocorrelation of $\hat{h}_t$ is higher than that of h_t. The latter is pronounced for predictors that aggregate $h_{<t}$ (§5).
- **Linguistic information**: Attributes determined by the context can be read out from $\hat{h}_t$, and token-local attributes from ε_t. In deeper layers, the component that carries the attributes shifts toward $\hat{h}_t$ (§6).
- **Functional effect**: As predictor quality improves, the KL divergence when h_t is replaced with $\hat{h}_t$ generally decreases. The output is more sensitive to reducing ε_t than to increasing it by the same amount (§7).

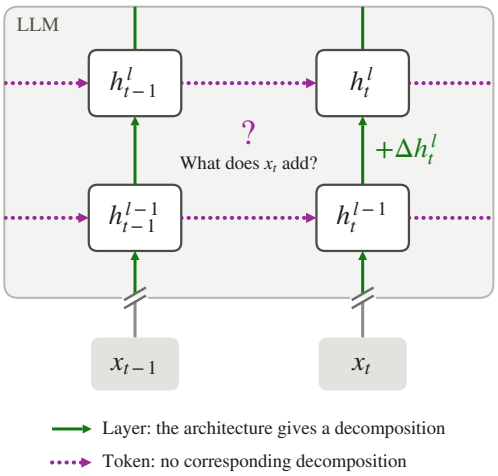

Figure 1: Hidden states of an LLM arranged by layer (vertical) and token position (horizontal). Along the layer axis, the architecture gives the decomposition $h^l_t = h^{l-1}_t + \Delta h^l_t$, but along the token axis there is no corresponding decomposition.

2 Related work

Decomposing representations along the token axis. The closest work to ours is Temporal Feature Analysis (TFA; Lubana et al., 2026), which decomposes hidden states into a component that can be explained by the context and a newly added component. However, TFA selects the context component from past states using the current state h_t as the query, so this component is an explanation conditioned on h_t. In contrast, our $\hat{h}_t$ is the conditional expectation before seeing h_t, and ε_t corresponds to the component that could not be known before seeing the token. Therefore, replacing h_t with $\hat{h}_t$ is an operation that removes this component (§7).

Interpretability along the token axis. Work in this direction falls into two broad groups. The first quantifies the contribution of each position as a measure (Abnar & Zuidema, 2020; Kobayashi et al., 2020; 2021; Ferrando et al., 2022; Mohebbi et al., 2023; Modarressi et al., 2023; Oh & Schuler, 2023). What we ask is not the contribution of each position but how much of the current hidden state was already predictable from the past context. The second describes structure along the token axis. Song & Zhong (2026) decompose hidden states into positional and contextual components, and Tamkin et al. (2020) decompose them by frequency along the token axis, showing that low frequencies carry topic information and high frequencies carry part-of-speech information. Hosseini & Fedorenko (2023) showed that trajectories along the token axis become straighter in deeper layers. These works characterize this structure but do not separate representations into the part that was predictable and the part added by the token.

Representation analysis with inductive biases over token positions. Other work learns features by assuming structure along the token axis. Temporal SAEs (Bhalla et al., 2026) encourage high-level features to be consistent across adjacent positions, and Persistent SAEs (Luo et al., 2026) learn how long each feature persists. Song et al. (2025) identify temporal causal structure among fea-

tures. These works use structure along the token axis to learn features and do not aim to decompose representations. We do not impose smoothness along the token axis as a constraint, yet the autocorrelation of $\hat{h}_t$ turns out to be higher than that of h_t, and that of ε_t nearly zero (§5.2). Note that these works and TFA all learn dictionaries of sparse features, whereas we work with dense representations in order to test the predict-and-decompose framework itself.

Prediction from past states. Predicting the current state only from past states has also been used for purposes other than decomposing representations. PHi (Herrmann et al., 2025) converts hidden states into latent variables with a VAE, predicts the current latent variable only from the latent variables at past positions, and uses the KL divergence between this prediction and the actual latent variable as a measure of the information added by the token at each position. Among speculative decoding methods (Leviathan et al., 2023; Chen et al., 2023), some predict the next hidden state from past hidden states to accelerate LLM inference. EAGLE (Li et al., 2024) points out that past hidden states alone do not determine the next hidden state (feature uncertainty); this undetermined component corresponds to our ε_t. FIRP (Wu et al., 2024) inserts predicted hidden states into intermediate layers, which is the same operation as our intervention in §7.

3 Predictive decomposition along the token axis

We write the hidden state of a large language model at layer l and position t as $h_t \in \mathbb{R}^d$, the sequence from position 1 to t as $h_{\leq t}$, and the sequence before t as $h_{<t}$. We denote by x_t the token input at position t. Since the decomposition is defined for each layer, we omit the index of the analyzed layer l.

In a Transformer with causal attention (Vaswani et al., 2017), given the computation up to position $t-1$, the hidden state h_t at position t is a deterministic function of the input token x_t.

Before x_t is observed, h_t is uncertain, and this uncertainty can be expressed by the conditional distribution $p(h_t \mid h_{<t})$. Because the context constrains which tokens can come next, it narrows down this distribution. We take the expectation of this distribution as the component determined by the context alone:

$$\hat{h}_t := \mathbb{E}[h_t \mid h_{<t}], \qquad \varepsilon_t := h_t - \hat{h}_t, \qquad h_t = \hat{h}_t + \varepsilon_t.$$

Here $\hat{h}_t$ is defined as a function of $h_{<t}$ alone and **does not depend on** x_t. The only new information that enters at position t beyond the context is x_t, so ε_t is the component that could not be predicted from the context, that is, the component attributable to the token input at position t. Strictly speaking, ε_t is the component that could not be predicted from $h_{<t}$ before observing the current token, and it is not the entire causal contribution of x_t. Much of the effect of a token that is easy to predict from the context falls into $\hat{h}_t$. In this paper, we treat ε_t as the contribution of the token in this sense. Below, we call $\hat{h}_t$ the **predictive component** and ε_t the **innovation component**. This split has the same form as splitting a signal into the component predicted from the past and the unpredicted component (the innovation) in time-series analysis and filtering, and we adopt this terminology (Revach et al., 2022). It is also similar in form to predictive coding, which splits a signal into a component predicted from the past and a prediction error (Lotter et al., 2017; Millidge et al., 2021).

The conditional expectation $\mathbb{E}[h_t \mid h_{<t}]$ equals the minimizer of the squared error in predicting h_t from $h_{<t}$:

$$\mathbb{E}[h_t \mid h_{<t}] \;=\; \arg\min_f \; \mathbb{E}\Big[\, \big\| h_t - f(h_{<t}) \big\|^2 \,\Big].$$

This equality directly connects the definition of the decomposition to a procedure for approximating it. The true conditional expectation cannot be computed without knowing the distribution, but a predictor trained to minimize the squared error approaches $\hat{h}_t$, which is the optimum of this objective. We approximate $\hat{h}_t$ with a predictor $f_\theta(h_{<t})$ that takes past hidden states as input. Because this is an approximation, we need to check how much the observed properties depend on the choice of f_θ, so we perform the same measurements with several predictors of different architectures (§4).

The decomposition in this section has the same additive form as the decomposition along the layer axis, $h^l = h^{l-1} + \Delta h^l$, with $\hat{h}_t$ corresponding to h^{l-1} and ε_t to Δh^l. Since the decomposition is exact and both terms lie in the same space as h_t, dropping ε_t and passing $\hat{h}_t$ to the subsequent computation yields a computation with the token's contribution removed (§7).

4 EXPERIMENTAL SETUP

Model and data. We analyze Pythia-410m-deduped. From the deduplicated Pile (Gao et al., 2020; Biderman et al., 2023), we take the first 128 tokens of 200k documents that have at least 128 tokens. Position t is thus counted from the beginning of the document. We extract hidden states at five layers at depths of 0 / 25 / 50 / 75 / 100% (L00 / L06 / L12 / L18 / L24; L00 is the output of the token embedding and L24 is the output of the last block). At position 1, $h_{<1}$ is empty and $\hat{h}_1$ reduces to the constant $\mathbb{E}[h_1]$, so all measurements are made at $t \geq 2$.

Splits. We split the 200k sequences into train 160k / val 20k / test 20k. The predictors, the standardization coefficients, and the baseline of R^2 are fit on the training split, and checkpoints are selected on the validation split; all reported numbers are measured on the test split.

Standardization. In the residual stream, a few specific dimensions take extremely large values at a small number of positions (Sun et al., 2024), so if squared errors are summed in the original coordinates, the sum is dominated by those few dimensions (Timkey & van Schijndel, 2021). Large differences in scale across dimensions also make optimizing the predictors harder. We therefore standardize each dimension by subtracting its median m_i and dividing by its median absolute deviation (MAD) s_i, and write the result as $\tilde{h}_t$. Predictors are trained and R^2 is measured in this space; when we look at quantities in the original coordinates, such as the autocorrelation (§5) and the intervention (§7), we map back with the same coefficients. Appendix B explains why we use the median and MAD.

