SPEX: Scaling Feature Interaction Explanations for LLMs

Justin Singh Kang[*1] **Landon Butler**[*1] **Abhineet Agarwal**[*2] **Yigit Efe Erginbas**[1]

Ramtin Pedarsani[3] **Kannan Ramchandran**[1] **Bin Yu**[1 2]

Abstract

Large language models (LLMs) have revolutionized machine learning due to their ability to capture complex interactions between input features. Popular post-hoc explanation methods like SHAP provide *marginal* feature attributions, while their extensions to interaction importances only scale to small input lengths (≈ 20). We propose *Spectral Explainer* (SPEX), a model-agnostic interaction attribution algorithm that efficiently scales to large input lengths (≈ 1000). SPEX exploits underlying natural sparsity among interactions—common in real-world data—and applies a sparse Fourier transform using a channel decoding algorithm to efficiently identify important interactions. We perform experiments across three difficult long-context datasets that require LLMs to utilize interactions between inputs to complete the task. For large inputs, SPEX outperforms marginal attribution methods by up to 20% in terms of faithfully reconstructing LLM outputs. Code at `https://github.com/basics-lab/spectral-explain`.

1 Introduction

Large language models (LLMs) perform remarkably well across many domains by modeling complex interactions among features. As LLMs are increasingly used in high-stakes applications, they require trustworthy explanations to aid in responsible decision-making. Post-hoc explainability approaches for LLMs fall into two categories: (i) methods like Shapley values Lundberg & Lee (2017) and LIME Ribeiro et al. (2016) compute *marginal* feature attribution but do not consider interactions. (ii) *Interaction indices* such as Faith-Shap Tsai et al. (2023) attribute interactions up to a given order d: For n input features, they compute attributions by considering all $O(n^d)$ interactions. This is infeasible for even small n and d. This motivates the central question of this paper:

Can interaction attributions be efficiently computed for a large input space n?

We answer this question affirmatively with *Spectral Explainer* (SPEX) which leverages information-theoretic tools to efficiently identify important interactions at LLM scale. The scale of SPEX ($n \approx 1000$) is enabled by the observation that LLM outputs are often driven by a small number of *sparse* interactions between inputs Tsui & Aghazadeh (2024); Ren et al. (2024a) (See Fig. 2). SPEX discovers important interactions by using a sparse Fourier transform to efficiently search for interactions which avoids the exhaustive enumeration required by existing approaches. Specifically, for s sparse Fourier transforms with interactions of maximum degree d, SPEX has computational complexity of $\tilde{O}(sdn)$. In contrast, popular interaction attribution approaches scale as $\Omega(n^d)$.

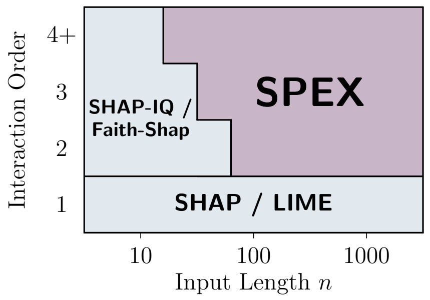


Figure 1: Marginal attribution scales to large n, but does not capture interactions. Interaction indices only work for small n. SPEX computes interactions *and* scales.

2 Methods

Value Function Let $\mathbf{x}$ be the input to the LLM where $\mathbf{x}$ consists of n input features, e.g., words. For $\mathbf{x} =$ "Her acting never fails to impress", $n = 6$. For input $\mathbf{x}$, let $f(\mathbf{x}_S) \in \mathbb{R}$ be the output of the LLM

Equal contibution [1]Department of Electrical Engineering and Computer Science, UC Berkeley [2]Department of Statistics, UC Berkeley [3]Department of Electrical and Computer Engineering, UC Santa Barbara.

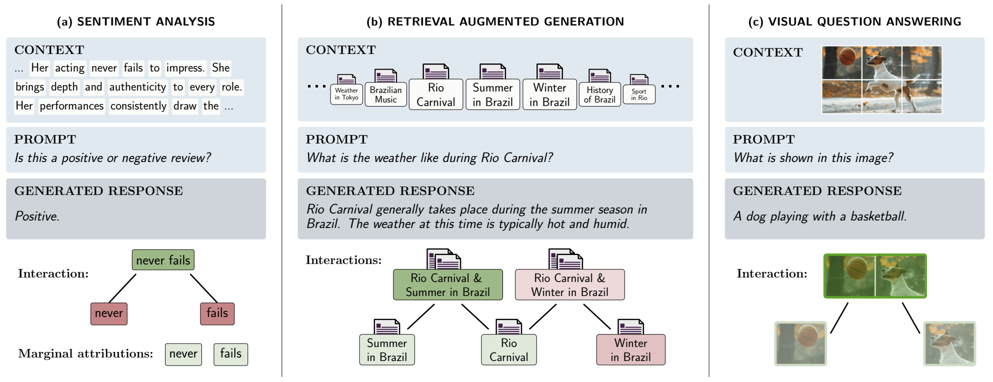

(a) SENTIMENT ANALYSIS
CONTEXT
... Her acting never fails to impress. She brings depth and authenticity to every role. Her performances consistently draw the ...
PROMPT
Is this a positive or negative review?
GENERATED RESPONSE
Positive.
Interaction:
never fails
never
fails
Marginal attributions:
never
fails
(b) RETRIEVAL AUGMENTED GENERATION
CONTEXT
Weather in Tokyo
Brazilian Music
Rio Carnival
Summer in Brazil
Winter in Brazil
History of Brazil
Sport in Rio
PROMPT
What is the weather like during Rio Carnival?
GENERATED RESPONSE
Rio Carnival generally takes place during the summer season in Brazil. The weather at this time is typically hot and humid.
Interactions:
Rio Carnival & Summer in Brazil
Rio Carnival & Winter in Brazil
Summer in Brazil
Rio Carnival
Winter in Brazil
(c) VISUAL QUESTION ANSWERING
CONTEXT
PROMPT
What is shown in this image?
GENERATED RESPONSE
A dog playing with a basketball.
Interaction:

Figure 2: (a) Sentiment analysis: SPEX identifies the double negative "never fails". Marginal approaches assign positive sentiment to "never" and "fails". (b) Retrieval augmented generation (RAG): SPEX finds the *combination* of documents the LLM used to answer the question while ignoring unimportant information. (c) Visual question answering: SPEX identifies interaction between image patches required to correctly summarize the image.

under masking pattern S. For $S \subseteq [n]$, we define $\mathbf{x}_S$ as a masked input where S denotes the coordinates in $\mathbf{x}$ we replace with the `[MASK]` token. For example, if $S = \{3\}$, then $\mathbf{x}_S$ is "Her acting `[MASK]` fails to impress". In sentiment analysis, (see Fig. 2(a)) $f(\mathbf{x}_S)$ is the logit of the positive class. For text generation tasks, we *scalarize* generated text by measuring the negative log-perplexity of the producing the **original** output for the unmasked input $\mathbf{x}$. We suppress dependence on $\mathbf{x}$ and write $f(\mathbf{x}_S)$ as $f(S)$.

Fourier Transform of Value Function Let $\mathbb{F}_2^n = \{0,1\}^n$, and addition between two elements in $\mathbb{F}_2$ as XOR. Since there are 2^n possible masks S, we equivalently write $f : \mathbb{F}_2^n \to \mathbb{R}$, where $f(S) = f(\mathbf{m})$ with $S = \{i : m_i = 1\}$. That is, $\mathbf{m} \in \mathbb{F}_2^n$ is a binary vector representing a *masking pattern*. If $m_i = 0$ we evaluate the model after masking the i^{th} input. The Fourier transform O'Donnell (2014) $F : \mathbb{F}_2^n \to \mathbb{R}$ of f is an *orthonormal* transform of the form:

$$f(\mathbf{m}) = \sum_{\mathbf{k} \in \mathbb{F}_2^n} (-1)^{\langle \mathbf{m}, \mathbf{k} \rangle} F(\mathbf{k}). \tag{1}$$

Sparsity f is *sparse* if $F(\mathbf{k}) \approx 0$ for most of the $\mathbf{k} \in \mathbb{F}_2^n$. Moreover, we call f *low degree*, if large $F(\mathbf{k})$ have small $|\mathbf{k}|$. Ren et al. (2024b); Kang et al. (2024) and experiments in Appendix C establish that deep-learning based value functions f are sparse and low degree.

Problem Statement Our goal is to compute an *approximate surrogate* $\hat{f}$. SPEX finds a small set of $\mathbf{k}$ with $|\mathbf{k}| \ll n$ denoted $\mathcal{K}$, and $\hat{F}(\mathbf{k})$ for each $\mathbf{k} \in \mathcal{K}$ such that

$$\hat{f}(\mathbf{m}) = \sum_{\mathbf{k} \in \mathcal{K}} (-1)^{\langle \mathbf{m}, \mathbf{k} \rangle} \hat{F}(\mathbf{k}). \tag{2}$$

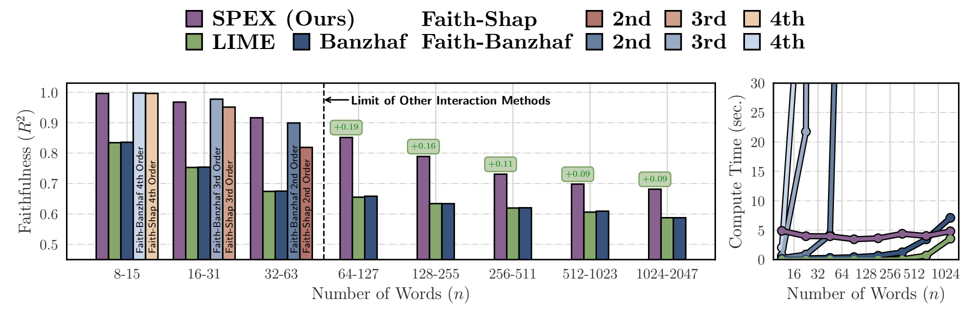
This goal is motivated by the Fourier sparsity that commonly occurs in real-world data and models. Popular interaction indices enumerate all $O(n^d)$ interactions, and determine $\mathcal{K}$ by formulating it as a LASSO problem solved via ℓ_1-penalized regression Tibshirani (1996). This is computationally infeasible (see Fig 3(a)), motivating the need to efficiently search the space of interactions.

