Supplementary Document
Fast Generic Interaction Detection for Model Interpretability and Compression

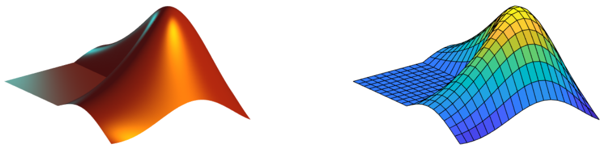
A PROBLEMS WITH ANALYTICAL EVALUATION

In this section, we first illustrate the conceptual difference between local and global interaction and then state the problems of analytical evaluation.

Illustration of Local vs. Global Interaction. A quick example showing the difference of local/global interaction is the MATLAB symbol[1] (Figure 6(a)). There is no local interaction inside the flat region, but two input variables globally interact.

Analytical evaluation works for neural networks with differentiable activation functions, *e.g.*, `sigmoid`, `tanh`, etc. However, it is problematic when we use neural networks with piece-wise linear activation functions (PLNN), such as `ReLU`, `Leaky ReLU`, and so on. The Hessian will be a zero matrix at every second-order differentiable point, and it will no longer provide information about local interaction. Imagine the function landscape is joint with many facets, and the interaction information is hidden in the boundary of those facets (the set of non-differentiable points).

A simple example is checkerboard-like function, *e.g.*, XOR function $F(x_1, x_2) = \mathbb{1}\{x_1 x_2 > 0\}$, where x_1, x_2 are uniformly drawn from $[-1, 1]$. In this case, x_1 and x_2 are clearly interacted, however, the second-order derivatives $\frac{\partial F}{\partial x_1 \partial x_2}$ will be zero for almost all sampled points. The interaction information is hidden at the line $x_1 = 0$ and $x_2 = 0$.



(a) origin function (b) approximated function

Figure 6: The MATLAB symbol. (a)Imagine the MATLAB symbol as the landscape of a function $F(x_1, x_2)$, there is no interaction locally in the flat region, but x_1 and x_2 interact globally, i.e., there is no global decomposition as $F(x_1, x_2) = a(x_1) + b(x_2)$. (b) (Problems with analytical evaluation) The function is approximated by a PLNN, and the landscape is spliced by flat facets. The Hessian matrix is a zero matrix on almost every point (no local interaction), but x_1 and x_2 still interact globally.

B PROOF OF THEOREM 3.1 AND DISCUSSION ON THE CHOICE OF h

Theorem 3.1. *For any x and y, function F shows no interaction between them, i.e., it can be decomposed as $F(x, y) = a(x) + b(y)$* ***if and only if****, $\forall\, h, k > 0$, $F(x+h, y+k) - F(x+h, y-k) - F(x-h, y+k) + F(x-h, y-k) = 0$.*

Proof. $(\Rightarrow)$ is trivial. $(\Leftarrow)$ We will directly have, $\forall h, k > 0$ and $\forall x, y$,

$$\frac{F(x+h, y+k) - F(x+h, y-k)}{k} = \frac{F(x-h, y+k) - F(x-h, y-k)}{k},$$

$$\frac{F(x+h, y+k) - F(x-h, y+k)}{h} = \frac{F(x+h, y-k) - F(x-h, y-k)}{h}.$$

---

[1] With permission of MathWorks.

Let $k \to 0$, we will have $\frac{\partial F}{\partial y}(x+h,y) = \frac{\partial F}{\partial y}(x-h,y)$ for any positive h; let $h \to 0$, we will have $\frac{\partial F}{\partial x}(x,y+k) = \frac{\partial F}{\partial x}(x,y-k)$ for any positive k. Thus, $\frac{\partial F}{\partial y}$ is irrespective of x and $\frac{\partial F}{\partial x}$ is irrespective of y. $\square$

Remark B.1. *Theorem 3.1 can be extended for higher-order interaction with the same machinery.*

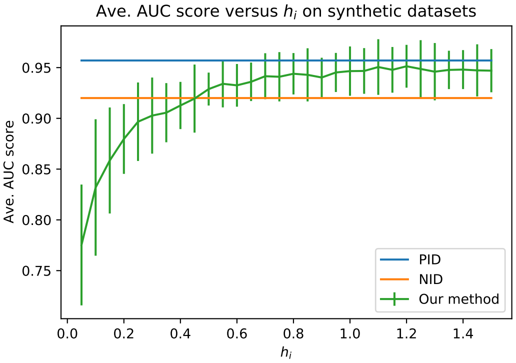


Figure 7: Average ROC-AUC score versus $h_{i(j)}$ on synthetic datasets

A guide for the choice of $h_{i(j)}$. We empirically test the sensitivity of $h_{i(j)}$, see Figure 7. For *standardized features*, we suggest using $h_{i(j)} \in [0.6, 0.9]$ as the rule-of-thumb.

Further discussions on the choice of $h_{i(j)}$. People may argue that for the interaction detection task, a small number of evaluations for each feature pair are already enough to get pretty good results, and the benefits of using the UCB algorithm is trivial. This is not always true. We address this from two perspectives in the following. First, in many scenarios, the function can only be evaluated sequentially, and the evaluation may be very expensive. The benefits of the UCB algorithm are revealed then. Second, we empirically study the performance of evaluating each arm evenly under different configurations (perturbation size $h_{i(j)}$, and the number of evaluations for each arm), see Figure 8. The ROC-AUC scores were obtained from the OverparaFC model trained as described in Section 5.2. We expect that given the fixed perturbation size h, the ROC-AUC scores will rise as the number of evaluations increases. This is consistent with the observation that the bricks are brighter from top to bottom. Also, we observed that with a properly chosen $h \in [0.6, 0.9]$, the interactions can be correctly detected with a relatively small number of evaluations (the bricks are brighter in the middle, see dataset 3, 4, and 5). The corresponding reasons can be that (a) if h is too small, four evaluated points lie in the same flat region with high probability, which is harmful to interaction detection; (b) if h is too large, some evaluated points may be out of the distribution of the training data, which makes the detected interaction strength less representative. Even though the suggestion of using $h_{i(j)} \in [0.6, 0.9]$ is purely empirical, the plots here help us better understand the essence of statistical interactions and their detection process.

C PROOF OF THEOREM 3.2

Recall that we construct the $1-\delta$ confidence intervals of $\hat{f}_\ell(r)$ as,

$$C(\ell) = \begin{cases} \sqrt{\frac{2\sigma^2 \log \frac{2}{\delta}}{\ell}} & \text{if } \ell \le m \\ 0 & \text{if } \ell > m \end{cases},$$

$$\mu_r \in \left[\hat{f}_\ell(r) - C(\ell), \hat{f}_\ell(r) + C(\ell)\right], \text{ w.p. } 1-\delta.$$

This is derived from concentration inequality (Inequality 2.9 in Wainwright (2019)). We set $\delta = \frac{2}{n^3 m}$, which is a small number and easy to analyze. We have,

Lemma C.1 (high probability event/clean event)**.** *With probability (w.p.) $1 - \frac{2}{n^2}$, all true values μ_r lie in the their confidence intervals during the run of the algorithm.*

