Tractable Shapley Values and Interactions via Tensor Networks

Farzaneh Heidari
DIRO & Mila
Université de Montréal

Chao Li
RIKEN AIP

Guillaume Rabusseau
DIRO & Mila, CIFAR AI Chair
Université de Montréal

Abstract

We show how to replace the $O(2^n)$ coalition enumeration over n features behind Shapley values and Shapley-style interaction indices with a *few-evaluation* scheme on a tensor-network (TN) surrogate: TN-SHAP. The key idea is to represent a predictor's local behavior as a factorized multilinear map, so that coalitional quantities become *linear probes* of a coefficient tensor. TN-SHAP replaces exhaustive coalition sweeps with just a small number of targeted evaluations to extract order$-k$ Shapley interactions. In particular, both order-1 (single-feature) and order-2 (pairwise) computations have cost $O\big(n\,\mathrm{poly}(\chi) + n^2\big)$, where χ is the TN's maximal cut rank. We provide theoretical guarantees on the approximation error and tractability of TN-SHAP. On UCI datasets, our method matches enumeration on the fitted surrogate while reducing evaluation by orders of magnitude and achieves **25–1000**$\times$ wall-clock speedups over KernelSHAP-IQ at comparable accuracy, while amortizing training across local cohorts.

1 INTRODUCTION

Explaining *how* features (individually and jointly) drive predictions is crucial for decision support, scientific discovery, and mechanistic interpretability. Shapley values (Shapley, 1953) and their higher-order generalizations (Shapley–Taylor, Shapley Interaction Indices; SII Grabisch and Roubens, 1999a) give principled, symmetry-respecting attributions. Yet, outside special architectures, computing these indices remains difficult in practice.

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300. Copyright 2026 by the author(s).

Existing approaches either (i) leverage structural assumptions (e.g., trees (Lundberg et al., 2018), linear/GAM-style models (Sundararajan et al., 2020; Bordt and von Luxburg, 2023)), (ii) rely on heavy sampling with delicate variance/weighting tuning (Lundberg and Lee, 2017; Fumagalli et al., 2024), or (iii) amortize estimation with additional training overhead (Jethani et al., 2022). In this work, we follow the same general principle: we show that when a predictor can be well approximated by a low-rank multilinear tensor-network surrogate in a lifted feature space, Shapley values and Shapley-style interactions can be computed exactly and efficiently in practice.

In this paper, we introduce TN-SHAP: a method to compute Shapley values and Shapley-style interactions *exactly on a multilinear TN function or surrogate*, using only a handful of structured evaluations. For non-multilinear predictors, approximation quality depends on surrogate fidelity. To reduce $O(2^n)$ complexity to $O(n)$, TN-SHAP uses three insights:

First, we exploit the algebraic structure of multilinear functions to map coalition queries directly to tensor coefficients. Since each Shapley marginal contribution $v(C \cup \{i\}) - v(C)$ (see Section 2.1) corresponds to a specific subset of monomial coefficients in the multilinear expansion, we can extract these values without explicitly evaluating all 2^n coalitions—the tensor representation already encodes them (Owen, 1972; Roth, 1988).

Second, we introduce diagonal selector matrices that transform the combinatorial problem of coalition enumeration into polynomial interpolation. By augmenting feature inputs with thin diagonal matrices $S_r(t) = \mathrm{Diag}(t, 1)$, we aggregate all coalitions of the same size into powers of t, allowing TN-SHAP to recover all n size-aggregated marginal contributions by solving a single $n \times n$ linear system rather than querying exponentially many subsets.

Third, we leverage tensor network factorizations to make multilinear evaluation tractable at scale. By decomposing the coefficient tensor into a TN with

bond dimension χ, we reduce the cost of each forward pass from $O(2^n)$ to $O(\text{poly}(\chi))$, enabling the n evaluations required for interpolation to be completed in $O(n \cdot \text{poly}(\chi))$ time total, achieving polynomial rather than exponential scaling in the number of features. Our main contributions can be summarized as follows:

- **Unified probe–interpolation scheme:** We develop an exact algorithm for computing Shapley values and k-way interactions through structured TN evaluations on the surrogate function, requiring only $O(n)$ forward passes instead of $O(2^n)$ coalition queries (Section 3).
- **Feature-map–enhanced TN surrogates:** We show how learned or hand-crafted feature lifts can preserve the multilinearity required for exact computation while improving surrogate fidelity (Section 3.3).
- **Empirical validation:** On UCI benchmarks and synthetic teachers, our method achieves 25–1000× wall-clock speedups over KernelSHAP-IQ at comparable accuracy while guaranteeing exactness on the surrogate (Section 5).

2 BACKGROUND AND PRELIMINARIES

We summarize coalitional values (Shapley and Shapley-style interactions), the multilinear extension used to link them to coefficients, and the tensor/tensor-network (TN) notation we will use throughout.

2.1 Coalitional Values and the Shapley Value

Let $N = \{1, \dots, n\}$ index features and let $C \subseteq N$. For an input $x \in \mathbb{R}^n$, a *coalition value* $v(x, C)$ is the expected model output when features in C are clamped to x_C (restriction of x to coordinates in C) and the complement $\bar{C}$ is marginalized. We consider two standard choices:

$$\textsc{Obs}: \quad v_{\text{obs}}(x, C) = \mathbb{E}_{z\sim D}[f(z) \,|\, z_C = x_C]\,, \tag{2.1}$$

$$\textsc{Int}: \quad v_{\text{int}}(x, C) = \mathbb{E}_{z\sim D}[f(x_C, z_{\bar{C}})]\,. \tag{2.2}$$

where expectations are over the data distribution D. The Shapley value $\phi^{\text{SV}}(i; x)$ of feature $i \in N$ (Shapley, 1953) is

$$\sum_{s=0}^{n-1} \frac{s!(n-s-1)!}{n!} \sum_{\substack{C\subseteq N\setminus\{i\} \\ |C|=s}} [v(x, C\cup\{i\}) - v(x, C)].$$

Shapley values are weighted averages of marginal contributions where weights depend *only on coalition size* $|C|$, not on specific coalition membership. This size-aggregation structure is precisely what multilinear representations used by TN-SHAP can exploit for efficient computation.

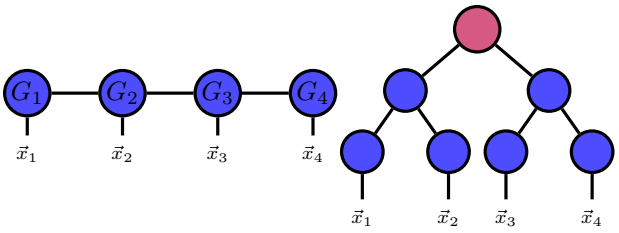

Figure 1: Two common tensor-network topologies used in this work: a tensor train (left) and a balanced binary tree (right). Only physical input legs $\vec{x}_k$ are shown.

Tractable Shapley values with multilinearity The multilinear extension (MLE) is the critical bridge between exponentially-many coalitions and polynomial-time computation. A multilinear function has the form

$$f(x) = \textstyle\sum_{T\subseteq N} c_T \prod_{j\in T} x_j, \tag{2.3}$$

where each coefficient c_T encodes the contribution of the monomial indexed by the subset T. In a multilinear representation, each coalition query $v(x, C)$ reads a *subset of coefficients* whose indices are contained in C: setting $x_j = 1$ for $j \in C$ and $x_j = 0$ otherwise yields $f(x) = \sum_{T\subseteq C} c_T$. This means:

- Coalition values are linear combinations of tensor coefficients c_T.
- Shapley marginals $v(x, C\cup\{i\}) - v(x, C)$ correspond to specific coefficient subsets.
- The 2^n coalition queries reduce to structured reads from a tensor.

Collecting the coefficients c_T into an order-n tensor $\mathcal{T}$, we can view f as a multilinear map, and this tensor structure is what tensor networks will make tractable.

2.2 Tensors and Tensor Networks

Tensors as multilinear maps Let $\tilde{x}_i \in \mathbb{R}^{d_i}$ be $[x_i, 1]^\top$. An order-n tensor $\mathcal{T} \in \mathbb{R}^{d_1\times\cdots\times d_n}$ induces a map

$$g(\tilde{x}_1, \dots, \tilde{x}_n) \;=\; \mathcal{T} \times_1 \tilde{x}_1 \times_2 \tilde{x}_2 \cdots \times_n \tilde{x}_n, \tag{2.4}$$

where $\times_i$ denotes the "tensor-vector" product along mode i. See Kolda and Bader (2009) for background on tensors and mode-wise products. In this work, we use g both as (i) a *surrogate* fit to f locally/globally, and (ii) an exact realization when f is already multilinear.

Tensor Networks (TN) Tensor networks (TNs) represent large tensors by factorizing them into smaller tensors connected via contractions. Each node has physical legs (dangling edges, determining the tensor's dimensions) and internal legs (edges connecting nodes), whose dimensions, called bond dimensions, control the tradeoff between expressiveness and efficiency. See Figure 1. A key complexity measure is the maximal cut rank χ: the largest product of bond dimensions across any cut in the TN. This captures practical cost, as contraction complexity scales with the largest such cut, or equivalently, the maximal rank across bipartitions. Common TN topologies include tensor trains (TT) and balanced binary trees (Oseledets, 2011; Cichocki et al., 2016).

Contraction (forward pass). *Contracting* a TN means summing over all internal indices until only the physical legs remain, analogous to a forward pass through the network. The runtime of such a forward pass is determined by the maximal cut rank χ, and scales polynomially in χ for commonly used TN structures. We refer to each evaluation of $g(x)$ via its TN representation as a single forward contraction.

TN Surrogates We write g for a TN-based surrogate of f and $\mathcal{T}$ for the corresponding tensor. We consider both *fitted surrogate*, where $\mathcal{T}$ is learned to approximate f on a local/global region, and *truncated surrogate*, where $\mathcal{T}$ is a low rank TN approximation of an exact TN for f.

3 METHOD

We first develop the geometric intuition behind our approach (Sec. 3.1), showing how diagonal evaluation reduces exponential queries to polynomial interpolation. We then formalize this via diagonal selectors and prove they yield the exact probe function needed for Shapley computation (Sec. 3.2). Finally, we show how tensor networks make this tractable and how feature maps enhance surrogate fidelity (Sec. 3.3).

3.1 Multilinearity & Efficient Computation

To understand why these ingredients yield an $O(n)$ algorithm, consider the geometry of coalition evaluation. Computing all 2^n coalitions corresponds to querying the function at every vertex of the $[0,1]^n$ hypercube—an exponential task.

Multilinear functions have special structure: they are uniquely determined by their values on any $(n+1)$ points along a diagonal line through the hypercube. Specifically, if we evaluate f at points $(t, t, \ldots, t)$ for $t \in \{0, \frac{1}{n}, \frac{2}{n}, \ldots, 1\}$, the resulting polynomial $p(t) = f(t, \cdots, t)$ encodes all size-aggregated coalition values:

$$p(t) \;=\; f(t,\ldots,t) = \sum_{T\subseteq N} c_T\, t^{|T|} = \sum_{s=0}^{n} t^s \sum_{\substack{T\subseteq N\\ |T|=s}} c_T. \tag{3.1}$$

where the coefficient of t^s aggregates all coalitions of size s. Since Shapley values only depend on these size-aggregated marginals (see Section 2.1), we can recover them by polynomial interpolation, reducing 2^n queries to just $n+1$. For simplicity we present our method when all feature are mapped to 2 dimensional vectors: $\tilde{x}_i = [\phi(x_i), 1]$. The general case is treated in Appendix B.

3.2 Computing Shapley Values via Diagonal Selectors

We now show how diagonal selectors enable polynomial interpolation for Shapley values.

To compute Shapley values for feature i, we need all marginal contributions $v(x, C\cup\{i\}) - v(x, C)$ for $C \subseteq N \setminus \{i\}$. By introducing a parameter t that scales features uniformly, we can aggregate all coalitions of the same size into polynomial coefficients.

Diagonal Selectors To implement this diagonal evaluation strategy while maintaining the TN structure, we introduce *selector matrices*:

$$S_r(t) = \mathrm{Diag}(t, 1) = \begin{bmatrix} t & 0 \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{2\times 2}. \tag{3.2}$$

When applied to a lifted feature, the selector scales the data-dependent coordinate while preserving the bias coordinate:

$$S_r(t)\tilde{x}_r = S_r(t) \begin{bmatrix} x_r \\ 1 \end{bmatrix} = \begin{bmatrix} t\cdot x_r \\ 1 \end{bmatrix}. \tag{3.3}$$

This elegantly implements the diagonal probe: coalitions of size s naturally accumulate with weight t^s. The selector $S_r(t)$ implements a soft coalition membership indicator:

- When $t = 1$: feature r is fully present ($S_r(1)\tilde{x}_r = [x_r, 1]^\top$)
- When $t = 0$: feature r is replaced by its constant channel ($S_r(0)\tilde{x}_r = [0, 1]^\top$)
- For intermediate t: the function interpolates between these extremes

This will allow us to extract marginal contributions through polynomial interpolation.

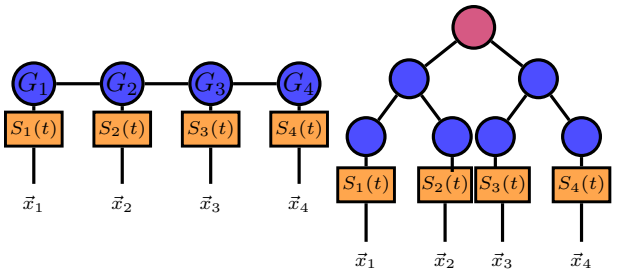

Figure 2: Two TNs with selector matrices $S_k(t)$: tensor train (left) and balanced binary tree (right).

