MEASURING STRUCTURE IN TABULAR FOUNDATION MODEL PRIORS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tabular Foundation Models (TFMs) are trained to predict labels given in-context datasets and queries sampled from tasks (data-generating processes). These tasks are sampled from diverse but synthetic priors. The prior determines how often different tasks appear during pretraining, hence the finite computation spent training on them. However, it is unclear from the various choices in designing a prior, how much compute to allocate to each task. Motivated by the notion of epiplexity, we characterise the learnable predictive structure accessible to a model to help better allocate compute across tasks. To do this, we introduce task expected predictive information gain (TEPIG), a model-dependent measure of the expected reduction in joint predictive log loss obtained by observing training datapoints from a task. We illustrate its behaviour in controlled examples and test the hypothesis that TEPIG, when measured with cheaper proxy models, can guide the allocation of pretraining compute for TFMs. In a tractable Gaussian process setting, TEPIG-guided sampling improves predictive accuracy and convergence when evaluated under the original prior. On more complex priors, including a state-of-the-art prior used for open tabular foundation models, it allows for training in fewer steps. These findings demonstrate TEPIG's practical value for measuring model-dependent structure in tabular foundation model priors and suggest its potential as a tool for more efficient compute allocation for TFM training.

1 INTRODUCTION

Current prior-fitted Tabular Foundation Models (TFMs) are pretrained entirely on synthetic data, yet perform well on real-world benchmarks (Hollmann et al., 2023; Erickson et al., 2025). The training procedure consists of sampling a dataset and prediction queries from a data-generating process (which we refer to as a task), with the tasks themselves sampled from a prior distribution. During inference, TFMs are provided with a context dataset and prediction query, and are able to in-context predict without retraining (Hollmann et al., 2023; Garnelo et al., 2018a). The TFM predicts well if the prior over tasks captures the salient statistical structure the TFM encounters during inference (Nagler, 2023). As such, this prior is usually a complex mix of models and noise distributions, including distributions over hyperparameters (Qu et al., 2026). The downstream effects of the multiple decisions in prior construction are typically hard to ascertain. Although benchmark validation can guide these choices, thoroughly optimising numerous decisions against benchmarks can be computationally prohibitive (Qu et al., 2026).

Sampling a task more often devotes more compute for the TFM to learn the predictive distribution for that task. With finite model capacity and a limited compute budget, the training dynamics of the TFM may cause certain tasks to be learned more quickly than others. For example task A may have high noise, with the optimal prediction being to predict a dataset's mean; while task B may require the TFM to learn complicated patterns in the context dataset to predict well. Thus, spending more compute on task A, by sampling it more often, may be wasting compute that is needed to learn task B. This raises the question: can properties of synthetic tasks guide finite compute allocation, through the frequency of sampling tasks, to areas where the TFM training would most benefit?

Allocating more compute to areas where the TFM has a high predictive loss is insufficient. High loss for a task does not distinguish between information the model failed to exploit or presence of high irreducible noise (Mindermann et al., 2022). We instead focus on the expected predictive benefit a

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.

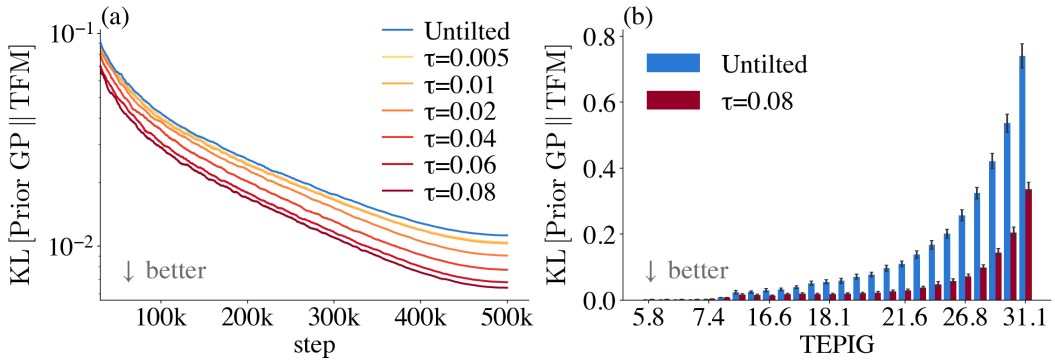


Figure 1: Illustration of TEPIG-guided prior reweighing in the case of a Gaussian process (GP) prior. We use TEPIG to tilt the training distribution (the prior), such that we train more frequently on high-TEPIG datasets. Higher τ indicates stronger tilting. (a) TEPIG-guided reweighting improves convergence and predictive performance *on the original prior* as shown by lower predictive KL to the ground-truth PPD. (b,c) The regions of the prior where the predictive KL to the ground truth PPD is highest are identified by high TEPIG. Tilting towards high TEPIG reduces this gap.

model obtains by conditioning on a task's observations. Taking inspiration from epiplexity (Finzi et al., 2026) and building on expected predictive information gain (Bickford Smith et al., 2023), we define *task expected predictive information gain* (TEPIG). For a given predictor q, TEPIG measures the expected reduction in the predictor q's log loss for joint predictions before and after observing a context dataset from that task. Low TEPIG implies q achieves little average log loss reduction by observing a context dataset from a task. This may reflect accurate context-free prediction, insufficient context information, or limitations of the predictor. In contrast, high TEPIG implies predictor q gets high average benefit from context datasets for query predictions.

In this paper, we test the hypothesis that allocating more pretraining compute to high-TEPIG tasks can improve TFM predictive performance. Importantly, we test whether we can measure TEPIG relative to proxy models that are cheaper than pretraining a large TFM. A simple Gaussian process (GP) example motivates our hypothesis. In Figure 1, we illustrate that tilting a prior towards higher TEPIG leads to faster FM training on the original, compared to training on the original prior itself. Thus, even though models with higher TEPIG distributions are trained on a different prior, under finite model capacity and training budget they perform better than the model that is trained on the correct prior distribution. These results suggest that TEPIG can guide the allocation of synthetic training tasks to better learn a prior.

For more complex priors, we develop practical methods for estimating TEPIG with proxy models and evaluate TEPIG-guided *tilting* by comparing predictive performance at matched training steps on the original (unmodified) priors and real-world datasets. We also investigate how prior design choices affect estimated TEPIG. Our experiments focus on regression, although the TEPIG definition also applies to classification. Overall, our results show that TEPIG is useful for better allocating compute for TFM training over a range of priors. Code is available at `https://anonymous.4open.science/r/TEPIG`

2 Background

The general task we focus on in this paper is to predict a target $y \in \mathcal{Y}$, given a prediction query $\boldsymbol{x} \in \mathcal{X}$ and a training dataset $\mathcal{D}_n = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$. Under a Bayesian model π, equipped with a prior $\pi(\theta)$ over data-generating processes (which we refer to as tasks) indexed by θ, and likelihood $\pi(\mathcal{D}_n \mid \theta) = \prod_{i=1}^n \pi(\boldsymbol{x}_i, y_i \mid \theta)$ the posterior predictive (PPD) is:

$$\pi(y \mid \boldsymbol{x}, \mathcal{D}_n) = \int \pi(y \mid \boldsymbol{x}, \theta)\pi(\theta \mid \mathcal{D}_n)\mathrm{d}\theta, \tag{1}$$

where $\pi(\theta \mid \mathcal{D}_n)$ is the posterior over parameters. For data sampled under π, the PPD is the optimal prediction rule under the log loss score (Madigan & Raftery, 1994). However, in general, the posterior $\pi(\theta \mid \mathcal{D}_n)$, and hence the PPD, is not analytically tractable, necessitating approximations.

2.1 Tabular Foundation Models

Rather than approximating the posterior over parameters to compute the PPD, a tabular foundation model directly learns a map from a dataset and query to a predictive distribution (Müller et al., 2022; Garnelo et al., 2018b;a)). A TFM parametrises a neural network with weights ρ resulting in a predictive distribution which we write as q_ρ (Garnelo et al., 2018b; Nguyen & Grover, 2022; Hollmann et al., 2023). To approximate the posterior predictive of the Bayesian model of interest π, the network is trained to minimise

$$\mathbb{E}_{\pi(\boldsymbol{x},\mathcal{D}_n)}[\mathbb{D}_{\mathrm{KL}}(\pi(y \mid \boldsymbol{x}, \mathcal{D}_n) \,\|\, q_\rho(y \mid \boldsymbol{x}, \mathcal{D}_n))], \tag{2}$$

which is equivalent to maximizing the expected log likelihood $\mathbb{E}_\pi[\log q_\rho(y \mid \boldsymbol{x}, \mathcal{D}_n)]$ over samples from $\pi(y, \boldsymbol{x}, \mathcal{D}_n, \theta)$. To sample training data from $\pi(\boldsymbol{x}, \mathcal{D}_n)$, first, a task index $\theta \sim \pi(\theta)$ is drawn; then, training data and query point are sampled from the task specified via θ: $y, \boldsymbol{x}, \mathcal{D}_n \sim \pi(y, \boldsymbol{x}, \mathcal{D}_n \mid \theta)$. With infinite compute and enough capacity, the neural process loss eq. (2) should allow for learning the exact posterior predictive (Bruinsma, 2022, Prop. 3.34). However, in practice, with limited compute resources for neural network training, eq. (2) is non-zero. In general, how often a task is sampled, as specified by the prior π, specifies how much of the finite compute is designated to learn the PPD of a task.

2.2 Reduction in Predictive Uncertainty

TFMs are trained to predict held-out labels given an in-context dataset and queries. To understand how much information a TFM can learn from a dataset sampled from a prior, we require a measure of predictive structure in a distribution. To do this, we measure the expected reduction in predictive entropy that conditioning on a typical dataset from the prior provides. This is defined as the *expected predictive information gain* (Bickford Smith et al., 2023) for a prior π,

$$\mathrm{EPIG} := \mathbb{E}_{\pi(\mathcal{D}_n, \boldsymbol{x}_{1:m}, y_{1:m}, \theta)}[\log \pi(y_{1:m} \mid \boldsymbol{x}_{1:m}, \mathcal{D}_n) - \log \pi(y_{1:m} \mid \boldsymbol{x}_{1:m})]. \tag{3}$$

EPIG is typically used for active learning — to select points that are most informative for a predictive distribution. Here, it motivates a measure of predictive structure that a model can extract. We provide additional background on measuring structure in cases of intractable PPD under π in Appendix A.

3 Related Work

Priors for TFMs tend to be a diverse mixture of structural causal models (SCM; Peters et al. (2017)), with mechanisms being chosen from a wide variety of models such as neural networks, trees and Gaussian processes (Qu et al., 2026; Tao et al., 2026; Eo et al., 2026). Numerous (hyper-)prior distributions determine the hyperparameters of the graphs and the mechanisms. Our work studies whether the predictive benefit of these tasks, relative to a predictor, can guide how frequently tasks are sampled during pretraining, and hence the compute available for a task.

Our measure of predictive information follows from the work on Bayesian experimental design (Rainforth et al., 2024). Lindley (1956) proposes measuring the information provided by an experiment through the expected reduction in uncertainty about a parameter. However, information about parameters need not be useful for prediction. EPIG (eq. (3)) instead directly measures the information gain about predictions (Bickford Smith et al., 2023). Whereas the aim of the above measures is to select informative points in an active learning setting, we test whether such ideas can be used for changing the compute-allocation in pretraining by changing the frequency of sampling tasks.

In practice, the predictive information a model can extract crucially depends on its specification. $\mathcal{V}$-information proposes measuring information with respect to a predictive family (Xu et al., 2020). Subsequently, works like Ethayarajh et al. (2022) use this to study dataset and instance difficulty in language modeling. Epiplexity takes a coding perspective, measuring extractable structure through the description length of the model that minimises a two-part code under computational or parametric constraints (Finzi et al., 2026). We follow the above and define our measure of a task's predictive

benefit relative to a certain predictor. Our estimator for TEPIG that assumes predictive convergence is the same as the area-under-the-loss-curve estimator proposed for epiplexity (Finzi et al., 2026). However, we measure it for samples in a task, instead of the full learning curve for a TFM.

Our use of predictive information to guide training is related to training-data selection and curriculum learning. Mindermann et al. (2022) prioritise examples using the difference between a model's current loss and a proxy's loss. Related approaches optimise the simplex over discrete data domains for LLM pretraining. DoReMi learns mixture weights using a proxy model before training a larger model (Xie et al., 2023). Other methods adapt mixture weights during training using estimates of learning potential (Jiang et al., 2025) or epiplexity gain (Su et al., 2026). Instead of learning a simplex over discrete data domains, we propose scoring each task from which data are sampled.

4 TEPIG: Measuring predictive structure in tasks

The first aim of our paper is to identify tasks in a given prior that have learnable predictive structure, relative to a predictor. EPIG in section 2.2 assumes that the PPD induced by the prior π is computable. For complicated priors, this is not the case. Hence, we require a notion of predictive structure that is extractable by a specified predictive model. In this section we define the *task expected predictive information gain*, which is the predictive structure extractable by a predictor q, from a task indexed by θ. In the following, q can be, but does not need to be a pretrained TFM.

Definition of TEPIG: Given a task θ[1] from which datasets of size n and m query points can be sampled, $\mathcal{D}_n, \boldsymbol{x}_{1:m}, y_{1:m} \sim \pi(\mathcal{D}_n, \boldsymbol{x}_{1:m}, y_{1:m} \mid \theta)$, we define the *task expected predictive information gain* relative to a predictor q as:

$$\mathrm{TEPIG}(q, \theta, n, m) := \mathbb{E}_{\pi(\mathcal{D}_n, \boldsymbol{x}_{1:m}, y_{1:m} \mid \theta)}[\log q(y_{1:m} \mid \boldsymbol{x}_{1:m}, \mathcal{D}_n) - \log q(y_{1:m} \mid \boldsymbol{x}_{1:m})]. \quad (4)$$

Here, the first term measures the log probability of *jointly* predicting m datapoints $y_{1:m}$ given $\boldsymbol{x}_{1:m}$ with model q, after conditioning q on $\mathcal{D}_n$ (either in-context or by training). The second term measures the same probability of jointly predicting the same m datapoints, but when q has not been trained on any data, or no data is provided in-context.

Collectively, the difference of the two terms measures the expected reduction in the joint predictive log loss on m queries for q, before and after seeing n datapoints from the task. As we consider this reduction relative to a predictive model q, we illustrate how TEPIG behaves for a simple sinusoidal function and different predictors q in Figure 2. Analogous illustrations for two additional functions are in Appendix C. We note that TEPIG is well-defined for both regression and classification cases as both can be formulated in the likelihood framework.

Choice of m. For the rest of the work, for simplicity of exposition and implementation, we assume $m = n$ and write $\mathrm{TEPIG}(q, \theta, n)$. While n is a parameter of TEPIG, we note that setting $m = 1$ corresponds to a reduction in single point's log loss, and thus only takes the end points of the learning curve over $m + n$ points. Setting $m > 1$ is desirable as it takes the full learning trajectory into account. Appendix B provides a concrete example in which a linear and a sinusoidal task receive indistinguishable single point TEPIG scores, while the sinusoidal task receives a higher joint TEPIG score due to slower learning.

TEPIG and prior-posterior KL While TEPIG is fully justified from the perspective of predictive joint information gain, we can directly relate it to the notion of information gain with respect to a posterior over parameters for Bayesian models q (Lindley, 1956) in Appendix E.

4.1 Estimating TEPIG

If a predictor q allows for joint predictions at multiple test-feature-locations $\boldsymbol{x}_{1:m}$ at once, TEPIG in eq. (4) can be estimated directly. However, most tabular machine learning methods and TFMs only allow for marginal predictions at different test-locations $\boldsymbol{x}_i$, i.e. a separate predictive distribution for each test target conditional on the same dataset (Hollmann et al., 2023; Qu et al.,

[1]This θ can but does not need to be drawn from a prior $\theta \sim \pi(\theta)$. The only important thing is that we can sample training-and-test-points repeatedly according to $\pi(\cdot \mid \theta)$

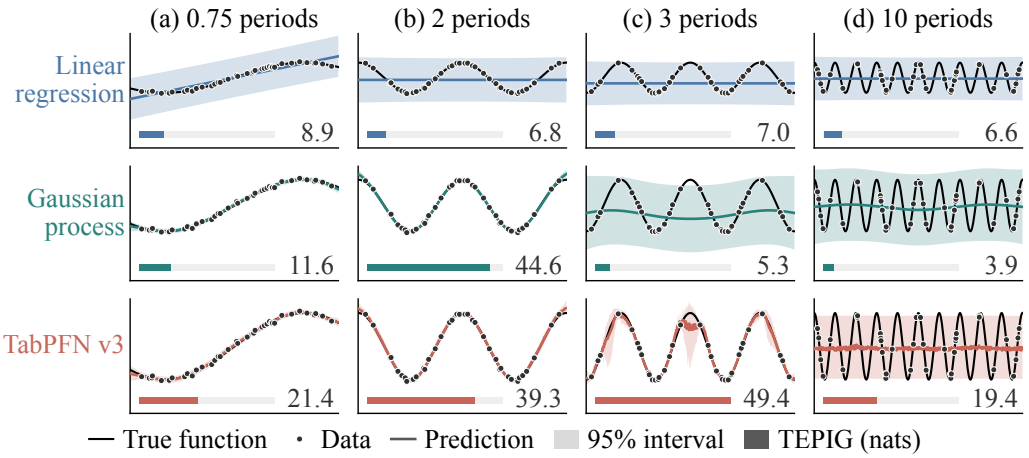


Figure 2: TEPIG captures learnable predictive structure for a given model. We illustrate this using sinusoidal tasks of increasing frequency. In column (a), linear regression, a Gaussian process (GP), and TabPFNv3 (Grinsztajn et al., 2026) all fit the 0.75-period sinusoid well. In column (b), the linear model cannot capture the oscillations and thus attains a lower TEPIG score. Whereas the GP and TabPFNv3 still provide good fits and obtain higher TEPIG scores than in (a). In column (c), the GP also fails to capture the oscillations and receives a lower TEPIG score, while TabPFNv3 achieves its highest TEPIG among the four tasks. In column (d), none of the models captures the oscillations, and all three receive comparatively low TEPIG scores.

