XPFN: LEARNING ZERO-SHOT TABULAR PREDICTION AND EXPLANATION FROM PRIOR-SAMPLED RESPONSE FUNCTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tabular foundation models (TFMs) make predictions in a single forward pass, but explaining these predictions still relies largely on expensive post-hoc attribution. Extending the prior-data fitted network (PFN) paradigm from prediction to explanation is a promising alternative, as it would let a TFM itself act as a cross-task Shapley estimator. However, the cost and error of generating attribution supervision have kept such models far below the pretraining scale of TFMs. We introduce XPFN, a TFM that produces zero-shot regression predictions together with their Shapley attributions in one forward pass. Our key design is to **draw attribution supervision from the prior itself**. A structural causal model generates the feature columns and a prior-sampled response function generates the target, so **the function that defines each synthetic task can be queried under any feature coalition**. This yields exact coalition labels without fitting a surrogate model or re-evaluating the TFM. On 40 real datasets, XPFN's direct attributions agree with the Shapley values of its own predictions ($r = 0.91$), and its predictions are competitive with tuned tree ensembles such as XGBoost and CatBoost. Its explanations are up to 90$\times$ faster than TreeSHAP on XGBoost and over 1,700$\times$ faster than KernelSHAP on TabPFN-v3 and TabICLv2. Controlled ablations further show that our decoupled prior delivers more accurate and cheaper supervision labels than existing methods, which enables efficient pretraining of XPFN on over 10 million synthetic tabular tasks.

1 INTRODUCTION

Tabular foundation models (TFMs) have reshaped tabular modelling by enabling a single pretrained model to predict on unseen tasks via in-context learning (ICL), without any task-specific fitting (Hollmann et al., 2025; Qu et al., 2026). Prediction alone, however, is rarely sufficient. In domains such as finance, medicine and scientific discovery, feature attributions are needed to interpret individual predictions, validate model behaviour and support downstream decisions (Fonseca & Stoyanovich, 2026; Rundel et al., 2024). While TFMs consolidate prediction across tasks into one pretrained model, their outputs are still explained mostly with post-hoc methods such as SHAP (Lundberg & Lee, 2017), which is expensive because model-agnostic implementations re-evaluate the TFM over many feature coalitions for every prediction. This raises a natural question: can a single TFM produce both prediction and explanation in one forward pass?

Existing work improves SHAP efficiency through approximation or amortization. KernelSHAP (Lundberg & Lee, 2017; Covert & Lee, 2021) and TabPFN-specific adaptations (Rundel et al., 2024) reduce TFM explanation costs, and TFMs have also been used to estimate the conditional expectations required by SHAP (Olsen & Christensen, 2026). These methods, however, still run separately for each explained instance (Lundberg & Lee, 2017; Lundberg et al., 2020; Rundel et al., 2024). Amortized methods instead move this computation into training. FastSHAP learns an attribution estimator from the coalition-specific outputs of a fixed predictor, which makes explanation cheap at inference (Jethani et al., 2022). Recent work extends this idea to cross-task pretraining. ExplainerPFN estimates Shapley attributions on unseen tables, and ShapPFN jointly predicts targets and feature attributions (Fonseca & Stoyanovich, 2026; Sena & Azevedo, 2026).

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.

To our knowledge, no such model has been pretrained at the scale of established TFMs, so how well zero-shot explanation generalizes across diverse tasks remains open. The main bottleneck is the cost of generating reliable attribution supervision, which takes one of two expensive routes (Figure 1(a,b)). ExplainerPFN (Fonseca & Stoyanovich, 2026) fits a task-specific base predictor, such as an MLP, and explains it to obtain Shapley-value labels, which costs a model fit plus a SHAP computation per task and introduces error wherever the fitted predictor misses the synthetic task. ShapPFN (Sena & Azevedo, 2026) evaluates the TFM itself on masked inputs by Monte Carlo sampling, which costs a forward pass per sampled coalition. Either way, this cost limits the scale and diversity of explanation pretraining.

We remove this bottleneck by drawing attribution supervision directly from the prior-data process (Figure 1(c)) (Hollmann et al., 2025). A prior-data fitted network (PFN) is trained on carefully designed priors, so the prior can supply not only the data of each task but also the answers to attribution queries about it, if it keeps the function that generated the target. A structural causal model (SCM) prior generates the feature columns of each synthetic table, while a lightweight response function sampled from a separate prior over functions generates the target column. The response function requires no fitting and is, by construction, the deterministic regression function of the synthetic task. **It can therefore be evaluated directly under feature coalitions**, providing consistent prediction and coalition labels at low cost. On this basis, we propose XPFN, a natively explainable tabular PFN for regression that produces a base value and per-feature attributions in a single forward pass, with their additive sum reconstructing the prediction.

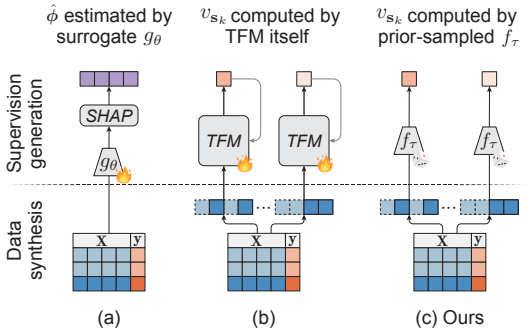

Figure 1: Different supervision paradigms. (a) SHAP values of a fitted surrogate g_θ. (b) Repeated evaluation of the TFM itself on masked inputs. (c) Query the response function f_τ that generated the targets (XPFN).

Our contributions are three-fold:

- We propose a decoupled prior whose sampled response function generates the target of each synthetic task and supplies coalition labels, providing efficient supervision for tabular feature attribution.
- We introduce XPFN, a TFM that produces predictions and Shapley feature attributions in one forward pass, and show that its training objective is minimized by the Shapley values up to a shift shared by all features.
- We scale the pretraining of XPFN to 12 million tasks, and provide systematic evaluation of its performances against previous paradigms.

2 Background and Related Work

2.1 Shapley-Based Feature Attribution and Amortized Estimation

For a tabular task with feature set $\mathcal{F} = \{1, \ldots, M\}$, predictor f, and an instance $\mathbf{x}$, let $v_{\mathbf{x}}(S)$ denote the expected predictor output associated with a feature coalition $S \subseteq \mathcal{F}$. Features in S retain their values in $\mathbf{x}$, while the remaining features are considered unobserved. The Shapley value of feature m is then defined as its average marginal contribution across all possible coalitions:

$$\phi_m(v_{\mathbf{x}}) = \sum_{S \subseteq \mathcal{F} \setminus \{m\}} \frac{|S|!(M - |S| - 1)!}{M!} \left[v_{\mathbf{x}}(S \cup \{m\}) - v_{\mathbf{x}}(S) \right]. \tag{1}$$

The resulting Shapley values satisfy the efficiency property $\sum_{m=1}^{M} \phi_m(v_{\mathbf{x}}) = v_{\mathbf{x}}(\mathcal{F}) - v_{\mathbf{x}}(\emptyset)$ (Shapley, 1953; Lundberg & Lee, 2017). Here, $v_{\mathbf{x}}(\emptyset)$ is the expected output when no feature value is

retained from $\mathbf{x}$. The definition of $v_{\mathbf{x}}(S)$ determines how the unobserved features are handled. Conditional attribution draws the excluded features from their conditional distribution given $\mathbf{x}_S$, thereby preserving dependencies between the included and excluded features. Interventional attribution instead draws the excluded features from an unconditional background distribution, independently of $\mathbf{x}_S$ (Sundararajan & Najmi, 2020; Janzing et al., 2020; Chen et al., 2020; Covert et al., 2021).

Exact Shapley computation requires coalition values for all 2^M feature subsets and becomes costly when evaluating f is expensive. KernelSHAP approximates Shapley values through weighted regression over a limited number of sampled coalitions. TreeSHAP instead exploits the structure of tree ensembles to compute attributions efficiently (Lundberg & Lee, 2017; Covert & Lee, 2021; Lundberg et al., 2020). Rather than approximating each explanation independently, learned estimators amortize post-hoc computation through training (Wang et al., 2021; Covert et al., 2023). FastSHAP and InstaSHAP train attribution estimators from coalition-specific outputs of a fixed predictor, enabling low-cost explanations at inference (Jethani et al., 2022; Enouen & Liu, 2025). Stochastic amortization further reduces supervision cost by using noisy labels from low-budget Shapley estimators (Covert et al., 2024). ViaSHAP jointly learns prediction and attribution through Shapley-value regression, but is trained on each dataset separately (Alkhatib et al., 2025). XPFN is an amortized estimator, but pretrained across tasks rather than trained for one predictor.