To compute the Shapley value for feature i, we construct a probe function $G_i(t;x)$ that aggregates all marginal contributions $v(x, C\cup\{i\}) - v(x, C)$ for $C \subseteq N \setminus \{i\}$. For a fixed instance x and target feature i, we define:

$$\begin{aligned} G_i(t;x) := g\big(&M(t)S_1(t)\tilde{x}_1, \dots, M(t)S_{i-1}(t)\tilde{x}_{i-1}, S_i(1)\tilde{x}_i, \\ &M(t)S_{i+1}(t)\tilde{x}_{i+1}, \dots, M(t)S_n(t)\tilde{x}_n\big) \\ - g\big(&M(t)S_1(t)\tilde{x}_1, \dots, M(t)S_{i-1}(t)\tilde{x}_{i-1}, S_i(0)\tilde{x}_i, \\ &M(t)S_{i+1}(t)\tilde{x}_{i+1}, \dots, M(t)S_n(t)\tilde{x}_n\big), \end{aligned} \tag{3.4}$$

where $M(t) = \begin{bmatrix} 1 & 0 \\ 0 & t+1 \end{bmatrix}$. This applies selectors $M(t)S_r(t)$ to all features $r \neq i$ (the "complement features"), while evaluating feature i in two configurations:

- Included: $S_i(1)\tilde{x}_i = [x_i, 1]^\top$ (feature present)
- Off (bias): $S_i(0)\tilde{x}_i = [0, 1]^\top$ (feature absent)

The difference isolates the marginal contribution of feature i across all coalitions of complement features.

Due to multilinearity, each coalition $C \subseteq N \setminus \{i\}$ with $|C| = s$ contributes a term scaled by t^s (one factor of t per included feature). Therefore:

$$G_i(t;x) = \sum_{s=0}^{n-1} m_s^{(i)}(x) \cdot t^s, \tag{3.5}$$

where the coefficient $m_s^{(i)}(x) = \sum_{\substack{C\subseteq N\setminus\{i\} \\ |C|=s}} \big[v(x, C\cup\{i\}) - v(x, C)\big]$ aggregates all size-s marginal contributions. See Appendix C for the derivation from equations (3.4) to (3.5).

Relation to the Möbius transform. The induced set function $v(x,\cdot)$ can equivalently be represented by its Möbius coefficients $\mu(S)$ Grabisch et al. (2000), defined by $\mu(S) = \sum_{T\subseteq S}(-1)^{|S|-|T|}\, v(x,T), \quad S \subseteq N$. In this representation, Shapley admits the classical interaction-weighted form

$$\Phi_i^{\mathrm{SV}}(x) = \sum_{S\subseteq N:\, i\in S} \frac{1}{|S|}\,\mu(S), \tag{3.6}$$

showing that $\Phi_i^{\mathrm{SV}}(x)$ averages all interaction terms involving feature i with weight $1/|S|$. See Appendix D for details.

Algorithm 1 TN-SHAP FewEval (single-feature Shapley $\Phi^{\mathrm{SV}}(i;x)$)

Require: Instance x, feature index i, probe nodes $\{t_\ell\}_{\ell=0}^{n-1}$ (all distinct)

Lift inputs once: $\tilde{x}_r \leftarrow [\phi_r(x_r), 1]$ for all $r \in N$
for $\ell = 0$ to $n-1$ **do** ▷ Two TN forwards per node (on/off difference)
  For all $r \neq i$: $\widetilde{x}_r^{(\ell)} \leftarrow S_r(t_\ell)\,\tilde{x}_r$ with $S_r(t) = \mathrm{Diag}(t, 1)$
  $\widetilde{x}_i^{\mathrm{on}} \leftarrow S_i(1)\,\tilde{x}_i, \quad \widetilde{x}_i^{\mathrm{off}} \leftarrow S_i(0)\,\tilde{x}_i$
  $g_1^{(\ell)} \leftarrow g\big(\widetilde{x}_1^{(\ell)}, \dots, \widetilde{x}_{i-1}^{(\ell)}, \widetilde{x}_i^{\mathrm{on}}, \widetilde{x}_{i+1}^{(\ell)}, \dots, \widetilde{x}_n^{(\ell)}\big)$
  $g_0^{(\ell)} \leftarrow g\big(\widetilde{x}_1^{(\ell)}, \dots, \widetilde{x}_{i-1}^{(\ell)}, \widetilde{x}_i^{\mathrm{off}}, \widetilde{x}_{i+1}^{(\ell)}, \dots, \widetilde{x}_n^{(\ell)}\big)$
  $h_\ell \leftarrow g_1^{(\ell)} - g_0^{(\ell)}$ ▷ $h_\ell = G_i(t_\ell; x)$
end for
Build Vandermonde $V \in \mathbb{R}^{n\times n}$ with $V_{\ell+1,r+1} = t_\ell^r$ for $\ell, r = 0, \dots, n-1$
Solve $V\,m^{(i)}(x) = h$ stably (e.g., QR); denote $m^{(i)}(x) = (m_0^{(i)}, \dots, m_{n-1}^{(i)})^\top$
Set Shapley weights $\alpha_s = \dfrac{s!(n-s-1)!}{n!}$ for $s = 0, \dots, n-1$
return $\Phi^{\mathrm{SV}}(i;x) \leftarrow \sum_{s=0}^{n-1} \alpha_s\, m_s^{(i)}(x)$

Polynomial interpolation The coefficients $\{m_s^{(i)}(x)\}_{s=0}^{n-1}$ can simply and efficiently be recovered from n evaluations using polynomial interpolation: `1)`. Evaluate $G_i(t;x)$ at n distinct points: $h_\ell = G_i(t_\ell;x)$ for $\ell = 0, \dots, n-1$; `2)`. Solve the structured linear system (also known as Vandermonde system[1]) $Vm^{(i)} = h$, where $V_{\ell+1,r+1} = t_\ell^r$ for each $0 \le \ell, r < n$; `3)`. Compute the Shapley values as: $\Phi_i^{\mathrm{SV}}(x) = \sum_{s=0}^{n-1} \frac{s!(n-s-1)!}{n!} m_s^{(i)}(x)$. The entire process, summarized in Algorithm 1, requires only n evaluations of the surrogate, exponentially fewer than the 2^n coalition queries needed by enumeration.

3.3 Efficient Implementation with Tensor Networks

The selector-probe method developed above applies directly to tensor network surrogates, where the exponential speedup becomes practically realizable through efficient TN contractions.

[1] A linear system with a Vandermonde matrix, standard in polynomial interpolation (Davis, 1963).

From tensor map to TN probe. Let g be the multilinear TN map in Eq. (2.4) realized by a coefficient tensor $\mathcal{T} \in \mathbb{R}^{2\times\cdots\times 2}$ (binary lifts), so that $g(\tilde{x}_1, \ldots, \tilde{x}_n) = \mathcal{T} \times_1 \tilde{x}_1 \cdots \times_n \tilde{x}_n$. The selector action translates naturally to TN operations: applying $S_r(t) = \mathrm{Diag}(t, 1)$ to feature r means contracting the r-th physical leg of $\mathcal{T}$ with $S_r(t)\tilde{x}_r$ instead of $\tilde{x}_r$. This process is illustrated in Figure 2.

Feature maps for enhanced surrogates While the diagonal selector method works with binary features $[x_i, 1]^\top$, real-world functions often exhibit nonlinear behavior that binary encodings cannot capture. Feature maps let us build more expressive surrogates while preserving the multilinear structure TN-SHAP requires.

We lift each scalar feature x_i to a vector $\tilde{x}_i \in \mathbb{R}^{d_i}$ via a feature map:

$$\phi_i : \mathbb{R} \to \mathbb{R}^{d_i - 1}, \qquad \tilde{x}_i = [\phi_i(x_i), 1]^\top \tag{3.7}$$

It is important to notice that the surrogate remains multilinear in the lifted features $(\tilde{x}_1, \ldots, \tilde{x}_n)$. This means each ϕ_i can be nonlinear in x_i, but the tensor network operates multilinearly on the resulting channels. Importantly, the multilinearity is not necessarily in the input space: it is assumed only after applying the per-feature lift ϕ_i, so the surrogate can still represent highly nonlinear functions of the original inputs.

The bias coordinate provides a constant channel, allowing selectors to implement inclusion by passing $\phi(x_i)$ and exclusion by substituting the fixed value 1, without changing the TN topology. Many feature maps can be used, including:

- Binary (default): $d_i = 2$, $\phi_i(x_i) = x_i$, which is simple and efficient but has limited expressiveness
- Polynomial: $d_i = k + 1$, $\phi_i(x_i) = [x_i, x_i^2, \ldots, x_i^k]^\top$, which can capture polynomial behavior
- Fourier: $d_i = 2k + 1$, with components $\sin(j\omega x_i)$, $\cos(j\omega x_i)$, which can model periodic patterns
- Learned (MLP): $d_i = r$, $\phi_i(x_i) = [\psi_i^{\mathrm{MLP}}(x_i)]^\top$, which is expressive and can adapt to data

Richer maps improve fidelity but can increase the TN size; we observe that even modest lifts (e.g., $d_i = 3$) are often effective.

Computational complexity. While the Shapley value computation through polynomial interpolation remains unchanged ($G_i(t; x)$ in (3.4) yields the polynomial in Eq. (3.5) leading to a Vandermonde system), the TN structure dramatically reduces computational cost. Each evaluation $G_i(t_\ell; x)$ requires only two TN forward passes (on/off toggle), hence the total cost to compute the Shapley value for feature i corresponds to $2n$ TN evaluations and solving a structured $n \times n$ (Vandermonde) linear system. The overall complexity is $O(n \cdot \mathrm{poly}(\chi) + n^2)$, where χ is the bond dimension. This achieves the promised exponential speedup: from $O(2^n)$ coalition queries to $O(n \cdot \mathrm{poly}(\chi))$ operations.

4 THEORETICAL ANALYSIS

Our method provides three key guarantees: exactness on the surrogate, controlled error relative to the original model, and polynomial-time complexity. We state the main results here, with full proofs in Appendix A.

Theorem 4.1 (Approximation Error for Coalitional Indices)**.** *Let f be the original model and g a multilinear surrogate. If the surrogate approximates coalition values uniformly well:*

$$\sup_{C\subseteq N} \big| v_g(x, C) - v_f(x, C) \big| \le \varepsilon,$$

then any size-weighted coalitional index $\Phi_i = \sum_{C\subseteq N\setminus\{i\}} w(|C|, n)\,[v(x, C \cup \{i\}) - v(x, C)]$ satisfies:

$$\big|\Phi_i^g - \Phi_i^f\big| \le 2\varepsilon \sum_{s=0}^{n-1} \big|w(s, n)\big| \binom{n-1}{s}. \tag{4.1}$$

Interpretation: The error in Shapley values (or other coalitional indices) is controlled by the surrogate's worst-case coalition approximation error ε. For Shapley values specifically, this simplifies to $|\phi_i^g - \phi_i^f| \le 2\varepsilon$ since the weights sum to 1.

Proof sketch: Each marginal contribution has error at most 2ε, and there are $\binom{n-1}{s}$ coalitions of size s. The bound follows by triangle inequality. See Appendix A for details.

Theorem 4.2 (Tractability: From Exponential to Polynomial)**.** *Let $\mathcal{T} \in \mathbb{R}^{2\times\cdots\times 2}$ be realized by a TN with maximum bond dimension χ. Any size-weighted coalitional index Φ_i is computable in $O(n\cdot poly(\chi)+n^2)$ time using diagonal selector probes and polynomial interpolation.*

Proof sketch: The diagonal probe $G_i(t)$ from Eq. (3.4) yields polynomial $\sum_s t^s m_s^{(i)}$. Evaluating at n points gives a Vandermonde system; solving recovers all size-aggregated marginals. Each TN forward costs $O(\mathrm{poly}(\chi))$. See Appendix A.

Specialization to Shapley values and interactions. Table 1 summarizes the complexity for common attribution tasks:

- **Shapley values:** The weights $w(s, n) = \frac{s!(n-s-1)!}{n!}$ sum to 1, yielding the tight error bound $|\phi_i^g - \phi_i^f| \le$

Table 1: TN-SHAP complexity by order

Method	TN fwd	Solve	Error
SV ($k=1$)	$2n$	n^2	2ε
Pair ($k=2$)	$4(n-1)$	$(n-1)^2$	4ε
k-SII	$2^k(n-k+1)$	$(n-k+1)^2$	$2^k\varepsilon$

2ε. Computing all n Shapley values requires $2n^2$ TN forwards total.

- **k-way interactions:** Higher-order Shapley interaction indices use inclusion-exclusion over 2^k subsets, requiring $2^k(n-k+1)$ TN forwards. The error grows as $2^k\varepsilon$ due to the signed sum. See Appendix E for details.

Practical runtime. In practice, the $O(n \cdot \text{poly}(\chi))$ complexity yields millisecond runtimes. Using Chebyshev-Gauss nodes[2] and QR factorization ensures numerical robustness, while the tensor network structure keeps computations tractable through efficient contractions.

5 EXPERIMENTS

We evaluate TN-SHAP across three settings: (1) *Synthetic validation* on exactly multilinear functions (runtime-focused, exact-recovery regime); (2) *Practical accuracy on real-world models* (UCI (Yeh, 2007; Dua and Graff, 2019; Tsanas and Xifara, 2012) MLP (Goodfellow et al., 2016) teachers) comparing TN surrogates to sampling baselines; and (3) *Rank ablations & training dynamics* on synthetic teachers to quantify capacity requirements and convergence behavior. Throughout this section, we report three quantities: (i) teacher/model queries used to fit the surrogate, (ii) surrogate fitting time, and (iii) attribution time once the surrogate is fitted.

5.1 Validation on Synthetic Multilinear Functions

We first validate on synthetic multilinear functions where ground truth is exactly computable and the multilinearity assumption holds by construction.

Synthetic setup. We generate synthetic functions from low-rank CP multilinear models with inputs scaled to $[-1, 1]$. For each dimension $d \in \{10, 20, 30, 40, 50\}$, We draw 10 test points and, for each, fit a binary tensor-tree surrogate (rank $\chi = 16$) using only the structured diagonal probes $G_i(t_\ell; x)$ at Chebyshev–Gauss nodes. Since the ground-truth functions are multilinear by construction, once the surrogate is fitted Shapley interactions can in principle be recovered exactly; we therefore focus here on runtime comparisons. Table 2 shows that TN-SHAP achieves orders-of-magnitude faster attribution than KernelSHAP-IQ while maintaining theoretical exactness in this setting. For $d = 40$ and $d = 50$, training requires careful initialization but successful fits yield millisecond-scale attributions.

[2] $t_\ell = \frac{1}{2}(1 + \cos\frac{(2\ell+1)\pi}{2m})$, optimal for polynomial interpolation stability.

Table 2: Runtime on synthetic multilinear functions (per instance).