2026)[2]. For such predictors, we can construct the joint required in TEPIG by autoregressively adding observed query and labels back into the context (Dawid & Vovk, 1999). Let $\ell_k(q, \theta) := \mathbb{E}_{\pi(\mathcal{D}_k, \boldsymbol{x}_{k+1}, y_{k+1} \mid \theta)}[-\log q(y_{k+1} \mid \boldsymbol{x}_{k+1}, \mathcal{D}_k)]$ denote the expected held-out negative log-likelihood loss of q after conditioning on k datapoints $\mathcal{D}_k$, with $\mathcal{D}_0 = \emptyset$. We can write eq. (4) as

$$\mathrm{TEPIG}(q, \theta, n) = \sum_{k=0}^{n-1} \left[\ell_k(q, \theta) - \ell_{k+n}(q, \theta)\right]. \tag{5}$$

Hence, in practice the estimation of the joint in TEPIG only requires evaluating the one-step held-out loss $-\log q(y_{k+1} \mid \boldsymbol{x}_{k+1}, \mathcal{D}_k)$ for context sizes $k = 0, \ldots, 2n-1$, rather than evaluating the joint predictive density directly. The expectation in ℓ_k can be estimated using Monte Carlo sampling from the task-specific data-generating process $\pi(\cdot \mid \theta)$.

Uncomputable $q(y_{1:m} \mid \boldsymbol{x}_{1:m})$. For many models q, it is not possible to directly compute the term $\log q(y_{1:m} \mid \boldsymbol{x}_{1:m})$ or the term ℓ_0, as they do not allow for computing log probabilities without conditioning (in-context or by training) on data. For these, we condition on a small dataset $\mathcal{D}_c$ of c samples from the task and drop the first c terms of the prequential sum in eq. (5), i.e. we replace $\mathbb{E}[-\log q(y_{1:m} \mid \boldsymbol{x}_{1:m})]$ with $\mathbb{E}[-\log q(y_{1:m} \mid \boldsymbol{x}_{1:m}, \mathcal{D}_c)]$, or equivalently start the sum in eq. (5) with $k = c$. In general, we note that keeping c small provides a closer approximation to eq. (4), and hence the predictive benefit that n context samples provide.

Faster prequential estimation by assuming predictive convergence The estimation of eq. (5) requires fitting $2n$ models or autoregressive generation steps, depending on q. To increase the speed of estimation, we make the assumption that the predictive loss has approximately converged by context size n. This allows us to use $\sum_{k=n}^{2n-1} \ell_k(q, \theta) \approx n\, \ell_n(q, \theta)$, where $\ell_n(q, \theta)$ is the expected test loss of q conditioned on $\mathcal{D}_n$. Thus, we replace the second loss sum by n times the test loss at context size n, yielding

$$\mathrm{TEPIG}_C(q, \theta, n) = \sum_{k=0}^{n-1} \ell_k(q, \theta) - n\, \ell_n(q, \theta). \tag{6}$$

[2]We note the architecture in Mortimer et al. (2026) as an exception, which allows for one forward pass estimation of TEPIG.

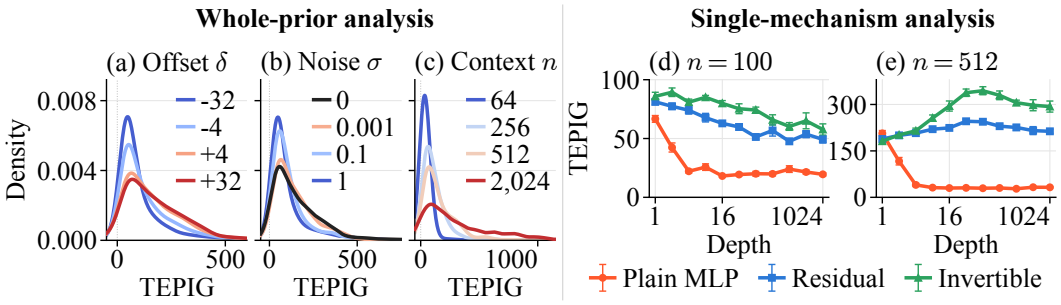


Figure 3: Investigating priors and mechanisms with TEPIG. All TEPIG estimates use TabICL v2 as the proxy model. **Left:** Distributions of estimated TEPIG under changes to (a) the graph connectivity offset δ, (b) the standard deviation σ of internal Gaussian noise, and (c) the context size n. **Right:** TEPIG versus depth for three randomly initialized mechanism architectures with $n = 100$ (d) and $n = 512$ (e) context points. Depth counts hidden layers for plain MLPs and residual blocks for residual architectures.

As illustrated in Figure 10, TEPIG_C is the area between the initial learning curve and the constant baseline $\ell_n(q, \theta)$. Throughout the paper, we use TEPIG_C instead of TEPIG and empirically validate the approximation in Equation 6 in Appendix D.4. We provide exact details on how to estimate TEPIG efficiently for different models in Appendix D, where we also explain that the assumption of predictive convergence especially speeds up TEPIG computation for TFMs.

5 Investigating Priors with TEPIG

Now we estimate TEPIG using the TabICL v2 model on its own prior (Qu et al., 2026), to understand how components of the prior affect the learnable predictive structure. Details are in Appendix F.

Increasing the overall connectedness of the sample graphs shifts the distribution towards higher TEPIG (Figure 3a). Conversely, increasing the variance of Gaussian noise at each node concentrates more probability at lower TEPIG (Figure 3b). Finally, increasing the context size from 64 to 2024 observations broadens the distribution (of the otherwise unchanged prior) towards higher TEPIG (Figure 3c), with an increasingly heavy right tail indicating greater diversity in terms of learnable structure in larger datasets. On the mechanism level, we compare plain MLPs, residual MLPs, and invertible ResNets (Behrmann et al., 2019; Marszałek et al., 2025) with depths $1, 2, 4, \ldots, 1024$ for $n \in \{100, 512\}$ in-context observations. Figure 3(d,e) shows that greater depth does not always mean more *accessible* structure: At $n = 100$, TEPIG generally decreases with depth across architectures, while at $n = 512$, it drops sharply by depth four for plain MLPs, remains stable for residual MLPs, and peaks around 64 blocks for the invertible ResNet. This is consistent with signal decay in deep plain MLPs (Schoenholz et al., 2017), which skip connections can mitigate (He et al., 2016), while invertible ResNets prevent information loss in the absence of noise (Behrmann et al., 2019).

6 Experiments: Tilting TFM priors towards higher TEPIG

To evaluate TEPIG, we investigate the impact on TFM pretraining of allocating more compute to high TEPIG tasks and lower compute for low TEPIG tasks, by changing the frequency of tasks based on TEPIG. We begin with Gaussian process priors, then consider the SCM prior proposed by Reuter et al. (2026), and finally examine the state-of-the-art TabICL v2 prior (Qu et al., 2026), albeit under a reduced training budget. We focus on regression priors throughout.

Tilting the pretraining prior. Across these experiments, we use TEPIG to reweight the training prior $\pi(\theta)$. This produces a tilted prior $\tilde{\pi}_\tau(\theta)$, with τ denoting the tilting strength parameter, that samples tasks with higher estimated TEPIG more frequently, and hence devotes more compute to higher TEPIG tasks. To accomplish this, we use exponential tilting towards higher TEPIG implemented using rejection sampling. Given a prior, τ is the only hyperparameter to be chosen for

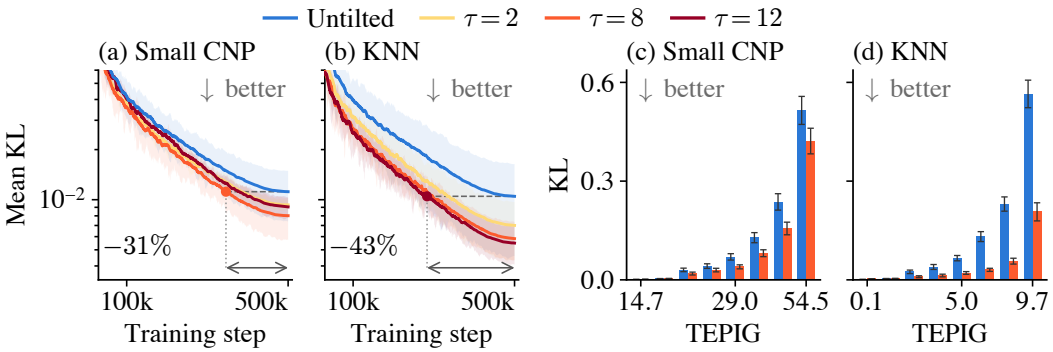


Figure 4: TEPIG-based tilting of the GP prior with the small CNP and KNN proxies (for the GP proxy, see Figure 1). (a,b) Mean predictive KL on the original prior during training; markers show when the best tested tilt reaches the untilted model's final KL. (c,d) Predictive KL estimates for untilted training and $\tau = 8$. Equally spaced bar groups represent length-scale bins, ordered and labelled by their mean TEPIG under the corresponding proxy. Tilting yields the largest absolute KL reductions in high-TEPIG groups. Shading and error bars show 95% confidence intervals across training runs and sampled tasks, respectively.

compute allocation — a higher value denoting a general shift towards high TEPIG tasks[3]. In general, we choose the strongest τ such that the acceptance rate in our rejection sampling is above 10%. (Details in Appendix G). We fix the number of context-points for each training run.

Proxy models for estimating TEPIG. We empirically test various proxy models q for computing TEPIG. In particular, whether q can be cheaper than the final TFM pretrained on the prior π. We consider existing pretrained TFMs, as well as a cheap K-nearest-neighbour (KNN) regressor. Appendix D describes how we compute scores with each model.

Evaluation. For all synthetic experiments, we evaluate on the original prior $\pi(\theta)$. We note that the tilted prior $\tilde{\pi}_\tau(\theta)$ induces a different posterior predictive than the one induced by $\pi(\theta)$. However, with limited compute budget, our hypothesis is that training on $\tilde{\pi}_\tau(\theta)$ should nevertheless improve performance compared to training on $\pi(\theta)$. We examine this in two ways. First, we compare the number of training steps needed to reach a given predictive error and the overall predictive performance after a fixed number of steps. Second, we examine where the gains occur. As higher TEPIG tasks are where a context dataset improves predictions the most, we expect improvements to be highest on high TEPIG tasks.

6.1 Gaussian processes as a tractable prior

We first consider a simple prior where the posterior predictive (PPD) is tractable. Our prior is a squared-exponential GP prior with a discrete uniform prior over lengthscales. We train a simple conditional neural process (CNP; Garnelo et al., 2018a) on the original prior and priors tilted towards higher estimated TEPIG (with increasing τ). We compare three proxy models used to compute TEPIG: the prior GP where we numerically marginalise over length scales, a smaller pretrained CNP on the original prior, and a KNN regressor. Evaluation is done on the original prior, using the KL divergence between the GP prior's PPD and the TFM's PPD. Further details are provided in Appendix H.

Tilting leads to faster training. For all three proxy models, tilted training reaches the baseline's final predictive KL in fewer updates: approximately 41% fewer training steps for the GP proxy (Figure 1a), and 31% and 43% fewer for the small CNP and KNN proxies (Figure 4a,b).

Improvements are larger at higher TEPIG. For both the small CNP and KNN proxies, higher estimated TEPIG corresponds to larger predictive KL under untilted training and show larger absolute reductions under tilting (Figure 4c,d). The GP proxy shows the same pattern (Figure 1). At low

[3]We note in Appendix G that higher τ does not monotonically lead to higher TEPIG tilting.

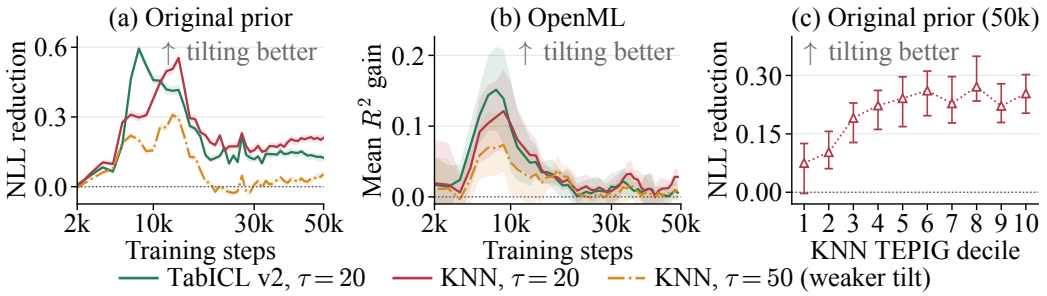


Figure 5: Improvements over untilted training: (a) median-NLL reduction on the original prior; (b) mean R^2 gain on OpenML; (c) median-NLL reduction for KNN tilting ($\tau = 20$) at 50,000 steps on fixed evaluation tasks from the original SCM prior, stratified by KNN TEPIG decile (1: lowest; 999–1,000 tasks per decile). Positive values favour tilting. Shading and error bars show pointwise 95% task-bootstrap intervals. KNN with $\tau = 50$ shifts the training distribution less than with $\tau = 20$ (section G).

TEPIG, changes are small and can include slight increases in KL. For KNN tilting at $\tau = 8$, predictive KL decreases by 0.36 nats in the group with the highest mean TEPIG (≈ 9.7), while increasing by $2.1 \cdot 10^{-3}$ nats in the group with the lowest mean TEPIG (≈ 0.1; Figure 4d). Results stratified by length scale are provided in Appendix Figures 12, 13, and 14.

6.2 SCM PRIOR

We next consider the SCM regression prior of Reuter et al. (2026). We compare original prior training with tilting towards higher estimated TEPIG, using TabICLv2 or KNN as proxy models. Models are compared at matched training steps on the original prior and benchmark OpenML regression datasets. Experimental details are provided in Appendix I.

Tilting improves performance at a fixed training budget. All tested configurations improve early performance on both the original prior and OpenML, with benefits generally narrowing during training (Figure 5 a,b). At 50,000 steps, tilting with $\tau = 20$ reduces median NLL on the original prior by approximately 0.12 with TabICL v2 and 0.21 with KNN. On OpenML, KNN tilting at $\tau = 20$ retains a mean R^2 gain of approximately 0.028 at the last shared checkpoint. The learning curves in Appendix I.5 show that this happens because the baseline nearly catches up to the tilted model, leading to a smaller benefit at the end of training.

Larger benefits on tasks with higher estimated TEPIG. For KNN tilting ($\tau = 20$), median-NLL reductions over the original prior baseline generally increase as TEPIG increases and notably without any loss in performance on low TEPIG regions (Figure 5 c).

Additional results. On this prior, we get gains in performance for tilting towards high TEPIG. We test the converse and tilt the prior towards lower TEPIG with the TabICL v2 proxy devoting more compute to low TEPIG tasks. This substantially worsens performance (Appendix Figure 15).

6.3 TABICLV2 PRIOR

Finally, we study the TabICLv2 (Qu et al., 2026) regression prior. To do so, we pretrain a smaller version of TabICLv2. We compare training on the original prior with tilting towards both higher and lower estimated TEPIG. As it performed the best in the previous experiments (and is the cheapest), we employ the KNN regressor as our proxy model. We choose $\tau = 2.5$ for high-TEPIG and low-TEPIG tilting. Real-world results on TabArena (Erickson et al., 2025) as well as architectural details we defer to Appendix J.

Tilting improves performance at a fixed training budget. In line with our findings on the SCM prior, training with high-TEPIG tilting improves early over the original model. Evaluating the pinball loss on datasets sampled from the original prior, we obtain the largest gains during early training

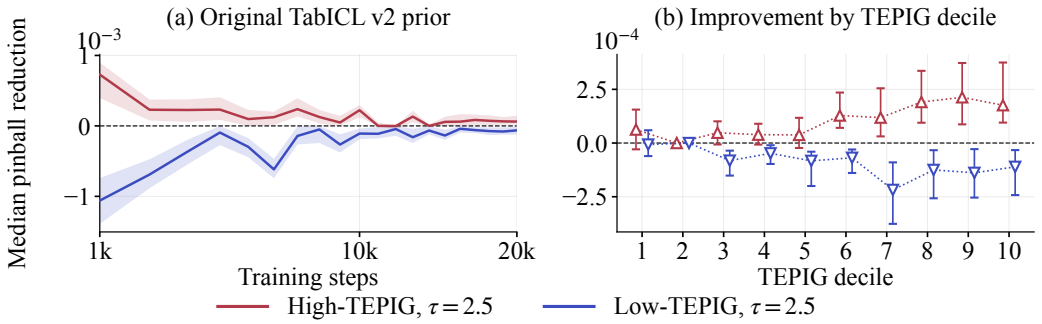


Figure 6: **TEPIG-based tilting of the TabICL v2 prior.** (a) Median reduction in pinball loss for the tilted models compared against the original. Shows that tilting towards high TEPIG results in reduction in loss throughout training, and the converse for tilting towards low TEPIG. (b) Median reduction in pinball loss for the same models at the end of training, stratified into equally sized groups by estimated TEPIG decile. Positive values favour tilting. Shows that for high TEPIG tilting, the gains are highest for higher TEPIG deciles, while not losing on low TEPIG deciles. Panel (a) uses square-root training step axis. Shading and error bars show 95% percentile intervals.

at a median pinball reduction of $7.3 \cdot 10^{-4}$ after 1,000 steps, which remains significant at most steps, and is $0.6 \cdot 10^{-4}$ at the final training step. To put these numbers into perspective, we report learning curves in Appendix J. and note that the absolute pinball loss drops from approximately 0.09 at 1,000 training steps to 0.07 at 20,000 steps. Finally, tilting towards low-TEPIG datasets during training decreases performance — the median pinball reduction being significantly lower on low-TEPIG tilting at the final checkpoint (Appendix J). On TabArena we see no significant difference in performance.