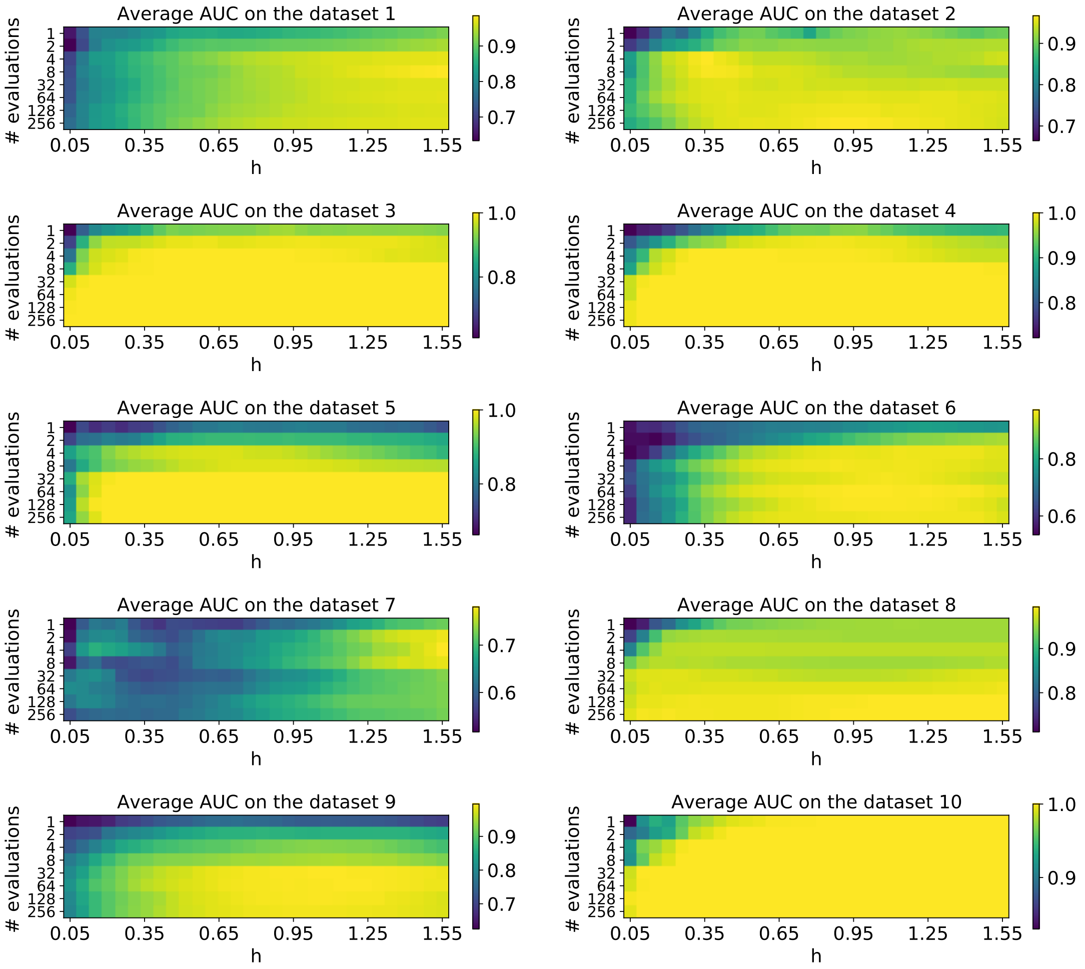


Figure 8: The ROC-AUC scores obtained for 10 synthetic datasets under different configurations of perturbation size h (horizontal axis), and the number of evaluations on each interaction pair (vertical axis). We expect that given the fixed perturbation size h, the ROC-AUC scores will rise as the number of evaluations increases. This is consistent with the observation that the bricks are brighter from top to bottom. Also, we observed that with a properly chosen $h \in [0.6, 0.9]$, the interactions can be correctly detected with a relatively small number of evaluations (the bricks are brighter in the middle, see dataset 3, 4, and 5).

Proof. It's equivalent to proof:

$$P(\{\, \forall\, t, \forall\, r \in [n], |\mu_r - \hat{\mu}_r(t)| \leq C_r(t)\}) \geq 1 - \frac{2}{n^2}$$

The opposite of this event is

$$\exists\, t \text{ and } r,\ \text{s.t.}, |\mu_r - \hat{\mu}_r(t)| > C_r(t).$$

Since we have,

$$P(\{|\mu_r - \hat{\mu}_r(t)| > C_r(t)\}) \leq \delta = \frac{2}{n^3 m},$$

and we evaluate at most nm times (n arms, each arm is pulled m times),

$$P(\{\exists\, t \text{ and } r,\ \text{s.t.}, |\mu_r - \hat{\mu}_r(t)| > C_r(t)\}) \leq \delta n m = \frac{2}{n^2} \quad \textit{(union bound)}$$

$$P(\{\, \forall\, t, \forall\, r \in [n], |\mu_r - \hat{\mu}_r(t)| \leq C_r(t)\}) \geq 1 - \frac{2}{n^2} \quad \textit{(clean event)}$$

□

Using Lemma C.1, we restate Theorem 3.2 as follows.

Theorem C.2 (Restating Theorem 3.2)**.** *With probability* $1 - \Theta(\frac{1}{n^2})$*, Algorithm 1 returns the* k*-strongest interactions with at most*

$$M \leq \sum_{i=1}^{n} \left(\left(\frac{24\sigma^2}{\Delta_i^2} \log(nm) \right) \wedge m \right)$$

local interaction strength evaluations. This takes $\mathcal{O}\left(\sum_{i=1}^{n} \log(n) \left(\left(\frac{\sigma^2 \log(nm)}{\Delta_i^2} \right) \wedge m \right) \right)$ *time.*

Proof. We first analyze the running of algorithm till the first strongest interaction comes out. If we choose to update arm $i \neq i_1^*$ at time t, then we have

$$\hat{\mu}_i(t) + C_i(t) \geq \hat{\mu}_{i_1^*}(t) + C_{i_1^*}(t). \tag{6}$$

For equation 6 to occur, at least one of the following three events must occur:

$$\begin{aligned} \mathcal{E}_1 &= \left\{ \hat{\mu}_{i_1^*}(t) \leq \mu_{i_1^*} - C_{i_1^*}(t) \right\}, \\ \mathcal{E}_2 &= \left\{ \hat{\mu}_i(t) \geq \mu_i + C_i(t) \right\}, \\ \mathcal{E}_3 &= \left\{ \Delta_i^{(1)} = \mu_{i_1^*} - \mu_i \leq 2C_i(t) \right\}. \end{aligned}$$

To see this, note that if none of $\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3$ occurs, we have

$$\hat{\mu}_i(t) + C_i(t) \overset{(\neg\mathcal{E}_2)}{<} \mu_i + 2C_i(t) \overset{(\neg\mathcal{E}_3)}{<} \mu_{i_1^*} \overset{(\neg\mathcal{E}_1)}{<} \hat{\mu}_{i_1^*} + C_{i_1^*}(t).$$

From Lemma C.1, $\mathcal{E}_1$ and $\mathcal{E}_2$ do not occur during the run of the algorithm with probability $1 - \frac{2}{n^2}$, because

$$\text{w.p. } \left(1 - \frac{2}{n^2}\right) : |\mu_i - \hat{\mu}_i(t)| \leq C_i(t), \forall\, i \in [n], \ \forall\, t. \tag{7}$$

It also implies w.p. $1 - \Theta(\frac{1}{n^2})$, the algorithm does not stop pulling arm i until event $\mathcal{E}_3$ stops occurring.

