THE STANDARDIZATION TRAP: CERTIFYING JOINT LABEL PROCESSING IN TABULAR FOUNDATION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Linear regression and kernel smoothing offer tractable explanations of in-context learning: in both, the features determine the weight assigned to each context label. However, whether this fixed-weight account describes pretrained tabular foundation models (TFMs) remains unclear. Testing this account using derivatives runs into a standardization trap: public TFM packages standardize the labels before the model sees them, yet ordinary derivatives also reflect behavior outside the set of standardized labels, making a model appear nonlinear even when every prediction it makes agrees with a fixed-weight map. We propose two certificates that depend only on predictions at standardized labels and can reject two distinct explanations: fixed-weight prediction and sums of independent nonlinear label transformations. Across the five public TFMs that we evaluate, our certificates show that changing one context label alters how other labels influence the prediction, a behavior we call joint processing. We further find that joint processing emerges with training and that attention scores carry most of the measured interaction. Together, these findings motivate TFM explanations that account for how context labels change the influence of individual examples.

1 INTRODUCTION

Influential studies have connected in-context learning to linear regression and kernel smoothing (Akyürek et al., 2023; von Oswald et al., 2023; Zhang et al., 2024; Collins et al., 2024). In fixed-weight versions of these accounts, the features determine how the examples are weighted, while the context labels supply the values being combined. Predictions change when labels change, but the coefficients do not. This gives a precise starting point for understanding adaptation without parameter updates.

We ask how far this account extends to pretrained tabular foundation models (TFMs). At inference, TFMs predict the label of a query using *labeled* context examples while keeping their parameters fixed. How do TFMs use those context labels? Do the labels only supply the values being combined, or do they also change how the examples are weighted?

Consider predicting a house price from nearby sales. One strategy assigns a fixed weight to each house using location and building attributes, then averages the observed prices. Another also examines the prices themselves: an unusually high price may be discounted when isolated but reinforced when nearby houses show the same pattern. In the second strategy, one price changes the weight of other prices, even though the houses, their attributes, and the query remain fixed. Do the labels of a pretrained TFM only supply values that fixed weights combine, or does one label also change the weight given to another?

To answer this question, we measure *label sensitivity*: the first derivative of the prediction with respect to one context label. For a model that assigns a fixed weight to a context observation, this sensitivity is the weight itself, so it does not move when the labels change. We therefore look at second derivatives and ask whether the sensitivity to one label moves when another label changes. Furthermore, a model could transform each label separately, for example by limiting extreme values, and then add the contributions. This way, the prediction might respond nonlinearly to label changes, while no label would change the weight of another. We therefore distinguish fixed affine predic-

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.

tion, independent nonlinear label transformations, and interaction across examples. That distinction allows us to precisely ask what kind of adaptation a model has learned, rather than treating every nonlinear response as the same phenomenon.

Measuring such label interactions with derivatives runs into a surprising trap: tests based on derivative analysis can suggest that labels interact even when every prediction that the model makes combines the labels with fixed weights. Label standardization restricts the labels entering the network to vectors with a fixed mean and scale. The raw labels never reach the network directly, yet ordinary derivatives also describe behavior away from the set of standardized labels. For example, adding to a fixed-weight map a component that is zero on this set leaves every prediction unchanged but changes the derivatives (Section 3.2). We could therefore mistake behavior that never affects the prediction for evidence of adaptation.

Our contributions are as follows:

1. **Certificates that distinguish three forms of label processing**: We derive two certificates that account for standardization and distinguish three forms of label processing: (1) fixed-weight prediction, in which the weights do not depend on the labels, (2) sums of independent nonlinear label transformations, in which a label changes only its own influence, and (3) joint processing, in which one label changes the influence of the others. A nonzero value of the first certificate excludes (1), and a nonzero value of the second excludes both (1) and (2), which establishes (3).
2. **Evidence of joint processing and its association with training**: Using the proposed certificates, we show evidence of joint processing in the internal readouts of all tested models. Randomly initialized copies of the same architectures also respond nonlinearly to labels, but they show far less interaction across examples, which indicates that joint processing appears with training rather than coming from the architecture alone.
3. **Identification of the derivative paths carrying the measured interaction**: We show that holding the attention scores fixed during differentiation removes most of the measured local curvature while leaving the predictions unchanged, so most of the measured interaction passes through the attention scores.

Our paper is organized as follows. Section 2 reviews related work on explanations of how in-context learning works. Section 3 introduces the standardization trap and derives our two certificates. Section 4 applies these certificates to five public TFMs and examines the roles of training and attention mechanisms in joint label processing. Finally, Section 5 discusses the implications and limitations of our findings.

2 Related work

Fixed-weight accounts of in-context learning. Several prior works have shown that a transformer can carry out linear regression, or steps of gradient descent, on the context examples (von Oswald et al., 2023; Akyürek et al., 2023; Ahn et al., 2023; Zhang et al., 2024), and attention can be interpreted as a kernel operation (Tsai et al., 2019; Collins et al., 2024). In the simple versions of these explanations, the prediction is a weighted sum of the context labels. The features then characterize the weight of each example. The labels only supply the values to be combined (i.e., changing a label never changes a weight). We test whether pretrained TFMs process context labels the same way. With the features fixed, can one set of weights explain the predictions for every set of labels?

Writing a model in kernel form does not settle the question, because the kernel itself may already depend on the labels. For example, Miftachov et al. (2026) fine-tune TabICL (Qu et al., 2025) so that its final prediction is a kernel average, but the inputs of this kernel are embeddings that the model has computed from the features and the labels together. Any adaptation to the labels has then happened before the kernel is applied. The whole computation therefore has to be examined, including how representations are built and how parameters such as a bandwidth are chosen.

Learning a task from its labels. In other accounts, the labels do more than supply values. They also help decide which map is applied. Prior-data fitted networks are trained to approximate a Bayesian predictor. Such a predictor updates its belief about the task from the observed labels (Müller et al., 2022). Bayesian accounts of in-context learning make the same point: the labeled examples reveal which task is being solved, and the prediction follows from that task (Xie et al.,

2022; Panwar et al., 2024). A transformer can also select among several algorithms according to how well each one predicts examples held out from its fit (Bai et al., 2023). In all three accounts, changing the labels can change the inferred task or the selected algorithm, and with it the weight given to every label. Our measurements agree with this picture. Nagler (2023) addresses a different question, why the bias of a PFN need not vanish as the context grows unless the model focuses on the examples near the query, whereas we study how a model responds when the labels change at a fixed context size.

From internal computation to prediction. Another line of work studies the internal computation of TFMs. Gupta et al. (2026b) show that the hidden layers of TabPFN encode intermediate quantities of this computation, such as partial products of an arithmetic target, and Ye et al. (2025) find that TabPFN v2 (Hollmann et al., 2025) can serve as a feature extractor whose features separate the classes well. Predictions are refined gradually across layers, with so much overlap that one layer applied six times matches six distinct layers in a small model (Rezaei Balef et al., 2026). Ablation experiments find that one attention head dominates the others at its peak layer (Gupta et al., 2026a). Model families also read the context labels in different ways, from an attention-weighted vote over the labels to a class-mean readout (Biloš et al., 2026). Under noisy labels and irrelevant features, TabPFN's attention stays on the informative features (Hu & Ghelichi, 2026). Benchmarks measure how well the models predict (Grinsztajn et al., 2022; Erickson et al., 2025; Purucker et al., 2026). For language models, some studies ask how much a prediction depends on the labels of the demonstrations (Min et al., 2022; Kossen et al., 2024). Other studies connect in-context learning to gradient descent, induction heads, and task vectors (Dai et al., 2023; Olsson et al., 2022; Hendel et al., 2023; Todd et al., 2024), or trace a prediction back to individual demonstrations (Zhou et al., 2024). These studies leave one research question open: does one label change how another label is used? We address this question in the following section.

3 Measuring how models use labels

This section presents how we address the research question introduced above. The objective is to measure how a TFM uses its context labels, and in particular whether changing one label changes how other labels influence the prediction. Derivatives are the natural tool, but using them directly runs into a trap: label standardization confines the labels reaching the network to a proper subset of the label space, while derivatives in ambient coordinates also describe behavior off that subset. A model that coincides with a fixed affine map on every admissible label vector can therefore exhibit nonzero second derivatives. We develop tests that remove this ambiguity, distinguishing joint processing from fixed affine prediction and from independent nonlinear transformations of individual labels.

3.1 From a label's value to its influence

Let $i \in \{1, \dots, n\}$ index the context examples. Each example has a feature vector $x_i \in \mathbb{R}^d$ and a label y_i. The context feature matrix and corresponding label vector are $X = [x_1 \cdots x_n]^\top \in \mathbb{R}^{n \times d}$ and $\mathbf{y} = (y_1, \dots, y_n)^\top \in \mathbb{R}^n$, respectively. For a query with feature vector x_*, write the prediction as $\hat{y} = f(x_*, X, \mathbf{y})$. Note that the network itself does not receive the raw labels $\mathbf{y}$. A TFM package first standardizes $\mathbf{y}$ to $\tilde{\mathbf{y}}$ in a wrapper, with components $\tilde{y}_i = (y_i - \bar{y})/s_y$, where $\bar{y}$ is the context-label mean and s_y is the standardization scale. The network computes the internal prediction $\hat{y}_{\text{int}} = g(\tilde{\mathbf{y}})$, and the output conversion returns $\hat{y}$ in the original label units. This distinction determines what we measure: the response of the internal map to standardized labels, with one estimator and a fixed output conversion.

We describe this response through the *label sensitivity* $a_i(\tilde{\mathbf{y}}) = \partial g(\tilde{\mathbf{y}})/\partial \tilde{y}_i$, which measures how strongly label i affects the prediction. Several existing interpretations of in-context learning describe the prediction as affine in the labels:

$$\hat{y}_{\text{int}} = b + \sum_{i=1}^{n} a_i(x_*, X)\, \tilde{y}_i, \tag{1}$$

where the offset b and the weights $a_i(x_*, X)$ depend on the features and the query but not on the labels. For example, the kernel-smoothing interpretation of Collins et al. (2024) takes a weighted

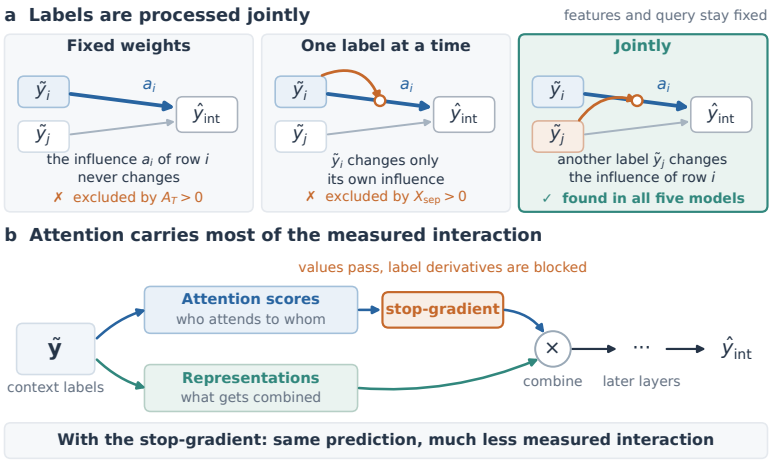

Figure 1: **Labels are processed jointly, and attention carries most of the measured interaction.** **a**: three ways a model could use context labels while the features and the query stay fixed. The blue arrow is the influence a_i of row i on the internal prediction $\hat{y}_{\text{int}}$. With fixed weights, this influence never changes. If labels are handled one at a time, a label can change only its own influence. In joint processing, another label $\tilde{y}_j$ changes how much row i matters, which is what all five models show (§4.2). **b**: inside an attention step, labels can act through the attention scores and through the representations that those scores combine. A stop-gradient, applied at every attention step, lets the scores pass unchanged but stops labels from influencing them when derivatives are taken. The prediction stays the same, while most of the measured interaction disappears (§4.3). The stop-gradient shows which derivative paths carry the interaction in the computation as implemented, not that these paths are necessary.

average of the labels, with nonnegative weights that sum to one. The converged single-layer linear-attention predictor analyzed by Zhang et al. (2024) allows weights of either sign. In these accounts, the sensitivities are the weights themselves, and stay constant when the context labels change. We call such a map a *fixed-weight* map.

Other interpretations let the labels change the weights themselves, so that $a_i = a_i(x_*, X, \tilde{\mathbf{y}})$. A map whose sensitivities change with the labels can still be of two kinds. It can transform *one label at a time* and add the results, so that a label changes only its own sensitivity, or the labels can interact, which we call *joint processing*.

To test whether a model uses a fixed-weight map, one can perturb the labels and compare the sensitivity vectors before and after the perturbation. A fixed-weight map returns the same vector twice, so a large change would seem to show that the weights are not fixed. In the next section, we show that this test falls into a trap, and so does the second-derivative test that extends it. We then present protocols that avoid this trap and certify how a model uses its context labels.

3.2 The standardization trap

A natural way to test for fixed weights is to differentiate the sensitivities once more. The Hessian measures how the sensitivity to label i changes when label j changes:

$$H_{ij}(\tilde{\mathbf{y}}) = \frac{\partial a_i}{\partial \tilde{y}_j} = \frac{\partial^2 g}{\partial \tilde{y}_i \, \partial \tilde{y}_j}.$$

A fixed-weight map has $H = 0$, and it is tempting to conclude in turn that a nonzero H means that the weights are not fixed. This conclusion falls into *the standardization trap*. The rest of this section explains the trap on an example and how to avoid it.