2.2 Tabular Foundation Models

PFN-style TFMs are pretrained on synthetic datasets sampled from a prior. At inference, a labelled context set is provided to the pretrained model, which performs ICL to predict the targets of unlabelled query rows (Müller et al., 2022). The TabPFN series targets predictive performance and task coverage, and TabPFN-v3 improves scalability and efficiency (Hollmann et al., 2025; Grinsztajn et al., 2026). TabICL and TabICLv2 redesign the architecture to support larger tables (Qu et al., 2025; 2026), LimiX uses masked joint-distribution modelling to support multiple structured-data tasks (Zhang et al., 2025a), and retrieval or routing methods give each query a local context (Thomas et al., 2024; Xu et al., 2025). As is common for TFMs, we call this inference *zero-shot*: no parameters are updated on the task, and the model conditions on a labelled context set. For attribution, the context carries no attribution examples.

SCM priors capture feature dependencies, while mixed-prior methods such as Mitra (Zhang et al., 2025b) combine complementary task families. These priors are built for predictive pretraining and need not expose a task-level response function that can be evaluated under arbitrary feature coalitions, which attribution supervision requires. Do-PFN, in contrast, trains on interventional outcomes queried from its SCM prior (Robertson et al., 2025).

2.3 Explaining Tabular Foundation Models

Applying post-hoc explanations to a TFM typically requires repeated model evaluations over feature coalitions. TabPFN-specific adaptations and context-caching strategies reduce this cost by reusing representations across perturbed inputs (Rundel et al., 2024; Grinsztajn et al., 2026). TFMs have also been used as in-context regressors to estimate the expected predictor output conditional on the observed features in each coalition (Olsen & Christensen, 2026), which avoids a conditional model per coalition but still needs repeated TFM evaluations.

Recent work amortizes this inference-time computation through cross-task pretraining. ExplainerPFN fits and explains a base predictor, such as an MLP, for each synthetic task. The resulting attributions are then used to train a TFM for zero-shot Shapley estimation (Fonseca & Stoyanovich, 2026). This process is costly to scale, and the errors of the fitted base predictor enter the attribution labels (Section 4.5). ShapPFN instead integrates prediction and attribution within a TFM, but constructs its coalition targets through repeated Monte Carlo evaluations of the TFM itself (Sena & Azevedo, 2026). Both target classification: ShapPFN is pretrained on binary tasks with 2–5 features and at most 200 rows and applied one-vs-all to multi-class tasks, and ExplainerPFN is pretrained on tasks with a binary target. GAMformer instead outputs an additive model in context, whose shape functions serve as the explanation (Mueller et al., 2024).

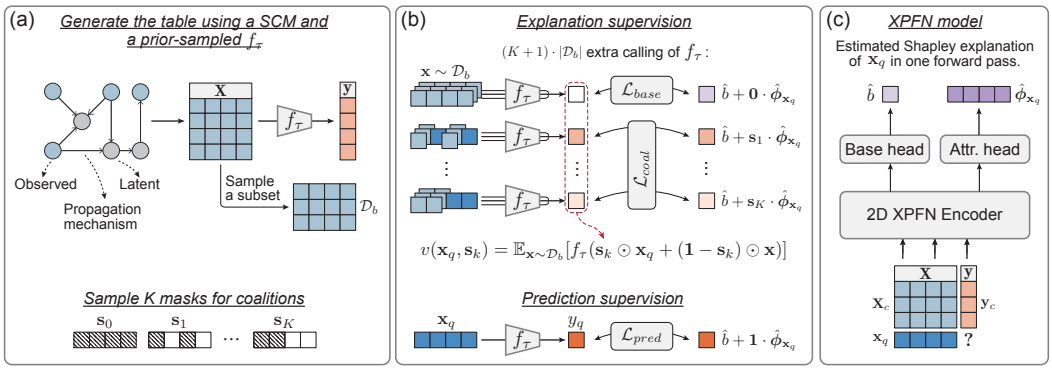

Figure 2: XPFN pretraining. (a) An SCM generates the features and a prior-sampled response function f_τ the targets, respectively. (b) f_τ under the empty, sampled and full masks gives the base-value, coalition and prediction targets. (c) XPFN returns a base value and per-feature attributions for all query rows in one forward pass.

3 The XPFN Framework

This section introduces the problem, prior, objective and architecture of XPFN (Figure 2).

3.1 Problem Formulation

We first cast conventional supervised tabular regression with explanation as a two-stage problem. Let $\mathcal{D}_c = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_c}$ denote a labelled context set, where each row (sample) contains M numerical or categorical features indexed by $\mathcal{F} = \{1, \ldots, M\}$. The N_q query rows $\mathbf{x}_{q,1}, \ldots, \mathbf{x}_{q,N_q}$ to be predicted and explained are stacked as $\mathbf{X}_q = [\mathbf{x}_{q,1}^\top; \ldots; \mathbf{x}_{q,N_q}^\top]$; $\mathbf{x}_q$ denotes any one of them. In the first stage, a task-specific predictor f_θ is fitted on $\mathcal{D}_c$. In the second, an explanation method such as KernelSHAP or FastSHAP is configured or trained to estimate the feature attributions of the predictions of f_θ on $\mathbf{X}_q$, which requires model-specific computation and often repeated evaluations.

We instead train a single TFM g_ω that maps $\mathcal{D}_c$ and $\mathbf{X}_q$ directly to a base value $\hat{b} \in \mathbb{R}$ and per-feature Shapley attributions $\hat{\mathbf{\Phi}}_q \in \mathbb{R}^{N_q \times M}$:

$$(\hat{b}, \hat{\mathbf{\Phi}}_q) = g_\omega(\mathcal{D}_c, \mathbf{X}_q). \tag{2}$$

Following the efficiency property (Section 2.1), the predictions are reconstructed as

$$\hat{\mathbf{y}}_q = \hat{b}\mathbf{1}_{N_q} + \hat{\mathbf{\Phi}}_q \mathbf{1}_M, \tag{3}$$

where $\mathbf{1}_d$ denotes a d-dimensional all-ones vector. The base value is shared across query rows and estimates the expected response over the empirical distribution of the context rows.

3.2 Decoupled Prior-Data Generation

Training a TFM to estimate feature attributions requires a prior that generates both prediction and attribution supervision. Conventional SCM priors generate feature and target columns from sampled directed acyclic graphs (DAGs). They suit predictive pretraining, but the target also depends on latent nodes and noise, so they do not in general expose a deterministic function of the observed features that can be evaluated under the feature coalitions required for Shapley values. Attribution supervision then has to come from fitted surrogates or from repeated evaluations of the TFM itself (Section 2.3). We instead decouple feature and target generation into an SCM prior and a prior-sampled, training-free response function f_τ (Figure 2(a)).

SCM prior. Following the SCM design of TabICLv2 (Qu et al., 2026), we use an SCM prior to generate the feature columns $\mathbf{X}$ of each pretraining table. At the core of the SCM is a DAG $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where each node $n \in \mathcal{N}$ contains a multi-dimensional state and each directed edge

specifies a propagation mechanism between two nodes. We sample a DAG and its edge-wise mechanisms, initialize root states from sampled source distributions, and propagate them through the graph. For nodes with multiple parents, parent states are either processed jointly after concatenation or processed separately before aggregation. Features are then collected from a subset of the nodes, transformed, and augmented with independently sampled noise columns, whose indices form $\mathcal{F}_{\text{noise}} \subseteq \mathcal{F}$.

Response-function prior. Once the features have been generated, we sample a deterministic response function f_τ from a separate prior over functions to generate the target column. A subset $E \subseteq \mathcal{F}_{\text{noise}}$ of the noise columns is first excluded, and f_τ selects its active features $A \subseteq \mathcal{F} \setminus E$, which may include both SCM-derived and noise columns. Since f_τ never reads a feature outside A, such a feature, in particular every feature in E, leaves every coalition value unchanged and is a null player with zero Shapley value. Each task therefore contains features whose attribution is known to be zero. We use either a single sampled response function or an ensemble of several to increase diversity. The prior covers common lightweight tabular predictors, including multilayer perceptrons, random forests and boosted trees, all randomly parameterized rather than fitted and hence training-free. Appendix A gives the settings of both priors.

