A STATISTICAL-PHYSICS-INSPIRED FRAMEWORK OF MACHINE LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

While the statistical and probabilistic aspects of neural networks (NNs) have been extensively studied, conventional NN training typically seeks a single optimal configuration of internal parameters, which is fundamentally underdetermined when the number of parameters exceeds the number of training samples. In this paper, we investigate a framework of statistical-physics-inspired machine learning (SML). Predictions are obtained as expectation values by integrating over all possible parameter configurations, rather than relying on the optimal parameter configuration. Such an algorithm allows regular system-level behavior to emerge naturally, without being overwhelmed by the underlying irregular parameter-space landscape. The number of unknowns scales with the training-data size, not the NN size, making the approach attractive to the high-dimensional low-sample-size regime. Realized in closed form on a power-law network (PN),[1] SML ranks first on standardized function benchmarks against 19 high-performance reference model variants, and outperforms most of them on 848 real-world datasets. SML-PN remains quite robust to label noise, irrelevant features, and contamination.

Keywords: Ensemble average; expectation value; maximum entropy; tabular regression; small data; high-dimensional low-sample-size

1 INTRODUCTION

Parametric models such as neural networks (NNs) represent input-output relationships through internal parameters $\{w_i\}_{i=1}^{m}$, conventionally trained to find a single optimum configuration $\{w_i\}_{\mathrm{o}}$ by minimizing a loss (Goodfellow et al., 2016; Murphy, 2012). When the number of internal parameters m exceeds the training-sample number n (Chowdhery et al., 2022), $\{w_i\}_{\mathrm{o}}$ is fundamentally underdetermined: multiple parameter configurations may fit the training data equally well, and choosing one of them may discard information from the others. The difficulty is sharpest in the high-dimensional low-sample-size (HDLSS) regime, where $n \leq l$, with l denoting the input dimension (Hall et al., 2005). Held-out data are especially costly there: every training sample spent on validating a hyperparameter choice is a sample not spent on learning.

Statistical physics may offer inspiration (Kardar, 2007). For example, an ideal gas has astronomically many parameter configurations (positions and momenta of all gas particles), yet a few measurements allow macroscopic observables (e.g., pressure) to be computed as ensemble averages. Regular system-level behavior emerges without resolving a specific configuration through chaotic microscopic dynamics (Supplementary Material (SM), SM Section 1; Anonymous, 2026).

In this paper, we investigate a framework of statistical-physics-inspired machine learning (SML), which similarly replaces the search for $\{w_i\}_{\mathrm{o}}$ with the analysis of the maximum-entropy probability distribution of $\{w_i\}$ (ρ_μ). Constrained by the training data (Jaynes, 1957a;b), ρ_μ predicts the output as the expectation value over all possible parameter configurations. Section 2 below briefly reviews the state of the art. Sections 3–4 develop SML and its closed-form power-law network (PN) realization. Sections 5–6 present benchmarks, robustness, and interpretability.

[1] Code, frozen dataset lists, and the figure pipeline are available in the supplementary code archive.

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.

2 RELATED WORK

SML averages over all possible parameter configurations, placing it among methods that combine many configurations rather than committing to a single one; the distinction from the conventional approaches lies in how ρ_μ is determined and utilized. Bayesian NNs average over a posterior constructed from a prior and likelihood (MacKay, 1992; Neal, 1996); deep ensembles and stochastic-gradient samplers average predictions from independently obtained point estimates (Lakshminarayanan et al., 2017; Welling & Teh, 2011; Srivastava et al., 2014). Energy-based models use exponential-form probabilities but still train toward a single internal-parameter configuration, e.g., by Markov-chain sampling (Hinton, 2002; LeCun et al., 2006; Hopfield, 1982). Kernel methods require a chosen kernel and hyperparameters (Schölkopf & Smola, 2002).

In contrast, SML introduces neither prior, likelihood, nor kernel, and does not seek any specific $\{w_i\}_{\mathrm{o}}$. The training-data constraints uniquely determine ρ_μ under the principle of maximum entropy. The number of unknowns depends on the amount of available data n, not the NN size m, making SML particularly attractive for HDLSS problems. This work advances the line of thinking of the early discussions of Boltzmann ensembles (Seung et al., 1992; Neal, 1996).

3 THE FRAMEWORK OF SML: ENSEMBLE AVERAGE AT MAXIMUM ENTROPY

In general, an NN can be formulated as

$$\widehat{y} = g\left(\mathbf{w}, \mathbf{x}\right), \tag{1}$$

where $\widehat{y}$ is the output, g is the architecture, $\mathbf{w} \triangleq \{w_i\}_{i=1}^{m}$ are the internal parameters, and $\mathbf{x} \triangleq \{x_k\}_{k=1}^{l}$ is the input. The probability ρ_μ of the μ-th possible parameter configuration $\mathbf{w}_\mu$ satisfies

$$\int \rho_\mu \, \mathrm{d}\Omega = 1, \tag{2}$$

where $\mathrm{d}\Omega \triangleq \mathrm{d}w_1 \mathrm{d}w_2 \ldots \mathrm{d}w_m$, integrating over all possible $\mathbf{w}$. Each training pair $(\mathbf{x}_j, \overline{y}_j)$ imposes

$$\int \rho_\mu \widehat{y}_{\mu j} \, \mathrm{d}\Omega = \overline{y}_j \qquad (j = 1, 2, \ldots, n), \tag{3}$$

where $\widehat{y}_{\mu j} = g(\mathbf{w}_\mu, \mathbf{x}_j)$, and $\mathbf{x}_j$ and $\overline{y}_j$ are the j-th input and output, respectively. Equation (3) treats measurements as expectation values: repeated observations under nominally identical conditions fluctuate and converge to $\overline{y}_j$. If $\rho_\mu \to \delta_{\overline{\mu}}$, where $\delta_{\overline{\mu}}$ is a delta function, Eq. (3) is reduced to the conventional optimum-configuration approach.

As in statistical physics, SML maximizes entropy $S = -\int \rho_\mu \ln \rho_\mu \, \mathrm{d}\Omega$ to find the "equilibrium" ρ_μ, which is not necessarily equivalent to the optimum $\{w_i\}_{\mathrm{o}}$. Under the constraints of Eqs. (2,3), the maximum S is obtained through stationarity of the Lagrangian (Kardar, 2007):

$$\frac{\partial \mathscr{L}}{\partial \rho_\mu} = 0, \tag{4}$$

$$\mathscr{L} = -\int \rho_\mu \ln \rho_\mu \, \mathrm{d}\Omega + \alpha_0 \left(\int \rho_\mu \, \mathrm{d}\Omega - 1 \right) + \sum_{j=1}^{n} \beta_j \left(\overline{y}_j - \int \rho_\mu \widehat{y}_{\mu j} \, \mathrm{d}\Omega \right),$$

where α_0 and β_j are Lagrange multipliers. Equation (4) gives $\rho_\mu = e^{\alpha_0 - \sum_j \beta_j \widehat{y}_{\mu j} - 1}$. With the partition function $Z = \int e^{-\sum_j \beta_j \widehat{y}_{\mu j}} \, \mathrm{d}\Omega$, normalization yields $Z = e^{1-\alpha_0}$, or $\alpha_0 = 1 - \ln Z$. Thus,

$$Z = \int e^{-\widehat{E}_\mu} \, \mathrm{d}\Omega, \tag{5}$$

$$\rho_\mu = \frac{1}{Z} e^{-\widehat{E}_\mu} = \frac{1}{Z} \prod_{j=1}^{n} e^{-\beta_j \widehat{y}_{\mu j}}, \tag{6}$$

where $\widehat{E}_\mu = \sum_j \beta_j \widehat{y}_{\mu j}$. Unlike energy-based configuration training (e.g., Hinton, 2002), the Boltzmannian ρ_μ remains the object of integration; unlike a Bayesian NN posterior (e.g., MacKay, 1992;

Neal, 1996), ρ_μ uses neither likelihood nor prior. The probability distribution in Eq. (6) is constrained by the training samples, not assembled from independently trained probabilistic experts. Equations (5)–(6) yield

$$S = -\int \rho_\mu \ln \rho_\mu \, \mathrm{d}\Omega = \ln Z + \sum_j \beta_j \overline{y}_j. \tag{7}$$

For a new input $\mathbf{x}_*$, prediction is the expected output,

$$y_* = \int \rho_\mu \widehat{y}_{\mu *} \, \mathrm{d}\Omega, \tag{8}$$

where $\widehat{y}_{\mu *} = g(\mathbf{w}_\mu, \mathbf{x}_*)$.

4 Power-law network (PN): an embodiment of SML

Through Eq. (6), ρ_μ is determined by n unknowns β_j, one per training sample. In statistical physics, the Lagrange multiplier is the inverse temperature, usually measured rather than calculated, as entailed by the zeroth law (Kardar, 2007). For SML, no direct measurement of β_j (e.g., by a known "thermometer" NN) has been formulated; β_j are computed from Eq. (3).

In principle, SML can be applied to a wide variety of NNs as long as the parameter-space integrals can be evaluated analytically or numerically, e.g., by phase-space quadrature. Here, we demonstrate SML with a simple power-law network (PN): $g(\mathbf{w}, \mathbf{x}) = g_1[g_2(\mathbf{w}, \mathbf{x})]$, where the activation function is $g_1(\cdot) = -\ln(\cdot)$, $g_2 = \prod_{k=1}^{l} x_k^{f_k(\mathbf{w})}$, $f_k(\mathbf{w}) = \sum_{i=1}^{m} \alpha_{ki} w_i$, and $\alpha_{ki} \geq 0$ are given coefficients. The multiplicative feature interaction in g_2 is somewhat similar to the product-of-experts model (Hinton, 1999). Combining g_1 and g_2 gives $\widehat{y}_{\mu j} = -\sum_{k=1}^{l} f_k(\mathbf{w}_\mu) \ln x_{jk}$, and Eq. (6) becomes

$$\rho_\mu = \frac{1}{Z} \left[\prod_{j=1}^{n} \prod_{k=1}^{l} x_{jk}^{\beta_j f_k(\mathbf{w}_\mu)} \right] = \frac{1}{Z} \left[\prod_{i=1}^{m} B_i^{w_{\mu i}} \right], \tag{9}$$

where $\{x_{jk}\}_{k=1}^{l}$ and $\{w_{\mu i}\}_{i=1}^{m}$ are the components of $\mathbf{x}_j$ and $\mathbf{w}_\mu$, respectively, $B_i = \prod_{j=1}^{n} \widetilde{x}_{ji}^{\beta_j}$, and $\widetilde{x}_{ji} = \prod_{k=1}^{l} x_{jk}^{\alpha_{ki}}$. The partition function is

$$Z = \int \left[\prod_{i=1}^{m} B_i^{w_i} \right] \mathrm{d}\Omega = \prod_{i=1}^{m} z_i, \tag{10}$$

$$z_i = \begin{cases} \dfrac{B_i^{\widehat{w}_i} - B_i^{\check{w}_i}}{\ln B_i}, & B_i \neq 1, \\ \widehat{w}_i - \check{w}_i, & B_i = 1. \end{cases}$$

where $\check{w}_i$ and $\widehat{w}_i$ are the lower and upper bounds of w_i, respectively. Substituting Eqs. (9)–(10) into Eq. (3) gives

$$\overline{y}_j = \sum_{k=1}^{l} \varepsilon_k \ln x_{jk} \qquad (j = 1, 2, \ldots, n), \tag{11}$$

where the interconnection factors are, for $k = 1, 2, \ldots, l$,

$$\varepsilon_k = -\sum_{i=1}^{m} \alpha_{ki} \frac{(\widehat{w}_i \ln B_i - 1) B_i^{\widehat{w}_i} - (\check{w}_i \ln B_i - 1) B_i^{\check{w}_i}}{(B_i^{\widehat{w}_i} - B_i^{\check{w}_i}) \ln B_i}.$$

For a new input $\mathbf{x}_*$, Eq. (8) predicts

$$y_* = \sum_{k=1}^{l} \varepsilon_k \ln x_{*k}, \tag{12}$$

where x_{*k} is the k-th feature of $\mathbf{x}_*$. Solving Eq. (11) for β_j determines ε_k in Eq. (12). Equation (11) is linear in $\ln x_{jk}$, while highly nonlinear in β_j and ρ_μ, which encode the training data.

The details of numerical implementation are given in SM Secs. 2–3. The numerical realization adds an intercept and a scale factor to accommodate the offset and scale of the target (SM Sec. 3.3). Equation (11) is solved by minimizing the training mean square error (MSE) with full-batch Adam (Kingma & Ba, 2015) on a fixed hierarchical coarse-to-fine schedule. Validation selects stopping and the retained checkpoint.

5 SMALL-DATA CASE STUDIES

SML-PN is evaluated in HDLSS ($n \leq l$), where $\{w_i\}_o$ is fundamentally underdetermined and never calculated, at three levels of increasing validity: model-synthesized data (Sec. 5.1), data synthesized by 13 standardized benchmark functions (Sec. 5.2), and 848 real-world regression datasets (Sec. 5.3). Two internal controls bracket the maximum-entropy contribution: feed-forward $\mathrm{NN_m}$ matches the trainable-parameter count m (SM Sec. 2.1), and log-linear ridge (LLR) shares SML-PN's prediction basis (Eq. 12).

Eight more high-performance reference model families are compared: XGBoost, CatBoost, random forests, a freely sized MLP, SVR, a Gaussian process, ridge, and TabPFN (Chen & Guestrin, 2016; Prokhorenkova et al., 2018; Breiman, 2001; Pedregosa et al., 2011; Smola & Schölkopf, 2004; Rasmussen & Williams, 2006; Hoerl & Kennard, 1970; Hollmann et al., 2025). Each model has a library-default fixed configuration and a per-run tuned configuration, yielding a total of 20 model variants. The model variants are ranked from 1 to 20 by cell test error. SML-PN uses a fixed configuration, with no search for architecture, learning-rate, or regularization; $\widehat{w}_i$ and $\check{w}_i$ are adapted with β_j during numerical solution of Eq. (11). SM Sec. 2 presents all the model variants and their search spaces; SM Secs. 3.1–3.2 present the protocols and data scaling; SM Secs. 3.3–3.5 present the auxiliary variables, the bounded-logarithm surrogate, the solver details, and the training procedures.

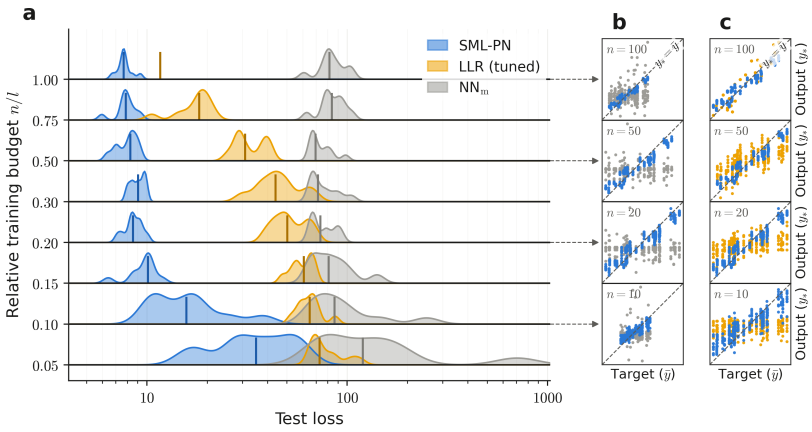

Figure 1: SML-PN greatly outperforms $\mathrm{NN_m}$ and log-linear ridge (LLR) on the synthesized data ($l = 100$). (**a**) Test-loss distributions (kernel-density estimates) over ten repetitions at each training budget n/l; vertical ticks mark medians. (**b**) Comparison between prediction (y_*) and true target ($\bar{y}$), pooled over repetitions: SML-PN (blue) vs. $\mathrm{NN_m}$ (grey) and (**c**) SML-PN (blue) vs. LLR (orange), at $n/l = 0.1, 0.2, 0.5, 1.0$. Dashed diagonals denote perfect prediction.

5.1 NETWORK-SYNTHESIZED DATASETS

Noise-free training samples ($l = 100$) are generated by the PN equation (see SM Sec. 4.1 for details). The goal is to evaluate SML-PN when the optimum parameter configuration $\{w_i\}_o$, i.e., the ground-truth parameter vector $\overline{\mathbf{w}} \triangleq \{\overline{w}_k\}$, is guaranteed to exist. This condition is assumed by most conventional machine-learning methods and is not unique to SML-PN. We draw $\overline{\mathbf{w}} \in \mathbb{R}^m$ from $\mathcal{N}(-2, 1)$ and draw inputs $\mathbf{x} \in \mathbb{R}^l$ uniformly from $[0.25, 1]^l$. After calculating the true target $\bar{y} = \sum_k \overline{w}_k \log x_k$, $\overline{\mathbf{w}}$ is discarded and never accessed by any model. SML-PN is compared to $\mathrm{NN_m}$ and to tuned log-linear ridge (LLR; SM Sec. 2.2). The training, validation, and test datasets are separated, and all models receive identical data. With only $n = 10$ training samples, SML-PN predicts the target quite well, whereas $\mathrm{NN_m}$ gives no meaningful predictions even at $n = 100$, and LLR approaches SML-PN only at $n = l$ (Fig. 1). SML-PN's test loss is typically over an order of magnitude lower, with a narrower, more stable distribution from $n/l = 0.10$ onward. Its predictions approach the identity line from $n = 10$, while the outputs of $\mathrm{NN_m}$ remain widely scattered. The high convergence rate of SML-PN toward the true target is somewhat analogous to statistical physics, which predicts the macrostates of complex systems from a small number of high-quality measurements (SM Sec. 4).

5.2 STANDARDIZED BENCHMARK FUNCTIONS

A more demanding test should not use data generated by PN, for which we evaluate all 13 continuous-input functions with adjustable l in the Virtual Library of Simulation Experiments (VLSE) (Surjanovic & Bingham, 2013). VLSE has been widely used in studies on optimization and regression (Fig. 2a). Each VLSE function is tested at $l = 20, 50, 100, 500$, forming 52 function–dimension units, with training budgets from $0.1l$ to l and ten independent repetitions per setting (SM Sec. 5). Nine VLSE functions are learnable and form the analysis set. Paired Wilcoxon signed-rank tests (Wilcoxon, 1945) with Holm correction (Holm, 1979) rank SML-PN significantly above all 19 reference model variants, with no ties (largest corrected $p = 1.7 \times 10^{-5}$; Fig. 2). On the other four VLSE functions (Sum Powers, Dixon-Price, Rosenbrock, and Rastrigin), no model variant achieves a median test $R^2 \geq 0.2$; they are effectively unlearnable in this HDLSS regime and are excluded from the analysis, since ranking them is uninformative (SM Sec. 5.1).

SML-PN is the best-performing model on 92% of the analysis-set units (33 of 36; Fig. 2a,b). The exception is Griewank, where tuned TabPFN leads despite its relatively low overall rank (Fig. 2c). Griewank's first-order importance is concentrated in a few coordinates, a setting unfavorable to SML-PN; this result reflects a limitation of PN rather than the nature of SML (SM Sec. 5). The advantage of SML-PN grows with input dimension and data scarcity (Fig. 2c,d). As l increases from 20 to 500, tuned boosting models, including XGBoost and CatBoost, deteriorate rapidly, while SML-PN remains consistently better. Reference model variants such as ridge or LLR approach SML-PN only at $n/l \geq 0.9$. LLR remains worse throughout the training-budget range despite sharing SML-PN's basis (Eqs. 11–12), suggesting that the high performance of SML-PN comes from the algorithm rather than the functional form.

At $n/l = 0.1$, the median R^2 of SML-PN rises from -0.11 at $l = 20$ to $0.58, 0.74, 0.91$ at $l = 50, 100, 500$, respectively (Fig. 2e); the absolute sample count also grows with l along this trajectory. At $l = 500$ and $n = 100$, it reaches a median $R^2 = 0.93$, versus 0.13 for the second-best model variant, SVR; all other reference model variants are at or below zero (Fig. 2f). To reach a median $R^2 = 0.5$, SML-PN needs only $n = 4, 5, 10, 50$ at the four input dimensions, versus 12–350 for ridge and LLR, close to their full training budgets (Fig. 2g).

5.3 REAL-WORLD DATASETS

Unlike the noise-free, well-specified synthetic cases in Secs. 5.1 and 5.2, real-world data have target measurement errors, heterogeneous, correlated features, and no prescribed functional class. In the current simple formulation, Eq. (3) treats each training target as the expectation value; data errors are not treated by any noise model, data averaging, or outlier rejection. Like statistical physics, SML favors a few high-quality measurements over many low-quality ones. To evaluate SML under more complicated conditions, we test it on all 848 eligible dense-numeric datasets from OpenML (Vanschoren et al., 2014). Datasets that are too sparse, too small for the protocol, unavailable, or malformed are not studied (SM Sec. 6.1). As in Sec. 5.2, each OpenML dataset tests 20 model variants at ten $n/l = 0.1, 0.2, \ldots, 1.0$, with 30 held-out validation samples, 30 test samples, and ten repetitions per setting. Ranking uses mean test error over repetitions (SM Sec. 3.1); paired Wilcoxon tests with Holm correction assess significance. The 6,777 ranked evaluation cells comprise 1.46 million training runs. Tuned model variants receive a fresh search per run (up to 51 fits); SML-PN remains fixed-configuration.