The trap comes from the standardization step (Section 3.1), which does more than change the label units. Because it fixes the mean and the squared norm of the labels, it confines the inputs of the

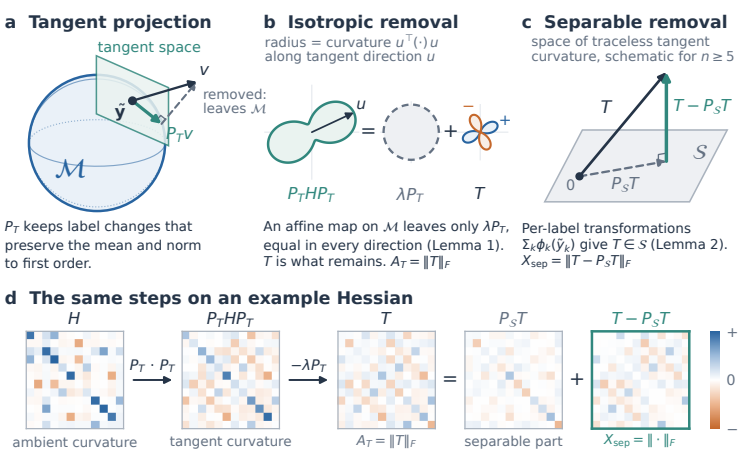

Figure 2: **The standardization trap and the two certificates** (§3.2). **a**: the network only ever sees labels on the standardization manifold $\mathcal{M}$, but derivatives also react to label changes that would leave it. For a label perturbation $v \in \mathbb{R}^n$, the projector P_T keeps the part that slides along $\mathcal{M}$, so that a small step keeps the label mean and changes the label norm only negligibly. **b**: each shape shows how strongly the prediction curves along every direction u within the tangent space, drawn as the distance from the center. This curvature splits into a part that is equal in every direction, λP_T, and the remainder T. A function such as g_a that is affine on $\mathcal{M}$ can produce only the first part (Lemma 1), so $A_T = \|T\|_F$ measures what such a map cannot explain. The equal part is enlarged for visibility. **c**: $\mathcal{S}$ contains every curvature pattern that arises when each label is transformed separately, as in $g_{a,\phi}$. $P_{\mathcal{S}}T$ is the closest such pattern to T, and the length of what is left over is X_{sep} (Lemma 2). **d**: the same steps on a constructed Hessian. It sums terms that act only off $\mathcal{M}$, a separate effect for each label, and couplings between pairs of labels. Only the pair couplings reach $T - P_{\mathcal{S}}T$.

network to the *standardization manifold*

$$\mathcal{M} = \{\tilde{\mathbf{y}} \in \mathbb{R}^n : \mathbf{1}^\top \tilde{\mathbf{y}} = 0,\ \|\tilde{\mathbf{y}}\|^2 = r^2\},$$

where $\mathbf{1}$ is the all-ones vector and r is the norm that the scale s_y imposes, with $r^2 = n$ for the population standard deviation. This manifold is a sphere inside the subspace of mean-zero vectors. Derivatives, however, also describe how g behaves when the labels leave this sphere, which never happens in use.

As a demonstration, consider a fixed-weight map as in Eq. (1), with offset b and weights $a \in \mathbb{R}^n$, and add a term that vanishes on $\mathcal{M}$:

$$g_a(\tilde{\mathbf{y}}) = b + a^\top \tilde{\mathbf{y}} + q\left(\|\tilde{\mathbf{y}}\|^2 - r^2\right), \qquad q \neq 0. \tag{2}$$

On every valid input, the added term is zero. Hence, g_a and the fixed-weight map $b + a^\top \tilde{\mathbf{y}}$ make exactly the same predictions, and no experiment that feeds labels to the model can tell them apart. Their derivatives nevertheless differ. The sensitivities of g_a are $a + 2q\tilde{\mathbf{y}}$, so they change whenever the labels are permuted, and its Hessian is $2qI$, which can be arbitrarily large. *Neither effect changes a single valid prediction.* Naive tests would therefore report that g_a adapts its weights to the labels, although it applies fixed weights wherever it is ever evaluated. Nothing prevents a trained network from containing such terms, because training never evaluates the network away from $\mathcal{M}$ and so never constrains how it extends there.

For any map g, a valid test must therefore depend only on the values that g takes on $\mathcal{M}$. The first step in building such a test is to keep only the label changes that stay on $\mathcal{M}$. A change $v \in \mathbb{R}^n$ of the labels keeps their mean when its entries sum to zero, and for small steps it keeps their norm when it is perpendicular to the current label vector $\tilde{\mathbf{y}}$. The changes that satisfy both conditions are the *tangent directions* at $\tilde{\mathbf{y}}$: they slide along the sphere instead of leaving it (Figure 2a). The matrix $P_T = I - \frac{\mathbf{1}\mathbf{1}^\top}{n} - \frac{\tilde{\mathbf{y}}\tilde{\mathbf{y}}^\top}{r^2}$ turns any change into a tangent one. The Hessian $H_{\tilde{\mathbf{y}}} = \nabla^2_{\tilde{\mathbf{y}}} g(\tilde{\mathbf{y}})$ involves two label changes at once, so we apply P_T on both sides. This gives the *tangent curvature* $P_T H_{\tilde{\mathbf{y}}} P_T$, which records how the sensitivities change when the labels move along tangent directions only.

This correction is not enough. For example, going back to our example with g_a, the Hessian $2qI$ gives the tangent curvature $2qP_T$ (since P_T is idempotent), which is not zero, so the test still fires. The reason is that $\mathcal{M}$ is curved, whereas a tangent direction is a straight line that touches $\mathcal{M}$ at one specific point only. A step of length δ along a unit tangent direction v therefore leaves the sphere slightly, since $\|\tilde{\mathbf{y}} + \delta v\|^2 = r^2 + \delta^2$. The gap δ^2 is too small to affect the slope of a function, but second derivatives measure exactly the terms in δ^2, and the added term of g_a turns this gap into the value $q\delta^2$.

The gap δ^2 does not depend on the direction v of the step. The tangent curvature $2qP_T$ that it produces for g_a therefore has the same value $2q$ in every tangent direction. We call a curvature that is equal in all tangent directions *isotropic*. Lemma 1 shows that this form is not special to g_a: every function that agrees with a fixed-weight map on $\mathcal{M}$ has an isotropic tangent curvature, whatever it does away from $\mathcal{M}$.

Lemma 1 (Affine predictions on the manifold)**.** *Let g be any smooth function on a neighborhood of $\mathcal{M}$ that agrees on $\mathcal{M}$ with an affine function of $\tilde{\mathbf{y}}$. Then at every $\tilde{\mathbf{y}} \in \mathcal{M}$,*

$$P_T H_{\tilde{\mathbf{y}}} P_T = \lambda(\tilde{\mathbf{y}})\, P_T$$

for some scalar $\lambda(\tilde{\mathbf{y}})$.

We provide the proof of Lemma 1 in Appendix A. Every function covered by the lemma is built like g_a: a fixed-weight map plus terms that vanish on $\mathcal{M}$. By Lemma 1, such a map can leave only isotropic curvature. We therefore define the operator T as the tangent curvature minus its isotropic part, and we measure its size A_T:

$$T = P_T H_{\tilde{\mathbf{y}}} P_T - \frac{\operatorname{tr}(P_T H_{\tilde{\mathbf{y}}} P_T)}{n-2} P_T, \qquad A_T = \|T\|_F, \tag{3}$$

where $\|\cdot\|_F$ is the Frobenius norm. The trace of the tangent curvature is the sum of the curvatures along $n-2$ perpendicular tangent directions, so dividing it by $n-2$ gives the average curvature over the tangent directions (Figure 2b), and T keeps only the departure from this average. For g_a, whose Hessian is $H_{\tilde{\mathbf{y}}} = 2qI$, both quantities in Eq. (3) can be computed by hand:

$$\begin{aligned} P_T H_{\tilde{\mathbf{y}}} P_T &= P_T(2qI)P_T = 2q\, P_T P_T = 2q\, P_T, \\ \operatorname{tr}(P_T H_{\tilde{\mathbf{y}}} P_T) &= 2q \operatorname{tr}(P_T) = 2q\,(n-2). \end{aligned}$$

The first line uses $P_T P_T = P_T$, since projecting twice changes nothing, and the second line uses $\operatorname{tr}(P_T) = n-2$, since P_T removes two directions from the n-dimensional identity. Substituting both lines into Eq. (3) gives $T = 2q\, P_T - \frac{2q\,(n-2)}{n-2} P_T = 0$, so the corrected test stays silent on g_a, as it should. The value of q plays no role in this cancellation, so the same argument gives $T = 0$ for every isotropic curvature λP_T, regardless of the value of λ. More generally, T depends only on the predictions on $\mathcal{M}$, regardless of how g extends away from the manifold. *A nonzero A_T therefore excludes fixed-weight prediction on valid labels.*

A nonzero A_T shows that g is nonlinear on $\mathcal{M}$, but nonlinearity alone is not enough to establish joint processing. A model can transform one label at a time and add the results, as in

$$g_{a,\phi}(\tilde{\mathbf{y}}) = g_a(\tilde{\mathbf{y}}) + \sum_{k=1}^{n} \phi_k(\tilde{y}_k), \tag{4}$$

where each smooth function ϕ_k acts on a single label, for example by damping extreme values. For most choices of ϕ_k, this map is not affine on $\mathcal{M}$, and it has $A_T > 0$. Yet no label changes the influence of another one, since the added term contributes only the diagonal matrix $D = \operatorname{diag}\big(\phi_k''(\tilde{y}_k)\big)$ to the Hessian. We cannot detect this case by looking for zero off-diagonal entries, because the projection mixes the coordinates: on $\mathcal{M}$ no label can move alone, since the other labels must compensate to keep the mean and the norm fixed, and $P_T D P_T$ is no longer diagonal. We test instead whether the corrected curvature T could have come from *some* diagonal matrix. Let $\mathcal{S} = \{P_T D P_T - \frac{\operatorname{tr}(P_T D P_T)}{n-2} P_T : D \text{ diagonal}\}$ be the linear space of corrected curvatures that one-label-at-a-time maps, such as $g_{a,\phi}$, can produce, and let $P_{\mathcal{S}}$ denote the orthogonal projection onto it. We measure what remains outside this space:

$$X_{\text{sep}} = \|T - P_{\mathcal{S}} T\|_F. \tag{5}$$

For $g_{a,\phi}$ we have $T \in \mathcal{S}$ and hence $X_{\text{sep}} = 0$, whatever the functions ϕ_k and the coefficient q. Figure 2 illustrates both projections, and Lemma 2 states the general result.

Lemma 2 (Certificate of joint label processing)**.** *Let g be smooth on a neighborhood of $\mathcal{M}$ and agree on $\mathcal{M}$ with an affine-plus-separable map $b + a^\top \tilde{\mathbf{y}} + \sum_k \phi_k(\tilde{y}_k)$. Then $T \in \mathcal{S}$ and $X_{\text{sep}} = 0$ at every point of $\mathcal{M}$. The statistic X_{sep} is extension-invariant: any two smooth maps that agree on $\mathcal{M}$ give the same value there.*

A nonzero X_{sep} therefore certifies that one label changes the influence of another. Both tests are one-sided: a zero result alone does not prove membership in the corresponding class. Table 1 follows the two examples and a joint-processing map through each correction. We provide in Appendix A the proofs, the computation of $P_{\mathcal{S}}$, and numerical checks of this pattern.

Table 1: **From the Hessian to the two certificates.** Each row applies one more correction to the Hessian H of the map. The last three columns give the result for g_a, which is a fixed-weight map on $\mathcal{M}$ (Eq. 2), for the one-label-at-a-time map $g_{a,\phi}$ (Eq. 4, with $D = \operatorname{diag}(\phi_k''(\tilde{y}_k))$), and for a map in which labels interact. Only A_T vanishes for every fixed-weight map, and only X_{sep} also vanishes for every one-label-at-a-time map.

Quantity	What it keeps	g_a	$g_{a,\phi}$	Joint map
Hessian H	every label change	$2qI$	$2qI + D$	$\neq 0$
Tangent curvature $P_T H P_T$	changes that stay on $\mathcal{M}$	$2qP_T$	$2qP_T + P_T D P_T$	$\neq 0$
T, of size A_T	minus the isotropic part	0	in $\mathcal{S}$, $\neq 0$	$\neq 0$
$T - P_{\mathcal{S}} T$, of size X_{sep}	minus the one-label-at-a-time part	0	0	$\neq 0$

The absolute values of A_T and X_{sep} themselves are hard to interpret, because they depend on the scale of the prediction and of its sensitivities, which differs between models and datasets. We therefore report two shares:

$$\bar{A}_T = \frac{A_T^2}{\|P_T H_{\tilde{\mathbf{y}}} P_T\|_F^2}, \qquad \bar{X}_{\text{sep}} = \frac{X_{\text{sep}}^2}{A_T^2}. \tag{6}$$

Both lie between 0 and 1, because each correction removes a part that is orthogonal to the part it keeps, and we set a share to 0 when its denominator vanishes. $\bar{X}_{\text{sep}}$ is the share of the corrected curvature that no one-label-at-a-time map can produce, and like X_{sep} it depends only on the predictions on $\mathcal{M}$. $\bar{A}_T$ is different: its denominator still contains the isotropic part of the tangent curvature, which depends on how g behaves away from $\mathcal{M}$. Apart from vanishing exactly when A_T does, $\bar{A}_T$ serves another purpose: $1 - \bar{A}_T$ is the fraction of the computed tangent curvature that the isotropic correction subtracts, so it shows how much this correction changes the measurement in the tested networks. A fixed-weight map has $\bar{A}_T = 0$, and a one-label-at-a-time map has $\bar{X}_{\text{sep}} = 0$.

4 Experiments and results

Our experiments answer two questions about pretrained TFMs: (1) do the models process their context labels jointly, which is the question that the two certificates were built to settle (Section 4.2), and (2) where does the interaction come from, both whether it is already present before training and which paths inside the network carry it (Section 4.3). Section 4.1 first presents the setup that all experiments share.

4.1 Experimental setup

Models. We evaluate five public TFMs: TabICL-v2 (Qu et al., 2026), TabPFN-v3 (Grinsztajn et al., 2026), TabPFN-v2.6 (Grinsztajn et al., 2025), TabFM 1.0 (Kong & Das, 2026), and TabDPT v1.2 (Ma et al., 2025; Hosseinzadeh et al., 2026). They differ in architecture, attention schedule, readout, and pretraining data (Appendix B.1). The first four are pretrained on synthetic tasks and have related designs: TabICL-v2, TabPFN-v3 and TabFM build a representation of each row with attention across features and rows and then predict in a final in-context stage, whereas TabPFN-v2.6 alternates attention across features and rows in every layer. TabDPT is pretrained on real tables, which makes it the most independent of the five.