Larger benefits on tasks with higher estimated TEPIG. We confirm the above findings on stratified performance improvements. For high-TEPIG tilted models, we obtain significantly better performance in the high-TEPIG region while also *not losing on the low-TEPIG end*. In particular, high-TEPIG tilting yields median pinball improvement of $0.6 \cdot 10^{-4}$ in the first decile and an increase of $1.8 \cdot 10^{-4}$ in the top decile. Again, the low-TEPIG control has significantly worse results at high estimated TEPIG deciles. As we generally find a heavy-tailed distribution of the TabICLv2 datasets over TEPIG, we provide a more detailed analysis on high-structure datasets in Appendix J.

7 Conclusion, Limitations and Outlook

We introduced TEPIG, a predictor-dependent measure of the expected reduction in joint predictive log loss obtained from task's context, and studied its use for allocating compute across pretraining tasks by changing the frequency of sampling tasks. On the evaluated regression priors, tilting towards higher TEPIG improves the reported predictive metrics on the original prior, and leads to better performance in fewer training steps. Notably, these gains were made with TEPIG scores computed from cheaper proxies, for example the KNN regressor. These results support TEPIG as a signal for allocating a finite training budget.

We note that TEPIG depends on the predictor q used to estimate it, and can fail when q cannot capture a task's statistical structure. Although we empirically test different proxy models for scoring TEPIG, we believe testing other proxies such as smaller TFMs trained on the original prior could be a fruitful avenue. Furthermore, gains obtained for our TFMs on real-world data could be due to better learning of the prior itself, rather than the tilting. In particular, TEPIG has no notion of how suitable a prior is for real-world data, and only allows for better compute allocation to learn a prior. We thus see TEPIG as complementary to benchmark validation rather than a replacement.

Our prior analysis (section 5) further shows how prior design choices affect estimated TEPIG, hinting at the broad potential of TEPIG as a tool for designing new priors for novel tasks. The tilting parameter τ collapses the compute allocation for tasks across these complex choices to a one-dimensional parameter. Selecting τ against downstream benchmarks, rather than searching over the full set of prior hyperparameters, could be a natural next step towards more efficient prior design.

AI USE STATEMENT

In this work, we used generative AI tools to polish our writing, draft portions of the Appendix, identify relevant literature, assist with the presentation and checking of mathematical arguments, and support the implementation, execution, and monitoring of experiments. We also used these tools to obtain supplementary critical feedback on the manuscript, including whether the interpretation and presentation of experimental results were supported by the reported evidence. All AI-assisted work was reviewed by the authors.

Research ideas and writing. Generative AI was not used to originate the research ideas or construct the logical structure and core content of our scientific arguments. These contributions were developed by the authors. AI assistance with writing included suggestions for clearer phrasing, improved readability, and drafts of Appendix material. We also used AI for generating improved figures and visualisations of our results. All proposed wording incorporated into the manuscript was carefully reviewed by all authors.

Related work. We used generative AI tools to identify potentially relevant publications that we were not previously aware of or had not found through our extensive literature searches using conventional approaches. This assistance supplemented our own literature review. Publications identified with AI assistance were read and assessed as carefully as publications found through other means.

Mathematical statements and proof. AI was not used to devise our theorem, develop the central idea of its proof, or formulate the final theorem statement. AI assistance included formatting the proof, suggesting minor extensions primarily involving changes in notation, and providing supplementary checks of the mathematical reasoning. Two authors checked the proof, and AI-generated feedback served as an additional source of scrutiny. The authors reviewed the suggested changes and determined the final mathematical content and presentation.

Implementation and experiments. Coding assistants were used to help implement, run, and monitor our experiments. We carefully reviewed the AI-assisted code, tested the implementations for correctness, and checked the reported experimental results. Generative AI assistants were not used to produce synthetic data. Responsibility for the implementations, experimental procedures, and interpretation of the results remained with the authors.

We take responsibility for the final content of this work, including text, mathematical statements and proofs, experimental results, code, and other artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We release our code at `https://anonymous.4open.science/r/TEPIG`, including the exact pipelines used to obtain our experimental results and scripts to reproduce all figures. For the SCM prior of Reuter et al. (2026) and the TabICL v2 prior of Qu et al. (2026), we build on the original authors' code and follow their implementations and training procedures as closely as possible, documenting our adaptations in the Appendix. Detailed experimental setups, hyperparameters, and evaluation procedures for the GP, SCM, and TabICL v2 experiments are provided in Appendices H, I, and J, respectively. The procedures for estimating TEPIG and conducting the prior analysis are described in Appendices D and F. The full assumptions and the complete proof of our theoretical result are provided in Appendix E.

REFERENCES

Jens Behrmann, Will Grathwohl, Ricky T. Q. Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 573–582. PMLR, 2019. URL https://proceedings.mlr.press/v97/behrmann19a.html.

Freddie Bickford Smith, Andreas Kirsch, Sebastian Farquhar, Yarin Gal, Adam Foster, and Tom Rainforth. Prediction-oriented Bayesian active learning. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pp. 7331–7348. PMLR, 2023. URL https://proceedings.mlr.press/v206/bickfordsmith23a.html.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

Wessel Bruinsma. *Convolutional Conditional Neural Processes*. PhD thesis, University of Cambridge, 2022. URL https://www.repository.cam.ac.uk/handle/1810/354383.

A. Philip Dawid and Vladimir G. Vovk. Prequential probability: principles and properties. *Bernoulli*, 5(1):125–162, 1999. doi: 10.2307/3318616. URL https://doi.org/10.2307/3318616.

B. Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979. ISSN 00905364, 21688966. URL http://www.jstor.org/stable/2958830.

Moonjung Eo, Min-Kook Suh, Hye-Seung Cho, Jiwon Kim, Seoyoon Kim, Sangjun Nam, and Soonyoung Lee. EXAONE tabular 1.0: Technical report. *arXiv preprint arXiv:2608.25774*, 2026. URL https://arxiv.org/abs/2608.25774.

Nick Erickson, Lennart Purucker, Andrej Tschalzev, David Holzmüller, Prateek Desai, David Salinas, and Frank Hutter. TabArena: A living benchmark for machine learning on tabular data. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/1697e3fb412da11dc9488249f9e7bbc9-Abstract-Datasets_and_Benchmarks_Track.html.

Kawin Ethayarajh, Yejin Choi, and Swabha Swayamdipta. Understanding dataset difficulty with $\mathcal{V}$-usable information. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 5988–6008. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/ethayarajh22a.html.

Marc Finzi, Shikai Qiu, Yiding Jiang, Pavel Izmailov, J Zico Kolter, and Andrew Gordon Wilson. From entropy to epiplexity: Rethinking information for computationally bounded intelligence. *arXiv preprint arXiv:2601.03220*, 2026. URL https://arxiv.org/abs/2601.03220.

Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and S. M. Ali Eslami. Conditional neural processes. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1704–1713. PMLR, 2018a. URL https://proceedings.mlr.press/v80/garnelo18a.html.

Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J. Rezende, S. M. Ali Eslami, and Yee Whye Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018b. URL https://arxiv.org/abs/1807.01622.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Mihir Manium, Shi Bin Hoo, Magnus Bühler, Anurag Garg, et al. TabPFN-3: Technical report. *arXiv preprint arXiv:2605.13986*, 2026. URL https://arxiv.org/abs/2605.13986.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Computer Vision – ECCV 2016*, volume 9908 of *Lecture Notes in Computer Science*, pp. 630–645. Springer, 2016. doi: 10.1007/978-3-319-46493-0_38. URL https://doi.org/10.1007/978-3-319-46493-0_38.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=cp5PvcI6w8_.

Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.

Yiding Jiang, Allan Zhou, Zhili Feng, Sadhika Malladi, and J. Zico Kolter. Adaptive data optimization: Dynamic sample selection with scaling laws. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/923285deb805c3e14e1aeebc9854d644-Abstract-Conference.html.

Benjamin Jäger, Nick Erickson, Léo Grinsztajn, Felix Birkel, Klemens Flöge, Oscar Key, Kürşat Kaya, Jonas Kübler, Adèle Frankel, Tobias Schröder, Anurag Garg, Jan Hendrik Metzen, David Salinas, Simon Bing, Kristina Collins, Tuana Çelik, Vahid Balazadeh, Lydia Sidhoum, Tomás Pereda, Brendan Roof, Andrej Tschalzev, Siyuan Guo, Philipp Singer, Lennart Purucker, Jake Robertson, Marie Salmon, Philipp Jund, Jerry Chen, Diana Kriuchkova, Arthur Cahu, Eliott Kalfon, Adrian Hayler, Georg Grab, Vitor Monteiro, Lilly Wehrhahn, Dominik Safaric, Clara Cornu, Alan Arazi, Rylee Grace, Simone Alessi, Mihir Manium, Bernhard Schölkopf, Yann LeCun, Madelon Hulsebos, Sauraj Gambhir, Noah Hollmann, and Frank Hutter. TabPFN-3.5: Technical report, 2026. URL https://arxiv.org/abs/2609.17895.

D. V. Lindley. On a measure of the information provided by an experiment. *The Annals of Mathematical Statistics*, 27(4):986–1005, 1956.

Jan-Matthis Lueckmann, Jan Boelts, David Greenberg, Pedro Goncalves, and Jakob Macke. Benchmarking simulation-based inference. In *International conference on artificial intelligence and statistics*, pp. 343–351. PMLR, 2021.

David Madigan and Adrian E Raftery. Model selection and accounting for model uncertainty in graphical models using Occam's window. *Journal of the American Statistical Association*, 89 (428):1535–1546, 1994. doi: 10.1080/01621459.1994.10476894. URL https://doi.org/10.1080/01621459.1994.10476894.

Patryk Marszałek, Tomasz Kuśmierczyk, Witold Wydmański, Jacek Tabor, and Marek Śmieja. ZEUS: Zero-shot embeddings for unsupervised separation of tabular data. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/7d3298e48220b289318b533a848ea069-Abstract-Conference.html.

Sören Mindermann, Jan M Brauner, Muhammed T Razzak, Mrinank Sharma, Andreas Kirsch, Winnie Xu, Benedikt Höltgen, Aidan N Gomez, Adrien Morisot, Sebastian Farquhar, and Yarin Gal. Prioritized training on points that are learnable, worth learning, and not yet learnt. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 15630–15649. PMLR, 2022. URL https://proceedings.mlr.press/v162/mindermann22a.html.

Philip Mortimer, Cristiana Diaconu, Tommy Rochussen, Bruno Kacper Mlodozeniec, and Richard E. Turner. Incremental transformer neural processes. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=diRgjfD2vu.

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do Bayesian inference. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=KSugKcbNf9.

Thomas Nagler. Statistical foundations of prior-data fitted networks. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 25660–25676. PMLR, 2023. URL https://proceedings.mlr.press/v202/nagler23a.html.

Tung Nguyen and Aditya Grover. Transformer neural processes: Uncertainty-aware meta learning via sequence modeling. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/nguyen22b.html.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Ëdouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12 (85):2825–2830, 2011. URL https://www.jmlr.org/papers/v12/pedregosa11a.html.

J. Peters, D. Janzing, and B. Schölkopf. *Elements of Causal Inference: Foundations and Learning Algorithms*. MIT Press, Cambridge, MA, USA, 2017.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICLv2: A better, faster, scalable, and open tabular foundation model. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=SxsyLjIfWB.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Tom Rainforth, Adam Foster, Desi R Ivanova, and Freddie Bickford Smith. Modern Bayesian experimental design. *Statistical Science*, 39(1):100–114, 2024. doi: 10.1214/23-STS915. URL https://doi.org/10.1214/23-STS915.

Arik Reuter, Anish Dhir, Cristiana Diaconu, Jake Robertson, Ole Ossen, Frank Hutter, Adrian Weller, Mark van der Wilk, and Bernhard Schölkopf. Use what you know: Causal foundation models with partial graphs. In *Forty-third International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.14972.

Samuel S. Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep information propagation. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=H1W1UN9gg.

Ellen Su, Andres Potapczynski, Shikai Qiu, Edward Hughes, and Andrew Gordon Wilson. Epiplexity guided data selection and generation for out-of-distribution generalization. *arXiv preprint arXiv:2608.11746*, 2026. URL https://arxiv.org/abs/2608.11746.

Yefan Tao, Xiyuan Zhang, Xinyi Liu, Boran Han, Danielle Maddix, Haoyang Fang, Zhen Han, Jiading Gai, Xuanqing Liu, Michael Bohlke-Schneider, et al. Mitra-v2 technical report. *arXiv preprint arXiv:2609.04540*, 2026. URL https://arxiv.org/abs/2609.04540.

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

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy Liang, Quoc V. Le, Tengyu Ma, and Adams Wei Yu. DoReMi: Optimizing data mixtures speeds up language model pretraining. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/dcba6be91359358c2355cd920da3fcbd-Abstract-Conference.html.

Yilun Xu, Shengjia Zhao, Jiaming Song, Russell Stewart, and Stefano Ermon. A theory of usable information under computational constraints. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1eBeyHFDH.

A Additional Background: $\mathcal{V}$-information and Epiplexity

In eq. (3), predictive information gain is defined using the posterior predictive induced by π. When this distribution is intractable, we evaluate predictive information using tractable predictors. This requires positing a family of predictors $\mathcal{V}$, which may be constrained through some parametrisation or finite compute. For this predictive family $\mathcal{V}$, the predictive $\mathcal{V}$-information is defined by (Xu et al., 2020)

$$I_{\mathcal{V}}(X \to Y) = \inf_{f \in \mathcal{V}} \mathbb{E}[-\log f[\varnothing](Y)] - \inf_{f \in \mathcal{V}} \mathbb{E}[-\log f[X](Y)]. \tag{7}$$

Here, $f[x]$ and $f[\varnothing]$ are predictive distributions over Y with and without side information x, respectively. The expectations are taken under the joint distribution of X and Y.

In contrast, the recently proposed notion of epiplexity (Finzi et al., 2026) requires a cost to be paid for the predictor itself. Where $\mathcal{V}$-information fixes a family and takes the infimum over it, epiplexity (Finzi et al., 2026) scores a model by the two-part code

$$P^\star = \arg\min_{P \in \mathcal{P}_T} \{|P| + \mathbb{E}[\log 1/P(X)]\}, \qquad S_T(X) := |P^\star|, \quad H_T(X) := \mathbb{E}[\log 1/P^\star(X)],$$

where $\mathcal{P}_T$ is the set of programs that both sample and evaluate probabilities within a time bound T.

Similar to the concepts above, we measure learnable structure relative to a predictor, in the absence of the "ground-truth" prior-induced posterior predictive.

B Why n Held-out Data Points?

A natural alternative to TEPIG in eq. (4) is to measure the reduction in predictive uncertainty for a single held-out data point, i.e. to set $m = 1$:

$$\mathrm{TEPIG}_1(q, \theta, n) := \mathbb{E}_{\pi(\mathcal{D}_n, \boldsymbol{x}, y|\theta)} \left[\log q(y \mid \boldsymbol{x}, \mathcal{D}_n) - \log q(y \mid \boldsymbol{x})\right]. \tag{8}$$

Recall from section 4.1 that

$$\ell_k(q, \theta) := \mathbb{E}_{\pi(\mathcal{D}_k, \boldsymbol{x}_{k+1}, y_{k+1}|\theta)} \left[-\log q(y_{k+1} \mid \boldsymbol{x}_{k+1}, \mathcal{D}_k)\right] \tag{9}$$

denotes the expected held-out negative log-likelihood after conditioning on k context points, with $\mathcal{D}_0 = \emptyset$. The single-point criterion can then be written as

$$\mathrm{TEPIG}_1(q, \theta, n) = \ell_0(q, \theta) - \ell_n(q, \theta). \tag{10}$$

It therefore depends only on the two endpoints of the expected loss curve. In particular, it is insensitive to how quickly the task becomes predictable between zero and n observations.

Figure 7 shows a simple example. We compare a linear regression task θ with a sinusoidal regression task θ'. Both tasks have observation-noise standard deviation $\sigma = 0.1$. The linear function is scaled such that both tasks have the same marginal signal power,

$$\mathbb{E}_{\pi(\boldsymbol{x}|\theta)}[f_\theta(\boldsymbol{x})^2] = \mathbb{E}_{\pi(\boldsymbol{x}|\theta')}[f_{\theta'}(\boldsymbol{x})^2].$$

We use the same fixed GP predictor q for both tasks. For the GP predictor and input distribution used in this example, the matched signal power and identical observation noise yield the same expected prior-predictive loss, $\ell_0(q, \theta) = \ell_0(q, \theta')$.