SML-PN has the best aggregate performance among all fixed model variants (Fig. 3a). Compared with the tuned model variants, SML-PN attains a mean rank of 7.00, statistically tied with tuned CatBoost (6.79) and behind tuned XGBoost (5.58). Unlike the boosting models, SML-PN has no explicit noise-reduction mechanism. Tuned XGBoost and CatBoost perform considerably worse on the noise-free VLSE data (Sec. 5.2), and their fixed-configuration model variants significantly underperform SML-PN in all cases; the model scales also differ drastically (SM Secs. 6.6–6.7).

At $n/l = 0.1$, SML-PN has the highest rate of positive test R^2 (24.0%, versus XGBoost's 22.0%) and second-largest share of cell wins (24.3%, versus XGBoost's 26.1%; Fig. 3b,c): under greatest scarcity, its statistics-based prediction is most likely to extract useful signals. As n/l approaches 1, tuned boosting strengthens and TabPFN becomes competitive, while SML-PN remains in the leading group. Per-dataset comparisons show the overall trend (Fig. 3d): 70% and 57% favor SML-PN against MLP and TabPFN, 49% and 46% against SVR and CatBoost, and 38% against XGBoost.

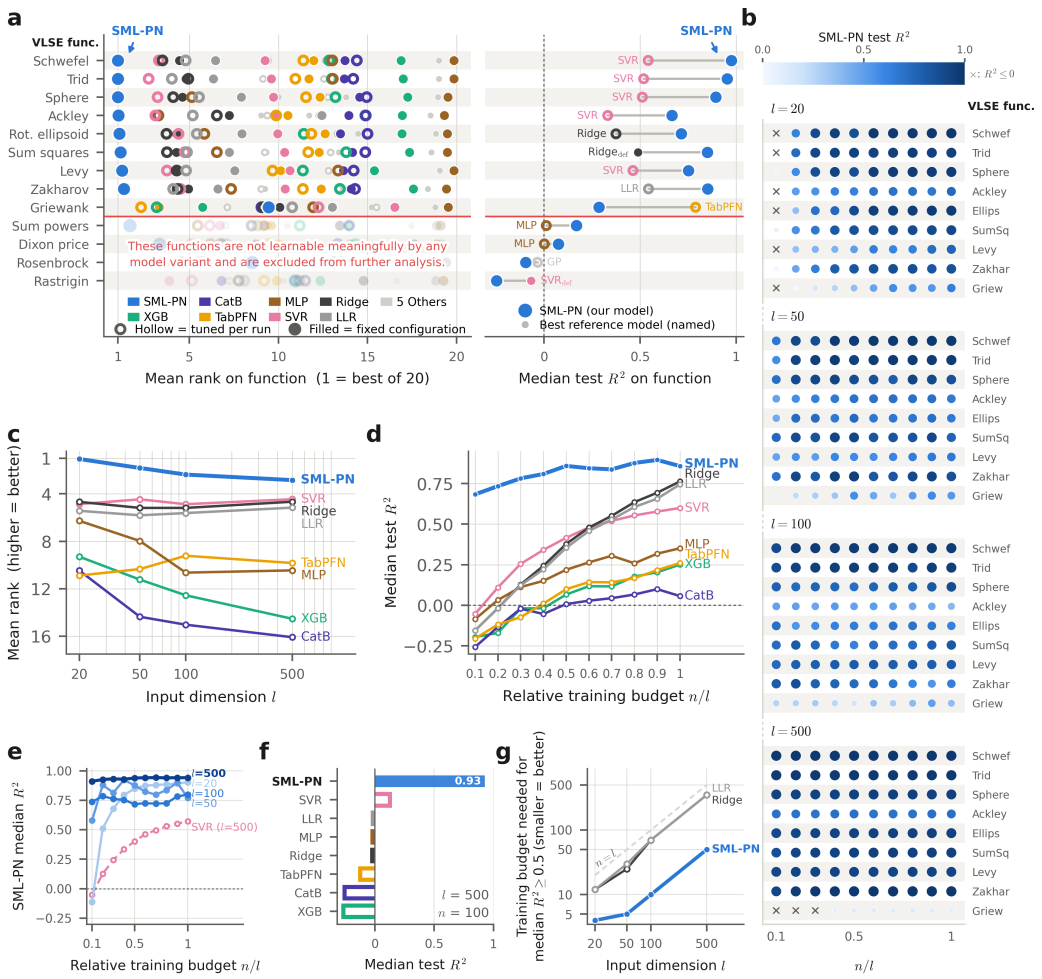

Figure 2: SML-PN outperforms all 19 reference model variants on most VLSE benchmark functions. Hollow symbols: per-run tuned model variants; solid symbols: fixed-configuration model variants, including SML-PN. (**a**) All 13 VLSE functions: mean rank (1 = best of 20), averaged over the input dimensions l and the training budgets n (left), and median test R^2 of SML-PN versus the best reference model variant, named at its marker (right). Results of the nine learnable functions (the analysis set): (**b**) the test R^2 of SML-PN across functions, relative training budgets n/l, and input dimensions l; (**c**) mean ranks versus the input dimension l; (**d**) median test R^2 versus n/l, pooled over the evaluation units; (**e**) SML-PN versus the second-best model variant, SVR; (**f**) median test R^2 versus seven reference model variants at $l = 500$ and $n/l = 0.2$; (**g**) the number of training samples required to reach a median $R^2 \geq 0.5$, with the dashed diagonal marking the full training budget $n = l$. MLP, TabPFN, XGBoost, and CatBoost never reach this threshold and are excluded from (g).

Across datasets, the per-dataset rank distribution of SML-PN peaks at 3.4 (the density strips in Fig. 3a), close to the peak of XGBoost (3.2) and well ahead of that of CatBoost (5.1). SML-PN is the outright best of all 20 model variants on 19.2% of datasets, more than double the champion rate of CatBoost (8.1%) and second only to XGBoost (23.5%); its standing is strongest in narrow tables (mean rank 4.0 for $l \leq 5$; Table S21). SM Sec. 6 provides full results.

6 Robustness and Sensitivity Tests

We test measurement noise, irrelevant features, and outliers or mislabeled training samples, and examine ε_k as an intrinsic measure of reliance on each input. Unlike Sec. 5, this section compares

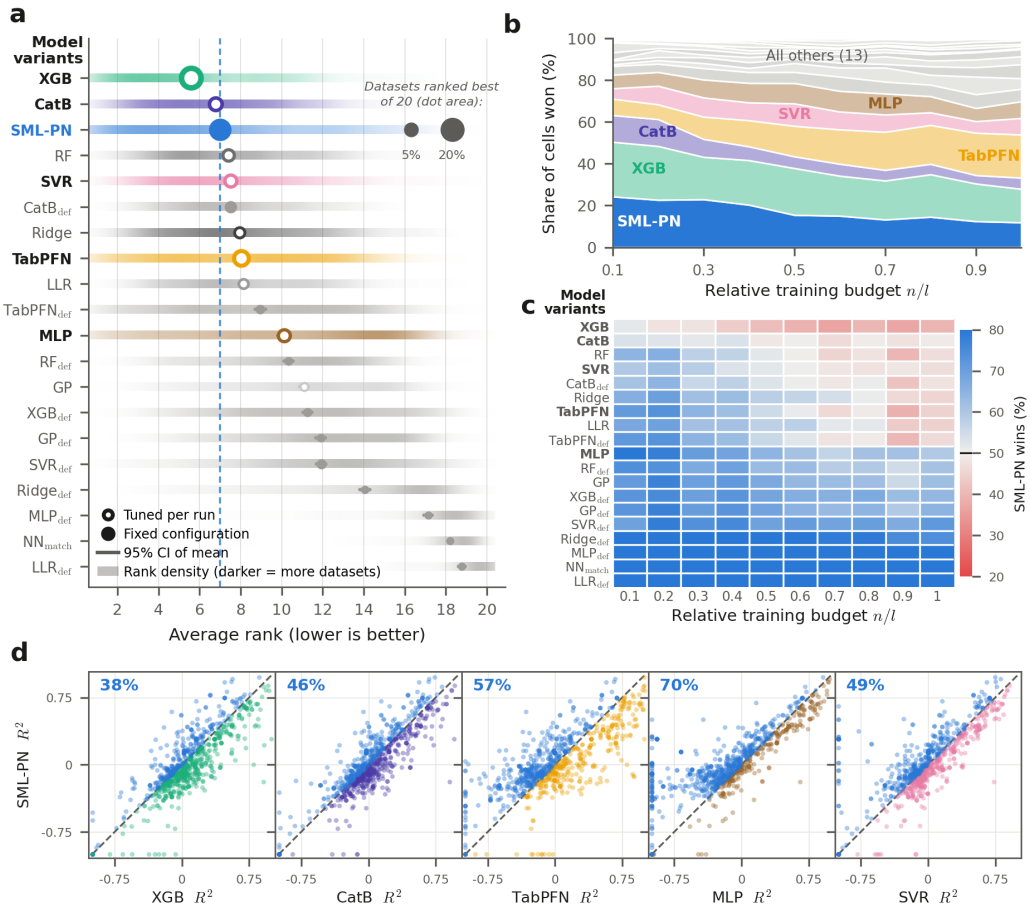

Figure 3: OpenML benchmark results. (**a**) Mean rank (1 = best of 20). Empty/solid symbols: per-run tuned/fixed-configuration model variants; row shading: per-dataset mean-rank distribution (darker = more datasets, scaled per model variant); thin lines: 95% bootstrap confidence intervals (CI); dot area: percentage of datasets on which the model variant is the best. (**b**) Fraction of evaluation cells won versus n/l. (**c**) SML-PN's head-to-head win rate versus n/l; blue/red indicates that SML-PN/the reference model variant wins on most datasets. Bold labels mark the five strongest reference model variants. (**d**) Per-dataset median test R^2 against the five strongest reference model variants; each dot is a dataset, and percentages give the fraction above parity.

SML-PN with its own unperturbed baseline. Four well-learned VLSE functions (Levy, Schwefel, Sphere, and Trid) use $l = 100$ informative features and $n = 50$ or 100, following SM Sec. 3.

Replacement contamination (RC; Fig. 4a) replaces a fraction $0 \leq p \leq 50\%$ of training targets with $\tilde{y}+5\sigma_{\tilde{y}}$, where $\tilde{y}$ is the true target and $\sigma_{\tilde{y}}$ its standard deviation. Additive contamination (AC; Fig. 4b) retains the n clean training samples and appends $p_+ n$ corrupted ones, giving $(1+p_+)n$ samples in total ($0 \leq p_+ \leq 100\%$). Validation and test targets remain clean. RC tests tolerance to lost clean signal; AC tests absorption of adversarial signal. Each function–n–contamination setting uses ten repetitions with independent random contamination draws and parameter initializations; corrupted data are randomly indexed over $1,\ldots,n$ for RC and $1,\ldots,(1+p_+)n$ for AC.

SML-PN degrades only mildly in the RC and AC tests. At $p = 0.5$ ($n = 100$), the median R^2 falls less than 10%. Doubling the training set with corrupted data ($p_+ = 100\%$) leaves Schwefel at 0.89, Trid at 0.86, Sphere at 0.82, and Levy at 0.70. At $n = 50$, the curves shift downward without changing the trend, i.e., sample size reinforces resilience. Equation (3) is based on the ensemble average, and parameter-space integration spreads the influence of each displaced training-data constraint. No explicit detection step, threshold, or robust loss is required.

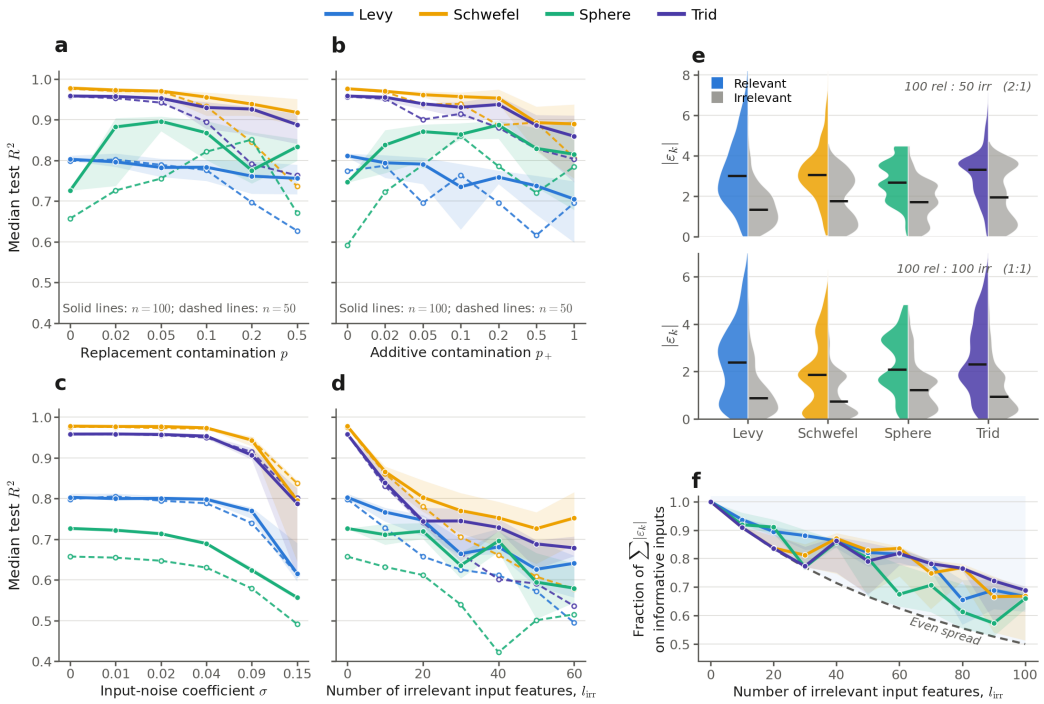

Figure 4: SML-PN is quite robust. (**a**) Replacement contamination: a fraction p of the training targets are replaced by strongly shifted outliers. (**b**) Additive contamination: a fraction p_+ of corrupted training samples are appended. (**c**) Gaussian noise added to training input. (**d**) Irrelevant features appended to 100 informative inputs. Solid lines with interquartile ribbons: $n/l = 1$; dashed lines: $n/l = 0.5$. (**e**) Split violins of $|\varepsilon_k|$ for relevant (colored, left) and irrelevant (grey, right) features, at relevant-to-irrelevant ratios 2:1 (top) or 1:1 (bottom); bars mark medians. (**f**) The fraction of total $\sum_k |\varepsilon_k|$ carried by informative inputs versus irrelevant features: median over ten repetitions with interquartile ribbons. The dashed line denotes evenly distributed weight.

To test the effects of noise in training input (Fig. 4c), the clean input $\mathbf{x}$ is changed to $\mathbf{x} + \sigma\mathbf{x}_e$, with $\mathbf{x}_e \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_l)$ and $0 \leq \sigma \leq 0.15$ being randomly generated noise. The validation inputs are similarly contaminated, so that early stopping reflects the noisy training. Each function–n–σ setting uses ten random draws and parameter initializations. The test R^2 degrades quite mildly with σ. Noise perturbs the training-data constraints in Eq. (11), causing bounded shifts in β_j. Parameter-space integration averages out the isolated perturbations.

In Fig. 4d, we append l_{irr} irrelevant input features onto the $l = 100$ relevant ones. They are drawn from the same distribution as the relevant features but never used in the target calculation, i.e., they are unrelated to $\bar{y}$. The total input dimension becomes $l + l_{\text{irr}}$, while $\bar{y}$ and the training task remain unchanged. Each function–n–l_{irr} setting uses shared seeds across the training, validation, and test datasets, with ten independent random splits and initializations. Degradation is never catastrophic: at $n = 100$ and $l_{\text{irr}} = 60$, the median R^2 remains approximately 0.5–0.8. With $n = 50$, performance declines faster because information is scarcer and the training samples are more diluted.

The interconnection factors ε_k (Eq. 11) distinguish relevant from irrelevant input features, potentially offering early detection of dilution (Fig. 4e). The relevant features' median $|\varepsilon_k|$ is 1.6–2.3 times that of the irrelevant features at $l_{\text{irr}} = 50$, and 1.7–2.7 times at $l_{\text{irr}} = 100$. With 100 irrelevant features, informative inputs retain 0.66–0.69 of total $\sum_k |\varepsilon_k|$, versus 0.50 for evenly spread weight (Fig. 4f); the margin widens monotonically with dilution. This concentration on informative inputs is rooted in the intrinsic relevance signal encoded in Eq. (11).

7 DISCUSSION

An intuitive way to view SML is as follows: a student shadowing a coach learns the coach's expected behavior, without reconstructing the coach's neural circuitry (Pouget et al., 2013). Equation (3) is based on the integration of the probability distribution ρ_μ over all possible $\{w_i\}$, rather than on a specific $\{w_i\}_o$. Each training sample lowers the maximum entropy that the system can reach. The maximum-entropy solution of ρ_μ is unique, even if multiple parameter configurations may fit the training data equally well. The expectation value of y_* is not directly affected by the saddle points and local minima that shape gradient trajectories. As in kernel methods (Schölkopf & Smola, 2002), the number of unknowns of SML equals n (the fewest that the training data can support), independent of the model size m, avoiding the non-uniqueness of $m > n$. When the data cannot resolve the target, as for Rosenbrock and Rastrigin, the expectation falls back to the least-committal prediction and the test R^2 approaches zero; when they can, ρ_μ concentrates on relevant information. The computational target is therefore determined entirely by the training data, rather than selected from among equally optimal solutions. Once $\{\beta_j\}$ are obtained, the prediction follows from Eq. (12) with no further fitting (SM Secs. 3.3–3.4). This approach especially suits HDLSS, where, ideally, a prediction should encode the training-data information and nothing else.

Interestingly, although the prediction never uses $\{w_i\}_o$, the maximum ρ_μ from Eq. (6) does identify the most probable parameter configuration $\{w_i\}_{max}$, raising the possibility of hybrid schemes: $\{w_i\}_{max}$ as a principled initialization for gradient training, or gradient information to narrow the parameter bounds. Combining such concepts with unsupervised objectives (Chen et al., 2020) and policy-gradient learning (Sutton et al., 2000) are future directions.

LLR shares SML-PN's log-feature map and function family but uses penalized least squares. Their performance gap (Fig. 2d,g) reflects how the coefficients are determined, not what the model can express. The nonlinear β_j–ε_k relation requires no kernel matrix, prescribed covariance, or inversion. SM Secs. 6.6–6.7 (Fig. S10) compare the model scales and search cost. Ensemble averaging retains PN's additive log-feature class when $n \leq l$ (Eq. 12). For $n > l$, the constraints in Eq. (11) cease to hold exactly, but least-squares determination of ε_k remains serviceable, and redundancy may mitigate measurement noise. At $n = l$, SML-PN reduces to a closed linear form, like many other conventional methods, which reflects a saturation limit of PN rather than of SML. PN lacks rotation invariance; tasks requiring it may use data augmentation or symmetry-aware feature engineering (Cohen & Welling, 2016; Bronstein et al., 2021).

The robustness of SML-PN is relatively interpretable (Sec. 6): because the training-data constraints bind ensemble means, integration absorbs corrupted training samples without identifying them first, and the same solution yields ε_k, revealing how much each input carries the fit without post hoc probing or extra computation. The same integrals yield predictive means and variances, opening the door to built-in uncertainty quantification.

PN is used here because its parameter-space integrals have analytical solutions. Extending SML to deeper or domain-specialized networks is feasible when the required integrals are tractable, and is the subject of ongoing work. In Eq. (3), $\bar{y}$ is a scalar for simplicity. SML extends readily to vector output (SM Sec. 7).

8 CONCLUDING REMARKS

SML makes predictions by integrating over all possible parameter configurations to obtain the expectation value of the output under maximum entropy, with the training data as constraints. It never computes any specific parameter configuration. The unknowns β_j correspond one-to-one to the training samples. Regular system-level behavior emerges without resolving the irregular parameter-space landscape. In the fixed-configuration power-law network (PN) realization, SML-PN ranks first against 19 reference model variants across all relevant VLSE benchmarks and leads on 92% of learnable problems. Across all 848 relevant OpenML datasets, SML-PN is the best fixed-configuration model; compared to tuned model variants, it is statistically tied with CatBoost and surpassed only by XGBoost by a relatively small margin. SML-PN's advantage is stronger with fewer training samples, when information is more valuable. Label noise, irrelevant features, and contamination cause only mild degradation, without explicit robust losses or outlier rejection. The interconnection factors ε_k give feature-relevance signals for built-in interpretation.

ETHICS STATEMENT

This work uses only publicly available benchmark data: the analytic test functions of the Virtual Library of Simulation Experiments and 848 public regression datasets from OpenML. No human subjects, personally identifiable information, or private data were involved, and no new data were collected. SML-PN is a general-purpose regression method, and we are aware of no application-specific risk arising from the results reported here beyond those common to predictive modeling. The authors declare no conflicts of interest.

REPRODUCIBILITY STATEMENT

Every result in this paper is reproducible from the material accompanying the submission. SM Sec. 3 specifies the complete comparison protocol: dataset eligibility and exclusion rules, the training-budget ladder, the validation and test split sizes, the number of independent repetitions, and the per-repetition seeding applied identically to every model variant. SM Sec. 2 gives the full configuration and, for the tuned model variants, the search space of all 19 reference model variants. SM Secs. 4–6 report the per-experiment settings behind Figs. 1–3 and the model-scale comparison (SM Secs. 6.6–6.7), and SM Sec. 7 derives the multi-output extension. SML-PN itself has no hyperparameter search: a single fixed configuration, stated in Sec. 4 of the main text, produces every SML-PN number in the paper.