Tasks and Protocol. We compute the two certificates on four regression datasets with 256 context rows: one synthetic dataset and the OpenML datasets `kin8nm`, `pol`, and `superconduct`. The

synthetic dataset is a smooth regression task on 16 continuous features that we generate ourselves (Appendix B.2), so that it serves as a control that real tables cannot provide: we can redraw its label noise from the true model, no model can have seen it during pretraining, and it has no outliers or discrete features that could explain what we measure. We call one model on one dataset a *setting*, which gives 20 settings. In each setting, we repeat the measurement on three randomly sampled contexts, with four queries each. Appendix Table 5 lists the datasets, context sizes, and repetitions of every experiment in the paper.

Implementation Details. Since we run our analysis based on TFM responses, we take derivatives with respect to the standardized labels and keep the conversion back to the original label units fixed (Section 3.1). The differentiated prediction comes from a single estimator inside each package, whereas the packages' default predictions combine several estimators, which we test separately (Section 4.2). Second derivatives need an attention implementation that can be differentiated twice, which the fused kernels used in deployment are not, so we switch to standard (eager) attention for these computations (Appendix B.3).

4.2 To what extent do TFMs process labels jointly?

We first determine which of the three descriptions of Section 3.1 fits each TFM: fixed weights, one label at a time, or joint processing (Figure 1a). To this end, we compute the exact Hessian of the internal prediction in each of the 20 settings and apply the two certificates, so that each description is tested by a statistic that is zero whenever the description holds. Table 2 reports the results.

Table 2: **All five models process labels jointly, and most of the interaction passes through attention.** The shares (Eq. (6)) and the relative fall of A_T and X_{sep} when derivatives cannot pass through the attention scores (Section 4.3), in percent, each averaged over the four datasets. A fixed-weight map would give $\bar{A}_T = 0$, and a one-label-at-a-time map would give $\bar{X}_{\text{sep}} = 0$. Appendix Table 8 gives every setting.

	Share (%)		Attention drop (%)	
Model	$\bar{A}_T$	$\bar{X}_{\text{sep}}$	A_T	X_{sep}
TabICL-v2	98.7	63.1	81.6	82.0
TabPFN-v3	98.5	70.8	85.9	90.0
TabPFN-v2.6	97.7	73.4	90.9	95.3
TabFM	99.2	67.5	78.2	82.1
TabDPT	98.3	40.4	68.0	75.3
All models	98.5	63.0	80.9	85.0

Fixed weights do not explain the predictions of any model, since $\bar{A}_T$ is nonzero in every setting (Appendix Table 8). Its average of 98.5% in Table 2 also shows that the isotropic correction subtracts only 1.5% of the computed tangent curvature, so for these models the correction changes the measurement little, although the test would not be valid without it. Transforming one label at a time does not explain the predictions either. Such a map would give $\bar{X}_{\text{sep}} = 0$, whereas $\bar{X}_{\text{sep}}$ averages 63% over the 20 settings and lies between 23% and 92% in every one of them (Appendix Table 8).

The package default inference procedures of the tested models combine several estimators. We tested those settings and found that the combined predictions are not affine in the labels either (Appendix C.2).

4.3 Where do the interactions come from?

The certificates analyzed above establish that labels interact inside the models, but they do not say where this interaction comes from. We first test whether the architecture alone produces the interaction or whether it appears with training, and we then identify which paths inside the network carry it.

Training changes the interaction among labels. We want to know whether the interaction comes with the architecture or appears with training. We therefore compare each trained model with 12

randomly initialized copies of the same architecture on two of the four datasets. A randomly initialized network is already a nonlinear function of its labels, so it departs from fixed weights before training: A_T is nonzero for both the trained models and the random copies and therefore cannot tell them apart. The normalized certificate $\bar{X}_{\mathrm{sep}}$ does separate trained from untrained networks. On the two datasets of this comparison, every query of a trained model gives at least 29%, whereas the random copies stay at most 3.4% (Appendix C.3).

Attention carries most of the measured interaction. Attention combines representations with weights that it computes from attention scores, and labels can change both the representations being combined and the scores that weight them. To separate these contributions, we block derivatives through the attention scores while keeping their values unchanged in the forward computation (Figure 1b), so that the model makes exactly the same prediction while its derivatives retain only the paths that do not pass through these scores. Blocking these derivative paths removes most of the measured interaction (columns "Attention drop" in Table 2). The cross-example residual X_{sep} falls by 85% on average, from 75% for TABDPT to 95% for TABPFN-V2.6, and the corrected curvature norm A_T falls by 81% on average. These percentages describe reductions in norms, whereas the comparison with random copies uses shares of squared curvature, and they neither measure accuracy loss nor establish that these paths are necessary for good predictions. Appendix C.4 compares the curvature operators directly and describes a forward intervention. These results add to the ongoing debate on whether attention weights explain the predictions of a model (Wiegreffe & Pinter, 2019; Jain & Wallace, 2019; Bibal et al., 2022; Bhan et al., 2023).

5 Discussion and conclusion

In the evaluated models, context labels can change how other labels affect the prediction. The packages standardize the labels before the networks see them, and ordinary derivatives also describe how the network would behave on labels that are not standardized, so they can report nonlinearity even for models that apply fixed weights to every valid input. Our two certificates measure behaviors on standardized labels only. The first certificate A_T is zero for every fixed-weight map, and the second certificate X_{sep} is zero for every map that transforms one label at a time. Both are nonzero in all five models, so no model is explained by either fixed weights or transforming its labels one at a time. Of the curvature that departs from fixed weights, 23% to 92% comes from interaction that no one-label-at-a-time map can produce. This interaction is much weaker in randomly initialized copies of the same architectures, which indicates that it comes with training rather than from the architecture alone. Blocking derivatives through the attention scores removes most of it while leaving the predictions unchanged, so in the networks as implemented most of the interaction passes through these scores.

Our findings guide explanations of how TFMs use their context labels. They exclude the fixed-weight account, in which the features alone set the weight of each label, because in every evaluated model a change in one label also changes the weights of the others. Yet they do not identify the computation behind this interaction, since several computations make labels interact: (1) the model may use all labels to infer which task generated the data, so that changing any label changes the inferred task and with it the weight of every label, (2) a local regression may reweight the neighbors of the query by their labels, and (3) the representations of the examples may themselves depend on the labels. Our certificates detect the interaction but cannot tell these computations apart. The fixed-weight account also remains a useful approximation even though it fails as an exact description: a fixed-weight map fitted to a model's predictions explains a median of 72% of the variation in these predictions when the labels are redrawn from the same task (Appendix C.6). An explanation should therefore state for which labels it holds and whether it reproduces only the predictions or also how one label changes the influence of the others.

The remaining question is whether these label interactions improve prediction, since our derivative tests establish their presence but not their necessity. Blocking the attention scores does not answer it either, because the predictions stay the same. Resolving it requires an intervention that removes the interaction from the forward computation and a measurement of the resulting change in predictive accuracy.

AI USE STATEMENT

We used generative AI tools to assist us in writing the code that implements our methods and runs the experiments. The tools also helped refine the wording of the two certificates and their proofs. We drafted the structure and content of the manuscript, and the tools assisted in polishing the text and figures. They also helped us verify every reported number against the saved experimental records. We reviewed the manuscript through several rounds of internal review. We take full responsibility for the final content of this work, including all text, claims, and results produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The supplementary material contains the code of every experiment, the saved outputs behind the reported numbers, and scripts that recompute the tables and figures from these outputs without running any model. The exact Hessians are too large to include, so a few values computed directly from them, listed in its README, cannot be recomputed from the package. Appendix A gives the proofs and the computation of the certificates, and Appendix B lists the models, checkpoints, preprocessing and datasets, with the grid and replication of every experiment in Table 5. The supplementary material also records the package versions and the digests of the loaded model states. Running the models requires a GPU, whereas recomputing the reported numbers from the saved outputs runs on a CPU.

REFERENCES

Kwangjun Ahn, Xiang Cheng, Hadi Daneshmand, and Suvrit Sra. Transformers learn to implement preconditioned gradient descent for in-context learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? Investigations with linear models. In *International Conference on Learning Representations (ICLR)*, 2023.

Yu Bai, Fan Chen, Huan Wang, Caiming Xiong, and Song Mei. Transformers as statisticians: Provable in-context learning with in-context algorithm selection. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Milan Bhan, Nina Achache, Victor Legrand, Annabelle Blangero, and Nicolas Chesneau. Evaluating self-attention interpretability through human-grounded experimental protocol. In *Explainable Artificial Intelligence: First World Conference, xAI 2023*, volume 1903 of *Communications in Computer and Information Science*, pp. 26–46. Springer, 2023. doi: 10.1007/978-3-031-44070-0_2.

Adrien Bibal, Rémi Cardon, David Alfter, Rodrigo Wilkens, Xiaoou Wang, Thomas François, and Patrick Watrin. Is attention explanation? An introduction to the debate. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3889–3900, 2022.

Marin Biloš, James T. Wilson, Anderson Schneider, and Yuriy Nevmyvaka. A mechanistic study of tabular foundation models. *arXiv preprint arXiv:2605.21288*, 2026.

Liam Collins, Advait Parulekar, Aryan Mokhtari, Sujay Sanghavi, and Sanjay Shakkottai. In-context learning with transformers: Softmax attention adapts to function Lipschitzness. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 92638–92696, 2024.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. Why can GPT learn in-context? Language models implicitly perform gradient descent as meta-optimizers. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 4005–4019, 2023.

Nick Erickson, Lennart Purucker, Andrej Tschalzev, David Holzmüller, Prateek Mutalik Desai, David Salinas, and Frank Hutter. TabArena: A living benchmark for machine learning on tabular

data. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2025. URL https://arxiv.org/abs/2506.16791.

Google Research. TabFM source code. GitHub repository, 2026. URL https://github.com/google-research/tabfm.

Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2022.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Benjamin Jäger, Dominik Safaric, Simone Alessi, Adrian Hayler, Mihir Manium, Rosen Yu, Felix Jablonski, Shi Bin Hoo, Anurag Garg, Jake Robertson, Magnus Bühler, Vladyslav Moroshan, Lennart Purucker, Clara Cornu, Lilly Charlotte Wehrhahn, Alessandro Bonetto, Bernhard Schölkopf, Sauraj Gambhir, Noah Hollmann, and Frank Hutter. TabPFN-2.5: Advancing the state of the art in tabular foundation models. *arXiv preprint arXiv:2511.08667*, 2025.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, et al. TabPFN-3: Technical report. *arXiv preprint arXiv:2605.13986*, 2026.

Atharva Gupta, Dhruv Kumar, Murari Mandal, and Saurabh Deshpande. Where computation lives inside TabPFN: Causal localisation of attention head function. *arXiv preprint arXiv:2606.12917*, 2026a.

Aviral Gupta, Armaan Sethi, and Dhruv Kumar. TabPFN through the looking glass: An interpretability study of TabPFN and its internal representations. *arXiv preprint arXiv:2601.08181*, 2026b.

Roee Hendel, Mor Geva, and Amir Globerson. In-context learning creates task vectors. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 9318–9333, 2023.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025. doi: 10.1038/s41586-024-08328-6.

Rasa Hosseinzadeh, Alex Labach, Zexin Xue, Shuyi Han, Valentin Thomas, and Anthony L. Caterini. TabDPT-Turbo: Efficient in-context learning for tabular prediction. *arXiv preprint arXiv:2608.01400*, 2026.

James Hu and Mahdi Ghelichi. Noise immunity in in-context tabular learning: An empirical robustness analysis of TabPFN's attention mechanisms, 2026. arXiv:2604.04868.

Sarthak Jain and Byron C. Wallace. Attention is not explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 3543–3556, 2019.

Weihao Kong and Abhimanyu Das. Introducing TabFM: A zero-shot foundation model for tabular data. Google Research blog, 2026. URL https://research.google/blog/introducing-tabfm-a-zero-shot-foundation-model-for-tabular-data/.

Jannik Kossen, Yarin Gal, and Tom Rainforth. In-context learning learns label relationships but is not conventional learning. In *International Conference on Learning Representations (ICLR)*, 2024.

John M. Lee. *Introduction to Smooth Manifolds*, volume 218 of *Graduate Texts in Mathematics*. Springer, 2nd edition, 2013.

Junwei Ma, Valentin Thomas, Rasa Hosseinzadeh, Alex Labach, Hamidreza Kamkari, Jesse C. Cresswell, Keyvan Golestan, Guangwei Yu, Anthony L. Caterini, and Maksims Volkovs. TabDPT: Scaling tabular foundation models on real data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Ratmir Miftachov, Bruno Charron, and Simon Valentin. Interpretable tabular foundation models via in-context kernel regression. *arXiv preprint arXiv:2602.02162*, 2026.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 11048–11064, 2022.

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do Bayesian inference. In *International Conference on Learning Representations (ICLR)*, 2022.

Thomas Nagler. Statistical foundations of prior-data fitted networks. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pp. 25660–25676, 2023.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, et al. In-context learning and induction heads. *Transformer Circuits Thread*, 2022.

Madhur Panwar, Kabir Ahuja, and Navin Goyal. In-context learning through the Bayesian prism. In *International Conference on Learning Representations (ICLR)*, 2024.

Lennart Purucker, Andrej Tschalzev, Nick Erickson, Gioia Blayer, David Holzmüller, Alan Arazi, Alexander Pfefferle, Mustafa Tajjar, Gaël Varoquaux, and Frank Hutter. Beyond IID: How general are tabular foundation models, really? *arXiv preprint arXiv:2606.30410*, 2026. URL `https://arxiv.org/abs/2606.30410`.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICLv2: A better, faster, scalable, and open tabular foundation model. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026.

Amir Rezaei Balef, Mykhailo Koshil, and Katharina Eggensperger. Is one layer enough? Understanding inference dynamics in tabular foundation models. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026.

Eric Todd, Millicent L. Li, Arnab Sen Sharma, Aaron Mueller, Byron C. Wallace, and David Bau. Function vectors in large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Yao-Hung Hubert Tsai, Shaojie Bai, Makoto Yamada, Louis-Philippe Morency, and Ruslan Salakhutdinov. Transformer dissection: An unified understanding for transformer's attention via the lens of kernel. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4344–4353, 2019.