Although the two tasks have the same initial expected loss, the sinusoidal task requires more observations before the predictor can resolve its local variation. The linear task is learned after relatively few observations, whereas the loss of the sinusoidal task remains elevated for substantially longer. Nevertheless, after $n = 100$ observations, the predictor has learned both functions approximately to the same noise floor. The initial and final losses are therefore nearly identical for the two tasks.

This produces the seemingly paradoxical result shown in fig. 8. The single-point criterion $\mathrm{TEPIG}_1(q, \theta, n)$ assigns almost exactly the same score to the linear and sinusoidal tasks, because it only observes the two endpoints of their loss curves. It discards the fact that the sinusoidal task requires many more observations to become predictable.

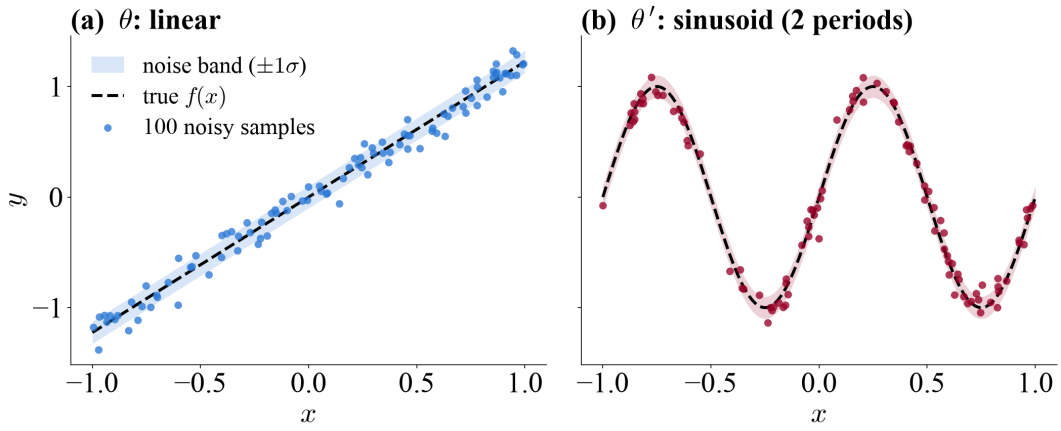


Figure 7: The two regression tasks used in our illustrative example. **(a)** The linear task θ. **(b)** The sinusoidal task θ'. The dashed curves show the conditional mean functions, the shaded regions show one observation-noise standard deviation σ around each mean, and the points show 100 noisy observations. The two tasks have matched marginal signal power and identical observation noise.

By contrast, the joint criterion considers the learning trajectory. As shown in eq. (5),

$$\mathrm{TEPIG}(q, \theta, n) = \sum_{k=0}^{n-1} \left[\ell_k(q, \theta) - \ell_{k+n}(q, \theta)\right]. \tag{11}$$

Under the predictive-convergence assumption used in eq. (6), this is approximated by

$$\mathrm{TEPIG}_C(q, \theta, n) = \sum_{k=0}^{n-1} \left[\ell_k(q, \theta) - \ell_n(q, \theta)\right]. \tag{12}$$

This approximation is the discrete area of the loss curve above its level at context size n. It therefore captures the difference between the learning trajectories. In this example, the ratio between the sinusoidal and linear scores is

$$\frac{\mathrm{TEPIG}_1(q, \theta', n)}{\mathrm{TEPIG}_1(q, \theta, n)} = 1.004 \;\; [0.995,\, 1.013], \qquad \frac{\mathrm{TEPIG}_C(q, \theta', n)}{\mathrm{TEPIG}_C(q, \theta, n)} = 1.424 \;\; [1.307,\, 1.546],$$

where brackets give 95% percentile bootstrap confidence intervals over 20 independent repetitions. The interval for TEPIG_1 contains one, whereas the interval for TEPIG_C lies well above it.

Considering n jointly predicted held-out points therefore makes TEPIG sensitive to the learning trajectory over context sizes $0, \ldots, 2n-1$, rather than only to the difference between predictions with zero and n context points. Under predictive convergence, TEPIG_C captures this dependence through the loss curve up to context size n. This sensitivity is desirable when TEPIG is used as a measure of learnable structure: two tasks that are eventually predicted equally well can still contain different amounts of structure for a given predictor if they require substantially different numbers of observations to learn.

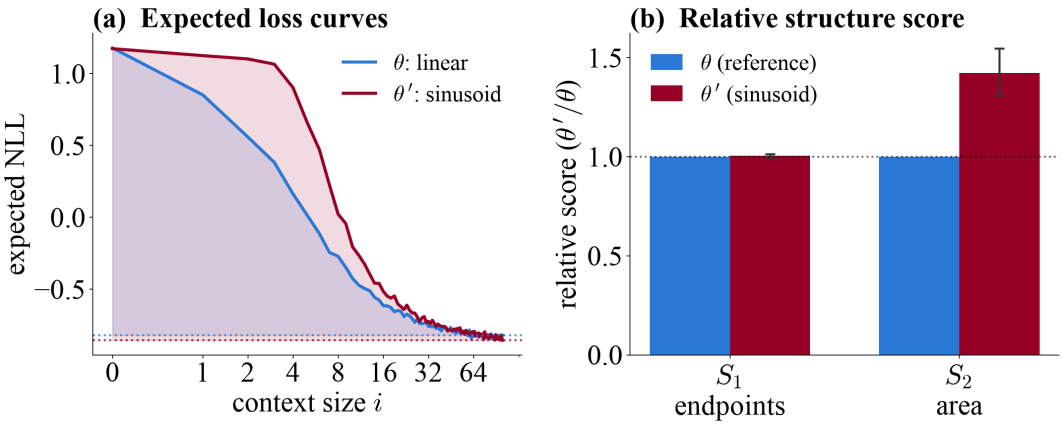


Figure 8: Comparison of single-point predictive information gain TEPIG_1 and the predictive-convergence approximation TEPIG_C to joint predictive information gain. **(a)** Expected held-out negative log-likelihood $\ell_k(q, \theta)$ as a function of context size k (note the logarithmic context axis); the dotted lines mark each task's loss at $k = n$, and the shaded regions are the areas summed by TEPIG_C. The linear and sinusoidal tasks have nearly identical initial and final losses, but the sinusoidal task becomes predictable more slowly. **(b)** The endpoint-only score TEPIG_1 assigns almost identical values to the two tasks. The area-based score TEPIG_C captures the different learning trajectories and assigns a larger value to the sinusoidal task. The bars show each score for θ' relative to its value for θ; error bars denote 95% bootstrap confidence intervals over 20 repetitions.

C Further Illustrations of TEPIG

C.1 Additional function families

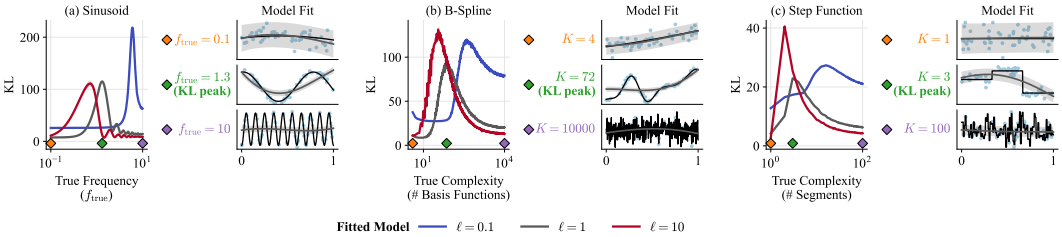


Figure 9: TEPIG captures learnable structure for a given model: increasing the complexity of a relationship increases TEPIG only while that structure remains learnable. In (a), we illustrate this using $y = \sin(2\pi f x) + \epsilon$, with $x \sim \text{Uniform}(0, 1)$ and $\epsilon \sim \mathcal{N}(0, 0.01)$. **Left**: Larger length scale λ implies stronger smoothness assumptions. TEPIG initially rises and then falls with frequency f for each of three GPs. The peak shifts towards higher frequencies for smaller-length-scale models, reflecting their ability to capture more wiggliness. **Right**: example fits for $\lambda = 1.0$ illustrate three regimes. At $f = 0.1$, the almost constant relationship is easy to learn and yields low TEPIG. At $f = 1.3$, TEPIG peaks, and the GP captures the sinusoid reasonably well. At $f = 10$, the oscillations are too rapid for the GP to capture, leading to low TEPIG again. (b) Random cubic B-splines with K_b evenly spaced basis functions and coefficients $w_i \sim \mathcal{N}(0, 1)$, rescaled to unit standard deviation, with $\epsilon \sim \mathcal{N}(0, 0.01)$. (c) Random piecewise-constant step functions with K_s equal-width segments whose heights are drawn from $\mathcal{N}(0, 1)$ and rescaled to unit standard deviation, with $\epsilon \sim \mathcal{N}(0, 0.1)$; in both cases $x \sim \text{Uniform}(0, 1)$. As in the main text, TEPIG increases with complexity while the structure remains learnable, peaks, and then decreases once the relationship becomes too complex for the proxy model to capture, with the peak occurring at higher complexity for less regularised (smaller λ) models.

D Estimating TEPIG for different ML Models

TEPIG can be estimated for any predictor providing a log-likelihood (or something that can be interpreted as such). For all point predictors, we assume a Gaussian observation model with fixed variance σ^2, so *differences* in NLL equal differences in squared error. This allows to rank the TEPIG of different tasks relative to a fixed model, while comparing score magnitudes across predictors q requires specifying their predictive likelihoods on a common target scale.

The computational cost of computing TEPIG for different models q depends on whether model fits can be reused across context sizes.

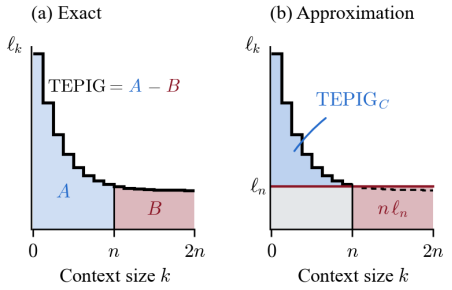


Figure 10: (a) Exact TEPIG is the difference between loss sums A and B. (b) Under predictive convergence, subtracting the baseline $n\ell_n$ leaves TEPIG$_C$.

D.1 K-nearest-neighbour

For our experiments, we use a custom KNN regressor, similar to the nearest-neighbour regressor in Scikit-Learn (Pedregosa et al., 2011) but with adaptive neighbour counts and tricube distance weighting. At each context size $k \geq 2$, we select the number of neighbours $K \in \{1, \ldots, \min(30, k-1)\}$ by minimising leave-one-out squared error (which can be done very efficiently once the distance-matrix between all points is computed). For context size $k = 1$, we use $K = 1$.

Our implementation recomputes distances at each context size, but reuses neighbour searches across candidate counts and vectorises computations across datasets. We use MSE, corresponding up to

scale and an additive constant to Gaussian NLL with fixed variance, and compute the predictive-convergence estimator TEPIG_C (eq. (6)) using every context size $k = 1, \ldots, n$.

Since KNN is very cheap to fit, the overhead it introduces for scoring datasets can become basically unnoticeable compared to the computations of the mechanisms for data-sampling that need to be done when generating data from the complex TabPFN v2 regression prior (Qu et al., 2026) and the SCM prior in (Reuter et al., 2026).

D.2 Autoregressive (causally masked) Transformers

A causally masked predictor, such as basically all modern LLMs (Radford et al., 2019), but also the recently proposed Incremental Transformer Neural Process (Mortimer et al., 2026), can evaluate all next-target log densities in a single forward pass. This is because each prediction sees its query input and preceding labelled examples but no current or future targets. Computing TEPIG with those models thus requires only a single forward-pass.

D.3 Estimating TEPIG efficiently with TFMs

TFMs with full self-attention among context points, such as TabPFN and TabICL (Hollmann et al., 2023; Grinsztajn et al., 2026; Jäger et al., 2026; Qu et al., 2026), require separate forward passes for different context sizes. Each context point's representation depends on the other points in the context, so changing the context requires recomputing these representations. Estimating the loss curve therefore incurs the accumulated cost of evaluating every selected context size.

To quantify this cost, consider a fixed ordering of $2n$ samples. At context size k, the first k samples form the labelled context and the remaining $2n-k$ samples serve as held-out prediction targets. The context points attend to one another, requiring k^2 attention scores/interactions, while the held-out points attend to the context, requiring $(2n-k)k$ scores/interactions. Assuming a constant cost C per interaction, the attention cost of this evaluation is

$$A(k) = Ck^2 + C(2n-k)k = 2Cnk.$$

Thus, although attention within the context has quadratic cost, the combined context and prediction cost is linear in k when the total number of samples is fixed at $2n$. Summing over all context sizes from 1 through K gives the number of operations $A_{\leq K}$ up to size K:

$$A_{\leq K} = \sum_{k=1}^{K} 2Cnk = CnK(K+1),$$

which grows quadratically with the maximum context size K.

The predictive-convergence approximation allows us to stop evaluating the loss curve at context size n and use the loss at n as the plateau estimate. This avoids evaluating the larger, more expensive contexts. Relative to evaluating all context sizes through $2n-1$, the fraction of attention cost retained is

$$\frac{A_{\leq n}}{A_{\leq 2n-1}} = \frac{Cn^2(n+1)}{2Cn^2(2n-1)} = \frac{n+1}{2(2n-1)} \longrightarrow \frac{1}{4}.$$

The resulting saving of attention scores/interaction therefore approaches 75% and is already 74.62% at $n = 100$.

D.4 Validating the estimation choices for TEPIG

We validate our two approximations to TEPIG using TabICL v2 (Qu et al., 2026) as the proxy model q on datasets drawn from the prior it was trained on ($n = m = 512$), reporting Spearman rank correlation as TEPIG is used to rank datasets. Predictive convergence (TEPIG_C, eq. (6)) agrees with the un-approximated TEPIG (eq. (5)) at 0.9593 $[0.9546, 0.9634]$ over $4{,}805$ datasets, at half the curve evaluations. Figure 11 ablates the group size g against the $g = 1$ estimator: $g = 20$ reduces the evaluated context sizes from 511 to 26 while retaining a correlation of 0.9086 $[0.8962, 0.9197]$. For larger group sizes, the correlation decreases further. ,s

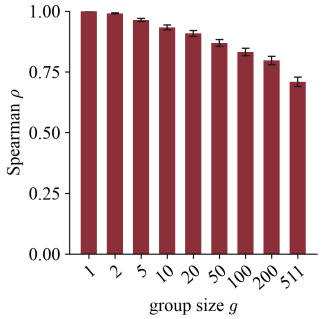


Figure 11: Spearman correlation between TEPIG at group size g and the $g = 1$ reference, over $5{,}000$ datasets. Error bars are 95% paired-bootstrap CIs; $g = 1$ is one by construction.

D.5 Choice of proxy model

TEPIG for tilting priors is especially useful if a cheap proxy model q can be used for this task. We therefore compare task rankings from nine proxy models with a reference ranking computed using TabICL v2 (Qu et al., 2026), on both tabular priors we study.

All proxies are point predictors scored by squared error (section D), so only the task *ranking* they induce is comparable to the TabICL v2 reference ranking, and we report Spearman rank correlation. Every proxy uses $n = m = 512$, one random context ordering, and group size $g = 20$. Within each prior, all proxy models and TabICL v2 see exactly the same task draws and the same context orderings, so all correlations are paired; intervals are 95% paired-bootstrap percentile intervals over $4{,}000$ resamples. Apart from the custom KNN variants, the proxies are the corresponding scikit-learn regressors at their default settings (Pedregosa et al., 2011).

We compute the TabICL v2 reference rankings with different TEPIG estimators on the two priors, reflecting what was computationally feasible on each. On the TabICL v2 prior we use TabICL v2's own TEPIG_C (eq. (6)) over $5{,}000$ tasks, evaluated both at $g = 1$ and at $g = 20$. The latter matches the proxies' own group size. On the SCM prior from Reuter et al. (2026) we use the un-approximated TEPIG (eq. (5)) over $1{,}000$ tasks, evaluated at $g = 5$, a resolution that retains a correlation of $0.9642\ [0.9570, 0.9706]$ with $g = 1$ (section D.4).

Table 1: Spearman rank correlation between task rankings from each proxy model and the TabICL v2 reference ranking, on the TabICL v2 prior and on the SCM prior.

	TabICL v2 prior ($N = 5{,}000$)		SCM prior ($N = 1{,}000$)
Proxy	vs. TEPIG_C, g=1	vs. TEPIG_C, g=20	vs. TEPIG, g=5
Random forest	$0.420\ [0.395, 0.445]$	$0.476\ [0.452, 0.500]$	$0.135\ [0.072, 0.199]$
Hist. gradient boosting	$0.433\ [0.407, 0.457]$	$0.473\ [0.449, 0.496]$	$0.270\ [0.207, 0.330]$
KNN (tricube, adaptive K)	$0.433\ [0.409, 0.457]$	$0.468\ [0.445, 0.491]$	$0.206\ [0.142, 0.267]$
KNN (uniform, K=5)	$0.415\ [0.389, 0.439]$	$0.450\ [0.426, 0.473]$	$0.258\ [0.196, 0.318]$
Extra trees	$0.345\ [0.319, 0.371]$	$0.398\ [0.373, 0.422]$	$0.080\ [0.015, 0.144]$
xRFM	$0.346\ [0.319, 0.371]$	$0.362\ [0.336, 0.388]$	$0.202\ [0.137, 0.267]$
LightGBM	$0.192\ [0.164, 0.220]$	$0.258\ [0.232, 0.286]$	$-0.051\ [-0.114, 0.014]$
Gaussian process	$0.153\ [0.126, 0.180]$	$0.144\ [0.116, 0.171]$	$0.261\ [0.201, 0.321]$
Ridge	$0.038\ [0.009, 0.066]$	$0.097\ [0.069, 0.125]$	$-0.118\ [-0.181, -0.052]$

All proxies are evaluated at group size $g = 20$ with $n = m = 512$. Brackets give 95% paired-bootstrap percentile intervals over $4{,}000$ resamples.