The accompanying code release contains the frozen dataset lists that define the 848-dataset OpenML corpus, so the corpus is pinned independently of subsequent changes to the OpenML server, together with the per-run artifacts for every reported cell. Every figure and table can therefore be regenerated from the released tables without re-running the benchmark, and each benchmark cell can be recomputed from the released code. Code, frozen dataset lists, and per-run artifacts are available in the supplementary code archive.

THE USE OF LARGE LANGUAGE MODELS

Large language models were used as an authoring and engineering aid in preparing this work: editing and condensing manuscript prose, and assisting with analysis, figure-generation, and experiment-harness code. They were not used to generate research ideas, to design the method, to produce or interpret experimental results, or as any component of SML or of the reported experiments. All scientific claims, experimental designs, and numerical results are the authors' own. The authors verified every reported number against the underlying run artifacts and take full responsibility for the content of this paper, including any text that originated as a model suggestion.

REFERENCES

Anonymous. Supplementary Material for this manuscript, 2026. Included as supplementary material with this submission.

L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković. Geometric deep learning: grids, groups, graphs, geodesics, and gauges. *arXiv:2104.13478*, 2021.

T. Chen and C. Guestrin. XGBoost: a scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, 2016.

T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, pp. 1597–1607, 2020.

A. Chowdhery et al. PaLM: Scaling Language Modeling with Pathways. *arXiv:2204.02311*, 2022.

Code artifacts. Code, frozen dataset lists, and per-run artifacts of the present investigation. Included as supplementary material with this submission.

T. S. Cohen and M. Welling. Group equivariant convolutional networks. In *International Conference on Machine Learning*, pp. 2990–2999, 2016.

I. Daubechies. *Ten Lectures on Wavelets*. SIAM, 1992.

X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pp. 315–323, 2011.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

P. Hall, J. S. Marron, and A. Neeman. Geometric representation of high dimensional low sample size data. *Journal of the Royal Statistical Society: Series B*, 67(3):427–444, 2005.

K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: surpassing human-level performance on ImageNet classification. In *IEEE International Conference on Computer Vision*, pp. 1026–1034, 2015.

G. E. Hinton. Products of experts. In *9th International Conference on Artificial Neural Networks (ICANN)*, volume 1, pp. 1–6, 1999.

G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

W. Hoeffding. A class of statistics with asymptotically normal distribution. *Annals of Mathematical Statistics*, 19(3):293–325, 1948.

A. E. Hoerl and R. W. Kennard. Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

N. Hollmann, S. Müller, L. Purucker, A. Krishnakumar, M. Körfer, S. B. Hoo, R. T. Schirrmeister, and F. Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637: 319–326, 2025.

S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.

J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proc. National Academy of Sciences*, 79(8):2554–2558, 1982.

E. T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106(4):620–630, 1957a.

E. T. Jaynes. Information theory and statistical mechanics. II. *Physical Review*, 108(2):171–190, 1957b.

M. Kardar. *Statistical Physics of Particles*. Cambridge University Press, 2007.

D. P. Kingma and J. Ba. Adam: a method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

D. P. Kingma and M. Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014.

B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, volume 30, pp. 6402–6413, 2017.

Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. J. Huang. A tutorial on energy-based learning. In *Predicting Structured Data*. MIT Press, 2006.

D. J. C. MacKay. A practical Bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472, 1992.

K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

V. Nair and G. E. Hinton. Rectified linear units improve restricted Boltzmann machines. In *International Conference on Machine Learning*, pp. 807–814, 2010.

R. M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag, 1996.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, et al. PyTorch: an imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, pp. 8024–8035, 2019.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, et al. Scikit-learn: machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

A. Pouget, J. M. Beck, W. J. Ma, and P. E. Latham. Probabilistic brains: knowns and unknowns. *Nature Neuroscience*, 16(9):1170–1178, 2013.

L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin. CatBoost: unbiased boosting with categorical features. In *Advances in Neural Information Processing Systems*, volume 31, pp. 6638–6648, 2018.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer, 2nd edition, 2004.

A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola. *Global Sensitivity Analysis: The Primer*. Wiley, 2008.

B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.

H. S. Seung, H. Sompolinsky, and N. Tishby. Statistical mechanics of learning from examples. *Physical Review A*, 45(8):6056–6091, 1992.

A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 2004.

I. M. Sobol'. Sensitivity estimates for nonlinear mathematical models. *Mathematical Modelling and Computational Experiment*, 1(4):407–414, 1993.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1): 1929–1958, 2014.

S. Surjanovic and D. Bingham. Virtual Library of Simulation Experiments: Test Functions and Datasets. Simon Fraser University, 2013. URL `http://www.sfu.ca/~ssurjano`.

R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 12, pp. 1057–1063, 2000.

J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo. OpenML: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60, 2014.

M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *Proc. ICML*, pp. 681–688, 2011.

F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256, 1992.

SUPPLEMENTARY MATERIAL FOR A STATISTICAL-PHYSICS-INSPIRED FRAMEWORK OF MACHINE LEARNING

1 STATISTICAL PHYSICS: ENSEMBLE AVERAGE AND EXPECTATION VALUES

Consider a classical ideal gas as an example of a large-scale chaotic system. It consists of N_{gp} gas particles (hard spheres). Each possible microstate is defined by a set of internal parameters $\{w_i\}_{i=1}^{m}$, e.g., the momentum and position of every gas particle. The number of parameters is $m = 6N_{gp}$.

In isolation, all possible microstates have equal probability $\overline{\rho}_g$. Denote two possible microstates by "A" and "B", respectively. In microstate "A", the gas particles are uniformly distributed throughout the container. In "B", the gas particles are concentrated in a corner of the container. Although "A" and "B" are equally likely to occur, "B" is irrelevant to the calculation of macroscopic temperature and pressure.

In statistical physics, the analysis is typically not based on any particular microstate $\{w_i\}_{\mathrm{o}}$, even if $\{w_i\}_{\mathrm{o}}$ may have the maximum $\overline{\rho}_g$. Instead, to evaluate system-level observables (e.g., temperature and pressure), we compute ensemble averages over the entire phase space to obtain their expectation values. The integration smooths out the effects of irregular parameter-space landscape, so that regular macroscopic properties naturally emerge from tortuous microstates.

The equilibrium state is investigated through the principle of maximum entropy, similarly to Eq. (4) in the main text:

$$\frac{\partial \mathscr{L}}{\partial \rho_\mu} = 0,$$

where ρ_μ is the probability of the μ-th possible microstate, and

$$\mathscr{L} = -\sum_\mu \rho_\mu \ln \rho_\mu + \alpha_0 \left(\sum_\mu \rho_\mu - 1 \right) + \beta \left(U - \sum_\mu \rho_\mu \varepsilon_\mu \right)$$

is the Lagrangian, α_0 and β are the Lagrange multipliers, U is the internal energy, ε_μ is the energy of the μ-th possible microstate, and the units are chosen such that the Boltzmann constant $k_B = 1$. The Lagrangian takes into account the completeness condition

$$\sum_\mu \rho_\mu = 1$$

and energy conservation

$$\sum_\mu \rho_\mu \varepsilon_\mu = U,$$

which are counterparts of Eqs. (2) and (3) in the main text, respectively. The concepts of the maximum entropy ($\partial \mathscr{L} / \partial \rho_\mu = 0$) and the expectation value of measurement result ($\sum_\mu \rho_\mu \varepsilon_\mu = U$) are the foundation of statistical physics, leading to the Boltzmann factor $e^{-\beta \varepsilon_\mu}$, the Maxwell-Boltzmann distribution (for classical particles), Bose-Einstein distribution (for bosons), and Fermi-Dirac distribution (for fermions).

The physical meaning of β is clear: the inverse temperature. In physics, temperature is typically measured rather than calculated from microscopic dynamics. In the present investigation, the counterpart of β is $\{\beta_j\}_{j=1}^{n}$, which are calculated, since the measurement method (e.g., using a known "thermometer" neural network) has not been formulated.

2 REFERENCE MODEL VARIANTS

2.1 THE PARAMETER-MATCHED BASELINE MODEL $\mathrm{NN_M}$

A conventional neural network ($\mathrm{NN_m}$) is employed as the baseline model. It has the same number of trainable parameters as that of SML-PN. Because SML-PN and $\mathrm{NN_m}$ are matched in size, they mainly differ in how the parameters are determined: SML-PN integrates the probability of parameter configuration (ρ_μ) over the parameter space under the principle of maximum entropy, whereas $\mathrm{NN_m}$ fits the parameters by gradient descent.

NN_m is an extensively studied fully connected feed-forward network with ReLU activation (Nair & Hinton, 2010; Glorot et al., 2011). Like SML-PN, NN_m consists of one hidden layer. It maps the input vector $\mathbf{x}$ to a scalar output, $\widehat{y} = W_2\, ReLU(W_1\, x + b_1) + b_2$, with $W_1 \in R^{h \times l}$, $b_1 \in R^h$, $W_2 \in R^{1 \times h}$, $b_2 \in R$, h is the hidden-layer width, and l the input dimension. Weights are initialized by He (Kaiming-normal, fan-in) initialization (He et al., 2015) and biases at zero. The implementation uses PyTorch (Paszke et al., 2019).

The hidden-layer width h is the only quantity that varies across experiments, and it is not tuned: h is specifically chosen as $h = round((n + 2l + 1) / (l + 2))$, so that the total number of trainable-parameters of NN_m is close to that of SML-PN. A similar yet more powerful reference model is the freely sized MLP, discussed in SM Sec. 2.2. MLP comprises a few hidden layers of 16-264 units.

Table S1: **List of the reference models**

Reference model	**Abbrev.**	**Implementation**	**Model variants**	**Tuned dims.***	**Fits per run****
SML-PN (this work)	SML-PN	in-house (PyTorch)	fixed	—	1
Param.-matched feed-forward NN	NN_m	in-house (PyTorch)	fixed	—	1
XGBoost	XGB	xgboost	fixed + tuned	10	51
CatBoost	CB	catboost	fixed + tuned	6	51
Random forest	RF	scikit-learn	fixed + tuned	4	51
Multilayer percep. (freely sized)	MLP	scikit-learn	fixed + tuned	6	51
ε-SVR	SVR	scikit-learn	fixed + tuned	3 (exhaustive)	25
Gaussian-process regression	GP	scikit-learn	fixed + tuned	1 (kernel family)	4
Ridge regression	Ridge	scikit-learn	fixed + tuned	1 (α)	9
Log-linear ridge	LLR	scikit-learn	fixed + tuned	1 (α)	9
TabPFN	TabPFN	official release	fixed + tuned	1 (ensemble size)	4
Total			**20 model variants**		

* Tuned dims.: the number of hyperparameters over which the tuned model variant searches.
** Fits per run: the number of candidate configurations trained and scored in one run before one of them is kept; the first candidate is always the library default.

NN_m is the architectural counterpart of PN. PN has a continuous-parameter neuron-style architecture in which inputs are combined through learned weights followed by a nonlinear activation. In this sense, PN belongs to the architectural family of feed-forward networks, rather than to tree ensembles based on recursive partitioning or Gaussian processes based on kernel methods. Such similarity between PN and NN_m makes direct parameter matching and an identical training protocol meaningful.

2.2 Complete comparison field

There are 11 models in the comparison field, giving 20 model variants (Table S1). Two are fixed-configuration by design and are described in SM Sec. 2.1: SML-PN (this work) and the parameter-matched NN_m. There are nine other high-performance reference models, each with two model variants: a default fixed-configuration model variant (the library-default configuration, fitted once per run) and a tuned model variant (subjected to a fresh validation-based search in every run). A run here means a single combination of evaluation unit, training budget, and repetition. The nine reference models are listed below:

- XGBoost (Chen & Guestrin, 2016) and CatBoost (Prokhorenkova et al., 2018), gradient-boosted tree ensembles;
- random forest (Breiman, 2001);
- a freely sized multilayer perceptron (scikit-learn MLPRegressor) (Pedregosa et al., 2011), the capacity-unrestricted counterpart of NN_m;

- ε-SVR (Pedregosa et al., 2011; Smola & Schölkopf, 2004), with RBF, polynomial, or sigmoid kernel;
- Gaussian-process regression (Rasmussen & Williams, 2006), whose continuous hyperparameters (length scales and noise level) are fitted to the training samples by maximizing the log marginal likelihood with ten restarts, as is standard for Gaussian processes; the kernel family serves as the tuned degree of freedom;
- ridge regression (Hoerl & Kennard, 1970);
- log-linear ridge (LLR) (Hoerl & Kennard, 1970), ridge on logarithmic features, which reproduces SML-PN's prediction basis and therefore serves as the shared-basis control; and
- TabPFN (Hollmann et al., 2025), the prior-fitted foundation model, used through the official release with its published inference-time ensembling.

Implementations are scikit-learn, XGBoost, CatBoost, PyTorch, and the official TabPFN release. SML-PN and $\mathrm{NN_m}$ are implemented with in-house codes. Table S1 lists the 20 model variants entering every evaluation cell. Full settings and search spaces are given in Tables S2–S12.

Table S2: SML-PN: The configuration is fixed and used in all evaluation cells. No quantity is searched, and none is selected separately per dataset, input dimension, or training budget.

Setting	**Values**
Trainable parameters	$\beta_j \in R^n$; $\check{w}_i$, $\hat{w}_i \in R^l$; scale factor T; intercept b (total number: $n + 2l + 2$)
Initialization of β_j	log-normal, mean $\frac{1.5}{n}$, variance $\frac{1}{n}$ (positive by construction)
Initialization of parameter bounds	$\check{w}_i = w_0 - \lvert\delta\rvert$, $\hat{w}_i = w_0 + \lvert\delta\rvert$; $w_0 = -1.2$; $\delta \sim N(1,\ 0.2^2)$
T, b initialization	$T = 1$; $b \sim N(0,\ 0.01^2)$
Objective	$\mathrm{MSE} + \lambda \lVert \mathrm{ReLU}(\check{\mathbf{w}} - \hat{\mathbf{w}}) \rVert_2$, $\lambda = 100$
Optimizer	Adam, full batch
Schedule	Hierarchical coarse-to-fine tying (SM Sec. 3.4)
Number of tying stages	$\lceil \log_2 max(n,\ l) \rceil + 1$, followed by one full-training stage
Tying stage: max epochs / patience / lr	10,000 / 4,000 / $0.1 \times 0.5^{k-1}$, floored at 10^{-3}
Final stage: max epochs / patience / lr	50,000 / 20,000 / 10^{-3} (10^{-4} for T and b)
Numerical guards	$log\ B$ clamped to $[-4,\ 4]$; β_j to $[-13,\ 10]$; $\check{w}_i$, $\hat{w}_i$, b to $[-15,\ 15]$; T to $[0.01,\ 100]$; gradients to $\pm\ 10^3$
Input / target scaling	$[0.9,\ 1.2]$ / $[-10,\ 10]$
Model selection	Checkpoint of lowest validation loss
Hyperparameters searched	None

Table S3: $\mathrm{NN_m}$: The parameter-matched feed-forward control model.

Setting	**Values**
Architecture	Fully connected, one hidden layer, ReLU activation
Hidden-layer width h	$round((n + 2l + 1)\ /\ (l + 2))$
Initialization	He (Kaiming-normal, fan-in) weights; biases at zero
Objective	Mean squared error
Optimizer	Adam, learning rate 10^{-3}
Mini-batch size	32
Max epochs / patience	50,000 / 20,000
Model selection	Checkpoint of lowest validation loss
Input / target scaling	$[0,\ 1]$ / $[0,\ 1]$
Hyperparameters searched	None

Table S4: XGBoost: Ten hyperparameters searched; 50 sampled configurations plus the default, for a total of 51 fits per run. Distributions follow (Chen & Guestrin, 2016).

Held fixed in both model variants. 50 early-stopping rounds on the validation set; single thread; per-repetition seed; no feature scaling. The default model variant uses the library defaults at these settings, with the estimator count capped at 1000; the tuned model variant draws the count from the space below.

Searched by the tuned model variant	**Range / distribution**
Learning rate	log-uniform $[e^{-7},\ 1]$
Maximum depth	integer uniform $[1,\ 10]$
Subsample	uniform $[0.2,\ 1]$
Colsample by tree	uniform $[0.2,\ 1]$
Colsample by level	uniform $[0.2,\ 1]$
Minimum child weight	log-uniform $[e^{-16},\ e^{5}]$
α (L1 penalty)	log-uniform $[e^{-16},\ e^{2}]$
λ (L2 penalty)	log-uniform $[e^{-16},\ e^{2}]$
γ (split-loss threshold)	log-uniform $[e^{-16},\ e^{2}]$
Number of estimators	integer uniform $[100,\ 4000]$

Table S5: CatBoost: Six hyperparameters searched; 51 fits per run.

Held fixed in both model variants. 50 early-stopping rounds with best-model restoration; RMSE loss; depth 6; single thread; per-repetition seed; no feature scaling. The default model variant is capped at 1000 iterations; the tuned model variant draws the count from the space below.

Searched by the tuned model variant	**Range / distribution**
Learning rate	log-uniform $[e^{-5},\ 1]$
Random strength	integer uniform $[1,\ 20]$
L2 leaf regularization	log-uniform $[1,\ 10]$
Bagging temperature	uniform $[0,\ 1]$
Leaf-estimation iterations	integer uniform $[1,\ 20]$
Number of iterations	integer uniform $[100,\ 4000]$

Table S6: Random forest: Four hyperparameters searched; 51 fits per run.

Held fixed in both model variants. Bootstrap resampling; single thread; per-repetition seed; no feature scaling; all remaining settings at the scikit-learn defaults.

Searched by the tuned model variant	**Range / distribution**
Number of estimators	integer uniform $[20,\ 200]$
Maximum features	{log2, sqrt}
Maximum depth	integer uniform $[1,\ 45]$
Minimum samples to split	$\{5, 10\}$

Table S7: Multilayer perceptron (scikit-learn MLPRegressor), the capacity-unrestricted reference model: Six hyperparameters searched; 51 fits per run.[*]

Held fixed in both model variants. At most 2000 optimizer iterations per fit; per-repetition seed; inputs and targets scaled to $[0,\ 1]$.

Searched by the tuned model variant	**Range / distribution**
Hidden-layer depth	$\{1, 2, 3\}$
Nodes per layer	integer uniform $[16,\ 264]$
Activation	{ReLU, tanh}
α (L2 penalty)	log-uniform $[e^{-7},\ 0.1]$
Initial learning rate	log-uniform $[e^{-4},\ 0.5]$
Early-stopping mode	{training loss, internal validation split}

[*] The sampled early-stopping mode governs only what scikit-learn monitors while fitting a single candidate; selection across candidates always uses the external validation set.

Table S8: ε-SVR: The space is purely categorical, so that the full $4 \times 2 \times 3$ grid is enumerated rather than sampled: 24 combinations plus the default, i.e., 25 fits per run.

Held fixed in both model variants. $\varepsilon = 0.1$; polynomial degree 3; zero independent kernel term; tolerance 10^{-3}; shrinking heuristic on; inputs and targets scaled to $[0, 1]$.

Searched by the tuned model variant	**Values (enumerated exhaustively)**
C	{0.1, 1, 10, 100}
γ	{auto, scale}
Kernel	{RBF, polynomial, sigmoid}

Table S9: Gaussian-process (GP) regression: The continuous hyperparameters are fitted on the training set by marginal likelihood, as is standard for GPs.*

Held fixed in both model variants. Length scales and noise level fitted on the training set by maximizing the log marginal likelihood, with 10 restarts; initial length scale 1.0 with bounds $[10^{-2}, 10^{2}]$; initial white-noise level 10^{-6}; jitter 10^{-10}; isotropic length scale (no ARD); internal target normalization; per-repetition seed; inputs and targets scaled to $[0, 1]$.

Searched by the tuned model variant	**Values**
Kernel family	{RBF, Matérn $\nu = 3/2$, Matérn $\nu = 5/2$, rational quadratic}; the default model variant uses RBF

* Only the kernel family is selected on the validation set, giving 4 fits per run.

Table S10: Ridge regression.

Held fixed in both model variants. Intercept fitted; inputs and targets scaled to $[0, 1]$.

Searched by the tuned model variant	**Values**
Penalty α	$\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^{2}, 10^{3}, 10^{4}\}$; the default model variant uses $\alpha = 1$, the scikit-learn default

Table S11: Log-linear ridge (LLR), the shared-basis control: Ridge on logarithmic features, i.e., the prediction basis of Eq. (12).*

Held fixed in both model variants. Prediction $\widehat{y} = \sum_k \varepsilon_k \, log \, x_k + b$; intercept fitted; inputs scaled to $[0.9, 1.2]$ and targets to $[-10, 10]$, identical to SML-PN.

Searched by the tuned model variant	**Values**
Penalty α	the same nine-point grid as Table S10; the default model variant uses $\alpha = 0$, i.e., unregularized least squares on the log-features

* The coefficients are obtained through penalized least squares instead of maximum entropy.

Table S12: TabPFN: The prior-fitted network is used as released; only the inference-time ensemble size is selected, giving 4 fits per run.