SPEX The key to efficient search is realizing that we are not solving an *arbitrary* regression problem. Instead, we can design masking patterns $\mathbf{m}$ to exploit the sparse algebraic structure induced by the Fourier transform (1). We embed a BCH Code Lin & Costello (1999), a widely used algebraic channel code into the masking patterns to identify $\mathcal{K}$ efficiently. We now provide a brief overview of SPEX. A complete overview is provided in Appendix B. The high-level description consists of three parts:

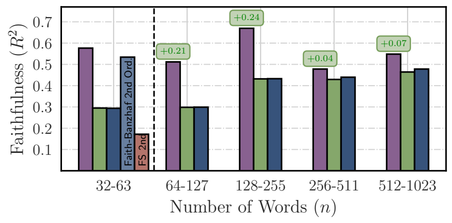
Step 1: Design a minimal set of masking patterns $\mathbf{m}$ and query $f(\mathbf{m})$ for each $\mathbf{m}$.

Step 2: Efficiently learn the surrogate function $\hat{f}$ via (2) from the set of collected samples $f(\mathbf{m})$.

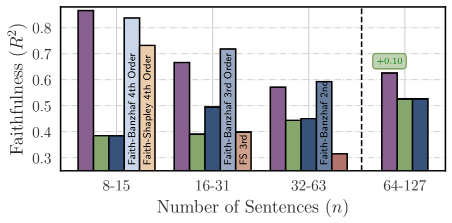
Step 3: Use $\hat{f}$ and its transform $\hat{F}$ to identify important interactions for attribution.



(a) *Sentiment*



(b) *DROP*



(c) *HotpotQA*

Figure 3: (a) SPEX is more faithful than baselines across datasets. High-order Faith-Banzhaf indices have competitive faithfulness, but rapidly increase in computational cost.

3 Experiments

Datasets We use three popular datasets that require the LLM to understand interactions between features. *Sentiment* primarily composed of the *Large Movie Review Dataset* Maas et al. (2011), *HotpotQA* Yang et al. (2018) a multiphop question answer dataset and *Discrete Reasoning Over Paragraphs* (DROP) Dua et al. (2019), a comprehension benchmark requiring discrete reasoning.

Models For *DROP* and *HotpotQA*, (generative tasks) we use `Llama-3.2-3B-Instruct` Grattafiori et al. (2024) with 8-bit quantization. For *Sentiment* (classification), we use the encoder-only fine-tuned `DistilBERT` model Sanh et al. (2019); Odabasi (2025).

Baselines We compare against popular marginal metrics LIME, SHAP, and the Banzhaf value. For interactions, we consider Faith-Shapley, Faith-Banzhaf, and the Shapley-Taylor Index. We compute all benchmarks where computationally feasible. That is, we always compute marginal attributions and interaction indices when n is sufficiently small. In figures, we only show the best performing baselines. Results and implementation details for all baselines can be found in Appendix C.

3.1 Metrics

We compare SPEX to other methods across a variety of well-established metrics to assess performance.

Faithfulness: To characterize how well the surrogate function $\hat{f}$ approximates the true function, we define *faithfulness* Zhang et al. (2023) as the R^2 of the learned surrogate function over 10,000 random *test* masks per-sample. We report the average R^2 across samples.

Top-r Removal: For each explanation method, we rank the top r features and measure the change in model output as we mask these features.

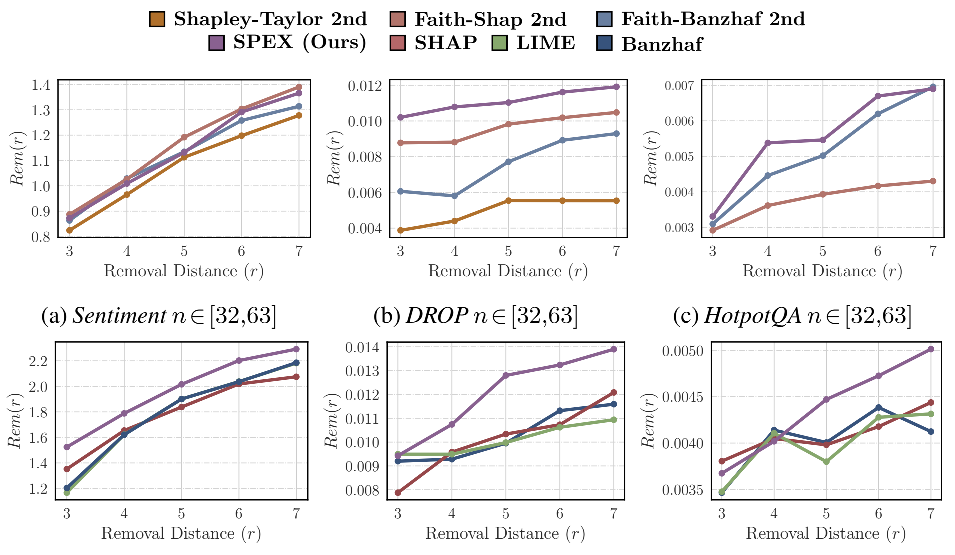
Recovery Rate@r: *HotpotQA* contains human-labeled annotations for the sentences required to correctly answer the question. For each interaction attribution method, we rank the top r interactions, and measure whether these interactions contain the sentences used to correctly answer the question.

3.2 Faithfulness and Runtime

Fig. 3 shows the faithfulness and run-time of SPEX compared to other methods.

Comparison to Interaction Indices We maintain competitive performance with best-performing indices which enumerate *all possible interactions*; SPEX does not. This difference is reflected in the runtimes of Fig. 3(a), which shows other interaction indices have run-time that explodes as n increases.

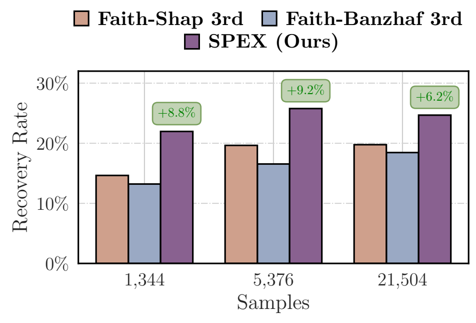
Comparison to Marginal Attributions For input lengths n too large to run interaction indices, SPEX is significantly more faithful than marginal attribution approaches across all three datasets.



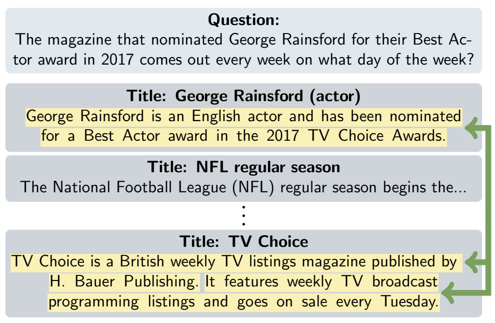
(a) *Sentiment* $n \in [32,63]$ (b) *DROP* $n \in [32,63]$ (c) *HotpotQA* $n \in [32,63]$

(d) *Sentiment* $n \in [64,127]$ (e) *DROP* $n \in [64,127]$ (f) *HotpotQA* $n \in [64,127]$

Figure 4: For $n \in [32,63]$, SPEX performs competitively with interaction indices on *Sentiment* and out-performs otherwise. For n too large for other interaction indices, we outperform marginal methods.



(a) Recovery rate@10 for *HotpotQA*



Question:
The magazine that nominated George Rainsford for their Best Actor award in 2017 comes out every week on what day of the week?

Title: George Rainsford (actor)
George Rainsford is an English actor and has been nominated for a Best Actor award in the 2017 TV Choice Awards.

Title: NFL regular season
The National Football League (NFL) regular season begins the...

⋮

Title: TV Choice
TV Choice is a British weekly TV listings magazine published by H. Bauer Publishing. It features weekly TV broadcast programming listings and goes on sale every Tuesday.

(b) Human-labeled interaction identified by SPEX.

Figure 5: (a) SPEX recovers more human-labeled features with fewer training masks. (b) For a long-context example ($n = 128$ sentences), SPEX identifies the three human-labeled sentences as the most important third order interaction and ignores irrelevant information.

Sample Efficiency Appendix C shows SPEX out-performs other approaches with fewer masks.

3.3 Removal

Fig. 4 plots the change in model output as we mask the top r features for different regimes of n. **Small** n : SPEX is competitive with other interaction indices for *Sentiment*, and out-performs them for *HotpotQA* and *DROP*. Performance of SPEXin this task is particularly notable since Shapley-based methods are designed to identify a small set of influential features. SPEX does not optimize for this metric, but instead learns the function $f(\cdot)$ over all possible 2^n masks. **Large** n : SPEX out-performs all marginal approaches, indicating the utility of considering interactions.

3.4 Recovery Rate of Human-Labeled Interactions

We compare the recovery rate for $r = 10$ of SPEX against third order Faith-Banzhaf and Faith-Shap interaction indices. Third order interaction indices are chosen since human labeled sentences contain at most three sentences, i.e., maximum degree $d = 3$. Results are shown in Fig. 5a as we vary number of training masks. SPEX not only has the highest recovery rate, it is also the most sample-efficient.

Example of Learned Interaction by SPEX Fig. 5b displays a long-context example (128 sentences) from *HotpotQA* whose answer is contained in the three highlighted sentences. SPEX identifies the three human-labeled sentences as the most important third order interaction while ignoring unimportant contextual information. Other third order methods are not computable at this length.

REFERENCES

Andisheh Amrollahi, Amir Zandieh, Michael Kapralov, and Andreas Krause. Efficiently learning fourier sparse set functions. *Advances in Neural Information Processing Systems*, 32, 2019.

John F Banzhaf III. Weighted voting doesn't work: A mathematical analysis. *Rutgers L. Rev.*, 19:317, 1964.

Alexander Binder, Grégoire Montavon, Sebastian Bach, Klaus-Robert Müller, and Wojciech Samek. Layer-wise relevance propagation for neural networks with local renormalization layers, 2016. URL `https://arxiv.org/abs/1604.00825`.

Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

Jianbo Chen, Le Song, Martin Wainwright, and Michael Jordan. Learning to explain: An information-theoretic perspective on model interpretation. In *International conference on machine learning*, pp. 883–892. PMLR, 2018.

Benjamin Cohen-Wang, Harshay Shah, Kristian Georgiev, and Aleksander Madry. Contextcite: Attributing model generation to context, 2024. URL `https://arxiv.org/abs/2409.00729`.