E Relating TEPIG to divergence between prior and posterior over parameters

Throughout, let θ denote the fixed data-generating task, while ϕ denotes the parameter variable in the Bayesian model q. The model q need not coincide with the data-generating distribution π. Under the assumptions below, TEPIG equals the expected KL divergence from the context posterior $q(\phi \mid \mathcal{D}_n)$ to the prior $q(\phi)$, plus a correction term. This correction compares the context posterior's expected

KL divergence to the test-only posterior and to the posterior obtained by combining the context and test sets.

Theorem 1. *Let $n, m \geq 1$ and let $\mathcal{D}_n$ and $(\boldsymbol{x}_{1:m}, y_{1:m})$ be context and prediction sets sampled independently conditional on the same task θ, so that*

$$\pi(\mathcal{D}_n, \boldsymbol{x}_{1:m}, y_{1:m} \mid \theta) = \pi(\mathcal{D}_n \mid \theta)\pi(\boldsymbol{x}_{1:m}, y_{1:m} \mid \theta).$$

Suppose that q admits a Bayesian representation in which all posteriors and predictive densities below arise from the same prior and likelihood by Bayesian updating and marginalisation. Specifically, let $q(\phi)$ be the prior density and $q(\mathcal{D}_n \mid \phi)$ the context likelihood, with marginal likelihood

$$q(\mathcal{D}_n) = \int q(\mathcal{D}_n \mid \phi) q(\phi) \, \mathrm{d}\phi,$$

and context posterior

$$q(\phi \mid \mathcal{D}_n) = \frac{q(\mathcal{D}_n \mid \phi) q(\phi)}{q(\mathcal{D}_n)}.$$

Conditional on ϕ and the query inputs, the prediction labels are independent of the context under q:

$$q(y_{1:m} \mid \boldsymbol{x}_{1:m}, \mathcal{D}_n, \phi) = q(y_{1:m} \mid \boldsymbol{x}_{1:m}, \phi).$$

Unlabelled test inputs do not update parameter beliefs under q, either before or after conditioning on the context:

$$q(\phi \mid \boldsymbol{x}_{1:m}) = q(\phi), \qquad q(\phi \mid \mathcal{D}_n, \boldsymbol{x}_{1:m}) = q(\phi \mid \mathcal{D}_n).$$

The joint predictive densities used in TEPIG are the corresponding Bayesian marginals:

$$q(y_{1:m} \mid \boldsymbol{x}_{1:m}) = \int q(y_{1:m} \mid \boldsymbol{x}_{1:m}, \phi) q(\phi) \, \mathrm{d}\phi,$$

$$q(y_{1:m} \mid \boldsymbol{x}_{1:m}, \mathcal{D}_n) = \int q(y_{1:m} \mid \boldsymbol{x}_{1:m}, \phi) q(\phi \mid \mathcal{D}_n) \, \mathrm{d}\phi.$$

The corresponding prediction-only and combined posteriors are

$$q(\phi \mid \boldsymbol{x}_{1:m}, y_{1:m}) = \frac{q(y_{1:m} \mid \boldsymbol{x}_{1:m}, \phi) q(\phi)}{q(y_{1:m} \mid \boldsymbol{x}_{1:m})},$$

$$q(\phi \mid \mathcal{D}_n, \boldsymbol{x}_{1:m}, y_{1:m}) = \frac{q(y_{1:m} \mid \boldsymbol{x}_{1:m}, \phi) q(\phi \mid \mathcal{D}_n)}{q(y_{1:m} \mid \boldsymbol{x}_{1:m}, \mathcal{D}_n)}.$$

Assume that all normalising constants are strictly positive and finite almost surely under $\pi(\mathcal{D}_n, \boldsymbol{x}_{1:m}, y_{1:m} \mid \theta)$. All relevant densities and log-densities are assumed finite almost surely under the measures used below, and all log-likelihood expectations, KL divergences, and outer expectations appearing below are assumed well defined and finite. Then

$$\mathrm{TEPIG}(q, \theta, n, m) = \mathbb{E}_{\pi(\mathcal{D}_n \mid \theta)} \left[\mathbb{D}_{\mathrm{KL}}\big(q(\phi \mid \mathcal{D}_n) \,\big\|\, q(\phi)\big) \right] + \eta(q, \theta, n, m), \tag{13}$$

where

$$\eta(q, \theta, n, m) := \mathbb{E}_{\pi(\mathcal{D}_n, \boldsymbol{x}_{1:m}, y_{1:m} \mid \theta)} \Bigg[\mathbb{D}_{\mathrm{KL}}\big(q(\phi \mid \mathcal{D}_n) \,\big\|\, q(\phi \mid \mathcal{D}_n, \boldsymbol{x}_{1:m}, y_{1:m})\big) - \mathbb{D}_{\mathrm{KL}}\big(q(\phi \mid \mathcal{D}_n) \,\big\|\, q(\phi \mid \boldsymbol{x}_{1:m}, y_{1:m})\big) \Bigg]. \tag{14}$$

The correction vanishes when the context posterior $q(\phi \mid \mathcal{D}_n)$ has the same expected KL divergence to the test-only posterior $q(\phi \mid \boldsymbol{x}_{1:m}, y_{1:m})$ as to the combined posterior $q(\phi \mid \mathcal{D}_n, \boldsymbol{x}_{1:m}, y_{1:m})$.

The expected posterior-to-prior KL measures how much observing the context changes the model's beliefs about its parameters. TEPIG measures the resulting improvement in joint predictive log loss on test data from the same task. In general, these quantities need not agree: a context can distinguish parameter settings whose predictions at the test inputs are nearly identical. The correction accounts for this difference by comparing the context posterior with the posteriors obtained from the test set alone and from both sets together.

Related ideas. The proof uses the standard ELBO decomposition (Blei et al., 2017). The expected posterior-to-prior KL in equation 13 also connects TEPIG to the parameter information gain underlying BALD (Houlsby et al., 2011), although TEPIG averages over data from a fixed task rather than the model's predictive distribution.

Proof of Theorem 1. Fix a realisation $(\mathcal{D}_n, \boldsymbol{x}_{1:m}, y_{1:m})$ for which the stated conditions hold. We apply the exact evidence lower bound (ELBO) identity twice, using $q(\phi \mid \mathcal{D}_n)$ as the auxiliary density in both applications. This identity expresses the negative log marginal likelihood as the expected negative log likelihood plus the KL divergence from the auxiliary density to the prior, minus its KL divergence to the updated posterior.

For a generic Bayesian model with prior $p(\phi)$ and likelihood $p(\mathcal{Z} \mid \phi)$, write

$$p(\mathcal{Z}) = \int p(\mathcal{Z} \mid \phi) p(\phi) \, \mathrm{d}\phi, \qquad p(\phi \mid \mathcal{Z}) = \frac{p(\mathcal{Z} \mid \phi) p(\phi)}{p(\mathcal{Z})}.$$

For any auxiliary density $r(\phi)$, assuming $p(\mathcal{Z}) > 0$ and all terms are finite, the identity is

$$\begin{aligned} -\log p(\mathcal{Z}) = \underbrace{\mathbb{E}_{r(\phi)}[-\log p(\mathcal{Z} \mid \phi)] + \mathbb{D}_{\mathrm{KL}}\big(r(\phi) \,\big\|\, p(\phi)\big)}_{-\,\mathrm{ELBO}(r;\mathcal{Z})} \\ - \mathbb{D}_{\mathrm{KL}}\big(r(\phi) \,\big\|\, p(\phi \mid \mathcal{Z})\big) \,. \end{aligned} \tag{15}$$

Equivalently, the log marginal likelihood equals the ELBO plus $\mathbb{D}_{\mathrm{KL}}(r(\phi) \| p(\phi \mid \mathcal{Z}))$; this nonnegative KL divergence is the gap that makes the ELBO a lower bound. We apply equation 15 twice below, using $q(\phi \mid \mathcal{D}_n)$ as the auxiliary density in both applications.

With prior $q(\phi)$ and likelihood $q(y_{1:m} \mid \boldsymbol{x}_{1:m}, \phi)$, the exact ELBO identity gives

$$\begin{aligned} -\log q(y_{1:m} \mid \boldsymbol{x}_{1:m}) = \mathbb{E}_{q(\phi \mid \mathcal{D}_n)}\left[-\log q(y_{1:m} \mid \boldsymbol{x}_{1:m}, \phi)\right] \\ + \mathbb{D}_{\mathrm{KL}}\big(q(\phi \mid \mathcal{D}_n) \,\big\|\, q(\phi)\big) \\ - \mathbb{D}_{\mathrm{KL}}\big(q(\phi \mid \mathcal{D}_n) \,\big\|\, q(\phi \mid \boldsymbol{x}_{1:m}, y_{1:m})\big) \,. \end{aligned} \tag{16}$$

Equivalently, taking logarithms of the prediction-only posterior formula, then taking expectations under $q(\phi \mid \mathcal{D}_n)$, adding and subtracting $\mathbb{E}_{q(\phi \mid \mathcal{D}_n)}[\log q(\phi \mid \mathcal{D}_n)]$, and rearranging yields equation 16.

We now use $q(\phi \mid \mathcal{D}_n)$ as the prior, with the same likelihood $q(y_{1:m} \mid \boldsymbol{x}_{1:m}, \phi)$ and the same auxiliary density $q(\phi \mid \mathcal{D}_n)$. The marginal likelihood is $q(y_{1:m} \mid \boldsymbol{x}_{1:m}, \mathcal{D}_n)$, and the updated posterior is

$$q(\phi \mid \mathcal{D}_n, \boldsymbol{x}_{1:m}, y_{1:m}) = \frac{q(y_{1:m} \mid \boldsymbol{x}_{1:m}, \phi) q(\phi \mid \mathcal{D}_n)}{q(y_{1:m} \mid \boldsymbol{x}_{1:m}, \mathcal{D}_n)}.$$

The exact ELBO identity therefore gives

$$\begin{aligned} -\log q(y_{1:m} \mid \boldsymbol{x}_{1:m}, \mathcal{D}_n) = \mathbb{E}_{q(\phi \mid \mathcal{D}_n)}\left[-\log q(y_{1:m} \mid \boldsymbol{x}_{1:m}, \phi)\right] \\ + \underbrace{\mathbb{D}_{\mathrm{KL}}\big(q(\phi \mid \mathcal{D}_n) \,\big\|\, q(\phi \mid \mathcal{D}_n)\big)}_{=0} \\ - \mathbb{D}_{\mathrm{KL}}\big(q(\phi \mid \mathcal{D}_n) \,\big\|\, q(\phi \mid \mathcal{D}_n, \boldsymbol{x}_{1:m}, y_{1:m})\big) \,. \end{aligned}$$

Since the auxiliary density equals the prior in this application, the middle KL divergence vanishes. Rearranging gives

$$\begin{aligned} \mathbb{E}_{q(\phi \mid \mathcal{D}_n)}\left[-\log q(y_{1:m} \mid \boldsymbol{x}_{1:m}, \phi)\right] \\ = -\log q(y_{1:m} \mid \boldsymbol{x}_{1:m}, \mathcal{D}_n) \\ + \mathbb{D}_{\mathrm{KL}}\big(q(\phi \mid \mathcal{D}_n) \,\big\|\, q(\phi \mid \mathcal{D}_n, \boldsymbol{x}_{1:m}, y_{1:m})\big) \,. \end{aligned} \tag{17}$$

Substitute equation 17 into equation 16 to eliminate the common expected negative log likelihood. Rearranging yields

$$\begin{aligned} \log q(y_{1:m} \mid \boldsymbol{x}_{1:m}, \mathcal{D}_n) - \log q(y_{1:m} \mid \boldsymbol{x}_{1:m}) \\ = \mathbb{D}_{\mathrm{KL}}\big(q(\phi \mid \mathcal{D}_n) \,\big\|\, q(\phi)\big) \\ + \mathbb{D}_{\mathrm{KL}}\big(q(\phi \mid \mathcal{D}_n) \,\big\|\, q(\phi \mid \mathcal{D}_n, \boldsymbol{x}_{1:m}, y_{1:m})\big) \\ - \mathbb{D}_{\mathrm{KL}}\big(q(\phi \mid \mathcal{D}_n) \,\big\|\, q(\phi \mid \boldsymbol{x}_{1:m}, y_{1:m})\big) \,. \end{aligned} \tag{18}$$

Taking expectations under $\pi(\mathcal{D}_n, \boldsymbol{x}_{1:m}, y_{1:m} \mid \theta)$ gives

$$\begin{aligned}
&\mathbb{E}_{\pi(\mathcal{D}_n, \boldsymbol{x}_{1:m}, y_{1:m}|\theta)}\left[\log \frac{q(y_{1:m} \mid \boldsymbol{x}_{1:m}, \mathcal{D}_n)}{q(y_{1:m} \mid \boldsymbol{x}_{1:m})}\right] \\
&\quad = \mathbb{E}_{\pi(\mathcal{D}_n|\theta)}\left[\mathbb{D}_{\mathrm{KL}}\big(q(\phi \mid \mathcal{D}_n) \,\big\|\, q(\phi)\big)\right] \\
&\qquad + \mathbb{E}_{\pi(\mathcal{D}_n, \boldsymbol{x}_{1:m}, y_{1:m}|\theta)}\Big[\mathbb{D}_{\mathrm{KL}}\big(q(\phi \mid \mathcal{D}_n) \,\big\|\, q(\phi \mid \mathcal{D}_n, \boldsymbol{x}_{1:m}, y_{1:m})\big) \\
&\qquad\qquad - \mathbb{D}_{\mathrm{KL}}\big(q(\phi \mid \mathcal{D}_n) \,\big\|\, q(\phi \mid \boldsymbol{x}_{1:m}, y_{1:m})\big)\Big].
\end{aligned}$$

The first KL divergence depends only on $\mathcal{D}_n$, which is why its expectation reduces to the context marginal $\pi(\mathcal{D}_n \mid \theta)$. The left-hand side is $\mathrm{TEPIG}(q, \theta, n, m)$, and the final expectation is $\eta(q, \theta, n, m)$ from equation 14. Thus

$$\mathrm{TEPIG}(q, \theta, n, m) = \mathbb{E}_{\pi(\mathcal{D}_n|\theta)}\left[\mathbb{D}_{\mathrm{KL}}\big(q(\phi \mid \mathcal{D}_n) \,\big\|\, q(\phi)\big)\right] + \eta(q, \theta, n, m),$$

which proves equation 13 for arbitrary n and m. □

F Details on investigating the TabICL v2 prior

Throughout section 5, the proxy model q is the pretrained TabICL v2 regressor (Qu et al., 2026) used with a single ensemble member and a single context ordering. All scores are estimates of TEPIG_C (eq. (6)) with $c = 1$, computed with the grouped estimator of section D. At each context size $k < n$, the one-step loss averages over the unused context rows together with n' held-out points; the plateau loss at n uses the held-out points only. Scores are reported in nats.

F.1 Whole-prior analysis

Prior and interventions. All datasets are drawn from the TabICL v2 graph-based SCM regression prior with its default configuration changing one hyperparameter at a time:

- **Connectivity.** The graph sampler draws a global offset $o \sim \mathrm{Cauchy}(0, 1) + \delta$ and includes each forward edge $i \to j$ with probability $\mathrm{sigmoid}(o + \alpha_i + \beta_j)$, where the node importances α_i, β_j are also Cauchy distributed. We vary the connectivity offset $\delta \in \{-32, -4, 0, 4, 32\}$, where $\delta = 0$ is the original prior.
- **Context size.** We vary $n \in \{64, 128, 256, 512, 1024, 2024\}$ for the original prior. Each dataset is generated once with 2,536 rows. The first n rows form the context, and the last 512 rows form a held-out set shared across all n, so comparisons across context sizes are paired at the dataset level.
- **Internal noise.** We add Gaussian noise with a fixed standard deviation σ to every node of the SCM, at the insertion point of the prior's native Gaussian-noise option (after the node's converters, before passing the node to its children). We vary $\sigma \in \{0, 10^{-4}, 10^{-3}, 0.01, 0.03, 0.1, 0.3, 1\}$, where $\sigma = 0$ is the original prior. No noise is added directly to the observed targets.

Sampling and estimation settings. Table 2 summarises the three whole-prior studies. Feature counts follow the original prior over 2–100 features, with padding to width 100. We retain its preprocessing and rejection rules, and use checkpoint `tabicl-regressor-v2-20260212.ckpt`. Each replicate is a fresh prior dataset. The context-size study reuses the same parent dataset and held-out set at every n, without regenerating or rescaling the data; context membership is nested, although each n has its own random ordering. Connectivity and noise studies reuse seed indices across settings, but interventions can change downstream random-number consumption and rejection. They therefore compare prior distributions rather than strictly paired realisations of the same SCM.