Since f_τ is the regression function of the synthetic task, it can be evaluated directly under feature coalitions. Following interventional Shapley attribution (Lundberg & Lee, 2017; Sundararajan & Najmi, 2020; Covert et al., 2021), we represent a coalition by a binary mask $\mathbf{s} \in \{0,1\}^M$, with $s_m = 1$ if feature m keeps its value, and define for a query $\mathbf{x}_q$

$$v_\tau(\mathbf{x}_q, \mathbf{s}) = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_b}\left[f_\tau(\mathbf{s} \odot \mathbf{x}_q + (\mathbf{1} - \mathbf{s}) \odot \mathbf{x})\right], \tag{4}$$

where $\odot$ denotes element-wise multiplication and $\mathbf{x}$ is drawn uniformly from a background set $\mathcal{D}_b$ of context feature rows. During pretraining, $\mathcal{D}_b$ is a random subset of the context rows; in evaluation, reference attributions and post-hoc baselines use the first 64 context rows. XPFN itself takes no background input; its implicit background is the context. Section 4.5 compares these coalition values with those of a fitted surrogate.

3.3 Training Objective

As shown in Figure 2(b), we adopt a FastSHAP-style coalition objective for Shapley attribution. For each query $\mathbf{x}_q$, we draw K masks $\{\mathbf{s}_k\}_{k=1}^K$ from the Shapley-kernel distribution (Jethani et al., 2022):

$$p_{\text{Sh}}(\mathbf{s}) \propto \frac{M-1}{\binom{M}{|\mathbf{s}|}|\mathbf{s}|(M-|\mathbf{s}|)}, \qquad 0 < |\mathbf{s}| < M, \tag{5}$$

where $|\mathbf{s}| = \sum_{m=1}^M s_m$. Writing $\hat{\boldsymbol{\phi}}_q \in \mathbb{R}^M$ for the row of $\hat{\boldsymbol{\Phi}}_q$ that belongs to $\mathbf{x}_q$ and $\mathbb{E}_q$ for the average over the query rows of a task (the average over tasks is implicit), the coalition loss is

$$\mathcal{L}_{\text{coal}} = \mathbb{E}_q\left[\frac{1}{K}\sum_{k=1}^K \left(\hat{b} + \mathbf{s}_k^\top \hat{\boldsymbol{\phi}}_q - v_\tau(\mathbf{x}_q, \mathbf{s}_k)\right)^2\right]. \tag{6}$$

Because the masks are sampled from p_{Sh}, the average over masks is an unbiased estimate of the Shapley-kernel-weighted regression loss. The base-value loss is

$$\mathcal{L}_{\text{base}} = \mathbb{E}_q\left[\left(\hat{b} - v_\tau(\mathbf{x}_q, \mathbf{0})\right)^2\right]. \tag{7}$$

Following Equation 3, the prediction loss is

$$\mathcal{L}_{\text{pred}} = \mathbb{E}_q\left[\left(\hat{b} + \mathbf{1}^\top \hat{\boldsymbol{\phi}}_q - y_q\right)^2\right], \tag{8}$$

where $y_q = f_\tau(\mathbf{x}_q) = v_\tau(\mathbf{x}_q, \mathbf{1})$ is the noise-free target of $\mathbf{x}_q$. The total objective is

$$\mathcal{L} = \mathcal{L}_{\text{base}} + \lambda\mathcal{L}_{\text{coal}} + (2-\lambda)\mathcal{L}_{\text{pred}}, \tag{9}$$

where $\lambda \in [0,2]$ balances coalition reconstruction and prediction. In Figure 2(b), the empty mask $\mathbf{s}_0 = \mathbf{0}$ gives the target of the base value, the K sampled masks give the coalition term, and the full mask $\mathbf{1}$ gives the prediction term. For any $\lambda \in (0,2]$, **the minimizer of this objective in expectation over the masks gives the Shapley values of each query up to a shift shared by all features** (Proposition 1, Appendix D).

3.4 XPFN ARCHITECTURE

To demonstrate the effect of the decoupled prior, we adopt an encoder-only 2D Transformer architecture similar to TabPFN-v2 (Hollmann et al., 2025). Numerical features of all rows are standardized with the mean and standard deviation of $\mathcal{D}_c$. Each standardized numerical cell is mapped through a learned linear projector, whereas categorical features are mapped to shared learned embeddings. A random identifier vector is attached to each column so that the model can tell columns apart; identifiers are redrawn for every pretraining task and fixed at evaluation. The cell embeddings are processed by 2D Transformer blocks that alternate attention across features and across rows, with residual connections and LayerNorm.

We replace TabPFN's binned bar-distribution output with a point-valued base head and an attribution head. As illustrated in Figure 2(c), the pooled context representation is passed through an MLP to estimate the base value $\hat{b}$, and the query feature representations are decoded by the attribution head into $\hat{\mathbf{\Phi}}_q \in \mathbb{R}^{N_q \times M}$. All targets, and in pretraining also the coalition values, are standardized with the mean μ_c and standard deviation σ_c of the context labels, and the outputs are mapped back as $\hat{b} \leftarrow \mu_c + \sigma_c \hat{b}$ and $\hat{\mathbf{\Phi}}_q \leftarrow \sigma_c \hat{\mathbf{\Phi}}_q$. The shift μ_c is common to all coalition values and cancels in every marginal contribution, so it enters only the base value, and Equation 3 holds on the original target scale.

4 EXPERIMENTS

In this section, we evaluate the prediction and attribution of XPFN on a synthetic explainable AI (XAI) benchmark with exact Shapley values (Section 4.2) and on real-world datasets (Section 4.3). We then report its pretraining and inference cost (Section 4.4) and ablate the supervision paradigm, pretraining scale, prior and the training objective (Section 4.5).

4.1 EXPERIMENTAL SETUP

XPFN training. Our main model, XPFN-9.8M, contains 16 2D Transformer blocks with 9.8 million parameters and is pretrained on 12 million synthetic tasks. The ablation studies use smaller 10-block variants (XPFN-6.2M), pretrained from stretch on 3.5 million tasks for the prior and training objective studies, while pretrained on 720k tasks for the supervision-paradigm studies. Appendix B gives the details.

Benchmarks. **XAI-Bench** (Liu et al., 2021) provides tasks with known data-generating functions (six mechanisms, known feature correlation $\rho \in \{0, 0.5, 0.9\}$). The experiments were independently conducted three times with different random seeds, on 512 context and 128 query rows. We use its 5- and 10-feature settings (54 tasks each), where all 2^M coalitions can be computed for exact Shapley values. **Real-40** comprises 40 public regression datasets, 25 of them from OpenML (Vanschoren et al., 2014), with 144–53,940 rows and 2–47 features (Appendix B). Each dataset is randomly split three times into 70% training and 30% test rows, and up to 128 test rows are explained (i.e., used as query rows). **Prior-500** consists of 500 tasks separately drawn from our decoupled prior (each with 3–10 features and 64–2048 samples), whose exact Shapley values are computed from the response function. We use it to analyse label quality, pretraining scale and supervision sources.

Baselines. We compare XPFN with Random Forest (**RF**) (Breiman, 2001), **XGBoost** (Chen & Guestrin, 2016), **LightGBM** (Ke et al., 2017), **CatBoost** (Prokhorenkova et al., 2018), an **MLP** and **FT-Transformer** (Gorishniy et al., 2021), **TabPFN-v3** (Grinsztajn et al., 2026) and **TabICLv2** (Qu et al., 2026), covering tree ensembles, deep tabular models and TFMs. All models are fitted on the same rows for fair comparison. On Real-40, all tree models are tuned over 50 hyperparameter candidates by three-fold cross-validation; the MLP and FT-Transformer use fixed default configurations with early stopping on 20% of the fitting rows. TabPFN-v3 and TabICLv2 use a single estimator, since KernelSHAP needs numerous forward passes per explanation. Tree models are explained by interventional TreeSHAP and the other baselines by KernelSHAP for efficiency (Lundberg & Lee, 2017; Rundel et al., 2024).

Table 1: XAI-Bench prediction and attribution against exact Shapley values. Bold and underline mark the best and second best.