Held fixed in both model variants. Official release, on CPU; no external feature scaling; softmax temperature 0.9; pretraining limits enforced; per-repetition seed.

Searched by the tuned model variant	**Values**
Inference-time ensemble size	{4, 8, 16, 32}; the default model variant uses the library default

3 EXPERIMENTAL SETUP AND COMPARISON PROTOCOL

All numerical codes are available in the supplementary code archive. This section specifies the numerical protocol in Sec. 5 of the main text. More details of the experimental results are given in SM Secs. 4–6. Throughout the manuscript, the following notations and terminology are used:

- Model variant: one of the 20 machine-learning methods being evaluated, including SML-PN (this work), $\mathrm{NN_m}$, and nine other high-performance reference models (each having a library-default configuration and a per-run tuned configuration). Altogether, there are 20 model variants, as listed in Table S1.
- The number of internal parameters of a model is denoted m.
- Configuration (also referred to as parameter configuration or internal-parameter configuration): the set of hyperparameter values that a model is fitted with, either the library default or the one chosen by a per-run search.
- Sample: one row of training data (an input vector $\mathbf{x}$ and its target y).
- Training-set size n (also referred to as the training budget): the number of training samples.
- Input dimension l: the number of features of the input vector $\mathbf{x}$ after preprocessing.
- Relative training budget n/l. Every evaluation unit is assessed at ten relative training budgets, $n/l = 0.1, 0.2 \ldots 1.0$.
- Function: one of the 13 closed-form VLSE equations (see SM Sec. 5). A function is a mathematical formula, from which training samples as well as validation and test data are generated at a chosen input dimension, and shared by all model variants.
- Dataset: one OpenML entry (i.e., a real-world case), providing a table of multiple training samples (see SM Sec. 6).
- Evaluation unit (also referred to as unit): a case study on which a learning curve is measured, i.e., a VLSE function with a given input dimension or an OpenML dataset.
- Cell: an evaluation unit with a given training budget. Within a cell, all 20 model variants receive identical training, validation, and test data, and are ranked from 1 to 20.
- Repetition: one of the ten runs of a cell, each under its own seed. Every quantity reported for a cell is a mean over the ten repetitions.
- Benchmark: a complete collection of evaluation units assessed under one protocol. This work uses two benchmarks, the VLSE benchmark and the OpenML benchmark.
- In tables and figures, the subscript "def" in a model name denotes the default model variant with a fixed configuration; the absence of the subscript denotes the tuned model variant, except for SML-PN and NNm, whose configurations are always fixed.

3.1 DATA PARTITIONING, TRAINING BUDGETS, AND REPETITIONS

- Training budget. Every evaluation unit (e.g., a function-dimension unit in Sec. 5.2 in the main text or a dataset in Sec. 5.3 in the main text) is evaluated at ten training budgets (the network-synthesized experiment of SM Sec. 4 uses 20) defined relative to the input dimension l: $n/l = 0.1, 0.2, \ldots, 1.0$. All results reported in this work are for $n \leq l$. Training budgets for which $n + 60$ exceeds the number of available rows are skipped, while the remaining budgets are retained. A dataset for which no budget is feasible is excluded from the corpus.
- Partitioning. Within a repetition, the rows are shuffled once, and a validation set and a test set (30 samples each) are taken from fixed positions in the shuffled rows. They are identical across all ten training budgets: only the training set grows, such that the ten points of a learning curve are measured against the same evaluation set, and a change in error reflects the training-set size rather than a change of evaluation data. Within a given cell, all 20 model variants receive identical training, validation, and test partitions.
- Preprocessing. All sample-dependent preprocessing is fitted on the training set and then applied unchanged to the validation and test sets: integer encoding of nominal columns,

imputation of missing entries, detection and removal of constant features, and the min-max scalings (see SM Sec. 3.2). Free-text and row-identifier columns are dropped beforehand. No statistics of the validation or test set enter any fitted model at any stage. Note that validation and test features are not guaranteed to fall inside the nominal scaled range.

- Repetitions. Each setting is tested repeatedly ten times under nominally identical conditions, with seeds $s_r = 42 + r$ for $r = 0, \ldots, 9$. A single seed governs the whole repetition, including the shuffle that fixes the partition, the draw of the training set, the Python, NumPy, and PyTorch global generators, and every model's initialization and internal randomness. Thus, repetitions vary the data partition and the model initialization jointly, i.e., the spread across repetitions is the combined variability of the two.
- Scoring and ranking. Predictions and targets are inverse-transformed to the unit's native target scale before any metric is computed, so that all 20 model variants are scored on one common scale, irrespective of the scaling under which they are trained (see SM Sec. 3.2). Within each (unit, budget) cell the model variants are then ranked from 1 to 20 by test error, smallest first, on their mean over the ten repetitions. A cell is therefore an evaluation of the whole field at a given training budget, from which every rank in this work is aggregated. Ranking by mean squared error (MSE) and ranking by test R^2 coincide, since for a fixed evaluation sample R^2 is a strictly decreasing affine function of the MSE. Unless stated otherwise, the reported statistics are means of per-unit mean ranks, medians of test R^2, or the quantities specified in the captions, taken across repetitions.

3.2 Feature and target scaling

Inputs and targets are rescaled by a min-max transform whose range is fixed in advance for each reference model, according to the model's domain of validity. They never vary across datasets, input dimensions, training budgets, or repetitions.

- SML-PN and log-linear ridge (LLR). Inputs are rescaled to $[0.9,\ 1.2]$ and targets are rescaled to $[-10,\ 10]$. The input range is strictly positive and bounded away from the origin, as required by the power-law network (PN) architecture. It avoids extreme values and singularity. The target range keeps Eq. (11) in the main text suited to Adam without compressing the dynamic range of the targets. LLR is given the same scaling by design, so that it occupies the reachable set of SML-PN and the gap between them isolates how the coefficients are determined rather than what each model can express.
- Feed-forward, kernel, and linear models. $\mathrm{NN_m}$, MLP, SVR, the Gaussian process, and ridge take inputs and targets on $[0,\ 1]$, the standard range for ReLU networks and for distance- and penalty-based estimators.
- Tree ensembles. XGBoost, CatBoost, and the random forest are left unscaled. Recursive partitioning is invariant under monotone transformations of individual features, so that scaling would change nothing else but the provenance of the numbers.
- TabPFN is not scaled externally. The released model uses its own internal normalization, and external feature scaling is contraindicated by the authors.

3.3 The SML-PN model

In SML-PN, the unknowns are the n Lagrange multipliers β_j, one per training sample, determined by the principle of maximum entropy through Eq. (11). They are obtained numerically by minimizing the mean squared error (MSE) between the ensemble average of Eq. (12) and the training targets. The β_j values corresponding to the lowest validation error are taken as the solution.

In the present investigation, we use the simplest possible single-layer PN configuration, i.e., $m = l$. The output is

$$\widehat{y} = -\log \prod_k x_k^{f_k} = -\sum_k f_k \log x_k,$$

where $f_k = \sum_i \alpha_{ki} w_i = w_k$, and α_{ki} is a delta function δ_{ki}.

Besides β_j, a number of other quantities are also calculated through MSE minimization: the parameter bounds, the scale factor, and the intercept. The parameter bounds ($\check{w}_i$ and $\hat{w}_i$) enter Eqs. (9) and (10) as the limits of integration. The closed-form derivation in Sec. 4 of the main text holds for a fixed choice of them. Here, they are treated as tunable. Such an implementation is not generally required by SML and is optional. It helps to improve numerical convergence. Since each bound is adjusted independently, the implementation calculates $2l$ bound values alongside the n Lagrange multipliers. The bounds are numerical aids that set the limits of integration; no training sample is attached to any of them. Every training sample continues to enter through its own Lagrange multiplier in Eq. (3). Because $\hat{w}_i \geq \check{w}_i$ by definition, the objective carries a barrier term $MSE + 100 \parallel ReLU(-(\hat{w}_i - \check{w}_i)) \parallel_2$, which is inactive whenever the integration interval is non-degenerate.

A scalar $T > 0$ rescales the interconnection factors, i.e., $\varepsilon_k \leftarrow \varepsilon_k / T$. It plays the role of the overall normalization of the Boltzmann form of Eq. (6), adjusting the overall scale of the Lagrange multipliers.

An intercept scalar b is added to the prediction as $\hat{y} = \sum_k \varepsilon_k \, tanh(log\, x_k^*) + b$. Ridge and LLR are likewise fitted with an intercept. Therefore, this treatment does not prioritize SML-PN in the comparison process. The tanh function is a bounded surrogate for the logarithm, such that the prediction is finite when a validation or test feature falls outside the training range. Over the scaled input domain $[0.9,\ 1.2]$, one has $log\ \mathbf{x} \in [-0.105,\ 0.182]^l$, where $\tanh(u)/u$ departs from unity by at most 1.1% and remains numerically inert within the operating range.

The total number of trainable quantities is $n + 2l + 2$. Of these, the n Lagrange multipliers are inherent. The remaining $2l + 2$ unknowns are auxiliary support and scale variables of the numerical realization. The parameter bounds ($\hat{w}_i$ and $\check{w}_i$) are not part of the general SML formulation of Sec. 3 of the main text. For PN, $\hat{w}_i$ and $\check{w}_i$ are the integration limits in Eqs. (9)–(11), and their values are adjusted numerically together with β. None of them is selected on held-out data.

The initialization procedure is identical in every run, with the random draws set by the seed of the repetition (SM Sec. 3.1): β_j is drawn log-normal with mean $1.5 / n$ and variance $1 / n$, hence positive by construction. The parameter bounds are set to $\check{w} = w_0 - |\delta|$ and $\hat{w} = w_0 + |\delta|$ with $w_0 = -1.2$ and $\delta \sim N(1,\ 0.2^2)$; $T = 1$; and $b \sim N(0,\ 0.01^2)$. Inside the constraint path, the training inputs are rescaled column-wise to $[0.9,\ 1.2]$ and $log\ B$ is clamped to $[-4,\ 4]$ before exponentiation, both for numerical conditioning and implementation stability. The complete configuration is listed in Table S2.

No quantity in this section is selected specifically for any dataset, input dimension, or training budget. That is, SML-PN has an identical initial configuration for every evaluation cell in Sec. 5 in the main text.

3.4 Numerical procedure for SML-PN

Let $\theta = (\beta,\ \check{\mathbf{w}},\ \hat{\mathbf{w}},\ b) \in \mathbb{R}^{n+2l+1}$ collect the Lagrange multipliers, the parameter bounds ($\hat{w}_i$ and $\check{w}_i$), and the intercept b, and let T be the scale factor; T is treated separately because it is trained only in the last stage. For a training input $\mathbf{x}_j$, the prediction of SM Sec. 3.3 is $\hat{y}_j(\theta, T) = \sum_k [\varepsilon_k(\theta)/T] \tanh\left(\log x_{jk}\right) + b$, where ε_k follows in closed form from Eqs. (9)–(11) in the main text. Full-batch Adam minimizes

$$J(\theta, T) = \frac{1}{n} \sum_{j=1}^{n} \left[\hat{y}_j(\theta, T) - \bar{y}_j\right]^2 + \lambda \, \left\|\mathrm{ReLU}\left(\check{\mathbf{w}} - \hat{\mathbf{w}}\right)\right\|_2 . \tag{S1}$$

The first term in Eq. (S1) is the training mean squared error (MSE), and the second term is the barrier discussed in SM Sec. 3.3, with the weight λ of Table S2: it is zero as long as every $\hat{w}_i$ stays above $\check{w}_i$, and it only penalizes a crossing. The validation mean squared error V is computed at every epoch and is used only to decide when to stop and which state to keep; it never enters a gradient.

With fixed parameter bounds, Eq. (11) is the stationarity condition of the convex potential $\ln Z + \sum_j \beta_j \bar{y}_j$, such that its solution is unique when it exists. Here, we solve Eq. (S1) instead because its least-squares form remains defined when the constraints cannot be met exactly. The map from θ to ε_k is nonlinear: in Eqs. (9)–(11), ε_k depends on β through the exponentials B_i and on the parameter bounds through $B_i^{\hat{w}_i}$ and $B_i^{\check{w}_i}$. The value of J is governed by coupled quantities: a rescaling of β_j

changes $\log B_i$, and the gap between $\widehat{w}_i$ and $\check{w}_i$ influences how strongly ε_k responds to B_i. The schedule below first adjusts a few coupled quantities and then releases the individual components as degrees of freedom step by step.

In tying stage k, the current values of $\theta^{(k-1)}$ and T are frozen, and the components of θ are partitioned into groups. Each group g receives one affine pair (a_g, c_g), initialized at $(1,0)$, and component r enters J with the effective value

$$\theta_r = a_{g(r)}\,\theta_r^{(k-1)} + c_{g(r)}, \tag{S2}$$

where $g(r)$ is the group of r. Only the pairs are trained, such that a stage with G groups has $2G$ unknowns, and by the chain rule

$$\frac{\partial J}{\partial a_g} = \sum_{r\in g} \theta_r^{(k-1)} \frac{\partial J}{\partial \theta_r}, \qquad \frac{\partial J}{\partial c_g} = \sum_{r\in g} \frac{\partial J}{\partial \theta_r}. \tag{S3}$$

A group therefore moves together, by a common rescaling and shift of its members driven by their pooled gradient, while the members keep their individual values; the group members are not averaged out. In the global stage, where every component shares one pair, all $\widehat{w}_i - \check{w}_i$ are multiplied by the same factor a and keep their sign as long as $a > 0$; the barrier term covers the remaining cases. When a stage ends, the transform is absorbed, $\theta_r^{(k)} = a_{g(r)}\theta_r^{(k-1)} + c_{g(r)}$, and the next stage starts from the identity.

The partitions are fixed by n and l and refine one another. Stage 1 uses a single group for all θ. Stage 2 uses one group per type: β_j, $\widehat{w}_i$, $\check{w}_i$, and b. Stage $k \geq 3$ splits each type of length N into $C = \min\left(2^{k-2}, N\right)$ contiguous chunks of nearly equal size, component $r = 0, \ldots, N-1$ going to chunk $\lfloor rC/N \rfloor$; the splitting continues while $2^{k-2} < \max(n,l)$, such that there are $K = \lceil \log_2 \max(n,l) \rceil + 1$ tying stages (eight for $n = l = 100$: global, per type, and 2, 4, 8, 16, 32, and 64 chunks per type). "Coarse to fine" thus refers to the resolution at which the parameters are grouped, not to any spatial or frequency scale. A final stage then trains every component of θ and T individually. Tying stage k uses learning rate $\eta_k = \max\left(0.1 \times 0.5^{k-1},\ 10^{-3}\right)$, at most $E_{\text{tie}} = 10{,}000$ epochs, and stops after $P_{\text{tie}} = 4{,}000$ epochs without validation improvement; the final stage uses $\eta = 10^{-3}$ ($\eta/10$ for T and b), $E_{\text{fin}} = 50{,}000$, and $P_{\text{fin}} = 20{,}000$. Gradients are clipped to $\pm 10^3$ and the parameters are kept inside the boxes of Table S2. Algorithm S1 below lists the procedure. Every stage is left through its state of lowest validation loss, so that the retained validation loss never increases from one stage to the next, and the state kept at the end of the run (the best one over all stages) is the one evaluated on the test set.

The schedule is fixed and never adjusted per dataset or per run: the partitions, epoch caps, and learning rates depend on n and l only and are applied unchanged to every evaluation cell. The purpose is numerical conditioning, not model selection. The validation set is used only to decide when to stop and which state to keep, i.e., no model variant in the field receives more validation information than another.

3.5 Training, model selection, and hyperparameter optimization

Every model variant sees the same training and validation sets, and the test set is touched once, after all selection is completed. For various model variants, three different mechanisms are used for the validation set:

i) Early stopping and checkpoint selection, for the iteratively trained models. SML-PN and NN_m evaluate the validation loss at every epoch and restore the parameter state of lowest validation loss. XGBoost and CatBoost cap the number of boosting rounds generously and let the validation set select the effective number of trees, through 50 early-stopping rounds. Both MLP model variants stop training through the built-in convergence criterion of scikit-learn; the tuned model variant also searches which loss this criterion monitors (Table S7).

ii) Candidate selection over a search space, for the tuned model variants of XGBoost, CatBoost, random forest, MLP, SVR, ridge, and LLR. Each candidate configuration is fitted on the training set alone and scored on the validation set, and the candidate of lowest validation MSE is retained.

Algorithm S1 Coarse-to-fine training of SML-PN.

Require: training set, validation set; θ^0 and $T = 1$ (Table S2); epoch caps $E_{\text{tie}}, E_{\text{fin}}$; patiences $P_{\text{tie}}, P_{\text{fin}}$; learning rates η_k, η
Ensure: (θ^*, T^*), the state of lowest validation loss

$\theta \leftarrow \theta^0$; $(\theta^*, T^*) \leftarrow (\theta, T)$; $V^* \leftarrow V(\theta, T)$
$K \leftarrow \lceil \log_2 \max(n, l) \rceil + 1$
for $k = 1, \ldots, K$ **do** ▷ tying stages, coarse to fine; T frozen
  $\mathscr{G}_k \leftarrow$ groups of stage k ▷ one group; one per type; then 2^{k-2} chunks per type
  $(a_g, c_g) \leftarrow (1, 0)$ for $g \in \mathscr{G}_k$; $s \leftarrow 0$
  for $e = 1, \ldots, E_{\text{tie}}$ **do**
    $\theta_r^{\text{eff}} \leftarrow a_{g(r)} \theta_r + c_{g(r)}$ ▷ Eq. (S2)
    Adam step on $J(\theta^{\text{eff}}, T)$ w.r.t. $\{(a_g, c_g)\}$, rate η_k ▷ Eq. (S3)
    if $V(\theta^{\text{eff}}, T) < V^*$ **then** $\theta^* \leftarrow \theta^{\text{eff}}$; $V^* \leftarrow V(\theta^{\text{eff}}, T)$; $s \leftarrow 0$
    else $s \leftarrow s + 1$; **if** $s = P_{\text{tie}}$ **then break**
    end if
  end for
  $\theta \leftarrow \theta^*$ ▷ absorb the transform; restore the best state
end for
$s \leftarrow 0$ ▷ final stage
for $e = 1, \ldots, E_{\text{fin}}$ **do**
  Adam step on $J(\theta, T)$ w.r.t. (θ, T), rate η ($\eta/10$ for T and b)
  if $V(\theta, T) < V^*$ **then** $(\theta^*, T^*) \leftarrow (\theta, T)$; $V^* \leftarrow V(\theta, T)$; $s \leftarrow 0$
  else $s \leftarrow s + 1$; **if** $s = P_{\text{fin}}$ **then break**
  end if
end for
return (θ^*, T^*)

iii) Discrete configuration selection, for the two models that have no training loop to stop early: Gaussian process and TabPFN. The Gaussian process selects its kernel family on the validation set from {RBF, Matérn $\nu = 3/2$, Matérn $\nu = 5/2$, rational quadratic}, its continuous hyperparameters being fitted by marginal likelihood as above. TabPFN selects its inference-time ensemble size from {4, 8, 16, 32}.

More details of hyperparameter optimization are given below:

- Default and tuned model variants. The default model variant of a model has its search disabled; the tuned model variant is the same family with the search enabled. In every search, the library-default configuration is evaluated as candidate 0, so that a tuned model variant may not exhibit a worse MSE than its default counterpart. Consequently, the deficit that the tuned model variant shows on test reflects selection noise rather than the effects of an unfavorable search space. Candidates that raise an exception or return non-finite predictions are skipped and the search continues, which may occasionally occur in the boosting or MLP spaces that are numerically fragile at $n \leq l$. A run fails only if every one of its candidates fails.
- Provenance of search spaces. The parameter spaces for MLP, random forest, SVR, CatBoost, and XGBoost are not chosen by us, but are given by the baseline hyperparameter tables in (Hollmann et al., 2025), and are adopted verbatim. The parameter spaces for the Gaussian process, ridge, LLR, and TabPFN are listed in Tables S9–S12.
- Search budget. The number of candidate fits per run is 51 for XGBoost, CatBoost, random forest, and the MLP: 50 sampled configurations plus the default; 25 for SVR, whose space is categorical and is therefore enumerated exhaustively as a $4\times2\times3$ grid rather than sampled; 9 for ridge and for LLR; 4 for the Gaussian process and for TabPFN; 1 for SML-PN and for NN_m, for which no search is performed. Sampling is seeded separately for each repetition. For the reference models, Tables S2–S12 give the settings held fixed and the spaces searched by the tuned model variants.

4 NETWORK-SYNTHESIZED EXPERIMENT

This experiment is idealized to test whether, when the true parameter configuration $\overline{\mathbf{w}}$ is known to exist, SML-PN can capture the system behavior with a small amount of training data. Because the target is generated by the PN equation, the comparison may be biased in favor of SML-PN. The experiment is therefore reported merely as a controlled recovery test.

In this section, the ground-truth parameter vector $\overline{\mathbf{w}}$ is used only for data generation. It is never accessed by any of the models (SML-PN, tuned log-linear ridge (LLR), or $\mathrm{NN_m}$), and SML-PN never attempts to estimate it, as Eq. (8) in the main text does not depend on any particular parameter configuration $\{w_i\}_{\mathrm{o}}$.