Predictors. We vary the predictors f_θ along two axes. The first axis is how much of the past is used as input. We use predictors whose input is only the previous state $\tilde{h}_{t-1}$ and predictors whose input is all past states $\tilde{h}_{<t}$. The second axis is capacity, which we vary within each group. For the predictors that take only the previous state, we use three: copy, affine, and MLP. The copy predictor returns $\tilde{h}_{t-1}$ as is (no parameters). The affine predictor applies an affine transformation to $\tilde{h}_{t-1}$. The MLP predictor adds to $\tilde{h}_{t-1}$ the output of a two-layer MLP ($d \to 4d \to d$) applied to it. For the predictors that take all past states, we use Transformers. We stack 1 / 2 / 4 blocks, each of which aggregates all of $\tilde{h}_{<t}$ with causal attention and then applies an MLP (Transformer $\times 1$ / $\times 2$ / $\times 4$, abbreviated as Trf$\times 1$ / $\times 2$ / $\times 4$). The number of heads, the MLP hidden size, and the activation function match those of the LLM (details in Appendix A).

R^2. We define R^2 in the standardized space as a single value pooled over all dimensions and positions:

$$R^2 = 1 - \frac{\sum_n \sum_{t \geq 2} \left\| \tilde{h}_t^{(n)} - f_\theta(\tilde{h}_{<t}^{(n)}) \right\|^2}{\sum_n \sum_{t \geq 2} \left\| \tilde{h}_t^{(n)} - \bar{\tilde{h}} \right\|^2}.$$

The sums run over sequences n and positions $t \geq 2$. The numerator is the MSE of f_θ, and the denominator is the MSE of the constant predictor $\bar{\tilde{h}}$ (the mean over all train positions), which uses no input at all. That is, R^2 is one minus the normalized MSE, which measures the former in units of the latter; 0 corresponds to the constant predictor and 1 to perfect prediction. This normalization is what makes values comparable across layers and predictors.

Training. All predictors are trained with the same settings in all conditions (Appendix A), and we take the checkpoint from the epoch with the best val R^2.

Table 1: R^2 (test). $R^2 = 0$ corresponds to the constant predictor that uses no input.

	L00	L06	L12	L18	L24
copy	-0.980	-1.616	-1.431	-1.239	-0.148
affine	0.030	0.123	0.161	0.200	0.304
MLP	0.046	0.412	0.442	0.457	0.393
Trf×1	0.114	0.421	0.447	0.472	0.446
Trf×2	0.161	0.439	0.456	0.490	0.478
Trf×4	0.179	0.444	0.466	0.497	0.492

5 Basic properties of the learned decomposition

In terms of R^2, the differences among predictors are small, but in terms of autocorrelation, the properties of the decomposition change substantially depending on whether the predictor aggregates the entire past.

5.1 Prediction accuracy

Table 1 shows R^2 for 5 layers $\times$ 6 predictors (per-dimension R^2 is given in Appendix C).

As a prerequisite for the subsequent analyses, this table shows how well $\hat{h}_t$ is approximated. Among the predictors that take only the previous state, R^2 rises substantially from affine to MLP, but extending the input to the entire past (MLP $\rightarrow$ Transformer) raises it by at most 0.04 at L06–L18. The copy predictor has negative R^2 at all layers and is worse than the constant predictor.

5.2 Autocorrelation along the token axis

We measure how similar the states at positions t and $t + k$ are. For a component c, let $a = c_t - m_c$ and $b = c_{t+k} - m_c$ (m_c is the per-dimension median), and define the autocorrelation as

$$\rho_{\text{med}}(k) = \frac{\text{median}\langle a, b\rangle}{\text{median}\,\|a\|^2}.$$

We use the median for centering and aggregation because a few specific dimensions take extremely large values at a small number of positions and would dominate the mean (Appendix B). Under this definition, ρ_{med} is exactly 1 at $k = 0$, and values near 0 indicate little alignment between the two positions.

Figure 2(a) shows the decay of the autocorrelation for L12 / Transformer $\times$4. Over the whole range of lags from 1 to 32, the autocorrelation of $\hat{h}_t$ is consistently higher than that of h_t, and the autocorrelation of ε_t is nearly zero. This result, that the predictable component persists across positions while the unpredictable component does not, is consistent with the intent of the decomposition, which splits h_t into the component $\hat{h}_t$ predictable from the context and the unpredictable component ε_t. The same holds at all five layers. In deeper layers, both h_t and $\hat{h}_t$ persist longer (Appendix D).

Figure 2(b) shows the lag-1 autocorrelation for each predictor. The autocorrelation of ε_t is nearly zero for all predictors from MLP upward (for copy and affine, it deviates from 0). At L12, in contrast, only the Transformers, which aggregate $h_{<t}$, raise the autocorrelation of $\hat{h}_t$ above that of h_t (0.324), to about 0.55; for affine and MLP, which take only the previous state, it is close to that of h_t. Since adding blocks changes it by only $+0.011$, what matters is not capacity but aggregating all past states.

6 Reading out linguistic attributes

We construct 10 probing tasks spanning the linguistic hierarchy: surface (2: whether the token is capitalized and whether it consists only of digits), part of speech (1: POS, 16 classes), morphology (2: verb tense and noun number), syntax (2: dependency label, 8 classes, and subject number at

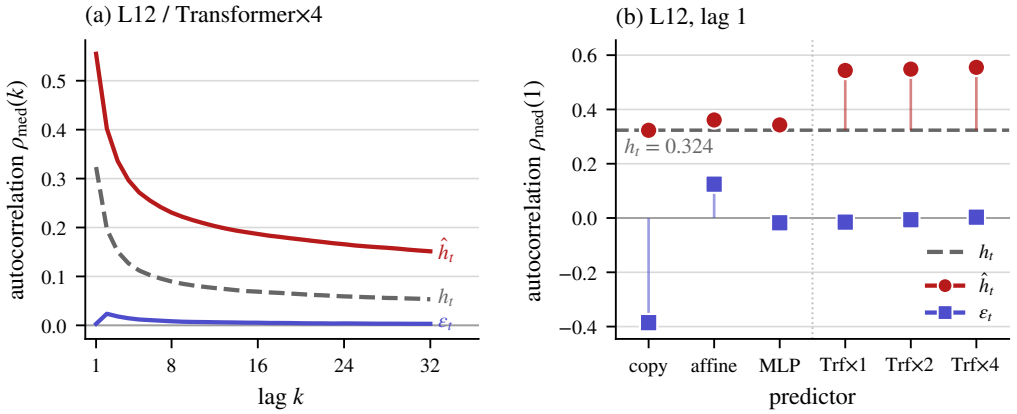

Figure 2: (a) Decay of the autocorrelation for L12 / Transformer $\times 4$. (b) Lag-1 autocorrelation at L12 for each predictor. Since h_t does not depend on the predictor, it is shown as a single dashed line. The $\hat{h}_t$ of copy lies on the dashed line because $\hat{h}_t = h_{t-1}$.

verb positions), semantics (1: named-entity type, 4 classes), structure (1: whether the token is inside quotes), and discourse (1: document source, 9 classes). Appendix E describes how the labels are assigned and defines each task.

We **train linear probes (Alain & Bengio, 2017; Belinkov, 2022) only on h_t and apply them without retraining to $\hat{h}_t$ and ε_t**. A linear probe outputs logits $z(v) = Wv + b$ for an input $v \in \mathbb{R}^d$. Since $h_t = \hat{h}_t + \varepsilon_t$ holds exactly,

$$Wh_t + b = (W\hat{h}_t + b) + W\varepsilon_t$$

is an identity. This setup lets us ask **which of the two components carries the attribute along the directions in h_t that the probe uses**.

The probes are trained on 10,000 val sequences and evaluated by macro-AUROC on 10,000 test sequences (only the document source uses separate sequences; see Appendix A, which also describes the input coordinates). AUROC is computed from the logit z_c of each class. Since adding a per-class constant does not change the ranking, the AUROC does not depend on which component the bias b is assigned to. For h_t, the AUROC ranges from 0.906 to 1.000 for all tasks from L06 onward.

Figure 3 shows $\mathrm{AUROC}(\hat{h}_t) - \mathrm{AUROC}(\varepsilon_t)$ for 5 layers $\times$ 10 tasks with Transformer $\times 4$. The innovation component ε_t has the higher AUROC at one or more layers for five tasks, and for noun number and verb tense, the differences are large up to the middle layers. For capitalization and dependency label, the differences are small even at the layers where ε_t has the higher AUROC.

For the remaining five tasks, $\hat{h}_t$ **has the higher AUROC at all layers**. The differences are largest for document source, subject number, and whether the token is inside quotes, that is, for attributes determined before that position.

Across layers, **for all 10 tasks, the difference shifts monotonically in favor of $\hat{h}_t$ from L06 to L24**, and for four tasks the sign changes across layers. Which component carries an attribute is determined by the combination of task and layer, and in deeper layers, the component that carries the attributes shifts toward $\hat{h}_t$.