	5 features			10 features		
Method	$R^2 \uparrow$	Pearson $r \uparrow$	NRMSE $\downarrow$	$R^2 \uparrow$	Pearson $r \uparrow$	NRMSE $\downarrow$
RF + TreeSHAP	0.965	0.925	0.310	0.940	0.860	0.384
XGBoost + TreeSHAP	0.973	**0.975**	0.195	0.951	0.943	0.272
LightGBM + TreeSHAP	0.948	0.950	0.277	0.943	0.907	0.336
CatBoost + TreeSHAP	0.968	0.948	0.327	0.952	0.907	0.385
MLP + KernelSHAP	0.938	0.929	0.291	0.906	0.871	0.384
FT-Transformer + KernelSHAP	0.951	0.953	0.285	0.941	0.879	0.339
TabPFN-v3 + KernelSHAP	**0.994**	<u>0.970</u>	**0.141**	**0.988**	**0.957**	**0.198**
TabICLv2 + KernelSHAP	<u>0.988</u>	0.954	<u>0.177</u>	0.976	0.940	<u>0.221</u>
XPFN-9.8M	0.986	0.954	0.227	<u>0.977</u>	<u>0.955</u>	0.257

Table 2: Real-40 prediction results (in terms of R^2).

Model	Capped	Full
RF	0.673	0.756
XGBoost	0.687	0.765
LightGBM	0.688	0.763
CatBoost	0.703	0.773
MLP	0.675	0.752
FT-Transformer	0.669	0.752
TabPFN-v3	<u>0.740</u>	**0.796**
TabICLv2	**0.741**	<u>0.792</u>
XPFN-9.8M	0.701	0.763

Table 3: Real-40 prediction correlation (Fit R^2) and attribution fidelity of XPFN against the Shapley values of baseline models and XPFN itself (last row).

Explained predictor	Fit $R^2 \uparrow$	Pearson $r \uparrow$	NRMSE $\downarrow$
RF	0.912	0.862	0.488
XGBoost	0.924	0.871	0.434
LightGBM	0.936	0.872	0.437
CatBoost	0.915	0.869	0.462
MLP	0.954	0.872	0.386
FT-Transformer	0.969	0.885	0.399
TabPFN-v3	0.893	0.855	0.447
TabICLv2	0.890	0.862	0.471
XPFN-9.8M (own predictions)	–	0.913	0.262

Reference attributions and metrics. We score the estimated attributions against reference Shapley values (Fonseca & Stoyanovich, 2026; Sena & Azevedo, 2026). On XAI-Bench, the reference is the exact Shapley values of the data-generating function, so the score also reflects the explained predictor's own error. On Real-40, where no ground truth exists, the reference is the Shapley values of the explained predictor itself (*self-fidelity*). We report R^2 for prediction, and for attribution the **Pearson correlation** r between a query's M attributions and the reference, averaged over queries, and the **NRMSE** (Appendix B); Appendix C.3 and the ablation tables also report **Spearman** and **Cosine** (Liu et al., 2021; Sena & Azevedo, 2026).

4.2 Evaluation on a Synthetic Benchmark

We first evaluate XPFN on XAI-Bench, whose generating functions differ from our prior (Table 1). Across methods, TabPFN-v3, explained with KernelSHAP, is the strongest overall in prediction and attribution, followed by TabICLv2 and our XPFN. At ten features XPFN ranks second in R^2, while scores 0.955 in Pearson correlation, which is close to the 0.957 of TabPFN-v3. In addition, XPFN achieves a NRMSE of 0.257, below that of all tree ensemble and deep tabular baselines. Note that XPFN achieves this from one forward pass of milliseconds per task. Across two different dimensionalities, XPFN remains stable in performance compared to many baseline models (such as XGBoost and FT-Transformer), indicating the strong zero-shot tabular prediction and attribution ability via intense pretraining.

4.3 Real-World Tasks

Predictive performance. We first examine whether learning attributions costs predictive accuracy on real-world datasets (Table 2). With a 2,048-row context (marked as Capped), TabICLv2

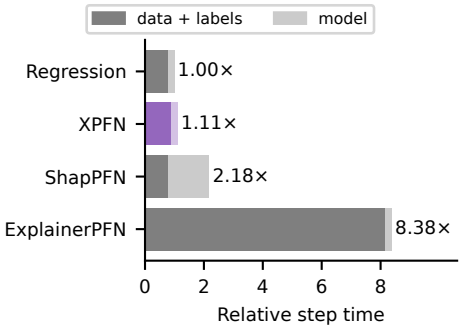

Figure 3: Per-step pretraining time relative to regression pretraining (Table 10).

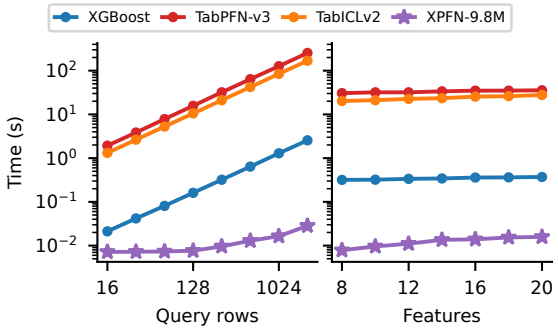

Figure 4: Explanation time against query rows (10 features) and features (256 query rows).

and TabPFN-v3 predict best ($R^2 = 0.741$ and 0.740), followed by CatBoost (0.703) and our XPFN (0.701), **stronger than tuned XGBoost and LightGBM** (0.687, 0.688) and exceeds FT-Transformer without any fitting or tuning on the task. With the whole training split given as context (marked as Full in Table 2), XPFN exhibits similar predictive performance as tuned XGBoost and LightGBM. This indicates that these supervised models benefit from increased training samples, while our XPFN was trained on tables with up to 2.048 rows.

Self-fidelity. We then compare XPFN's direct output attributions with the those estimated by applying KernelSHAP to its own predictions (last row of Table 3). We observe that these two results reach a Pearson correlation of 0.913. Since KernelSHAP on the same predictor needs a median of 36 coalitions to reach this correlation, we demonstrate that XPFN is at median $250\times$ **faster** than applying KernelSHAP to a similar but prediction-only TFM (Appendix B).

Explaining existing black-box models. In practice, the model to be explained is often an existing predictor, such as a complex ensembled model or another TFM. Post-hoc explainers then either run heavy computation over feature coalitions, as KernelSHAP does, or require access to the model's internal structure, as TreeSHAP does. **Using the model's predictions as context targets essentially turns XPFN into an explainer of that model**, which needs only its predictions on the context rows, without access to the model itself. Note that its attributions are accurate to the extent that XPFN's predictions match those of the model. As shown in Table 3, XPFN imitates the baseline models with fit R^2 of 0.89–0.97 and recovers their Shapley values with $r = 0.855$–0.885, without any model-specific explainer.

4.4 Computational Efficiency

Pretraining. We first compare the per-step pretraining cost of the three supervision paradigms compared in Figure 1. We implement both alternatives based on the original work (Sena & Azevedo, 2026; Fonseca & Stoyanovich, 2026) and adapt to regression. ShapPFN-style labels follow the Monte Carlo scheme of Sena & Azevedo (2026), and ExplainerPFN-style labels follow the official surrogate recipe of Fonseca & Stoyanovich (2026). Coalition supervision from f_τ increases the step time of regression pretraining by 11.2%, whereas ShapPFN-style supervision needs $2.18\times$ and ExplainerPFN-style supervision $8.38\times$ the time (see Figure 3 and Table 10 for details). Coalition labels drawn from our prior thus add little to the cost of regression pretraining.

Zero-shot inference. In general, with the number of query rows increases from 16 to 2,048, the explanation time of XPFN-9.8M grows from 0.007 to 0.028 seconds, $90\times$ faster than XGBoost with TreeSHAP and about $5{,}900\times$ and $8{,}900\times$ faster than TabICLv2 and TabPFN-v3 with KernelSHAP at 2,048 query rows (Figure 4). At 20 features the margins remain over $1{,}700\times$. Note these baseline times exclude any model fitting and hyperparameter tuning.

Table 4: Prior choice ablation with XPFN-6.2M on Real-40 (Appendix A, Appendix B).