4.1 DATA GENERATION

The training data are produced by the power-law network (PN) for $l = 100$. The number of internal parameters $m = l = 100$. A ground-truth parameter vector $\overline{\mathbf{w}} \in \mathbb{R}^{100}$ is drawn from $\mathscr{N}(-2,1)$. Inputs $\mathbf{x}$ are drawn coordinate-wise from a uniform distribution, $\mathbf{x} \sim \mathscr{U}(0,1)^{100}$ followed by $\mathbf{x} \leftarrow 0.25 + 0.75\ \mathbf{x}$, so that $\mathbf{x} \in [0.25, 1]^{100}$. Then, the target is calculated from a general power law definition as $\overline{y} = \sum_k \overline{w}_k \log x_k$. The configuration and the solver schedule of SML-PN are given in SM Sec. 3.4; LLR uses the tuned configuration of Table S11, and $\mathrm{NN_m}$ is detailed in SM Sec. 2.1.

4.2 SAMPLING, PARTITIONING, AND REPETITIONS

One data pool (160 sets of $\mathbf{x}$ and matching $\overline{y}$) is generated per experiment: 100 candidate training samples, 30 validation samples, and 30 test samples. The validation and test sets are fixed, identical for every training budget and every repetition.

Twenty training budgets are evaluated, $n \in \{5, 10, 15, \ldots, 100\}$, i.e., n/l varies from 0.05 to 1.00. Each of the ten repetitions $r = 0, \ldots, 9$ uses seed $s_r = 42 + r$ to govern the draw of the training samples and every source of model randomness. The n training samples are drawn without replacement from the fixed pool of 100 candidate training samples. Because the draw is realized as a permutation, the training samples of a given repetition are nested across various training budgets. That is, more training samples are added to the existing one as n/l increases.

The repetitions vary the training samples and model initialization, but not the evaluation data or the ground-truth function. Thus, the spread across repetitions is not an estimate of variability over independent problem instances. Moreover, at $n = l = 100$, the training samples consume the entire pool, such that all ten repetitions share an identical training set. This effect is reflected in Fig. S1, as the bands become narrower toward $n/l = 1$.

4.3 RESULTS

Fig. 1 in the main text shows the results of eight of the 20 training budgets as the kernel-density profiles of the test loss of SML-PN, tuned LLR, and $\mathrm{NN_m}$. Each profile is a Gaussian kernel-density estimate over ten repetitions with a bandwidth $0.35\widehat{\sigma}$, where $\widehat{\sigma}$ is the standard deviation for the n training samples, drawn as a filled profile, individually normalized to unit maximum. The vertical tick marks the median value of each training budget (n/l).

Fig. S1 shows the results for all 20 training budgets and additionally reports the same runs evaluated using test R^2. The fit quality, instead of the scale of loss, determines whether the model produces meaningful predictions. $\mathrm{NN_m}$ is sized by the parameter-matching rule $h = \mathit{round}\,((n+2l+1)/(l+2))$. Here, $h = 2$ for $n \leq 50$ and $h = 3$ for $n \geq 55$, and the resulting numbers of trainable parameters are 205 and 307, respectively, close to those of SML-PN ($n+2l+2$ trainable quantities). The step in h at $n = 55$ is reflected by the discontinuity in the $\mathrm{NN_m}$ curves in Fig. S1, which is an artefact of the matching rule, not of the data.

Remarkably, SML-PN can make meaningful predictions with a small training budget as low as $n/l = 0.05$, with a median test R^2 of 0.47. As n increases to 10 and 20, the median test R^2 rises to 0.76 and 0.87, respectively. The median test R^2 saturates near 0.88 as n further increases. In comparison, the median test R^2 of $\mathrm{NN_m}$ lies between -0.80 and -0.04 in the entire range of n/l,

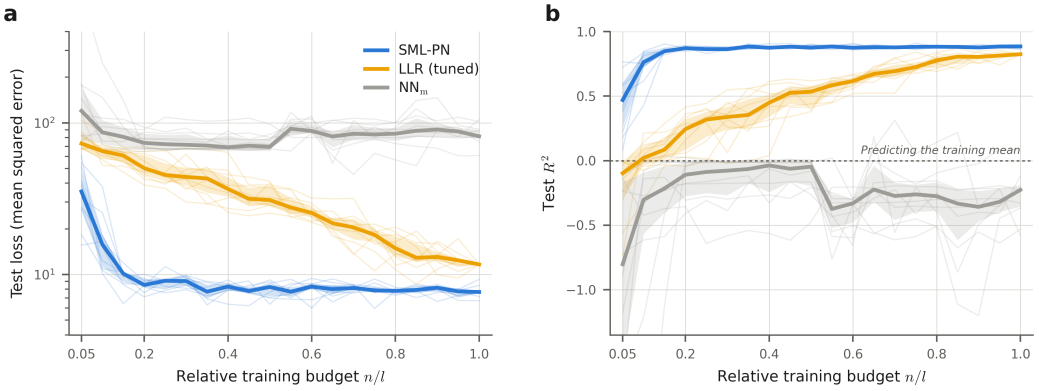

Figure S1: (**a**) Test loss (mean squared error on the 30 held-out samples) against the relative training budget n/l for SML-PN (blue), tuned LLR (orange), and $\mathrm{NN_m}$ (grey). For each model, the faint lines are the ten individual repetitions, and the bold line is their median; the band indicates the interquartile range. (**b**) The test R^2. The dashed line at $R^2 = 0$ is the accuracy of predicting the training mean; $\mathrm{NN_m}$ remains below it, and LLR is comparable with SML-PN only as n approaches l. The step in the $\mathrm{NN_m}$ curves near $n/l = 0.55$ is where the parameter-matching rule changes the hidden-layer width.

i.e., the prediction of $\mathrm{NN_m}$ is never better than the training mean. Tuned LLR, which shares SML-PN's prediction basis, reaches a median test R^2 of only 0.02 at $n = 10$, 0.24 at $n = 20$, and 0.53 at $n = 50$; it approaches SML-PN (0.82 versus 0.88) only at $n = l$, where the training set is the entire pool. Such a result demonstrates that SML-PN can predict results lying inside its reachable parameter space from very few training samples, provided that $\overline{\mathbf{w}}$ exists. Note that the existence of $\overline{\mathbf{w}}$ is a fundamental assumption of most conventional machine-learning algorithms and is not unique to SML. The main difference is that a conventional algorithm searches for $\overline{\mathbf{w}}$, while SML does not.

Fig. S2 shows how SML-PN, LLR, and $\mathrm{NN_m}$ score on the data that they are actually trained on. In HDLSS settings, a conventional machine-learning method that selects a single parameter configuration will typically reproduce the training targets, but generalize poorly to the test data. In contrast, SML-PN's training, validation, and test R^2 are 0.535, 0.480 and 0.469 at $n = 5$, and from $n = 15$ onward the three curves agree to within 0.03. The generalization gap $R^2_{train} - R^2_{test}$ has a median of -0.023 over all n/l, where the subscripts "train" and "test" indicate the median test R^2 on training samples and test set, respectively. LLR fits its training split almost exactly (median training R^2 between 0.86 and 1.00 from $n = 20$ onward) while its validation and test scores lag far behind; its generalization gap has a median of $+0.33$ over all n/l, with a maximum of $+0.63$ at $n = 15$. $\mathrm{NN_m}$ exhibits the opposite trend: its generalization gap reaches $+1.07$ at $n = 5$ and $+0.41$ at $n = l$, and its validation and test scores stay below zero over all n/l. The distinction between SML-PN and the other two models follows directly from Eq. (3), in which each training target is reproduced on average at the ensemble level; no individual parameter configuration is required to fit any training sample. SML-PN does not fit a single parameter configuration: its prediction is the expectation value over ρ_μ, which encodes only the training constraints and assigns probability to every possible configuration, and the numerical solution is the state retained by validation-based stopping (SM Sec. 3.4), reached before the constraints are met exactly. The absence of a generalization gap therefore arises without any regularization tuned for the purpose (there is none).

5 Standardized-benchmark results

Sec. 5.2 of the main text evaluates all 20 model variants on all relevant functions in the Virtual Library of Simulation Experiments (VLSE) (Surjanovic & Bingham, 2013). The benchmark constructions and the testing results are detailed in this section. We also report how broadly the advantage of SML-PN holds and what each model variant does when it fails.

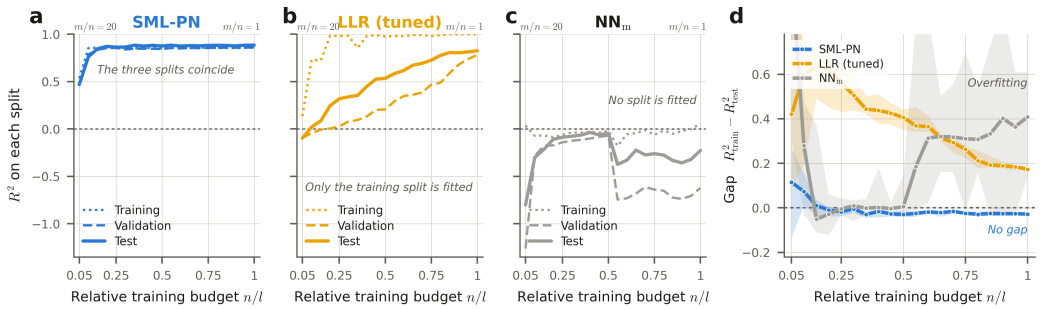

Figure S2: Fitting without overfitting on network-synthesized data ($l = 100$). **(a)** Median R^2 over the ten repetitions for SML-PN, **(b)** tuned LLR, and **(c)** NN_m on the three data splits: training (dotted), validation (dashed) and test (solid) sets. For SML-PN, the three curves are indistinguishable from $n/l = 0.15$ onward; LLR fits only its training split. The ratio of the number of internal parameters to the number of training samples, marked above each panel, runs from $m/n = 20$ at the smallest training budget to $m/n = 1$ at the largest training budget. **(d)** The generalization gap $R^2_{train} - R^2_{test}$ (median of the ten repetitions), with the bands indicating the interquartile ranges. The dashed line at zero marks equal accuracy on seen and unseen data; SML-PN tracks it at every n/l, while LLR and NN_m cannot. The band of NN_m is clipped at the top of the panel at the smallest training budgets.

5.1 The 13 functions and the evaluation protocol

There are 13 functions in VLSE with continuous inputs and adjustable input dimensionality. All of them are tested in the present investigation. The input dimension is $l = 20, 50, 100$ or 500, yielding 52 function-dimension units. Table S13 shows these functions. Inputs are drawn coordinate-wise from $\mathscr{U}(0.1, 1)$, so that each unit samples the hypercube $[0.1, 1]^l$. The calculated targets are noise-free. The choice of input range is examined in SM Sec. 5.5.

For each function-dimension unit, one data pool is generated at seed 42, containing l candidate training rows plus 30 validation rows and 30 test rows (each row indicates a generated dataset of $\mathbf{x}$ and $\bar{y}$). The validation and test sets are fixed across all training budgets and repetitions, while the ten repetitions draw nested training sets from the pool, as described in SM Sec. 4.2. With ten training budgets $n/l = 0.1, 0.2, \ldots, 1.0$ and ten repetitions, there are 520 evaluation cells and 104,000 individual training runs for the 20 model variants.

In this study, a function is considered learnable if at least one model variant achieves a median test R^2 greater than 0.2. Among the 13 VLSE functions, four do not meet this requirement: Sum Powers, Dixon-Price, Rosenbrock, and Rastrigin. For them, the best median test R^2 are 0.17, 0.08, -0.03, and -0.07, respectively (see Tables S14 and S15). Performance rankings of the model variants on such functions merely reflect which model shrinks most strongly toward the training mean, rather than which model actually learns. Therefore, these four functions are excluded from our analysis. Compared with other VLSE functions, there are not statistically significant differences in cross-coordinate interactions or high-frequency oscillations.

The other 9 functions form the analysis set. For each of them, the highest median test R^2 is at least 0.67. Unless stated otherwise, all results reported here are computed over the analysis set, comprising 36 function-dimension units, 360 evaluation cells, and 72,000 individual runs. Figure 2(a) in the main text and Fig. S4 present the performance of these functions, and SM Sec. 5.3 explains why the four excluded functions are unreachable.

Within each (function-dimension unit, training budget) cell, the 20 model variants are ranked 1-20 on their mean test R^2 over ten repetitions, and the cell R^2 is likewise the mean over the ten repetitions. Per-unit and per-function statistics are means of the cell ranks.

Table S13: The 13 VLSE functions. "Cat." indicates the SML-PN outcome category in SM Sec. 5.3 (A: learned, B: partially learned, C: not learned).

VLSE Function	**Formula**	**Cat.**
Schwefel	$418.9829\,l - \sum_k x_k \sin\left(\sqrt{\lvert x_k \rvert}\right)$	A
Trid	$\sum_k (x_k - 1)^2 - \sum_{k \geq 2} x_k x_{k-1}$	A
Sphere	$\sum_k x_k^2$	A
Ackley	$-20\exp\left(-0.2\sqrt{\sum_k x_k^2 / l}\right) - \exp\left(\sum_k \cos(2\pi x_k)/l\right) + 20 + e$	A
Rotated Ellipsoid	$\sum_i \sum_{k \leq i} x_k^2$	A
Sum Squares	$\sum_k k\, x_k^2$	A
Levy	$\sin^2(\pi w_1) + \sum_{k<l} (w_k - 1)^2 \left[1 + 10\sin^2(\pi w_k + 1)\right] +$ $(w_l - 1)^2 \left[1 + \sin^2(2\pi w_l)\right]$, with $w_k = 1 + (x_k - 1)/4$	A
Zakharov	$\sum_k x_k^2 + S^2 + S^4$, with $S = \sum_k 0.5\, k\, x_k$	A
Griewank	$1 + \sum_k x_k^2/4000 - \prod_k \cos\left(x_k/\sqrt{k}\right)$	A
Sum Powers	$\sum_k \lvert x_k \rvert^{k+1}$	B
Dixon-Price	$(x_1 - 1)^2 + \sum_{k \geq 2} k \left(2x_k^2 - x_{k-1}\right)^2$	B
Rosenbrock	$\sum_{k<l} \left[100\left(x_{k+1} - x_k^2\right)^2 + (1 - x_k)^2\right]$	C
Rastrigin	$10l + \sum_k \left[x_k^2 - 10\cos(2\pi x_k)\right]$	C

Table S14: Mean rank (1 = best of 20) of every model variant on each function, pooled over the four input dimensions and the ten training budgets.*

	Function												
Model variant	*Schwefel*	*Trid*	*Sphere*	*Ackley*	*Rotated Ellipsoid*	*Sum Squares*	*Levy*	*Zakharov*	*Griewank*	*Sum Powers*	*Dixon-Price*	*Rosenbrock*	*Rastrigin*
SML-PN	1.00	1.00	1.00	1.02	1.08	1.15	1.23	1.32	9.45	1.68	3.32	8.52	10.98
SVR	3.30	2.72	3.22	3.10	4.40	4.75	3.75	4.50	12.25	6.15	5.05	5.92	4.95
Ridge	3.50	4.95	4.00	6.55	3.70	3.75	4.30	4.32	9.15	8.43	6.78	5.75	7.78
$\text{Ridge}_{\text{def}}$	4.45	4.82	4.72	7.25	4.30	4.28	4.20	4.97	10.07	13.52	17.60	17.98	14.78
LLR	4.80	4.03	5.55	5.20	5.42	4.88	4.82	4.10	10.78	8.82	7.10	5.52	8.10
LLR_{def}	6.48	6.35	7.92	12.42	7.95	7.25	7.55	6.60	13.02	18.88	19.18	19.40	18.82
MLP	13.02	13.05	5.15	3.28	5.85	6.55	13.35	7.35	11.95	5.48	6.20	4.58	11.18
GP	9.12	8.88	9.78	9.32	9.75	9.07	10.32	8.72	15.05	9.43	5.40	3.70	7.12
TabPFN	11.40	10.95	11.55	9.88	11.85	11.68	9.68	11.38	2.30	9.15	9.50	11.12	9.20
GP_{def}	9.57	9.43	10.07	9.07	10.22	9.48	10.95	9.55	15.68	10.50	5.68	5.18	8.10
SVR_{def}	9.25	9.60	9.72	9.88	9.95	10.35	10.65	10.38	16.52	6.72	5.38	6.62	4.08
$\text{TabPFN}_{\text{def}}$	11.98	11.70	12.32	10.52	12.62	12.28	10.12	12.40	3.32	9.57	10.00	11.40	9.90
XGB	12.85	13.00	13.20	14.88	11.42	13.80	11.38	13.45	3.18	11.20	14.32	13.58	10.90
RF	12.65	12.55	12.78	12.28	13.30	12.92	12.78	13.35	6.82	7.02	9.15	8.00	7.30
RF_{def}	12.78	13.22	13.80	11.72	14.48	13.40	12.78	13.85	7.38	9.72	12.95	13.35	10.30
CatB_{def}	13.72	13.70	14.55	13.95	13.48	14.15	13.65	14.08	9.95	9.22	8.72	7.35	6.80
CatB	14.40	14.20	14.95	14.90	15.00	14.88	14.22	14.30	9.05	13.98	13.32	13.10	11.50
XGB_{def}	16.88	17.00	17.25	16.95	16.92	17.35	15.38	17.58	5.75	13.95	14.02	12.98	12.35
NN_{m}	19.00	19.00	18.95	18.52	18.82	18.58	19.05	18.38	18.80	17.48	17.42	17.02	16.62
MLP_{def}	19.85	19.85	19.50	19.30	19.48	19.48	19.85	19.42	19.52	19.10	18.90	18.92	19.25

* Sum Powers, Dixon-Price, Rosenbrock, and Rastrigin are excluded from every aggregate.

Table S15: Median test R^2 of every model variant on each function, over the same cells.

	Function												
Model variant	*Schwefel*	*Trid*	*Sphere*	*Ackley*	*Rotated Ellipsoid*	*Sum Squares*	*Levy*	*Zakharov*	*Griewank*	*Sum Powers*	*Dixon-Price*	*Rosenbrock*	*Rastrigin*
SML-PN	0.976	0.954	0.894	0.665	0.714	0.851	0.752	0.853	0.285	0.168	0.075	–0.097	–0.248
SVR	0.542	0.519	0.512	0.332	0.346	0.435	0.463	0.494	0.251	–0.016	–0.019	–0.064	–0.073
Ridge	0.528	0.495	0.451	0.186	0.374	0.478	0.423	0.539	0.341	–0.026	–0.036	–0.055	–0.117
$\text{Ridge}_{\text{def}}$	0.493	0.503	0.452	0.157	0.346	0.489	0.426	0.537	0.270	–0.232	–0.387	–0.702	–0.456
LLR	0.527	0.505	0.427	0.201	0.344	0.462	0.421	0.544	0.313	–0.035	–0.036	–0.055	–0.123
LLR_{def}	0.441	0.394	0.339	–0.051	0.214	0.352	0.344	0.435	0.221	–0.797	–1.259	–1.593	–1.020
MLP	–0.061	–0.058	0.383	0.240	0.253	0.281	0.006	0.269	0.234	0.011	0.001	–0.043	–0.141
GP	0.267	0.270	0.228	0.133	0.169	0.272	0.109	0.320	0.167	–0.040	–0.036	–0.034	–0.090
TabPFN	0.069	0.067	0.053	0.040	–0.008	0.102	0.096	0.083	0.789	–0.040	–0.097	–0.091	–0.138
GP_{def}	0.270	0.270	0.231	0.147	0.157	0.281	0.109	0.320	0.167	–0.051	–0.034	–0.037	–0.106
SVR_{def}	0.145	0.142	0.144	0.103	0.127	0.136	0.120	0.159	0.083	–0.003	–0.001	–0.069	–0.068
$\text{TabPFN}_{\text{def}}$	0.066	0.057	0.051	0.040	–0.011	0.100	0.084	0.078	0.785	–0.041	–0.099	–0.093	–0.138
XGB	0.014	0.022	0.011	–0.017	0.041	–0.017	0.077	0.011	0.620	–0.092	–0.164	–0.225	–0.194
RF	0.064	0.076	0.062	0.036	–0.032	0.056	0.054	0.066	0.560	–0.003	–0.058	–0.073	–0.108
RF_{def}	–0.012	–0.009	–0.037	0.024	–0.065	0.024	–0.007	–0.023	0.560	–0.052	–0.125	–0.125	–0.134
CatB_{def}	0.002	0.032	–0.007	–0.010	–0.047	–0.015	0.020	0.023	0.402	–0.019	–0.053	–0.057	–0.100
CatB	–0.016	–0.015	–0.026	–0.024	–0.088	–0.008	0.002	0.017	0.394	–0.114	–0.137	–0.156	–0.176
XGB_{def}	–0.121	–0.122	–0.112	–0.095	–0.159	–0.085	–0.018	–0.075	0.606	–0.147	–0.139	–0.146	–0.178
NN_{m}	–1.003	–0.974	–0.476	–0.377	–0.580	–0.354	–0.608	–0.482	–0.055	–0.629	–0.511	–0.626	–0.624
MLP_{def}	–1.423	–1.344	–0.784	–0.625	–1.036	–0.411	–1.536	–0.473	–0.431	–1.056	–1.421	–1.146	–1.266

Table S16: Paired Wilcoxon signed-rank tests of SML-PN against the reference model variants across the 36 per-unit mean ranks of the analysis set, Holm-corrected over the 19 comparisons.*

Model variant	Mean rank	Rank gap	Units won by SML-PN	W	p	p_{holm}	Significant
MLP_{def}	19.58	17.56	36	0	1.58e-07	3.01e-06	yes
NN_{m}	18.79	16.76	36	0	1.66e-07	3.01e-06	yes
SVR_{def}	10.70	8.67	35	1	1.81e-07	3.08e-06	yes
GP_{def}	10.45	8.42	35	1	1.82e-07	3.08e-06	yes
GP	10.00	7.98	35	1	1.82e-07	3.08e-06	yes
MLP	8.84	6.81	35	2	1.99e-07	3.08e-06	yes
LLR_{def}	8.39	6.37	34	6	2.78e-07	3.62e-06	yes
XGB_{def}	15.67	13.64	33	8	3.25e-07	3.90e-06	yes
CatB	13.99	11.96	34	13	4.96e-07	5.45e-06	yes
RF_{def}	12.60	10.57	33	17	6.86e-07	6.86e-06	yes
CatB_{def}	13.47	11.44	34	26	1.41e-06	1.27e-05	yes
RF	12.16	10.13	33	29	1.78e-06	1.42e-05	yes
SVR	4.67	2.64	35	33	2.43e-06	1.70e-05	yes
LLR	5.51	3.48	35	35	2.83e-06	1.70e-05	yes
XGB	11.91	9.88	33	36	3.06e-06	1.70e-05	yes
$\text{Ridge}_{\text{def}}$	5.45	3.42	34	37	3.30e-06	1.70e-05	yes
Ridge	4.91	2.89	34	38	3.55e-06	1.70e-05	yes
$\text{TabPFN}_{\text{def}}$	10.81	8.78	33	45	6.03e-06	1.70e-05	yes
TabPFN	10.07	8.04	33	52	1.01e-05	1.70e-05	yes

* Subscript "def" denotes the library-default configuration of a reference model; the per-run tuned model variant does not have any subscript. Subscript "m" of NN_{m} denotes the feed-forward control matched to SML-PN in trainable-parameter count. "W" denotes the Wilcoxon signed-rank statistic computed over the 36 per-unit mean ranks, and p_{Holm} is the corresponding p-value after Holm correction across the 19 comparisons. A corrected p below 0.05 indicates that SML-PN differs significantly from the corresponding model variant.