Dheeru Dua, Yizhong Wang, Swabha Dasigi, Sameer Singh, Matt Gardner, and Tom Kwiatkowski. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2368–2378, 2019.

James Enouen, Hootan Nakhost, Sayna Ebrahimi, Sercan O Arik, Yan Liu, and Tomas Pfister. TextGen-SHAP: Scalable post-hoc explanations in text generation with long documents. 2023. URL `https://arxiv.org/pdf/2312.01279`.

Yigit Efe Erginbas, Justin Kang, Amirali Aghazadeh, and Kannan Ramchandran. Efficiently computing sparse fourier transforms of q-ary functions. In *IEEE International Symposium on Information Theory (ISIT)*, pp. 513–518, 2023. doi: 10.1109/ISIT54713.2023.10206686.

Fabian Fumagalli, Maximilian Muschalik, Patrick Kolpaczki, Eyke Hüllermeier, and Barbara Eva Hammer. SHAP-IQ: Unified approximation of any-order shapley interactions. In *Conference on Neural Information Processing Systems*, 2023. URL `https://openreview.net/forum?id=IEMLNF4gK4`.

Michel Grabisch. k-order additive discrete fuzzy measures and their representation. *Fuzzy Sets and Systems*, 92(2):167–189, 1997. ISSN 0165-0114. doi: https://doi.org/10.1016/S0165-0114(97)00168-1. URL `https://www.sciencedirect.com/science/article/pii/S0165011497001681`. Fuzzy Measures and Integrals.

Michel Grabisch. Bases and Transforms of Set Functions. In Susanne Saminger-Platz and Radko Mesiar (eds.), *On Logical, Algebraic, and Probabilistic Aspects of Fuzzy Set Theory*, volume 336, pp. 215–231. Springer International Publishing, Cham, 2016. ISBN 978-3-319-28807-9 978-3-319-28808-6. doi: 10.1007/978-3-319-28808-6_13. URL `http://link.springer.com/10.1007/978-3-319-28808-6_13`. Series Title: Studies in Fuzziness and Soft Computing.

Michel Grabisch, Jean-Luc Marichal, and Marc Roubens. Equivalent representations of set functions. *Mathematics of Operations Research*, 25(2):157–178, 2000.

Aaron Grattafiori et al. The llama 3 herd of models, 2024. URL `https://arxiv.org/abs/2407.21783`.

John C Harsanyi. *A bargaining model for the cooperative n-person game*. PhD thesis, Department of Economics, Stanford University, Stanford, CA, USA, 1958.

Haitham Hassanieh, Piotr Indyk, Dina Katabi, and Eric Price. Simple and practical algorithm for sparse Fourier transform. In *SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1183–1194, 2012. doi: 10.1137/1.9781611973099.93. URL `https://epubs.siam.org/doi/abs/10.1137/1.9781611973099.93`.

Matt Hostetter. Galois: A performant NumPy extension for Galois fields. November 2020. URL https://github.com/mhostetter/galois.

Aliyah R. Hsu, Georgia Zhou, Yeshwanth Cherapanamjeri, Yaxuan Huang, Anobel Y. Odisho, Peter R. Carroll, and Bin Yu. Efficient automated circuit discovery in transformers using contextual decomposition, 2024. URL https://arxiv.org/abs/2407.00886.

Justin S Kang, Yigit E Erginbas, Landon Butler, Ramtin Pedarsani, and Kannan Ramchandran. Learning to understand: Identifying interactions via the mobius transform. *arXiv preprint arXiv:2402.02631*, 2024.

Patrick Kolpaczki, Maximilian Muschalik, Fabian Fumagalli, Barbara Hammer, and Eyke Hüllermeier. Svarm-iq: Efficient approximation of any-order shapley interactions through stratification. *arXiv preprint arXiv:2401.13371*, 2024.

Xiao Li, Joseph Kurata Bradley, Sameer Pawar, and Kannan Ramchandran. The SPRIGHT algorithm for robust sparse Hadamard Transforms. In *IEEE International Symposium on Information Theory (ISIT)*, pp. 1857–1861, 2014. doi: 10.1109/ISIT.2014.6875155.

Shu Lin and Daniel J Costello. *Error control coding*. Pearson, Upper Saddle River, NJ, 2 edition, December 1999.

Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 4768–4777, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P11-1015.

Arian Maleki. *Approximate message passing algorithms for compressed sensing*. PhD thesis, Stanford University, 2010.

Maximilian Muschalik, Hubert Baniecki, Fabian Fumagalli, Patrick Kolpaczki, Barbara Hammer, and Eyke Hüllermeier. shapiq: Shapley interactions for machine learning. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=knxGmi6SJi.

Katka Odabasi. DistilBERT Finetuned Sentiment. Accessed January, 2025. URL https://huggingface.co/lyrisha/distilbert-base-finetuned-sentiment.

Ryan O'Donnell. *Analysis of boolean functions*. Cambridge University Press, 2014.

Lucas Monteiro Paes, Dennis Wei, Hyo Jin Do, Hendrik Strobelt, Ronny Luss, Amit Dhurandhar, Manish Nagireddy, Karthikeyan Natesan Ramamurthy, Prasanna Sattigeri, Werner Geyer, et al. Multi-level explanations for generative language models. 2024. URL https://arxiv.org/pdf/2403.14459.

Sameer Pawar and Kannan Ramchandran. Computing a k-sparse n-length Discrete Fourier Transform using at most $4k$ samples and $O(klogk)$ complexity. In *IEEE International Symposium on Information Theory (ISIT)*, pp. 464–468, 2013. doi: 10.1109/ISIT.2013.6620269.

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

Jie Ren, Zhanpeng Zhou, Qirui Chen, and Quanshi Zhang. Can we faithfully represent absence states to compute shapley values on a DNN? In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=YV8tP7bW6Kt.

Qihan Ren, Jiayang Gao, Wen Shen, and Quanshi Zhang. Where we have arrived in proving the emergence of sparse interaction primitives in DNNs. In *International Conference on Learning Representations*, 2024a. URL `https://openreview.net/forum?id=3pWSL8My6B`.

Qihan Ren, Yang Xu, Junpeng Zhang, Yue Xin, Dongrui Liu, and Quanshi Zhang. Towards the dynamics of a DNN learning symbolic interactions. 2024b. URL `https://arxiv.org/pdf/2407.19198`.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " 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*, pp. 1135–1144, 2016.

M Roubens. Interaction between criteria and definition of weights in mcda problems. In *44th Meeting of the European Working Group "Multicriteria Aid for Decisions", Brussels, Belgium*, 1996.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108, 2019.

C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3): 379–423, 1948. doi: 10.1002/j.1538-7305.1948.tb01338.x.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

Peter Stobbe and Andreas Krause. Learning Fourier sparse set functions. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, Proceedings of Machine Learning Research, pp. 1125–1133, La Palma, Canary Islands, Apr 2012. URL `https://proceedings.mlr.press/v22/stobbe12.html`.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pp. 3319–3328. PMLR, 2017.

Mukund Sundararajan, Kedar Dhamdhere, and Ashish Agarwal. The Shapley Taylor interaction index. In *International Conference on Machine Learning*, pp. 9259–9268, Jul 2020. URL `https://proceedings.mlr.press/v119/sundararajan20a.html`.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1):267–288, 1996.

Che-Ping Tsai, Chih-Kuan Yeh, and Pradeep Ravikumar. Faith-shap: The faithful Shapley interaction index. *Journal of Machine Learning Research*, 24(94):1–42, 2023.

Darin Tsui and Amirali Aghazadeh. On recovering higher-order interactions from protein language models. In *ICLR 2024 Workshop on Generative and Experimental Perspectives for Biomolecular Design*, 2024. URL `https://openreview.net/forum?id=WfA5oWpYT4`.

Jiachen T. Wang and Ruoxi Jia. Data Banzhaf: A robust data valuation framework for machine learning. In *International Conference on Artificial Intelligence and Statistics*, volume 206, pp. 6388–6421, 25–27 Apr 2023. URL `https://proceedings.mlr.press/v206/wang23e.html`.

Sean Williams and James Huckle. Easy problems that llms get wrong. *arXiv preprint arXiv:2405.19616*, 2024.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*, 2018.

Yifan Zhang, Haowei He, Zhiquan Tan, and Yang Yuan. Trade-off between efficiency and consistency for removal-based explanations. *Advances in Neural Information Processing Systems*, 36:25627–25661, 2023.

A Related Work

Model-Agnostic Feature Attributions LIME Ribeiro et al. (2016), SHAP Lundberg & Lee (2017), and Banzhaf values Wang & Jia (2023) are popular model-agnostic feature attribution approaches. SHAP and Banzhaf use game-theoretic tools for feature attribution, while LIME fits a sparse linear model. Chen et al. (2018) utilize tools from information theory for feature attributions. Other methods Sundararajan et al. (2017); Binder et al. (2016) instead utilize internal model structure to derive feature attributions.

Interaction Indices Tsai et al. (2023) and Sundararajan et al. (2020) extend Shapley values to consider interactions. Fumagalli et al. (2023) provide a general framework towards interaction attribution but can only scale to at most $n \approx 20$ input features. Ren et al. (2023; 2024b) theoretically study sparse interactions, a widely observed phenomenon in practice. Kang et al. (2024) show that sparsity under the Möbius transform Harsanyi (1958) can be theoretically exploited for efficient interaction attribution. In practice, the proposed algorithm fails due to noise being amplified by the non-orthogonality of the Möbius basis. Our work utilizes the *orthonormal* Fourier transform, which improves robustness by preventing noise amplification. Hsu et al. (2024) apply tools from mechanistic interpretability such as circuit discovery for interaction attribution.

Feature Attribution in LLMs Enouen et al. (2023); Paes et al. (2024) propose hierarchical feature attribution for language models that first groups features (paragraphs) and then increase the feature space via a more fine-grained analysis (sentences or words/tokens). Cohen-Wang et al. (2024) provide marginal feature importances via LASSO. These works do not explicitly compute interaction attributions.

B Algorithm Details

B.1 Introduction

This section provides the algorithmic details behind SPEX. The algorithm is derived from the sparse Fourier (Hadamard) transformation described in Li et al. (2014). Many modifications have been made to improve the algorithm and make it suitable for use in this application, to the point where the original algorithm of Li et al. (2014) completely fails for all problems we consider in this paper. In this work we focus on applications and defer theoretical analysis to future work.