Dimension	TN-SHAP (ms)	KernelSHAP-IQ (ms)
10	3.5	18.7
20	6.3	115.9
30	11.1	194.9
40	14.2	319.4
50	19.7	447.3

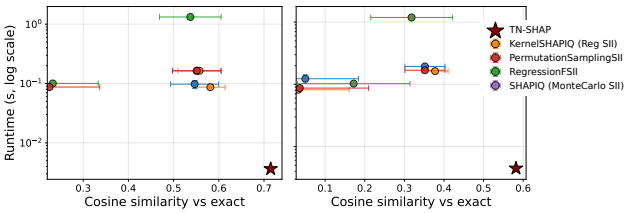

Figure 3: Concrete: runtime ($\log y$) vs. cosine (x); k=2 (left), k=3 (right). Cosine Similarity of RegressionFSII is compared with the Faith-Shap target here. TN-SHAP achieves higher similarity at millisecond scale.

5.2 Practical Accuracy on Real-World Models

Real-world teachers (MLPs on UCI regression tasks) are only approximately multilinear, so exact recovery is not guaranteed. We therefore assess how well TN surrogates recover Shapley values and higher-order interactions in practice, comparing against sampling-based baselines under matched query budgets.

For each center test instance $\mathbf{x}_0$, we select a cohort of 100 nearby points via k-NN in standardized feature space and fit a *single* local binary tensor-tree surrogate (rank χ=16) jointly on this cohort. Before the TN, each coordinate is lifted by a scalar feature map $\phi : \mathbb{R} \to \mathbb{R}$ implemented per feature with a trainable MLP with one hidden layer of 64 neurons and ReLU activation. A constant 1 is appended as a bias term. We evaluate UCI regression tasks (Concrete, Diabetes, Energy) with 3-layer MLP teachers. The local training data around $\mathbf{x}_0$ combine: (i) a modest Gaussian neighbourhood of the cohort, $M \approx 50$–100 samples with $\sigma = 0.1 \times \text{std}(X_{\text{train}})$; and (ii) $2n^2$

Table 3: `Diabetes`: mean±std cosine similarity and MSE($\times 10^{-4}$) w.r.t. exact teacher; runtime in ms. Best times and cosine similarity are bold; values in parentheses include amortized training cost (+65.4ms). SHAPIQ refers to the Regression SII variant (KernelSHAPIQ).

Method	Budget	Time [ms]	Cos ↑	MSE ↓
Order $k=1$				
TN-SHAP	–	**2.8±1.2** (68.2)	**0.994±0.006**	4.8±3.8
SHAPIQ	100	87.5±9.6	0.990±0.009	7.9±4.1
SHAPIQ	1000	635±214	0.980±0.019	15.4±6.9
Order $k=2$				
TN-SHAP	–	**3.9±0.3** (69.3)	**0.637±0.187**	3.9±3.4
SHAPIQ	100	87.5±8.9	0.498±0.204	5.7±2.5
SHAPIQ	1000	633±197	0.633±0.177	3.38±1.21
Order $k=3$				
TN-SHAP	–	**5.1±0.4** (70.5)	0.143±0.395	1.9±2.8
SHAPIQ	100	178±23	0.085±0.152	1.9±1.0
SHAPIQ	1000	738±220	**0.175±0.209**	2.83±0.67

selector-weighted interpolation configurations, i.e., evaluations of $G_i(t_\ell)$ for each $i \in [n]{:=}\{1,\dots,n\}$ at Chebyshev–Gauss nodes t_ℓ. For n=8–10 this adds $\approx$ 160–200 structured evaluations that stabilize the polynomial interpolation. The total teacher-call budget is $M+2n^2 \approx 250$ per center, well below exhaustive enumeration ($2^n \approx 1024$). Once fitted, we reuse the same surrogate to compute Shapley values and higher-order interactions for all 100 points via cheap tensor contractions. We compare against KernelSHAP-IQ (Fumagalli et al., 2024) under budgets of 100–2000 queries and report accuracy against ground-truth SII obtained by exhaustive enumeration; full details appear in Appendix F.

Figure 3 and Table 3 show a clear accuracy–efficiency advantage for TN-SHAP on the Concrete case study and diabetes dataset (averaged over 89 points, entire test set). For k=1, 2, 3 , TN-SHAP attains comparable cosine similarity while operating at millisecond scale, whereas SHAPIQ baselines (KernelSHAP-IQ (Reg SII) (Fumagalli et al., 2024), SHAP-IQ (Monte Carlo) (Muschalik et al., 2024a)) requires hundreds of milliseconds to reach comparable (often lower) accuracy. The log-scaled scatter further highlights a persistent ∼1–2×10 speed gap at similar or better similarity. In short, the local TN surrogate delivers strong practical accuracy with dramatically reduced attribution time for higher-order interactions.

Practical considerations. *Choice of feature map.* Across datasets, a simple *binary feature map* (i.e., using $\tilde{x}_i = [x_i, 1]$) works well for first–order attributions (k=1), but we observe a consistent drop in cosine similarity for higher–order interactions (k=2, 3). To probe whether this is a representational bottleneck rather than an algorithmic one, we replaced the binary map with the simplest learned embedding: an MLP with one hidden layer (64 ReLU units) followed by a linear output *without bias* that produces a *one–dimensional* feature map. This 1D learned map significantly improves the cosine similarity for k=2, 3 across our benchmarks, while leaving k=1 essentially unchanged. A broader study of higher–dimensional learned maps (and their effect on interaction orders $k\geq$2) is deferred to Appendix F.4.

Multilinearity. Neural networks are not globally multilinear, but in practice they are often *locally* close enough. TN-SHAP works best in small neighborhoods around the query point, after simple feature preprocessing that smooths local behavior (our 64-unit ReLU feature maps) and when the local function has low effective interaction order. We measure this by tracking the surrogate R^2 in the local cohort. The method can struggle when activations saturate or the function has sharp regime changes; in those cases, we found that using richer feature maps, increasing the tensor rank (we find χ = 15–20 usually suffices), or shrinking the neighborhood solves the problem. Despite these caveats, across UCI benchmarks we find enough local multilinearity for TN-SHAP to deliver large speedups while keeping attribution quality on par with—or better than—sampling methods.

Table 4: Rank ablation vs. rank-14 tensor tree teacher. First column is student's training R^2 and follows order-wise Shapley interactions vs exact ground-truth R^2.

		R^2 by interaction order		
Rank	Train R^2	k=1	k=2	k=3
2	0.704	0.895	0.853	0.731
3	0.797	0.963	0.903	0.736
4	0.860	0.964	0.962	0.911
5	0.978	0.998	0.998	0.990
6	0.991	0.999	0.997	0.995
8	0.999	1.000	1.000	1.000
10	1.000	1.000	1.000	1.000

5.3 Synthetic Experiments: Understanding Rank Requirements

To understand how tensor rank affects approximation quality, we conduct controlled experiments on synthetic teachers where ground truth is exactly computable.

Setup. We train student TNs of varying rank to approximate teachers with known structure: (1) a TN-tree teacher with rank 16, (2) a low-rank TN-tree with rank 3, and (3) an exactly multilinear function (which would correspond to a full rank TN-tree). This allows

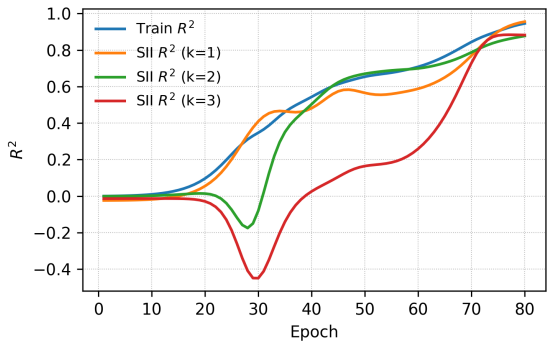

Figure 4: Training dynamics: R^2 vs. epoch for the student fit (Train R^2) and SII (k=1, 2, 3).

us to isolate the effect of rank mismatch.

We present here the results for the rank 16 TN-tree teacher (results for the other teachers are given in Appendix F.2). In this case, Table 4 reveals a critical phase transition. With insufficient rank ($r \leq 3$), the student captures first-order effects reasonably well (SII $R^2 = 0.96$ at $r = 3$) but underfits higher-order interactions (order-3 $R^2 = 0.74$). At $r = 4$, order-3 accuracy jumps to $R^2 = 0.91$, and by $r = 5$–6, all orders achieve $R^2 > 0.99$. Beyond this threshold, additional rank provides diminishing returns.

Figure 4 provides insight into the learning dynamics. Lower-order interactions stabilize early (order-1 plateaus by epoch 200), while higher orders continue improving. This hierarchical pattern suggests practical training strategies: monitor lower-order convergence as an early stopping criterion, and increase rank if higher orders plateau below acceptable accuracy. TN-SHAP delivers 20–100× speedups with superior accuracy on real data and exact recovery on synthetic multilinear functions. Rank $r = 5$–6 suffices for accurate recovery of all interaction orders, with hierarchical training dynamics providing natural convergence indicators.

5.4 High-dimensional Top-k Recovery Experiment

To test whether TN-SHAP remains accurate beyond the low-dimensional settings, we test on a higher-dimensional synthetic data following the protocol in (Mohammadi et al., 2025a). We evaluate three function families: degree-5 polynomial (POLY5), degree-10 polynomial (POLY10), and squared exponential (SQEXP), at dimensions $D \in \{50, 100\}$ using a tensor-train (TT/MPS) surrogate repeated over 100 random seeds. In each setting, we use a linear-cost query budget, corresponding to 500 teacher queries for $D = 50$ and 1000 for $D = 100$. We report both

Task	D	Top-K ↑	R^2 ↑	Train	Eval
poly5	50	0.84 ± 0.14	0.95 ± 0.05	6.47	2.88
poly5	100	0.90 ± 0.06	0.96 ± 0.06	14.73	3.44
poly10	50	0.64 ± 0.20	0.90 ± 0.09	7.08	3.12
poly10	100	0.76 ± 0.15	0.94 ± 0.07	14.88	3.50
sqexp	50	0.85 ± 0.08	0.98 ± 0.00	8.51	3.12
sqexp	100	0.98 ± 0.02	1.00 ± 0.00	25.72	3.15

Table 5: **High-dimensional synthetic evaluation under a linear-cost query budget.** Following the high-dimensional evaluation protocol of Mohammadi et al. (2025a). In each task, the target function depends on only 10% of the input dimensions, and we report Top-K significant-feature recovery accuracy, surrogate fidelity (R^2), surrogate training time, and attribution time (Eval).

Top-k significant-feature recovery accuracy and surrogate fidelity (R^2), together with surrogate training time and post-fit attribution time. As shown in Table 5, TN-SHAP remains stable and accurate in these higher-dimensional regimes: surrogate fidelity stays high across all settings, and Top-k recovery is strong even under the fixed query budget.

6 RELATED WORK

SHAP computation methods SHAP computation spans three categories: model-specific exact methods (TreeSHAP(Lundberg et al., 2018), DeepSHAP(Shrikumar et al., 2017)), sampling-based approximations (KernelSHAP(Lundberg and Lee, 2017), SAGE(Covert et al., 2020)), and amortized approaches (FastSHAP(Jethani et al., 2022)). TN-SHAP computes exact k-th order Shapley interactions using a small, structured set of evaluations with built-in reliability diagnostics via tensor decomposition quality, unique among current methods.

Shapley values and GAMs. A complementary line of work ((Bordt and von Luxburg, 2023),(Enouen and Liu, 2025), (Park et al., 2025)) shows that Shapley-based explanations correspond to generalized additive models (GAMs) with interaction terms. Bordt and von Luxburg (2023) prove that n-Shapley values, Shapley–Taylor interactions (Sundararajan et al., 2020) and Faith-Shap(Tsai et al., 2023a) recover GAMs with interactions up to order n; setting n=1 corresponds to the classical Shapley value, which recovers an additive decomposition without interactions. TN-SHAP bridges local Shapley explanations and global GAMs: fitting a order-k surrogate yields exact k-SII coefficients that match the corresponding GAM terms, unifying local attributions with global

additive structure.

Tensor networks and low-rank structure in ML. Prior TN methods store precomputed indices—Sobol values (Ballester-Ripoll et al., 2019), cooperative game solutions (Ballester-Ripoll, 2022), or sparse interaction tensors (Ryzhakov and Oseledets, 2022). While prior TN methods (Ballester-Ripoll et al., 2019; Ballester-Ripoll, 2022) use tensor decompositions to store precomputed Shapley values, we employ TNs as *computational architectures*: feature maps lift inputs to a multilinear space where Shapley queries become direct tensor contractions, enabling exact k-SII computation in $O(2^k(n-k+1)\cdot \mathrm{poly}(\chi))$ time without sampling or enumeration. Constructive schemes (Ryzhakov and Oseledets, 2022) provide sparse cores for interaction tensors, but still focus on representation rather than direct computation.

Local surrogates and faithfulness Local surrogate methods such as LIME (Ribeiro et al., 2016) and Anchors (Ribeiro et al., 2018) explain individual predictions by fitting interpretable models around specific instances. However, recent work has exposed critical faithfulness issues with these approaches (Adebayo et al., 2018; Slack et al., 2020; Laugel et al., 2019; Li et al., 2023; Hwang et al., 2025), demonstrating that local surrogates can be manipulated or may fail to capture the true decision boundary of the underlying model. TN-SHAP provide structural guarantees through rank bounds while computing exact SHAP values, addressing the faithfulness issues of unconstrained local methods like LIME.

Surrogate and kernel based Shapley methods. TN-SHAP is related to methods that approximate Shapley by exploiting structure in the model. RKHS-SHAP derives Shapley values for kernel machines using kernel mean embeddings, avoiding Monte Carlo estimation at attribution time (Chau et al., 2022). Product-kernel and fANOVA Gaussian-process methods similarly obtain exact or polynomial-time Shapley computation by leveraging strong kernel-factorization assumptions (Mohammadi et al., 2025a,b). FOURIER-SHAP represents binary-input predictors through a sparse Fourier expansion and then computes SHAP in closed form (Gorji et al., 2024). Regression-adjusted Monte Carlo methods combine sample reuse with a learned regression model to reduce variance (Witter et al., 2025). PROXYSPEX fits structured surrogates to recover a sparse set of influential interactions efficiently (Butler et al., 2025). These methods are complementary. TN-SHAP targets functions that admit a low-rank multilinear tensor-network surrogate in a learned feature space.