Let $\zeta_i^{(w)}$ index the iteration in which Algorithm 1 evaluates arm $\mathcal{A}_i$ for the last time before declaring it to be the w-th strongest interaction. Let's first consider how many pulls are needed for each arm to pick out the first strongest interaction. If $C_i(\zeta_i^{(1)}) \leq \frac{\Delta_i^{(1)}}{2}$ for arm $\mathcal{A}_i$, then we can stop evaluating arm $\mathcal{A}_i$, that is,

$$\begin{aligned} \frac{\Delta_i^{(1)}}{2} &\geq \sqrt{\frac{2\sigma^2 \log n^3 m}{T_i(\zeta_i^{(1)})}} \text{ or } C_i(\zeta_i^{(1)}) = 0, \\ \Longrightarrow T_i(\zeta_i^{(1)}) &\geq \frac{8\sigma^2}{(\Delta_i^{(1)})^2} \log(n^3 m) \text{ or } T_i(\zeta_i^{(1)}) \geq m. \end{aligned}$$

Note that $T_i(t)$ denotes for the number of pulls of arm $\mathcal{A}_i$ at iteration t. We will evaluate arm $\mathcal{A}_i$ at most $\frac{8\sigma^2}{(\Delta_i^{(1)})^2} \log(n^3 m) \wedge m$ times. To pick out the first strongest interaction, we need at most M evaluations satisfying

$$M \leq \sum_{i=1}^{n} \left(\left(\frac{8\sigma^2}{(\Delta_i^{(1)})^2} \log(n^3 m) \right) \wedge m \right).$$

This result can be extend to find all top k strongest interactions concretely,

$$
\begin{aligned}
M &\leq \sum_{i=1}^{n} \left(\left(\frac{8\sigma^2}{(\Delta_i^{(k)})^2} \log(n^3 m) \right) \wedge m \right), \\
&= \sum_{i=1}^{n} \left(\left(\frac{24\sigma^2}{(\Delta_i^{(k)})^2} \log(n \sqrt[3]{m}) \right) \wedge m \right), \\
&\leq \sum_{i=1}^{n} \left(\left(\frac{24\sigma^2}{(\Delta_i^{(k)})^2} \log(nm) \right) \wedge m \right).
\end{aligned}
$$

The Algorithm 1 takes $\mathcal{O}\left(\sum_{i=1}^{n} \log(n) \left(\left(\frac{\sigma^2 \log(nm)}{\Delta_i^2} \right) \wedge m \right)\right)$ time, where $\mathcal{O}(\log(n))$ is the time for maintaining a priority queue(to find the minimal LCB or maximal UCB) in each iteration. $\square$

Theorem C.3 (Remark 3.1)**.** *If we further assume that $\Delta_i \sim \mathcal{N}(\gamma, 1)$, for some γ, and $m = cn$, for $c \in [0, 1]$, then the expected pulls of arms (over randomness in Δ_i) satisfies*

$$\mathbb{E}[M] \leq \mathcal{O}(n \log(nm) + km)$$

with probability $1 - \Theta(\frac{1}{n^2})$ over randomness in Algorithm 1. Thus, the expected running time is well bounded by $\mathcal{O}(n \log(nm) \log(n) + km \log(n))$.

Proof. This follows directly from the Appendix 2 of (Bagaria et al., 2018a). $\square$

D DATASETS PRE-PROCESSING

D.1 SYNTHETIC TEST SUITE

To make our experimental results convincing, we follow the test suite ever used in (Tsang et al., 2018a) with the details given in Table 4. Among others, F_1 is a widely used test function for interaction detection, which can be generated as described in (Sorokina et al., 2008). For all the other functions, the input dimension is set to 10, and $x_1, \ldots, x_{10} \overset{i.i.d.}{\sim} U(-1, 1)$.

Table 4: Test suite of data-generating functions

$F_1(\mathbf{x})$	$\pi^{x_1 x_2} \sqrt{2x_3} - \sin^{-1}(x_4) + \log(x_3 + x_5) - \frac{x_9}{x_{10}} \sqrt{\frac{x_7}{x_8}} - x_2 x_7$
$F_2(\mathbf{x})$	$\pi^{x_1 x_2} \sqrt{2\lvert x_3\rvert} - \sin^{-1}(0.5x_4) + \log(\lvert x_3 + x_5\rvert + 1) + \frac{x_9}{1 + \lvert x_{10}\rvert} \sqrt{\frac{x_7}{1 + \lvert x_8\rvert}} - x_2 x_7$
$F_3(\mathbf{x})$	$\exp \lvert x_1 - x_2\rvert + \lvert x_2 x_3\rvert - (x_3^2)^{\lvert x_4\rvert} + \log(x_4^2 + x_5^2 + x_7^2 + x_8^2) + x_9 + \frac{1}{1 + x_{10}^2}$
$F_4(\mathbf{x})$	$\exp \lvert x_1 - x_2\rvert + \lvert x_2 x_3\rvert - (x_3^2)^{\lvert x_4\rvert} + (x_1 x_4)^2 + \log(x_4^2 + x_5^2 + x_7^2 + x_8^2) + x_9 + \frac{1}{1 + x_{10}^2}$
$F_5(\mathbf{x})$	$\frac{1}{1 + x_1^2 + x_2^2 + x_3^2} + \sqrt{\exp(x_4 + x_5)} + \lvert x_6 + x_7\rvert + x_8 x_9 x_{10}$
$F_6(\mathbf{x})$	$\exp(\lvert x_1 x_2\rvert + 1) - \exp(\lvert x_3 + x_4\rvert + 1) + \cos(x_5 + x_6 - x_8) + \sqrt{x_8^2 + x_9^2 + x_{10}^2}$
$F_7(\mathbf{x})$	$(\arctan(x_1) + \arctan(x_2))^2 + \max(x_3 x_4 + x_6, 0) - \frac{1}{1 + (x_4 x_5 x_6 x_7 x_8)^2} + \left(\frac{\lvert x_7\rvert}{1 + \lvert x_9\rvert}\right)^5 + \sum_{i=1}^{10} x_i$
$F_8(\mathbf{x})$	$x_1 x_2 + 2^{x_3 + x_5 + x_6} + 2^{x_3 + x_4 + x_5 + x_7} + \sin(x_7 \sin(x_8 + x_9)) + \arccos(0.9 x_{10})$
$F_9(\mathbf{x})$	$\tanh(x_1 x_2 + x_3 x_4) \sqrt{\lvert x_5\rvert} + \exp(x_5 + x_6) + \log\left((x_6 x_7 x_8)^2 + 1\right) + x_9 x_{10} + \frac{1}{1 + \lvert x_{10}\rvert}$
$F_{10}(\mathbf{x})$	$\sinh(x_1 + x_2) + \arccos(\tanh(x_3 + x_5 + x_7)) + \cos(x_4 + x_5) + \sec(x_7 x_9)$

D.2 REAL DATASETS

All the real datasets are publicly available. We preprocessed the datasets as follows. For Parkinsons data, we remove the column *motor UPDRS* and predict the *total UPDRS*. For SkillCraft data, the target is set to be *LeagueIndex*. For the Bike sharing data, the feature *weather* is converted into a one-hot representation. Before feeding the data into neural networks, we performed data normalization first for all datasets.

The Drug combination dataset is processed as follows. There are 110 features extracted, among which the first 50 features are the concentration of 50 unique FDA-approved drugs, and the last 60 features are the one-hot encodings for the cell lines.

E ROC-AUC FOR PAIRWISE INTERACTION DETECTION

We calculate the ROC-AUC scores for the synthetic datasets, where the ground truth interaction pairs are known. To obtain the ROC-AUC value, two vectors are needed, namely the *pairwise interaction score* $\in \mathbb{R}_+^{45}$and *ground truth* $\in \{0,1\}^{45}$, both of them are of dimension $\frac{p(p-1)}{2} = 45$. By setting different thresholds for the pairwise interaction score ranking, different classifiers with False Positive (FP) rate and True Positive (TP) rate can be obtained. The ROC-AUC value is approximated from those (FP, TP) pairs by the trapezoidal rule.

F PAIRWISE INTERACTION DETECTION ON HIGH- DIMENSIONAL DATASETS

We created two new high-dimensional datasets by simply combining the 10 synthetic datasets considered in the paper. The new datasets have either 50 (using F1-F5) or 100 features (using F1-F10). The labels of the two datasets are the sum of the original labels in each dataset of input dimension $p = 10$. The training sample size is increased to 500,000 to mimic big data. We compared 5 different model architectures and our method consistently outperforms NID, see the Table 5 below for the results.