Prior	$R^2 \uparrow$	$r \uparrow$	Cos. $\uparrow$	$\rho \uparrow$
BASE	0.623	0.834	0.847	0.628
BASE-ENS	0.671	0.828	0.835	0.658
FULL	0.626	**0.846**	**0.853**	0.649
FULL-ENS	**0.679**	0.841	0.849	**0.662**

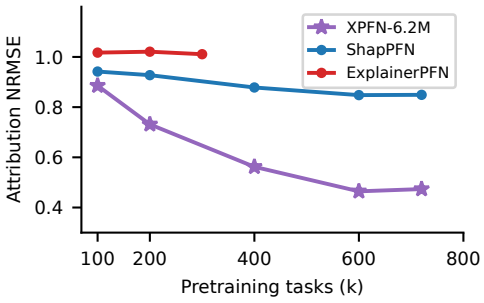

Figure 5: Attribution NRMSE against exact Shapley values on Prior-500 along pretraining, by supervision source (same task stream).

4.5 ABLATION STUDIES

Supervision paradigm. Here we compare the three supervision paradigms of Figure 1 in three aspects. First, we compare the labels used for pretraining. On Prior-500, where the response function is known, Shapley values solved from the exact coalition outputs of f_τ reach the lowest NRMSE of 0.092, whereas the MLP surrogate of ExplainerPFN and the ShapPFN-style Monte Carlo labels raise it to 0.380 and 0.384 (Appendix C.1). Second, we compare the training cost. On the same 720k-task training experiment, XPFN trains in 50 h and the ShapPFN model in 77 h, while ExplainerPFN had processed 33% of the tasks after twice this time (Appendix C.5). Finally, we compare the trained models. Between 100k and 720k tasks (Figure 5), we could observe that the attribution NRMSE of XPFN model falls from 0.89 to 0.47 and that of the ShapPFN from 0.94 to 0.85, while ExplainerPFN remains at 1.01–1.02 up to 300k tasks (Appendix C.2). We also observe a lower explanation fidelity for ShapPFN than XPFN on Real-40 datasets ($r = 0.64$ vs. 0.88), reaffirming the effectiveness of our design.

Response-function prior. We show in Table 4 that using full family of response functions (denoted as FULL) improves over a base prior of MLP and tree families (denoted as BASE). In addition, the function ensemble strategy (marked as ENS) further raise R^2 from 0.626 to 0.679 at a small cost in Pearson correlation. We therefore choose to use the full prior in XPFN pretraining.

Loss weight. With targets standardized within each task, the three terms of Equation 9 share a common scale. Raising λ from 0 to 2 lifts the Pearson correlation from 0.809 to 0.863, while R^2 falls from 0.659 to 0.601 (Appendix C.4). Although $\lambda = 0.5$ matches $\lambda = 1$ in Pearson correlation with a higher R^2, we keep $\lambda = 1$ rather than tune it on Real-40.

5 CONCLUSION AND FUTURE WORK

XPFN removes the supervision bottleneck of pretraining amortized attribution at scale by drawing the supervision directly from a decoupled prior. This prior generates Shapley values with a lower error against previous methods, while speeding up the training process of XPFN.

Pretrained on 12 million tasks, XPFN returns accurate predictions, and Shapley attributions in one forward pass. Experiments on synthetic and real-world benchmarks show that XPFN retains strong predictive performance comparable to tuned Gradient-Boosted Decision Tree models, recovers high-fidelity Shapley attributions, and substantially reduces both pretraining overhead and explanation latency.

XPFN is currently limited to regression and to empirical-interventional attribution over the context distribution. State-of-the-art TFMs such as TabPFN-v3 and TabICLv2 remain more accurate predictors overall. Future work includes extending XPFN to classification tasks and post-training.

Reproducibility statement

Appendix A specifies the prior, Appendix B the pretraining step (Algorithm 1), the baselines and the evaluation protocol, Appendix C the additional results, and Appendix D the proof of Proposition 1.

AI use statement

AI coding agents were used to implement and run evaluation and benchmarking scripts, monitor training runs and aggregate results. Generative AI tools were also used to assist with manuscript wording, integrating reported results into the text and tables, and LaTeX formatting. The authors are responsible for all content of the paper.

References

Amr Alkhatib, Roman Bresson, Henrik Boström, and Michalis Vazirgiannis. Prediction via Shapley value regression. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 1056–1101. PMLR, 2025.

Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 2001. doi: 10.1023/A:1010933404324.

Hugh Chen, Joseph D. Janizek, Scott M. Lundberg, and Su-In Lee. True to the model or true to the data? arXiv preprint arXiv:2006.16234, 2020.

Tianqi Chen and Carlos 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. doi: 10.1145/2939672.2939785.

Ian Covert and Su-In Lee. Improving KernelSHAP: Practical Shapley value estimation using linear regression. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 3457–3465. PMLR, 2021.

Ian Covert, Scott Lundberg, and Su-In Lee. Explaining by removing: A unified framework for model explanation. *Journal of Machine Learning Research*, 22(209):1–90, 2021.

Ian Covert, Chanwoo Kim, and Su-In Lee. Learning to estimate Shapley values with vision transformers. In *International Conference on Learning Representations*, 2023.

Ian Covert, Chanwoo Kim, Su-In Lee, James Zou, and Tatsunori Hashimoto. Stochastic amortization: A unified approach to accelerate feature and data attribution. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

James Enouen and Yan Liu. InstaSHAP: Interpretable additive models explain Shapley values instantly. In *International Conference on Learning Representations*, 2025.

João Fonseca and Julia Stoyanovich. ExplainerPFN: Towards tabular foundation models for model-free zero-shot feature importance estimations. arXiv preprint arXiv:2601.23068, 2026.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. In *Advances in Neural Information Processing Systems*, volume 34, pp. 18932–18943, 2021.

Léo Grinsztajn et al. TabPFN-3: Technical report. arXiv preprint arXiv:2605.13986, 2026.

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

Dominik Janzing, Lenon Minorics, and Patrick Blöbaum. Feature relevance quantification in explainable AI: A causal problem. In *Proceedings of the Twenty-Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 2907–2916. PMLR, 2020.

Neil Jethani, Mukund Sudarshan, Ian Connick Covert, Su-In Lee, and Rajesh Ranganath. FastSHAP: Real-time Shapley value estimation. In *International Conference on Learning Representations*, 2022.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Yang Liu, Sujay Khandagale, Colin White, and Willie Neiswanger. Synthetic benchmarks for scientific research in explainable machine learning. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021.

Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, volume 30, pp. 4765–4774, 2017.

Scott M. Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M. Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. From local explanations to global understanding with explainable AI for trees. *Nature Machine Intelligence*, 2:56–67, 2020. doi: 10.1038/s42256-019-0138-9.

Andreas Mueller, Julien Siems, Harsha Nori, David Salinas, Arber Zela, Rich Caruana, and Frank Hutter. GAMformer: Bridging tabular foundation models and interpretable machine learning. arXiv preprint arXiv:2410.04560, 2024.

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

Lars Henry Berge Olsen and Dennis Christensen. Computing conditional Shapley values using tabular foundation models. arXiv preprint arXiv:2602.09489, 2026.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. CatBoost: Unbiased boosting with categorical features. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 50817–50847. PMLR, 2025.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICLv2: A better, faster, scalable, and open tabular foundation model. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*. PMLR, 2026.

Jake Robertson, Arik Reuter, Siyuan Guo, Noah Hollmann, Frank Hutter, and Bernhard Schölkopf. Do-PFN: In-context learning for causal effect estimation. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

David Rundel, Julius Kobialka, Constantin von Crailsheim, Matthias Feurer, Thomas Nagler, and David Rügamer. Interpretable machine learning for TabPFN. In Luca Longo, Sebastian Lapuschkin, and Christin Seifert (eds.), *Explainable Artificial Intelligence*, volume 2154 of *Communications in Computer and Information Science*, pp. 465–476, Cham, 2024. Springer. doi: 10.1007/978-3-031-63797-1_23.

Luan Borges Teodoro Reis Sena and Francisco Galuppo Azevedo. Real-time explanations for tabular foundation models. 2nd DATA4Science Workshop at ICLR 2026; arXiv:2603.29946, 2026.

Lloyd S. Shapley. A value for n-person games. In Harold W. Kuhn and Albert W. Tucker (eds.), *Contributions to the Theory of Games II*, volume 28 of *Annals of Mathematics Studies*, pp. 307–317. Princeton University Press, 1953.

Mukund Sundararajan and Amir Najmi. The many Shapley values for model explanation. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9269–9278. PMLR, 2020.