Limitations and future work The tensor network fit of TN-SHAP depends on the chosen tensor-network topology and, for structured factorizations such as tensor trains (TT/MPS) and tensor trees (HT/TTN), on the ordering or grouping of features. In this work, we use simple random feature orderings / groupings in our experiments, and found that this already works well empirically. Still, we do not claim that random ordering is optimal. Better strategies are possible and could further improve both approximation quality and computational efficiency, especially in higher dimensions. There is room for further speedups and degree-wise regularization (e.g., (Convy and Whaley, 2022)). A complementary direction is to relate tensor–network *rank* to impossibility and robustness results in explainability (Günther et al., 2025), clarifying the boundary between tractable and provably intractable attribution regimes. Another promising direction for future work is to study whether structure-exploiting surrogates such as TN-SHAP could be useful in settings such as safety research and mechanistic interpretability, where high variance explanations are undesirable. More broadly, this viewpoint aligns with leveraging information structure instead of random sampling, as discussed in (Neyman, 2024, 2025).

7 CONCLUSION

We presented TN-SHAP, a tensor–network approach that turns Shapley values and Shapley-style interactions from an exponential enumeration problem into a small number of structured evaluations. By combining selector-weighted contractions with a single Vandermonde solve, TN-SHAP reduces the cost from $O(2^n)$ coalitions to $O\big(n\,\mathrm{poly}(\chi)\big)$ forward passes, is exact on a multilinear surrogate, and comes with tractability and approximation guarantees. On UCI regressors, TN-SHAP achieves millisecond-scale per-instance attributions and 25–1000$\times$ wall-clock speedups over sampling baselines at comparable accuracy, while amortizing training across local cohorts.

8 ACKNOWLEGMENTS

Guillaume Rabusseau acknowledges the support of the CIFAR AI Chair program. This work was supported by the Mitacs Globalink Research Award program and RIKEN. Chao Li was supported by the JSPS KAKENHI Grant Numbers JP24K03005. Farzaneh Heidari thanks Tensor Learning Team at RIKEN AIP for their insightful discussions and for welcoming her during her research stay at RIKEN AIP.

References

Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., and Kim, B. (2018). Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems*, pages 9505–9515.

Ballester-Ripoll, R. (2022). Tensor approximation of cooperative games and their semivalues. *Int. J. Approx. Reasoning*, 142(C):94–108.

Ballester-Ripoll, R., Paredes, E. G., and Pajarola, R. (2019). Sobol tensor trains for global sensitivity analysis. *Reliability Engineering & System Safety*, 183:311–322.

Björck, Å. and Pereyra, V. (1970). Solution of vandermonde systems of equations. *Mathematics of Computation*, 24(112):893–903.

Bordt, S. and von Luxburg, U. (2023). From shapley values to generalized additive models and back. In *International Conference on Artificial Intelligence and Statistics*, pages 709–745. PMLR.

Butler, L., Agarwal, A., Kang, J. S., Erginbas, Y. E., Yu, B., and Ramchandran, K. (2025). Proxyspex: Inference-efficient interpretability via sparse feature interactions in llms. *arXiv preprint arXiv:2505.17495*.

Chau, S. L., Hu, R., Gonzalez, J., and Sejdinovic, D. (2022). Rkhs-shap: Shapley values for kernel methods. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2022)*.

Cichocki, A., Lee, N., Oseledets, I., Phan, A.-H., Zhao, Q., Mandic, D. P., et al. (2016). Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions. *Foundations and Trends® in Machine Learning*, 9(4-5):249–429.

Convy, I. and Whaley, K. B. (2022). Interaction decompositions for tensor network regression. *Machine Learning: Science and Technology*, 3(4):045027.

Covert, I., Lundberg, S. M., and Lee, S.-I. (2020). Understanding global feature contributions with additive importance measures. *Advances in neural information processing systems*, 33:17212–17223.

Davis, P. (1963). *Interpolation and Approximation*. Blaisdell book in pure and applied sciences. Introductions to higher mathematics. Dover Publications.

Dua, D. and Graff, C. (2019). Diabetes. UCI Machine Learning Repository. University of California, Irvine.

Enouen, J. and Liu, Y. (2025). Instashap: Interpretable additive models explain shapley values instantly. *arXiv preprint arXiv:2502.14177*.

Fumagalli, F., Muschalik, M., Kolpaczki, P., Hüllermeier, E., and Hammer, B. (2024). Kernelshap-iq: Weighted least-square optimization for shapley interactions. *arXiv preprint arXiv:2405.10852*.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT Press.

Gorji, A., Amrollahi, A., and Krause, A. (2024). Shap values via sparse fourier representation. *arXiv preprint arXiv:2410.06300*.

Grabisch, M., Marichal, J.-L., and Roubens, M. (2000). Equivalent representations of set functions. *Mathematics of Operations Research*, 25(2):157–178.

Grabisch, M. and Roubens, M. (1999a). An axiomatic approach to the concept of interaction among players in cooperative games. *International Journal of game theory*, 28(4):547–565.

Grabisch, M. and Roubens, M. (1999b). An axiomatic approach to the concept of interaction among players in cooperative games. *International Journal of Game Theory*, 28(4):547–565.

Grasedyck, L. (2010). Hierarchical singular value decomposition of tensors. *SIAM Journal on Matrix Analysis and Applications*, 31(4):2029–2054.

Günther, E., Szabados, B., Bhattacharjee, R., Bordt, S., and von Luxburg, U. (2025). Informative post-hoc explanations only exist for simple functions. *arXiv preprint arXiv:2508.11441*.

Horn, R. A. and Johnson, C. R. (2012). *Matrix Analysis*. Cambridge University Press, 2 edition.

Hwang, H., Bell, A., Fonseca, J., Pliatsika, V., Stoyanovich, J., and Whang, S. E. (2025). Shap-based explanations are sensitive to feature representation. In *Proceedings of the 2025 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '25, page 1588–1601, New York, NY, USA. Association for Computing Machinery.

Jethani, N., Sudarshan, M., Covert, I., Lee, S.-I., and Ranganath, R. (2022). Fastshap: Real-time shapley value estimation. *ICLR 2022*.

Kolda, T. G. and Bader, B. W. (2009). Tensor decompositions and applications. *SIAM Review*, 51(3):455–500.

Laugel, T., Lesot, M.-J., Marsala, C., Renard, X., and Detyniecki, M. (2019). The dangers of post-hoc interpretability: Unjustified counterfactual explanations. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 2801–2807.

Levine, Y., Sharir, O., Cohen, N., and Shashua, A. (2019). Quantum entanglement in deep learning architectures. *Physical review letters*, 122(6):065301.

Li, X., Du, M., Chen, J., Chai, Y., Lakkaraju, H., and Xiong, H. (2023). $\mathcal{M}^4$: A unified xai benchmark for faithfulness evaluation of feature attribution methods across metrics, modalities and models. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S., editors, *Advances in Neural Information Processing Systems*, volume 36, pages 1630–1643. Curran Associates, Inc.

Lundberg, S. M., Erion, G. G., and Lee, S.-I. (2018). Consistent individualized feature attribution for tree ensembles. *arXiv preprint arXiv:1802.03888*.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, volume 30, pages 4765–4774.

Mohammadi, M., Chau, S. L., and Muandet, K. (2025a). Computing exact shapley values in polynomial time for product-kernel methods. *arXiv preprint arXiv:2505.16516*.

Mohammadi, M., Muandet, K., Tiddi, I., Ten Teije, A., and Chau, S. L. (2025b). Exact shapley attributions in quadratic-time for fanova gaussian processes. *arXiv preprint arXiv:2508.14499*.

Muschalik, M., Baniecki, H., Fumagalli, F., Kolpaczki, P., Hammer, B., and Hüllermeier, E. (2024a). shapiq: Shapley interactions for machine learning. *Advances in Neural Information Processing Systems*, 37:130324–130357.

Muschalik, M., Baniecki, H., Fumagalli, F., Kolpaczki, P., Hammer, B., and Hüllermeier, E. (2024b). shapiq: Shapley interactions for machine learning. In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C., editors, *Advances in Neural Information Processing Systems*, volume 37, pages 130324–130357. Curran Associates, Inc.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pages 807–814.

Neyman, E. (2024). *Algorithmic bayesian epistemology*. Columbia University.

Neyman, E. (2025). Competing with sampling.

Oseledets, I. V. (2011). Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317.

Owen, G. (1972). Multi-linear extensions of games. *Management Science*, 18(5):P64–P79.

Park, S., Kong, I., Choi, Y., Park, C., and Kim, Y. (2025). Tensor product neural networks for functional anova model. *arXiv preprint arXiv:2502.15215*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, É. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2018). Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Roth, A. E., editor (1988). *The Shapley Value: Essays in Honor of Lloyd S. Shapley*. Studies in Game Theory. Cambridge University Press, Cambridge, UK and New York, NY, USA.

Rudin, W. (1976). *Principles of Mathematical Analysis*. McGraw–Hill, 3 edition.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Ryzhakov, G. and Oseledets, I. (2022). Constructive tt-representation of the tensors given as index interaction functions with applications. *arXiv preprint arXiv:2206.03832*.

Shapley, L. S. (1953). A value for n-person games. In Kuhn, H. W. and Tucker, A. W., editors, *Contributions to the Theory of Games II*, pages 307–317. Princeton University Press, Princeton.

Shrikumar, A., Greenside, P., and Kundaje, A. (2017). Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMlR.

Slack, D., Hilgard, S., Jia, E., Singh, S., and Lakkaraju, H. (2020). Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 180–186.

Sundararajan, M., Dhamdhere, K., and Agarwal, A. (2020). The shapley taylor interaction index. In

International conference on machine learning, pages 9259–9268. PMLR.

Sundararajan, M. and Najmi, A. (2020). The many Shapley values for model explanation. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 9269–9278. PMLR.

Trefethen, L. N. (2013). *Approximation Theory and Approximation Practice*. SIAM, Philadelphia, PA.

Tsai, C.-P., Yeh, C.-K., and Ravikumar, P. (2023a). Faith-shap: The faithful shapley interaction index. *Journal of Machine Learning Research*, 24(94):1–42.

Tsai, C.-P., Yeh, C.-K., and Ravikumar, P. (2023b). Faith-shap: The faithful shapley interaction index. *Journal of Machine Learning Research*, 24(94):1–42.

Tsanas, A. and Xifara, A. (2012). Energy efficiency. UCI Machine Learning Repository. University of California, Irvine.

Witter, R. T., Liu, Y., and Musco, C. (2025). Regression-adjusted monte carlo estimators for shapley values and probabilistic values. *arXiv preprint arXiv:2506.11849*.

Yeh, I.-C. (2007). Concrete compressive strength. UCI Machine Learning Repository. University of California, Irvine.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes/No/Not Applicable] Yes
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes/No/Not Applicable] Yes
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes/No/Not Applicable] Yes

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes/No/Not Applicable] Yes
   (b) Complete proofs of all theoretical results. [Yes/No/Not Applicable] Yes
   (c) Clear explanations of any assumptions. [Yes/No/Not Applicable] Yes

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes/No/Not Applicable] Yes
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes/No/Not Applicable] Yes
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes/No/Not Applicable] Yes
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes/No/Not Applicable] Yes

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes/No/Not Applicable] Yes
   (b) The license information of the assets, if applicable. [Yes/No/Not Applicable] Not Applicable
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes/No/Not Applicable] Yes
   (d) Information about consent from data providers/curators. [Yes/No/Not Applicable] Not Applicable
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Yes/No/Not Applicable] Not Applicable

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Yes/No/Not Applicable] Not Applicable
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Yes/No/Not Applicable] Not Applicable
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Yes/No/Not Applicable] Not Applicable

Supplementary Materials

This supplement provides theoretical foundations, algorithmic details, and extended experimental results for TN-SHAP. We begin with complete proofs (Section A), technical details on feature maps and selectors (Section B) and algorithms for k-way interactions (Section E); then conclude with comprehensive experimental protocols and additional results (Section F).

A Proofs of Main Theorems

We provide complete proofs for the tractability and approximation theorems stated in Section 3 of the main paper. These results establish that TN-SHAP achieves exponential speedup from $O(2^n)$ to $O(n \cdot \text{poly}(\chi) + n^2)$ while maintaining controlled approximation error. The proofs rely on the multilinear structure of the surrogate and the size-aggregation property of Shapley weights, showing how diagonal selectors enable efficient extraction of all coalition values through polynomial interpolation.

Proof of Theorem 4.1. Fix a center point x. For any model h, we abbreviate the coalition value function as $v_h(C) := v_h(x, C)$ and define the local Shapley value of feature i as

$$\Phi_i^h = \sum_{C \subseteq N \setminus \{i\}} w(|C|, n)\big(v_h(C \cup \{i\}) - v_h(C)\big),$$

where $w(|C|, n) = \frac{|C|!\,(n-|C|-1)!}{n!}$ are the standard Shapley weights. Let v_f and v_g denote the coalition value functions for the original target function f and for its multilinear (TN) surrogate g, respectively. Assume that the surrogate uniformly approximates the true coalition values within ε, i.e.,

$$\sup_{C \subseteq N} \big|v_g(C) - v_f(C)\big| \le \varepsilon.$$

For a fixed feature $i \in N$, we have

$$\begin{aligned}
\big|\Phi_i^g - \Phi_i^f\big| &= \Big| \sum_{C \subseteq N \setminus \{i\}} w(|C|, n) \Big([v_g(C \cup \{i\}) - v_f(C \cup \{i\})] - [v_g(C) - v_f(C)] \Big) \Big| \\
&\le \sum_{C \subseteq N \setminus \{i\}} |w(|C|, n)| \Big(|v_g(C \cup \{i\}) - v_f(C \cup \{i\})| + |v_g(C) - v_f(C)| \Big) \\
&\le 2\varepsilon \sum_{C \subseteq N \setminus \{i\}} |w(|C|, n)|.
\end{aligned}$$

The inequality follows from the triangle inequality (or equivalently, from the Cauchy–Schwarz inequality (see, e.g., Rudin, 1976; Horn and Johnson, 2012)). Grouping subsets by their size $s = |C|$ (there are $\binom{n-1}{s}$ such coalitions) yields

$$\big|\Phi_i^g - \Phi_i^f\big| \;\le\; 2\varepsilon \sum_{s=0}^{n-1} |w(s, n)| \binom{n-1}{s},$$

which establishes the desired bound. □

Proof of Theorem 4.2. Fix a center point $x \in \mathbb{R}^n$. For each feature j, define the lifted input $\tilde{x}_j = [x_j,\, 1]^\top \in \mathbb{R}^2$ and the diagonal selector

$$S_j(t) = \text{Diag}(t,\, 1) \in \mathbb{R}^{2\times 2}, \qquad t \in [0, 1].$$

Setting $t = 1$ *includes* feature j (uses $[x_j, 1]^\top$), while $t = 0$ *excludes* it by zeroing the data channel and retaining the bias, $[0, 1]^\top$. Thus, for any coalition $C \subseteq N$, the coalition value $v_g(x, C)$ is obtained by using $S_j(1)\tilde{x}_j$ for $j \in C$ and $S_j(0)\tilde{x}_j$ for $j \notin C$.