Johannes von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

Sarah Wiegreffe and Yuval Pinter. Attention is not not explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 11–20, 2019.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit Bayesian inference. In *International Conference on Learning Representations (ICLR)*, 2022.

Han-Jia Ye, Si-Yang Liu, and Wei-Lun Chao. A closer look at TabPFN v2: Understanding its strengths and extending its capabilities. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Ruiqi Zhang, Spencer Frei, and Peter L. Bartlett. Trained transformers learn linear models in-context. *Journal of Machine Learning Research*, 25(49):1–55, 2024. URL https://jmlr.org/papers/v25/23-1042.html.

Zijian Zhou, Xiaoqiang Lin, Xinyi Xu, Alok Prakash, Daniela Rus, and Bryan Kian Hsiang Low. DETAIL: Task demonstration attribution for interpretable in-context learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

A PROOFS AND COMPUTATION OF THE CERTIFICATES

This appendix proves the two lemmas that Section 3.2 states: Lemma 1 for the corrected curvature A_T, and Lemma 2 for the residual X_{sep}. It then gives the linear-algebra computation that turns the definition of X_{sep} into a number, extends the same manifold correction to the first-derivative test of Section 3.1, and calibrates these statistics against maps whose class is known in advance.

A.1 PROOF OF LEMMA 1

The lemma must show that a map which predicts exactly like a fixed-weight map on the standardization manifold $\mathcal{M}$ can still differ from it in its second derivatives, but only in a restricted way: the tangent curvature it can produce is the same in every tangent direction. The idea is that such a map differs from the fixed-weight map only through terms built out of the two functions whose common zero set is $\mathcal{M}$, and each such term adds curvature only along the two directions normal to $\mathcal{M}$, which the tangent projector P_T removes, except for a single scalar rescaling of P_T itself that survives the removal.

Write $u(\tilde{\mathbf{y}}) = \mathbf{1}^\top \tilde{\mathbf{y}}$ for the sum of the standardized labels and $w(\tilde{\mathbf{y}}) = \|\tilde{\mathbf{y}}\|^2 - r^2$ for the deviation of their squared norm from r^2, so that $\mathcal{M} = \{u = 0,\, w = 0\}$ is exactly their common zero set (Section 3.2). Since $\mathbf{1}$ and $\tilde{\mathbf{y}}$ are linearly independent whenever $r > 0$, the gradients $\nabla u = \mathbf{1}$ and $\nabla w = 2\tilde{\mathbf{y}}$ are independent on $\mathcal{M}$, so $\mathcal{M}$ is a regular level set of the pair (u, w). If g agrees on $\mathcal{M}$ with the affine map $b + \tilde{a}^\top \tilde{\mathbf{y}}$, Hadamard's lemma applied in local coordinates adapted to (u, w) writes the difference between g and this affine map as a combination of u and w with smooth coefficient functions p and q defined near each point of $\mathcal{M}$ (Lee, 2013, Ch. 5 and Thm. C.15):

$$g = b + \tilde{a}^\top \tilde{\mathbf{y}} + p(\tilde{\mathbf{y}})\, u(\tilde{\mathbf{y}}) + q(\tilde{\mathbf{y}})\, w(\tilde{\mathbf{y}}).$$

Because u and w vanish exactly on $\mathcal{M}$, this equation only restates that g and the affine map coincide there, while attaching a smooth multiplier to each of the two ways of moving off $\mathcal{M}$.

At a point of $\mathcal{M}$, $u = w = 0$ while their gradients need not vanish, so differentiating gives

$$\begin{aligned} \nabla g &= \tilde{a} + p\,\mathbf{1} + 2q\,\tilde{\mathbf{y}}, \\ H &= 2qI + \mathbf{1}\nabla p^\top + \nabla p\,\mathbf{1}^\top + 2\tilde{\mathbf{y}}\nabla q^\top + 2\nabla q\,\tilde{\mathbf{y}}^\top. \end{aligned}$$

The gradient of g on $\mathcal{M}$ is therefore the fixed weight vector $\tilde{a}$ plus a term along each of the two normal directions, and the Hessian keeps the isotropic term $2qI$ and picks up four more terms, each proportional to $\mathbf{1}$ or to $\tilde{\mathbf{y}}$.

The normal space of $\mathcal{M}$ at $\tilde{\mathbf{y}}$ is spanned by $\mathbf{1}$ and $\tilde{\mathbf{y}}$, which are orthogonal because the standardized labels have zero mean, $\mathbf{1}^\top \tilde{\mathbf{y}} = 0$. The tangent projector $P_T = I - \mathbf{1}\mathbf{1}^\top/n - \tilde{\mathbf{y}}\tilde{\mathbf{y}}^\top/r^2$ of Section 3.2 therefore has rank $n - 2$ and annihilates both normal directions, $P_T\mathbf{1} = P_T\tilde{\mathbf{y}} = 0$. Projecting the Hessian above on both sides removes all four extra terms, since each carries a factor of $\mathbf{1}$ or $\tilde{\mathbf{y}}$, and leaves

$$P_T H P_T = 2q\, P_T,$$

which is the isotropic form the lemma claims, with $\lambda(\tilde{\mathbf{y}}) = 2q(\tilde{\mathbf{y}})$. This proves Lemma 1.

The intrinsic Riemannian Hessian of g restricted to $\mathcal{M}$ is the curvature computed from the values that g takes on $\mathcal{M}$ alone, and not from how g extends off the manifold. The gradient identity above shows that it differs from $P_T H P_T$ only by the isotropic correction $-(\tilde{\mathbf{y}}^\top \nabla g / r^2) P_T$. Since this correction is itself proportional to P_T, it is removed along with every other isotropic term once the trace is subtracted, so the traceless part of $P_T H P_T$ equals the traceless part of the intrinsic Hessian. Writing $\mathrm{traceless}(B) = B - \mathrm{tr}(B)P_T/(n-2)$ for a tangent-projected operator $B = P_T B P_T$, this traceless part is exactly $T = \mathrm{traceless}(P_T H P_T)$ of Eq. (3). Hence T depends only on the predictions that g makes on $\mathcal{M}$ and not on how g is defined away from it, while the isotropic part $\lambda(\tilde{\mathbf{y}})$ cannot be identified from on-manifold behavior alone, since any value of λ is compatible with the same restriction of g to $\mathcal{M}$.

A.2 PROOF OF LEMMA 2

The lemma must show that the same argument extends to the larger class of maps that transform each label separately before combining the results, so that the corrected curvature such a map can

produce always lies in the space $\mathcal{S} = \{\text{traceless}(P_T D P_T) : D \text{ diagonal}\}$ of Section 3.2, and the residual X_{sep} of Eq. (5) is zero.

Let $f = b + a^\top \tilde{\mathbf{y}} + \sum_k \phi_k(\tilde{y}_k)$ agree with g on $\mathcal{M}$, where each ϕ_k acts on a single standardized label as in $g_{a,\phi}$ (Eq. (4)). The difference $g - f$ vanishes identically on $\mathcal{M}$, so applying the decomposition of the previous proof to $g - f$ shows that its own tangent curvature is isotropic. The map f itself produces the tangent curvature $P_T D P_T$, where $D = \text{diag}(\phi_k''(\tilde{y}_k))$ collects the curvature of each ϕ_k and the linear term $a^\top \tilde{\mathbf{y}}$ contributes none. Adding this curvature back gives

$$P_T H_g P_T = P_T D P_T + 2q\, P_T$$

for the tangent curvature of g itself. Removing the trace cancels the isotropic term $2qP_T$ and leaves $T = \text{traceless}(P_T D P_T)$, which is a member of $\mathcal{S}$ by definition, so $T \in \mathcal{S}$ and hence $X_{\text{sep}} = \|T - P_{\mathcal{S}} T\|_F = 0$.

As in the proof of Lemma 1, the traceless projected Hessian of any difference that vanishes on $\mathcal{M}$ is itself zero, so T, and with it X_{sep}, does not depend on which off-manifold extension g happens to have.

Both certificates are one-sided: a value of zero is consistent with the corresponding class of maps, fixed-weight for A_T and one-label-at-a-time for X_{sep}, but it does not by itself prove that a model belongs to that class, since some other map outside the class could in principle also give a zero value on the labels tested.

A.3 Computing the projection onto $\mathcal{S}$

Evaluating X_{sep} requires the orthogonal projection $P_{\mathcal{S}} T$ of the measured curvature T onto $\mathcal{S}$, the space of corrected curvatures that a one-label-at-a-time map can produce. Because $\mathcal{S}$ is spanned by one operator per label, this projection reduces to a small linear solve rather than a search over label maps.

For each label k, let E_{kk} be the matrix with a single 1 in position (k, k) and zeros elsewhere, so that a diagonal matrix D is a combination of the E_{kk}, and set $B_k = \text{traceless}(P_T E_{kk} P_T)$ for the corrected curvature that this single-label bump produces. The operators $B_1, \ldots, B_n$ therefore span $\mathcal{S}$.

Write $\beta = \text{diag}(T)$ for the vector of diagonal entries of the measured curvature. Because T is already tangent-projected and traceless, its inner product with each basis operator reduces to a single diagonal entry, $\langle T, B_k \rangle = T_{kk}$, and the projection coefficients solve the normal equations of this basis, $Gc = \beta$, where the Gram matrix of inner products between basis operators is

$$G_{jk} = (P_T)_{jk}^2 - \frac{(P_T)_{jj}(P_T)_{kk}}{n-2}.$$

The orthogonal projection is then $P_{\mathcal{S}} T = \sum_k c_k B_k$ with $c = G^+ \beta$, using the Moore–Penrose pseudoinverse because the basis is not independent, and $X_{\text{sep}} = \left\| T - \sum_k c_k B_k \right\|_F$.

This solve stays well posed despite the rank deficiency of G. If v lies in the null space of G, the corresponding combination of basis operators vanishes, $\sum_k v_k B_k = 0$, so its inner product with T is also zero, $v^\top \beta = 0$, and the system $Gc = \beta$ therefore remains consistent on the subspace where G is singular. One such null direction is known in closed form, $G\mathbf{1} = 0$, because summing the E_{kk} over every label gives the identity and $\text{traceless}(P_T I P_T) = \text{traceless}(P_T) = 0$. The pseudoinverse handles this direction by returning the minimum-norm solution among those that solve the system.

An independent implementation, which projects onto $\mathcal{S}$ using an orthonormal basis rather than the B_k above, gives the same residual to a relative error of at most 2×10^{-16} on the coupled control below and on model Hessians of all five models. On the two null controls, both implementations return values at the floating-point floor of about 3×10^{-14}.

A nonzero X_{sep} is possible only when $\mathcal{S}$ is a proper subspace of the space in which T lives. $\mathcal{S}$ is the image of the diagonal matrices under $D \mapsto \text{traceless}(P_T D P_T)$, so its dimension is at most $n-1$ because this map has a nontrivial null space. The traceless symmetric operators on the tangent space, where T lives, form a space of dimension $(n-2)(n-1)/2 - 1$, which exceeds $n-1$ once $n \geq 5$. For $n \geq 5$, $\mathcal{S}$ is therefore always a proper subspace, a condition met by a wide margin at the

context size $n = 256$ that the paper uses for the certificates. For smaller contexts, $\mathcal{S}$ fills the whole space at most label vectors but not at all of them: at $n = 4$ and $\tilde{\mathbf{y}} \propto (1, 1, -1, -1)$, $\mathcal{S}$ has dimension one inside a space of dimension two.

A.4 The corrected perturbation test

Section 3.1 describes a first test for fixed-weight prediction: perturb the context labels and compare the label sensitivities $a_i = \partial g(\tilde{\mathbf{y}})/\partial \tilde{y}_i$ before and after. Section 3.2 shows that this test falls into the standardization trap, as does the second-derivative test built from the Hessian: a map can change its sensitivities under a permutation of the labels while still making exactly the same predictions on every valid input. This section defines the naive statistic precisely and repairs it with the same manifold correction used for A_T and X_{sep}.

Writing $a(\tilde{\mathbf{y}}) = \nabla_{\tilde{\mathbf{y}}} g(\tilde{\mathbf{y}})$ for the vector collecting these sensitivities and P_π for the matrix that permutes the context labels according to a permutation π, the naive permutation statistic averages the relative change of this vector over queries x_* and permutations π:

$$\Delta_{\text{perm}} = \mathbb{E}_{x_*,\pi}\left[\frac{\|a(P_\pi \tilde{\mathbf{y}}) - a(\tilde{\mathbf{y}})\|_2}{\|a(\tilde{\mathbf{y}})\|_2}\right]. \tag{7}$$

A fixed-weight map keeps a unchanged under any permutation of the labels, so $\Delta_{\text{perm}} = 0$ for such a map exactly. But Δ_{perm} can be nonzero for a map that is nonetheless a fixed-weight map on every valid input. For g_a (Eq. (2)), whose sensitivities are $a + 2q\tilde{\mathbf{y}}$ (Section 3.2), a permutation changes them by

$$a(P_\pi \tilde{\mathbf{y}}) - a(\tilde{\mathbf{y}}) = 2q\,(P_\pi \tilde{\mathbf{y}} - \tilde{\mathbf{y}}),$$

a vector that lies entirely in the span of $\tilde{\mathbf{y}}$ and $P_\pi \tilde{\mathbf{y}}$, and Δ_{perm} reports this change even though g_a makes the same predictions as the fixed-weight map $b + a^\top \tilde{\mathbf{y}}$ everywhere it is ever evaluated.