Table 5: ROC-AUCs comparison for high-dimensional datasets

ROC-AUC score (NID/Ours)	data size: 500,000 * 100	data size: 500,000 * 50
Big network[1] with main effect + L1reg	0.743/**0.768**	0.831/**0.864**
Big network without main effect + L1reg	0.699/**0.700**	0.803/**0.855**
Small network[2] with main effect + L1reg	0.731/**0.744**	0.836/**0.859**
Small network without main effect + L1reg	0.701/**0.742**	0.796/**0.840**
Standard big network without main effect	0.646/**0.653**	0.583/**0.793**

[1] Big network: p-5000-900-400-100-30-1
[2] Small network: p-140-100-60-20-1

Here we also report the computational complexity of our UCB-based interaction detection method. Due to the randomness of the UCB algorithm, the number of pulls fluctuates. Note that we select top $k = 200$ interaction pairs for 100-dimensional case, and the total number of pulls is around 40000, while the naive approach needs $100 \times 100 \times 99/2 = 495000$ pulls. We select top $k = 100$ interaction pairs for 50-dimensional case, and the total number of pulls is around 21000, while the naive approach needs $100 \times 50 \times 49/2 = 122500$ pulls.

Discussion on High Dimensional Data: For high dimensional data, the Hessian matrix will be too huge to handle. One possible solution is to focus on the important features. We may first take advantage of DeepLIFT(Shrikumar et al., 2017) or SHAP (Lundberg & Lee, 2017) to screen out the most important features, and then apply our interaction detection method. If the number of important features is still too large to handle, we may use ANOVA and F-test to select an affordable number of interaction candidates to pull $\log(n)$ times, and explore the remaining pairs with a small probability, say 5%, like the common strategy used in reinforcement learning..

G EXTENSION TO HIGHER-ORDER INTERACTIONS

We define three-way interaction as follows, and higher-order interactions can be defined similarly.

Definition G.1 (three-way interaction)**.** *A function* $F : \mathbb{R}^p \to \mathbb{R}$ *is said to exhibit an interaction among three of its variables* x_i*,* x_j *and* x_k *if,*

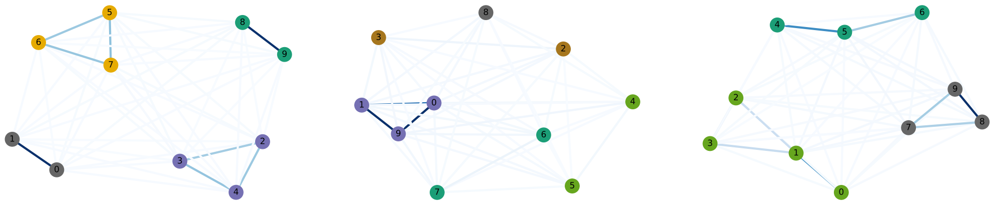
$$E_{\mathbf{x}}\left[\frac{\partial^3 F(\mathbf{x})}{\partial x_i \partial x_j \partial x_k}\right]^2 > 0.$$

The naive method to detect higher-order interactions is quite similar to FANOVA graph Muehlenstaedt et al. (2012). We give three examples of detecting higher-order interactions from the synthetic data. We detect the pairwise interaction strength first, then build a weighted graph (the edge weight corresponds to the interaction strength). By setting a proper number of clusters, higher-order interactions can be discovered correctly, see Figure 9.

(a) $y = x_0x_1 + x_2x_3x_4 + x_5x_6x_7 + x_8x_9$

(b) $y = e^{|x_0-x_1|} * x_9 + |x_2 * x_3| - x_4^{2|x_5|} + (x_6x_7)^2 + x_8$

(c) $y = x_0x_1 + |x_1 + x_2x_3| + x_4x_5 - (x_5^2)^{x_6} - e^{x_7+x_8x_9}$



(a) (b) (c)

Figure 9: The nodes (features) are clustered correctly. The color of edges indicates pairwise interaction strength. Nodes with the same color belong to the same cluster.

Algorithm 2 `Hierarchical Higher-Order Interaction Detection`

```
Require: The target number of i-way interactions $\{k^{(i)}\}_{i=2}^{p}$.
 1: Detect top-$k^{(2)}$ pairwise interaction via UCB algorithm.
 2: Construct an undirected graph $\mathcal{G}$ based on the detected pairwise interactions.
 3: Enumerate all the cliques $\mathfrak{C}$ in the graph $\mathcal{G}$.
 4: for $i = 3, 4, \cdots, p$ do
 5:     if $\mathfrak{C}^{(i)} = \varnothing$ then
 6:         break
 7:     end if
 8:     Find $\mathfrak{C}^{(i)}_{\text{refine}} = \{C \in \mathfrak{C}^{(i)} \mid \text{every } i-1 \text{ complete subgraph of } C \text{ admits a detected interaction}\}$
 9:     Detect the top $k^{(i)}$-strongest $i$-way interactions in $\mathfrak{C}^{(i)}_{\text{refine}}$ via the UCB algorithm.
10: end for
11: return All the detected interactions with their strengths.
```

One drawback of the above naive clustering method is that it doesn't allow for the overlap of nodes, i.e. one variable may appear in several interactions (F_8 in the test suite). To address the above problem, we propose the Algorithm 2, which detects the higher-order interactions hierarchically. We first construct an undirected graph $\mathcal{G}$ from the detected pairwise interactions. To shrink our search space for higher-order interactions, we restrict ourselves to the set of all cliques $\mathfrak{C}$ of the graph $\mathcal{G}$, where a clique $C \in \mathfrak{C}$ in a graph is a subset of vertices that are all joined by edges. We use $\mathfrak{C}^{(i)} \subseteq \mathfrak{C}$ to denote the collection of the i-cliques (the cliques with i vertices). For example, to detect 3-way interactions, one only needs to search in the $\mathfrak{C}^{(3)}$, which contains the cliques with three vertices. Furthermore, we can shrink the search space to $\mathfrak{C}^{(i)}_{\text{refine}}$ for i-way interactions, based on the detected $(i-1)$-way interactions (see line 8 in Algorithm 2). For example, while detecting 4-way interactions, we put 4-clique $\{2, 3, 4, 6\}$ into consideration only if all the 3-way interactions $\{3, 4, 6\}$, $\{2, 4, 6\}$, $\{2, 3, 6\}$, $\{3, 4, 6\}$ exist. We then use the UCB algorithm to verify if the cliques in $\mathfrak{C}^{(i)}_{\text{refine}}$ admit the interaction relationship. In this way, we detect the i-way interactions from the detected $(i-1)$-way interactions information.

The 3-way interaction strength can be calculated from Equation 8. However, with the order of interaction increasing, the number of function evaluations for one gradient computation increases geometrically. We do not recommend detecting the interaction whose order is higher than four, which is uneconomical to compute and hard to interpret. We found our proposed Algorithm 2 can successfully detect the higher-order interactions for F_8 in the test suite [2].

$$\begin{aligned}\frac{\partial^2 F(\mathbf{x})}{\partial x_i \partial x_j \partial x_k} \approx \frac{1}{8h^3}[&+F(\mathbf{x}+h(\mathbf{e}_i+\mathbf{e}_j+\mathbf{e}_k))+F(\mathbf{x}+h(-\mathbf{e}_i-\mathbf{e}_j+\mathbf{e}_k))\\ &+F(\mathbf{x}+h(\mathbf{e}_i-\mathbf{e}_j-\mathbf{e}_k))+F(\mathbf{x}+h(-\mathbf{e}_i+\mathbf{e}_j-\mathbf{e}_k))\\ &-F(\mathbf{x}+h(-\mathbf{e}_i+\mathbf{e}_j+\mathbf{e}_k))-F(\mathbf{x}+h(\mathbf{e}_i-\mathbf{e}_j+\mathbf{e}_k))\\ &-F(\mathbf{x}+h(\mathbf{e}_i+\mathbf{e}_j-\mathbf{e}_k))-F(\mathbf{x}+h(-\mathbf{e}_i-\mathbf{e}_j-\mathbf{e}_k))]\end{aligned} \tag{8}$$

H MORE ON THE PARAMETRIC ACE

In section 4, we introduced a parametric ACE model based on the following generalized linear additive model with main effects and pairwise feature interactions, namely,

$$\theta(Y)=\sum_{i=1}^{p}\beta_i s_i\left(x_{u_i};\boldsymbol{\theta}_{s_i}\right)+\sum_{i=1}^{R}\beta_{p+i} r_i\left(\mathbf{x}_{\mathcal{I}_i};\boldsymbol{\theta}_{r_i}\right)+\epsilon. \tag{9}$$

The consideration of interactions improves the model performance significantly, compare to Generalized Additive Neural Networks (GANN)(Potts, 1999; Agarwal et al., 2020)[3], see Table 6.