Let g be the multilinear TN surrogate realizing $\mathcal{T}$ with maximum bond dimension χ; one forward evaluation of g costs $O(\text{poly}(\chi))$ (Levine et al., 2019; Oseledets, 2011; Grasedyck, 2010). Write $v_g(C) := v_g(x, C)$.

For the Inclusion–exclusion probe, fix $i \in N$. Define

$$G_i(t) := g\big(\{S_j(t)\tilde{x}_j\}_{j\neq i},\, S_i(1)\tilde{x}_i\big) - g\big(\{S_j(t)\tilde{x}_j\}_{j\neq i},\, S_i(0)\tilde{x}_i\big).$$

By multilinearity, $G_i(t)$ is a univariate polynomial of degree at most $n-1$ *(here we focus on the size-*1 *case, i.e., the Shapley value for a single index* i*)*:

$$G_i(t) = \sum_{s=0}^{n-1} m_s^{(i)}\, t^s, \qquad m_s^{(i)} = \sum_{\substack{C\subseteq N\setminus\{i\}\\ |C|=s}} \big[v_g(C\cup\{i\}) - v_g(C)\big].$$

In this formulation, each factor $S_j(t)$ contributes a factor t exactly when j is *included* in C; the two evaluations with $S_i(1)$ and $S_i(0)$ then implement inclusion/exclusion on i, producing the Shapley marginal $v_g(C\cup\{i\})-v_g(C)$. By multilinearity, $G_i(t)$ is a univariate polynomial of degree at most $n-1$ (size-1 Shapley for index i); to recover its coefficients, we interpolate G_i at n distinct points $\{t_\ell\}_{\ell=0}^{n-1} \subset (0,1)$ to form the Vandermonde system

$$V\, m^{(i)} = q, \qquad V_{\ell+1,s+1} = t_\ell^s,\ m^{(i)} = (m_0^{(i)},\dots,m_{n-1}^{(i)})^\top,\ q_\ell = G_i(t_\ell).$$

Each q_ℓ requires two forward passes (inclusion and exclusion of i) each costing $O\big(\mathrm{poly}(\chi)\big)$. Since G_i is evaluated at n interpolation points, the total probing cost is $\boldsymbol{O(n\cdot \mathrm{poly}(\chi))}$. The coefficient vector $m^{(i)}$ is then obtained by solving $V\, m^{(i)} = q$ in $\boldsymbol{O(n^2)}$ time using a stable Vandermonde solver (Björck and Pereyra, 1970).

The resulting coefficients $\{m_s^{(i)}\}_{s=0}^{n-1}$ represent the aggregated marginal contributions of feature i grouped by coalition size, so the Shapley index follows directly as

$$\Phi_i = \sum_{s=0}^{n-1} w(s,n)\, m_s^{(i)}.$$

Forming this weighted combination requires only $O(n)$ additional operations, which is negligible compared to the probing and interpolation steps. The total cost is therefore $\boldsymbol{O(n\cdot \mathrm{poly}(\chi)+n^2)}$, establishing the polynomial-time tractability claim. □

B Feature Maps and Selectors

The feature map design is crucial for balancing expressiveness with computational tractability. Here we formalize the general d_i-dimensional lifting scheme sketched in the main paper, showing how various feature maps with arbitrary output dimensions can enhance surrogate fidelity (empirically in Section F.4) while preserving the multilinear structure required for exact Shapley computation. We also detail the thin diagonal selector construction and its role in maintaining constant TN topology during coalition probing.

B.1 General Feature Map Construction

We lift each scalar x_i to $\tilde{x}_i = [\phi_i(x_i),\, 1]^\top \in \mathbb{R}^{d_i}$, where $\phi_i : \mathbb{R} \to \mathbb{R}^{d_i-1}$ is a feature map providing a low-dimensional nonlinear embedding of the input. The last entry acts as a bias channel to preserve the affine closure of multilinear terms.

B.2 Diagonal Selector Properties

To toggle coalitions without altering TN topology, we use *thin diagonal selectors*

$$S_i(t) = \mathrm{Diag}(t\, I_{d_i-1}, 1) \in \mathbb{R}^{d_i\times d_i},$$

which scale only the data-dependent channels and keep the bias channel intact:

$$\tilde{x}_i = \begin{bmatrix}\phi_i(x_i)\\ 1\end{bmatrix} \in \mathbb{R}^{d_i}, \qquad S_i(t) = \mathrm{Diag}\big(t\, I_{d_i-1},\, 1\big) \in \mathbb{R}^{d_i\times d_i}.$$

$$S_i(1)\,\tilde{x}_i = \begin{bmatrix}\phi_i(x_i)\\ 1\end{bmatrix}, \qquad S_i(0)\,\tilde{x}_i = \begin{bmatrix}\mathbf{0}_{d_i-1}\\ 1\end{bmatrix}.$$

This design ensures that the surrogate remains multilinear in the lifted inputs and that inclusion/exclusion operations can be expressed by simple elementwise scaling rather than structural rewiring.

B.3 Complexity and Stability Considerations

Let χ denote the maximal bond dimension and $d_{\max} = \max_i d_i$. The forward cost per contraction scales as $O(\mathrm{poly}(\chi)\, d_{\max})$. We observe numerically that increasing d_i above 3–4 offers diminishing returns for $k \le 3$ interactions (see Section F.4).

C Tensor contraction view of TN-SHAP

The following lemma formally shows the equality between equations (3.5) and (3.4).

Lemma C.1. *For a fixed instance x and target feature i, define the function $G_i(t;x)$ by*

$$\begin{aligned} G_i(t;x) := g\big(&M(t)S_1(t)\tilde{x}_1, \dots, M(t)S_{i-1}(t)\tilde{x}_{i-1}, S_i(1)\tilde{x}_i,\\ &M(t)S_{i+1}(t)\tilde{x}_{i+1}, \dots, M(t)S_n(t)\tilde{x}_n\big)\\ - g\big(&M(t)S_1(t)\tilde{x}_1, \dots, M(t)S_{i-1}(t)\tilde{x}_{i-1}, S_i(0)\tilde{x}_i,\\ &M(t)S_{i+1}(t)\tilde{x}_{i+1}, \dots, M(t)S_n(t)\tilde{x}_n\big), \end{aligned} \tag{3.4}$$

where $M(t) = \begin{bmatrix}1 & 0\\ 0 & t+1\end{bmatrix}$ *and* $S(t) = \begin{bmatrix}t & 0\\ 0 & 1\end{bmatrix}$.

$G_i(t;x)$ is a polynomial in t of degree at most $n-1$ given by

$$G_i(t;x) = \sum_{s=0}^{n-1} m_s^{(i)}(x)\, t^s, \tag{3.5}$$

where

$$m_s^{(i)}(x) = \sum_{\substack{C \subseteq N\setminus\{i\}\\ |C|=s}} \big(v(x, C\cup\{i\}) - v(x,C)\big).$$

Proof. Using the tensor form of the multilinear model,

$$g(\tilde{z}_1, \dots, \tilde{z}_n) = T \times_1 \tilde{z}_1 \times_2 \tilde{z}_2 \cdots \times_n \tilde{z}_n,$$

Equation (3.4) becomes

$$G_i(t;x) = T \times_{j\neq i} M(t)S_j(t)\tilde{x}_j \times_i S_i(1)\tilde{x}_i - T \times_{j\neq i} M(t)S_j(t)\tilde{x}_j \times_i S_i(0)\tilde{x}_i.$$

Since

$$S_i(1)\tilde{x}_i = \begin{pmatrix}x_i\\ 1\end{pmatrix}, \qquad S_i(0)\tilde{x}_i = \begin{pmatrix}0\\ 1\end{pmatrix},$$

and for every $j \neq i$,

$$M(t)S_j(t)\tilde{x}_j = \begin{pmatrix}tx_j\\ t+1\end{pmatrix} = t\begin{pmatrix}x_j\\ 1\end{pmatrix} + \begin{pmatrix}0\\ 1\end{pmatrix},$$

we obtain

$$G_i(t;x) = T \times_{j\neq i} \left(t\begin{pmatrix}x_j\\ 1\end{pmatrix} + \begin{pmatrix}0\\ 1\end{pmatrix}\right) \times_i \begin{pmatrix}x_i\\ 1\end{pmatrix} - T \times_{j\neq i} \left(t\begin{pmatrix}x_j\\ 1\end{pmatrix} + \begin{pmatrix}0\\ 1\end{pmatrix}\right) \times_i \begin{pmatrix}0\\ 1\end{pmatrix}.$$

By multilinearity, expanding the sum over the modes $j \neq i$ gives

$$G_i(t;x) = \sum_{C\subseteq N\setminus\{i\}} \left[T \times_{j\in C} \begin{pmatrix}x_j\\ 1\end{pmatrix} \times_{j\notin C,\, j\neq i} \begin{pmatrix}0\\ 1\end{pmatrix} \times_i \begin{pmatrix}x_i\\ 1\end{pmatrix} - T \times_{j\in C} \begin{pmatrix}x_j\\ 1\end{pmatrix} \times_{j\notin C,\, j\neq i} \begin{pmatrix}0\\ 1\end{pmatrix} \times_i \begin{pmatrix}0\\ 1\end{pmatrix}\right] t^{|C|}.$$

The bracketed term is exactly

$$v(x, C \cup \{i\}) - v(x, C).$$

Grouping terms by $s = |C|$ yields

$$G_i(t; x) = \sum_{s=0}^{n-1} \left(\sum_{\substack{C \subseteq N \setminus \{i\} \\ |C|=s}} \big(v(x, C \cup \{i\}) - v(x, C)\big) \right) t^s = \sum_{s=0}^{n-1} m_s^{(i)}(x)\, t^s,$$

as claimed. □

D Connection to the Möbius transform.

The same set function $v(x, \cdot) : 2^N \to \mathbb{R}$ also admits an equivalent expansion in terms of its Möbius coefficients (or interaction coefficients). Define the Möbius transform of v by

$$\mu(S) := \sum_{T \subseteq S} (-1)^{|S|-|T|} v(x, T), \qquad S \subseteq N, \tag{D.1}$$

so that v can be recovered as

$$v(x, C) = \sum_{S \subseteq C} \mu(S).$$

For any $C \subseteq N \setminus \{i\}$, the marginal contribution of feature i is

$$\Delta_i v(C) := v(x, C \cup \{i\}) - v(x, C) = \sum_{S \subseteq C} \mu(S \cup \{i\}), \tag{D.2}$$

showing that only interaction terms involving feature i contribute to $\Delta_i v(C)$.

Combining (D.2) with the polynomial expansion of $G_i(t; x)$, we obtain

$$G_i(t; x) = \sum_{C \subseteq N \setminus \{i\}} \Delta_i v(C)\, t^{|C|}, \tag{D.3}$$

and exchanging the order of summation gives

$$G_i(t; x) = \sum_{S \subseteq N \setminus \{i\}} \mu(S \cup \{i\}) \sum_{C \supseteq S} t^{|C|}.$$

Thus, $G_i(t; x)$ may equivalently be viewed as a polynomial whose coefficients aggregate Möbius interaction terms involving feature i, grouped by subset size. Finally, summing up the marginal contribution over subset sizes recovers the standard Möbius-form expression for the Shapley value:

$$\Phi_i^{\mathrm{SV}}(x) = \sum_{\substack{S \subseteq N \\ i \in S}} \frac{1}{|S|}\, \mu(S). \tag{D.4}$$

Therefore, the Shapley value averages all interaction terms containing feature i, with weight inversely proportional to their order.

E Order-k Shapley Interactions

We proceed in three steps. First, we present explicit algorithms for computing (i) pairwise ($k = 2$) Shapley interaction indices §E.1and (ii) general order-k indices (Grabisch and Roubens, 1999a; Sundararajan and Najmi, 2020) using diagonal selector probes and polynomial interpolation (§E.2). Second, we analyze the computational complexity of the general case: a direct inclusion–exclusion evaluation of $\Delta_C v_g(\cdot)$ requires 2^k forward passes per

probe node, resulting in an overall cost of $O(2^k n\,\mathrm{poly}(\chi)+n^2)$ (§E.2). Finally, we exploit multilinearity to derive the *signed–toggle identity*

$$\Delta_C v_g(x) \;=\; g\Big(\{S_j(t)\tilde{x}_j\}_{j\in N\setminus C},\, \{(S_i(1)-S_i(0))\tilde{x}_i\}_{i\in C}\Big),$$

which collapses the 2^k inclusion–exclusion terms into a single evaluation per probe. This reduces the overall complexity to $O(n\,\mathrm{poly}(\chi)+n^2)$ while preserving the same interpolation-based recovery of size-aggregated coefficients (§E.3).

E.1 Pairwise Interactions

To build intuition for our general framework, we begin with the simplest nontrivial case, $|C|=2$, where $C=\{i,j\}$. Pairwise interactions capture how the joint effect of features i and j differs from the sum of their individual effects (Grabisch and Roubens, 1999b), and they illustrate the key components of our method, inclusion–exclusion, multilinearity, and polynomial interpolation, before extending to higher orders.