The repair projects this span out before comparing sensitivities, the same correction that P_T performs for the second-derivative test. For each of six permutations π_k, let $a_0 = \nabla g(\tilde{\mathbf{y}})$ and $a_k = \nabla g(P_{\pi_k} \tilde{\mathbf{y}})$ denote the sensitivities at the identity and permuted labels, and let Q_k project orthogonal to $\text{span}\{\mathbf{1}, \tilde{\mathbf{y}}, P_{\pi_k} \tilde{\mathbf{y}}\}$. The intersection-projected displacement averages the residual change left after this projection:

$$\Delta^{\perp}_{\text{perm}} = \frac{1}{6}\sum_{k=1}^{6} \frac{\|Q_k(a_k - a_0)\|_2}{\|Q_k a_0\|_2 + \eta}.$$

A second statistic asks whether a single fixed weight vector can fit the tangent-projected sensitivities at the identity and all six permutations at once. With $P_k = P_T(P_{\pi_k} \tilde{\mathbf{y}})$ the tangent projector evaluated at the k-th permuted point, and P_0 the tangent projector at the identity, the least-squares residual is

$$\text{LS} = \frac{\left(\min_{\hat{a}} \sum_{k=0}^{6} \|P_k(\hat{a} - a_k)\|_2^2\right)^{1/2}}{\left(\sum_{k=0}^{6} \|P_k a_k\|_2^2\right)^{1/2} + \eta}.$$

Both statistics use the numerical guard $\eta = 10^{-300}$ to keep the ratio finite when its denominator is itself close to zero.

Both statistics vanish for every map that is affine on $\mathcal{M}$, by the gradient identity from the proof of Lemma 1: on $\mathcal{M}$, the gradient of such a map differs from the fixed weight vector $\tilde{a}$ only by components along $\mathbf{1}$ and $\tilde{\mathbf{y}}$, which at a permuted point become components along $\mathbf{1}$ and $P_{\pi_k} \tilde{\mathbf{y}}$, exactly the directions that Q_k removes and that the single choice $\hat{a} = \tilde{a}$ absorbs once the sensitivities are projected by P_k. Reported values average the per-query statistics within each setting.

A.5 Calibration on maps with known behavior

We check the lemmas and the statistics above against three constructed maps whose class is fixed by design, rather than only against the trained models, so that a nonzero or a zero value can be read against a known answer before it is trusted on a model whose class is unknown. All three controls pass through the same estimator as the models, so the check exercises the same computation that produces the numbers reported for the models and not a separate shortcut.

Table 3: **Calibration on maps with known behavior.** Values use the same estimator as the models, at $n = 256$ in float64. The raw statistics Δ_{perm} and $\|P_T H P_T\|_F$ can be nonzero even when the map is affine on $\mathcal{M}$, which is the trap of Section 3.2 made numerical. Only X_{sep} is also zero on a map that transforms one label at a time. The first-order statistics LS and $\Delta^{\perp}_{\text{perm}}$ are zero on every map that is affine on $\mathcal{M}$.

Control	Δ_{perm}	$\|P_T H P_T\|_F$	A_T	X_{sep}	LS	$\Delta^{\perp}_{\text{perm}}$
Affine on $\mathcal{M}$	1.21	38.7	4×10^{-14}	3×10^{-14}	6×10^{-14}	7×10^{-14}
Separable on $\mathcal{M}$	1.23	9.7	1.11	3×10^{-14}	0.36	0.53
Coupled	0.96	0.73	0.72	0.72	0.75	0.96

The affine control on $\mathcal{M}$ is a fixed-weight map, built like g_a (Eq. (2)). The separable control on $\mathcal{M}$ transforms one label at a time, built like $g_{a,\phi}$ (Eq. (4)). The coupled control lets labels interact. Table 3 reports all six statistics on the three controls.

The two raw statistics are far from zero on the affine control: the naive permutation statistic Δ_{perm} and the norm of the tangent curvature before its isotropic part is removed. This is the standardization trap of Section 3.2 made numerical, since a fixed-weight map still produces a large apparent change under these uncorrected measurements. The corrected curvature A_T removes this artifact and is zero, to within floating-point error, on the affine control, but it is not zero on the separable control, which does transform each label nonlinearly and so has genuine tangent curvature. X_{sep} is zero on both the affine and the separable control, because Lemma 2 covers the affine control as the special case $\phi_k = 0$ of the affine-plus-separable class, and it is nonzero on the coupled control, where every statistic in the table is nonzero.

The separable control's combination of a nonzero A_T and a zero X_{sep} is the pattern that the main claim of Section 4.2 rests on: a model can fail the fixed-weight test and still pass the one-label-at-a-time test, and the calibration confirms that the estimator reports exactly this pattern when it is the true answer. Float32 calibration puts a floor under the zeros in the table, $A_T \le 1.1 \times 10^{-5}$ on the affine null and $X_{\text{sep}} \le 9.3\times10^{-6}$ on the separable null, which corresponds to a share of the squared curvature $\bar{X}_{\text{sep}}$ (Eq. (6)) of 7×10^{-11}.

A separate, descriptive check perturbs pairs of labels directly rather than computing the Hessian. Because a bump to one label that preserves the mean and the norm of the labels changes the standardized values of other rows as well, such pairwise perturbations can still respond to the separable control even though it transforms only one label at a time, so they remain a descriptive check and not a substitute for X_{sep}.

B Experimental details

This appendix documents the models, the datasets, and the Hessian computations behind the results of Sections 3 and 4, in the order a reader needs them: which models we evaluate and where each one first receives the context labels (Appendix B.1), which datasets and how many repetitions every experiment in the paper uses (Appendix B.2), and how we compute and validate the exact Hessians that the two certificates depend on (Appendix B.3).

B.1 Models and where the labels enter

This subsection specifies the five models evaluated in Section 4.2, the estimator inside each package whose output our derivatives target, and the point in each architecture where the context labels first enter the computation, so that the certificates of Section 3 can be traced back to a documented instrument. Table 4 lists the five models together with the *readout* we differentiate in each one: the specific function of the network's output that supplies the internal prediction $\hat{y}_{\text{int}}$ of Section 3.1. We differentiate this single readout inside each package rather than its deployed default prediction, because that default combines several such estimators, and Appendix C.2 tests the default predictions separately.

Table 4: **The five evaluated models and the readout we differentiate.** The identifier names the checkpoint or model version that we load, and the readout names the function of the network's output that supplies the internal prediction $\hat{y}_{\text{int}}$ rather than the package's default combined prediction. Every derivative computation in this paper uses this single readout, except the readout-head check of Appendix C.1.

Model	Identifier	Readout
TABICL-V2 (Qu et al., 2026)	`tabicl-regressor-v2`	linear mean-over-quantiles
TABPFN-V3 (Grinsztajn et al., 2026)	`ModelVersion.V3`	bucket-softmax
TABPFN-V2.6 (Grinsztajn et al., 2025)	`ModelVersion.V2_6`	bucket-softmax
TABFM 1.0 (Kong & Das, 2026)	`tabfm-1.0.0` (regression)	scalar regression output
TABDPT v1.2 (Hosseinzadeh et al., 2026)	`tabdpt1_2.safetensors`	bucket-softmax

The five models also differ in where the context labels first reach the network, which matters for reading the results of Section 4.2. TABICL-V2 introduces the standardized label vector at two points in its computation, once before the column-wise set transformer that builds per-feature representations from the context, and once more at the in-context stage that forms the prediction from those representations, so that the same label vector we differentiate feeds both sites in a full forward pass. TABPFN-V3 follows the same two-stage design: it adds an encoding of each context label to the cell embeddings of that row before the attention that builds the row representations, and a second encoding before its in-context stage, and the same label vector feeds both sites. TABPFN-V2.6 injects the labels once, as an additional column that its alternating attention across features and rows then shares across the context. According to its released code (Google Research, 2026), TABFM also introduces the labels at two points: it adds an encoding of each context label to every feature cell of that row before attention across features and rows, and a second encoding to the compressed row representation before the in-context stage, and the same label vector feeds both sites. Released TABDPT v1.2 (Hosseinzadeh et al., 2026) instead re-encodes the labels into the values that attention combines at every layer, so its queries and keys receive no direct label injection even though its deeper residual streams still carry label-derived information indirectly. This description holds for the released TABDPT v1.2 inference code that we evaluate.

We run each package with its default feature preprocessing for a single estimator, except that we turn off the feature normalization of TABFM, and we run every derivative computation in `float32`. We differentiate with respect to the internal standardized label vector $\tilde{\mathbf{y}}$ of Section 3.1, and we treat the output conversion back to raw label units as a fixed map that plays no part in the derivative. Package versions, constructors, runtime settings, and the digests that identify the exact model state loaded for each experiment are recorded in the supplementary material.

Forward passes and first-order derivatives use each package's own attention computation, with mixed precision and the optional fused kernels of TABICL-V2 and TABDPT turned off, and the Hessian experiments select the standard, non-fused attention backend for the whole run. Second derivatives need an attention implementation with a second backward pass, so every Hessian in this paper substitutes standard (eager) softmax attention, and Appendix B.3 validates this substitution against the deployed kernels directly. Under eager attention, the first derivatives of the readout match those computed with the native kernels to a relative error of at most 8×10^{-5} for all five models, and the forward prediction of TABFM matches its native prediction to 1.2×10^{-5} standard deviations of the output.

B.2 DATASETS AND PROTOCOL

This subsection lists the datasets behind every experiment in the paper and the grid and replication each one uses, so that Table 5 serves as a single reference for what a given result covers and how many times we repeat the measurement.

Datasets. The main experiment of Section 4.2 evaluates four datasets: a synthetic regression task we call `synthetic_mi` (the generator paragraph below), and the OpenML datasets `kin8nm` (189), `pol` (201), and `superconduct` (43174). An eight-dataset *core* grid extends this set with `ailerons` (296), `abalone` (44956), `concrete` (44959), and `house_16H` (574), and supplies

Table 5: **Grid and replication of every experiment in the paper.** Each row names an experiment and gives its grid (the datasets and models it covers) and its replication. A *setting* is one model evaluated on one dataset. *Stat 4*, *Core 8*, and *Core 7* are the dataset grids defined in the text. n is the number of context rows, n_q the number of query points, M the number of label draws used to fit a surrogate, and *probes* the number of random tangent directions used to estimate a Hessian statistic without forming the full matrix (Appendix B.3).

Experiment	Grid and settings	Replication
First-order tests	Core 8, 4 models, 32	$n = 256$, 3 contexts, $n_q = 8$, 6 permutations
Projected Hessian	Stat 4, 4 models, 16	$n_q = 4$, 3 contexts, 16 paired probes
Probe-based certificates	Stat 4, 4 models, 16	$n = 256$, 3 contexts, 64 probes, $n_q = 8$
Forward second differences	Stat 4, 4 models, 16	$n = 256$, 3 contexts, first query, 4 tangent directions per context, 3 step sizes
Exact Hessian	Stat 4, 5 models, 20	$n = 256$, 3 contexts, $n_q = 4$
Hessian validation	2 datasets, 5 models, 10	$n = 256$
Readout head	2 datasets, 2 models, 4	$n = 256$, 2 contexts, $n_q = 2$
Default predictions	2 datasets, 3 models, 6	2 contexts, and the feature orderings of TABDPT on the same 2 datasets
Surrogate replacement	Core 7, 3 models, 21	$n = 96$, $M = 400$, 300 test queries
Observed-feature comparator	Core 7, 4 models, 28	$n = 96$, 300 test queries
Task-preserving surrogate	3 datasets, 4 models, 12	$n = 64$, $n_q = 6$, $M = 210$ (wild bootstrap: `kin8nm`, `pol`, `superconduct`)
Constructed local model	3 datasets, 3 models, 9	$n = 128$, $n_q = 5$, 24 label draws (construction: 3 synthetic tasks at $n = 96$)
Context size, R_T	2 datasets, 3 models, 6	$n = 128$–1024, 1 context, $n_q = 2$, 8 probes
Context size, $\bar{X}_{\text{sep}}$	2 datasets, 2 models, 4	$n = 128$, 256, 512, 2 contexts, $n_q = 2$
Random copies	2 datasets, 5 models, 10	12 copies per model, each evaluated on both datasets
Random copies, Δ_{perm}	2 datasets, 4 models, 8	$n = 128$ (TABICL-V2: 256), 12 copies per model
Attention-entropy control	2 datasets, 2 models, 4	3 copies per model, each evaluated on both datasets
All-layer route patch	Stat 4, 4 models, 16	$n = 256$, $n_q = 8$, 6 permutations
Controlled models	Gaussian-process task	3 seeds, and a five-seed extension of 75 runs

the settings for the first-order tests and for the surrogate-replacement panel, except that the seven-dataset surrogate panel excludes `abalone`.

Table 5 records the grid and the replication of every experiment in the paper, writing *Stat 4* for the four-dataset main grid above, *Core 8* for the eight-dataset grid, and *Core 7* for its seven-dataset restriction, each crossed with the stated number of models. The rows with four models do not include TABFM, which enters only the exact-Hessian experiments and the Hessian validation of Appendix B.3.

Pretraining overlap. The evaluated TABDPT v1.2 checkpoint (TabDPT-Turbo) was pretrained on 1,445 OpenML datasets, from which its authors removed possible duplicates of the CC18, CTR23, and TabArena benchmarks, and no list of these datasets is released (Hosseinzadeh et al., 2026). We therefore cannot check which of our real datasets it saw during pretraining. The 123-dataset manifest of the original TABDPT (Ma et al., 2025) contains `ailerons` directly and `pol` and `house_16H` only through classification variants. Such an overlap does not change the measured response of the evaluated map, but it limits how far TABDPT's results on these datasets can be read as evidence of generalization independent of pretraining.

Metrics. The sensitivities $a_i = \partial \hat{y}_{\text{int}} / \partial \tilde{y}_i$ that the certificates use are local derivatives, and they describe a rate of change rather than an additive contribution to a finite prediction change. The first-

order tests permute the standardized labels inside the model, so the context rows stay aligned without any further check. The surrogate experiments of Appendix C.6 fit an affine surrogate to a model's predictions and evaluate it on label draws held out from fitting. The surrogate-replacement panel fits it on 400 label permutations, choosing the ridge regularization from a grid spanning 10^{-3} to 10^{2} on an 85/15 split of these draws, and evaluates it on 100 fresh permutations. The task-preserving surrogate fits it on 210 of 300 label draws, choosing the regularization from a grid spanning 10^{-3} to 10^{3} by leave-one-out cross-validation, and evaluates it on the remaining 90. The task-preserving draws of that appendix change the moments of the labels and are fitted in raw label coordinates, unlike the derivative measurements elsewhere in this appendix, which use standardized labels. Because the fit uses finitely many draws and a regularized estimator, a surrogate R^2 below 1 does not by itself establish that the residual gap reflects the underlying map rather than fitting noise or the chosen regularization.