Valentin Thomas, Junwei Ma, Rasa Hosseinzadeh, Keyvan Golestan, Guangwei Yu, Maksims Volkovs, and Anthony Caterini. Retrieval & fine-tuning for in-context tabular models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 108439–108467, 2024.

Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. OpenML: Networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60, 2014. doi: 10.1145/2641190.2641198.

Rui Wang, Xiaoqian Wang, and David I. Inouye. Shapley explanation networks. In *International Conference on Learning Representations*, 2021.

Derek Qiang Xu, F. Olcay Cirit, Reza Asadi, Yizhou Sun, and Wei Wang. Mixture of in-context prompters for tabular PFNs. In *International Conference on Learning Representations*, 2025.

Xingxuan Zhang et al. LimiX: Unleashing structured-data modeling capability for generalist intelligence. arXiv preprint arXiv:2509.03505, 2025a.

Xiyuan Zhang, Danielle Maddix Robinson, Junming Yin, Nick Erickson, Abdul Fatir Ansari, Boran Han, Shuai Zhang, Leman Akoglu, Christos Faloutsos, Michael W. Mahoney, Tony Hu, Huzefa Rangwala, George Karypis, and Yuyang (Bernie) Wang. Mitra: Mixed synthetic priors for enhancing tabular foundation models. In *Advances in Neural Information Processing Systems*, volume 38, 2025b.

A Synthetic Task Prior

Features. Each task has 64–2,048 rows and 2–32 feature columns. We sample a Cauchy or hierarchical DAG with equal probability, with 8–24 levels, at most five parents per node and 2–64 latent coordinates per node (Qu et al., 2026). Root inputs are drawn from continuous, count, binary and categorical source families; non-root states are produced by sampled linear, smooth, rule-based, random-basis and tree-ensemble mechanisms. Rows are independent, drawn from a prototype mixture, near-duplicates, or placed on a latent manifold. The observed columns are augmented with redundant copies, transformations, categorical values, missing values and independently sampled noise columns (Hollmann et al., 2025; Qu et al., 2026).

Response function. After all feature transformations, a deterministic response function f_τ is drawn: a single function or, with probability 0.5, an ensemble of two to four experts. Each expert uses a subset of the columns, and the experts are combined by a weighted, gated or interaction sum. The function families are linear and smooth functions, rules, random-feature networks, and boosted-tree models, **all sampled rather than fitted to the task**. Each noise column is excluded from the candidate columns of f_τ with probability 0.75. Therefore, every column that f_τ does not use is a null player by construction. Because columns are selected individually, correlated columns can be jointly informative, or only one of them can enter f_τ. The clean function defines the query targets and the coalition values. Context labels receive Gaussian, Student-t, mixture or outlier noise (base probabilities 0.55, 0.20, 0.15 and 0.10).

Queries, background and masks. The query fraction is drawn from 0.10–0.35; the other rows form the context. The background $\mathcal{D}_b$ is a uniformly random subset of 24–128 context rows (fewer if the context is smaller). If $2^M - 2 \leq 32$ ($M \leq 5$), all nonempty, nonfull masks are enumerated and weighted by $p_{\mathrm{Sh}}(\mathbf{s})$ in $\mathcal{L}_{\mathrm{coal}}$ (see Equation 5). Otherwise, an even number $K \approx 3M$ of masks, clipped to 16–192, is drawn from p_{Sh} in complementary pairs (Jethani et al., 2022). The same clean f_τ is evaluated for every background-replaced query of a task. Because $\mathcal{D}_b$ is a uniformly random subset of the context, the coalition labels are unbiased estimates of the coalition values with the whole context as background (Equation 4), which is the background XPFN uses implicitly.

B Training and Evaluation Details

Main model. XPFN-9.8M has 16 Transformer layers with embedding size 192, three attention heads and a feed-forward width of 768. It is trained on 12 million tasks, 60 per update, with AdamW (learning rate 6×10^{-5}, weight decay 0.01), a 500-update linear warm-up at the beginning of the retraining, a constant learning rate up to 20% of training and then cosine decay to 6×10^{-6}, in bfloat16 mixed precision. Training ran on three NVIDIA H100 GPUs with data parallelism for about 12 days (about 35 GPU-days).

Algorithm 1 One XPFN pretraining step (per task; tasks are batched per update)

Require: model g_ω; loss weight λ; mask-budget rule and background-size range $[n_{\min}, n_{\max}]$ (Appendix A)

1: Sample a DAG and its mechanisms from the SCM prior; propagate root states, collect feature columns, apply transformations and append noise features ▷ $\mathbf{X} \in \mathbb{R}^{N \times M}$
2: Sample the excluded set $E \subseteq \mathcal{F}_{\text{noise}}$, the active set $A \subseteq \mathcal{F} \setminus E$ and a response function f_τ (single or ensemble) from the response-function prior
3: Split the rows into a context set $\mathcal{D}_c$ and query rows $\mathbf{X}_q$
4: $y_i \leftarrow f_\tau(\mathbf{x}_i) + \varepsilon_i$ for context rows and $y_q \leftarrow f_\tau(\mathbf{x}_q)$ for query rows ▷ label noise ε_i
5: Draw $n_b \in [n_{\min}, n_{\max}]$ and a uniformly random background $\mathcal{D}_b \subseteq \mathcal{D}_c$ of n_b feature rows
6: **for all** query rows $\mathbf{x}_q$ **do**
7: If $2^M - 2 \leq 32$, take all $K = 2^M - 2$ nonempty, nonfull masks, weighted by p_{Sh} in $\mathcal{L}_{\text{coal}}$; otherwise draw $K/2$ masks from p_{Sh} (Equation 5) and add their complements
8: Evaluate $v_\tau(\mathbf{x}_q, \mathbf{s})$ by Equation 4 for $\mathbf{s}_0 = \mathbf{0}$ and $\mathbf{s}_1, \ldots, \mathbf{s}_K$ ▷ $n_b(K+1)$ calls of f_τ
9: If with the auxiliary term, solve the Shapley-kernel-weighted least-squares problem on $\{\mathbf{s}_k\}$ with endpoints $v_\tau(\mathbf{x}_q, \mathbf{0})$, $f_\tau(\mathbf{x}_q)$ and $\phi_j = 0$ for $j \in E$, giving $\tilde{\boldsymbol{\phi}}_q$
10: **end for**
11: Standardize labels and coalition values as $(\cdot - \mu_c)/\sigma_c$, with μ_c, σ_c from the context labels, and divide $\tilde{\boldsymbol{\phi}}_q$ by σ_c
12: $(\hat{b}, \hat{\boldsymbol{\Phi}}_q) \leftarrow g_\omega(\mathcal{D}_c, \mathbf{X}_q)$ ▷ one forward pass
13: $\mathcal{L} \leftarrow \mathcal{L}_{\text{base}} + \lambda \mathcal{L}_{\text{coal}} + (2 - \lambda)\mathcal{L}_{\text{pred}}$ ▷ Equation 9
14: If with the auxiliary term, $\mathcal{L} \leftarrow \mathcal{L} + \mathbb{E}_q \|\hat{\boldsymbol{\phi}}_q - \tilde{\boldsymbol{\phi}}_q\|^2$
15: Update ω with AdamW on $\nabla_\omega \mathcal{L}$, averaged over the tasks of the batch

Ablation models. The prior and loss-weight ablations (Tables 4 and 9) use XPFN-6.2M models trained for 43.2k updates on 3.5 million tasks. They are evaluated on one Real-40 split with at most 512 context rows, 32 background rows and a 512-mask KernelSHAP reference computed on the same model. In the ablation of three supervision paradigms, XPFN-6.2M models are trained on an identical sequence of 720k tasks (Table 11).

Real-40 datasets. Table 5 lists the 40 datasets: 25 from OpenML (Vanschoren et al., 2014), six from the UCI repository, two from scikit-learn and seven from the seaborn sample datasets. Categorical columns are taken from explicit metadata rather than inferred from values. Each dataset is split three times at random into 70% training and 30% test rows (we use seeds 20260408, 20260409 and 20260410). In the Capped setting, `analcatdata_supreme` has a constant target on the scored rows, so R^2 is averaged over the other 39 datasets.