The sign pattern is nearly identical for MLP and Transformer $\times 1$ / $\times 2$ but breaks down for copy and affine. Appendix F shows the full results for all six predictors.

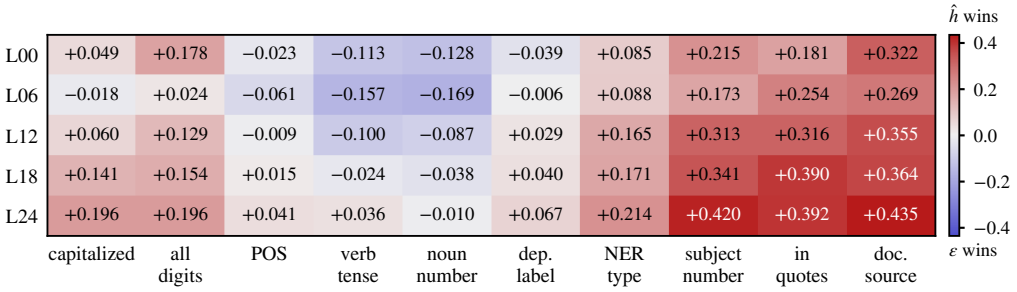

Figure 3: $\mathrm{AUROC}(\hat{h}_t) - \mathrm{AUROC}(\varepsilon_t)$ for Transformer $\times 4$, 5 layers $\times$ 10 tasks. The probes are trained on h_t and applied to both components with the weights fixed. Positive values indicate that the predictive component carries the attribute, and negative values that the innovation component does.

7 Intervention on the innovation component

We now carry out the operation described in §3, that is, dropping ε_t and passing $\hat{h}_t$ to the subsequent computation. We replace only the hidden state at position t (Meng et al., 2022) and use the KL divergence to measure how far the next-token distribution at that position moves from the original distribution (Conmy et al., 2023). We use the direction $D_{\mathrm{KL}}(p_{\mathrm{orig}} \,\|\, p_{\mathrm{int}})$ and interpret it as the information gained when updating from p_{int} to p_{orig}. We evaluate at 16 positions in each of 1,000 test sequences, 16,000 points in total.

We also analyze how the KL changes when the dropped part is added back with a scaling coefficient α:

$$h_t \leftarrow \hat{h}_t + \alpha\, \varepsilon_t.$$

Here $\alpha = 0$ is the intervention above, and $\alpha = 1$ gives h_t itself.

Because absolute KL values are hard to interpret on their own, we use two reference points. The **global mean** is the mean over the training split, a constant vector that uses no input at all. Replacing h_t with the global mean corresponds to mean ablation used in interpretability research (Wang et al., 2023). The other, $h_{\mathbf{mix}}$, is the average of the hidden states $h_t(x)$ obtained by placing each candidate token x at position t, weighted by the LLM's own next-token distribution:

$$h_{\mathrm{mix}} = \sum_x p(x \mid x_{<t})\, h_t(x).$$

This corresponds to $\mathbb{E}[h_t \mid x_{<t}]$ when the token at position t follows the LLM's next-token distribution, and it can be constructed without using the actual token. Even when the sum is truncated to the top 1,000 tokens, each position requires 1,000 LLM computations, so we use h_{mix} only as a reference point for reading KL values. Details are given in Appendix A.

The KL of the trained predictors lies between the two reference points, except for affine at L06. Even for h_{mix}, the KL is not zero. This is because h_{mix} is a state constructed without the actual token at position t and does not contain what is specific to that token.

For all five trained predictors, **the KL is minimized at L12**. In contrast, R^2 is highest at L18 (only affine is highest at L24; Table 1). **The layer where $\hat{h}_t$ best approximates h_t and the layer where dropping ε_t changes the output least do not coincide.** Across predictors, the KL decreases in the order affine, MLP, Transformer $\times 1$, $\times 2$, $\times 4$. The only exception is the reversal of MLP and Transformer $\times 1$ at the three middle layers.

Figure 4 shows the α-sweep at L12. First, **the quality of the predictor shifts the curve vertically but barely changes its shape**. Only copy has a different shape.

Table 2: Median KL divergence (nats) when h_t is replaced. Bold marks the minimum over layers. See Appendix G for the mean.

	L00	L06	L12	L18	L24
global mean	3.276	**2.637**	3.032	4.001	7.259
copy	**5.146**	6.831	6.900	6.926	6.900
affine	3.107	2.729	**2.220**	2.319	3.421
MLP	2.997	2.182	**1.706**	1.844	2.610
Trf×1	2.483	2.316	**1.730**	1.912	2.238
Trf×2	2.043	2.051	**1.667**	1.767	1.926
Trf×4	1.856	1.936	**1.455**	1.646	1.767
h_{mix}	0.586	**0.493**	0.544	0.585	0.750

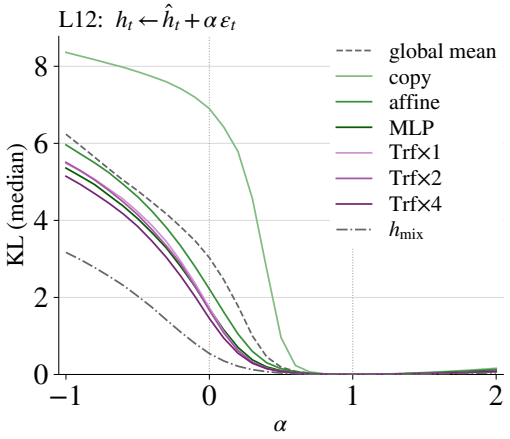

Figure 4: Median KL of the α-sweep at L12.

Second, **the curves are asymmetric around** $\alpha = 1$. For Transformer ×4, halving ε_t gives 0.077 nats and scaling it by 1.5 gives 0.024 nats, so reducing it has a roughly three times larger effect. Both points are $0.5\|\varepsilon_t\|$ away from h_t, so the asymmetry does not follow from the construction. The model's output is more sensitive to reducing the innovation component than to increasing it by the same amount. This asymmetry is seen at all layers (Appendix H).

8 Case study: changes in confidence along spans

We apply the probes of §6 (trained on h_t, with the weights fixed) to h_t and $\hat{h}_t$ (Transformer ×4) at each position at L12 and compare the confidence for class c, $\pi_c(v) = \mathrm{softmax}(z(v))_c$ ($\sigma(z(v))$ for two classes). The difference $\Delta\pi_c = \pi_c(h_t) - \pi_c(\hat{h}_t)$ is the change in confidence when ε_t is added to $\hat{h}_t$. Figure 5 shows three spans that correspond to different linguistic levels: word, sentence, and document. (a) is a word span, where the tokens of a person name (Luke Dobson) and a school name (Dover High) are labeled with the part of speech PROPN. (b) is a sentence span, where the tokens of a quoted utterance are labeled as inside quotes. (c) is a document span, where all tokens of a patent document are labeled with the document source USPTO (US patent documents).

We selected examples with clear patterns by visually inspecting 100 sequences sampled at random from test and 100 sequences for the document source. The three examples share a common pattern. The predictive component $\hat{h}_t$ predicts the label well in the middle of a span but misses it at the start and end, which are hard to predict; at these positions, ε_t switches the confidence.

In (a), at the first token of each name (Luke, Dover), the confidence of $\hat{h}_t$ remains around 0.5. This is because the context does not determine whether a name starts there. Adding ε_t raises the confidence to nearly 1, so $\Delta\pi_c$ is positive. At the following tokens, Dob and son, $\hat{h}_t$ also predicts PROPN. The two names differ at their ends. At the "," after Luke Dobson, where both the first and last names are complete, $\hat{h}_t$ also predicts that the name ends, and $\Delta\pi_c$ is nearly zero. At the "," after Dover High, in contrast, $\hat{h}_t$ predicts that the name continues, so ε_t lowers the confidence and $\Delta\pi_c$ is negative.

In (b), $\hat{h}_t$ cannot predict that an utterance starts at the opening quote, and ε_t raises the confidence that the token is inside quotes. From the next token, was, $\hat{h}_t$ also predicts inside, and in the middle of the utterance both are almost certain that the tokens are inside. At the closing quote, $\hat{h}_t$ still predicts inside, but ε_t lowers the confidence, and from the next token, said, $\hat{h}_t$ also predicts outside.