The density panels display four settings from each grid: $\delta \in \{-32, -4, 4, 32\}$, $\sigma \in \{0, 0.001, 0.1, 1\}$, and $n \in \{64, 256, 512, 2024\}$. Gaussian KDEs use all finite scores, including

Table 2: Whole-prior settings for Figure 3(a–c). Counts are per intervention value; all studies use one context ordering and one ensemble member.

Intervention	Datasets	Context n	Held-out n'	Group size g
Connectivity	5,000	512	512	20
Internal noise	1,000	512	512	20
Context size	1,000	64–2,024	512	8

negative values and tails, with a common bandwidth within each intervention. The bandwidth is 1.5 times the robust Silverman bandwidth computed from pooled scores and the smallest per-setting sample count. Axis limits restrict only the displayed region; densities are not renormalised. These curves describe variation across datasets rather than confidence intervals.

F.2 Single-mechanism analysis

Mechanisms. Inputs are drawn as $\boldsymbol{x} \sim \mathcal{N}(0, I_5)$ and targets as $y = f(\boldsymbol{x}) + \varepsilon$ with $\varepsilon \sim \mathcal{N}(0, 0.01^2)$, where f is a randomly initialised network with input dimension and hidden width five:

- **Plain MLP:** L hidden layers with ReLU activations and a linear output layer.
- **Residual MLP:** L residual blocks $\boldsymbol{z} \mapsto \boldsymbol{z} + g(\boldsymbol{z})$, where g is a two-layer ReLU MLP, followed by a linear readout.
- **Invertible ResNet** (Behrmann et al., 2019): L residual blocks $\boldsymbol{z} \mapsto \boldsymbol{z} + 0.9\, g(\boldsymbol{z})$, where g is a two-layer ELU MLP with spectrally normalised weights. Since $0.9\, g$ is a contraction, each block is invertible; the blocks are followed by a linear readout.

All weights use PyTorch's default initialisation for linear layers. Depth L counts hidden layers for the plain MLP and residual blocks for the residual architectures, with $L \in \{1, 2, 4, \ldots, 1024\}$.

Estimation. For each architecture and depth, we draw 10 random mechanisms. For each mechanism, we sample 5 datasets of n context and $n' = n$ held-out points, with $n \in \{100, 512\}$, and use group size $g = 20$. Figure 3(d,e) shows the mean over mechanisms with one standard error.

G TEPIG-based tilting

We use estimated TEPIG to sample tasks with higher scores TEPIG more frequently during pretraining. Candidate tasks are drawn from the original prior, $\theta \sim \pi(\theta)$, and assigned a score $s(\theta) = \widehat{\mathrm{TEPIG}}_C(q, \theta, n)$ using a fixed proxy model q, as discussed in Appendix D. We accept each candidate with probability $a_\tau(\theta)$ and train on the accepted tasks, which come from the tilted distribution

$$\tilde{\pi}_\tau(\theta) \propto a_\tau(\theta)\pi(\theta).$$

Rank-based tilting. To reduce sensitivity to the numerical scale of scores and extreme values, we convert each score into a percentile,

$$u(\theta) = \widehat{F}(s(\theta)) \in [0, 1],$$

where $\widehat{F}$ is the empirical cumulative distribution function of scores from a fixed reference sample drawn from the original prior, typically containing 10,000 tasks. For example, $u(\theta) = 0.9$ indicates that the candidate's score is at the 90th percentile of the reference distribution.

We favour higher-TEPIG tasks using the acceptance probability

$$a_\tau(\theta) = \epsilon + (1 - \epsilon)e^{\tau(u(\theta)-1)},$$

with $\tau \geq 0$ and $\epsilon \in (0, 1]$. Candidates with larger percentiles have higher acceptance probabilities. At $\tau = 0$, every candidate is accepted, recovering the original prior. The floor ϵ ensures that every task has acceptance probability at least ϵ, preserving the support of the original prior.

Tilting strength. Because the acceptance floor contributes an untilted component, larger τ need not produce stronger tilting (Table 3). We therefore select the tested setting with the largest estimated $D_{\mathrm{KL}}(\pi(\theta) \,\|\, \tilde{\pi}_\tau(\theta))$, subject to an overall acceptance rate of at least 10%.

Score-based tilting. For the GP proxy in section H, we apply the exponential rule directly to the estimated TEPIG score:

$$a_{\mathrm{GP},\tau}(\theta) = \min\{1, e^{\tau(s(\theta)-s_0)}\}, \qquad s_0 = 35.$$

Candidates with scores at or above s_0 are always accepted. Below this threshold, acceptance decreases exponentially with the score's distance from s_0. This rule depends on the numerical scale of the scores and has no additive acceptance floor.

Tilting towards lower TEPIG. As a control, we also favour lower-TEPIG tasks by reversing the percentile preference in the rank-based rule:

$$a_{-,\tau}(\theta) = \epsilon + (1-\epsilon)e^{-\tau u(\theta)}.$$

Candidates with smaller percentiles then have higher acceptance probabilities, with the same acceptance floor ϵ.

G.1 How strong is tilting?

Table 3: Estimated tilting strength and expected acceptance rates for the GP and SCM prior experiments.

Proxy	Direction	τ	$D_{\mathrm{KL}}(\pi(\theta) \,\|\, \tilde{\pi}_\tau(\theta))$	C2ST	Acceptance (%)
GP prior					
GP	High	0.005	0.00052	0.01225	87.32
GP	High	0.01	0.00210	0.02464	76.32
GP	High	0.02	0.00854	0.04985	58.51
GP	High	0.04	0.03556	0.10219	34.86
GP	High	0.06	0.08367	0.15759	21.21
GP	High	0.08	0.15632	0.21673	13.21
Small CNP	High	2	0.12064	0.21132	46.07
Small CNP	High	8	0.46712	0.43342	16.87
Small CNP	High	12	0.44904	0.43574	12.92
KNN	High	2	0.09853	0.21165	46.07
KNN	High	8	0.42761	0.43314	16.87
KNN	High	12	0.43048	0.43488	12.92
SCM prior					
TabICL v2	High	20	0.16678	0.25292	14.50
KNN	High	20	0.17139	0.24697	14.51
KNN	High	50	0.08626	0.13602	11.81
TabICL v2	Low	10	0.23152	0.31643	18.99

KL is KDE-estimated in nats from the proxy-score distributions under the original and tilted priors. For a fixed score function $s(\theta)$, their KL is $D_{\mathrm{KL}}(\pi(\theta) \,\|\, \tilde{\pi}_\tau(\theta))$, since acceptance depends on θ only through $s(\theta)$. C2ST denotes $2\times$ held-out classification accuracy of a random forest -1 for distinguishing the two score distributions (Lueckmann et al., 2021).

H Gaussian-process prior: experimental details and full results

Setup. We sample tasks from a zero-mean GP with a squared exponential kernel, signal variance 1, observation-noise variance 0.01, and length scale $\lambda \sim \mathrm{Uniform}(0.6, 33)$. Each task contains $n = 100$ context and $m = 100$ query points, with inputs sampled uniformly from $[-3, 3]$. The target CNP has 1,129,346 parameters and is trained for 500,000 updates with batches of 64 accepted tasks. We use Adam with peak learning rate 10^{-3}, 2,000 warmup steps, and cosine decay to 10^{-5}. Architecture and training settings are fixed across tilts.

CNP architecture and optimisation. The CNP encodes each context pair (x_i, y_i), averages its representations over context points, and concatenates the resulting representation with each query input before decoding a Gaussian mean and standard deviation. Encoder and decoder are residual MLPs: an input projection, four residual blocks, and a LayerNorm followed by an output projection. Each block applies LayerNorm, a linear layer, GELU, LayerNorm, and a second linear layer before adding the residual. Table 4 gives the widths. The predictive standard deviation is $\mathrm{softplus}(a)+10^{-4}$ for decoder output a. Training minimises Gaussian NLL averaged over tasks and query points, with gradient norm clipped at 1, default Adam moments, and no weight decay.

Table 4: CNP architectures in the GP experiments. Both encoder and decoder have four residual blocks.

Model	Hidden width	Representation width	Parameters
Target CNP	256	128	1,129,346
Frozen CNP proxy	128	64	286,146

Small CNP proxy pretraining. The frozen proxy is the final seed-0 checkpoint of an untilted run with 500,000 updates, batches of 64 tasks, and 100 context and 100 query points per task. It uses maximal-update parametrisation including a scaled decoder readout, and a Muon/Adam optimiser

with base rates 0.02 and 10^{-3}, respectively, with the same 2,000-step warmup and cosine multiplier decaying to 0.01.

Proxy models. The GP proxy marginalises over length scale using a 20-point grid with posterior-weighted components; saved score diagnostics use 40 points. The small CNP proxy has 286,146 parameters, is pretrained on the original prior, and remains frozen during tilting. The KNN proxy uses squared prediction error in place of log loss. All scores are TEPIG_C estimates (eq. (6)), i.e. the excess loss $\ell_k(q,\theta) - \ell_n(q,\theta)$ accumulated over context sizes $k < n$; the GP score includes the zero-context term $\ell_0(q,\theta)$, while the small CNP and KNN proxies start from one context point ($c = 1$). General estimation procedures are described in Appendix D.

Tilting. We use the tilting rules of section G. The GP proxy uses score-based tilting a_{GP} with $\tau \in \{0.005, 0.01, 0.02, 0.04, 0.06, 0.08\}$. The small CNP and KNN proxies use rank-based tilting towards higher TEPIG, a_+, with floor $\epsilon = 0.05$ and $\tau \in \{2, 8, 12\}$.

Evaluation and intervals. Evaluation always uses the original prior. Learning curves report analytic Gaussian predictive KL, averaged over target points and 2,000 fixed tasks per run. We evaluate every 250 updates and smooth with a centered window of 25 evaluations. The GP comparison uses 20 runs per arm, except 17 at $\tau = 0.08$; the small CNP and KNN proxy comparisons each use three. Shading shows pointwise 95% Student-t intervals across runs.

The predictive-error panels report excess NLL relative to the same GP evaluation reference on 8,000 fresh tasks per run. Its expectation is the predictive KL. Their 95% Student-t intervals describe task-sampling uncertainty conditional on the fitted models. All figures share length-scale bins, with finer resolution in the shortest 5% of the prior; the dotted line separates this region. Bin positions are equally spaced.

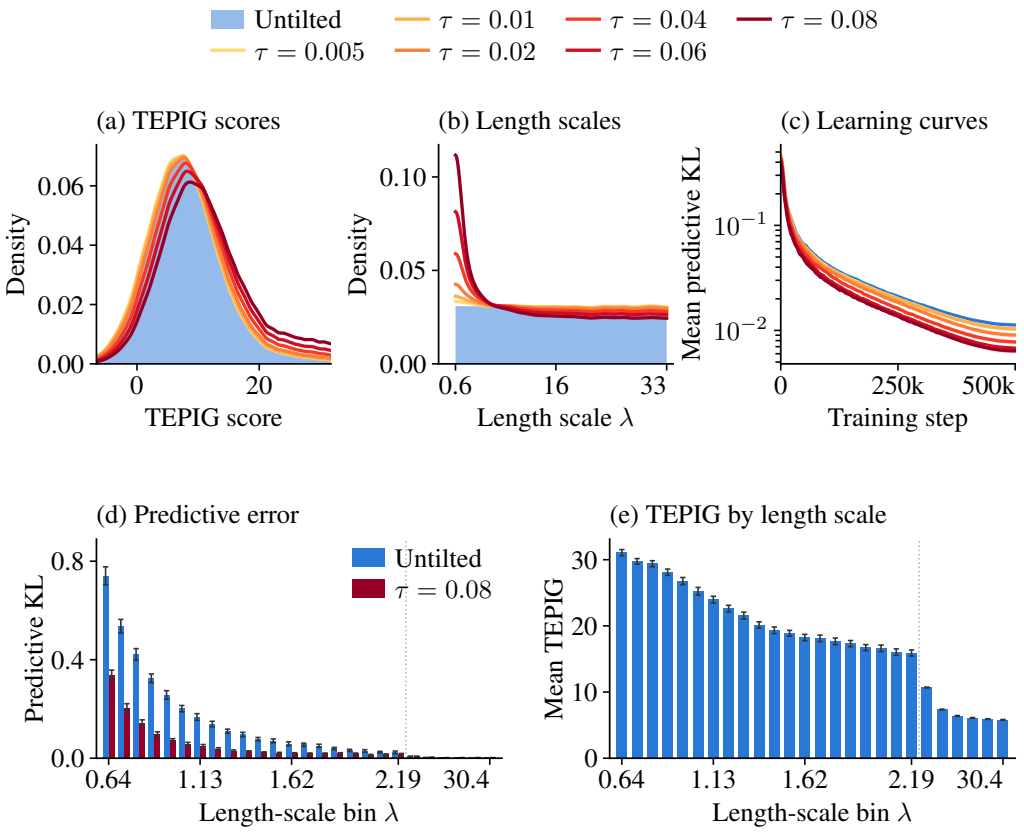


Figure 12: **GP proxy experiment.** (a,b) Score and induced length-scale densities for all six tested tilts. (c) Learning curves on the original prior. (d) Fresh-task predictive error for untilted training and $\tau = 0.08$. (e) Mean GP-estimated TEPIG by length scale. Bands show 95% intervals across training runs; bar intervals describe task-sampling uncertainty.

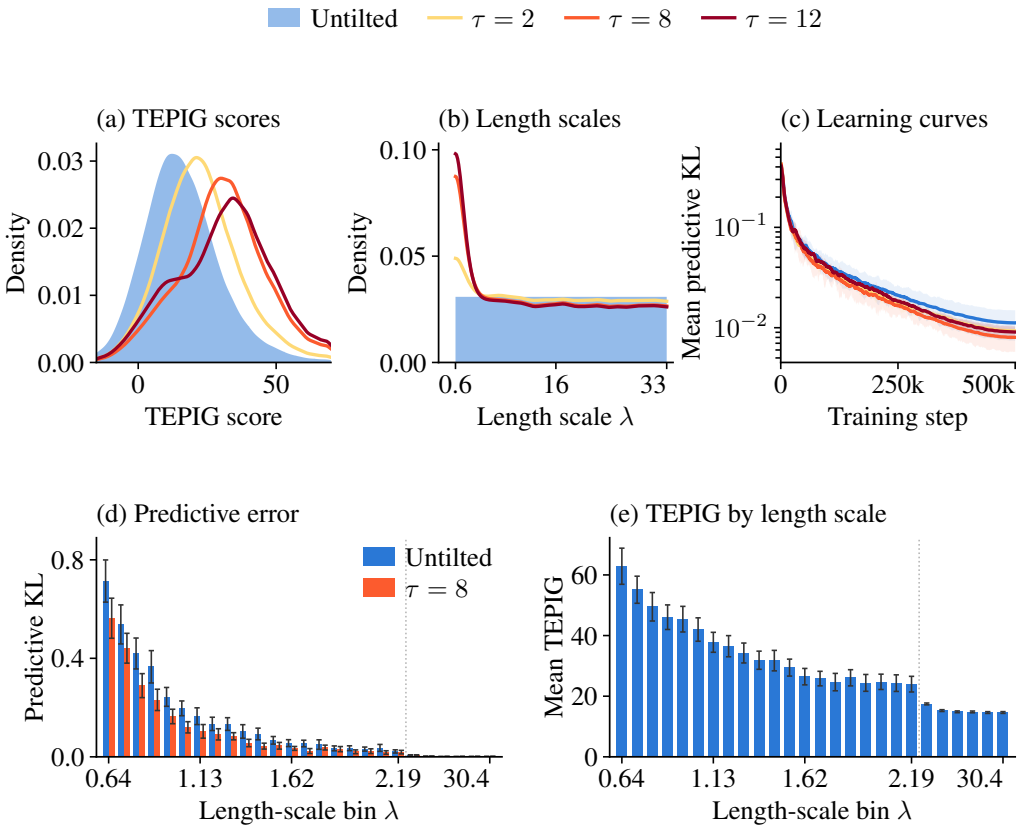


Figure 13: **Small CNP proxy experiment.** Panels follow Figure 12, using rank-based tilting. Panel (d) compares untilted training with $\tau = 8$; panel (e) shows scores from the frozen small CNP. The intermediate strength $\tau = 8$ yields the lowest final mean KL among the tested strengths.