Inclusion–exclusion probe. For a fixed pair $\{i,j\}$, the inclusion–exclusion principle expresses the second-order marginal contribution as a signed sum over all four inclusion patterns of the two features. For example, for $i=1$ and $j=2$

$$\begin{aligned} Q_{\{i,j\}}(t;x) =& g(S_i(1)\tilde{x}_i, S_j(1)\tilde{x}_j, S_3(t),\dots,S_n(t)) - g(S_i(1)\tilde{x}_i, S_j(0)\tilde{x}_j, S_3(t),\dots,S_n(t)) \\ & - g(S_i(0)\tilde{x}_i, S_j(1)\tilde{x}_j, S_3(t),\dots,S_n(t)) + g(S_i(0)\tilde{x}_i, S_j(0)\tilde{x}_j, S_3(t),\dots,S_n(t)). \end{aligned} \tag{E.1}$$

Each term corresponds to one configuration of feature inclusion, and the alternating signs implement a discrete second-order finite difference over the lifted inputs. By multilinearity of g, $Q_{\{i,j\}}(t;x)$ is a polynomial in t of degree at most $n-2$:

$$Q_{\{i,j\}}(t;x) = \sum_{s=0}^{n-2} c_s^{(ij)}(x)\, t^s,$$

where $c_s^{(ij)}(x)$ aggregates all coalitional contributions of size s among the remaining $n-2$ features.

Polynomial interpolation. Evaluating (E.1) at $n-1$ distinct points $\{t_\ell\}_{\ell=0}^{n-2}\subset(0,1)$ gives the Vandermonde system

$$V\,c^{(ij)} = q, \qquad V_{\ell+1,s+1} = t_\ell^s, \quad q_\ell = Q_{\{i,j\}}(t_\ell;x),$$

whose solution $c^{(ij)}$ recovers all size-aggregated coefficients. These coefficients encode how the rest of the features interact jointly with the pair $\{i,j\}$.

Recovering the interaction index. The pairwise Shapley–Taylor (or SII) interaction index (Grabisch and Roubens, 1999a; Sundararajan and Najmi, 2020) follows as a weighted sum of the coefficients:

$$\Phi^{\mathrm{SII}}(\{i,j\};x) = \sum_{s=0}^{n-2} \beta_s(n,2)\, c_s^{(ij)}(x), \qquad \beta_s(n,2) = \frac{s!\,(n-2-s)!}{(n-1)!}. \tag{E.2}$$

The weights $\beta_s(n,2)$ ensure the correct combinatorial normalization across coalition sizes.

Algorithm 2 implements this procedure end-to-end: it builds lifted inputs, evaluates the inclusion–exclusion probe at $n-1$ interpolation points, solves the Vandermonde system, and combines the coefficients to produce $\Phi^{\mathrm{SII}}(\{i,j\};x)$. This pairwise case illustrates the fundamental computational structure of our method and directly generalizes to higher-order feature interactions.

Algorithm 2 TN-SHAP Pairwise Interaction $\Phi^{\mathrm{SII}}(\{i,j\};x)$

Require: Instance $x \in \mathbb{R}^n$; feature pair (i,j); distinct probe points $\{t_\ell\}_{\ell=0}^{n-2}$
Ensure: Pairwise SII $\Phi^{\mathrm{SII}}(\{i,j\};x)$
Lift once: $\tilde{x}_r \leftarrow [\phi_r(x_r), 1]$ for all $r \in N$
for $\ell = 0$ **to** $n-2$ **do** ▷ degree $\le n-2 \Rightarrow n{-}1$ points
  $\tilde{x}^{(\ell)}_{r\notin\{i,j\}} \leftarrow S_r(t_\ell)\,\tilde{x}_r$
  $\tilde{x}_i^{(1)} \leftarrow S_i(1)\tilde{x}_i, \quad \tilde{x}_i^{(0)} \leftarrow S_i(0)\tilde{x}_i$
  $\tilde{x}_j^{(1)} \leftarrow S_j(1)\tilde{x}_j, \quad \tilde{x}_j^{(0)} \leftarrow S_j(0)\tilde{x}_j$
  $q_\ell \leftarrow \; g(\dots,\tilde{x}_i^{(1)},\tilde{x}_j^{(1)},\dots) - g(\dots,\tilde{x}_i^{(1)},\tilde{x}_j^{(0)},\dots) - g(\dots,\tilde{x}_i^{(0)},\tilde{x}_j^{(1)},\dots) + g(\dots,\tilde{x}_i^{(0)},\tilde{x}_j^{(0)},\dots)$ ▷ 4 TN forwards
end for
Build Vandermonde $V \in \mathbb{R}^{(n-1)\times(n-1)}$, $V_{\ell+1,s+1} = t_\ell^s$
Solve $V\,c^{(ij)} = q$ (e.g., QR) to obtain coefficients $c_s^{(ij)}$, $s = 0{:}n-2$
Combine: $\Phi^{\mathrm{SII}}(\{i,j\};x) = \sum_{s=0}^{n-2} \beta_s(n,2)\, c_s^{(ij)}$
return $\Phi^{\mathrm{SII}}(\{i,j\};x)$
Complexity: $4(n{-}1)$ TN forwards + $O((n{-}1)^2)$ for the solve; each forward is $O(\mathrm{poly}(\chi))$.

Complexity. The procedure requires $4(n-1)$ forward evaluations of the tensor network (since $2^2 = 4$ inclusion patterns) and $O((n-1)^2)$ time for solving the Vandermonde system. This cost is polynomial in n and the bond dimension χ, and it reveals the essential tradeoff that generalizes to the 2^k scaling for higher-order interactions.

E.2 Higher-Order Feature Interactions

The same principles underlying the pairwise case extend naturally to any coalition $C \subseteq N$ of size $|C| = k$. Higher-order interactions quantify how the joint effect of a group of k features differs from the sum of their contributions taken in smaller coalitions (Grabisch and Roubens, 1999b), capturing nonlinear and synergistic dependencies that arise only when these features act together.

General inclusion–exclusion probe. For an arbitrary coalition C with $|C| = k$, the k-th order marginal contribution is defined as the alternating sum over all 2^k inclusion patterns:

$$Q_C(t;x) \;=\; \sum_{\mathbf{b}\in\{0,1\}^k} (-1)^{k-\|\mathbf{b}\|_1}\; g\Big(\{S_r(t)\tilde{x}_r\}_{r\notin C}, \{S_i(b_i)\tilde{x}_i\}_{i\in C}\Big), \tag{E.3}$$

where $\mathbf{b} = (b_i)_{i\in C}$ is a binary vector indicating inclusion ($b_i = 1$) or exclusion ($b_i = 0$) of each feature i in C. The signs implement inclusion–exclusion over C, while the selectors $S_r(t)$ modulate the contribution of the remaining $n-k$ features. By multilinearity, $Q_C(t;x)$ is a univariate polynomial in t of degree at most $n-k$:

$$Q_C(t;x) = \sum_{s=0}^{n-k} c_s^{(C)}(x)\, t^s,$$

where each coefficient $c_s^{(C)}(x)$ aggregates the contributions of all coalitions of size s among the remaining features.

Interpolation and coefficient recovery. Evaluating $Q_C(t;x)$ at $n-k+1$ distinct probe points $\{t_\ell\}_{\ell=0}^{n-k} \subset (0,1)$ gives the Vandermonde system

$$V\,c^{(C)} = q, \qquad V_{\ell+1,s+1} = t_\ell^s, \quad q_\ell = Q_C(t_\ell;x),$$

whose solution $c^{(C)} = (c_0^{(C)},\dots,c_{n-k}^{(C)})^\top$ recovers all size-grouped contributions.

Interaction index reconstruction. The order-k Shapley–Taylor (or SII) interaction index (Grabisch and Roubens, 1999a; Sundararajan and Najmi, 2020) follows as a size-weighted linear combination of these coefficients:

$$\Phi^{\mathrm{SII}}(C;x) = \sum_{s=0}^{n-k} \beta_s(n,k)\, c_s^{(C)}(x), \qquad \beta_s(n,k) = \frac{s!\,(n-k-s)!}{(n-k+1)!}. \tag{E.4}$$

This expression exactly recovers the canonical multilinear extension of the SII under size-based weighting. Algorithm 3 implements the full procedure: for a target coalition C, it constructs the inclusion–exclusion probe (E.3), evaluates it at $n-k+1$ probe points, solves the Vandermonde system, and combines the recovered coefficients via (E.4). This general routine reduces to the pairwise algorithm when $k=2$.

Algorithm 3 TN-SHAP k-Way Interaction $\Phi^{\mathrm{SII}}(S;x)$ for $S \subseteq N,\ |S|=k$

Require: Instance $x \in \mathbb{R}^n$; target set $S \subseteq N$ with $|S|=k$; points $\{t_\ell\}_{\ell=0}^{n-k}$
Ensure: $\Phi^{\mathrm{SII}}(S;x)$
Lift once: $\tilde{x}_r \leftarrow [\phi_r(x_r), 1]$ for all $r \in N$
for $\ell = 0$ **to** $n-k$ **do** ▷ degree $\le n-k \Rightarrow n{-}k{+}1$ points
  $\tilde{x}^{(\ell)}_{r\notin S} \leftarrow S_r(t_\ell)\,\tilde{x}_r$
  $q_\ell \leftarrow \sum_{\mathbf{b}\in\{0,1\}^k} (-1)^{k-\|\mathbf{b}\|_1}\, g\big(\{\tilde{x}^{(\ell)}_{r\notin S}\}, \{S_i(b_i)\tilde{x}_i\}_{i\in S}\big)$ ▷ 2^k TN forwards per node
end for
Build Vandermonde $V \in \mathbb{R}^{(n-k+1)\times(n-k+1)}$, $V_{\ell+1,s+1} = t_\ell^s$
Solve $V\,c^{(S)} = q$ to obtain $c_s^{(S)},\ s = 0{:}n-k$
Combine: $\Phi^{\mathrm{SII}}(S;x) = \displaystyle\sum_{s=0}^{n-k} \beta_s(n,k)\, c_s^{(S)}$
return $\Phi^{\mathrm{SII}}(S;x)$

Weights: $\beta_s(n,k) = \dfrac{s!\,(n-k-s)!}{(n-k+1)!}$ (standard SII normalization; adjust if a different convention is used).
Complexity: $2^k(n{-}k{+}1)$ TN forwards + $O((n{-}k{+}1)^2)$ for the solve; each forward is $O(\mathrm{poly}(\chi))$.

Computational complexity. A naive implementation requires 2^k evaluations of g for each probe point (one for each inclusion pattern) and $O((n-k+1)^2)$ time for solving the Vandermonde system. The overall complexity is therefore

$$O\big(2^k(n-k+1)\,\mathrm{poly}(\chi)\big) + O((n-k+1)^2) = O\big(2^k n\,\mathrm{poly}(\chi) + n^2\big),$$

polynomial in n and the bond dimension χ, with exponential dependence on k arising from the intrinsic combinatorial structure of the inclusion–exclusion expansion. The signed–toggle simplification described in §E.3 reduces this dependence to $O(1)$ per probe point while preserving the same interpolation-based recovery.

E.3 Signed–Toggle Simplification

The inclusion–exclusion probe above naively requires 2^k evaluations per subset S. When the surrogate g is multilinear in the lifted inputs $\{\tilde{x}_i\}$ and the selectors act linearly on those inputs, the exponential loop collapses to a single contraction. By multilinearity, $g(\ldots, a_i{+}b_i, \ldots) = g(\ldots, a_i, \ldots) + g(\ldots, b_i, \ldots)$, hence

$$g(\ldots, S_i(1)\tilde{x}_i, \ldots) - g(\ldots, S_i(0)\tilde{x}_i, \ldots) = g\big(\ldots, \big(S_i(1){-}S_i(0)\big)\tilde{x}_i, \ldots\big).$$

Applying this recursively for all $i \in S$ yields

$$\sum_{\mathbf{b}\in\{0,1\}^k} (-1)^{k-\|\mathbf{b}\|_1}\, g\big(\ldots, \{S_\ell(b_\ell)\tilde{x}_\ell\}_{\ell\in S}, \ldots\big) \;=\; g\big(\ldots, \{(S_\ell(1){-}S_\ell(0))\tilde{x}_\ell\}_{\ell\in S}, \ldots\big). \tag{E.5}$$

For the diagonal selectors $S_i(t) = \mathrm{Diag}(t\,I_{d_i-1},\, 1)$ used here,

$$S_i(1) - S_i(0) = \mathrm{Diag}(I_{d_i-1},\, 0),$$

which zeroes the bias channel and keeps only the data channels. Thus each $(S_i(1) - S_i(0))$ acts as a discrete derivative / projection operator on feature i.

Collapsed probe and complexity. With Equation E.5, the probe for a subset S becomes

$$Q_S(t;x) \;=\; g\big(\{S_r(t)\tilde{x}_r\}_{r\notin S},\ \{(S_i(1) - S_i(0))\tilde{x}_i\}_{i\in S}\big),$$

reducing the per–probe-point cost from 2^k forwards to 1 forward. Evaluating at $n{-}k{+}1$ points and solving the Vandermonde system (degree $\le n{-}k$) gives the overall complexity

$$O\big((n{-}k{+}1)\,\mathrm{poly}(\chi)\big) + O\big((n{-}k{+}1)^2\big) = O\big(n\,\mathrm{poly}(\chi) + n^2\big).$$

The interpolation step and the size–based reconstruction $\Phi^{\mathrm{SII}}(S;x) = \sum_{s=0}^{n-k} \beta_s(n,k)\, c_s^{(S)}(x)$ are unchanged.

F Comprehensive Experimental Evaluation

We provide full experimental protocols, extended baselines comparisons, and additional ablation studies beyond those in the main paper. This includes detailed hyperparameter settings, convergence criteria, cohort selection procedures, and the complete teacher-student rank sweep analysis. We also present comprehensive runtime-accuracy trade-offs across all UCI benchmarks and analyze the impact of feature map dimensionality on higher-order interaction recovery.

F.1 Experimental Setup and Protocols

Hardware and Environment. All experiments were conducted on NVIDIA Tesla V100-SXM2-32GB GPUs with CUDA acceleration. We used PyTorch (Paszke et al., 2019) with automatic mixed precision for training and inference. For reproducibility, we fixed random seeds (seed=42 for synthetic experiments, seed=2711 for UCI benchmarks) and recorded complete hardware specifications for each run. The primary compute nodes were Intel Xeon E5-2698 v4 @ 2.20GHz (503GB RAM).

Datasets. Table 6 summarizes the three UCI (Dua and Graff, 2019; Yeh, 2007; Tsanas and Xifara, 2012) regression tasks used in our experiments. All features and targets were standardized with scikit-learn's `StandardScaler` (Pedregosa et al., 2011). For local explanations, we selected cohorts of 100 neighbors via k-NN in standardized feature space around each test instance.