(c) is a patent document whose opening discusses a medical topic. At the beginning, both h_t and $\hat{h}_t$ have low confidence for USPTO and misclassify it as PubMed (medical abstracts). At "art" in the patent boilerplate phrase "in the art," ε_t raises the confidence to nearly 1, and h_t switches to USPTO. The predictive component $\hat{h}_t$ catches up at the next position. Afterwards, the confidence of

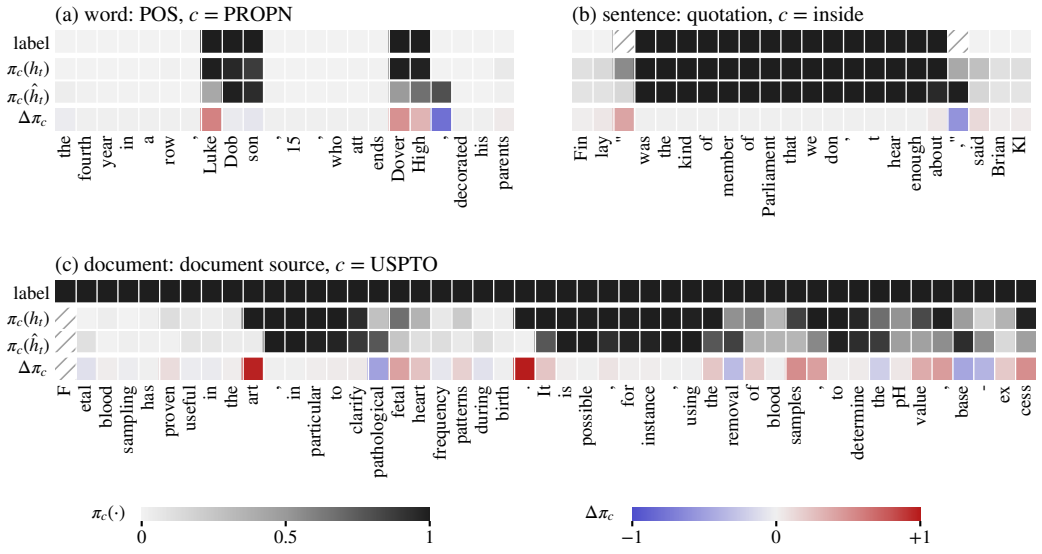

Figure 5: Confidence along spans at L12 with Transformer $\times 4$. In the label row, black means c; hatched cells have no value (quote labels, and values at position 1). Appendix I gives the sequences and position ranges.

both drops temporarily where medical terms continue, but at the sentence-final ".", ε_t again brings it back to USPTO first, and $\hat{h}_t$ catches up at the next position.

This pattern appears in all of the word, sentence, and document spans selected as examples. Since span-level labels are widely used in interpretability (Tenney et al., 2019; Marks & Tegmark, 2024), this pattern may be relevant to such analyses in general. Thus, the contributions of the two components change systematically with the position within a span.

9 Limitations and Conclusion

Limitations. First, $\hat{h}_t$ is an approximation by trained predictors, and we do not know how close it is to the true conditional expectation. As more Transformer blocks are stacked, R^2 keeps rising and the KL keeps falling (Tables 1 and 2). We report differences among predictors throughout, but the values may change further with better predictors. Second, we analyzed only one model, Pythia-410m-deduped, and the sequences are limited to the first 128 tokens of documents from its training corpus. We have not verified whether the same properties hold for other models, for text not used to train the model, or for longer contexts. Third, we have not yet quantified how general the pattern seen in §8 is. In addition, there is no established way to compare our decomposition with other decomposition methods along the token axis, such as TFA (§2), and developing one is left for future work.

Conclusion. The hidden state at each position of an LLM mixes a component that is already predictable from the past context and a component that becomes available only upon observing the current token. We separated the two with the conditional expectation and showed that they differ in three aspects: temporal structure, linguistic information, and effect on the model output. The predictive component $\hat{h}_t$ persists longer than h_t and carries attributes determined by the context, whereas the innovation component ε_t does not persist and carries token-local attributes. In deeper layers, the component that carries the attributes shifts toward $\hat{h}_t$. The output is more sensitive to reducing ε_t than to increasing it by the same amount. In future work, using SAE features (Bricken et al., 2023; Cunningham et al., 2024) instead of probes would allow the same analysis without labeled data. The predictive decomposition along the token axis, like the decomposition along the layer axis, could serve as a unit for analyzing internal representations.

AI USE STATEMENT

In this work, we used generative AI tools for feedback on the research methodology and experimental design, proposing some hypotheses, implementing methods, writing code to clean and reformat data, interpreting results, and translating the manuscript into English. We have not used generative AI tools to formulate mathematical claims, and synthetic data generation, proofs, and qualitative data analysis are not applicable to this work. The authors developed the theoretical framework, and AI was used to polish its description and check its logic. Additionally, we used generative AI tools for creating figures, creating and editing code other than the above, drafting and editing the manuscript, surveying related literature, formatting references, and considering the structure of the paper. We have reviewed all AI-assisted work. All experiments were run in the authors' environment, and the numbers in the paper were checked against the experimental outputs. Figures and tables were not generated directly by AI; they were created from the experimental results with code. We checked all cited works in their original sources. We take final responsibility for the content of this paper, including text, claims, and artifacts.

REPRODUCIBILITY STATEMENT

All experiments in this work use only publicly available models and data. The analyzed model, Pythia-410m-deduped, the deduplicated Pile from which we take the sequences (§4), and `monology/pile-uncopyrighted`, used for the document-source task, are all publicly available on Hugging Face. Appendix A describes how the sequences are extracted and split. It also summarizes the standardization, the architectures and parameter counts of the predictors, the training settings, the training and evaluation of the probes, the intervention settings, and the compute resources. Appendix E describes how the probe labels are assigned, and Appendix I gives the sequence indices and positions used in the case study. Results that the main text shows only for selected settings such as L12 are given in the appendix as figures and tables for all layers and predictors.

REFERENCES

Samira Abnar and Willem Zuidema. Quantifying attention flow in Transformers. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4190–4197, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.385. URL https://aclanthology.org/2020.acl-main.385/.

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. In *International Conference on Learning Representations (Workshop Track)*, 2017. URL https://arxiv.org/abs/1610.01644.

Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, March 2022. doi: 10.1162/coli_a_00422. URL https://aclanthology.org/2022.cl-1.7/.

Usha Bhalla, Alex Oesterling, Claudio Mayrink Verdun, Himabindu Lakkaraju, and Flavio P. Calmon. Temporal sparse autoencoders: Leveraging the sequential nature of language for interpretability. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=bojVI4l9Kn.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, Usvsn Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 2397–2430. PMLR, 2023. URL https://proceedings.mlr.press/v202/biderman23a.html.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nicholas L Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu,

Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Chris Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. URL https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023. URL https://arxiv.org/abs/2302.01318.

Arthur Conmy, Augustine Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. In *Advances in Neural Information Processing Systems*, volume 36, pp. 16318–16352, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/34e1dbe95d34d7ebaf99b9bcaeb5b2be-Abstract-Conference.html.

Róbert Csordás, Christopher D Manning, and Chris Potts. Do language models use their depth efficiently? In *Advances in Neural Information Processing Systems*, volume 38, pp. 160313–160362, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/eae3af0f5868f0a2eceb74208966d55b-Abstract-Conference.html.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=F76bwRSLeK.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for Transformer circuits. *Transformer Circuits Thread*, 2021. URL https://transformer-circuits.pub/2021/framework/index.html.

Javier Ferrando, Gerard I. Gállego, and Marta R. Costa-jussà. Measuring the mixing of contextual information in the Transformer. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 8698–8714, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.595. URL https://aclanthology.org/2022.emnlp-main.595/.

Javier Ferrando, Gabriele Sarti, Arianna Bisazza, and Marta R. Costa-jussà. A primer on the inner workings of Transformer-based language models. *arXiv preprint arXiv:2405.00208*, 2024. URL https://arxiv.org/abs/2405.00208.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800GB dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020. URL https://arxiv.org/abs/2101.00027.

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 30–45, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.3. URL https://aclanthology.org/2022.emnlp-main.3/.

Vincent Herrmann, Róbert Csordás, and Jürgen Schmidhuber. Measuring in-context computation complexity via hidden state prediction. In *International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 23063–23082. PMLR, 2025. URL https://proceedings.mlr.press/v267/herrmann25a.html.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength natural language processing in Python. Zenodo, 2020. URL https://doi.org/10.5281/zenodo.1212303.

Eghbal Hosseini and Evelina Fedorenko. Large language models implicitly learn to straighten neural sentence trajectories to construct a predictive representation of natural language. In *Advances in Neural Information Processing Systems*, volume 36, pp. 43918–43930, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/88dddaf430b5bc38ab8228902bb61821-Abstract-Conference.html.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. Attention is not only a weight: Analyzing Transformers with vector norms. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7057–7075, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.574. URL https://aclanthology.org/2020.emnlp-main.574/.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. Incorporating residual and normalization layers into analysis of masked language models. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 4547–4568, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.373. URL https://aclanthology.org/2021.emnlp-main.373/.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from Transformers via speculative decoding. In *International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19274–19286. PMLR, 2023. URL https://proceedings.mlr.press/v202/leviathan23a.html.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 28935–28948. PMLR, 2024. URL https://proceedings.mlr.press/v235/li24bt.html.

William Lotter, Gabriel Kreiman, and David Cox. Deep predictive coding networks for video prediction and unsupervised learning. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=B1ewdt9xe.