5.2 STANDINGS ACROSS THE ANALYSIS SET

Fig. S3 gives the standings of all 20 model variants, which may be compared to Fig. 3(a) in the main text. SML-PN attains a mean rank of 2.03 of 20 across the 360 (function-dimension unit, training budget) evaluation cells, the best of the 20. The mean reflects the typical performance: the median of the 36 per-unit mean ranks of SML-PN is 1.00, with an interquartile range of 1.00-1.15. That is, on most units SML-PN is the best model at every training budget. SML-PN is the outright best-performing model on 33 of the 36 function-dimension units (91.7%). The second-best model variant is tuned TabPFN, with a rate of 8.3% (on the three Griewank units). Paired Wilcoxon signed-rank tests over the 36 per-unit mean ranks (Holm-corrected across the 19 comparisons) place every other model variant significantly below SML-PN (largest corrected $p = 1.7 \times 10^{-5}$). Table S16 gives the full results.

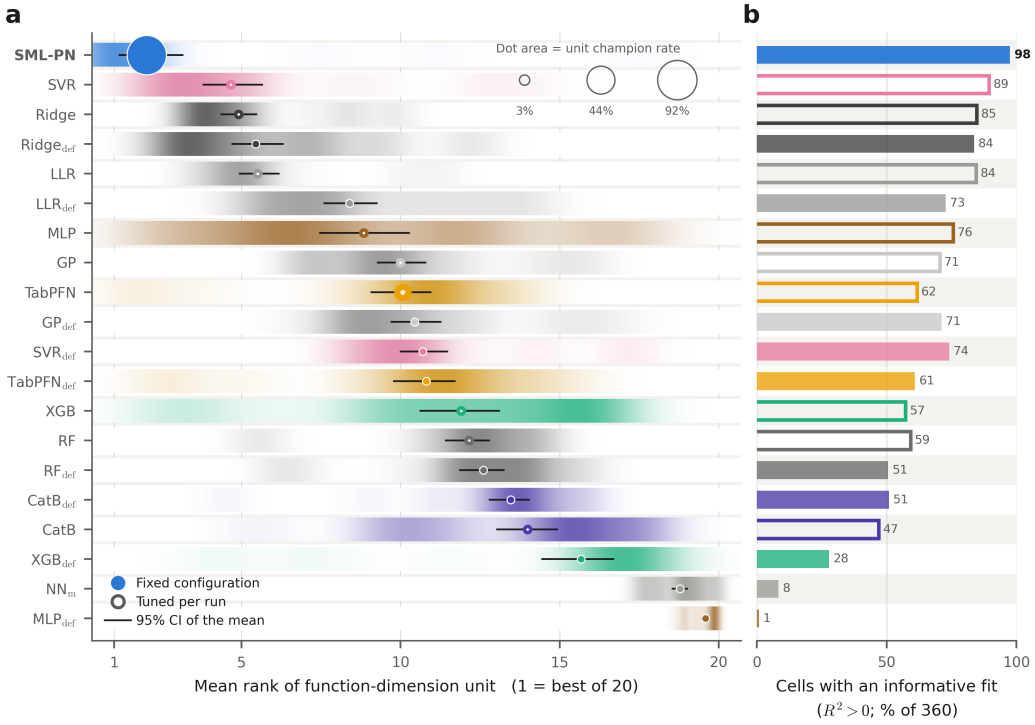

Figure S3: Standings of the 20 model variants across the VLSE function-dimension units, computed over the analysis set, i.e., the nine learnable functions (36 units, 360 cells). **(a)** Distribution of the 36 per-unit mean ranks of each model variant, drawn as a density strip normalized to unit maximum in each row; the dot marks the mean and the black line is the 95% bootstrap interval; the dot area indicates the unit champion rate of the model variant. Empty symbols denote the tuned model variants that perform fresh hyperparameter search in every run; solid symbols denote the model variants with fixed configuration (denoted by subscript "def"); SML-PN has a fixed configuration. **(b)** Percentage of the 360 evaluation cells in which the model variant achieves $R^2 > 0$. Paired Wilcoxon signed-rank tests with Holm correction place all 19 reference model variants significantly below SML-PN (the largest corrected $p = 1.7 \times 10^{-5}$); see Table S16 for more details.

Fig. S3(b) reports how often a model variant produces an informative fit (if it can make any meaningful prediction). Ranking is relative, and the ranking of an evaluation cell where no model variant works is uninformative. SML-PN achieves $R^2 > 0$ (i.e., better than the training mean) in 351 of the 360 cells (97.5%), more than any other model variant: SVR 89%, ridge 85%, the log-linear ridge 84%, the tuned MLP 76%, TabPFN 62%, tuned XGBoost 57%, tuned CatBoost 47%, and the parameter-matched NN_m 8%.

Table S17: Mean rank of every model variant as a function of the input dimension l, over the nine-function analysis set.

	l				
Model variant	20	50	100	500	All
SML-PN	1.07	1.82	2.38	2.84	2.03
SVR	4.87	4.47	4.88	4.46	4.67
Ridge	4.67	5.14	5.18	4.67	4.91
$\text{Ridge}_{\text{def}}$	7.67	4.92	4.11	5.11	5.45
LLR	5.43	5.81	5.62	5.17	5.51
LLR_{def}	8.60	9.21	7.56	8.21	8.40
MLP	6.28	7.98	10.64	10.46	8.84
GP	11.83	9.97	10.24	7.97	10.00
TabPFN	10.89	10.34	9.22	9.83	10.07
GP_{def}	12.41	10.00	10.61	8.77	10.45
SVR_{def}	10.98	11.53	10.43	9.86	10.70
$\text{TabPFN}_{\text{def}}$	12.12	11.26	9.76	10.10	10.81
XGB	9.30	11.23	12.56	14.53	11.90
RF	12.10	11.48	11.89	13.17	12.16
RF_{def}	13.84	12.13	12.07	12.36	12.60
CatB_{def}	13.41	14.01	13.89	12.57	13.47
CatB	10.47	14.37	15.03	16.09	13.99
XGB_{def}	15.22	15.79	15.92	15.76	15.67
NN_{m}	19.61	19.19	18.12	18.23	18.79
MLP_{def}	19.23	19.34	19.89	19.87	19.58

Table S17 presents the standings as a function of the input dimension l. As l grows from 20 to 500, the tuned gradient-boosting models deteriorate sharply: XGBoost from a mean rank of 9.3 to 14.5, and CatBoost from 10.5 to 16.1. SML-PN moves from 1.07 to 2.84. The mean ranks of SVR, ridge, and log-linear ridge remain roughly flat.

5.3 Per-function performance

Fig. S4 gives the per-function results for all 13 functions, including the four unlearnable ones excluded from the analysis set. Tables S14 and S15 give the underlying per-function mean ranks and median test R^2 for all 20 model variants.

For the nine-function analysis set, the overall result clearly demonstrates that SML-PN is the best model. On eight of the nine functions (Schwefel, Trid, Sphere, Ackley, Rotated Ellipsoid, Sum Squares, Levy and Zakharov), SML-PN attains a median test R^2 between 0.67 and 0.98 and a mean rank between 1.00 and 1.32, while the strongest reference model on each function (most often SVR or ridge) reaches only 0.33 to 0.54.

The ninth function, Griewank, is the boundary case, the only one in the analysis set on which SML-PN is outperformed by other model variants. TabPFN reaches a median R^2 of 0.79, whereas median R^2 of SML-PN is 0.29. The shortfall is not one of form: over the sampling range, the cosine product carries almost all of Griewank's variance, and the function is almost entirely (98.9%) additive (Sobol', 1993; Saltelli et al., 2008; Hoeffding, 1948). Thus, its target is of the kind that SML-PN's additive prediction represents directly. What sets Griewank apart is how it distributes the signal. The $\sqrt{k}$ term in the cosine function makes the first-order importance of coordinate k decay as $\frac{1}{\sqrt{k}}$: at $l = 500$ the first ten inputs carry 74% of the first-order variance and the last hundred carry only 5%. This concentration, rather than the input dimension itself, drives the dimensional trend. SML-PN assigns one interconnection factor per input and carries no penalty term, which fixes the number of unknowns at n without tuning. Penalized estimators can damp low-signal inputs at the cost of a regularization constant, and partitioning models can decline to select them. The two trends separate as the number of low-signal coordinates grows. Pooled over the ten training budgets, SML-PN leads the field at $l = 20$ with a median test R^2 of 0.42 against 0.33 for the strongest reference model, and gives way to 0.40, 0.30 and 0.10 at $l = 50$, 100 and 500. Its predictions stay informative throughout, the pooled median R^2 being 0.29, but the ranking is lower than that of some other models.

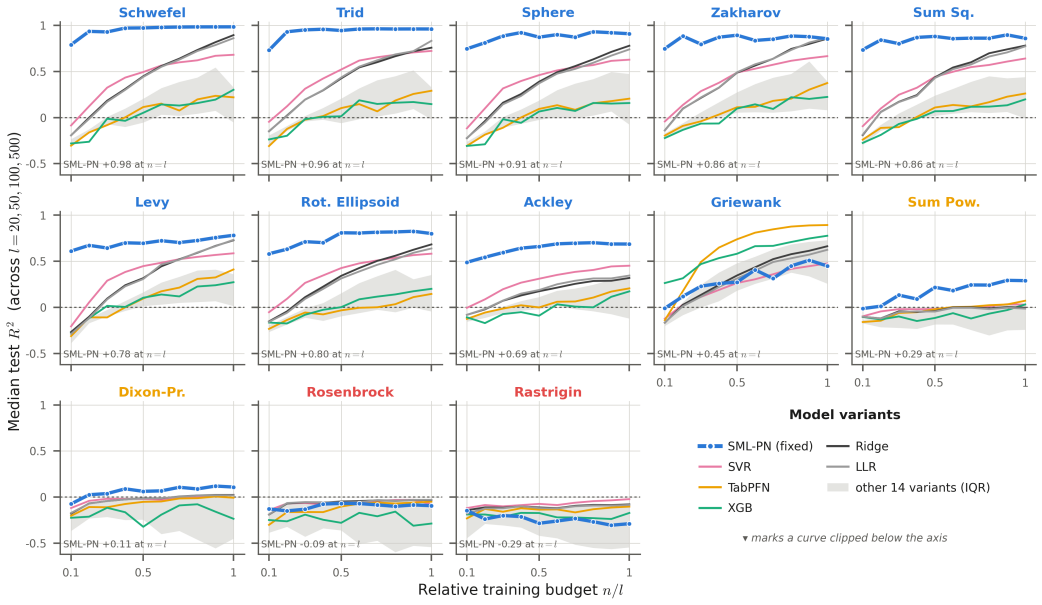

Figure S4: Per-function learning curves for all 13 functions. Median test R^2 is plotted against the relative training budget n/l, taken across four different input dimensions. Six model variants are compared: SML-PN, the second-best reference model (SVR), the two linear controls (ridge and the log-linear ridge that shares the prediction basis of SML-PN), TabPFN, and tuned XGBoost. The grey band indicates the interquartile range of the other 14 model variants. The last four panels show the test results of the unlearnable functions excluded from the analysis set (Sum Pow., Dixon-Pr., Rosenbrock, and Rastrigin).

SML-PN is not blind to relevance: Section 6 of the main text shows that the interconnection factors concentrate on informative inputs unaided, retaining about two thirds of the total $|\varepsilon_k|$ where an even spread would give one half. That tilt is graded rather than selective, however, and it is measured against appended features carrying no signal at all, whereas the weak coordinates of Griewank still carry signal: at the largest input dimension a competitive fit would have to concentrate far more steeply than two thirds. We tested this effect by replacing $\sqrt{k}$ with a constant, the geometric mean of $\sqrt{1} \ldots \sqrt{l}$. The product form, the quadratic term, the sampling domain, and the protocol remain unchanged, while the coordinates are exchangeable: on the control, the first ten inputs carry 2.1% of the first-order variance at $l = 500$, against 2.0% for an even spread. With the training budget equal to the input dimension and ten repetitions, SML-PN achieves a median test R^2 of 0.95, 0.93, 0.88 and 0.94 at $l = 20$, 50, 100 and 500 on the control, a change of -0.01 across the range, against 0.64, 0.64, 0.23 and 0.12 on Griewank itself under the identical protocol. Since the two targets differ only in how first-order importance is distributed across coordinates, the decline is attributable to that distribution alone, not to the multiplicative form of the target, the sampling domain, or the number of training samples. In other words, Griewank penalizes the property of SML-PN that is beneficial elsewhere. The same construction of SML-PN leads on the other eight learnable functions and places SML-PN first among the fixed-configuration model variants across the 848 real-world datasets (see SM Sec. 6).

The four unlearnable functions are excluded from the analysis set. On Sum Powers and Dixon-Price, the best any model variant could achieve is a median R^2 of $+0.17$ and $+0.08$, respectively, both by SML-PN; on Rosenbrock and Rastrigin, the best is -0.03 and -0.07, respectively. The performance is strongly dependent on the input dimension: at $l = 20$, the best of the field does clear the moderate-fit threshold on Sum Powers (0.42) and Rastrigin (0.26), and in both cases the best model is SML-PN, ahead of the next model variant by 0.25 and 0.13, respectively. The outcome categories in Table S13 report this division. Category A is the nine functions of the analysis set, on which SML-PN clears the threshold; category B, Sum Powers and Dixon-Price, is where it recovers

some signal but stays below the threshold over the grid; category C, Rosenbrock and Rastrigin, is where the best of the field is negative over the grid, no better than predicting the training mean.

5.4 Breadth of the advantage, and the shape of failure

Fig. S5 reports the details that are not reflected by the mean ranks.

- Breadth. Figure S5(a) presents, for each training budget and each model variant, the fraction of the 36 units on which SML-PN attains the lowest test root mean square error (RMSE). The color scale is centered at 50%; a gold cell indicates SML-PN is outperformed by other model variants. Across all 190 model-budget combinations, the RMSE value ranges from 78% to 100%, and none is below 50%. Taken over all 360 cells, SML-PN attains the lowest RMSE on 92% of them against ridge and the log-linear ridge, the two reference models that share its prediction basis, and on 95% against SVR. The advantage is global.
- The scarcest budget. Figure S5(b) focuses on $n/l = 0.1$, i.e., only one training sample per ten input features. SML-PN achieves $R^2 > 0$ on 81% of the 36 units, against 39% for the tuned MLP, 31% for SVR, 22% for ridge, 17% for tuned XGBoost, 14% for TabPFN, and 0% for the Gaussian process and $\mathrm{NN_m}$. Where data are scarcest, SML-PN is by a wide margin the model most likely to extract any signal at all.
- The shape of failure. Figure S5(c) takes all 72,000 individual runs of the analysis set and demonstrates how often each model variant fails completely, i.e., the test R^2 is below -1, a prediction whose squared error exceeds the target variance. SML-PN fails in 0.22% of all runs, the lowest rate of all 20 model variants, and its worst R^2 is -2.9. In comparison, the parameter-matched $\mathrm{NN_m}$ fails in 21.0% of runs with a worst R^2 of -1.9×10^5, and the default MLP fails in 38.3% of runs.

Fig. S5 supports the claim in Sec. 7 of the main text. Across the 800 runs on Rosenbrock and Rastrigin, the two unlearnable functions that no model fits, SML-PN's errors remain bounded: its R^2 never falls below -1.45, whereas the test R^2 of $\mathrm{NN_m}$ is -5.7×10^4 and the default MLP reaches -8.5×10^4. This result does not imply that SML-PN collapses toward the training mean while others scatter. In fact, the well-regularized model variants also remain bounded, and in SM Sec. 4 it is $\mathrm{NN_m}$, not SML-PN, that contracts toward a constant. The prediction of SML-PN is based on the principle of maximum entropy and is the expectation value over the normalized ρ_μ. Since the expectation value of a bounded quantity does not diverge, the failure mode is a loss of resolution rather than an unbounded excursion.

5.5 The input range

VLSE reports a canonical domain for each function, i.e., the box over which the function is defined (see Table S18). Those boxes are wider than the restricted ranges used in Sec. 5.2 in the main text. We repeated the experiment twice on all 13 functions, with the only difference being the box from which inputs are randomly drawn: one on each function's canonical domain, and the other on 1/10 of the canonical domain. Figure S6 reports all three settings: the restricted range used in this investigation, the canonical domain, and 1/10 of the canonical domain.

Table S18: Canonical domain of each function, per coordinate, as reported by VLSE (Surjanovic & Bingham, 2013).

Function	Canonical domain	Function	Canonical domain
Schwefel	$[-500, 500]$	Griewank	$[-600, 600]$
Trid	$[-l^2, l^2]$	Sum Powers	$[-1, 1]$
Sphere	$[-5.12, 5.12]$	Dixon-Price	$[-10, 10]$
Zakharov	$[-5, 10]$	Rosenbrock	$[-5, 10]$
Sum Squares	$[-10, 10]$	Rastrigin	$[-5.12, 5.12]$
Levy	$[-10, 10]$	Rotated Ellipsoid	$[-65.536, 65.536]$
Ackley	$[-32.768, 32.768]$		

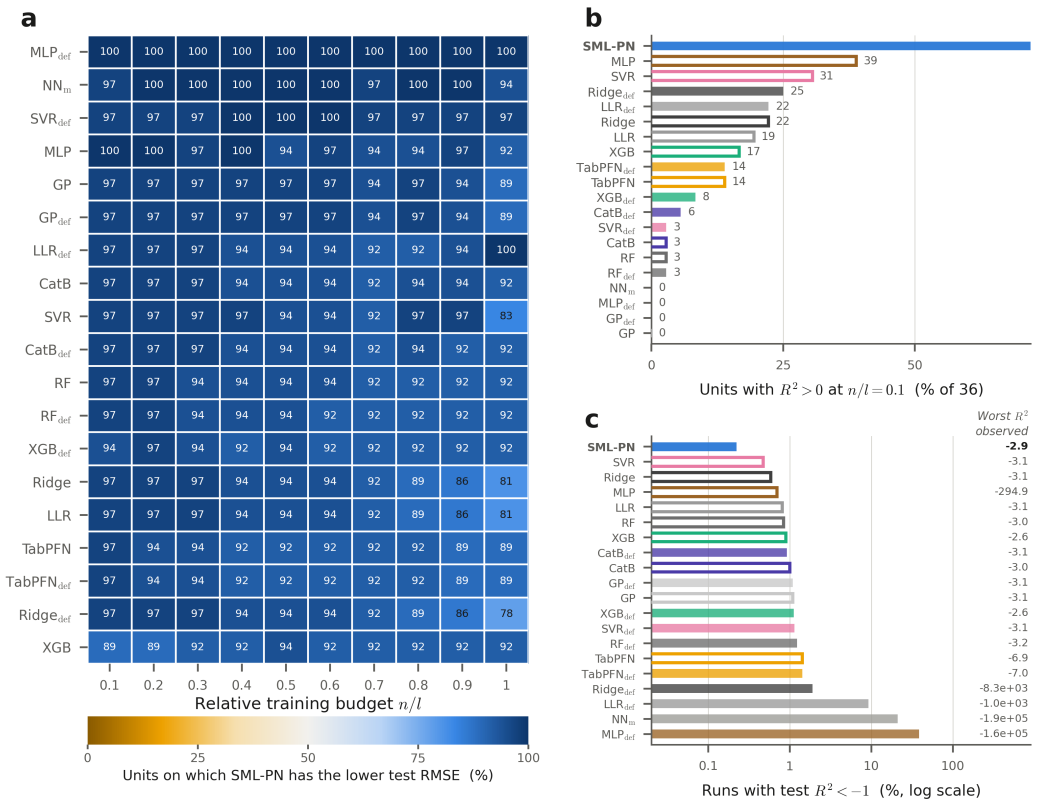

Figure S5: **(a)** For each model variant and each relative training budget, the fraction of the 36 units on which SML-PN attains the lower test RMSE. The subscript "def" denotes the default model variant; the absence of the subscript denotes the tuned model variant (except for SML-PN). The color scale is centered at 50%. **(b)** At the scarcest training budget $n/l = 0.1$, the fraction of units on which each model variant achieves $R^2 > 0$. **(c)** Over all 72,000 individual runs, the fraction for which the test R^2 falls below -1. The worst observed R^2 of each model variant is given in the column on the right-hand side.