Table 6: NRMSE of the GANN versus our proposed ParaACE network on the synthetic datasets in (Table 4). (The results were averaged over 5 folds.)

	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	average	CR
GANN	0.033	0.063	0.064	0.068	0.053	0.052	0.028	0.032	0.040	0.033	**0.046**	**959**
ParaACE	0.025	0.035	0.031	0.030	0.038	0.025	0.016	0.019	0.023	0.021	**0.026**	**283**

The idea behind the original nonparametric ACE algorithm (Breiman & Friedman, 1985) is to alternate between an inner process for finding the optimal transformation functions of the inputs and an outer process for finding the optimal transformation function of the output.

Here, we made several modifications. First, all the transformation functions are represented by small subnetworks. Second, we reformulate the above regression equation approximately as

$$Y \approx \theta^{-1}\left(\sum_{i=1}^{p}\beta_i s_i\left(x_{u_i};\boldsymbol{\theta}_{s_i}\right)+\sum_{i=1}^{R}\beta_{p+i} r_i\left(\mathbf{x}_{\mathcal{I}_i};\boldsymbol{\theta}_{r_i}\right)\right). \tag{10}$$

Therefore, the fix-up layer shown in Figure 3 can be understood as the inverse optimal transformation of the output $\theta^{-1}(\cdot)$. To be more general, $\theta^{-1}(\cdot)$ takes all input transformations as individuals instead of their high-level summary (weighted sum). Our parametric ACE then alternately tunes the fix-up layer conditioned on the current optimal transformation layer (outer iteration) and tunes the optimal transformation layer conditioned on the current fix-up layer until some convergence condition is met.

Another important function of the fix-up layer is to alleviate the negative impacts of wrongly detected pairwise interactions and/or undetected higher-order interactions altogether on the output. For this purpose, we could make this fix-up layer a network of small subnetworks, so that the whole architecture is a network of small subnetworks. Each subnetwork can be regarded as a meta-neuron that is expected to be more competent than any SOTA activation function. With such a nice structure, we have divided the hyper-parameters naturally into blocks and the whole network can hopefully be made adaptive to new tasks more rapidly by tuning just a few blocks.

---

[2] A demo can be found at `http://github.com/xxx`.

[3] GANN is a parametric version of GAM, which only considers the transformations of univariates (single features).

I EXTENSION TO CLASSIFICATION TASK

We generated the datasets for binary classification from the synthetic regression datasets (1000 samples with injected noise). We choose the median of the target as a threshold to separate each synthetic regression the dataset into two classes. The comparison between the baseline OverparaFC and our proposed ParaACE in terms of classification accuracy is shown in Table 7. For real-world datasets, we picked **Higgs Boson** (Baldi et al., 2014), **Spambase** Dua & Graff (2017), and **Diabetes** (Turney, 1994).Both the classification accuracy and the compression ratio are reported in Table 8.

Table 7: Accuracy of the baseline OverparaFC versus our proposed ParaACE network on the synthetic classification datasets. The results were averaged over 5 folds.

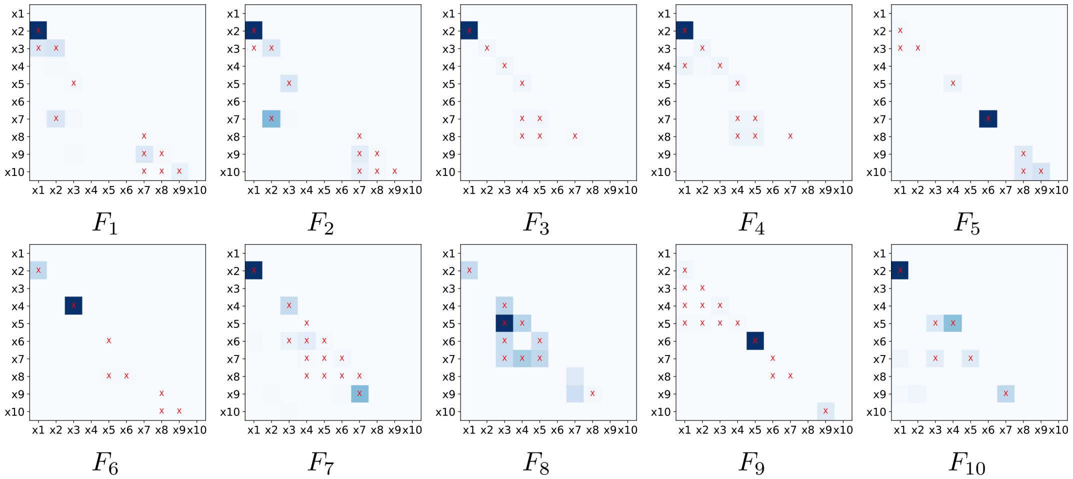
	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	Average	CR
OverparaFC	0.955	0.896	0.887	0.872	0.907	0.939	0.964	0.971	0.932	0.953	**0.927**	**1**
ParaACE	0.949	0.908	0.94	0.946	0.916	0.94	0.96	0.935	0.919	0.942	**0.936**	**283**

Table 8: Performance comparison between OverparaFC and ParaACE on various real-world classification datasets.

			OverparaFC		ParaACE	
Datasets	N	p	Accuracy	Parameters	Accuracy	CR
Higgs Boson	98050	28	$0.698 \pm 3.0e-3$	5049461	$\mathbf{0.730} \pm 2.8e-3$	**184**
Spambase	4601	57	$0.950 \pm 8.0e-3$	5194461	$\mathbf{0.950} \pm 7.9e-2$	**120**
Diabetes	768	8	$0.741 \pm 2.5e-2$	4949461	$\mathbf{0.789} \pm 1.7e-2$	**302**

J DETECTED PAIRWISE INTERACTIONS FOR SYNTHETIC AND REAL DATA

Synthetic data: Figure 10 shows the pairwise interaction strength produced by our proposed method.



F_1 F_2 F_3 F_4 F_5

F_6 F_7 F_8 F_9 F_{10}

Figure 10: Heat maps of pairwise interaction strength proposed by our method for functions F_1-F_{10} (Table 4). Cross-marks indicate the ground truth interactions.

Real data: We provide the detected interactions for five real datasets in Figure 11. For the Cal housing dataset, we terminate the UCB algorithm when $k = 20$ interactions stood out; for the other four datasets, we terminate at $k = 50$. The green, orange, and blue points represent the UCB, estimated mean, and LCB respectively. For the drug combination dataset, the results are shown in Figure 12.