Ekdeep Singh Lubana, Can Rager, Sai Sumedh R. Hindupur, Valérie Costa, Greta Tuckute, Oam Patel, Sonia Krishna Murthy, Thomas Fel, Daniel Wurgaft, Eric J. Bigelow, Johnny Lin, Demba Ba, Martin Wattenberg, Fernanda Viegas, Melanie Weber, and Aaron Mueller. Priors in time: Missing inductive biases for language model interpretability. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=4J2e3nWiC8.

Haoyan Luo, Mateo Espinosa Zarlenga, and Mateja Jamnik. Persistent sparse autoencoders: Learning feature-specific timescales in language model representations. *arXiv preprint arXiv:2607.17117*, 2026. URL https://arxiv.org/abs/2607.17117.

Samuel Marks and Max Tegmark. The geometry of truth: Emergent linear structure in large language model representations of true/false datasets. In *Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=aajyHYjjsk.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*, volume 35, pp. 17359–17372, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/6f1d43d5a82a37e89b0665b33bf3a182-Abstract-Conference.html.

Beren Millidge, Anil Seth, and Christopher L. Buckley. Predictive coding: a theoretical and experimental review. *arXiv preprint arXiv:2107.12979*, 2021. URL https://arxiv.org/abs/2107.12979.

Ali Modarressi, Mohsen Fayyaz, Ehsan Aghazadeh, Yadollah Yaghoobzadeh, and Mohammad Taher Pilehvar. DecompX: Explaining Transformers decisions by propagating token decomposition. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2649–2664, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.149. URL https://aclanthology.org/2023.acl-long.149/.

Hosein Mohebbi, Willem Zuidema, Grzegorz Chrupała, and Afra Alishahi. Quantifying context mixing in Transformers. In Andreas Vlachos and Isabelle Augenstein (eds.), *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 3378–3400, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.eacl-main.245. URL https://aclanthology.org/2023.eacl-main.245/.

Byung-Doh Oh and William Schuler. Token-wise decomposition of autoregressive language model hidden states for analyzing model predictions. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10105–10117, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.562. URL https://aclanthology.org/2023.acl-long.562/.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text Transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL https://jmlr.org/papers/v21/20-074.html.

Guy Revach, Nir Shlezinger, Xiaoyong Ni, Adrià López Escoriza, Ruud J. G. van Sloun, and Yonina C. Eldar. KalmanNet: Neural network aided Kalman filtering for partially known dynamics. *IEEE Transactions on Signal Processing*, 70:1532–1547, 2022. doi: 10.1109/TSP.2022.3158588. URL https://doi.org/10.1109/TSP.2022.3158588.

Jiajun Song and Yiqiao Zhong. Uncovering hidden geometry in Transformers via disentangling position and context. *Harvard Data Science Review*, 8(2), 2026. doi: 10.1162/99608f92.373b4512. URL https://hdsr.mitpress.mit.edu/pub/l9iaw8m3.

Xiangchen Song, Jiaqi Sun, Zijian Li, Yujia Zheng, and Kun Zhang. LLM interpretability with identifiable temporal-instantaneous representation. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://openreview.net/forum?id=TdmzrkdLG0.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced Transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. doi: 10.1016/j.neucom.2023.127063. URL https://doi.org/10.1016/j.neucom.2023.127063.

Mingjie Sun, Xinlei Chen, J. Zico Kolter, and Zhuang Liu. Massive activations in large language models. In *Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=F7aAhfitX6.

Alex Tamkin, Dan Jurafsky, and Noah Goodman. Language through a prism: A spectral approach for multiscale language representations. In *Advances in Neural Information Processing Systems*, volume 33, pp. 5492–5504, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/3acb2a202ae4bea8840224e6fce16fd0-Abstract.html.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R. Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. What do you learn from context? Probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SJzSgnRcKX.

William Timkey and Marten van Schijndel. All bark and no bite: Rogue dimensions in Transformer language models obscure representational quality. In Marie-Francine Moens, Xuanjing Huang,

Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 4527–4546, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.372. URL https://aclanthology.org/2021.emnlp-main.372/.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=NpsVSN6o4ul.

Pengfei Wu, Jiahao Liu, Zhuocheng Gong, Qifan Wang, Jinpeng Li, Jingang Wang, Xunliang Cai, and Dongyan Zhao. FIRP: Faster LLM inference via future intermediate representation prediction. In Derek F. Wong, Zhongyu Wei, and Muyun Yang (eds.), *Natural Language Processing and Chinese Computing (NLPCC)*, volume 15361 of *Lecture Notes in Computer Science*, pp. 158–169, Singapore, 2024. Springer. doi: 10.1007/978-981-97-9437-9_13. URL https://link.springer.com/chapter/10.1007/978-981-97-9437-9_13.

A EXPERIMENTAL DETAILS

Data. We read the train split of `EleutherAI/the_pile_deduplicated` on Hugging Face as a stream in order (without shuffling) and take the first 128 tokens of each document that has at least 128 tokens, until we have 200,000 sequences. In this order, the first 160,000 sequences form the train split, the next 20,000 the val split, and the remaining 20,000 the test split. Following the training setup of Pythia, we do not prepend a BOS token, so position 1 is the first token of a document. The deduplicated Pile has no subset names, so for the document-source task (§6) only, we take separate sequences from a version that has them (`monology/pile-uncopyrighted` on Hugging Face). We read this version as a stream shuffled with a buffer of 10,000, and the seed for this shuffle and for the random split below is 0. From each of the 9 subsets, we take the first 128 tokens of 1,000 documents that have at least 128 tokens, and within each subset we randomly assign 700 sequences to probe training and 300 to evaluation. The standardization coefficients and the predictors are those obtained on the data of §4; these sequences are never used to train the predictors.

Standardization. The median m_i and the MAD s_i of each dimension are computed for each layer from all positions of the first 50,000 train sequences.

Predictors. Table 3 summarizes the predictors. Every predictor except copy adds a learned term to its input $\tilde{h}_{t-1}$, and the output layer of this term is initialized to zero, so every predictor is initialized to the copy predictor. A Transformer block is a pre-LN residual block, $x \leftarrow x + \mathrm{Attn}(\mathrm{LN}(x))$ followed by $x \leftarrow x + \mathrm{MLP}(\mathrm{LN}(x))$. The number of heads (16), the MLP hidden size ($4d$), and the activation (GELU) match those of the LLM. The block differs from the LLM's block in three ways. The LLM adds the attention and MLP outputs in parallel, whereas the predictor applies them in sequence. The attention projections have no bias. For positions, instead of RoPE (Su et al., 2024), a bias learned for each head and each relative distance (Raffel et al., 2020) is added to the attention scores. Attention is causal, and the output at position $t-1$ is the prediction of $\tilde{h}_t$. There is no final LayerNorm.

Table 3: Predictors ($d = 1024$; the MLP hidden size is $4d$). A separate predictor is trained for each layer.

Predictor	Input	Output	Parameters
copy	$\tilde{h}_{t-1}$	$\tilde{h}_{t-1}$	0
affine	$\tilde{h}_{t-1}$	$\tilde{h}_{t-1} + W\tilde{h}_{t-1} + b$	1.05M
MLP	$\tilde{h}_{t-1}$	$\tilde{h}_{t-1} + W_2\,\mathrm{GELU}(W_1\tilde{h}_{t-1} + b_1) + b_2$	8.39M
Trf$\times$1 / $\times$2 / $\times$4	$\tilde{h}_{<t}$	output of 1 / 2 / 4 blocks described above	12.6M / 25.2M / 50.4M

Training. The loss is the squared error in the standardized space, averaged over positions $t \geq 2$ and all dimensions. We use AdamW (learning rate 10^{-3}, weight decay 0.01) with batches of 128 sequences for 10 epochs. The learning rate warms up linearly for 200 steps and then decays to 0 with a cosine schedule, and gradients are clipped to a norm of 1.0. At the end of each epoch we measure R^2 on val, and we keep the weights of the best epoch. We train each of the five predictors other than copy once for each of the five layers. We used NVIDIA RTX A6000 GPUs.

Figure 6 shows the val R^2 of the trained predictors at each epoch. At L06–L18, the Transformers reach their best val R^2 at epoch 5 or 6; after that, their train R^2 keeps rising while their val R^2 falls (by up to 0.09 by epoch 10). The values in Table 1 are test R^2 measured with the weights of the epoch with the best val R^2. Affine and MLP reach their best at epoch 10, but they gain at most 0.0006 in the last epoch. For every trained predictor, val and test R^2 differ by at most 0.0035.

Probe training. The probe is an L2-regularized logistic regression (scikit-learn, $C = 10^{-3}$, lbfgs with at most 1,000 iterations, class weights inversely proportional to class frequencies). It is trained on the first 10,000 val sequences and evaluated on the first 10,000 test sequences, using all labeled positions $t \geq 2$. Only the document-source task uses the separate sequences described above (6,300 for training and 2,700 for evaluation). All 50 probes (5 layers $\times$ 10 tasks) converged with lbfgs.