Metrics. For a task with query attributions $\hat{\boldsymbol{\Phi}}_q$ and reference $\boldsymbol{\Phi}_q \in \mathbb{R}^{N_q \times M}$, with rows $\hat{\boldsymbol{\phi}}_q$ and $\boldsymbol{\phi}_q$,

$$r = \frac{1}{N_q} \sum_q \operatorname{corr}\big(\hat{\boldsymbol{\phi}}_q, \boldsymbol{\phi}_q\big), \qquad \text{NRMSE} = \frac{\big(\frac{1}{N_q M} \sum_{q,m} (\hat{\phi}_{q,m} - \phi_{q,m})^2\big)^{1/2}}{\operatorname{std}\big(\{\phi_{q,m}\}_{q,m}\big)},$$

where corr is the Pearson correlation over the M features and queries with a constant reference or estimate are skipped. R^2 is computed over the scored test rows of a task. All metrics are averaged over seeds within a dataset and then over datasets.

Evaluation. Unless stated otherwise, all methods share the same context in evaluation (at most 2,048 rows on Real-40, 512 on XAI-Bench), 64 context rows as background for references and post-hoc baselines, and up to 128 query rows. The Full setting of Table 2 uses the whole training split as context and scores all test rows. Reference attributions are exact where feasible, by TreeSHAP for tree models, or by enumerating all coalitions for $M \leq 12$, and otherwise use 4,096 Shapley-kernel masks.

Equivalent KernelSHAP budget. To compare XPFN with KernelSHAP at equal accuracy, we define the equivalent budget B^* of a dataset as the number of coalitions that KernelSHAP needs on the same predictor to match the Pearson correlation of XPFN's direct attributions. Section 4.3 reports its median (36 coalitions) and the resulting speedup ($250\times$). To find B^*, we re-solve KernelSHAP from subsets of the stored reference coalitions at budgets $32, 64, \ldots, 2{,}048$, capped at the 2^M coalitions that exist. For each budget, we draw five subsets in proportion to the kernel weights, always keeping the empty and the full coalition, and average their per-query Pearson correlation with the reference. Budgets whose least-squares system is not full rank are skipped. B^* is the smallest budget whose correlation reaches that of XPFN, interpolated linearly between adjacent budgets. On 31 of the 40 datasets, KernelSHAP already matches XPFN at the smallest full-rank budget, which we then take as B^*. Finally, we multiply B^* by the measured time of one coalition evaluation to obtain the KernelSHAP time and report medians over datasets.

C Additional Results

C.1 Label Quality

Table 6: Coalition-label quality on Prior-500 (500 tasks, 32 queries each). All sources use the same $K \leq 64$ masks and 64-row background. Coalition NRMSE compares the labels with the exact $v_\tau(\mathbf{x}_q, \mathbf{s})$; ϕ NRMSE and Pearson compare the Shapley values solved from the labels by kernel-weighted least squares with the exact Shapley values. With the exact labels of f_τ, the remaining error comes from the finite mask budget. The lower rows take a trained model's coalition predictions as labels. Means over tasks; the surrogate row averages five repeats.

Coalition-label source	Coalition NRMSE ↓	ϕ NRMSE ↓	ϕ Pearson ↑
Generating function f_τ	0.000	0.092	0.988
MLP surrogate (ExplainerPFN@300k tasks)	0.367	0.380	0.873
XPFN-9.8M	0.181	0.204	0.958
XPFN-6.2M@720k tasks	0.374	0.354	0.900
ShapPFN@720k tasks	0.383	0.384	0.884

Table 5: Real-40 datasets. Rows and features after preprocessing; Cat.: number of categorical features; Capped: more than 2,048 training rows, so the Capped setting of Table 2 subsamples the context. [§]More features than the 32 seen in pretraining. [‡]Constant target on the scored rows of the Capped setting; R^2 is undefined there.

Dataset	Source	Rows	Features	Cat.	Capped
abalone	UCI	4,177	10	–	✓
ailerons	OpenML	13,750	40[§]	–	✓
airfoil_self_noise	UCI	1,503	5	–	
analcatdata_supreme[‡]	OpenML	4,052	7	–	✓
bank32nh	OpenML	8,192	32	–	✓
bank8fm	OpenML	8,192	8	–	✓
bodyfat	OpenML	252	14	–	
boston	OpenML	506	22	–	
california_housing	scikit-learn	20,640	8	–	✓
combined_cycle_power_plant	UCI	9,568	4	–	✓
concrete_strength	UCI	1,030	8	–	
cps_85_wages	OpenML	534	23	–	
cpu_act	OpenML	8,192	21	–	✓
cpu_small	OpenML	8,192	12	–	✓
dataset_sales	OpenML	10,738	14	–	✓
delta_elevators	OpenML	9,517	6	–	✓
diabetes	scikit-learn	442	10	–	
diamonds	OpenML	53,940	9	3	✓
elevators	OpenML	16,599	18	–	✓
flights	seaborn	144	2	1	
healthexp_life_expectancy	seaborn	274	8	–	
kaggle_bike_sharing_demand	OpenML	10,886	47[§]	–	✓
kin8nm	OpenML	8,192	8	–	✓
machine_cpu	OpenML	209	6	–	
mpg	seaborn	392	9	–	
penguins_body_mass	seaborn	342	11	–	
pol	OpenML	15,000	26	–	✓
pollen	OpenML	3,848	4	–	✓
puma8nh	OpenML	8,192	8	–	✓
rainfall_bangladesh	OpenML	16,755	46[§]	–	✓
space_ga	OpenML	3,107	6	–	✓
stock	OpenML	950	9	–	
student_performance_por	OpenML	649	30	17	
swd	OpenML	1,000	10	–	
taxis_fare	seaborn	6,433	16	–	✓
tips	seaborn	244	6	4	
titanic_fare	seaborn	891	11	6	
wind	OpenML	6,574	14	–	✓
wine_quality_red	UCI	1,599	11	–	
wine_quality_white	UCI	4,898	11	–	✓

C.2 Pretraining Scale

Table 7: Direct attributions against exact Shapley values on Prior-500, and Real-40 R^2 (Capped setting), along pretraining. Note that ExplainerPFN does not give predictive output, only attributions.

Run	Tasks	ϕ NRMSE ↓	Pearson r ↑	Real-40 R^2 ↑
XPFN-9.8M	2.43M	0.367	0.865	0.621
	5.16M	0.346	0.872	0.633
	6.75M	0.322	0.885	0.565
	8.39M	0.303	0.890	0.699
	9.40M	0.293	0.895	0.660
	11.54M	0.287	0.898	0.686
XPFN-6.2M	100k	0.886	0.565	−0.685
	200k	0.731	0.697	0.348
	400k	0.562	0.770	−0.058
	600k	0.464	0.812	0.471
	720k	0.473	0.816	0.297
ShapPFN	100k	0.942	0.465	−0.875
	200k	0.927	0.551	0.126
	400k	0.878	0.695	0.264
	600k	0.848	0.698	0.370
	720k	0.849	0.601	0.339
ExplainerPFN	100k	1.017	0.192	–
	200k	1.021	0.161	–
	300k	1.011	0.133	–

C.3 Spearman and Cosine Agreement

Table 8: Spearman rank correlation ρ and cosine similarity between each query's attributions and the reference, averaged as in Tables 1 and 3.

	XAI-Bench (Table 1)				Real-40 (Table 3)		
	5 features		10 features		Explained		
Method	ρ ↑	Cos. ↑	ρ ↑	Cos. ↑	predictor	ρ ↑	Cos. ↑
RF + TreeSHAP	0.891	0.949	0.748	0.914	RF	0.763	0.874
XGBoost + TreeSHAP	<u>0.934</u>	**0.982**	<u>0.819</u>	**0.965**	XGBoost	0.775	0.888
LightGBM + TreeSHAP	0.916	0.967	0.789	0.943	LightGBM	0.773	0.884
CatBoost + TreeSHAP	0.902	0.950	0.769	0.931	CatBoost	0.780	0.884
MLP + KernelSHAP	0.873	0.937	0.715	0.883	MLP	0.779	0.885
FT-Tr. + KernelSHAP	0.906	0.951	0.752	0.912	FT-Transformer	0.814	0.902
TabPFN-v3 + KernelSHAP	**0.940**	<u>0.978</u>	**0.840**	<u>0.963</u>	TabPFN-v3	0.733	0.868
TabICLv2 + KernelSHAP	0.928	0.968	0.815	0.949	TabICLv2	0.767	0.860
XPFN-9.8M	0.915	0.963	<u>0.819</u>	0.960			

C.4 Loss Weight

Table 9: Loss-weight ablation with XPFN-6.2M on Real-40 (protocol in Appendix B, *Ablation models*).