Table 6: Dataset characteristics used in our experiments.

Dataset	Task	# Samples	# Features	Target
Diabetes	Regression	442	10	Disease progression
Concrete	Regression	1,030	8	Compressive strength
Energy (Y1)	Regression	768	8	Heating load

Teacher Models. We trained 3-layer MLPs (Rumelhart et al., 1986; Paszke et al., 2019) with architecture [input $\to$ 256 $\to$ 256 $\to$ 128 $\to$ 1] using ReLU activations. Teachers were trained until validation $R^2 \geq 0.95$ or 500 epochs, using Adam optimizer with learning rate 10^{-3} and early stopping (patience=50).

TN Surrogate Configuration. Each surrogate is modeled as a binary tensor tree (Grasedyck, 2010) with bond dimension $\chi = 16$. The feature maps are learned through single-hidden-layer MLPs with 64 ReLU (Nair and Hinton, 2010) units, producing scalar outputs that are concatenated with a bias term. For every test instance, the training data consist of two parts: a Gaussian neighborhood—using either $M = 100$ or the number of test samples, whichever is smaller, with standard deviation $\sigma = 0.1 \times \mathrm{std}(X_{\mathrm{train}})$—and $2n^2$ structured selector-weighted probes placed at Chebyshev–Gauss nodes (Trefethen, 2013). Optimization is performed using Adam with learning rate 10^{-3} for up to 1,500 epochs, applying early stopping based on validation R^2. A single surrogate is trained for each 100-point cohort to amortize computation across similar instances.

Baseline Methods. We benchmarked TN-SHAP against established Shapley and interaction estimators under matched query budgets. All baselines were implemented using the SHAP-IQ library (Muschalik et al., 2024b), and we included all baseline methods available in that framework: KernelSHAPIQ (Fumagalli et al., 2024), a regression-based variant of the Shapley Interaction Index (SII); PermutationSampling (Tsai et al., 2023b), a traditional mean-estimation approach; RegressionFSII (Tsai et al., 2023b), which performs feature selection via regression; and SHAPIQ Monte Carlo (Muschalik et al., 2024a), a Monte Carlo estimator of SII values. All methods were evaluated at budgets $B \in \{50, 100, 500, 1000, 2000, 10000\}$ model evaluations.

Evaluation Protocol. For each test instance: (1) compute exact Shapley values/interactions via exhaustive enumeration (ground truth), (2) apply TN-SHAP using the fitted local surrogate, (3) run all baselines with specified budgets, (4) measure cosine similarity and MSE relative to ground truth, (5) record wall-clock time with CUDA synchronization. Results were aggregated across test sets with mean $\pm$ std reported. We evaluated TN-SHAP on three UCI regression benchmarks, comparing against sampling-based baselines across interaction

orders $k \in \{1, 2, 3\}$. Tables 8–10 present comprehensive results for the Diabetes dataset, showing that TN-SHAP achieves comparable or superior accuracy while operating at millisecond timescales.

F.2 Synthetic Validation Experiments

We evaluate how well tensor network (TN) students of varying ranks recover Shapley values and higher-order interactions from known target functions. Each teacher defines a multilinear mapping $f : \mathbb{R}^4 \to \mathbb{R}$, and students are trained to fit f and reproduce its Shapley decomposition up to order $k = 3$ using our TN-SHAP interpolation procedure (thin-diagonal paths, degree-d fits from d+1 Chebyshev nodes, and inclusion–exclusion for $k \geq 2$).

We consider three teacher targets of increasing complexity:

- TensorTree (rank 3): Balanced binary TN with physical dimensions $[2, 2, 2, 2]$.
- TensorTree (rank 16): Higher-capacity variant.
- Generic Multilinear: Explicit multilinear function up to order 3 with sparsity 0.3.

Student TN ranks are $\{2, 4, 6, 8, 10, 16\}$. For each teacher–rank pair we train on 10,000 i.i.d. inputs $x \sim \mathcal{N}(0, I_4)$ for up to 1,500 epochs using Adam (learning rate 10^{-3}) with a ReduceLROnPlateau scheduler; early stopping employs patience = 200 with stringent loss/R^2 thresholds. We run **10** seeds per configuration; seeds are drawn deterministically from an RNG initialized with `12345`.

Immediately after training each student, we compute TNShap interactions for orders $k \in \{1, 2, 3\}$ on **128** random test inputs and report an *aggregated* R^2 by stacking all interaction values across inputs (one score per order). We summarize results as mean ± std across the 10 seeds.

Experiments ran on a Tesla V100–SXM2–32GB GPU; CUDA acceleration was enabled for training and batched evaluations.

Table 7: Aggregated Teacher–Student Rank Sweep (R^2 mean ± std across 10 seeds, 128 test points). Results shown for TNShap with TensorTree and Generic Multilinear teachers.

Teacher	Student Rank	Train R^2	Order-1 R^2	Order-2 R^2	Order-3 R^2
TensorTree Rank 3	2	0.8068 ± 0.3048	0.7853 ± 0.3503	0.8652 ± 0.2797	0.8428 ± 0.4298
	4	0.9970 ± 0.0040	0.9948 ± 0.0056	0.9994 ± 0.0008	0.9970 ± 0.0057
	6	0.9999 ± 0.0000	0.9992 ± 0.0003	1.0000 ± 0.0000	0.9993 ± 0.0003
	8	1.0000 ± 0.0000	0.9988 ± 0.0007	1.0000 ± 0.0000	0.9993 ± 0.0004
	10	1.0000 ± 0.0000	0.9992 ± 0.0006	1.0000 ± 0.0000	0.9994 ± 0.0004
	16	1.0000 ± 0.0000	0.9996 ± 0.0002	1.0000 ± 0.0000	0.9997 ± 0.0001
TensorTree Rank 16	2	0.7863 ± 0.2833	0.8034 ± 0.3037	0.7920 ± 0.2856	0.8726 ± 0.3516
	4	0.9978 ± 0.0056	0.9947 ± 0.0127	0.9990 ± 0.0028	0.9996 ± 0.0003
	6	1.0000 ± 0.0000	0.9994 ± 0.0002	1.0000 ± 0.0000	0.9998 ± 0.0001
	8	1.0000 ± 0.0000	0.9993 ± 0.0002	1.0000 ± 0.0000	0.9998 ± 0.0000
	10	1.0000 ± 0.0000	0.9994 ± 0.0004	1.0000 ± 0.0000	0.9998 ± 0.0001
	16	1.0000 ± 0.0000	0.9996 ± 0.0001	1.0000 ± 0.0000	0.9999 ± 0.0000
Generic Multilinear	2	0.8751 ± 0.1457	–5.5812 ± 9.6013	0.7849 ± 0.2398	–0.0658 ± 1.9349
	4	0.9996 ± 0.0008	0.9118 ± 0.0539	0.9991 ± 0.0019	0.9821 ± 0.0109
	6	1.0000 ± 0.0000	0.9410 ± 0.0092	1.0000 ± 0.0001	0.9861 ± 0.0023
	8	1.0000 ± 0.0000	0.9396 ± 0.0116	0.9999 ± 0.0001	0.9829 ± 0.0039
	10	1.0000 ± 0.0000	0.9430 ± 0.0091	1.0000 ± 0.0001	0.9832 ± 0.0031
	16	1.0000 ± 0.0000	0.9514 ± 0.0053	1.0000 ± 0.0001	0.9854 ± 0.0014

Table 7 reports the aggregated R^2 between teacher and student TNShap interaction tensors across ten random seeds. For both TensorTree teachers, performance improved rapidly with rank: low-rank students (rank 2) achieved moderate R^2 (≈ 0.8) across orders, while rank 4–6 students nearly matched the teachers ($R^2 > 0.99$). Beyond rank 6, all interaction orders reached near-perfect agreement ($R^2 > 0.999$), confirming that the student TNs can reliably recover the underlying Shapley structure once expressive capacity suffices. For the Generic Multilinear teacher, rank 2 students underfit, yielding unstable or even negative R^2, but ranks ≥ 4 stabilized

($R^2_{k=1} \approx 0.94$, $R^2_{k=3} \approx 0.98$), demonstrating that modest tensor ranks can already capture complex multilinear dependencies. Overall, these results validate that the proposed TN-SHAP approach faithfully recovers first-, second-, and third-order Shapley interactions when the student TN rank matches the intrinsic multilinearity of the target.

We performed a focused rerun for the *Generic Multilinear* teacher with a rank-2 student across **30** random seeds, reporting aggregated R^2 (stacked across test inputs) as mean ± std:

$$\text{Train } R^2 = 0.9592 \pm 0.0792, \qquad R^2_{k=1} = -0.4547 \pm 2.4141, \qquad R^2_{k=2} = 0.9245 \pm 0.1717, \qquad R^2_{k=3} = 0.8186 \pm 0.4037.$$

The negative and highly variable R^2 for k=1 is *expected* in this setting and reflects metric instability rather than a failure of TNShap. In the Generic Multilinear teacher, most of the signal resides in higher-order terms; consequently, the *first-order* Shapley values $\{\phi_i\}$ have *very small variance* across features and test points. Since $R^2 = 1 - \text{MSE}/\text{Var}(y)$, a near-zero $\text{Var}(\phi_i)$ makes R^2 ill-conditioned: even small absolute errors inflate the ratio and can yield large negative values. In contrast, the *pairwise* and *three-way* interactions (k=2, 3) exhibit larger variance and are therefore stably recovered by the student (high positive R^2). Practically, this indicates that with rank 2 the model underfits main effects while still capturing a substantial portion of higher-order structure. Figure 5 visualizes the safe-R^2 scores for different student ranks when the ground-truth tensor network has rank 14. Each cell corresponds to the reconstruction quality of order-k interactions for a given student rank. Performance improves monotonically with rank, saturating near 1.0 once the student rank exceeds the ground-truth rank. Lower-rank students underestimate higher-order interactions ($k \geq 3$), confirming that sufficient tensor rank is crucial for capturing complex coalitional structures.

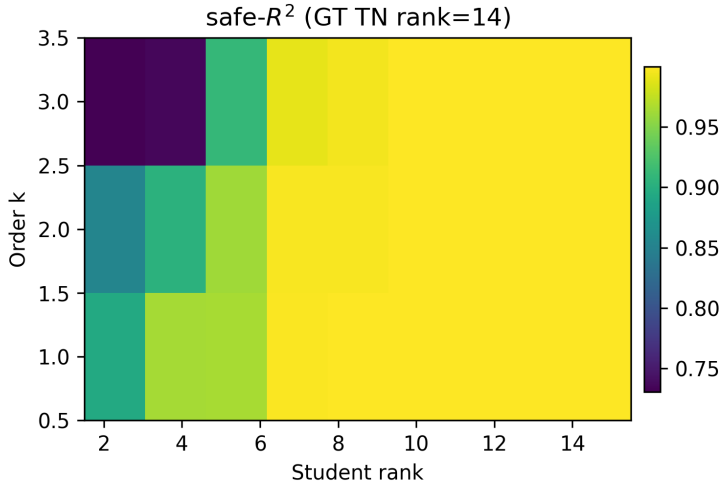

Figure 5: Rank ablation heatmap: safe-R^2 for student ranks under a ground-truth tensor network of rank 14.

Scaling Experiments. For dimensions $d \in \{10, 20, 30, 40, 50\}$, we generated CP-decomposed multilinear functions and fitted rank-16 tensor tree students. Each configuration used 10 test points with exact ground truth for runtime comparison against KernelSHAP-IQ. TN-SHAP achieved consistent millisecond-scale attribution across all dimensions, with runtime scaling approximately linearly in d.

F.3 UCI Benchmark Extended Results

We benchmark TN-SHAP on the UCI Diabetes regression task against standard Shapley/interaction estimators KernelSHAPIQ (Regression SII) (Fumagalli et al., 2024), RegressionFSII (Tsai et al., 2023a), and PermutationSamplingSII (Tsai et al., 2023a) , computing first-, second-, and third-order values under matched budgets and identical input preprocessing. The TN surrogate is a feature-mapped tensor network trained with GPU-accelerated contractions, enabling fast estimation of contributions and interactions while keeping the evaluation protocol identical across methods. We follow the experiment set up in Section F.1. The results in Tables 8–10