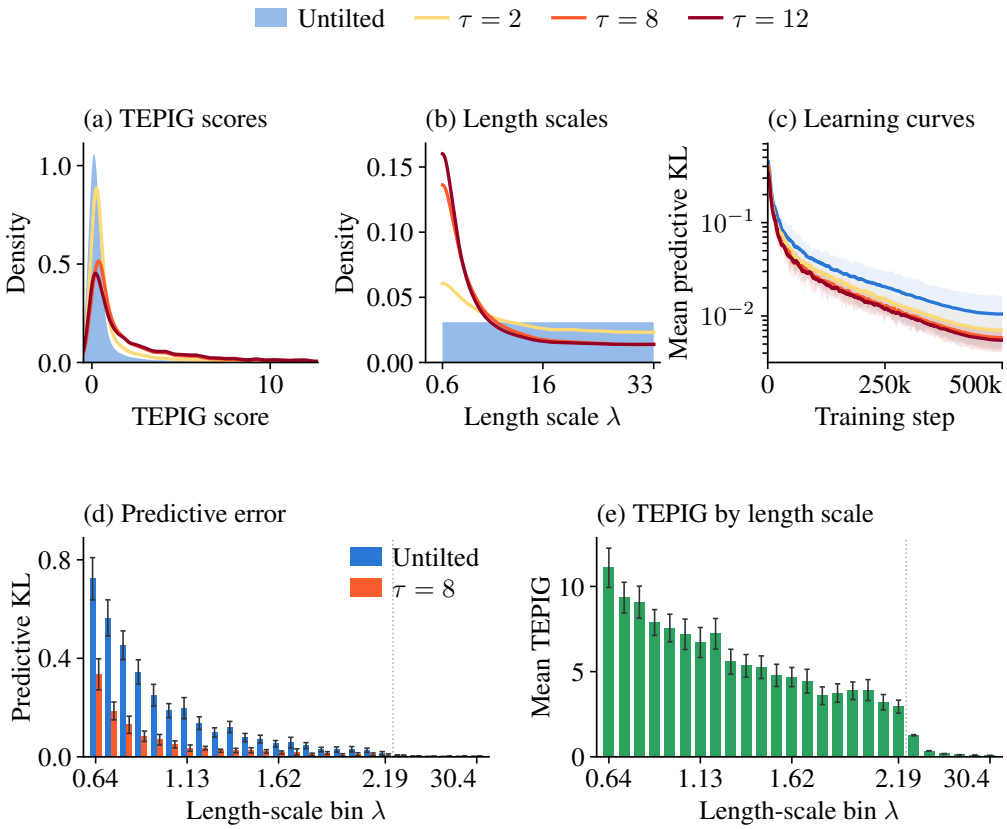


Figure 14: **KNN proxy experiment.** Panels follow Figure 12, using rank-based tilting and squared-error scores. Panel (d) compares untilted training with $\tau = 8$, matching the main-text numerical example. The largest tested $\tau = 12$ yields the lowest final mean KL in panel (c).

I SCM PRIOR: EXPERIMENTAL DETAILS AND ADDITIONAL RESULTS

I.1 PRIOR, TRAINING, AND TILTING

Prior and training setup. We use the SCM regression prior of Reuter et al. (2026) and exactly their training setup: Training tasks contain $n = 512$ context points and $m = 512$ query points, with up to 50 input features. The larger PFN uses six layers, embedding dimension 256, four attention heads for each of the feature and sample attention operations, a feed-forward expansion factor of two, and no dropout. Predictions use a 1,000-bin bar distribution, trained by minimizing query negative log-likelihood. The smaller model uses embedding dimension 128 with the same depth and attention-head counts.

Training runs for 50,000 optimizer steps using Adam with peak learning rate 10^{-4}. The learning rate increases linearly during the first 5,000 steps and then follows a cosine decay to 10^{-5}. We accumulate gradients over four batches of eight tasks, giving an effective batch size of 32 tasks per optimizer step, and clip the gradient norm at 1. Checkpoints are saved every 1,000 steps. Comparisons at matched training steps do not account for differences in model computation or the cost of scoring and rejecting candidate tasks.

Rank-based tilting. We use the rank-based tilting rules a_+ and a_- of section G with floor $\epsilon = 0.1$. Scores and reference values are clipped below at zero before ranking. Accepted tasks populate a buffer used for training.

The main comparison uses high-TEPIG tilting (a_+) with TabICL v2 at $\tau = 20$ and KNN at $\tau \in \{20, 50\}$. The low-TEPIG control (a_-) uses TabICL v2 at $\tau = 10$. Because of the floor, KNN at $\tau = 50$ rejects more candidates than at $\tau = 20$ but shifts the accepted distribution less, as approximately 85% rather than 69% of accepted tasks come from the untilted branch (table 3). The TabICL v2 proxy scores predictive NLL; the KNN proxy uses MSE. Training-task scoring uses $n = m = 512$, one random context ordering, and learning-curve group sizes (section D) of $g = 100$ for TabICL v2 and $g = 20$ for KNN. The KNN implementation uses tricube weights and leave-one-out selection of the neighbour count, capped at 30.

I.2 EVALUATION

Evaluation and comparison. The in-prior evaluation uses a fixed set of 10,000 tasks drawn from the original SCM prior, each with $n = 512$ context points and $m = 512$ query points. Checkpoints are evaluated every 1,000 steps from 1,000 to 50,000 steps. In Figure 5, every plotted method is compared with the same untilted baseline run, at the same training step. OpenML learning curves use 12 regression datasets and 46 shared checkpoints between 1,000 and 49,000 steps. We follow the real-world evaluation setup and benchmark implementation of Reuter et al. (2026).

Reported statistics. In-prior improvement is the difference between the two median task NLLs,

$$\mathrm{median}_j[\mathrm{NLL}_{\mathrm{untilted},j}] - \mathrm{median}_j[\mathrm{NLL}_{\mathrm{tilted},j}]\,.$$

The same definition is applied within each TEPIG group in Figure 5(c). In-prior R^2 improvement is the difference of mean task R^2 values. OpenML R^2 improvement is the mean paired difference across datasets, tilted minus untilted. For the additional OpenML NLL comparison in Figure 16, we instead report the median of paired within-dataset NLL differences, untilted minus tilted. Positive values indicate improvement in every panel.

Uncertainty and display. In-prior intervals use 2,000 paired task-bootstrap resamples with seed 42, applying the same task draws to the baseline and tilted model. The overall curves reuse whole-task trajectories across checkpoints; panel (c) resamples tasks within each fixed decile. OpenML curves use a centred moving average over five available checkpoints, with shorter windows at the endpoints; the corresponding intervals use 4,000 whole-dataset trajectory resamples with seed 42, recomputing smoothing within each draw. In-prior curves and the final-checkpoint estimates are unsmoothed.

All intervals are pointwise 95% percentile intervals. They do not quantify training-seed variability or uncertainty in the estimated TEPIG scores. In Figure 5, panels (a,b) display steps 2,000–50,000 on square-root axes.

Real-world performance on 12 OpenML datasets. We evaluate the untilted baseline and KNN-based high-TEPIG configuration ($\tau = 20$) at 50,000 training steps on 12 OpenML regression datasets. Both models receive identical splits containing 512 context and 512 query points. Table 5 reports per-dataset R^2. High-TEPIG tilting improves performance on 8 of the 12 datasets, increasing mean R^2 from 0.600 to 0.611. The largest gains occur on food delivery time, Miami housing, and houses, while performance decreases on four datasets, most noticeably physiochemical protein. These single-split results suggest a modest average benefit, with improvements varying across datasets.

Dataset	Untilted	High-TEPIG
Airfoil_self_noise	0.840	**0.867**
Another_Dataset_on_used_Fiat_500	**0.819**	0.812
Concrete_compressive_strength	0.835	**0.871**
Diamonds	**0.907**	0.903
Food_Delivery_Time	0.079	**0.158**
Healthcare_insurance_expenses	**0.844**	0.805
Houses	0.599	**0.639**
Miami_housing	0.802	**0.844**
Physiochemical_protein	**0.316**	0.231
QSAR_TID_11	0.080	**0.098**
Superconductivity	0.818	**0.833**
Wine_quality	0.261	**0.270**
Mean	0.600	0.611

Table 5: **Comparison on 12 OpenML datasets.** We report R^2 at 50,000 steps for KNN-based high-TEPIG tilting ($\tau = 20$) relative to the untilted baseline, following the benchmark protocol of Reuter et al. (2026). Both models receive the same 512 context and 512 query points per dataset. Results use one matched split per dataset; the mean row averages R^2 across datasets.

I.3 Tilting towards lower TEPIG

Figure 15 shows the low-TEPIG configuration, which uses the TabICL v2 proxy with rank-based tilting towards lower scores at $\tau = 10$. The high-TEPIG configuration using the same proxy at $\tau = 20$ is shown for context. At 50,000 steps, the low-TEPIG model's median NLL is approximately 1.55 higher than the untilted baseline. At the last shared OpenML checkpoint, its smoothed mean R^2 gain is approximately -0.057, with a pointwise 95% task-bootstrap interval of $[-0.096, -0.022]$.

I.4 Comparison across model sizes

Figure 16 compares KNN-based tilting ($\tau = 20$) for the smaller and larger models. Each tilted model is compared with its own untilted baseline at matched training steps, using the same 10,000 fixed SCM tasks and the same 12 OpenML dataset identities. The larger-model comparison is the KNN configuration from the main text.

At 50,000 steps, in-prior median-NLL reductions are 0.197 for the smaller model and 0.210 for the larger model, with pointwise 95% task-bootstrap intervals of $[0.178, 0.214]$ and $[0.195, 0.227]$, respectively. The corresponding mean R^2 gains are 0.038 and 0.040. On OpenML, the smoothed mean R^2 gain peaks later for the smaller model: approximately 0.208 at 17,000 steps, compared with 0.121 at 9,000 steps for the larger model. By the last common checkpoint at 49,000 steps, these gains are 0.014 ($[-0.018, 0.045]$) and 0.028 ($[0.017, 0.039]$), respectively. Thus, both sizes benefit from tilting, but the relative magnitude of the benefit varies during training. The statistics and bootstrap procedures follow the definitions above.

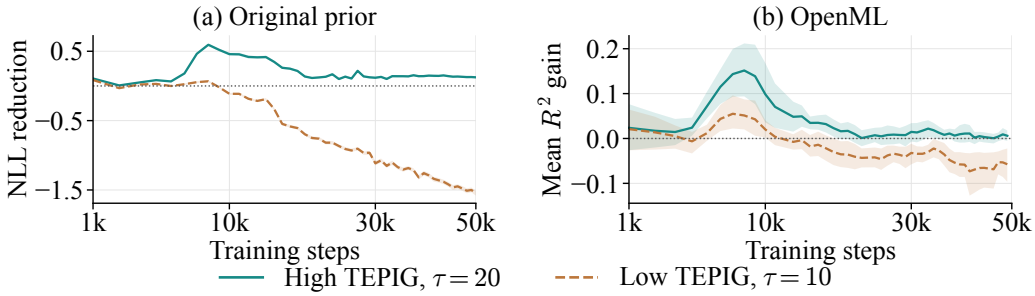


Figure 15: **Tilting towards lower TEPIG degrades later-training performance.** TabICL v2 proxy tilting towards high TEPIG ($\tau = 20$) or low TEPIG ($\tau = 10$), relative to the same untilted baseline. (a) Reduction in median NLL on 10,000 fixed tasks from the original SCM prior; (b) mean R^2 gain on 12 OpenML datasets. Negative values indicate worse performance. Training-step axes use a square-root scale. OpenML curves use a centered five-checkpoint moving average; in-prior curves are unsmoothed. Shading shows pointwise 95% paired task-bootstrap intervals, including their full negative tails. The configurations differ in tilting strength as well as direction.

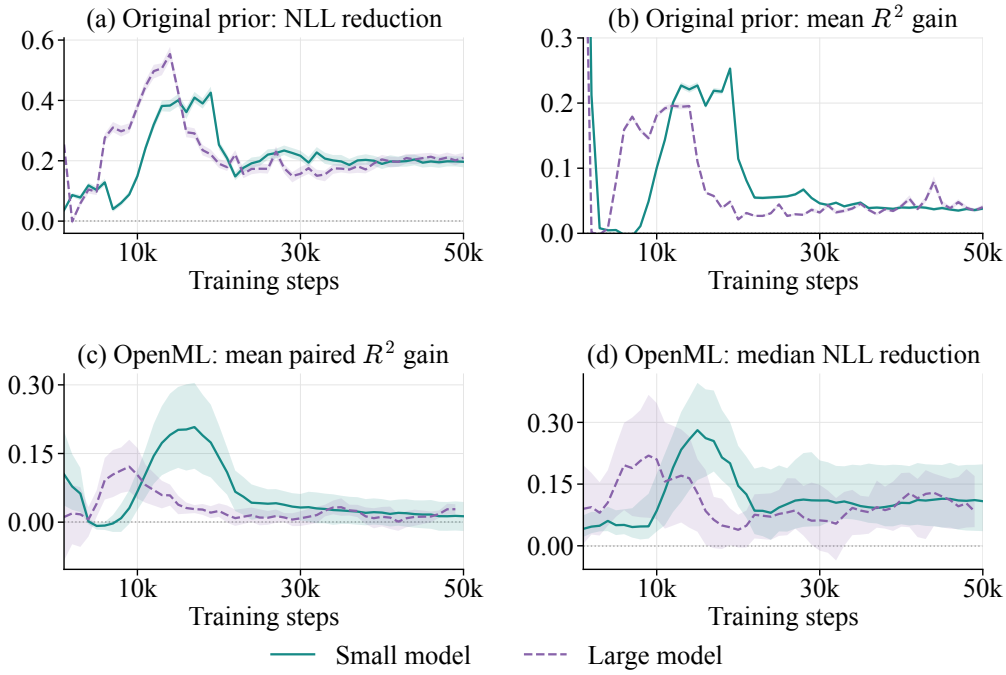


Figure 16: **Benefits of TEPIG-based tilting at two model sizes.** KNN tilting ($\tau = 20$) relative to the untilted baseline of each model size, at matched training steps. On the original SCM prior: (a) reduction in median NLL and (b) mean R^2 gain. On OpenML: (c) mean paired R^2 gain and (d) median paired NLL reduction. Positive values favour tilting. OpenML curves use a centered five-checkpoint moving average; in-prior curves are unsmoothed. Shading shows pointwise 95% paired task-bootstrap intervals for each within-size gain. Panel (b) displays only gains between 0 and 0.3; larger early gains are outside the displayed range.

I.5 ABSOLUTE LEARNING CURVES

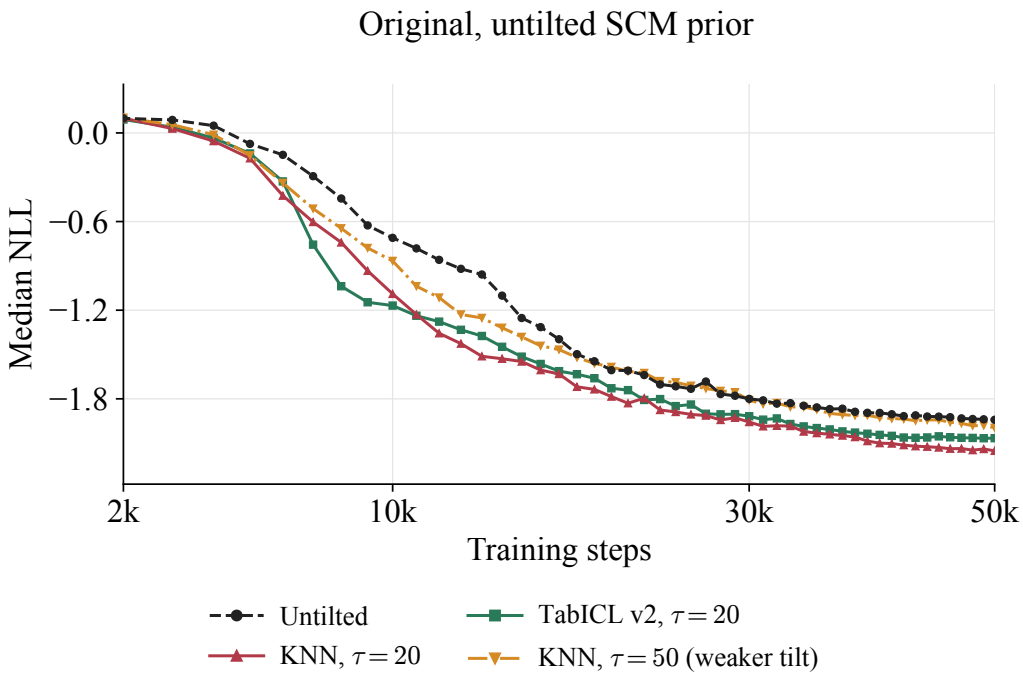


Figure 17: Median NLL on 10,000 fixed tasks from the original SCM prior. Curves are unsmoothed; the training-step axis uses a square-root scale.

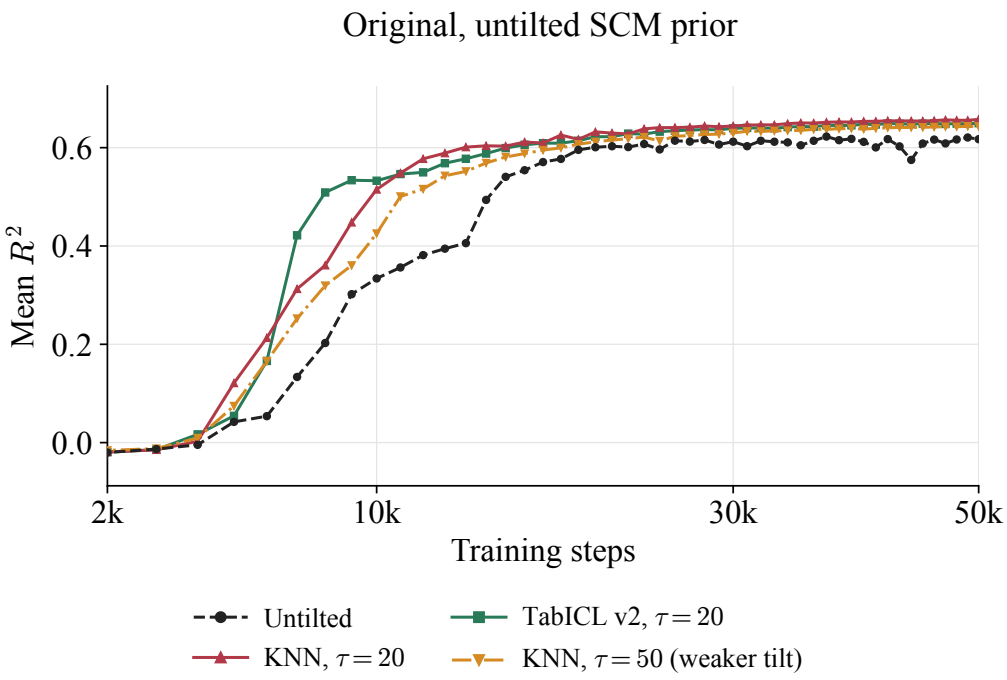


Figure 18: Mean R^2 on 10,000 fixed tasks from the original SCM prior. Curves are unsmoothed; the training-step axis uses a square-root scale.