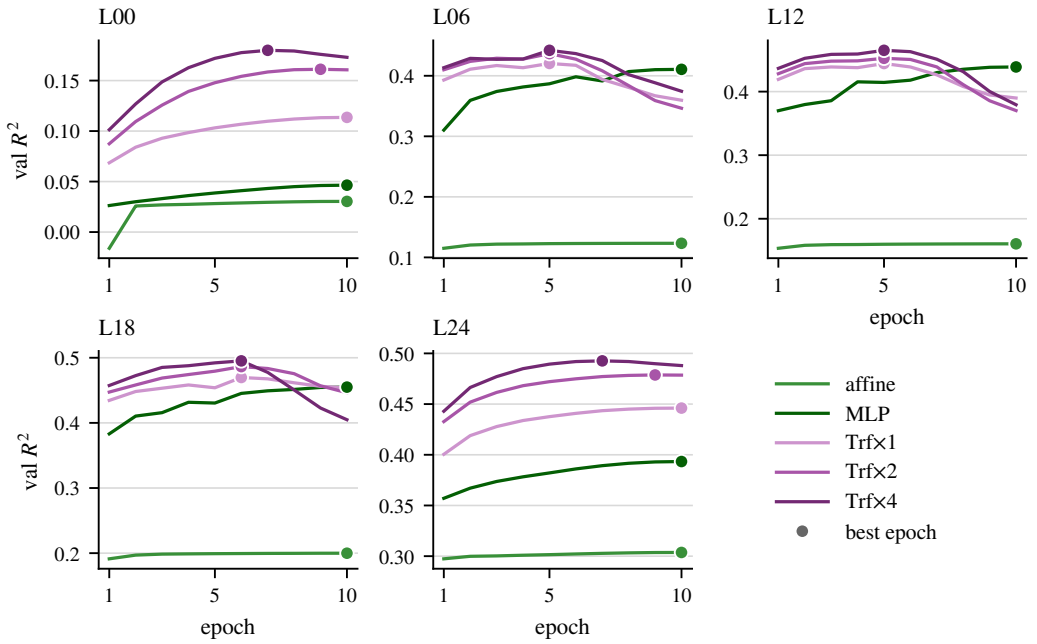

Figure 6: Val R^2 (20,000 sequences) of the trained predictors at each epoch. Dots mark the epoch with the best val R^2, whose weights we use. Colors follow Figure 4. The range of the y-axis differs across layers.

Probe coordinates. In the standardized space where the predictors operate, a few dimensions still have standard deviations orders of magnitude larger than the others even after division by the MAD, because of rare large values, and lbfgs does not converge in this space. We therefore compute the standard deviation σ_i of each dimension of $\tilde{h}_t$ on the probe training sequences for each layer (separately for the document-source sequences), and divide h_t, $\hat{h}_t$, and ε_t in the standardized space by the same σ_i. In the original coordinates, the probe input for h_t is $(h_{t,i} - m_i)/(s_i\sigma_i)$. Because all three components are divided by the same σ_i, the identity $Wh_t + b = (W\hat{h}_t + b) + W\varepsilon_t$ of §6 holds exactly in these coordinates. Over all 300 combinations (5 layers $\times$ 10 tasks $\times$ 6 predictors), the two sides differ by at most 3.8×10^{-5} (the logits are at most 54 in absolute value), which is within float32 rounding.

Probe evaluation. The macro-AUROC is the mean of the one-vs-rest AUROCs of the classes. As in §6, the score of each class is its logit z_c (with two classes, the single logit). Our implementation subtracts the mean over classes from the logits of multi-class probes, but the resulting macro-AUROC differs from that computed with the logits themselves by less than 10^{-6} in every condition.

Intervention. The KL divergence is measured on the first 1,000 test sequences, at 16 positions $t \geq 2$ drawn uniformly at random without replacement from each sequence. At position t only, we replace the output of block l of the LLM (for L00, the output of the token embedding) with $\hat{h}_t$ or one of the reference states, mapped back to the original coordinates, and we measure $D_{\mathrm{KL}}(p_{\mathrm{orig}} \,\|\, p_{\mathrm{int}})$ for the next-token distribution at position t, as in §7. The LLM runs in float16. The global mean is the mean over positions $t \geq 2$ of the first 20,000 train sequences. The α-sweep uses 31 values, $\alpha = -1.0, -0.9, \ldots, 2.0$, at the same 16,000 points, and its values at $\alpha = 0$ equal those in Table 2. At $\alpha = 1$, the state is the original one, so the KL should be 0; it is indeed at most 0.0021 nats in every layer and condition.

h_{mix}. The sum that defines h_{mix} (§7) is truncated to the top 1,000 tokens of the LLM's next-token distribution, and their probabilities are renormalized to sum to 1 and used as weights. The top

1,000 tokens cover a probability mass of 0.972 on average (median 0.991, minimum 0.52). h_{mix} differs from $\hat{h}_t = \mathbb{E}[h_t \mid h_{<t}]$ both in what it conditions on ($x_{<t}$ rather than $h_{<t}$) and in the distribution over which the expectation is taken (the LLM's next-token distribution rather than the corpus). In addition, h_{mix} is expensive to compute: the computation at position t must be redone for each candidate token, so each position requires 1,000 LLM computations. We therefore do not use h_{mix} as a target for the predictors, but only as a reference point for reading KL values. It is too expensive for analyses over many positions, such as the probes (§6).

B WHY WE USE THE MEDIAN AND MAD

In the residual stream, a few dimensions take values orders of magnitude larger than the others at rare positions (Sun et al., 2024). Table 4 shows the distribution of $\|h_t\|$ on 2,000 test sequences. At L06–L18, the median of $\|h_t\|$ is about 20–34, whereas its 99.9th percentile exceeds 900. L00 and L24 have no such positions.

The mean and the standard deviation are dominated by these rare positions. This is why the standardization in §4 subtracts the median rather than the mean and divides by the MAD rather than the standard deviation.

The autocorrelation (§5.2) uses medians for the same reason. With the usual definition, which centers each dimension by its mean and aggregates with means, $\rho_{\text{mean}}(k) = \text{mean}\langle a, b\rangle / \text{mean}\,\|a\|^2$, the denominator is dominated by the rare positions with large norms. At L06–L18, this denominator is 16–30 times that of the median-based definition, and the lag-1 autocorrelation of h_t collapses to 0.007–0.031. At L00 and L24, which have no such positions, the two definitions nearly agree.

Table 4: Distribution of $\|h_t\|$ in the original coordinates and the lag-1 autocorrelation of h_t under the two definitions (2,000 test sequences, positions $t \geq 2$). The denominator ratio is $\text{mean}\,\|a\|^2$ (mean-centered) divided by $\text{median}\,\|a\|^2$ (median-centered).

Layer	Median of $\|h_t\|$	99.9th pct. of $\|h_t\|$	Denominator ratio	$\rho_{\text{mean}}(1)$	$\rho_{\text{med}}(1)$
L00	0.8	0.8	0.9	0.010	0.001
L06	19.5	917.0	22.3	0.009	0.224
L12	19.6	1060.1	30.4	0.007	0.324
L18	33.6	1058.3	16.3	0.031	0.423
L24	42.1	95.6	1.1	0.434	0.431

C PER-DIMENSION R^2

The R^2 in Table 1 pools all dimensions, so it could be high even if a predictor fitted only a few high-variance dimensions. We therefore compute the per-dimension $R_i^2 = 1 - \text{Var}(\tilde{\varepsilon}_{t,i}) / \text{Var}(\tilde{h}_{t,i})$ in the standardized space, over positions $t \geq 2$ of 2,000 test sequences, with the variances computed within these sequences (unlike the definition in §4, the train mean is not used). Table 5 shows the distribution for Transformer $\times 4$. At all five layers, R_i^2 is positive for all 1,024 dimensions.

Table 5: Distribution of the per-dimension R_i^2 (1,024 dimensions) for Transformer $\times 4$.

Layer	Min	25th pct.	Median	75th pct.	Max
L00	0.136	0.163	0.172	0.183	0.557
L06	0.041	0.307	0.330	0.359	0.494
L12	0.169	0.349	0.371	0.399	0.615
L18	0.151	0.421	0.447	0.476	0.642
L24	0.326	0.460	0.483	0.512	0.713

D AUTOCORRELATION AT ALL LAYERS

Figure 7 shows the decay of the autocorrelation for Transformer ×4 at all five layers. At every layer, $\hat{h}_t$ lies above h_t at every lag, and ε_t stays near 0. At every lag, deeper layers have larger values for both h_t and $\hat{h}_t$. Table 6 gives the lag-1 values of $\hat{h}_t$ for all predictors and, for ε_t, the maximum of $|\rho_{\text{med}}(k)|$ over lags 1–32.