Relevant Literature on Sparse Transforms This work develops the literature on sparse Fourier transforms. The first of such works are Hassanieh et al. (2012); Stobbe & Krause (2012); Pawar & Ramchandran (2013). The most relevant literature is that of the sparse Boolean Fourier (Hadamard) transform Li et al. (2014); Amrollahi et al. (2019). Despite the promise of many of these algorithms, their application has remained relatively limited, being used in only a handful of prior applications. Our code base is forked from that of Erginbas et al. (2023). In this work we introduce a series of major optimizations which specifically target properties of explanation functions. By doing so, our algorithm is made significantly more practical and robust than any prior work.

Importance of the Fourier Transform The Fourier transform does more than just impart critical algebraic structure. The orthonormality of the Fourier transform means that small noisy variations in f remain small in the Fourier domain. In contrast, AND interactions, which operate under the non-orthogonal Möbius transform Kang et al. (2024), can amplify small noisy variations, which limits practicality. Fortunately, this is not problematic, as it is straightforward to generate AND interactions from the surrogate function $\hat{f}$. Many popular interaction indices have simple definitions in terms of F. Table 1 highlights some key relationships, and Appendix D provides a comprehensive list.

Shapley Value	Banzhaf Interaction Index	Möbius Coefficient
$\mathrm{SV}(i) = \sum_{S \ni i,\, \lvert S\rvert \text{ odd}} F(S)/\lvert S\rvert$	$I^{BII}(S) = (-2)^{\lvert S\rvert} F(S)$	$I^{M}(S) = (-2)^{\lvert S\rvert} \sum_{T \supseteq S} F(T)$

Table 1: Popular attribution scores in terms of Fourier coefficients

B.2 Directly Solving the LASSO

Before we proceed, we remark that in cases where n is not too large, and we expect the degree of nonzero $|\mathbf{k}| \leq d$ to be reasonably small, enumeration is actually not infeasible. In such cases, we can set up the LASSO problem directly:

$$\hat{F} = \underset{\tilde{F}}{\operatorname{argmin}} \sum_{\mathbf{m}} \left| f(\mathbf{m}) - \sum_{|\mathbf{k}| \leq d} \tilde{F}(\mathbf{k}) \right|^2 + \lambda \left\| \tilde{F} \right\|_1. \tag{3}$$

Note that this is distinct from the *Faith-Banzhaf* and *Faith-Shapley* solution methods because those perform regression over the AND, Möbius basis. We observe that the formulation above typically outperforms these other approaches in terms of faithfulness, likely due to the properties of the Fourier transform.

Most popular solvers use *coordinate descent* to solve (3), but there is a long line of research towards efficiently solving this problem. In our code, we also include an implementation of Approximate Message Passing (AMP) Maleki (2010), which can be much faster in many cases. Much like the final phase of SPEX, AMP is a low complexity message passing algorithm where messages are iteratively passed between factor nodes (observations) and variable nodes.

A more refined version of SPEX, would likely examine the parameters n and the maximum degree d and determine whether or not to directly solve the LASSO, or to apply the full SPEX, as we describe in the following sections.

B.3 Masking Pattern Design: Exploiting Structure - Quick Overview

We first highlight two important properties of Fourier transform related to masking design structure.

Aliasing (Coefficient Collapse) Property: For $b \leq n$ and $\mathbf{M} \in \mathbb{F}_2^{b \times n}$, let $u : \mathbb{F}_2^b \to \mathbb{R}$ denote a subsampled version of f. Then u has Fourier transform U:

$$u(\boldsymbol{\ell}) = f(\mathbf{M}^\top \boldsymbol{\ell}) \iff U(\mathbf{j}) = \sum_{\mathbf{M}\mathbf{k} = \mathbf{j}} F(\mathbf{k}). \tag{4}$$

Shift Property: For any function $f : \mathbb{F}_2^n \to \mathbb{R}$, if we shift the input by some vector $\mathbf{p} \in \mathbb{F}_2^n$, the Fourier transform changes as follows:

$$f_{\mathbf{p}}(\mathbf{m}) = f(\mathbf{m} + \mathbf{p}) \iff F_{\mathbf{p}}(\mathbf{k}) = (-1)^{\langle \mathbf{p}, \mathbf{k} \rangle} F(\mathbf{k}). \tag{5}$$

Designing Aliasing The aliasing property (4) dictates that when sampling according to $\mathbf{M} \in \mathbb{F}_2^{b \times n}$, all $F(\mathbf{k})$ with image $\mathbf{j} = \mathbf{M}\mathbf{k}$ are added together. If only *one* dominant $F(\mathbf{k})$ satisfies $\mathbf{M}\mathbf{k} = \mathbf{j}$, which can happen due to sparsity, we call it a *singleton*. We want $\mathbf{M}$ to maximize the number of singletons, since we ultimately use singletons to recover the dominant coefficients and estimate $\hat{F}$. SPEX uses $\mathbf{M}$ with elements chosen uniformly from $\mathbb{F}_2$. Such $\mathbf{M}$ has favorable properties regarding generating singletons.

Designing Shifts Once we create singletons, we need to identify them, extract the dominant index $\mathbf{k}$, and estimate $\hat{F}(\mathbf{k})$. The shift property (5) is critical for this task since the sign of the dominant $F(\mathbf{k})$ changes depending on $\langle \mathbf{p}, \mathbf{k} \rangle$. Thus, each time we apply a shift vector, we gather (potentially noisy) information about the dominant $\mathbf{k}$. Finding $\mathbf{k}$ and estimating $\hat{F}(\mathbf{k})$ can be modeled as communicating information over a noisy channel Shannon (1948), where the communication protocol is controlled by the shift vectors. We use the aforementioned BCH channel code, which requires only $\approx t\log(n)$ shifts to recover $\mathbf{k}$. The parameter t controls the robustness of the decoding procedure. Generally if the maximum degree is $|\mathbf{k}| = d$, we choose $t \geq d$. If $t - |\mathbf{k}| > 0$ we use the additional shifts to improve estimation of $\hat{F}(\mathbf{k})$. Since *most* of the time $|\mathbf{k}|$ is less than 5, we fix $t = 5$ for experiments in this paper.

Combined Masking Combining ideas from above, we construct $C = 3$ independently sampled $\mathbf{M}_c$ and p shifting vectors $\mathbf{p}_i$, which come from rows of a BCH parity matrix. Then, for $c \in [C]$ and $i \in [p]$, we entirely sample the function $u_{c,i}(\boldsymbol{\ell}) = f(\mathbf{M}_c^\top \boldsymbol{\ell} + \mathbf{p}_i)$. The total number of samples is $\approx C2^b t\log(n)$. We note that all model inference informed by our masking pattern can be conducted in parallel. The Fourier transform of each $u_{c,i}$, denoted $U_{c,i}$, is connected to the transform of the original function via

$$U_{c,i}(\mathbf{j}) = \sum_{\mathbf{k} : \mathbf{M}_c \mathbf{k} = \mathbf{j}} (-1)^{\langle \mathbf{p}_i, \mathbf{k} \rangle} F(\mathbf{k}). \tag{6}$$

B.4 Computing the Surrogate Function

Once we have the samples, we use an iterative message passing algorithm to estimate $\hat{F}(\mathbf{k})$ for a small (a-priori unknown) set of $\mathbf{k} \in \mathcal{K}$.

Bipartite Graph We construct a bipartite graph depicted in Fig. **??**. The observations $\mathbf{U}_c(\mathbf{j}) = (U_{c,0}(\mathbf{j}),\ldots,U_{c,p}(\mathbf{j}))$ are factor nodes, while the values $\hat{F}(\mathbf{k})$ correspond to variable nodes. $\hat{F}(\mathbf{k})$ is connected to $\mathbf{U}_c(\mathbf{j})$ if $\mathbf{M}_c\mathbf{k} = \mathbf{j}$.

Message Passing The messages from factor to variable are computed by attempting to decode a singleton via the Berlekamp-Massey algorithm. If a $\mathbf{k}$ is successfully decoded, $\mathbf{k}$ is added to $\mathcal{K}$ and $F(\mathbf{k})$ is estimated and sent to factor node $\hat{F}(\mathbf{k})$. The variable nodes send back the average of their received messages to all connected factor nodes. The factor nodes then update their estimates of $\hat{F}$, and attempt decoding again. The process repeats until convergence. Once complete the surrogate function is constructed from $\mathcal{K}$ and $\hat{F}(\mathbf{k})$ according to (2). Complete step-by-step details are in Appendix B, Algorithm B.6.1.

B.5 Masking Pattern Design and Model Inference - Full Overview

The first part of the algorithm is to determine which samples we collect. All steps of this part of the algorithm are outlined in Algorithm B.5. This is governed by two structures: the random linear codes $\mathbf{M}_c$ and the BCH parity matrix $\mathbf{P}$. Random linear codes have been well studied as central objects in error correction and cryptography. They have previously been considered for sparse transforms in Amrollahi et al. (2019). They are suitable for this application because they roughly uniformly hash $\mathbf{k}$ with low hamming weight.

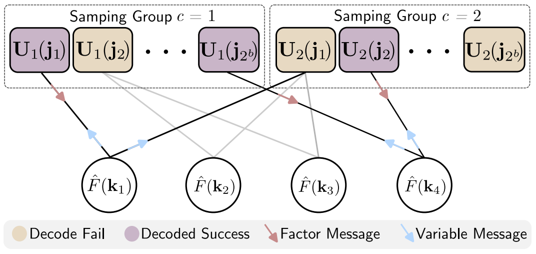


Figure 6: Each factor node $\mathbf{U}_C(\mathbf{j})$, attempts BCH decoding via the Berlekamp-Massey algorithm to recover $\mathbf{k}$. If the decoding succeeds, we estimate $F(\mathbf{k}_i)$ where $\mathbf{k}_i$ is the decoded singleton index. Variable nodes average received values, and send a message to connected factor nodes, which update according to the estimate, and iterate.

The use of the $\mathbf{P} \in \mathbb{F}_2^{p\times n}$, the parity matrix of a binary BCH code is novel. These codes are well studied for the applications in error correction Lin & Costello (1999), and they were once the preeminent form of error correction in digital communications. A primitive, narrow-sense BCH code is characterized by its length, denoted n_c, dimension, denoted k_c (which we want to be equal to our input dimension n) and its error correcting capability $t_c = 2d+1$, where d is the minimum distance of the code. For some integer $m > 3$ and $t_c < 2^{m-1}$, the parameters satisfy the following equations:

$$n_c = 2^m - 1 \tag{7}$$

$$p = n_c - k_c \leq mt. \tag{8}$$

Note that the above says we can bounds $p \leq t\lceil \log_2(n_c) \rceil$, and it is easy to solve for p given $n = k_c$ and t, however, explicitly bounding p in terms of n and t is difficult, so for the purpose of discussion, we simply write $p \approx t\log(n)$, since $n_c = p+n$, and we expect $n \gg p$ in nearly all cases of interest.