λ	Prediction	Explanation fidelity		
	$R^2 \uparrow$	Pearson $r \uparrow$	Cosine $\uparrow$	Spearman $\rho \uparrow$
0.00	<u>0.659</u>	0.809	0.822	0.617
0.50	**0.663**	0.840	0.850	0.629
0.75	0.630	0.827	0.837	0.625
1.00	0.612	0.839	0.845	0.640
1.25	0.627	0.847	<u>0.855</u>	0.660
1.50	0.613	<u>0.850</u>	0.854	<u>0.684</u>
2.00	0.601	**0.863**	**0.872**	**0.713**

C.5 Supervision Paradigms

Table 10 gives the per-step cost behind Figure 3, Table 11 the training cost of the three matched runs, and Table 12 the Real-40 self-fidelity of the two 720k-task XPFN models. At evaluation, XPFN uses one fixed draw of column identifiers (Section 3.4); the two 720k-task models are always evaluated with the same draw (seed 20260919), so that they differ only in their weights.

Table 10: Per-step pretraining cost on one GPU, relative to regression-only pretraining (first row in seconds). Each step uses two tables of 256 rows and 8 features, 4 coalitions and 4 background rows per query. ExplainerPFN-style labels are the SHAP values of an MLP fitted to each context. Data time sums over all data workers.

Pretraining	Data	Model	End-to-end
Regression (s)	0.097	0.031	0.128
XPFN	1.130×	1.054×	1.112×
ShapPFN	1.000×	5.940×	2.180×
ExplainerPFN	10.702×	0.985×	8.380×

Table 11: Training cost of the matched 720k-task models (identical task sequence; two RTX 6000 Ada GPUs, 32 CPU cores, 40 tasks per update). Wall-clock covers the full pipeline, including on-the-fly task and label generation, on larger tables than Table 10.

Model	Wall-clock (h)	Tasks per hour
XPFN-6.2M	50.5	14,300
ShapPFN-6.2M	76.7	9,400
ExplainerPFN-6.1M	$>98^{\dagger}$	2,450

†After 98 h, about twice XPFN's wall-clock, 238,720 of 720,000 tasks were done, and the full run would take about 294 h at this rate. We stopped it at 306,200 tasks (125 h) and evaluate that checkpoint.

Table 12: Real-40 prediction and self-fidelity of the matched 720k-task XPFN models (one split per dataset): direct attributions against the Shapley values of each model's own predictions.

Coalition targets	$R^2 \uparrow$	Pearson $r \uparrow$	NRMSE $\downarrow$
Generating function f_τ	0.223	0.875	0.398
Monte Carlo self-consistency	0.214	0.638	0.863

D MINIMIZERS OF THE PRETRAINING OBJECTIVE

In Equation 9 the empty and the full coalition enter as finite penalties rather than as the hard constraints under which the Shapley-kernel regression returns Shapley values (Lundberg & Lee, 2017). We show that this only affects the *total* attribution, not how it is divided among features. Intuitively, the endpoints see only $\hat{b}$ and $\hat{b} + \mathbf{1}^\top \hat{\boldsymbol{\phi}}$, so they cannot tell how the total is split; the split is fixed by the sampled coalitions alone, as in the constrained problem.

Fix a task, its background $\mathcal{D}_b$ and a query, and write $v(\mathbf{s})$ for its coalition values $v_\tau(\mathbf{x}_q, \mathbf{s})$, with prediction target $v(\mathbf{1}) = f_\tau(\mathbf{x}_q)$ (query targets are noise-free; Appendix A). Taking the expectation of Equation 9 over the mask draw replaces the average over the K sampled masks by an expectation under p_{Sh} (for $M \leq 5$, the p_{Sh}-weighted sum over all enumerated masks equals it exactly; pairing masks with their complements keeps this marginal, since $p_{\mathrm{Sh}}(\mathbf{s}) = p_{\mathrm{Sh}}(\mathbf{1} - \mathbf{s})$). Consider the resulting per-query objective

$$J(b, \boldsymbol{\phi}) = \big(b - v(\mathbf{0})\big)^2 + \lambda\, \mathbb{E}_{\mathbf{s}\sim p_{\mathrm{Sh}}}\big(b + \mathbf{s}^\top \boldsymbol{\phi} - v(\mathbf{s})\big)^2 + (2-\lambda)\big(b + \mathbf{1}^\top \boldsymbol{\phi} - v(\mathbf{1})\big)^2. \tag{10}$$

Proposition 1. *Let $M \geq 2$ and $\lambda \in (0, 2]$. Every minimizer $(b^*, \boldsymbol{\phi}^*)$ of J satisfies $\boldsymbol{\phi}^* = \boldsymbol{\phi}(v) + \delta\, \mathbf{1}_M$ for a scalar δ, where $\boldsymbol{\phi}(v)$ are the Shapley values of v. Hence $\boldsymbol{\phi}^*$ has the same per-query Pearson correlation with any reference and the same feature ranking as $\boldsymbol{\phi}(v)$.*

Proof. Write $\boldsymbol{\phi} = \mathbf{u} + (t/M)\mathbf{1}$ with $t = \mathbf{1}^\top \boldsymbol{\phi}$ and $\mathbf{1}^\top \mathbf{u} = 0$. Because p_{Sh} depends on $\mathbf{s}$ only through $|\mathbf{s}|$, every feature is treated alike. Hence $\mathbb{E}[h(|\mathbf{s}|)\,\mathbf{s}]$ is a multiple of $\mathbf{1}$ for any function h, so $\mathbb{E}[h(|\mathbf{s}|)\,\mathbf{s}^\top \mathbf{u}] = 0$; and $\mathbb{E}[\mathbf{s}\mathbf{s}^\top] = \alpha \mathbf{I} + \beta \mathbf{1}\mathbf{1}^\top$ with $\beta = \Pr(s_1 = s_2 = 1)$ and $\alpha = \Pr(s_1 = 1) - \beta = \Pr(s_1 = 1, s_2 = 0) > 0$, so $\mathbb{E}[(\mathbf{s}^\top \mathbf{u})^2] = \alpha \|\mathbf{u}\|^2$. Since $b + \mathbf{s}^\top \boldsymbol{\phi} = (b + t|\mathbf{s}|/M) + \mathbf{s}^\top \mathbf{u}$, the middle term of J equals $\lambda\, \mathbb{E}(b + t|\mathbf{s}|/M - v(\mathbf{s}))^2 + \lambda\big(\alpha \|\mathbf{u}\|^2 - 2\, \mathbf{u}^\top \mathbb{E}[v(\mathbf{s})\mathbf{s}]\big)$, and the two endpoint terms contain b and t but not $\mathbf{u}$. As $\lambda > 0$, $\mathbf{u}$ minimizes $\alpha \|\mathbf{u}\|^2 - 2\, \mathbf{u}^\top \mathbb{E}[v(\mathbf{s})\mathbf{s}]$ subject to $\mathbf{1}^\top \mathbf{u} = 0$, whatever b and t are. Since $\alpha > 0$, this problem is strictly convex; its unique solution $\mathbf{u}^*$, the zero-sum part of $\mathbb{E}[v(\mathbf{s})\mathbf{s}]/\alpha$, does not depend on λ, b, t or the endpoint terms.

The same separation applies when the endpoints are imposed as the constraints $b = v(\mathbf{0})$ and $b + \mathbf{1}^\top \boldsymbol{\phi} = v(\mathbf{1})$, which fix only b and t. The solution of that constrained problem is $\boldsymbol{\phi}(v)$ (Lundberg & Lee, 2017, Theorem 2), so its zero-sum part is also $\mathbf{u}^*$. Therefore $\boldsymbol{\phi}^*$ and $\boldsymbol{\phi}(v)$ differ only in their totals, i.e., by a multiple of $\mathbf{1}$. Pearson correlation and ranks are unchanged by adding the same constant to every entry. □

By efficiency, $\delta = \big[(\hat{y}^* - v(\mathbf{1})) - (b^* - v(\mathbf{0}))\big]/M$ with $\hat{y}^* = b^* + \mathbf{1}^\top \boldsymbol{\phi}^*$; the shift vanishes when the base value and the prediction are exact. Because $\mathbf{u}$ does not interact with b, the result also holds when b is shared across queries, with a query-specific δ. The auxiliary term of Appendix B is not part of J.