Synthetic generator. Features follow $x \sim \mathcal{U}[0,1]^{16}$. Rows of $A \in \mathbb{R}^{n_{\mathrm{idx}} \times 16}$ are drawn from a standard normal. The experiments on the certificate grid of Table 8 scale each row to unit Euclidean norm. These are the exact and probe-based certificates, the first-order tests, the forward second differences, the readout-head check, the context-size check of $\bar{X}_{\mathrm{sep}}$, the default-prediction check, the certificates of the random copies, and the Hessian validation on TABICL-V2 and TABFM. The other experiments, including the context-size check of R_T, divide A by its largest column sum of absolute values. The target is

$$f(x) = \sum_{k=1}^{n_{\mathrm{idx}}} \sum_{j=1}^{6} w_{kj} \exp\left[-\frac{(A_k^\top x - c_{kj})^2}{2(0.2)^2} \right], \qquad y = f(x) + 0.05\varepsilon,$$

where $w_{kj}, \varepsilon \sim \mathcal{N}(0,1)$ and $c_{kj} = 0.5 \sum_i A_{ki} + u_{kj}$ with $u_{kj} \sim \mathcal{U}(-0.6, 0.6)$. The core task uses four indices.

B.3 Computing and validating the Hessians

Second derivatives need attention implemented with eager softmax, because the fused attention kernels that packages deploy do not support a second backward pass (Appendix B.1), so every exact Hessian in this paper comes from a different computational path than the predictions and first-order sensitivities that accompany it. Before treating these Hessians as evidence, we must show that substituting eager attention at this one step leaves the computed derivatives unchanged, which is what this subsection validates.

Assembling and estimating the Hessians. We assemble each exact Hessian one column at a time. Differentiating the full sensitivity vector $a = \nabla_{\tilde{\mathbf{y}}} g(\tilde{\mathbf{y}})$ with respect to a single label $\tilde{y}_j$ gives $He_j = \partial a / \partial \tilde{y}_j$, the j-th column of $H_{\tilde{\mathbf{y}}}$, where e_j is the j-th standard basis vector in $\mathbb{R}^n$, and stacking its n columns recovers the exact Hessian. We compute each column twice on the same contexts, once from the full network and once with derivatives blocked from passing through the attention scores, the intervention of Section 4.3.

Forming every column of a full Hessian at every setting in the paper would be costly to repeat, so several of the appendix analyses that do not need the exact matrix instead estimate A_T from a small set of shared tangent Rademacher probes. These probes are random tangent-space directions with independent ± 1 entries, and we use 8, 16 or 64 of them per setting to estimate the projected norm and the trace of the tangent curvature $P_T H_{\tilde{\mathbf{y}}} P_T$. The squared trace uses a split-product estimator, which multiplies two independent probe-based estimates of the trace rather than squaring one, so that the noise of a single estimate does not enter as a bias. Because a probe-based estimate of A_T can occasionally fall outside the range that a true squared norm allows, we clamp it to its admissible range, which introduces a small null bias in return: in a 64-probe simulation of the affine control g_a (Eq. 2), this bias has mean 0.48 and 95th percentile 1.60, small against the projected norm of 38.7 measured in the same simulation. The exact Hessians reported in Table 2 do not use this clamp. We also re-center the standardized labels inside each tangent projector, which compensates for small floating-point differences between how different packages implement the standardization wrapper of Section 3.1.

Validating the eager-attention path. The functions we differentiate compose linear maps, softmax, smooth activations (GELU, SiLU), and normalization layers (LayerNorm, RMSNorm) away

Table 6: **Validating the eager-attention Hessians against the native fused-kernel gradient**, at $n = 256$ on `synthetic_mi` and `kin8nm`, two of the four datasets in the main grid. For each model–dataset setting, $\varepsilon^\star$ is the selected finite-difference step, cos and rel-L^2 are the cosine similarity and relative L^2 error between the eager Hessian-vector product and the central finite difference of the native gradient along the same probe direction, reported as the median over probes and queries with the maximum in parentheses, matched is the fraction of Hessian columns on which the two computations agree, and $\|H\|_F/\|a\|$ gives the scale of the Hessian relative to the sensitivity vector for context, not a validation criterion. Every setting meets the thresholds stated in the text.

Model	Dataset	$\varepsilon^\star$	cos	rel-L^2 med (max)	matched	$\|H\|_F/\|a\|$
TABICL-V2	`synthetic_mi`	3×10^{-3}	1.0000	0.0008 (0.0012)	1.00	5.51
TABICL-V2	`kin8nm`	3×10^{-3}	1.0000	0.0010 (0.0020)	1.00	4.62
TABPFN-V3	`synthetic_mi`	3×10^{-3}	1.0000	0.0014 (0.0046)	1.00	7.29
TABPFN-V3	`kin8nm`	3×10^{-3}	1.0000	0.0020 (0.0063)	1.00	4.99
TABPFN-V2.6	`synthetic_mi`	3×10^{-3}	1.0000	0.0004 (0.0006)	1.00	3.02
TABPFN-V2.6	`kin8nm`	3×10^{-3}	1.0000	0.0004 (0.0005)	1.00	3.14
TABFM	`synthetic_mi`	1×10^{-2}	1.0000	0.0024 (0.0033)	1.00	22.1
TABFM	`kin8nm`	1×10^{-2}	1.0000	0.0034 (0.0065)	1.00	5.78
TABDPT	`synthetic_mi`	1×10^{-3}	1.0000	0.0003 (0.0004)	1.00	8.40
TABDPT	`kin8nm`	1×10^{-3}	1.0000	0.0004 (0.0006)	1.00	6.93

from their degenerate points, and the discrete steps of the standardization wrapper sit outside the standardized-label leaf that these derivatives are taken against, so nothing discontinuous enters the derivative itself. We compare the exact Hessian-vector products computed under eager attention with central finite differences of the gradient returned by each package's native fused kernel:

$$\frac{a(\tilde{\mathbf{y}} + \varepsilon v) - a(\tilde{\mathbf{y}} - \varepsilon v)}{2\varepsilon},$$

where v is a probe direction and ε the finite-difference step size. We use the same seeded Rademacher directions for both computations, together with unit-vector directions that validate individual Hessian columns directly, and we select the step $\varepsilon^\star$ at a plateau in the relative error where adjacent step sizes give converged answers.

At $n = 256$, every one of the ten model–dataset settings we tested meets our validation thresholds: cosine similarity of at least 0.99, relative error of at most 0.05, and a matched-column fraction of at least 0.90 between the two computations. The agreement is in practice closer than these thresholds require. Every median cosine rounds to 1.0000, the highest median relative error is 0.0034, every matched-column fraction reaches 1.00, and the selected steps are 10^{-3} or 3×10^{-3}, except for TABFM, whose `float32` finite differences become noisy below 10^{-2}. Table 6 reports these ten settings, and the supplementary material holds the full validation and probe records.

C Additional results

C.1 Certificates in every setting

This subsection supports Table 2 of the main text (Section 4.2), which reports each certificate averaged over four datasets per model. It adds three things to that table: (1) the per-setting values behind its means, (2) a first-order version of the fixed-weight test that does not depend on differentiating the sensitivities twice, and (3) checks that the interaction found by the second-order certificates is not an artifact of the readout head or of the exact-Hessian instrument used to compute it. It closes with a look at how the certificate behaves at context sizes other than the $n = 256$ rows used in the main experiments.

First-order tests. A first-order counterpart to the fixed-weight test avoids differentiating the sensitivities twice. Appendix A defines the intersection-projected displacement $\Delta^{\perp}_{\mathrm{perm}}$ and the least-squares residual LS, two statistics built from the tangent gradient alone that are zero for every map that agrees with a fixed-weight map on $\mathcal{M}$. Both reject the affine null on all 32 core settings, with

Table 7: **First-order and probe-based certificate results, by setting.** Each row averages three sampled contexts and eight queries per setting at $n = 256$. The columns $A_T/\|P_T a\|$, $\bar{A}_T$, and Att. drop are estimated from tangent probes (Appendix B.3). $A_T/\|P_T a\|$ scales the certificate by the norm of the tangent gradient, so that models with larger raw sensitivities do not appear to have a proportionally larger certificate. Att. drop is the relative fall of A_T when derivatives cannot pass through the attention scores (Section 4.3), and the brackets enclose the three per-context paired-probe 95% bootstrap intervals. $\Delta^{\perp}_{\mathrm{perm}}$ and LS are the corrected first-order statistics of Appendix A, which likewise vanish for every map that is affine on $\mathcal{M}$.

Model	Dataset	$A_T/\|P_T a\|$	$\bar{A}_T$	Att. drop [interval]	$\Delta^{\perp}_{\mathrm{perm}}$	LS
TABICL-V2	synthetic_mi	5.08	0.991	0.874 [0.82, 0.93]	1.08	0.893
TABICL-V2	kin8nm	4.40	0.993	0.814 [0.74, 0.86]	1.02	0.865
TABICL-V2	pol	35.75	0.972	0.874 [0.79, 0.92]	1.12	0.783
TABICL-V2	superconduct	10.05	0.987	0.749 [0.68, 0.82]	0.99	0.789
TABPFN-V3	synthetic_mi	8.76	0.996	0.933 [0.88, 0.97]	1.01	0.915
TABPFN-V3	kin8nm	4.82	0.993	0.850 [0.81, 0.91]	1.00	0.907
TABPFN-V3	pol	57.03	0.960	0.882 [0.84, 0.91]	1.52	0.885
TABPFN-V3	superconduct	36.32	0.996	0.880 [0.72, 0.96]	1.14	0.907
TABPFN-V2.6	synthetic_mi	3.14	0.988	0.912 [0.86, 0.95]	1.21	0.899
TABPFN-V2.6	kin8nm	2.71	0.985	0.849 [0.83, 0.87]	1.10	0.885
TABPFN-V2.6	pol	16.30	0.949	0.964 [0.95, 0.97]	2.52	0.922
TABPFN-V2.6	superconduct	7.51	0.989	0.932 [0.91, 0.95]	1.38	0.901
TABDPT	synthetic_mi	9.27	0.994	0.840 [0.75, 0.89]	1.00	0.925
TABDPT	kin8nm	6.71	0.995	0.762 [0.73, 0.81]	1.01	0.922
TABDPT	pol	31.04	0.948	0.655 [0.50, 0.74]	1.33	0.810
TABDPT	superconduct	10.32	0.991	0.568 [0.50, 0.61]	0.99	0.879

LS residuals between 0.722 and 0.925, which shows that the departure from fixed weights does not depend on the second-order instrument.

Shares of the squared curvature in every setting. The exact normalized certificate $\bar{X}_{\mathrm{sep}}$ is 0.239 to 0.916 across the 20 settings, with a mean of 0.63 (Table 8). The exact $\bar{A}_T$ ranges from 0.952 to 0.997. On the 16 settings where both instruments were run, it ranges from 0.952 to 0.996 and closely matches the cheaper probe estimate of the same quantity, 0.948 to 0.996, which checks that the probe-based instrument used in Table 7 agrees with the exact Hessian used elsewhere.

The interaction is present before the readout head. Three of the models, the two TABPFN models and TABDPT, turn the network's logits over a grid of label bins into a prediction through a softmax (the bucket-softmax readout of Table 4). Such a head is itself a nonlinear map of the logits, so it can contribute curvature even when the logits are an affine function of the labels, through the chain-rule term $J_\ell^\top(\nabla^2 h)J_\ell$, where J_ℓ is the Jacobian of the logits with respect to the labels and h is the head. Because this term would inflate the certificates for a reason unrelated to joint label processing inside the network, we recompute both certificates from a frozen linear functional of the pre-head logits, which removes the term. On the two TABPFN models, evaluated on two datasets and two contexts each, all eight context-settings keep nonzero certificates under this readout, with $\bar{X}_{\mathrm{sep}}$ between 0.50 and 0.97. Their ratio to the values obtained with the mean readout is 0.95 ± 0.13, with a minimum of 0.70, which places most of the measured interaction upstream of the bucket head on these settings.

The differentiated curvature matches the curvature of actual predictions. The certificates above come from differentiating the internal prediction at the observed labels. As a check that this differentiated curvature matches the curvature that the network actually shows along the manifold, we compare it with a purely forward quantity: the second difference of the prediction itself, evaluated at points of $\mathcal{M}$ near the observed labels. For a tangent direction v, the second derivative of a prediction function F along the renormalized label path is

$$B_F(v) = v^\top H_F v - \frac{\|v\|^2}{r^2}\,\tilde{\mathbf{y}}_c^\top \nabla F, \tag{8}$$

Table 8: **Exact certificate of joint processing, by setting.** Behind the per-model means of Table 2, each row reports $\bar{X}_{\text{sep}}$ for one setting as the mean over three sampled contexts and four queries at $n = 256$, with the range of the three context means (Range). This certificate is exactly zero for every map that is affine plus one-label-at-a-time on $\mathcal{M}$ (Section 3), and its float32 implementation floors it at 7×10^{-11}. $\bar{A}_T$ also comes from the exact Hessian, whereas Table 7 estimates it from probes. $X_{\text{sep}}/\|P_T a\|$ scales X_{sep} by the norm of the tangent gradient. The two Att. drop columns give $1 - A_T(H_{\text{det}})/A_T(H)$ and $1 - X_{\text{sep}}(H_{\text{det}})/X_{\text{sep}}(H)$, the relative falls of A_T and X_{sep} when derivatives cannot pass through the attention scores.