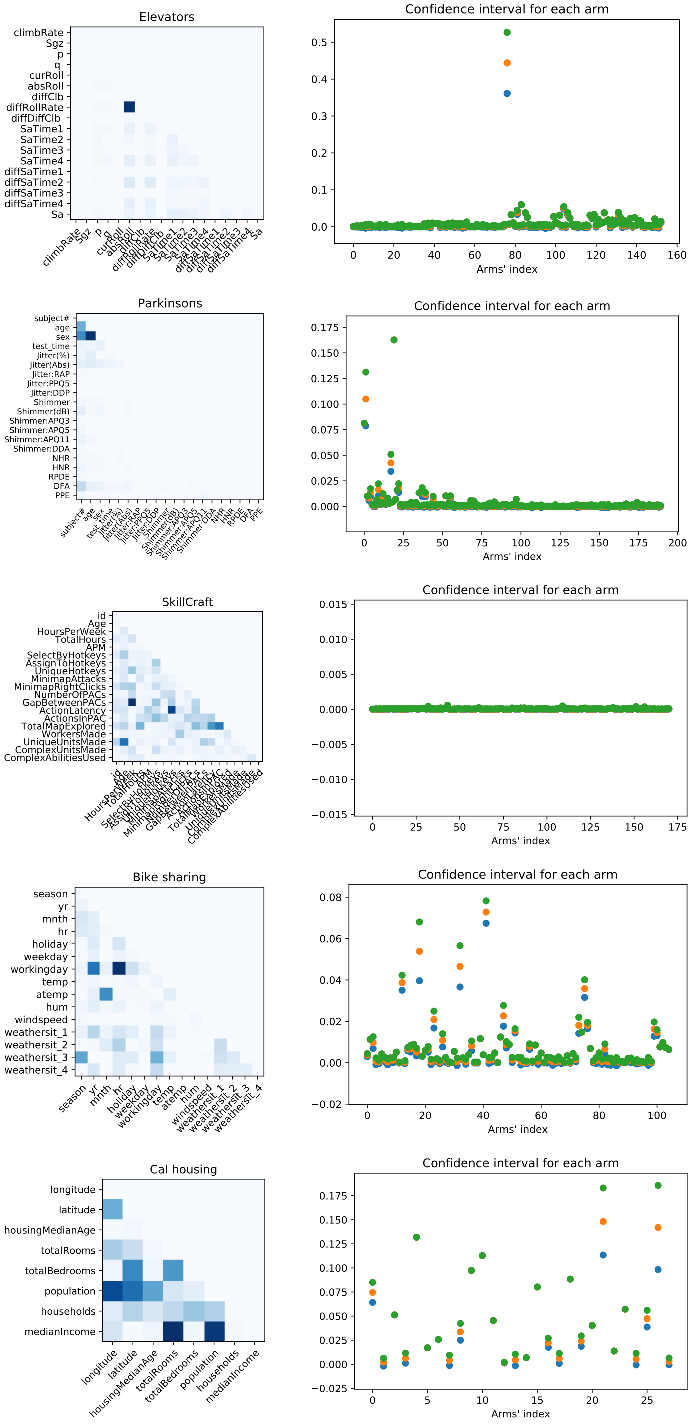
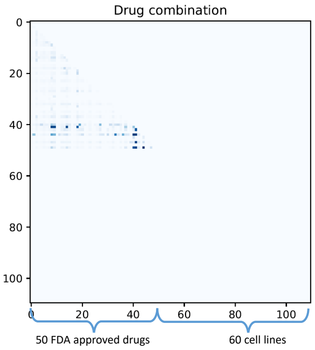


Figure 11: **Left:** Heat maps of pairwise interaction strengths on real datasets. **Right:** Confidence interval for each interaction pair. The green, orange, blue points denote the UCB, estimated mean, and LCB respectively.



Amifostine; Cabazitaxel
Estramustine phosphate sodium; Mitoxantrone
Thioguanine; Gefitinib
Thioguanine; Mitoxantrone
Thioguanine; Cytarabine hydrochloride
Thioguanine; Cabazitaxel
Thioguanine; Floxuridine
Gefitinib; Mitoxantrone
Gefitinib; Cytarabine hydrochloride
Gefitinib; Cabazitaxel
Ruxolitinib; Vinorelbine tartrate
Teniposide; Cytarabine hydrochloride
Teniposide; Cabazitaxel
Vinorelbine tartrate; Axitinib
Vinorelbine tartrate; Vandetanib
Vinorelbine tartrate; Cytarabine hydrochloride
Crizotinib; Mitoxantrone
Vandetanib; Mitoxantrone
Vismodegib; Cabazitaxel
Megestrol acetate; Mitoxantrone
Tamoxifen citrate; Cabazitaxel
Quinacrine hydrochloride; Mitoxantrone
Vincristine sulfate; Cabazitaxel
Uracil mustard; Cytarabine hydrochloride
Mitoxantrone; Cytarabine hydrochloride
Mitoxantrone; Cabazitaxel
Mitoxantrone; Floxuridine
Cytarabine hydrochloride; Arsenic trioxide
Cytarabine hydrochloride; Cabazitaxel
Cytarabine hydrochloride; Everolimus
Cytarabine hydrochloride; Floxuridine
Cabazitaxel; Everolimus
Cabazitaxel; Floxuridine
Everolimus; Floxuridine

Figure 12: Heat maps of pairwise interaction strengths for drug combination data. The top 34 detected interactions are listed, among which 15 of them (marked in red) are verified in the DrugBank (Wishart et al., 2018).

K EMPIRICAL RESULT FOR SAMPLE EFFICIENCY

Figure 13 shows our proposed framework is sample efficient. The data are from the test suite (Table 4), and noise η is injected. Increasing the training sample size, both over-parameterized neural nets and our proposed ParaACE perform better. But ParaACE is less data-hungry since fewer training samples are needed for ParaACE to achieve a similar performance of the over-parameterized neural nets. The experiments show that ParaACE still performs well, even if with small training samples.

L DETAILS ON THE COMPARISON WITH KD, LTH, AND SYNFLOW

L.1 COMPARISON WITH KD

KD is widely used for multi-class classification problems, which extracts knowledge from the ”soft label” by controlling the temperature, but fewer KD methods are there for regression problems. Currently, people mainly use the Teacher bounded regression loss Chen et al. (2017) and the hint loss to deal with the objective detection problem. Passing unlabeled data to the Teacher model to produce pseudo labels can also help. In these ways, the Student is expected to have a similar performance to the Teacher.

We let an over-parameterized FC (10-5000-900-400-100-30-1) be the Teacher, and a ReLU network with architecture (10-70-70-70-70-30-1) be the Student. The Student is trained with 800 original data and 4000 pseudo data (labeled by the Teacher). The loss function we adopted is L_2 + Hint loss, where the Hint loss is used on the layer with 30 neurons.

It shows that KD trained with pseudo data and carefully designed loss function achieves similar performance as the Teacher, while ParaACE trained with original data and simple L_2 loss achieves significantly better performance.

L.2 COMPARISON WITH LTH

We used the pruning technique proposed in the lottery tickets hypothesis (LTH) paper(Frankle & Carbin, 2019). We pruned the fully connected neural network after every 20 epochs. Every time 20 % weights were pruned in each layer except for the last layer. We trained 500 epochs, and set the batch size to be 400. We report the best test performance in each round of pruning in Figure 14.

Figure 14 shows that, in most cases pruning the network properly can improve the model performance. But if the network was over-pruned, the performance may drop.

L.3 COMPARISON WITH SYNFLOW

We implemented the single shot SynFlow for regression tasks based on the official code repository https://github.com/ganguli-lab/Synaptic-Flow (Tanaka et al., 2020). The sparsity is set to $10^{-2.447}$ for synthetic datasets, thus the neural network is compressed 280 times. For real-world datasets, we compressed the model by 120 times. And we retrained the neural network after the single-shot prune (post-training). Note that we did not prune the biases here.

We set the hyper parameters for the post-training process as follows. The batch size is set to be 500, and the number of epochs is 100. We use Adam as the optimizer with $\text{lr} = 0.001, \text{betas} = (0.9, 0.99)$.