We use the software package `galois` Hostetter (2020) to construct a generator matrix, $\mathbf{G} \in \mathbb{F}_2^{n_c \times k_c}$ in systematic form:

$$\mathbf{G} = \begin{bmatrix} \mathbf{I}_{k_c\times k_c} \\ \mathbf{P} \end{bmatrix} \tag{9}$$

```
1: **Input:** Parameters $(n,t,b,C=3,\gamma=0.9)$, Query function $f(\cdot)$
2: **for** $j=1$ **to** $n$, $i=1$ **to** $b$, $c=1$ **to** $C$ **do**        ▷ Generate random linear code
3:     $X_{ij} \sim \mathrm{Bern}(0.5)$
4:     $[\mathbf{M}_c]_{i,j} \leftarrow X_{i,j}$
```

```
5:  end for
6:  $\mathrm{Code} \leftarrow \mathrm{BCH}(n_c = n_c, k_c \geq n, t_c = t)$   ▷ Systematic BCH code with dimension $n$ and correcting capacity $t$
7:  $p \leftarrow n_c - n$
8:  $\mathbf{P} \leftarrow \mathrm{Code}.\mathbf{P}$
9:  $\mathcal{P} \leftarrow \mathrm{rows}(\mathbf{P}) = [\mathbf{0}, \mathbf{p}_1, \ldots, \mathbf{p}_p]$
10: for all $\boldsymbol{\ell} \in \mathbb{F}_2^b$, $i \in \{0, \ldots, p\}$, $c \in \{1, \ldots, C\}$ do
11:     $u_{c,i}(\boldsymbol{\ell}) \leftarrow f\left(\mathbf{M}_c^\top \boldsymbol{\ell} + \mathcal{P}[i]\right)$   ▷ Query the model at masking patterns
12: end for
13: for all $i \in \{0, \ldots, p\}$, $c \in \{1, \ldots, C\}$ do
14:     $U_{c,i} \leftarrow \mathrm{FFT}(u_{c,i})$   ▷ Compute the Boolean Fourier transform of the collected samples
15: end for
16: $\mathbf{U}_c \leftarrow [U_{c,1}, \ldots, U_{c,p}]$
17: Output: Processed Samples $\mathbf{U}_c, U_{c,0}\ c = 1, \ldots, C$
```

Note that according to (9) $\mathbf{P} \in \mathbb{F}_2^{p \times k_c}$. In cases where $k_c > n$, we consider only the first n rows of $\mathbf{P}$. This is a process known as *shortening*. Our application of this BCH code in our application is rather unique. Instead of the typical use of a BCH code as *channel correction* code, we use it as a *joint source channel code*.

Let $\mathbf{p}_0 = \mathbf{0}$, and let $\mathbf{p}_i, i = 1, \ldots, p$ correspond to the rows of $\mathbf{P}$. We collect samples written as:

$$u_{c,i}(\boldsymbol{\ell}) \leftarrow f\left(\mathbf{M}_c^\top \boldsymbol{\ell} + \mathbf{p}_i\right) \forall \boldsymbol{\ell} \in \mathbb{F}_2^b, c = 1, \ldots, C, i = 0, \ldots, p. \tag{10}$$

Note that the total number of unique samples can be upper bounded by $C(p+1)2^b$. For large n this upper bound is nearly always very close to the true number of unique samples collected. After collecting each sample, we compute the boolean Fourier transform. The forward and inverse transforms as we consider in this work are defined below.

$$\text{Forward:} \quad F(\mathbf{k}) = \frac{1}{2^n} \sum_{\mathbf{m} \in \mathbb{F}_2^n} (-1)^{\langle \mathbf{k}, \mathbf{m} \rangle} f(\mathbf{m}) \qquad \text{Inverse:} \quad f(\mathbf{m}) = \sum_{\mathbf{k} \in \mathbb{F}_2^n} (-1)^{\langle \mathbf{m}, \mathbf{k} \rangle} F(\mathbf{k}), \tag{11}$$

When samples are collected according to (10), after applying the transform in (11), the transform of $u_{c,i}$ can be written as:

$$U_{c,i}(\mathbf{j}) = \sum_{\mathbf{k} : \mathbf{M}_c \mathbf{k} = \mathbf{j}} (-1)^{\langle \mathbf{p}_i, \mathbf{k} \rangle} F(\mathbf{k}). \tag{12}$$

To ease notation, we write $\mathbf{U}_c = [U_{c,1}, \ldots, U_{c,p}]^T$. Then we can write

$$\mathbf{U}_c(\mathbf{j}) = \sum_{\mathbf{k} : \mathbf{M}_c \mathbf{k} = \mathbf{j}} (-1)^{\mathbf{P}\mathbf{k}} F(\mathbf{k}), \tag{13}$$

where we have used the notation $(-1)^{\mathbf{P}\mathbf{k}} = [(-1)^{\langle \mathbf{p}_0, \mathbf{k} \rangle}, \ldots, (-1)^{\langle \mathbf{p}_p, \mathbf{k} \rangle}]^T$. We call the $(-1)^{\mathbf{P}\mathbf{k}}$ the *signature* of $\mathbf{k}$. This signature helps to identify the index of the largest interactions $\mathbf{k}$, and is central to the next part of the algorithm. Note that we also keep track of $U_{c,0}(\mathbf{j})$, which is equal to the unmodulated sum $U_{c,0}(\mathbf{j}) = \sum_{\mathbf{k} : \mathbf{M}_c \mathbf{k} = \mathbf{j}} F(\mathbf{k})$.

B.6 Message Passing for Fourier Transform Recovery

Using the samples (13), we aim to recover the largest Fourier coefficients $F(\mathbf{k})$. To recover these samples we apply a message passing algorithm, described in detail in Algorithm B.6.1. The factor nodes are comprised of the $C2^b$ vectors $\mathbf{U}_c(\mathbf{j})\ \forall \mathbf{j} \in \mathbb{F}_2^b$. Each of these factor nodes are connected to all values $\mathbf{k}$ that are comprise their sum, i.e., $\{\mathbf{k} \mid \mathbf{M}_c \mathbf{k} = \mathbf{j}\}$. Since the number of variable nodes is too great, we initialize the value of each variable node, which we call $\hat{F}(\mathbf{k})$ to zero implicitly. The values $\hat{F}(\mathbf{k})$ for each variable node indexed by $\mathbf{k}$ represent our estimate of the Fourier coefficients.

B.6.1 The message from factor to variable

Consider an arbitrary factor node $\mathbf{U}_c(\mathbf{j})$ initialized according to (13). We want to understand if there are any large terms $F(\mathbf{k})$ involved in the sum in (13). To do this, we can utilize the signature sequences $(-1)^{\mathbf{Pk}}$. If $\mathbf{U}_c(\mathbf{j})$ is strongly correlated with the signature sequence of a given $\mathbf{k}$, i.e., if $\left|\langle(-1)^{\mathbf{Pk}},\mathbf{U}_c(\mathbf{j})\rangle\right|$ is large, and $\mathbf{M}_c\mathbf{k}=\mathbf{j}$, from the perspective of $\mathbf{U}_c(\mathbf{j})$, it is likely that $F(\mathbf{k})$ is *large*. Searching through all $\mathbf{M}_c\mathbf{k}=\mathbf{j}$, which, for a full rank $\mathbf{M}_c$ contains 2^{n-b} different $\mathbf{k}$ is intractable, and likely to identify many spurious correlations. Instead, we rely on the structure of the BCH code from which $\mathbf{P}$ is derived to solve this problem.

BCH Hard Decoding The BCH decoding procedure is based on an idea known generally in signal processing as "treating interference as noise". For the purpose of explanation, assume that there is some $\mathbf{k}^*$ with large $F(\mathbf{k}^*)$, and all other $\mathbf{k}$ such that $\mathbf{M}_c\mathbf{k}=\mathbf{j}$ correspond to small $F(\mathbf{k})$. For brevity let $\mathcal{A}_c(\mathbf{j})=\{\mathbf{k}\,|\,\mathbf{M}_c\mathbf{k}=\mathbf{j}\}$. We can write:

$$\mathbf{U}_c(\mathbf{j})=F(\mathbf{k}^*)(-1)^{\mathbf{Pk}^*}+\sum_{\mathcal{A}_c(\mathbf{j})\setminus\mathbf{k}^*}(-1)^{\mathbf{Pk}}F(\mathbf{k}) \tag{14}$$

After we normalize with respect to $U_{c,0}(\mathbf{j})$ this yields:

$$\frac{\mathbf{U}_c(\mathbf{j})}{U_{c,0}(\mathbf{j})} = \left(\frac{1}{1+\sum_{\mathcal{A}_c(\mathbf{j})\setminus\mathbf{k}^*}F(\mathbf{k})/F(\mathbf{k}^*)}\right)(-1)^{\mathbf{Pk}^*}+\left(\frac{\sum_{\mathcal{A}_c(\mathbf{j})\setminus\mathbf{k}^*}(-1)^{\mathbf{Pk}}F(\mathbf{k})}{F(\mathbf{k}^*)+\sum_{\mathcal{A}_c(\mathbf{j})\setminus\mathbf{k}^*}F(\mathbf{k})}\right) \tag{15}$$

$$= A(\mathbf{j})(-1)^{\mathbf{Pk}^*}+\mathbf{w}(\mathbf{j}). \tag{16}$$

As we can see, the ratio (16) is a noise-corrupted version of the signature sequence of $\mathbf{k}^*$. To estimate $\mathbf{Pk}$ we apply a nearest-neighbor estimation rule outlined in Algorithm B.6.1. In words, if the ith coordinate of the vector (16) is closer to -1 we estimate that the corresponding element of $\mathbf{Pk}$ to be 1, conversely, if the ith coordinate is closer to 1 we estimate the corresponding entry to be 0. This process effectively converts the multiplicative noise A and additive noise $\mathbf{w}$ to a noise vector in $\mathbb{F}_2$. We can write this as $\mathbf{Pk}^*+\mathbf{n}$. According to the Lemma B.1 if the hamming weight $\mathbf{n}$ is not too large, we can recover $\mathbf{k}^*$.