						Att. drop	
Model	Dataset	$\bar{X}_{\text{sep}}$	Range	$\bar{A}_T$	$X_{\text{sep}}/\|P_T a\|$	A_T	X_{sep}
TABICL-V2	synthetic_mi	0.864	[0.836, 0.900]	0.990	4.80	0.878	0.895
TABICL-V2	kin8nm	0.772	[0.700, 0.813]	0.993	4.17	0.820	0.848
TABICL-V2	pol	0.322	[0.306, 0.353]	0.977	20.7	0.885	0.838
TABICL-V2	superconduct	0.566	[0.552, 0.575]	0.990	5.94	0.681	0.700
TABPFN-V3	synthetic_mi	0.888	[0.844, 0.922]	0.996	8.49	0.935	0.968
TABPFN-V3	kin8nm	0.745	[0.721, 0.770]	0.994	5.25	0.864	0.919
TABPFN-V3	pol	0.496	[0.439, 0.530]	0.953	36.1	0.783	0.830
TABPFN-V3	superconduct	0.702	[0.662, 0.757]	0.995	30.5	0.852	0.884
TABPFN-V2.6	synthetic_mi	0.837	[0.772, 0.896]	0.981	2.82	0.891	0.952
TABPFN-V2.6	kin8nm	0.774	[0.741, 0.818]	0.982	2.36	0.857	0.931
TABPFN-V2.6	pol	0.567	[0.547, 0.599]	0.954	12.7	0.962	0.968
TABPFN-V2.6	superconduct	0.758	[0.739, 0.788]	0.990	6.48	0.926	0.961
TABFM	synthetic_mi	0.916	[0.882, 0.977]	0.995	11.4	0.921	0.941
TABFM	kin8nm	0.829	[0.817, 0.843]	0.991	5.25	0.893	0.928
TABFM	pol	0.251	[0.242, 0.260]	0.985	30.4	0.581	0.628
TABFM	superconduct	0.703	[0.672, 0.718]	0.997	8.43	0.734	0.789
TABDPT	synthetic_mi	0.447	[0.366, 0.553]	0.992	6.37	0.840	0.914
TABDPT	kin8nm	0.435	[0.417, 0.446]	0.996	4.32	0.785	0.899
TABDPT	pol	0.239	[0.174, 0.292]	0.952	16.3	0.550	0.527
TABDPT	superconduct	0.496	[0.423, 0.567]	0.992	7.37	0.546	0.673

where $\tilde{\mathbf{y}}_c$ is the centered standardized-label vector, H_F and ∇F are the Hessian and the gradient of F, and the second term removes the curvature that the path picks up from staying on the sphere, the same correction that builds T in Section 3. We compare $B_F(v)$ with central differences of the prediction along four unit tangent directions per context at the first query, for the four models other than TABFM, with steps of 0.05, 0.1, and 0.2. The full internal readout reaches a median setting-level relative error of 0.046, and 15 of the 16 settings fall below the descriptive threshold of 0.5, with only TABDPT on pol above it at 0.567.

Context-size dependence. Because the certificates in Table 2 use $n = 256$ context rows, we also check whether the interaction strengthens or weakens with more context. For this check we use the ratio $R_T = \|P_T H P_T\|_F / \|P_T a\|$. Unlike A_T, this ratio keeps the isotropic part of the tangent curvature and simply scales its norm by the tangent gradient norm. For TABPFN-V3, TABPFN-V2.6 and TABDPT on synthetic_mi and kin8nm, R_T varies by at most a factor of 2.4 and shows no consistent trend from 128 to 1024 rows, so the same ratio divided by $\sqrt{n}$ declines, and this descriptor does not show a strengthening of the curvature with context size. Two of the largest-context TABPFN-V2.6 settings exceeded available memory. For TABICL-V2 and TABPFN-V2.6 on the same two datasets, the exact normalized certificate $\bar{X}_{\text{sep}}$ ranges over the context means, with two contexts and two queries at each size, from 0.80 to 0.96 at $n = 128$, 0.72 to 0.91 at $n = 256$, and 0.67 to 0.91 at $n = 512$, a mild decline that leaves the certificate present at every tested size.

C.2 THE DEFAULT PREDICTIONS ARE NOT AFFINE IN THE LABELS

The two certificates of Section 3 are computed on a single internal estimator inside each package, since this is the quantity for which the network's own derivatives are well defined (Section 4.1). By contrast, each package returns a complete prediction that averages several such estimators together.

This subsection checks separately whether that complete prediction is also non-affine in the labels, the check that Section 4.2 of the main text cites.

Construction. The test perturbs the labels along two orthogonal tangent bumps δ_i and δ_j, each of scale ε, chosen so that $\delta_i^\top \delta_j = 0$. Starting from the standardized labels $\tilde{\mathbf{y}}$, a step v is applied through the exactly renormalized path $m(v) = \bar{\tilde{y}}\mathbf{1} + (\tilde{\mathbf{y}}_c + v)r/\|\tilde{\mathbf{y}}_c + v\|$, which keeps the mean and the norm of the labels fixed at every step, so that $m(v)$ always lies on the standardization manifold $\mathcal{M}$ of Section 3.2. The test evaluates the four-point difference

$$D_{ij} = g(m(\delta_i + \delta_j)) - g(m(\delta_i)) - g(m(\delta_j)) + g(\tilde{\mathbf{y}}).$$

For a map that is affine on $\mathcal{M}$, the order-ε^2 term that the four-point difference could otherwise pick up from the curved path is proportional to $\delta_i^\top \delta_j$, so the chosen orthogonality sets this term to zero and the affine null becomes $O(\varepsilon^3)$. The orthogonality of δ_i and δ_j is required for this cancellation, and a one-label-at-a-time map can still produce a nonzero response under this test. The native test applies the affine image of these paths in raw-label coordinates and evaluates the complete prediction returned by each package's default inference procedure, with both the label mean and the label scale held fixed.

Results. We calibrate the test on setting-specific controls: an affine map, and an impostor that agrees with an affine map on the fixed-moment set but not outside it, in the manner of g_a. A third control, a map that lets the labels interact, checks that the test fires. With the member seeds held fixed, the measured interaction exceeds the floor set by the first two controls by a factor of 44 to $16{,}900$ on 12 context-settings, spanning three models, two datasets, and two contexts. We had required in advance that this floor stay below 5% of the interaction of the third control. The two `kin8nm` contexts miss this requirement, with floors at 6 to 9%, and they give the smallest factors, 44 to 279, which still place the measured interactions far above the floor. Averaged over the two contexts of each model and dataset, the ratio between the complete and the single-estimator response is 0.41 to 1.03 (0.30 to 1.37 for single contexts). TABDPT additionally averages over several feature orderings, and this separate averaging axis exceeds its calibrated floor by a factor of 55 to 616 on four context-settings, with ratios of 0.54 to 1.54 between the averaged and the single response.

Scope. These functional checks support the finding that the default predictions are not affine in the labels. They do not exclude a complete prediction that transforms each label one at a time and combines the results without interaction, since such a map is also non-affine. The exact certificate X_{sep} therefore remains scoped to the internal, single-estimator paths measured in Section 4.2.

C.3 RANDOM COPIES OF THE TRAINED MODELS

Section 4.3 of the main text asks whether the interaction between labels comes with the architecture itself or appears only after training, and reports that the normalized certificate $\bar{X}_{\mathrm{sep}}$ separates trained networks from untrained ones while A_T does not (Section 4.3). This subsection gives the construction and the full range of values behind that comparison, together with a further control that matches the entropy of the attention scores between trained and random networks.

Random copies. For each of the five models, we build 12 random copies and evaluate each copy on the same two datasets, `synthetic_mi` and `kin8nm`. Each copy is built by calling the reset method of every module of the network, and we draw every parameter that these methods do not reach from a Gaussian with standard deviation 0.02. For TABFM, we also redraw from a standard normal the learned Fourier frequencies that its released code stores outside the parameter list. The copy is restored only after the standardization wrapper has been fit to the context, so that only the network's own parameters differ from the trained model.

Results. For the four models other than TABFM, the trained values are the means over the four queries of the first context of the grid behind Table 8, and each random copy is evaluated on the same context rows and labels with the first two of these queries. On `kin8nm`, the feature scaling of the two inputs also differs slightly, because it is fit on the context and query rows together. For TABFM, the trained values come instead from the first context and the first two queries of that grid, which match the shared context exactly on `synthetic_mi` and differ from it on `kin8nm` only

by a feature rescaling of at most 0.02. This comparison separates the training-associated share of the curvature from the generic curvature that any nonlinear network carries even before training. Every query of a trained model gives $\bar{X}_{\mathrm{sep}} \geq 29.5\%$, with means of 44% to 98%, whereas the random copies stay at most 1.7% as a two-query mean and at most 3.4% for a single query. The gradient-scaled certificate $X_{\mathrm{sep}}/\|P_T a\|$ shows the same separation: 2.49 to 21.7 for the trained models against a maximum of 0.037 to 0.32 across the random copies. Random copies still carry generic curvature, with a mean $A_T/\|P_T a\|$ of 2.0 to 2.3, so the corrected curvature norm A_T is nonzero for trained and random networks alike and, as a yes-or-no certificate, cannot distinguish them. For the four models other than TabFM, the naive permutation statistic Δ_{perm} of the trained models falls within the range of 12 separately drawn random copies on all eight settings (at $n = 128$ for all models but TabICL-v2), so this simpler statistic cannot separate trained from random either, in line with the same limit noted for A_T in the main text.

Attention-entropy control. A further control asks whether matching the entropy of the attention scores, rather than leaving random copies at their default entropy, closes the gap. It covers three random copies per architecture on two architectures (TabICL-v2 and TabPFN-v2.6) and two datasets, for 12 evaluations. Matching the attention entropy between the random copies and the trained network raises $\bar{X}_{\mathrm{sep}}$ only to 0.16 for TabICL-v2 and to 0.004 for TabPFN-v2.6, and raises the gradient-scaled certificate to at most 1.33, far short of the trained values above. The TabDPT entropy comparison is descriptive rather than a matched control, because scaling its logits did not match its entropy to the trained network's. Taken together, these controls support a training-associated change in how labels interact, without isolating that change as the only one that training introduces. Appendix C.5 follows the label response of small transformers through training, in models whose architecture we control.

C.4 Paths through the attention scores

Attention combines the representations with weights that a softmax computes from the attention scores. Section 4.3 shows that blocking derivatives through them removes most of the measured interaction between labels while leaving the prediction unchanged (Figure 1b), and Table 2 reports the resulting fall in A_T and X_{sep} for the exact Hessian. This subsection reports that comparison in full, gives a direct comparison between the intervened and the full curvature operator, states the scope of this derivative-path intervention, and then reports a second intervention that changes the forward computation itself.

We apply the intervention at every attention step the network executes: the attention scores keep their forward values unchanged, so the prediction does not change, while derivatives with respect to the labels are blocked from passing through them. Writing the resulting Hessian as H_{det}, the difference $H - H_{\mathrm{det}}$ isolates the curvature carried by the blocked derivative paths. For TabICL-v2, one label vector feeds both the column-wise set transformer and the label encoder of the in-context stage (Appendix B.1), and a replay of the full pipeline lets the derivatives pass through both sites, with forward-agreement checks confirming that the replay leaves the prediction unchanged. TabPFN-v3 and TabFM likewise receive one label vector at both of their label sites, and we differentiate their full forward pass, so their derivatives also pass through both sites.

A direct comparison of the full and the intervened curvature operator supports the same reading. The part of the curvature attributable to the blocked derivative paths, $\|P_T(H - H_{\mathrm{det}})P_T\|_F/\|P_T H P_T\|_F$, reaches 0.96, 0.99, 1.00, 0.94, and 0.93 for TabICL-v2, TabPFN-v3, TabPFN-v2.6, TabFM, and TabDPT, in that order, while the part that survives the intervention, $\|P_T H_{\mathrm{det}} P_T\|_F/\|P_T H P_T\|_F$, is 0.19, 0.14, 0.09, 0.22, and 0.32. Averaged over the twelve Hessians of each setting, the cosine between the two parts lies between -0.12 and 0.28, consistent with a small remainder and limited cancellation between them.

This decomposition is relative to the evaluated computational graph, and an algebraically equivalent implementation could redistribute the derivative paths between the two terms. The intervention also only blocks derivatives, so the forward computation and the prediction stay unchanged. It therefore does not by itself validate a model in which the attention scores are frozen at their base-label values and never recomputed as the labels change. Such a forward intervention would cache every attention-score tensor at the base labels and reuse it on every perturbed forward pass, and measuring its own curvature would require that frozen map's own gradient in the forward second-difference

Table 9: **Attention-path norm reductions.** Every entry is the relative fall of a norm under the intervention described in the text, not a share of squared curvature and not a loss of accuracy. Values average over four datasets, with three contexts per dataset and four queries per context (eight for the probe estimate of A_T) at $n = 256$ context rows. The probe columns estimate the tangent curvature norm $\|P_T H P_T\|_F$ from 16 shared probes and A_T from 64 probes, and the exact columns use Hessians assembled column by column on all 20 settings. We measured TABFM with the exact Hessian only, so the probe means cover the other four models.

	Probe estimate		Exact Hessian	
Model	$\|P_T H P_T\|_F$	A_T	A_T	X_{sep}
TABICL-V2	82.1%	82.8%	81.6%	82.0%
TABPFN-V3	87.7%	88.6%	85.9%	90.0%
TABPFN-V2.6	91.5%	91.4%	90.9%	95.3%
TABFM	–	–	78.2%	82.1%
TABDPT	69.6%	70.6%	68.0%	75.3%
Mean	82.7%	83.4%	80.9%	85.0%

formula of Eq. (8), together with a comparison against that map's own prediction differences. Recomputing the attention scores with a stop-gradient supplies neither. The ambient quotient kept in the supplementary records omits these gradient terms as well, so it does not identify a signed share of the intrinsic curvature either. The norm reductions reported above therefore keep their scope as comparisons between derivative paths in the computational graph as evaluated.

Replacing the cached attention maps. We report a second intervention that does change the forward computation, as a stress test of how much the attention scores contribute to the prediction itself, rather than only to its derivatives. We run two forward passes. The first pass computes the network on one set of labels and caches the attention scores of every attention step in the order the network computes them. The second pass recomputes the network's hidden states from a second, possibly different set of labels, but at every attention step it replaces its own attention scores with the corresponding cached ones, in the same order.

When both passes use the same labels, this hybrid computation reproduces the native internal forward pass exactly on all 16 settings of the four models other than TABFM, which this intervention does not cover. Replacing the cached scores by those computed from a zero label vector instead shifts the prediction by 0.60 to 0.97 standard deviations of the model's own native predictions, averaged within each model. Because the hybrid computation applies later-layer cached scores to hidden states that differ from the ones on which those scores were computed once the two passes receive different labels, the exact agreement obtained under identical labels validates only that the replay reproduces the native forward pass correctly. It does not by itself show that varying the cached scores isolates a causal effect of the attention scores.