M ACCURATE APPROXIMATION FUNCTION BENEFITS INTERACTION DETECTION

Suppose the underlying ground truth function is $f(\mathbf{x})$, and our approximation function is $g_{\boldsymbol{\theta}}(\mathbf{x})$, we omit the $\boldsymbol{\theta}$ in the following. We show that the estimation error for Hessian $\left|E_{\mathbf{x}}\left[\frac{\partial^2 g(\mathbf{x})}{\partial x_i \partial x_j}\right] - E_{\mathbf{x}}\left[\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j}\right]\right|$ is bounded by $o(\epsilon)$, when $|f(\mathbf{x}) - g(\mathbf{x})| < \epsilon$ for all $\mathbf{x}$. This implies more accurate pre-trained model leads to better interaction detection accuracy. According to the following theorem, deep neural networks are good surrogate functions due to the universal function approximation capability (Hornik, 1991).

Theorem M.1. *Assume x_i are uniformly drawn from $[-1, 1]$ independently, if there exist an $\epsilon > 0$, such that $|f(\mathbf{x}) - g(\mathbf{x})| \leq \epsilon$ for all $\mathbf{x} \in \mathbb{R}^p$, then*

$$E_{\mathbf{x}}\left[\frac{\partial^2 g(\mathbf{x})}{\partial x_i \partial x_j}\right] - E_{\mathbf{x}}\left[\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j}\right] \leq \epsilon.$$

Proof.

$$\begin{aligned}
&E_{\mathbf{x}}\left[\frac{\partial^2 g(\mathbf{x})}{\partial x_i \partial x_j}\right] \\
=&\int \frac{\partial^2 g(\mathbf{x})}{\partial x_i \partial x_j} p(\mathbf{x}) d\mathbf{x} \\
=&\int \frac{\partial^2 g(\mathbf{x})}{\partial x_i \partial x_j} p(x_i, x_j) p(\mathbf{x}_{\setminus ij}) d\mathbf{x} \\
=&\int p(\mathbf{x}_{\setminus ij}) \int_{-1}^{1} \int_{-1}^{1} \frac{\partial^2 g(\mathbf{x})}{\partial x_i \partial x_j} p(x_i, x_j) dx_i dx_j d\mathbf{x}_{\setminus ij} \\
&\quad [\text{where } p(x_i, x_j) = 1/4] \\
=&\frac{1}{4} \int p(\mathbf{x}_{\setminus ij}) \int_{-1}^{1} \int_{-1}^{1} \frac{\partial^2 g(\mathbf{x})}{\partial x_i \partial x_j} dx_i dx_j d\mathbf{x}_{\setminus ij},
\end{aligned}$$

where

$$\begin{aligned}
&\int_{-1}^{1} \int_{-1}^{1} \frac{\partial^2 g(\mathbf{x})}{\partial x_i \partial x_j} dx_i dx_j \\
=&\int_{-1}^{1} \frac{\partial g(\mathbf{x}_{\setminus ij}, x_i = 1, x_j) - \partial g(\mathbf{x}_{\setminus ij}, x_i = -1, x_j)}{\partial x_j} dx_j \\
=&\quad g(\mathbf{x}_{\setminus ij}, x_i = 1, x_j = 1) - g(\mathbf{x}_{\setminus ij}, x_i = 1, x_j = -1) \\
&- (g(\mathbf{x}_{\setminus ij}, x_i = -1, x_j = 1) - g(\mathbf{x}_{\setminus ij}, x_i = -1, x_j = -1)).
\end{aligned}$$

We can derive $E_{\mathbf{x}}\left[\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j}\right]$ in a similar fashion. Thus,

$$\begin{aligned}
&E_{\mathbf{x}}\left[\frac{\partial^2 g(\mathbf{x})}{\partial x_i \partial x_j}\right] - E_{\mathbf{x}}\left[\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j}\right] \\
=&\frac{1}{4} E_{\mathbf{x}_{\setminus ij}}[g(\mathbf{x}_{\setminus ij}, x_i = 1, x_j = 1) - f(\mathbf{x}_{\setminus ij}, x_i = 1, x_j = 1) \\
&- g(\mathbf{x}_{\setminus ij}, x_i = -1, x_j = 1) + f(\mathbf{x}_{\setminus ij}, x_i = -1, x_j = 1) \\
&- g(\mathbf{x}_{\setminus ij}, x_i = 1, x_j = -1) + f(\mathbf{x}_{\setminus ij}, x_i = 1, x_j = -1) \\
&+ g(\mathbf{x}_{\setminus ij}, x_i = -1, x_j = -1) - f(\mathbf{x}_{\setminus ij}, x_i = -1, x_j = -1)] \\
\leq&\frac{1}{4} E_{\mathbf{x}_{\setminus ij}}[4\epsilon] \\
=&\epsilon.
\end{aligned}$$

□

Theorem M.2. *Assume* x_i *are uniformly drawn from* $[-1, 1]$ *independently, and further assume* $\left|\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j}\right|, \left|\frac{\partial^2 g(\mathbf{x})}{\partial x_i \partial x_j}\right| \leq M, \forall (x_i, x_j)$. *If there exist an* $\epsilon > 0$, *such that* $|f(\mathbf{x}) - g(\mathbf{x})| \leq \epsilon$ *for all* $\mathbf{x} \in \mathbb{R}^p$, *then*

$$E_{\mathbf{x}}\left[\left|\frac{\partial^2 g(\mathbf{x})}{\partial x_i \partial x_j}\right|^2\right] - E_{\mathbf{x}}\left[\left|\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j}\right|^2\right] \leq 2M\epsilon.$$

Proof.

$$\begin{aligned}
&E_{\mathbf{x}}\left[\left|\frac{\partial^2 g(\mathbf{x})}{\partial x_i \partial x_j}\right|^2\right] - E_{\mathbf{x}}\left[\left|\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j}\right|^2\right] \\
=&E_{\mathbf{x}}\left[\left|\frac{\partial^2 g(\mathbf{x})}{\partial x_i \partial x_j}\right|^2 - \left|\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j}\right|^2\right] \\
=&E_{\mathbf{x}}\left[\left(\frac{\partial^2 g(\mathbf{x})}{\partial x_i \partial x_j} - \frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j}\right)\left(\frac{\partial^2 g(\mathbf{x})}{\partial x_i \partial x_j} + \frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j}\right)\right] \\
\leq&\left|E_{\mathbf{x}}\left[\left(\frac{\partial^2 g(\mathbf{x})}{\partial x_i \partial x_j} - \frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j}\right)\right]\right| \cdot \left|E_{\mathbf{x}}\left[\left(\frac{\partial^2 g(\mathbf{x})}{\partial x_i \partial x_j} + \frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j}\right)\right]\right| \\
\leq&\epsilon \cdot 2M \quad \text{(Using Theorem M.1)}
\end{aligned}$$

□

Note that M can be interpreted as the magnitude of the strongest local interaction. For the functions with strong interactions, higher approximation quality is desired.

Table 9: Ablation study to show the effectiveness of interaction detection on the synthetic datasets in (Table 4). (The results were averaged over 5 folds.)

	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	average	CR
ParaACE (with random interaction)	0.026	0.042	0.035	0.034	0.060	0.032	0.017	0.025	0.044	0.031	**0.035**	**283**
ParaACE (with detected interaction)	0.025	0.035	0.031	0.030	0.038	0.025	0.016	0.019	0.023	0.021	**0.026**	**283**

N ABLATION STUDY FOR THE EFFECTIVENESS OF INTERACTION DETECTION

To show how much performance gain we are able to obtain from the interaction detection procedure. We conduct an experiment that input random pairwise interactions to ParaACE to compare with the one with detected interactions on synthetic datasets, see Table 9.