Lemma B.1. *If* $|\mathbf{n}|+|\mathbf{k}^*|\leq t$*, where* $\mathbf{n}$ *is the additive noise in* $\mathbb{F}_2$ *induced by the noisy process in* (16) *and the estimation procedure in Algorithm B.6.1, then we can recover* $\mathbf{k}^*$.

Proof. Observe that the generator matrix of the BCH code is given by (9). Thus, there exists a codeword of the form

$$\mathbf{c}=\mathbf{G}\mathbf{k}^*=\begin{bmatrix}\mathbf{k}^*\\ \mathbf{Pk}^*\end{bmatrix} \tag{17}$$

Now construct the "received codeword" as in Algorithm B.6.1:

$$\mathbf{r}=\begin{bmatrix}\mathbf{0}\\ \mathbf{Pk}^*+\mathbf{n}\end{bmatrix} \tag{18}$$

Thus $|\mathbf{c}-\mathbf{r}|=|\mathbf{n}|+|\mathbf{k}^*|$. Since the BCH code was designed to be t error correcting, Decoding the code will recover $\mathbf{c}$, which contains $\mathbf{k}^*$. □

For decoding we use the implementation in the python package `galois` Hostetter (2020). It implements the standard procedure of the Berlekamp-Massey Algorithm followed by the Chien Search algorithm for BCH decoding.

```
1: Input: Observation $\mathbf{U}_c(\mathbf{j})$, Decoding function $\mathrm{Dec}(\cdot)$
2: $r_i \leftarrow 0\; i=1\ldots,n$
3: for all $i\in n+1,\ldots,n+p$ do
4:     $r_i \leftarrow \mathbb{1}\left\{\frac{U_{c,i}(\mathbf{j})}{U_{c,0}(\mathbf{j})}<0\right\}$
5: end for
6: dec, $\hat{\mathbf{k}} \leftarrow \mathrm{Dec}(\mathbf{r})$
7: Output: dec, $\hat{\mathbf{k}}$
```

BCH Soft Decoding In practice the conversion of the real-valued noisy observations (16) to noisy elements in $\mathbb{F}_2$ is a process that destroys valuable information. In coding theory, this is known as *hard input* decoding, which is typically suboptimal. For example, certain coordinates will have values $\frac{U_{c,i}(\mathbf{j})}{U_{c,0}(\mathbf{j})}\approx 0$. For such coordinates, we have low confidence about the corresponding value of

$(-1)^{\langle \mathbf{p}_i, \mathbf{k}^* \rangle}$, since it is equally close to $+1$ and -1. This uncertainty information is lost in the process of producing a hard input. With this so-called *soft information* it is possible to recover $\mathbf{k}^*$ even in cases where there are more than t errors in the hard decoding case. We use a simple soft decoding algorithm for BCH decoding known as a chase decoder. The main idea behind a chase decoder is to perform hard decoding on the d_{chase} most likely hard inputs, and return the decoder output of the most likely hard input that successfully decoded. In practical setting like the ones we consider in this work, we don't have an understanding of the noise in (16). A practical heuristic is to simply look at the *margin* of estimation. In other words, if $\left|\frac{U_{c,i}(\mathbf{j})}{U_{c,0}(\mathbf{j})}\right|$ is large, we assume it has high confidence, while if it is small, we assume the confidence is low. Interestingly, if we assume $A(\mathbf{j}) = 1$ and $\mathbf{w}(\mathbf{j}) \sim \mathcal{N}(0, \sigma^2)$ in (16), then the ratio corresponds exactly to the logarithm of the likelihood ratio (LLR) $\log\left(\frac{\Pr(\langle \mathbf{p}_i, \mathbf{k}^* \rangle = 0)}{\Pr(\langle \mathbf{p}_i, \mathbf{k}^* \rangle = 1)}\right)$. For the purposes of soft decoding we interpret these ratios as LLRs. Pseudocode can be found in Algorithm B.6.1.

Remark: BCH soft decoding is a well-studied topic with a vast literature. Though we put significant effort into building a strong implementation of SPEX*, we have used the simple Chase Decoder (described in Algorithm B.6.1 below) as a soft decoder. The computational complexity of Chase Decoding scales as $2^{d_{chase}}$, but other methods exist with much lower computational complexity and comparable performance.*

```
 1: Input: Observation $\mathbf{U}_c(\mathbf{j})$, Decoding function $\mathrm{Dec}(\cdot)$, Chase depth $d_{\text{chase}}$.
 2: $r_i \leftarrow 0\, i = 1 \ldots, n$
 3: $\mathcal{R} \leftarrow d_{\text{chase}}$ most likely hard inputs        ▷ Can be computed efficiently via dynamic programming
 4: dec $\leftarrow False$
 5: $j \leftarrow 0$
 6: while $dec$ is $False$ and $j \le d_{\text{chase}}$ do
 7:     $\mathbf{r}_{(n+1):(n+p)} \leftarrow \mathcal{R}[j]$
 8:     $j \leftarrow j + 1$
 9:     dec, $\hat{\mathbf{k}} \leftarrow \mathrm{Dec}(\mathbf{r})$
10: end while
11: Output: dec, $\hat{\mathbf{k}}$
```

If we successfully decode some $\mathbf{k}$ from the BCH decoding process via the bin $\mathbf{U}_c(\mathbf{j})$, we construct a message to the corresponding variable node. Before we do this, we verify that the $\mathbf{k}$ term satisfies $\mathbf{M}_c\mathbf{k} = \mathbf{j}$. This acts as a final check to increase our confidence in the output of $\mathbf{k}$. The message we construct is of the following form:

$$\mu_{(c,\mathbf{j}) \to \mathbf{k}} = \langle (-1)^{\mathbf{Pk}}, \mathbf{U}_c(\mathbf{j}) \rangle / p \tag{19}$$

To understand the structure of this message. This message can be seen as an estimate of the Fourier coefficient. Let's assume we are computing this message for some $\mathbf{k}^*$:

$$\mu_{(c,\mathbf{j}) \to \mathbf{k}^*} = F(\mathbf{k}^*) + \underbrace{\sum_{\mathcal{A}(\mathbf{j}) \setminus \mathbf{k}^*} \frac{1}{p} \langle (-1)^{\mathbf{Pk}}, (-1)^{\mathbf{Pk}^*} \rangle F(\mathbf{k})}_{\text{typically small}} \tag{20}$$

The inner product serves to reduce the noise from the other coefficients in the sum.

```
 1: Input: Processed Samples $\mathbf{U}_c, c = 1, \ldots, C$
 2: $\mathcal{S} = \{(c, \mathbf{j}) : \mathbf{j} \in \mathbb{F}_2^b, c \in \{1, \ldots, C\}\}$        ▷ Nodes to process
 3: $\hat{F}[\mathbf{k}] \leftarrow 0\, \forall \mathbf{k}$
 4: $\mathcal{K} \leftarrow \emptyset$
 5: while $|\mathcal{S}| > 0$ do        ▷ Outer Message Passing Loop
 6:     $\mathcal{S}_{\text{sub}} \leftarrow \emptyset$
 7:     $\mathcal{K}_{\text{sub}} \leftarrow \emptyset$
 8:     for $(c, \mathbf{j}) \in \mathcal{S}$ do
 9:         dec, $\mathbf{k} \leftarrow \mathrm{DecBCH}(\mathbf{U}_c(\mathbf{j}))$        ▷ Process Factor Node
10:         if dec then
11:             corr $\leftarrow \frac{\langle (-1)^{\mathbf{Pk}}, \mathbf{U}_c(\mathbf{j}) \rangle}{\|\mathbf{U}_c(\mathbf{j})\|^2}$
12:         else
13:             corr $\leftarrow 0$
14:         end if
```

```
15:         if corr $> \gamma$ then                                                ▷ Interaction identified
16:             $\mathcal{S}_{\text{sub}} \leftarrow \mathcal{S}_{\text{sub}} \cup \{(\mathbf{k}, c, \mathbf{j})\}$
17:             $\mathcal{K}_{\text{sub}} \leftarrow \mathcal{K}_{\text{sub}} \cup \{\mathbf{k}\}$
18:         else
19:             $\mathcal{S} \leftarrow \mathcal{S} \setminus \{(c, \mathbf{j})\}$                          ▷ Cannot extract interaction
20:         end if
21:     end for
22:     for $\mathbf{k} \in \mathcal{K}_{\text{sub}}$ do
23:         $\mathcal{S}_{\mathbf{k}} \leftarrow \{(\mathbf{k}', c', \mathbf{j}') \mid (\mathbf{k}', c', \mathbf{j}') \in \mathcal{S}_{\text{sub}}, \mathbf{k}' = \mathbf{k}\}$
24:         $\mu_{(c,\mathbf{j}) \to \mathbf{k}} \leftarrow \langle (-1)^{\mathbf{Pk}}, \mathbf{U}_c(\mathbf{j}) \rangle / p$
25:         $\mu_{\mathbf{k} \to \text{all}} \leftarrow \sum_{(\mathbf{k}, c, \mathbf{j}) \in \mathcal{S}_{\mathbf{k}}} \mu_{(c,\mathbf{j}) \to \mathbf{k}}$
26:         $\hat{F}(\mathbf{k}) \leftarrow \hat{F}(\mathbf{k}) + \mu_{\mathbf{k} \to \text{all}}$                    ▷ Update variable node
27:         for $c \in \{1, \ldots, C\}$ do
28:             $\mathbf{U}_c(\mathbf{M}_c \mathbf{k}) \leftarrow \mathbf{U}_c(\mathbf{M}_c \mathbf{k}) - \mu_{\mathbf{k} \to \text{all}} \cdot (-1)^{\mathbf{Pk}}$        ▷ Update factor node
29:             $\mathcal{S} \leftarrow \mathcal{S} \cup \{(c, \mathbf{M}_c \mathbf{k})\}$
30:         end for
31:     end for
32:     $\mathcal{K} \leftarrow \mathcal{K} \cup \mathcal{K}_{\text{sub}}$
33: end while
34: Output: $\left\{ \left( \mathbf{k}, \hat{F}(\mathbf{k}) \right) \mid \mathbf{k} \in \mathcal{K} \right\}$, interactions, and scalar values corresponding to interactions.
```

B.6.2 The message from variable to factor

The message from factor to variable is comparatively simple. The variable node takes the average of all the messages it receives, adding the result to its state, and then sends that average back to all connected factor nodes. These factor nodes then subtract this value from their state and then the process repeats.