An affine map fitted from the replayed internal outputs to the package's own predictions in the original label units reproduces these predictions to within 0.96% relative error. Replacing the cached scores worsens the normalized mean squared error (NMSE, the mean squared error divided by the variance of the query targets) on 13 of the 16 settings and improves it on the remaining three, with a median ratio of 5.12 between the patched and the native error. Interpolating between the true-label scores and the zero-label scores increases the displacement from the native prediction monotonically on 15 of the 16 settings. Replacing the cached scores at a single attention call can produce a larger displacement than replacing them at every call, which shows cancellation across calls and excludes an explanation in which each call's attention scores contribute additively to the total displacement. Per-setting results are in the supplementary material.

C.5 CONTROLLED MODELS

The pretrained models evaluated above cannot be retrained under a different wiring, so answering which components the measured interaction needs, and whether attention scores that depend on the labels are needed at all for a model to use labels well, requires models whose architecture we control

ourselves. This subsection trains small transformers in which we vary how labels enter the network, whether LayerNorm is present, and whether the attention scores are allowed to depend on the labels.

A controlled transformer on a Gaussian-process task. We train a three-layer, 64-dimensional row-token encoder for 4000 Adam steps on a two-dimensional radial-basis-function Gaussian-process prior, under either Gaussian observation noise or 15% outlier contamination. Across three seeds, the held-out loss falls from about 1.3 to 0.02 under Gaussian noise and 0.03 under contamination within 1000 steps. These models receive the raw labels without standardization, so every label vector is a valid input and the ambient derivatives are determined by the predictions, which makes the correction of Section 3.2 unnecessary here. We track two descriptors across training, computed directly on these derivatives: the permutation response Δ_{perm} (Eq. 7), which measures how much the sensitivity vector changes under a permutation of the context labels, and the off-diagonal ratio $\|H - \mathrm{diag}(H)\|_F / \|H\|_F$ of the ambient Hessian. The permutation response grows from $\Delta_{\mathrm{perm}} \approx 0.35$ at initialization to 0.85 ± 0.50 under Gaussian noise and 1.38 ± 0.29 under contamination, and the final off-diagonal ratio reaches 0.44 to 0.50. Because the Gaussian task fixes the kernel and the noise level, its Bayes-optimal predictor, the Gaussian-process posterior mean, is exactly linear in the labels, so this learned nonlinearity alone does not establish that it is useful, and the higher response under contamination is suggestive but not conclusive, given the spread across seeds. This small controlled model supports no claim about scale.

LayerNorm is not necessary. Because LayerNorm's normalizing statistics are themselves computed from representations that already mix the labels, we ask whether blocking derivatives through the attention scores is confounded with blocking derivatives through LayerNorm, or whether the two make separate contributions. In a transformer of the same size trained on the contaminated task, blocking derivatives through the attention scores removes $92 \pm 2\%$ of the ambient off-diagonal norm across three seeds, and blocking derivatives through both the attention scores and the LayerNorm statistics removes 98%. Blocking the LayerNorm statistics alone removes 85%, because normalization follows label mixing, so the two interventions overlap rather than acting on separate parts of the interaction. Yet a model trained with no LayerNorm at all reaches a comparable loss (0.028 against 0.026 for the model with LayerNorm), keeps an off-diagonal ratio of 0.41, and still loses 97% of its off-diagonal norm when derivatives cannot pass through the attention scores. LayerNorm is therefore not necessary for the interaction, since removing it leaves both the interaction and its concentration in the attention scores in place.

Useful label processing without label-dependent attention scores. We next ask whether a model can process labels nonlinearly, and use that nonlinearity effectively under contamination, without ever letting the attention scores depend on the labels. A three-seed comparison matches depth, width, number of heads, task, and training budget across four ways of wiring labels into the network: injecting them into the same tokens from which the attention scores are computed, as in four of the five pretrained models above (token injection), injecting them only into the values that attention combines, as in TABDPT (value-only injection), computing the attention scores from the features alone while a separate nonlinear stream processes the labels, so that the labels cannot influence the attention scores at all (the score-blocked model), and a fully label-linear control.

Under symmetric contamination, the score-blocked model reaches MSE 0.0215 ± 0.0008, within 9% of token injection's 0.0198 ± 0.0009, while the label-linear control reaches 0.0388 ± 0.0003, 96% worse than token injection. Clean-task MSE ranges from 0.0145 for token injection to 0.0214 for value-only injection. In the score-blocked model, a smaller share of the curvature comes from interaction that no one-label-at-a-time map can produce, $\bar{X}_{\mathrm{sep}} = 0.085 \pm 0.145$ against 0.243 ± 0.080 for token injection.

A five-seed extension of this comparison covers 75 runs with saved training curves. Under symmetric contamination, the score-blocked and label-linear losses are respectively 5% and 91% above token injection. Because the initial score-blocked model has more parameters than token injection (175,937 against 100,865), we also train a slim version with 107,993 parameters, 7% more than token injection's count, which reaches MSE 0.0243 ± 0.0016: within 20% of token injection and 1.6 times better than the label-linear control. One-sided outlier contamination reproduces the same ordering, with the score-blocked model 8% above token injection against 142% for the label-linear control. The share of the curvature norm that passes through the attention scores also changes with

how labels enter the network, but its direction reverses across contamination regimes: the share for value-only injection is below token injection's under symmetric contamination (0.829 against 0.914) and above it under one-sided contamination (0.986 against 0.931). These constructed models establish a useful alternative to label-dependent attention scores, without identifying a necessary pathway in the deployed models.

C.6 How close a fixed-weight map comes

Section 4.2 shows that no fixed-weight map reproduces any of the five models exactly, since $\bar{A}_T$ is nonzero in every setting (Table 2). Section 5 nonetheless reports that a fixed-weight surrogate fitted to a model's predictions reaches a median held-out R^2 of 0.72, so failing the exact test does not mean that a model is far from such a map on the labels it meets in practice. This subsection reports the comparators behind that finding: an observed-feature comparator tuned on the context, a surrogate fitted under a task-preserving label distribution, and a constructed local-linear model with an explicit label-adaptive gain. Figure 3 relates the fidelity of a permutation-fitted surrogate to the predictive advantage of the models.

An observed-feature comparator. A first comparator tunes Gaussian and dot-product kernel ridge regression, k-nearest-neighbor regression, and local-linear regression on a held-out fifth of the context rows, and we report the best of the four on the held-out queries, a choice that favors the comparator. Each candidate map is label-linear, while the tuning itself need not be. On 28 settings, four models and seven datasets at $n = 96$ context rows, the selected comparator comes within 10% of the model's accuracy on 9 of the 28 settings, with a median held-out NMSE gap of $+0.066$. A ridge surrogate fitted directly to the model's predictions performs worse than this comparator on 27 of the 28 settings. This comparator differs from the jointly trained, label-conditioned embeddings of Miftachov et al. (2026).

Task-preserving approximation. To keep the label distribution close to the one seen in practice rather than far from it, we draw fresh observation noise for the synthetic task and bootstrap the residuals of a gradient-boosted decision-tree (GBDT) fit on the real datasets:

$$y^{(k)} = \hat{m}(X) + \mathrm{resample}\big(y - \hat{m}(X)\big),$$

where $\hat{m}$ is the GBDT fit to the data and $\mathrm{resample}(\cdot)$ redraws residuals with replacement. We fit raw-label surrogates on 210 of 300 label draws and evaluate them on the other 90, at $n = 64$ context rows and six queries per setting. Table 10 reports a median held-out R^2 of 0.723 under this task-preserving resampling, against 0.48 under label permutations, while the label-linear Nadaraya–Watson control is recovered exactly, at $R^2 = 1.0000$ on all 12 settings.

Residual resampling of this kind removes the feature-dependent structure of the noise. A separate check on `kin8nm`, `pol` and `superconduct` instead uses wild-bootstrap multipliers, Rademacher and Mammen, which rescale each row's own residual by a random factor, of magnitude one for the Rademacher multiplier. The median R^2 is 0.708 under i.i.d. residual resampling, 0.752 under the Rademacher multiplier, and 0.771 under the Mammen multiplier, with a median span of 0.097 across the three within each setting. These resampling schemes are proxies for the task's label distribution, distinct from the fixed-moment manifold $\mathcal{M}$ that the certificates of Section 3.2 use. Because permutations may lie farther from TabDPT's real-data pretraining distribution, comparing near-task fidelity across models is the safer choice here than comparing permutation fidelity. A local Taylor-remainder check failed numerical stability on 2 of the 12 settings, so it supplies no claim about the remainder curve.

A constructed local model. A third comparator asks whether a simple, interpretable label-adaptive map can reproduce the models' behavior, rather than only matching their predictions. It weights each context row by a Gaussian kernel in the features together with an exponential gain in that row's own deviation from the mean context label, before a local-linear readout that uses these weights:

$$\tilde{w}_i = \exp\left(-\frac{\|x_* - x_i\|^2}{2h^2}\right)\exp\big(\lambda(y_i - \bar{y})\big),$$

where h is the kernel bandwidth and λ sets the strength of the label-adaptive gain. Choosing (h, λ) reproduces the aggregate response profile on three synthetic tasks with 3, 8, and 16 indices at $n = 96$,

Table 10: **Held-out surrogate fidelity.** Held-out R^2, averaged over the six queries, of a surrogate that is affine in $\mathbf{y}$, fitted to each model's predictions under three label-draw distributions: R^2_{task} under the task-preserving resampling described in the text, R^2_{perm} under label permutations, and $R^2_{\text{NW-lin}}$ when the same fitting protocol is applied to a label-linear Nadaraya–Watson control instead of a model. A value of 1 means that the surrogate reproduces the predictions exactly on held-out draws, so the control column shows that the protocol recovers a genuinely affine map whenever one is present. The bottom row gives the median over the 12 settings.

Model	Dataset	R^2_{task}	R^2_{perm}	$R^2_{\text{NW-lin}}$
TabICL-v2	synthetic_mi	0.97	0.68	1.00
TabICL-v2	kin8nm	0.71	0.65	1.00
TabICL-v2	pol	0.68	0.56	1.00
TabPFN-v3	synthetic_mi	0.93	0.28	1.00
TabPFN-v3	kin8nm	0.75	0.55	1.00
TabPFN-v3	pol	0.57	0.13	1.00
TabPFN-v2.6	synthetic_mi	0.99	0.42	1.00
TabPFN-v2.6	kin8nm	0.69	0.53	1.00
TabPFN-v2.6	pol	0.74	0.28	1.00
TabDPT	synthetic_mi	0.95	0.05	1.00
TabDPT	kin8nm	0.57	0.03	1.00
TabDPT	pol	0.71	0.62	1.00
median		**0.72**	0.48	1.00

and setting $\lambda = 0$ removes the label adaptation while still allowing the local-linear readout to carry signed weights.

This construction nonetheless fits the trained models poorly. On the nine settings of TabPFN-v3, TabPFN-v2.6 and TabDPT on synthetic_mi, kin8nm and pol at $n = 128$, its median held-out R^2 is 0.02, against approximately one for maps that already belong to this class. A variant that repeats the reweighting, recomputing the gain of each row from its deviation from the previous prediction for up to eight rounds, does not fit better, since its fit selects a single round on 8 of the 9 settings.

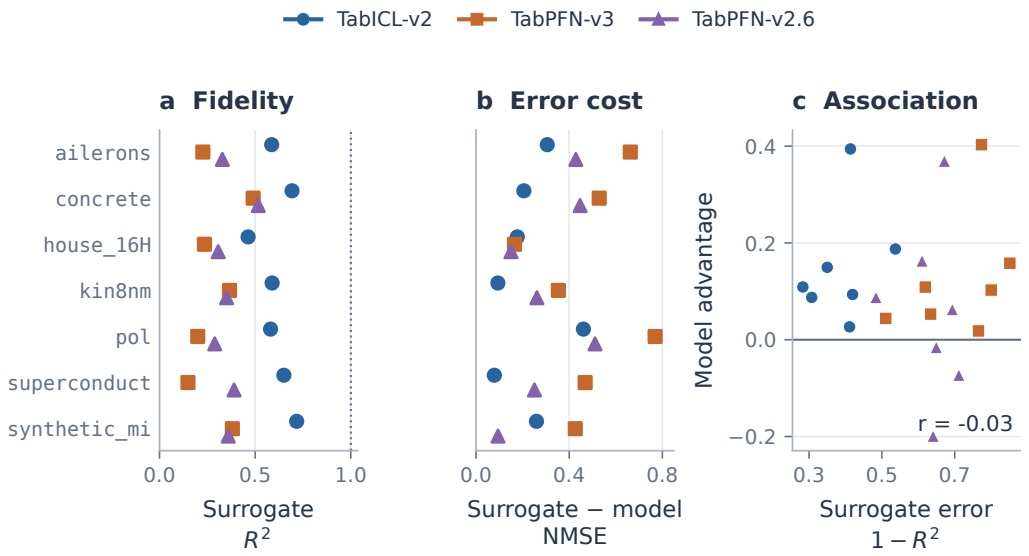

Figure 3: **Approximation fidelity and predictive cost.** The same 21 settings from three prior-fitted models show (a) the held-out permutation-surrogate R^2, (b) the increase in NMSE when the model's own predictions are replaced by this surrogate, and (c) the relation between the surrogate's error and the model's predictive advantage, which is the held-out NMSE of the best of three tuned label-linear smoothers (k-nearest neighbors, kernel ridge regression and local-linear regression) minus the model's own NMSE, so that a negative value means that a smoother predicts better than the model. Replacement increases the error on all 21 settings, but the correlation between the two magnitudes is -0.03, so this is an approximation comparison and not a causal ablation. TabDPT is not plotted here. A separate permutation fit with 200 label draws at $n = 96$ gives it an R^2 of -0.92 and -3.18 on `synthetic_mi` and `kin8nm`, against 0.05 and 0.03 under the protocol of Table 10, so its permutation fidelity depends strongly on the protocol.