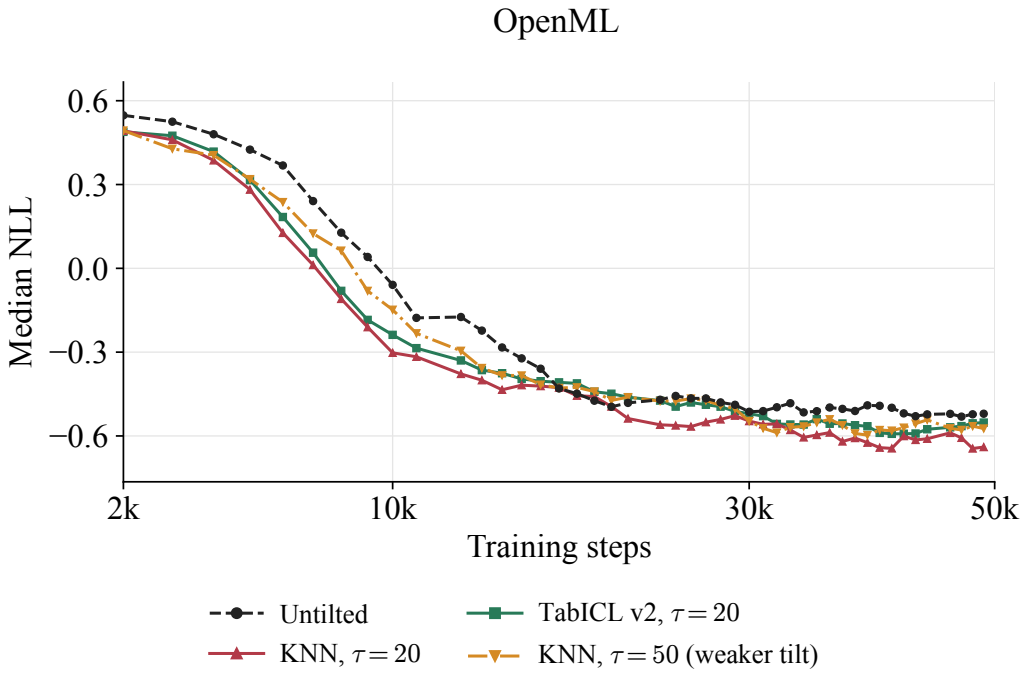


Figure 19: Median NLL on 12 OpenML datasets. Curves use a centered five-checkpoint moving average and a square-root training-step axis.

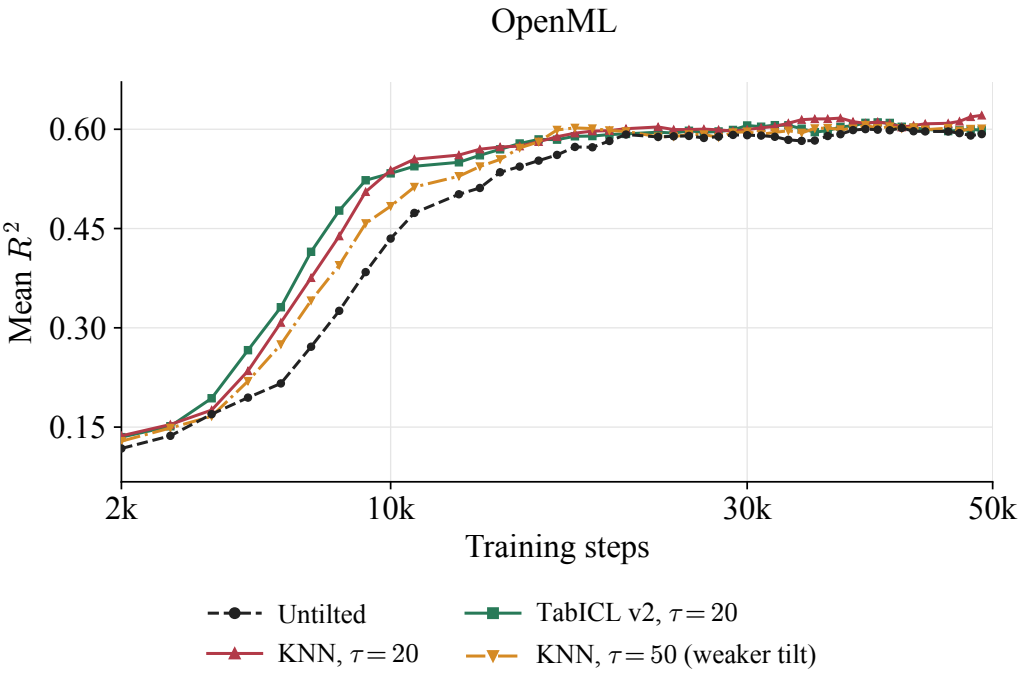


Figure 20: Mean R^2 on 12 OpenML datasets. Curves use a centered five-checkpoint moving average and a square-root training-step axis.

J TABICL V2 PRIOR

J.1 PRIOR, TRAINING, AND TILTING

Prior and training setup. We use the TabICL v2 (Qu et al., 2026) regression prior and largely follow their stage-1 training setup. In contrast to the original setting, we pretrain on 1024 training context points and 512 query points. We train on up to 100 features. Predictions use 999 quantiles, trained by minimizing average pinball loss.

Architecturally, TabICL v2 chains three separate Transformers (Vaswani et al., 2017). We copy this exactly and keep the column-wise embedding Transformer (TF_{col}), row-wise interaction Transformer (TF_{row}), and dataset-wise in-context learning Transformer (TF_{icl}) as is. For our experiments on a small TabICL version, we reduce the embedding dimension and the number of attention heads by half. Thus, TF_{col} and TF_{row} have 3 layers with 4 heads each of dimension 64, and TF_{icl} contains 12 layers with 4 heads at size 256. This yields a model with 7.6 million parameters. Our experiments on the standard TabICL v2 model follow the original setup.

We train for 20,000 training steps using Muon optimizer with cosine scheduling and peak learning rate of $1.1 \cdot 10^{-3}$. We accumulate gradients over four batches of sixteen tasks with an effective batch size of 64. Similar to the original setting, we train on 4 parallel H100 GPUs with 80GB memory. Gradient norms are clipped at 10 and checkpoints saved every 1,000 steps.

Rank-based tilting. The definition of the acceptance probability for this setting is identical to the SCM prior in Appendix I.

Our standard experiments run on $\tau = 2.5$ using a KNN proxy. Training-task scoring uses context size of 1024, query size 512, and learning-curve group sizes of 20. The rest is similar to above.

J.2 EVALUATION

Evaluation and comparison. The in-prior evaluation uses a fixed set of 1,000 tasks drawn from the original TabICL v2 prior. Checkpoints are evaluated at every 1,000 steps from 1,000 to 20,000 steps. Throughout, we compare every method with the untilted baseline run at the corresponding checkpoint. The stratified experiments use 500 datasets per decile, grouped by estimated TEPIG. To obtain an intuition about the dataset, the lowest decile contains all datasets with estimated TEPIG less than -2. The top decile, in contrast, contains all datasets with estimated TEPIG of at least 150. We provide more detailed results below.

Reported statistics. Similar to the original setup, we train the TabICL v2 prior on average pinball loss over 999 quantiles. Therefore, we report our results under pinball loss over 1,000 fixed datasets drawn from the original TabICL pretraining mix. To compare the differences, we report median pinball reduction with 95% percentile intervals. Similarly, we report median pinball reduction for the stratified per-TEPIG decile results with 95% percentile intervals. We note that the percentile intervals representing the 2.5% and 97.5% quantile do not yield similar guarantees as other techniques such as the basic bootstrap intervals (Efron, 1979). We do, however, find them intuitive in our discussion of 50%-quantile (median) improvement.

Top-decile TEPIG performance. To study the training behaviour in the top TEPIG decile, we replicate our stratified plot on the ten highest percentiles between 0.9 and 1.0. Figure 21 shows that our above results hold more generally. Evaluating our final checkpoint on 500 datasets per percentile, we observe significantly different performance for each of the high-TEPIG and low-TEPIG configurations. We report both mean and median.

Additional results. Finally, we also report stratified learning curves. Evaluating each of the 20 checkpoints on each of the 10 deciles, we observe improved performance at a fixed training budget across realisations of estimated TEPIG. Figure 22 shows that our general trend of substantial early training improvements occurs for most levels of structure. On top of that, the pattern is more pronounced as we increase dataset structure according to the estimated TEPIG.

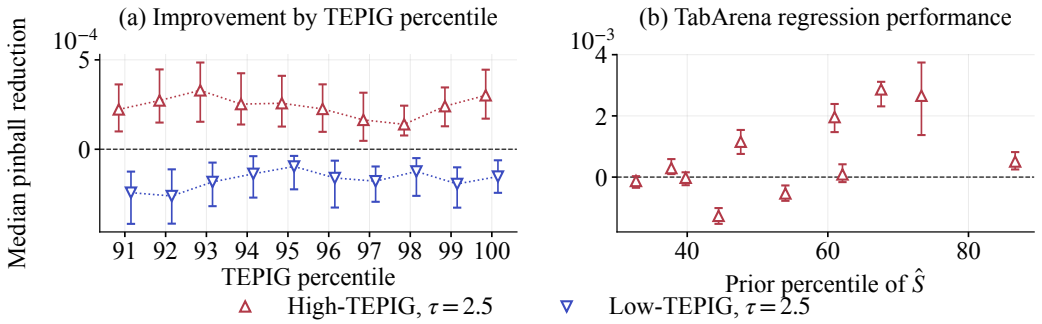


Figure 21: **Improvement over original prior and real-world data.** We present (a) median pinball reduction over 500 datasets per TEPIG percentile between 0.9 and 1.0; (b) median pinball reduction over 100 random train/test splits of 11 TabArena regression datasets, ordered by estimated TEPIG.

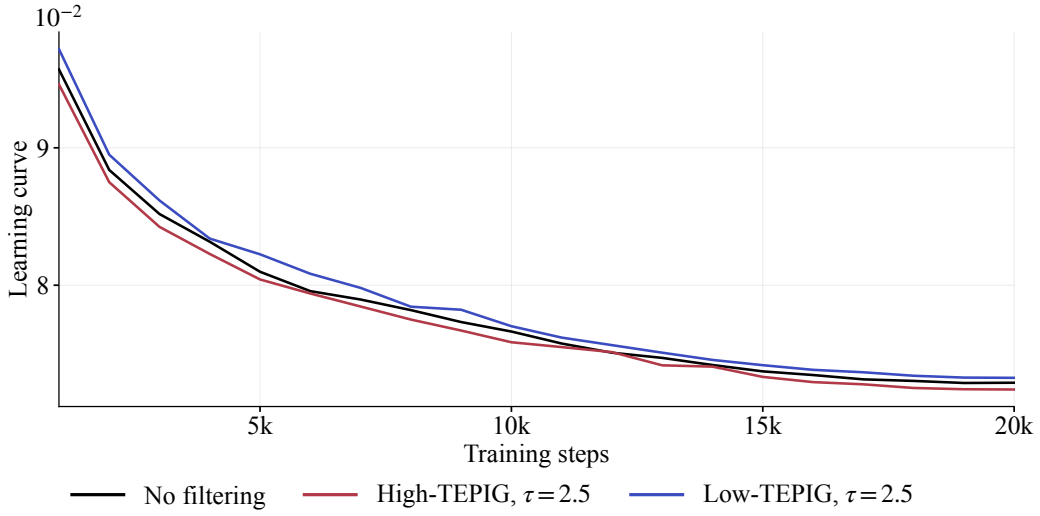
J.3 Real-world performance on 11 TabArena datasets

To conclude, we evaluate our high-TEPIG configuration on 11 TabArena regression datasets. As we train our setup on a fixed context length of 1024 data points, we drop `QSAR_fish_toxicity`, which is shorter in total length. Further, we only train on up to 100 features such that we exclude `QSAR_TID_11`. All other of the 13 total datasets we include. To accommodate for our setup with 1024 context points, we sample 100 train/test splits and evaluate the median pinball reduction over each of them. Our query set is restricted to at most 512 observations from the held out set not included in the 1024 context points. We check that using a flexible number of query examples yields similar results. For comparability across datasets, we standardise the target to zero mean and unit variance. Based on this, we then report percentile intervals of median pinball reduction.

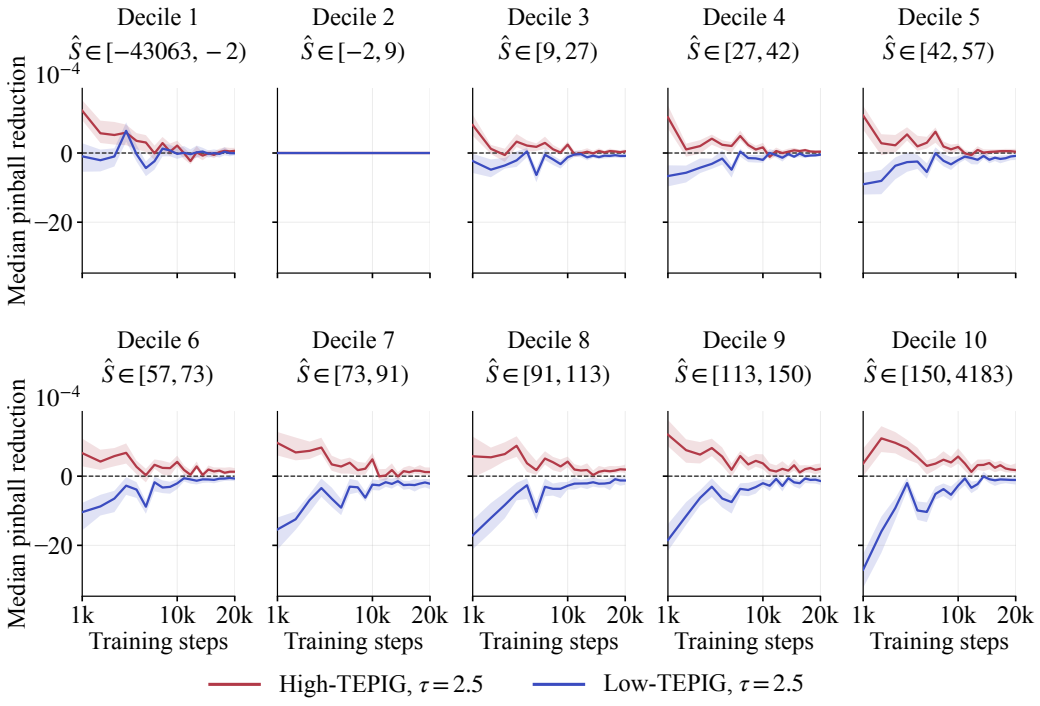
Dataset	$\hat{S}$ percentile	Untilted	High-TEPIG
Another_Dataset_on_used_Fiat_500	33	**0.858 ± 0.001**	**0.857 ± 0.001**
Diamonds	38	**0.973 ± 0.001**	0.969 ± 0.001
Food_Delivery_Time	40	**0.264 ± 0.003**	0.259 ± 0.003
Healthcare_insurance_expenses	44	**0.861 ± 0.003**	**0.860 ± 0.003**
Wine_quality	48	0.381 ± 0.004	**0.385 ± 0.004**
Superconductivity	54	**0.856 ± 0.002**	**0.855 ± 0.002**
Physiochemical_protein	61	0.494 ± 0.004	**0.503 ± 0.004**
Miami_housing	62	**0.899 ± 0.002**	0.895 ± 0.002
Houses	68	0.807 ± 0.002	**0.812 ± 0.002**
Concrete_compressive_strength	73	0.903 ± 0.018	**0.921 ± 0.012**
Airfoil_self_noise	87	**0.972 ± 0.001**	**0.972 ± 0.001**
Mean		0.752	0.754

Table 6: **Comparison on 11 TabArena datasets.** We report mean R^2 and one standard error over 11 TabArena regression datasets for high-TEPIG tilting relative to the untilted baseline.

Figure 21 displays the results. We order the datasets by TEPIG percentile on the x-axis. Generally, we find that real-world performance is better on high-structure datasets. *Cum grano salis*, the effect on medium-TEPIG datasets between the 30% quantile and the 70% quantile is ambiguous. Reporting the per-dataset mean R^2 over 100 random splits, we find that high median pinball reduction does not immediately translate into improved R^2. Table 6 represents the results. We take this as evidence that our method yields performance increase in the high-TEPIG region, but observe inconclusive results at lower structure.



(a) **Learning curve on original prior.** We present the pinball loss over 1,000 held-out tasks drawn from the original prior. The y-axis is log-scaled.



(b) **Stratified improvement.** We present the stratified median pinball reduction curves over 500 datasets per estimated TEPIG decile.

Figure 22: **Improvement over original prior.**