B.7 Computational Complexity

Generating masking patterns $\mathbf{m}$ Constructing each masking pattern requires $n2^b$ for each $\mathbf{M}_c$. The algorithm for computing it efficiently involves a gray iteratively adding to an n bit vector and keeping track of output in a Gray code. Doing this for all C, and then adding all p additional shifting vectors makes the cost $O(Cpn2^b)$.

Taking FFT For each $u_{c,i}$ we take the Fast Fourier transform in $b2^b$ time, with a total of $O(Cpb2^b)$. This is domiated by the previous complexity since, $b \le n$

Message passing One round of BCH hard decoding is $O(n_c t + t^2)$. For soft decoding, this cost is multiplied by $2^{d_{\text{chase}}}$, which we is a constant. Computing correlation vector is $O(np)$, dominated by the computation of $\mathbf{Pk}$. In the worst case we must do this for all $C2^b$ vectors $\mathbf{U}_c(\mathbf{j})$. We also check that $\mathbf{Mk} = \mathbf{j}$ before sending the message, which costs $O(nb)$ Thus processing all the factor nodes costs $O(C2^b(n_c t + t^2 + n(p+b)))$. The number of active (with messages to send) variable nodes is at most $C2^b$, and computing their factors is at most C. Thus computing factor messages are at most $C^2 2^b$ messages. Finally factor nodes are updated with at most $C2^b$ variable messages sending messages to at most C factor nodes each, each with cost $O(np)$. Thus, the total cost of processing all variable nodes is $O(C^2 2^b + C^2 2^b np)$. The total cost of message is dominated by processing the factors.

The total complexity is then $O(2^b(n_c t + t^2 + n(p+b))$. Note that $p = n_c - n = t\log(n_c)$. Due to the structure of the code and the relationship between n, p and n_c, one could stop here, and it would be best to if we want to consider very large t. For the purposes of exposition, we will assume that $t \ll n$, which implies $n > p$, and thus $p \approx t\log(n)$. In this case, we can write:

$$\text{Complexity} = O(2^b(nt\log(n) + nb)) \tag{21}$$

To arrive at the stated equation in Section 1, we take $2^b = O(s)$. Under the low degree assumption, we have $s = O(d\log(n))$. Then assuming we take $t = O(d)$, we arrive at a complexity of $O(sdn\log(n))$.

C Experiment Details

C.1 Implementation Details

Experiments are run on a server using Nvidia L40S GPUs and A100 GPUs. When splitting text into words or sentences, we make use of the default word and sentence tokenizer from `nltk` Bird et al. (2009). To fit regressions, we use the `scikit-learn` Pedregosa et al. (2011) implementations of `LinearRegression` and `RidgeCV`.

C.2 Datasets and Models

C.2.1 Sentiment Analysis

152 movie reviews were used from the *Large Movie Review Dataset* Maas et al. (2011), supplemented with 8 movie reviews from the *Stanford Sentiment Treebank* dataset Socher et al. (2013). These 160 reviews were categorized using their word counts into 8 groups ([8-15, 16-32, . . . , 1024-2047]), with 20 reviews in each group.

To measure the sentiment of each movie review, we utilize a `DistilBERT` model Sanh et al. (2019) fine-tuned for sentiment analysis Odabasi (2025). When masking, we replace the word with the `[UNK]` token. We construct an value function over the output logit associated with the positive class.

C.2.2 HotpotQA

We consider 100 examples from the *HotpotQA*Yang et al. (2018) dataset. These examples were categorized using the number of sentences into four groups ([8-15, 16-32, 32-64, 64-127]). We use a `Llama-3.2-3B-Instruct` model with 8-bit quantization. When masking, we replace with the `[UNK]` token, and measure the log-perplexity of generating the original output. Since *HotpotQA* is a multi-document dataset, we use the following prompt format.

Title: {title_1}
Content: {document_1}
. . .
Title: {title_m}
Content: {document_m}

Query: {question}. Keep your answers as short as possible.

C.2.3 DROP

We consider 100 examples from the *DROP* Yang et al. (2018) dataset. These examples were categorized using the number of words into six groups ([8-15, 16-32, 32-64, 64-127, 128-256, 512-1024]). We use the same model as *HotpotQA* and mask in a similar fashion. We use the following prompt format.

Context: {context}
Query: {question}. Keep your answers as short as possible.

C.2.4 Trolley Problem

The simplified trolley problem was inspired by the one provided in Williams & Huckle (2024). When masking, the `[UNK]` token was used to replace words. The following prompt was given to `gpt-4o-mini-2024-07-18`:

System: Answer with the one word True or False only. Any other answer will be marked incorrect.
User: {Masked Input} True or False: You should not pull the lever.

A value function was created by finding the difference between the model's logprob associated with the "True" token minus the logprob of the "False" token.

C.2.5 Visual Question Answering

The base image was partitioned into a 6×8 grid. To mask, Gaussian blur was applied to the masked cells. The masked image was input into `LLaVA-NeXT-Mistral-7B`, a large multimodal model, with the following prompt:

Context: $\{$masked image$\}$
Query: What is shown in this image?

The original output to the unmasked image is "A dog playing with a basketball." Using the masked images, we build a value function that measures the probability of generating the original output sequence (log probability).

C.3 Baselines

The following marginal feature attribution baselines were run:

1. *LIME*: LIME (Local Interpretable Model-agnostic Explanations) Ribeiro et al. (2016) uses LASSO to build a sparse linear approximation of the value function. The approximation is weighted to be *local*, using an exponential kernel to fit the function better closer to the original input (less maskings).
2. *SHAP*: Implemented using KernelSHAP Lundberg & Lee (2017), SHAP interprets the value function as a cooperative game and attributes credit to each of the features according to the Shapley value. KernelSHAP approximates the Shapley values of this game through solving a weighted least squares problem, where the weighting function is informed by the Shapley kernel, promoting samples where very either very few or most inputs are masked.
3. *Banzhaf*: Similar to Shapley values, Banzhaf values Banzhaf III (1964) represent another credit attribution concept from cooperative game theory. We compute the Banzhaf values by fitting a ridge regression to uniformly drawn samples, selecting the regularization parameter through cross-validation.

Furthermore, we compared against the following interaction attribution methods:

4. *Faith-Banzhaf*: The Faith-Banzhaf Interaction Index Tsai et al. (2023), up to degree t, provides the most faithful t^{th} order polynomial approximation of the value function under a uniform kernel. We obtain this approximation using cross-validated ridge regression on uniformly drawn samples.
5. *Faith-Shap*: Similarly, the Faith-Shapley Interaction Index Tsai et al. (2023), up to degree t, provides the most faithful t order polynomial approximation of the value function under a Shapley kernel. As described in Tsai et al. (2023), the indices can be estimated through solving a weighted least squares problem. We use the implementation provided in SHAP-IQ Muschalik et al. (2024).
6. *Shapley-Taylor*: The Shapley-Taylor Interaction Index Sundararajan et al. (2020), up to degree t, provides another interaction definition based on the Taylor Series of the Möbius transform of the value function. To estimate the interaction indices, we leverage the sample-efficient estimator SVARM-IQ Kolpaczki et al. (2024), as implemented in SHAP-IQ Muschalik et al. (2024).

C.4 Sample Complexity

The total number of samples needed for SPEX is $\approx C 2^b t \log(n)$. We fix $C = 3$ and $t = 5$. The table below presents the number of samples used in our experiments for various choices of sparsity parameter b and different input sizes n:

	Number of Inputs (n)						
Sparsity Parameter (b)	8–11	12–36	37–92	93–215	216–466	467–973	974–1992
4	1,008	1,344	1,728	1,968	2,208	2,448	2,688
6	4,032	5,376	6,912	7,872	8,832	9,792	10,752
8	16,128	21,504	27,648	31,488	35,328	39,168	43,008

Table 2: Number of samples needed for each b and n.

	n	Avg. Sparsity	Avg. Sparsity Ratio	SPEX	LIME	Banzhaf	Faith-Banzhaf			SHAP	Faith-Shap			Shapley-Taylor		
							2nd	3rd	4th		2nd	3rd	4th	2nd	3rd	4th
Sentiment	8-15	369.9	3.70×10^{-1}	1.00	0.83	0.84	0.96	0.99	1.00	0.62	0.93	0.98	1.00	0.72	0.81	0.92
	16-31	208.7	2.31×10^{-4}	0.97	0.75	0.75	0.93	0.98		0.20	0.89	0.95		0.46	-710.85	
	32-63	149.7	3.14×10^{-9}	0.93	0.67	0.68	0.90			-0.25	0.82			-0.30		
	64-127	118.4	2.19×10^{-19}	0.87	0.66	0.66				-1.17						
	128-255	113.5	1.40×10^{-38}	0.82	0.63	0.63				-3.94						
	256-511	100.4	5.48×10^{-78}	0.76	0.62	0.62				-6.77						
	512-1013	86.1	2.02×10^{-155}	0.73	0.61	0.61				-4.78						
	1024-2047	81.7	3.78×10^{-302}	0.71	0.59	0.59				-17.31						
DROP	32-63	77.1	1.97×10^{-14}	0.58	0.29	0.29	0.53			-0.07	0.17			N/A		
	64-127	56.3	2.59×10^{-24}	0.51	0.30	0.30				0.02						
	128-255	58.7	2.31×10^{-38}	0.67	0.43	0.43				0.14						
	256-511	56.7	1.29×10^{-76}	0.48	0.43	0.44				-5.29						
	512-1023	36.3	9.63×10^{-155}	0.55	0.46	0.48				-0.23						
HotpotQA	8-15	108.3	1.10×10^{-2}	0.87	0.38	0.38	0.63	0.77	0.84	-1.09	-19.14	-2.33	0.73	N/A	N/A	N/A
	16-31	96.4	3.30×10^{-4}	0.67	0.39	0.49	0.66	0.72		0.23	-2.28	0.40		N/A	N/A	
	32-63	79.4	5.86×10^{-9}	0.57	0.44	0.45	0.59			0.13	0.32			N/A		
	64-127	73.0	1.02×10^{-18}	0.63	0.53	0.53				-0.31						

Table 3: Faithfulness across all baseline methods for fixed $b = 8$. The average recovered sparsity and the average ratio between sparsity and total possible interactions is also reported.