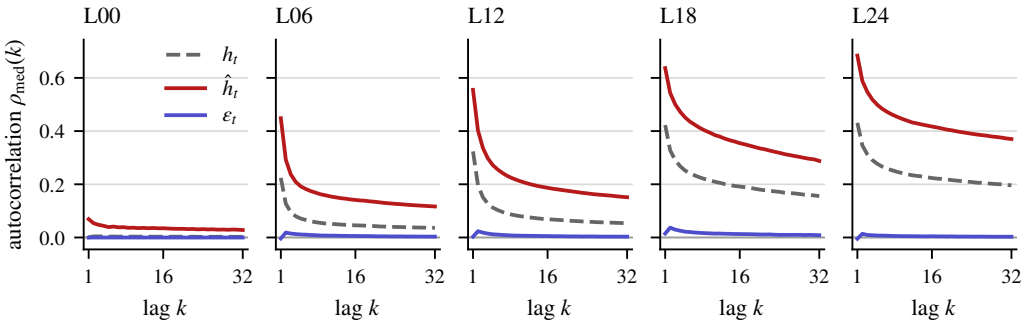

Figure 7: Decay of the autocorrelation $\rho_{\text{med}}(k)$ for Transformer ×4 (original coordinates, 2,000 test sequences). This extends Figure 2(a) to all five layers; the y-axis is shared.

Table 6: Autocorrelation ρ_{med} (original coordinates, 2,000 test sequences). The h_t row gives lag-1 values. For copy, $\hat{h}_t = h_{t-1}$, so its values nearly equal those of h_t, and the autocorrelation of $\varepsilon_t = h_t - h_{t-1}$ is strongly negative at lag 1 (the table shows absolute values).

	L00	L06	L12	L18	L24
h_t	0.001	0.224	0.324	0.423	0.431
$\hat{h}_t$ (lag 1)					
copy	0.001	0.225	0.323	0.424	0.430
affine	0.057	0.323	0.361	0.458	0.671
MLP	0.040	0.300	0.343	0.497	0.651
Trf×1	0.096	0.417	0.544	0.615	0.705
Trf×2	0.075	0.455	0.548	0.639	0.695
Trf×4	0.069	0.449	0.555	0.637	0.684
ε_t (max of $\lvert\rho_{\text{med}}(k)\rvert$ over lags 1–32)					
copy	0.546	0.427	0.385	0.381	0.390
affine	0.006	0.075	0.125	0.104	0.067
MLP	0.005	0.025	0.018	0.027	0.063
Trf×1	0.002	0.061	0.022	0.035	0.019
Trf×2	0.001	0.018	0.026	0.037	0.010
Trf×4	0.001	0.019	0.024	0.037	0.014

E TASK DEFINITIONS AND LABEL COUNTS

Table 7 gives the definition and the number of classes of each of the 10 tasks, together with the numbers of positions used to train and evaluate the probes. Labels for POS, morphology, syntax, and semantics are obtained by applying spaCy's `en_core_web_sm` (Honnibal et al., 2020) to the decoded text of each 128-token sequence. When a spaCy word spans several tokens, all overlapping tokens receive the same label. Positions without a label are excluded from both training and evaluation. For attributes restricted to certain word types (tense, noun number, and subject number), negative examples are restricted to the same word types. The 9 classes of the document source are Github, PubMed Abstracts, StackExchange, Wikipedia (en), FreeLaw, HackerNews, ArXiv, Enron Emails, and USPTO Backgrounds.

Table 7: Definitions of the 10 tasks. Task names are the labels used in Figures 3 and 8. Train and Eval give the numbers of labeled positions $t \geq 2$ (Train: 10,000 val sequences; Eval: 10,000 test sequences; the document source uses the separate sequences of Appendix A).

Level	Task	Definition	Classes	Train	Eval
surface	capitalized	Whether the first non-space character of the token is uppercase	2	1,270,000	1,270,000
surface	all digits	Whether the token, with surrounding spaces removed, consists only of digits	2	1,270,000	1,270,000
POS	POS	spaCy UPOS tag (X and SPACE excluded)	16	1,178,817	1,177,585
morphology	verb tense	Whether a VERB / AUX word with a Tense feature is past or non-past	2	107,280	106,880
morphology	noun number	Whether a NOUN word with a Number feature is singular or plural	2	238,452	238,400
syntax	dep. label	Dependency label of the word (nsubj, dobj, det, amod, prep, pobj, advmod, conj)	8	542,901	542,252
semantics	NER type	Which named-entity type (PERSON, ORG, GPE, DATE) the word belongs to; words outside these entities are excluded	4	158,640	158,316
syntax	subject number	At a VERB / AUX position, whether its subject (nsubj / nsubjpass) is singular or plural; if the subject has no Number feature, it is determined from the pronoun	2	60,217	59,615
structure	in quotes	Whether the token is inside double quotes (", “, ”); tokens containing a quote are excluded	2	1,257,380	1,256,574
discourse	doc. source	The Pile subset that the document belongs to; all positions of a document share the label	9	800,100	342,900

F PROBE RESULTS FOR ALL SIX PREDICTORS

Figure 8 shows $\mathrm{AUROC}(\hat{h}_t) - \mathrm{AUROC}(\varepsilon_t)$, as in Figure 3, for all six predictors. The probe is trained on h_t, so it is the same for all predictors. Of the 40 cells at L06–L24, the sign agrees with that of Transformer $\times 4$ in 39 cells for MLP, in all 40 for Transformer $\times 1$ and $\times 2$, in 27 for copy, and in 31 for affine. Table 8 gives the AUROC of the probe applied to h_t. Table 9 gives $\mathrm{AUROC}(\hat{h}_t)$ and $\mathrm{AUROC}(\varepsilon_t)$ for Transformer $\times 4$, from which Figure 3 is computed.

Table 8: Macro-AUROC of the probe applied to h_t.

Task	L00	L06	L12	L18	L24
capitalized	1.000	1.000	1.000	0.998	0.976
all digits	1.000	1.000	1.000	1.000	0.999
POS	0.969	0.981	0.980	0.977	0.967
verb tense	0.994	0.995	0.993	0.981	0.963
noun number	0.954	0.963	0.964	0.959	0.947
dep. label	0.901	0.964	0.966	0.959	0.940
NER type	0.857	0.915	0.921	0.919	0.906
subject number	0.805	0.967	0.963	0.945	0.928
in quotes	0.661	0.954	0.954	0.956	0.947
doc. source	0.782	0.992	0.995	0.994	0.990