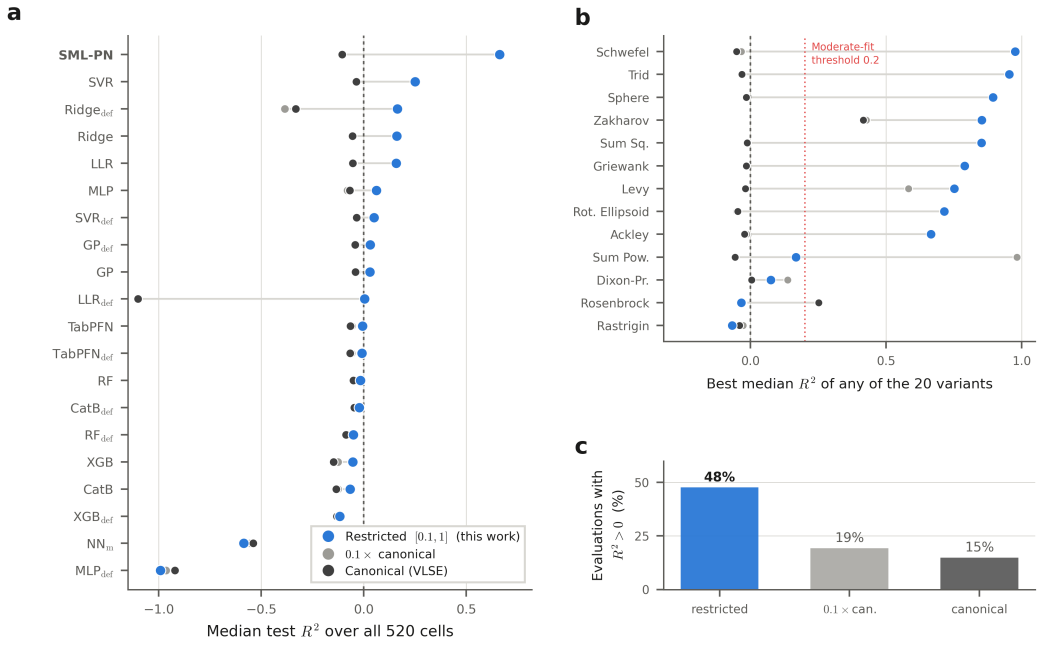

Figure S6: The input range and the function's capability to distinguish different models. The test is repeated on three different input ranges: each function's canonical domain, 1/10 of the canonical domain, and the restricted range $[0.1, 1]$ used in Sec. 5.2 in the main text. **(a)** Median test R^2 of each of the 20 model variants over all 520 evaluation cells of the full 13-function grid. The dashed line is $R^2 = 0$. The subscript "def" denotes the default model variant; the absence of the subscript denotes the tuned model variant (except for SML-PN). **(b)** Per function, the best median R^2 attained by any of the 20 model variants. The dotted line is the moderate-fit threshold 0.2. **(c)** Fraction of all (cell, model variant) evaluations that reach $R^2 > 0$.

In Fig. S6, the performance of all model variants drastically decreases as the input range is widened from the restricted range to the canonical domain or 1/10 of the canonical domain. The issue is the sampling density, rather than the properties of the functions or the model variants. Each unit supplies at most $n = l$ training samples ($l \geq 20$). The volume of a hypercube of side a grows as a^l, so that widening the box can greatly multiply the sampling volume. For instance, at $l = 20$, a tenfold widening inflates the volume by a factor of 10^{20}. The training set becomes sparse relative to the variation of the target, and no estimator can recover a structure that is not sufficiently sampled. In other words, what limits the test is the amount of target variation relative to n.

It is not reasonable to expect any model to extract meaningful predictions in such settings. Over their canonical domains, several VLSE functions oscillate many times along each coordinate (e.g., Rastrigin, Ackley, and Griewank) or vary over many orders of magnitude (e.g., Rosenbrock and Zakharov), whereas in the HDLSS regime of this study even the largest training budget, $n = l$, provides only one training sample per input dimension. A benchmark on which every model fails cannot distinguish the models from one another. The present study therefore uses one fixed restricted range, $[0.1, 1]$ per coordinate, identical for all functions and all model variants, because the results obtained on the canonical domain or on 1/10 of it primarily reflect data scarcity rather than intrinsic properties of the model variants. Nonetheless, it is interesting to note that in the few cases where either wider setting leaves anything learnable at all, SML-PN is usually the model that learns it. Of the seven (function, setting) combinations in which any model variant attains a positive best-of-field median R^2, SML-PN accounts for five: Zakharov at $+0.426$ and $+0.416$, Levy at $+0.582$, Dixon-Price at $+0.137$, and Rosenbrock at $+0.252$.

6 Real-world benchmark results

Sec. 5.3 of the main text evaluates the 20 model variants on real-world regression datasets from OpenML (Vanschoren et al., 2014). This section documents the corpus (SM Sec. 6.1) and the protocol (SM Sec. 6.2), gives the detailed report on every model variant that the main text summarizes (SM Sec. 6.3), and presents the performance of SML-PN as a function of the training budget n (SM Sec. 6.4) and the input dimension l (SM Sec. 6.5). SM Secs. 6.6–6.7 compare the model scales.

6.1 The corpus

OpenML offers 2,136 datasets (case studies) for supervised-regression tasks, restricted to active tasks with a target and deduplicated by dataset. Among these, three groups are irrelevant to our investigation on SML-PN. In the first group, 1,132 datasets are stored in sparse format (chemical-fingerprint and bag-of-words tables), 1,081 of which are from a single assay family, whereas the present study concerns dense tabular regression and represents each dataset as a dense numerical matrix. In the second group, 126 datasets have too few rows (input-output pairs) to support even the smallest training budget, after reserving 60 held-out rows. In the third group, 21 datasets are unavailable or malformed, or carry only a single input feature. Of the 857 datasets that remained, nine had not completed the full evaluation grid when we began our research.

The final corpus consists of 848 datasets. Figure S7 illustrates the above consideration. After preprocessing, the input dimension ranges from 2 to 1,140, while the number of rows (input-target pairs) ranges from 62 to 10^7. The distribution of l is relatively skewed: the median is 12, the interquartile ranges from 8 to 26, and two thirds of the datasets have $l \leq 20$.

6.2 The protocol

The protocol in SM Sec. 3 is applied on the corpus, with l being a property of the dataset rather than a design choice. No training set contains fewer than two samples. For each l, up to ten training budgets $n = round(q \cdot l)$ ($q = 0.1 \ldots 1.0$) are tested, unless l is too small. As shown in Fig. S7(d), 364 of the 848 datasets have sufficient rows for testing ten n/l; 24 datasets have insufficient rows to even reach $n = l$.

The corpus yields 6,777 evaluation cells and 1,461,840 individual training runs. A cell is one (dataset, training budget) combination, scored and ranked on the mean over ten repetitions. Of

the 848 datasets, 841 are tested by all 20 model variants. On the remaining 7 datasets, at least one reference model fails to produce a finite prediction and is therefore excluded from the ranking.

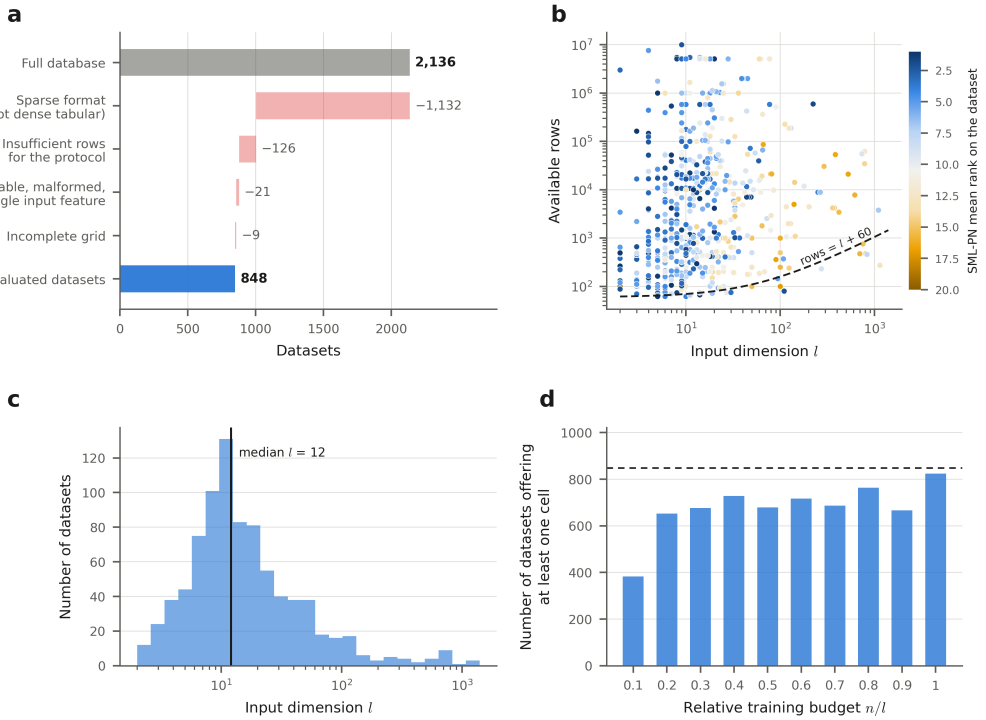

Figure S7: The OpenML corpus. **(a)** From the discovery query (the full database) to the corpus of the present investigation. A total of 1,288 datasets are excluded from our analysis because they are not relevant to the comparison study; no model performance is evaluated on them. **(b)** The available rows (the number of input-output pairs) and the input dimension l of the datasets. The dashed line marks the minimum requirement. The color of each dot (dataset) represents the mean rank of SML-PN. **(c)** Distribution of l of all datasets in the corpus. The median is 12. **(d)** The distribution of the datasets in the corpus that provide at least one evaluation cell at each relative training budget.

6.3 Standings

Table S19 gives the details behind Fig. 3(a) in the main text: for every model variant, the mean and median ranks over the 848 OpenML datasets, the champion rate, and the paired Wilcoxon test against SML-PN with Holm correction. The median per-dataset rank of SML-PN is 6.33, with an interquartile range of 3.56 to 9.89. That is, the typical outcome of SML-PN is better than what its mean rank (7.00) suggests. Of the 11 model variants that perform no per-run search, SML-PN is the best by a large margin over the default CatBoost (7.52) and the default TabPFN (8.96).

Fig. S8(a) shows the importance of the per-run search. A 51-fit per-run search improves the ranks of the nine reference models by 0.7–10.7 places, with a median improvement of 4.4 places. Every one of the nine default fixed-configuration reference model variants ranks well below SML-PN, and seven remain lower after tuning. Figure S8(b) reports that SML-PN achieves $R^2 > 0$ (i.e., exceeding the prediction of training mean) in 45.2% of the 6,777 evaluation cells, sixth of the 20 model variants; the highest rate is 53.0%, and 11 model variants are below 40%. Most datasets contain no recoverable signal from a handful of rows, and no selection is applied that could conceal it, which is why the benchmark is analyzed on ranks rather than on absolute accuracy. Figure S8(c) shows that severe failures (i.e., test R^2 falls below -1) occur in 3.97% of SML-PN's runs, behind only the tuned XGBoost (3.37%) and the tuned CatBoost (3.65%).

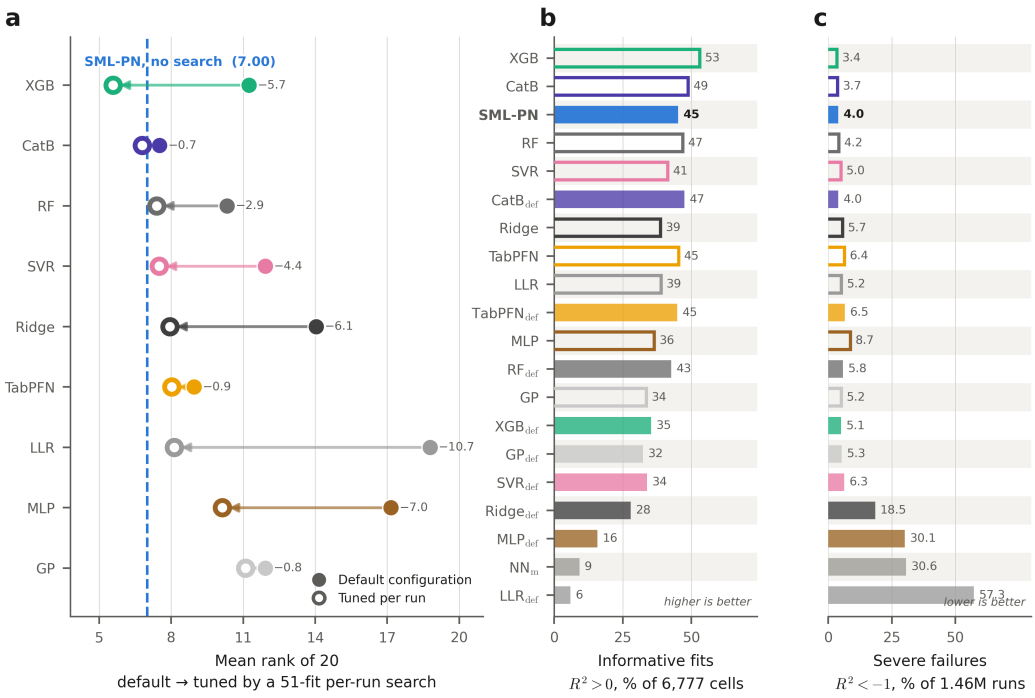

Figure S8: The importance of per-run search of the reference models. **(a)** Mean ranks of each reference model in its default fixed configuration and with a hyperparameter tuning search. SML-PN never performs any search. The number at the right-hand side of each pair indicates the number of ranking positions improved by the search. **(b)** Fraction of the 6,777 evaluation cells in which the model variant achieves $R^2 > 0$. **(c)** Fraction of the 1.46 million individual runs whose test R^2 falls below -1. Hollow bars/dots denote the model variants tuned per run; solid bars/dots denote fixed configuration.

Table S19: The performance of the 20 model variants across the 848 OpenML datasets*

Model variant	Config.	Mean rank	Median	Champion (%)	$R^2 > 0$ (%)	$R^2 < -1$ %	Holm p	Verdict
XGB	tuned	5.58	5.00	23.5	53.0	3.37	1.81e-12	SML-PN behind
CatB	tuned	6.79	6.67	8.1	48.8	3.65	2.75e-01	indistinguishable
SML-PN	fixed	7.00	6.33	19.2	45.2	3.97	—	—
RF	tuned	7.40	7.00	6.1	46.8	4.21	4.03e-02	SML-PN ahead
SVR	tuned	7.50	7.10	7.0	41.3	4.99	9.36e-03	SML-PN ahead
CatB_{def}	fixed	7.52	7.20	3.7	47.5	3.97	1.05e-02	SML-PN ahead
Ridge	tuned	7.95	7.57	4.5	38.7	5.69	6.76e-10	SML-PN ahead
TabPFN	tuned	8.02	8.22	12.0	45.4	6.39	5.81e-06	SML-PN ahead
LLR	tuned	8.13	7.70	3.3	39.0	5.21	2.48e-14	SML-PN ahead
$\text{TabPFN}_{\text{def}}$	fixed	8.96	9.15	1.1	44.8	6.53	9.80e-15	SML-PN ahead
MLP	tuned	10.13	10.57	7.0	36.4	8.71	2.98e-53	SML-PN ahead
RF_{def}	fixed	10.33	10.85	0.5	42.7	5.79	2.50e-41	SML-PN ahead
GP	tuned	11.09	11.44	1.4	33.5	5.18	5.87e-53	SML-PN ahead
XGB_{def}	fixed	11.25	11.33	0.7	35.3	5.06	1.02e-63	SML-PN ahead
GP_{def}	fixed	11.91	12.20	0.4	32.3	5.33	4.72e-69	SML-PN ahead
SVR_{def}	fixed	11.92	12.40	1.5	33.8	6.26	1.49e-73	SML-PN ahead
$\text{Ridge}_{\text{def}}$	fixed	14.04	15.33	0.1	27.8	18.51	6.28e-119	SML-PN ahead
MLP_{def}	fixed	17.15	18.20	0.0	15.8	30.08	4.94e-137	SML-PN ahead
NN_{m}	fixed	18.21	18.67	0.0	9.2	30.64	5.68e-138	SML-PN ahead
LLR_{def}	fixed	18.78	19.60	0.0	5.9	57.29	4.86e-137	SML-PN ahead

* "Champion" indicates the fraction of datasets on which the model variant is the best of the 20; "$R^2 > 0$" indicates the fraction of evaluation cells with an informative fit; "$R^2 < -1$" is the fraction of runs that fail severely. The last two columns give the paired Wilcoxon signed-rank test of SML-PN against the corresponding model variant over the per-dataset mean ranks, Holm-corrected across the 19 comparisons. The subscript "def" denotes the default model variant; the absence of the subscript denotes the tuned model variant (except for SML-PN and NN_{m}).

On the VLSE functions, SML-PN has the highest success rate and the lowest failure rate among the 20 model variants (see SM Sec. 5.4). On noisy real-world data, a boosting model's bounded leaf values provide extra protection. In contrast, the simple algorithm of SML in this research (Secs. 3–4 in the main text) does not have any explicit noise reduction mechanism.

6.4 Training budget

Table S20 shows the standings for various training budgets n/l, in addition to Fig. 3(b,c) in the main text. SML-PN is strongest where data are scarcest. Its mean rank changes from 6.00 at $n/l = 0.1$ to 8.44 at $n = l$. The rank of the tuned XGBoost is relatively flat over the same range. The trend of the tuned TabPFN is opposite, since a prior-fitted model needs training samples before its prior can act.

The ordering is the same in Fig. S9(a,b). At the two scarcest training budgets, SML-PN has the highest fraction of informative fits; when $n/l \geq 0.7$, it falls to the eighth. This effect is unsurprising: the parameter-space integration carries n unknowns, and the least-committal configuration is most valuable when the evidence is thinnest.

Table S20: Mean rank of every model variant at various relative training budget n/l.*

Model variant	n/l										
	all	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1**
XGB	5.58	5.29	5.01	5.33	5.45	5.53	5.77	5.83	5.93	6.40	6.25
CatB	6.79	5.61	5.30	5.89	6.52	6.99	7.40	7.64	7.91	8.72	8.27
SML-PN	7.00	6.00	5.74	6.70	6.76	7.49	7.80	8.35	8.11	8.95	8.44
RF	7.40	7.25	7.21	7.31	7.43	7.10	7.02	6.95	7.24	6.85	7.34
SVR	7.50	7.06	7.12	7.25	7.15	7.78	7.98	8.04	8.04	8.70	8.30
CatB_{def}	7.52	6.91	6.86	7.14	7.53	7.30	7.47	7.67	7.85	7.82	8.00
Ridge	7.95	8.08	7.94	8.11	8.26	8.55	8.45	8.69	8.24	8.33	7.79
TabPFN	8.02	8.65	9.24	8.80	8.02	7.64	7.06	6.76	6.78	5.78	6.47
LLR	8.13	8.15	8.52	8.56	8.30	8.21	8.32	8.49	8.16	8.04	7.83
$\text{TabPFN}_{\text{def}}$	8.96	9.58	10.14	9.71	9.01	8.56	7.98	7.63	7.68	6.60	7.43
MLP	10.13	12.15	11.60	10.73	10.42	10.17	10.22	9.89	9.97	10.38	9.91
RF_{def}	10.33	9.93	9.81	10.01	10.24	9.78	10.19	10.01	9.94	10.01	10.41
GP	11.09	10.62	11.02	10.94	11.08	10.96	10.89	10.81	10.88	10.29	10.61
XGB_{def}	11.25	10.47	10.60	10.91	11.40	11.46	11.47	11.51	11.74	11.76	11.64
GP_{def}	11.91	11.27	11.61	11.55	11.82	11.66	11.77	11.67	11.80	11.46	11.75
SVR_{def}	11.92	10.43	11.18	11.35	11.57	11.88	12.09	12.20	12.33	12.39	12.50
$\text{Ridge}_{\text{def}}$	14.04	16.01	15.87	14.84	14.34	14.22	13.59	13.41	13.07	12.92	12.77
MLP_{def}	17.15	18.76	18.07	17.61	17.46	17.45	17.27	17.00	16.91	17.09	16.79
NN_{m}	18.21	18.94	18.58	18.43	18.51	18.38	18.15	18.00	18.02	17.93	17.75
LLR_{def}	18.78	17.96	18.12	18.38	18.35	18.47	18.81	19.12	19.11	19.26	19.40

* The numbers of evaluation cells of various n/l (from left to right) are 383, 653, 676, 728, 679, 717, 687, 763, 667 and 824, respectively. The subscript "def" denotes the default model variant; the absence of the subscript denotes the tuned model variant (except for SML-PN and NN_{m}).