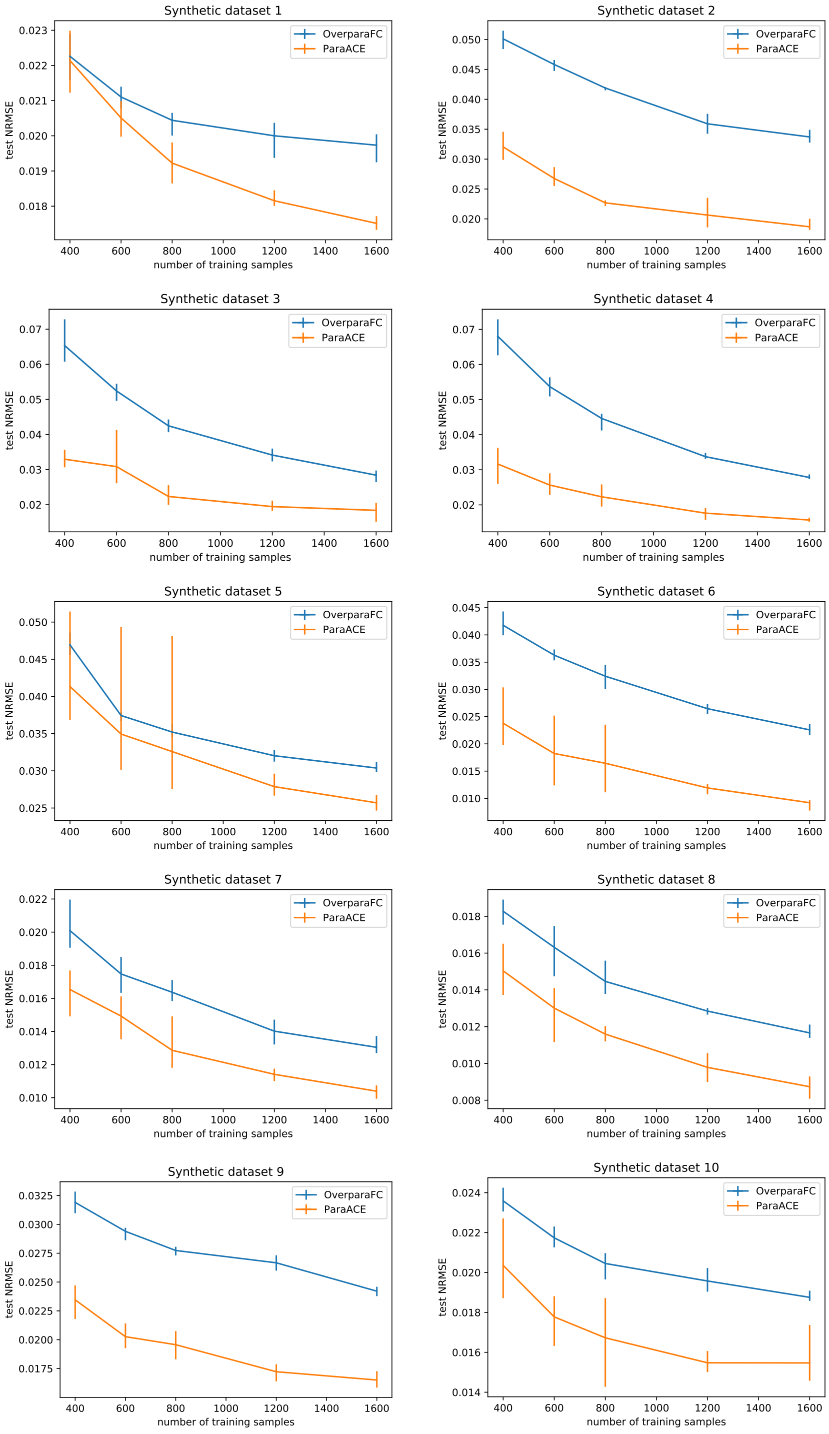


Figure 13: Performance comparison on synthetic datasets while reducing the number of training samples. For each dataset, we tried different training data 5 times in one experiment, and the performances (of ten experiments) were all tested on the same test set. The error bar shows the maximal and minimal test NRMSE in 5 folds.

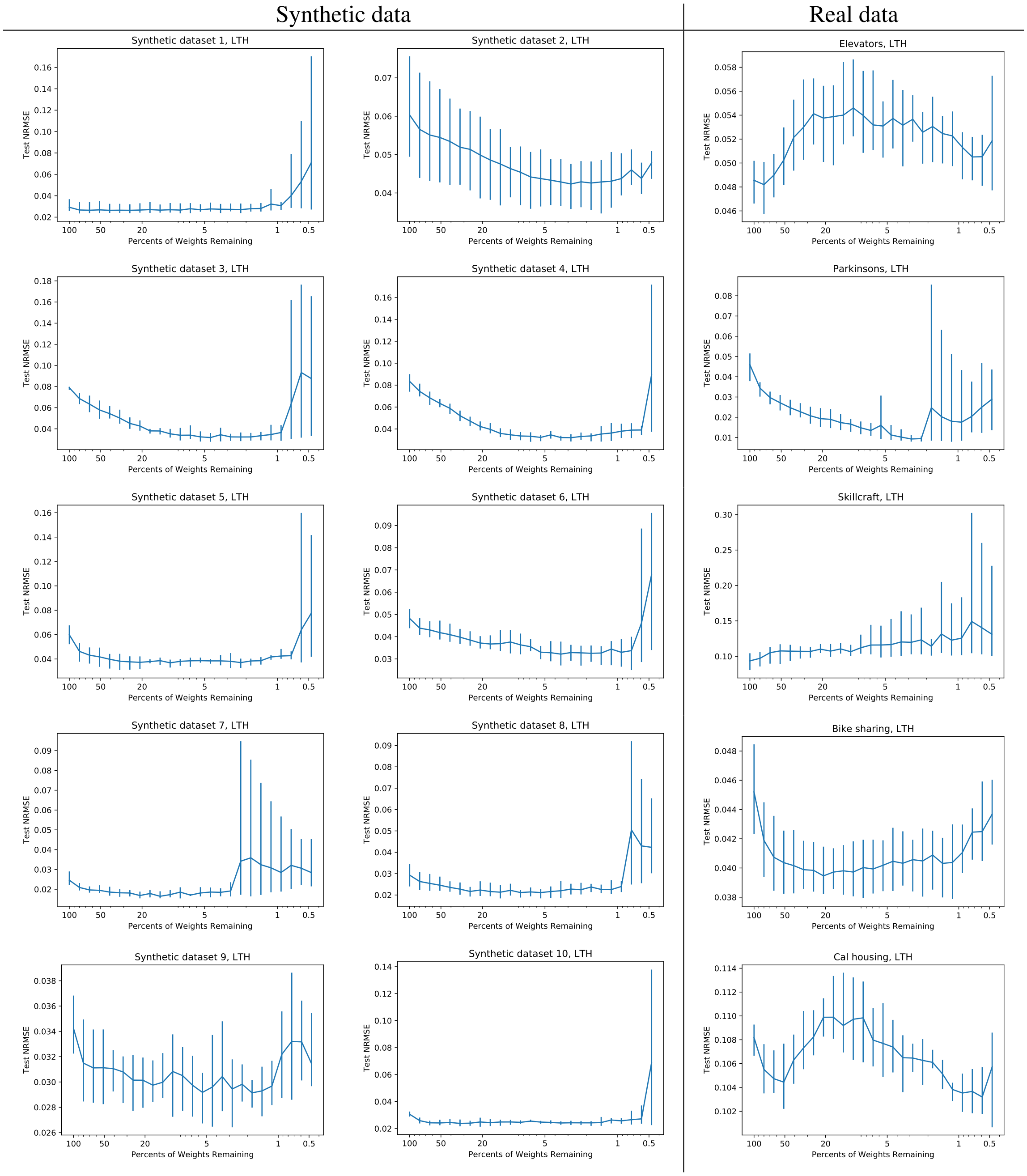


Figure 14: Performance comparison on both the synthetic and real datasets while reducing the percent of weights retained by the LTH. The error bar shows the maximal and minimal test NRMSE in 5 folds.