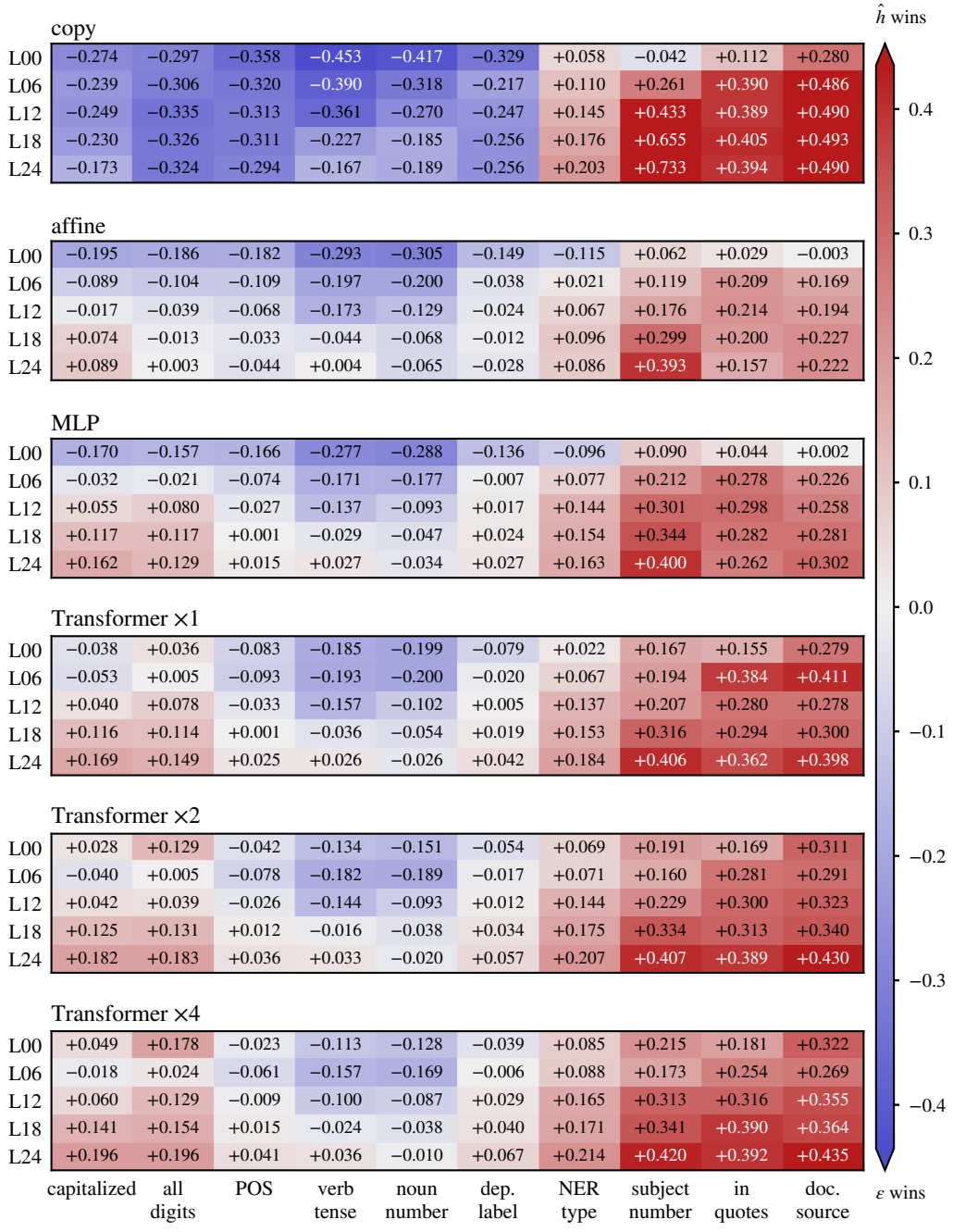

Figure 8: $\mathrm{AUROC}(\hat{h}_t) - \mathrm{AUROC}(\varepsilon_t)$ for all six predictors, in the same format as Figure 3; the last panel (Transformer $\times 4$) is Figure 3 itself. All six panels share the color range of Figure 3, ± 0.435; the cells beyond this range (7 cells for copy) are drawn in the end colors.

Table 9: AUROC($\hat{h}_t$) and AUROC(ε_t) for Transformer $\times 4$. Figure 3 shows their difference.

	L00		L06		L12		L18		L24	
Task	$\hat{h}_t$	ε_t	$\hat{h}_t$	ε_t	$\hat{h}_t$	ε_t	$\hat{h}_t$	ε_t	$\hat{h}_t$	ε_t
capitalized	0.931	0.882	0.907	0.925	0.918	0.857	0.925	0.784	0.911	0.715
all digits	0.932	0.755	0.912	0.888	0.919	0.790	0.920	0.766	0.923	0.727
POS	0.849	0.872	0.857	0.918	0.878	0.886	0.880	0.865	0.872	0.831
verb tense	0.842	0.956	0.818	0.975	0.850	0.950	0.860	0.884	0.858	0.822
noun number	0.779	0.907	0.762	0.931	0.817	0.904	0.831	0.869	0.828	0.838
dep. label	0.792	0.832	0.872	0.878	0.887	0.858	0.884	0.844	0.867	0.800
NER type	0.795	0.710	0.857	0.769	0.881	0.716	0.884	0.714	0.881	0.666
subject number	0.851	0.636	0.956	0.783	0.965	0.652	0.956	0.615	0.947	0.527
in quotes	0.696	0.515	0.952	0.698	0.956	0.640	0.960	0.570	0.959	0.567
doc. source	0.860	0.538	0.991	0.721	0.996	0.641	0.995	0.631	0.994	0.559

G MEAN KL DIVERGENCE

Table 2 reports the median KL. Since the distribution of the KL is heavy-tailed, Table 10 gives the mean over the same 16,000 points. With the mean as well, the KL is minimized at L12 for all five trained predictors and decreases as capacity increases. As in Table 2, the only exception is the reversal of MLP and Transformer $\times 1$ at the three middle layers. With the mean, affine at L06 also lies between the two reference points.

Table 10: Mean KL (nats, $\alpha = 0$) over the same 16,000 points as Table 2. Bold marks the minimum over layers in each row, as in Table 2.

	L00	L06	L12	L18	L24
global mean	4.044	**3.763**	3.812	5.155	8.071
copy	**6.039**	7.726	7.803	7.842	7.838
affine	3.956	3.687	**3.297**	3.439	4.768
MLP	3.871	3.320	**2.923**	3.071	3.977
Trf$\times$1	3.463	3.439	**2.965**	3.130	3.690
Trf$\times$2	3.198	3.291	**2.891**	3.020	3.382
Trf$\times$4	3.129	3.233	**2.804**	2.926	3.273
h_{mix}	2.605	2.528	2.315	**2.142**	2.297

H α-SWEEP AT ALL LAYERS

Figure 9 shows the α-sweep of Figure 4 at all five layers. Table 11 gives the median KL when ε_t is halved ($\alpha = 0.5$) divided by the median KL when ε_t is scaled by 1.5 ($\alpha = 1.5$). Both points are $0.5\|\varepsilon_t\|$ away from h_t, and the ratio exceeds 1 in every layer and condition. For the trained predictors, the ratio is 2.5–3.7 at L06–L24 and 9–15 at L00.

Table 11: Median KL at $\alpha = 0.5$ (ε_t halved) divided by that at $\alpha = 1.5$ (ε_t scaled by 1.5). Values above 1 mean that reducing ε_t changes the output more than increasing it by the same amount.

	L00	L06	L12	L18	L24
global mean	22.5	3.7	6.5	4.1	5.8
copy	150.5	28.1	19.2	9.8	4.5
affine	15.0	3.5	3.7	3.4	3.6
MLP	14.4	3.1	3.3	3.1	3.0
Trf$\times$1	11.2	3.1	3.3	3.2	2.8
Trf$\times$2	9.6	3.1	3.3	3.1	2.6
Trf$\times$4	9.1	3.0	3.2	3.0	2.5
h_{mix}	6.2	2.8	2.9	2.7	2.4

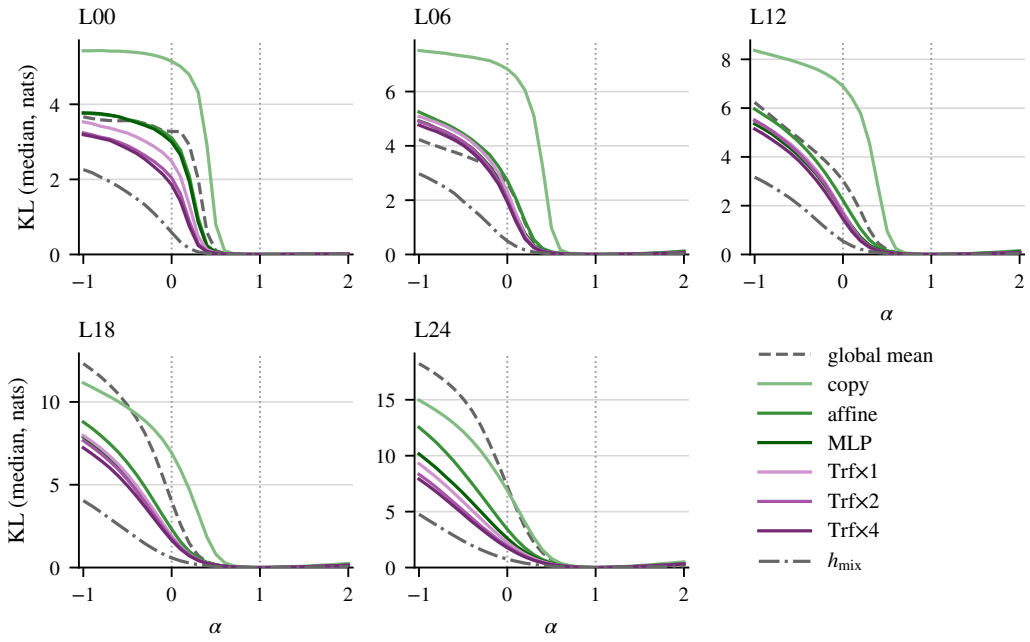

Figure 9: Median KL for $h_t \leftarrow \hat{h}_t + \alpha\,\varepsilon_t$ at each of the five layers. Colors and line styles follow Figure 4; darker colors indicate larger capacity. The range of the y-axis differs across layers.

I SEQUENCES AND POSITIONS OF THE CASE STUDY

The three examples in Figure 5 were chosen from 100 sequences sampled at random from the 20,000 test sequences and 100 sequences sampled at random from the 2,700 evaluation sequences of the document-source task. Table 12 gives the sequence index, the position range, and the text of each example. Sequence indices count from 0 within the 20,000 test sequences and within the 2,700 document-source evaluation sequences, respectively. Positions count from 1 at the first token of a document, as in the main text.

Table 12: Sequences and position ranges (inclusive) of the three examples in Figure 5.

	Data	Seq.	Positions	Text
(a)	test	13682	92–113	`the fourth year in a row, Luke Dobson, 15, who attends Dover High, decorated his parents`
(b)	test	7764	92–113	`Finlay "was the kind of member of Parliament that we don’t hear enough about," said Brian Kl`
(c)	doc. source (eval)	2236	1–47	`Fetal blood sampling has proven useful in the art, in particular to clarify pathological fetal heart frequency patterns during birth. It is possible, for instance, using the removal of blood samples, to determine the pH value, base-excess`