6.5 Input dimension

In the present investigation, the performance of the model variants can be more sensitive to the input dimension l than the training budget n, as suggested by Fig. S9(c-e) and Table S21. The mean rank of SML-PN degrades with l: 4.04 for $l \leq 5$, where it is the best of the 20 model variants and the champion on 38% of all datasets; the mean rank changes through 4.97, 6.60, 9.43 and 11.69, to 12.26 for $l > 100$. Tuned TabPFN varies in the opposite direction over the same range of l, from 11.56 to 2.53. Figure S9(d) shows clearly that SML-PN's standing is dependent more on the input dimension than the training budget: it is a top-five model at every training budget when $l \leq 20$. Its mean rank varies by less than two positions as n/l varies, but by eight positions as l changes. Note that the corpus is concentrated at small l (Fig. S7c): the two widest bands, 51–100 and > 100, contain only 48 and 54 of the 848 datasets, respectively, against 118–244 datasets in each of the four narrower bands. Their mean ranks are therefore less precise, with 95% bootstrap confidence intervals of about ± 1 rank, twice as wide as those of the narrower bands.

For the VLSE functions, the influence of l on the mean rank is reversed (see SM Sec. 5.2), which is not in conflict with Fig. S9(c-e). The absolute accuracy of SML-PN on the OpenML corpus rises with the input dimension: its median cell R^2 at $n = l$ is -0.05, $+0.07$, $+0.04$, $+0.11$, -0.01 and $+0.22$ across the six bands, respectively (see Fig. S9(f)), similar to the behavior in the VLSE experiment. The main difference lies in the reference model variants. On real-world datasets with many input features, SML-PN does not necessarily perform worse; rather, other models improve faster. At a fixed n/l, a larger l requires more training samples, and algorithms that need a large n (e.g., the prior-fitted transformer) benefit more. Moreover, unlike VLSE functions, a wide real-world input tends to be partly uninformative with more redundancy.

The above analysis is supported by Fig. S9(e), which groups the datasets by how much signal the whole set of 20 model variants recovers on them. SML-PN's standing improves as the problem becomes harder: its mean rank is 7.44 on the 440 datasets where some model variant reaches $R^2 > 0.3$, 6.73 on the 280 datasets where the best is 0–0.3, and 6.14 on the 128 datasets where no model variant recovers anything. Tuned TabPFN's mean rank worsens across the same three groups,

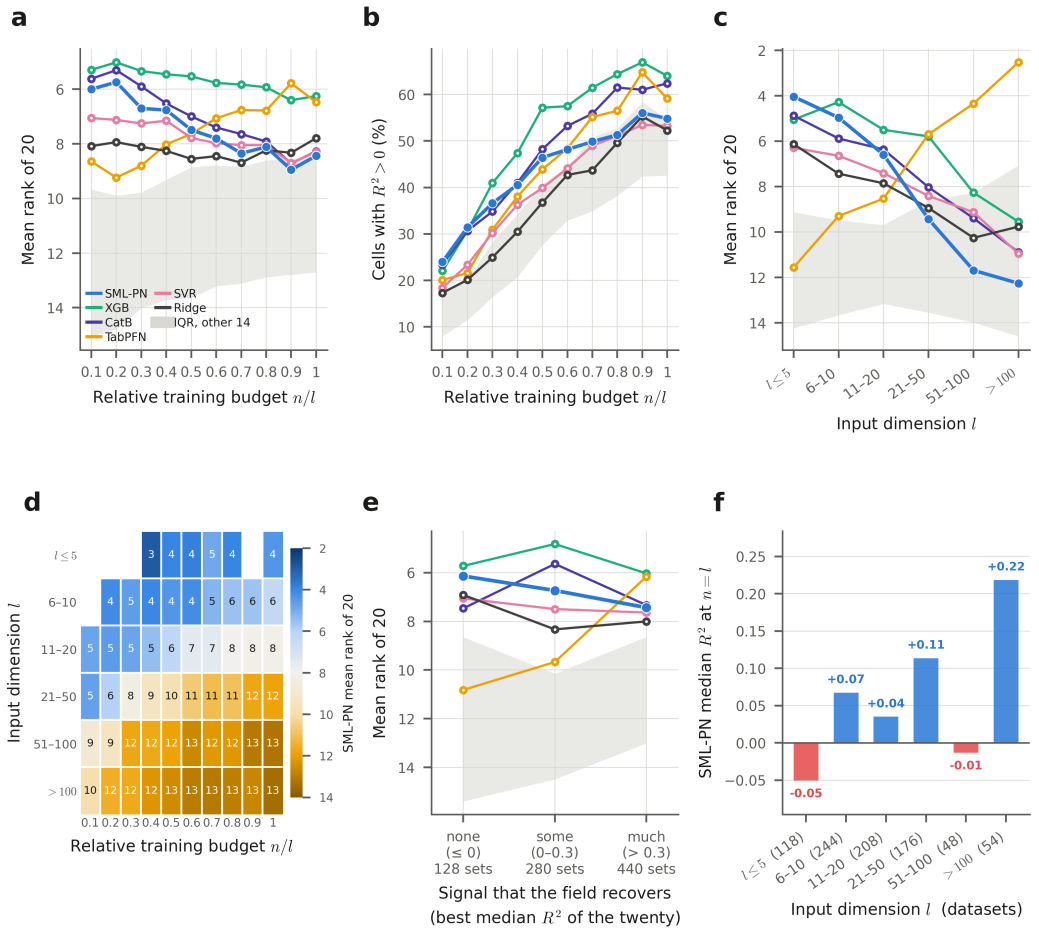

Figure S9: **(a)** Mean rank versus the relative training budget n/l; six model variants are shown by the lines, and the grey band shows the interquartile range of the other 14. **(b)** Fraction of the evaluation cells with an informative fit ($R^2 > 0$) versus n/l. **(c)** Mean rank versus the input dimension l. **(d)** The mean rank of SML-PN as a function of l and n/l; blank cells indicate the combinations that no dataset can realize. **(e)** Mean rank by the signal recovered by the comparison field (best median R^2 of the 20 model variants), grouped as none (≤ 0), some (0–0.3), or much (> 0.3). **(f)** The median cell R^2 of SML-PN at $n = l$: the absolute accuracy increases with l while the rank in panel (c) decreases, because other models may improve faster.

increasing from 6.16 to 10.82, and it is the only one of the six high-performance model variants to deteriorate so sharply.

Table S21: Mean rank of every model variant as a function of the input dimension l.*

	l						
Model variant	**all**	**≤ 5**	**6–10**	**11–20**	**21–50**	**51–100**	**> 100**
XGB	5.58	5.06	4.28	5.50	5.79	8.27	9.55
CatB	6.79	4.88	5.88	6.37	8.03	9.39	10.89
SML-PN	7.00	4.04	4.97	6.60	9.43	11.69	12.26
RF	7.40	8.36	8.14	7.01	6.43	6.28	6.94
SVR	7.50	6.29	6.65	7.41	8.41	9.13	10.95
CatB_{def}	7.52	7.54	8.06	7.53	7.44	6.42	6.54
Ridge	7.95	6.13	7.43	7.86	8.95	10.26	9.77
TabPFN	8.02	11.56	9.30	8.53	5.70	4.35	2.53
LLR	8.13	7.48	8.01	7.81	8.56	9.32	9.35
$\text{TabPFN}_{\text{def}}$	8.96	12.54	10.37	9.53	6.61	4.94	2.81
MLP	10.13	8.64	9.20	10.20	11.22	11.50	13.42
RF_{def}	10.33	11.82	11.17	10.45	8.80	9.67	7.85
GP	11.09	12.27	12.50	10.82	10.73	8.02	6.74
XGB_{def}	11.25	10.61	10.78	12.04	11.00	12.11	11.65
GP_{def}	11.91	13.03	13.21	11.75	11.61	9.06	7.47
SVR_{def}	11.92	11.98	12.06	12.53	11.56	10.18	11.28
$\text{Ridge}_{\text{def}}$	14.04	14.65	13.81	13.38	14.20	14.61	14.98
MLP_{def}	17.15	16.15	16.39	17.30	18.30	18.19	17.87
NN_{m}	18.21	18.08	18.52	18.31	18.39	17.83	16.73
LLR_{def}	18.78	18.77	19.26	18.60	18.40	18.78	18.71

* The subscript "def" denotes the default model variant; the absence of the subscript denotes the tuned model variant (except for SML-PN and NN_{m}).

6.6 Comparison of model scales

Fig. S10 shows the number of effective parameters (m) used by the 20 model variants in Sec. 5 of the main text, which measures the model scale. In addition to the trainable parameters of NNs, m also accounts for the internal and leaf nodes of the selected ensembles for the boosting models and forests, the $n_{\text{sv}} + 1$ retained support-vector coefficients for SVR, the n dual weights for GP, the $l + 1$ coefficients for the linear models, and the 2.5×10^7 frozen weights for TabPFN. m counts the parameters of the selected model; the searched hyperparameters and the candidate fits per run are reported separately in Table S23. For SML-PN, the configuration is fixed, while $m = n + 2l + 2$ is dependent on the training budget and the input dimension; like every value of m in Fig. S10 and Table S23, the reported $m = 39$ is a median. SM Sec. 6.7 gives the definitions and the values of m for every model variant.

Compared to SML-PN, the two models with matching or higher ranks in Sec. 5.3 of the main text have much larger m: tuned XGBoost and CatBoost respectively have nearly 2.0×10^4 and 3.2×10^4 effective parameters, 2-3 orders of magnitude higher than that of SML-PN. CatBoost and XGBoost each consumes a 51-fit random search over 6 or 10 hyperparameters per run, without which their default configurations fall to ranks of 7.5–11.2, well below that of SML-PN. TabPFN has the largest number of parameters among all models ($m \approx 2.5 \times 10^7$). In essence, m measures the scale of the selected model.

6.7 Effective parameter counts

SM Sec. 6.6 compares the model scales by the number of effective parameters m learned by each model variant during the numerical procedures. Table S22 provides the detailed definitions, and Table S23 reports the median values across all evaluation cells. The parameter count is an important indicator of the lower limit of computational cost. In general, for comparable numerical efficiency, a smaller parameter count suggests more cost-effective prediction.

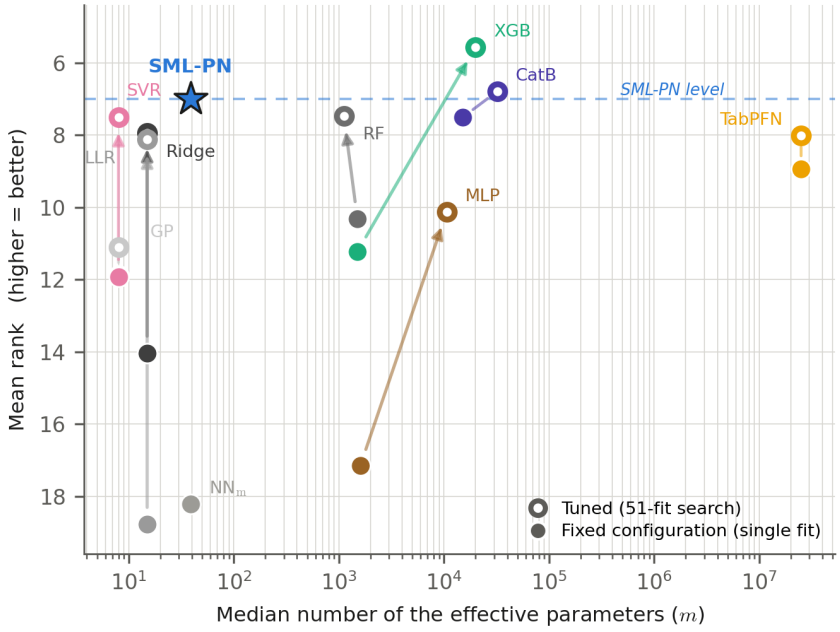

Figure S10: Performance rank versus the median number of effective parameters for the 20 model variants (indicated by different colors) on the OpenML benchmark across all training budgets. The arrows indicate the increase in performance and model scale as the models change from fixed configuration (solid symbols) to tuned model variants (empty symbols). The star symbol marks SML-PN, which is never tuned.

In Table S22, the parameter counts correspond to the actual settings used by each model variant in a given run, not to the largest settings allowed by its search space. To be conservative, the leaf count of every tree ensemble, the two boosting models included, is calculated as $\min\left(2^d, n\right)$ rather than 2^d, with d being the tree depth, since a tree cannot open more leaves than the training samples.

Table S22: Effective parameters learned in the numerical procedure.*

Family	**Effective learned parameters**
SML-PN	$n + 2l + 2$: one Lagrange multiplier per training sample, the adaptive bounds, the scale factor, and the intercept (see SM Sec. 3.3)
NN$_m$, MLP	Trainable weights and biases, as reported by the implementation
Ridge, LLR	$l + 1$: one coefficient per input feature, plus the intercept
XGBoost, RF	$n_{\text{trees}}\left[2 \cdot \min\left(2^d, n\right) - 1\right]$: internal and leaf nodes of the selected ensemble, with the leaf count capped by the training-set size
CatBoost	$n_{\text{it}}\left[2 \cdot \min\left(2^d, n\right) - 1\right]$: internal and leaf nodes of the selected ensemble, with the leaf count capped by the training-set size, at the library-default depth $d = 6$ (such that at these training budgets the training-set size caps the leaf count)
SVR	$n_{\text{sv}} + 1$: the dual coefficients actually retained, plus the intercept
GP	n: one dual weight per training sample; the kernel is nonparametric
TabPFN	2.5×10^7: the frozen weights of the released prior-fitted transformer, carried into every prediction

* n denotes the training-set size, l is the input dimension, m is the number of internal parameters of PN, d is the tree depth, n_{trees} is the number of trees, n_{it} is the number of boosting iterations, and n_{sv} is the number of support vectors.

Table S23: The effective parameter number of each model variant (the median over all evaluation cells).*

Model variant	Configuration	Mean rank	Effective parameters	Hyperparameters searched	Fits per run
XGB	tuned	5.58	2.00×10^4	10	51
CatB	tuned	6.79	3.21×10^4	6	51
SML-PN	fixed	7.00	39	0	1
RF	tuned	7.40	1.12×10^3	4	51
SVR	tuned	7.50	8	3	25
CatB_{def}	fixed	7.52	1.50×10^4	0	1
Ridge	tuned	7.95	15	1	9
TabPFN	tuned	8.02	2.50×10^7	1	4
LLR	tuned	8.13	15	1	9
$\text{TabPFN}_{\text{def}}$	fixed	8.96	2.50×10^7	0	1
MLP	tuned	10.13	1.06×10^4	6	51
RF_{def}	fixed	10.33	1.50×10^3	0	1
GP	tuned	11.09	8	1	4
XGB_{def}	fixed	11.25	1.50×10^3	0	1
GP_{def}	fixed	11.91	8	0	1
SVR_{def}	fixed	11.92	8	0	1
$\text{Ridge}_{\text{def}}$	fixed	14.04	15	0	1
MLP_{def}	fixed	17.15	1.60×10^3	0	1
NN_{m}	fixed	18.21	39	0	1
LLR_{def}	fixed	18.78	15	0	1

* The subscript "def" denotes the default model variant; the absence of the subscript denotes the tuned model variant (except for SML-PN and NN_{m}).

7 MULTI-FEATURE OUTPUT

Eqs. (11) and (12) in the main text assume a scalar output for simplicity. Here we extend the SML-PN formulation to an output vector $\mathbf{y}_j = \{y_{j\nu}\}$ ($\nu = 1, 2 \ldots \nu_0$), where ν_0 is the number of output features, and subscripts j and ν index the training samples ($j = 1, 2 \ldots n$) and the output features ($\nu = 1, 2 \ldots \nu_0$), respectively. Equation (3) in the main text becomes

$$\int \rho_\mu \widehat{y}_{\mu j\nu} \mathrm{d}\Omega = \overline{y}_{j\nu},$$

where $\overline{y}_{j\nu}$ is the ν-th feature of the output of the j-th training sample, and $\widehat{y}_{\mu j\nu}$ is the ν-th feature of the output of the μ-th possible parameter configuration $\mathbf{w}_\mu$ and the j-th training input $\mathbf{x}_j$. The Lagrangian is $\mathscr{L} = -\int \rho_\mu \ln \rho_\mu \mathrm{d}\Omega + \lambda_0 \left(\int \rho_\mu \mathrm{d}\Omega - 1\right) + \sum_{j=1}^{n} \sum_{\nu=1}^{\nu_0} \beta_{j\nu} \left(\overline{y}_{j\nu} - \int \rho_\mu \widehat{y}_{\mu j\nu} \mathrm{d}\Omega\right)$, where λ_0 and $\beta_{j\nu}$ are the Lagrange multipliers. According to Eq. (4) in the main text, we have $-\ln \rho_\mu - 1 + \lambda_0 - \sum_j \sum_\nu \beta_{j\nu} \widehat{y}_{\mu j\nu} = 0$, so that

$$\rho_\mu = e^{\lambda_0 - \sum_j \sum_\nu \beta_{j\nu} \widehat{y}_{\mu j\nu} - 1}.$$

Define $Z = \int e^{-\sum_j \sum_\nu \beta_{j\nu} \widehat{y}_{\mu j\nu}} \mathrm{d}\Omega$ as the partition function. Based on Eq. (2) in the main text, $Z = e^{1-\lambda_0}$, i.e., $\lambda_0 = 1 - \ln Z$. Consequently, $Z = \int e^{-\widehat{E}_\mu} \mathrm{d}\Omega$ and $\rho_\mu = \frac{e^{-\widehat{E}_\mu}}{Z}$ remain in the same forms as Eqs. (5) and (6) in the main text, respectively, where $\widehat{E}_\mu = \sum_j \sum_\nu \beta_{j\nu} \widehat{y}_{\mu j\nu}$. For a new input $\mathbf{x}_*$, the prediction of the ν-th output feature is

$$y_{*\nu} = \int \rho_\mu \widehat{y}_{\mu *\nu} \mathrm{d}\Omega.$$

where $\widehat{y}_{\mu *\nu}$ is the ν-th feature of the output of $\mathbf{w}_\mu$ and $\mathbf{x}_*$.

For the power-law network (PN), $\widehat{y}_{\mu jv} = -\ln\prod_{k=1}^{l} x_{jk}^{w_{\mu kv}} = -\sum_k w_{\mu kv}\ln x_{jk}$, and Eq. (9) in the main text becomes

$$\rho_\mu = \frac{1}{Z}e^{-\widehat{E}_\mu} = \frac{1}{Z}e^{\sum_j\sum_v \beta_{jv}\sum_k w_{\mu kv}\ln x_{jk}} = \frac{1}{Z}\prod_{v=1}^{v_0}\prod_{k=1}^{l} B_{kv}^{w_{\mu kv}},$$

where

$$B_{kv} = \prod_{j=1}^{n} x_{jk}^{\beta_{jv}},$$

$$Z = \int \prod_{v=1}^{v_0}\prod_{k=1}^{l} B_{kv}^{w_{\mu kv}}\,\mathrm{d}\Omega = \prod_{v=1}^{v_0}\prod_{k=1}^{l} z_{kv},$$

$$z_{kv} = \frac{B_{kv}^{\widehat{w}_{kv}} - B_{kv}^{\check{w}_{kv}}}{\ln B_{kv}},$$

and $\widehat{w}_{kv}$ and $\check{w}_{kv}$ are the upper and lower bounds of w_{kv}, respectively. The unknowns (β_{jv}) are solved through

$$\sum_k \ln x_{jk}\left[\int\left(\prod_{\overline{v}=1}^{v_0}\prod_{\overline{k}=1}^{l} B_{\overline{k}\overline{v}}^{w_{\mu\overline{k}\overline{v}}}\right) w_{\mu kv}\,\mathrm{d}\Omega\right] = -Z\overline{y}_{jv}.$$

That is,

$$\sum_{k=1}^{l} \varepsilon_{kv}\ln x_{jk} = \overline{y}_{jv}, \text{ where}$$

$$\varepsilon_{kv} = -\frac{(\widehat{w}_{kv}\ln B_{kv} - 1)\,B_{kv}^{\widehat{w}_{kv}} - (\check{w}_{kv}\ln B_{kv} - 1)\,B_{kv}^{\check{w}_{kv}}}{\left(B_{kv}^{\widehat{w}_{kv}} - B_{kv}^{\check{w}_{kv}}\right)\ln B_{kv}}.$$

For a new input $\mathbf{x}_* = \{x_{*k}\}$ $(k = 1, 2\ldots l)$, the prediction of the v-th output feature is

$$y_{*v} = \sum_{k=1}^{l} \varepsilon_{kv}\ln x_{*k}.$$