Table 8: Baseline Performance (Diabetes) — Order $k = 1$

Baseline	Order	Budget	Targets	Time (s)	Cosine Sim.	MSE
TN-SHAP	$k = 1$	–	89	**0.0069 ± 0.0323** (+0.0654)	0.9937 ± 0.0060	0.000499 ± 0.000397
KernelSHAPIQ	$k = 1$	50.0	89	0.0628 ± 0.0102	0.9827 ± 0.0109	0.001159 ± 0.000435
KernelSHAPIQ	$k = 1$	100.0	89	0.0898 ± 0.0177	0.9847 ± 0.0125	0.001124 ± 0.000436
KernelSHAPIQ	$k = 1$	500.0	89	0.2846 ± 0.0523	0.9910 ± 0.0086	0.000675 ± 0.000329
KernelSHAPIQ	$k = 1$	1000.0	89	0.4787 ± 0.0194	0.9863 ± 0.0114	0.001011 ± 0.000388
KernelSHAPIQ	$k = 1$	2000.0	89	0.4550 ± 0.0267	0.9923 ± 0.0064	0.000546 ± 0.000236
KernelSHAPIQ	$k = 1$	10000.0	89	0.4598 ± 0.0454	0.9921 ± 0.0071	0.000585 ± 0.000274
PermutationSamp	$k = 1$	50.0	89	0.1103 ± 0.0135	0.9897 ± 0.0082	0.000715 ± 0.000307
PermutationSamp	$k = 1$	100.0	89	0.1653 ± 0.0341	0.9853 ± 0.0115	0.001143 ± 0.000423
PermutationSamp	$k = 1$	500.0	89	0.6640 ± 0.0935	0.9913 ± 0.0082	0.000687 ± 0.000355
PermutationSamp	$k = 1$	1000.0	89	1.2574 ± 0.1079	0.9857 ± 0.0118	0.001042 ± 0.000401
PermutationSamp	$k = 1$	2000.0	89	2.5091 ± 0.2002	0.9921 ± 0.0065	0.000546 ± 0.000222
PermutationSamp	$k = 1$	10000.0	89	12.2420 ± 0.7666	0.9921 ± 0.0071	0.000585 ± 0.000272
RegressionFSII	$k = 1$	50.0	89	0.0606 ± 0.0069	0.9827 ± 0.0109	0.001159 ± 0.000435
RegressionFSII	$k = 1$	100.0	89	0.0875 ± 0.0111	0.9847 ± 0.0125	0.001124 ± 0.000436
RegressionFSII	$k = 1$	500.0	89	0.2793 ± 0.0289	0.9910 ± 0.0086	0.000675 ± 0.000329
RegressionFSII	$k = 1$	1000.0	89	0.4776 ± 0.0167	0.9863 ± 0.0114	0.001011 ± 0.000388
RegressionFSII	$k = 1$	2000.0	89	0.4536 ± 0.0149	0.9923 ± 0.0064	0.000546 ± 0.000236
RegressionFSII	$k = 1$	10000.0	89	0.4567 ± 0.0440	0.9921 ± 0.0071	0.000585 ± 0.000274
SHAPIQ (MonteCarlo SII)	$k = 1$	50.0	89	0.0610 ± 0.0059	0.8446 ± 0.0893	0.013974 ± 0.013453
SHAPIQ (MonteCarlo SII)	$k = 1$	100.0	89	0.0876 ± 0.0069	0.9120 ± 0.0684	0.008089 ± 0.007760
SHAPIQ (MonteCarlo SII)	$k = 1$	500.0	89	0.2751 ± 0.0265	0.9863 ± 0.0103	0.000989 ± 0.000387
SHAPIQ (MonteCarlo SII)	$k = 1$	1000.0	89	0.4850 ± 0.0339	0.9864 ± 0.0112	0.001011 ± 0.000398
SHAPIQ (MonteCarlo SII)	$k = 1$	2000.0	89	0.4575 ± 0.0276	0.9923 ± 0.0064	0.000546 ± 0.000236
SHAPIQ (MonteCarlo SII)	$k = 1$	10000.0	89	0.4616 ± 0.0703	0.9921 ± 0.0071	0.000585 ± 0.000274

compare the performance of TN-SHAP with several baseline Shapley estimation methods across increasing coalition orders $k = 1, 2, 3$ on the `Diabetes` dataset, (+0.0654) denotes the amortized training time (in seconds per test point) added to the per-point evaluation time for TN-SHAP. All other methods report per-point evaluation time only. For $k = 1$, TN-SHAP achieves near-perfect cosine similarity (0.9937 ± 0.0060) and extremely low mean-squared error (4.9×10^{-4}), outperforming all other baselines in accuracy and almost equal runtime to very low budget sampling methods (0.07 s versus 0.06–12 s). This confirms that the tensor network surrogate captures first-order (individual-feature) contributions efficiently and faithfully. At $k = 2$, the approximation difficulty increases: while TN-SHAP maintains a low MSE (3.8×10^{-4}) and fast inference, its cosine similarity (0.64 ± 0.19) reflects the inherent challenge of reconstructing pairwise interactions with a compact multilinear model. Nevertheless, TN-SHAP performs comparably or better than regression- and sampling-based methods, which require orders of magnitude more runtime. For $k = 3$, higher-order interaction estimation becomes unstable for all baselines, cosine similarity drops below 0.3 and MSE values remain small but noisy, yet TN-SHAP still retains competitive accuracy with the lowest computational cost. Overall, these results demonstrate that the tensor network approach provides an excellent trade-off between efficiency and fidelity, particularly at low and moderate interaction orders, validating its scalability as a surrogate for computing coalitional Shapley indices.

Table 9: Baseline Performance (Diabetes) — Order $k = 2$

Baseline	Order	Budget	Targets	Time (s)	Cosine Sim.	MSE
TN-SHAP	$k = 2$	–	89	**0.0058 ± 0.0097** (+0.0654)	0.6435 ± 0.1894	0.000377 ± 0.000339
KernelSHAPIQ	$k = 2$	50.0	89	0.0599 ± 0.0086	0.0909 ± 0.1035	0.126899 ± 0.123901
KernelSHAPIQ	$k = 2$	100.0	89	0.0891 ± 0.0136	0.4129 ± 0.1538	0.001030 ± 0.000212
KernelSHAPIQ	$k = 2$	500.0	89	0.2772 ± 0.0263	0.6505 ± 0.1831	0.000305 ± 0.000120
KernelSHAPIQ	$k = 2$	1000.0	89	0.4834 ± 0.0184	0.6462 ± 0.1761	0.000318 ± 0.000124
KernelSHAPIQ	$k = 2$	2000.0	89	0.4621 ± 0.0418	0.7204 ± 0.1681	0.000241 ± 0.000119
KernelSHAPIQ	$k = 2$	10000.0	89	0.4656 ± 0.0761	0.6787 ± 0.1834	0.000277 ± 0.000128
PermutationSamp	$k = 2$	50.0	89	0.0265 ± 0.0039	0.0000 ± 0.0000	0.000595 ± 0.000436
PermutationSamp	$k = 2$	100.0	89	0.0857 ± 0.0141	0.1560 ± 0.0966	0.000630 ± 0.000376
PermutationSamp	$k = 2$	500.0	89	0.2996 ± 0.0375	0.4324 ± 0.1572	0.000604 ± 0.000232
PermutationSamp	$k = 2$	1000.0	89	0.5874 ± 0.0487	0.4694 ± 0.1502	0.000811 ± 0.000161
PermutationSamp	$k = 2$	2000.0	89	1.1447 ± 0.1233	0.6941 ± 0.1509	0.000273 ± 0.000130
PermutationSamp	$k = 2$	10000.0	89	5.6492 ± 0.3430	0.6683 ± 0.1823	0.000286 ± 0.000122
RegressionFSII	$k = 2$	50.0	89	0.0585 ± 0.0040	0.0785 ± 0.0959	0.233139 ± 0.453246
RegressionFSII	$k = 2$	100.0	89	0.0867 ± 0.0061	0.5747 ± 0.1642	0.000443 ± 0.000150
RegressionFSII	$k = 2$	500.0	89	0.2765 ± 0.0160	0.6923 ± 0.1828	0.000263 ± 0.000127
RegressionFSII	$k = 2$	1000.0	89	0.4774 ± 0.0196	0.6892 ± 0.1814	0.000268 ± 0.000128
RegressionFSII	$k = 2$	2000.0	89	0.4549 ± 0.0232	0.7342 ± 0.1687	0.000230 ± 0.000121
RegressionFSII	$k = 2$	10000.0	89	0.4554 ± 0.0225	0.6916 ± 0.1866	0.000264 ± 0.000133
SHAPIQ (MonteCarlo SII)	$k = 2$	50.0	89	0.0619 ± 0.0107	0.0369 ± 0.1007	0.131029 ± 0.135230
SHAPIQ (MonteCarlo SII)	$k = 2$	100.0	89	0.0911 ± 0.0152	0.0624 ± 0.1233	0.043445 ± 0.039803
SHAPIQ (MonteCarlo SII)	$k = 2$	500.0	89	0.2819 ± 0.0403	0.2649 ± 0.1200	0.002771 ± 0.002456
SHAPIQ (MonteCarlo SII)	$k = 2$	1000.0	89	0.4899 ± 0.0220	0.6123 ± 0.1598	0.000365 ± 0.000139
SHAPIQ (MonteCarlo SII)	$k = 2$	2000.0	89	0.4621 ± 0.0150	0.7204 ± 0.1681	0.000241 ± 0.000119
SHAPIQ (MonteCarlo SII)	$k = 2$	10000.0	89	0.4619 ± 0.0207	0.6787 ± 0.1834	0.000277 ± 0.000128

Table 10: Baseline Performance (Diabetes) — Order $k = 3$

Baseline	Order	Budget	Targets	Time (s)	Cosine Sim.	MSE
TN-SHAP	$k = 3$	–	89	**0.0052 ± 0.0002** (+0.0654)	0.1433 ± 0.4013	0.000184 ± 0.000240
KernelSHAPIQ	$k = 3$	50.0	89	0.0964 ± 0.0099	-0.0060 ± 0.0731	0.000186 ± 0.000131
KernelSHAPIQ	$k = 3$	100.0	89	0.1645 ± 0.0319	0.0568 ± 0.0929	0.000248 ± 0.000101
KernelSHAPIQ	$k = 3$	500.0	89	0.3584 ± 0.0139	0.1476 ± 0.1919	0.000378 ± 0.000075
KernelSHAPIQ	$k = 3$	1000.0	89	0.5719 ± 0.0827	0.2009 ± 0.2349	0.000212 ± 0.000066
KernelSHAPIQ	$k = 3$	2000.0	89	0.5491 ± 0.0693	0.2730 ± 0.3015	0.000127 ± 0.000064
KernelSHAPIQ	$k = 3$	10000.0	89	0.5436 ± 0.0339	0.2252 ± 0.2796	0.000150 ± 0.000070
PermutationSamp	$k = 3$	50.0	89	0.0264 ± 0.0038	0.0000 ± 0.0000	0.000137 ± 0.000120
PermutationSamp	$k = 3$	100.0	89	0.0265 ± 0.0042	0.0000 ± 0.0000	0.000137 ± 0.000120
PermutationSamp	$k = 3$	500.0	89	0.2706 ± 0.0470	0.0693 ± 0.0799	0.000224 ± 0.000117
PermutationSamp	$k = 3$	1000.0	89	0.5102 ± 0.0702	0.0786 ± 0.1185	0.000386 ± 0.000103
PermutationSamp	$k = 3$	2000.0	89	0.9893 ± 0.1018	0.1530 ± 0.1659	0.000185 ± 0.000101
PermutationSamp	$k = 3$	10000.0	89	4.7880 ± 0.2098	0.1706 ± 0.2349	0.000209 ± 0.000076
RegressionFSII	$k = 3$	50.0	89	0.0587 ± 0.0038	0.0080 ± 0.0496	0.249489 ± 0.376970
RegressionFSII	$k = 3$	100.0	89	0.0917 ± 0.0069	-0.0040 ± 0.0579	0.887906 ± 0.275777
RegressionFSII	$k = 3$	500.0	89	0.2769 ± 0.0266	0.2364 ± 0.2925	0.000142 ± 0.000068
RegressionFSII	$k = 3$	1000.0	89	0.4805 ± 0.0153	0.2579 ± 0.3101	0.000133 ± 0.000067
RegressionFSII	$k = 3$	2000.0	89	0.4558 ± 0.0135	0.3096 ± 0.3361	0.000115 ± 0.000063
RegressionFSII	$k = 3$	10000.0	89	0.4641 ± 0.0354	0.2584 ± 0.3354	0.000128 ± 0.000071
SHAPIQ (MonteCarlo SII)	$k = 3$	50.0	89	0.0669 ± 0.0074	0.0059 ± 0.0578	0.051156 ± 0.053770
SHAPIQ (MonteCarlo SII)	$k = 3$	100.0	89	0.1011 ± 0.0177	0.0114 ± 0.0544	0.010474 ± 0.010887
SHAPIQ (MonteCarlo SII)	$k = 3$	500.0	89	0.2937 ± 0.0439	0.0183 ± 0.0473	0.017592 ± 0.016229
SHAPIQ (MonteCarlo SII)	$k = 3$	1000.0	89	0.5078 ± 0.0225	0.1013 ± 0.1486	0.000557 ± 0.000361
SHAPIQ (MonteCarlo SII)	$k = 3$	2000.0	89	0.4818 ± 0.0150	0.2730 ± 0.3015	0.000127 ± 0.000064
SHAPIQ (MonteCarlo SII)	$k = 3$	10000.0	89	0.4832 ± 0.0344	0.2252 ± 0.2796	0.000150 ± 0.000070

F.4 Feature Map Dimensionality Study

We study how the dimensionality of the per–feature nonlinear map $\psi : \mathbb{R} \to \mathbb{R}^{m_{\text{fmap}}}$ affects the quality of the learned multilinear surrogate. Each feature is first passed through a small MLP producing m_{fmap} output channels with $\psi(0) = 0$, and the resulting vector $\big[\psi(x_i), 1\big]$ forms the physical leg of each tensor–network (TN) leaf. We vary $m_{\text{fmap}} \in \{1, 2, 4, 8\}$, train a TN–based student on the dataset augmented with sampled interpolation points, generated from the `diabetes` regression benchmark, and evaluate its recovered interaction tensors against an exact teacher model for orders $k = 1, 2, 3$ using cosine similarity and mean–squared error (MSE).

m_{fmap}	Order $k = 1$		Order $k = 2$		Order $k = 3$	
	cos	MSE ($\times 10^{-4}$)	cos	MSE ($\times 10^{-4}$)	cos	MSE ($\times 10^{-4}$)
1	0.995	4.24	0.667	3.55	0.141	2.06
2	0.973	21.7	0.525	4.39	0.130	1.75
4	0.997	2.37	0.789	2.82	0.303	1.74
8	0.994	6.22	0.663	3.50	0.212	1.72

Table 11: TN selector vs. Teacher on `diabetes`: cosine similarity and mean squared error (MSE, scaled by 10^{-4}) between estimated and exact interaction tensors for different feature–map sizes.

For first–order effects ($k = 1$), all feature–map sizes yield near–perfect agreement with the teacher, confirming that linear attributions are easily captured. Higher–order interactions ($k = 2, 3$) benefit from richer embeddings: increasing m_{fmap} from 1 to 4 notably improves cosine similarity, suggesting that moderate latent width suffices to model nonlinear pairwise and triple dependencies, while larger maps give diminishing returns.

We also train a plain tensor network whose leaves receive only the raw augmented inputs $[x_i, 1]$, i.e., without any learned feature map ψ. This baseline isolates the contribution of the feature map while keeping the FewEval masking, shared Chebyshev grid, teacher model, training procedure, and evaluation metrics identical to the main study. On `diabetes`, the plain TN achieves strong first–order recovery (cos ≈ 0.996), moderate agreement for pairwise interactions (cos ≈ 0.71), and weak alignment for third–order terms (cos ≈ 0.12). These results indicate that the TN's multilinear structure over $[x_i, 1]$ captures most linear effects and part of the quadratic structure, whereas richer higher–order dependencies benefit from a learned per–feature embedding (e.g., a small–channel ψ), consistent with our ablation trends.

F.5 Code availability.

We provide an anonymized implementation of our method and experimental setup at `https://github.com/farzana0/TN-SHAP`.