C.5 Additional Results

C.5.1 Faithfulness

Faithfulness for a fixed sparsity parameter $b = 8$: We first measure the faithfulness by scaling the number of samples logarithmically with n. The exact number of samples used can be found in Table 2.

In Table 3, we showcase the average faithfulness of every runable method across every group of examples for the *Sentiment*, *DROP*, and *HotpotQA* datasets. Among marginal attribution methods, LIME and Banzhaf achieve the best faithfulness. SHAP's faithfulness worsens as n grows, though this is unsurprising, as Shapley values are only efficient (intended to sum to the unmasked output), not faithful.

Comparing to interaction methods, SPEX is comparable to the highest order Faith-Banzhaf that can feasible be run at every size of n. However, due to poor computation complexity scaling of this, and other interaction methods, these methods are only able to be used for small n. In particular, we found Shapley-Taylor difficult to run for the *DROP* and *HotpotQA* tasks, unable to finish within thirty minutes.

We also report the average sparsity and the average sparsity ratio (sparsity over total interactions) discovered by SPEX for each of the groups. For *Sentiment*, once reaching the $n \in [128-255]$ group, the average sparsity is found to be less than the number of inputs! Yet, SPEX is still able to achieve high faithfulness and significantly outperform linear methods like LIME and Banzhaf.

Faithfulness for varying the sparsity parameter b: $b = 8$ may not be the sparsity parameter that achieves the best trade-off between samples and faithfulness. For instance, with complex generative models, the cost or time per instance may necessitate taking fewer samples.

In Fig. 7, we showcase the faithfulness results for SPEX and all baseline methods when b is $4, 6, 8$. Since the samples taken by the algorithm grows with 2^b, $b = 8$ takes 16 times more samples than $b = 4$. Even in the low-sample regime, SPEX achieves high faithfulness, often surpassing linear models and second order models. At this scale, we find that third and fourth order models often do not have enough samples to provide a good fit.

C.5.2 Abstract Reasoning

We also evaluated the performance of `Llama 3.2 3B-Instruct` Grattafiori et al. (2024) on the modified trolley problem. As a reminder, the modified problem is presented below:

> A runaway trolley is heading **away** from five people who are tied to the track and cannot move. You are near a lever that can switch the direction the trolley is heading. Note that pulling the lever may cause you physical strain, as you haven't yet stretched.
>
> **True or False: You should not pull the lever.**

Across 1,000 evaluations, the model achieves an accuracy of just 11.8%. Despite a similar accuracy to `GPT-4o mini`, the SHAP and SPEX-computed interactions indicate that the two models are lead astray by different parts of the problem.

The most negative SHAP values of `Llama 3.2 3B-Instruct` appear for later terms such as *pulling* and *lever*, with surrounding words having positive SHAP values. The SPEX-computed interactions tell a different story; many of the words in the last sentence have a negative first order value, with a significant third order interaction between *you haven't yet*. Furthermore, the first word *A* possesses a strong negative second order interaction with *trolley*. Although counterintuitive—since the fact about stretching should only enhance the likelihood of a correct answer—removing the non-critical final sentence unexpectedly boosts the model's accuracy to 20.8%, a 9% improvement.

D Relationships between Fourier and Interaction Concepts

Fourier to Möbius Coefficients: The Möbius Coefficients, also referred to as the *Harsanyi dividends*, can be recovered through Grabisch (2016):

$$I^M(S) = (-2)^{|S|} \sum_{T \supseteq S} F(T). \tag{22}$$

Fourier to Banzhaf Interaction Indices: Banzhaf Interactions Indices Roubens (1996) have a close relationship to Fourier coefficients. As shown in Grabisch et al. (2000):

$$I^{BII}(S) = \sum_{T \supseteq S} \frac{2^{|S|}}{2^{|T|}} I^M(T). \tag{23}$$

Using the relationship from Eq. 22,

$$\begin{aligned}
I^{BII}(S) &= \sum_{T \supseteq S} \frac{2^{|S|}}{2^{|T|}} (-2)^{|T|} \sum_{R \supseteq T} F(R) && (24)\\
&= 2^{|S|} \sum_{T \supseteq S} (-1)^{|T|} \sum_{R \supseteq T} F(R) && (25)\\
&= 2^{|S|} \sum_{R \supseteq S} F(R) \sum_{S \subseteq T \subseteq R} (-1)^{|T|}, && (26)\\
&= (-2)^{|S|} F(S) && (27)
\end{aligned}$$

where the last line follows due to $\sum_{S \subseteq T \subseteq R} (-1)^{|T|}$ evaluating to 0 unless $R = S$.

When S is a singleton, we recover the relationship between Fourier Coefficients and the Banzhaf Value $BV(i)$:

$$BV(i) = I^{BII}(\{i\}) = -2F(\{i\}). \tag{28}$$

Fourier to Shapley Interaction Indices: Shapley Interaction Indices Grabisch (1997) are a generalization of Shapley values to interactions. Using the following relationship to Möbius Coefficients

Grabisch et al. (2000):

$$I^{SII}(S) = \sum_{T\supseteq S} \frac{I^M(S)}{|T|-|S|+1} \tag{29}$$

$$= \sum_{T\supseteq S}\sum_{R\supseteq T} \frac{(-2)^{|T|}F(R)}{|T|-|S|+1} \tag{30}$$

$$= \sum_{R\supseteq S} F(R) \sum_{S\subseteq T\subseteq R} \frac{(-2)^{|T|}}{|T|-|S|+1} \tag{31}$$

$$= \sum_{R\supseteq S} F(R) \sum_{j=|S|}^{|R|} \frac{(-2)^j}{j-|S|+1}\binom{|R|-|S|}{j-|S|} \tag{32}$$

$$= \sum_{R\supseteq S} F(R) \sum_{k=0}^{|R|-|S|} \frac{(-2)^{k+|S|}}{k+1}\binom{|R|-|S|}{k} \tag{33}$$

$$= (-2)^{|S|}\sum_{R\supseteq S} F(R) \sum_{k=0}^{|R|-|S|} \frac{(-2)^{k}}{k+1}\binom{|R|-|S|}{k} \tag{34}$$

Consider the following integral and an application of the binomial theorem:

$$\int_0^t (1+x)^{|R|-|S|}dx = \int_0^t \sum_{k=0}^{|R|-|S|}\binom{|R|-|S|}{k}x^k dx \tag{35}$$

$$= \sum_{k=0}^{|R|-|S|}\binom{|R|-|S|}{k}\int_0^t x^k dx \tag{36}$$

$$= \sum_{k=0}^{|R|-|S|}\binom{|R|-|S|}{k}\left(\frac{t^{k+1}}{k+1}\right) \tag{37}$$

Evaluating at $t=-2$:

$$\sum_{k=0}^{|R|-|S|}\binom{|R|-|S|}{k}\left(\frac{(-2)^k}{k+1}\right) = -\frac{1}{2}\int_0^{-2}(1+x)^{|R|-|S|}dx \tag{38}$$

$$= -\frac{1}{2}\cdot\frac{(-1)^{|R|-|S|+1}-1}{|R|-|S|+1} \tag{39}$$

$$= \begin{cases} \frac{1}{|R|-|S|+1}, & \text{if Parity}(|R|) = \text{Parity}(|S|) \\ 0, & \text{otherwise} \end{cases} \tag{40}$$

As a result, we find the relationship between Shapley Interaction Indices and Fourier Coefficients:

$$I^{SII}(S) = (-2)^{|S|} \sum_{\substack{R\supseteq S,\\ (-1)^{|R|}=(-1)^{|S|}}} \frac{F(R)}{|R|-|S|+1}. \tag{41}$$

When S is a singleton, we recover the relationship between Fourier Coefficients and the Shapley Value $SV(i)$:

$$SV(i) = I^{SII}(\{i\}) = (-2)\sum_{\substack{R\supseteq\{i\},\\ |R| \text{ is odd}}} \frac{F(R)}{|R|}. \tag{42}$$

Fourier to Faith-Banzhaf Interaction Indices: Faith-Banzhaf Interaction Indices Tsai et al. (2023) of up to degree ℓ are the unique minimizer to the following regression objective:

$$\sum_{S\subseteq[n]}\left(f(S) - \sum_{T\subseteq S, |T|\leq \ell} I^{FBII}(T,\ell)\right)^2. \tag{43}$$

Let $g(S)$ be the XOR polynomial up to degree ℓ that minimizes the regression objective. Appealing to Parseval's identity,

$$\sum_{S\subseteq[n]}(f(S)-g(S))^2=\sum_{S\subseteq[n]}(F(S)-G(S))^2 \tag{44}$$

$$=\sum_{S\subseteq[n],|S|\le\ell}(F(S)-G(S))^2+\sum_{S\subseteq[n],|S|>\ell}F(S)^2, \tag{45}$$

which is minimized when $G(S)=F(S)$ for $|S|\le\ell$. Using Eq. 22, it can be seen that the Faith-Banzhaf Interaction Indices correspond to the Möbius Coefficients of the function $f(S)$ truncated up to degree ℓ:

$$I^{FBII}(S,\ell)=(-2)^{|S|}\sum_{T\supseteq S,|T|\le\ell}F(T). \tag{46}$$

Fourier to Faith-Shapley Interaction Indices: Faith-Shapley Interaction Indices Tsai et al. (2023) of up to degree ℓ have the following relationship to Möbius Coefficients:

$$I^{FSII}(S,\ell)=I^M(S)+(-1)^{\ell-|S|}\frac{|S|}{\ell+|S|}\binom{\ell}{|S|}\sum_{T\supset S,|T|>\ell}\frac{\binom{|T|-1}{\ell}}{\binom{|T|+\ell-1}{\ell+|S|}}I^M(T) \tag{47}$$

$$=(-2)^{|S|}\sum_{T\supseteq S}F(T)+ \tag{48}$$

$$(-1)^{\ell-|S|}\frac{|S|}{\ell+|S|}\binom{\ell}{|S|}\sum_{T\supset S,|T|>\ell}\frac{\binom{|T|-1}{\ell}}{\binom{|T|+\ell-1}{\ell+|S|}}(-2)^{|T|}\sum_{R\supseteq T}F(R)$$

$$=(-2)^{|S|}\sum_{T\supseteq S}F(T)+ \tag{49}$$

$$(-1)^{\ell-|S|}\frac{|S|}{\ell+|S|}\binom{\ell}{|S|}\sum_{R\supset S,|R|>\ell}F(R)\sum_{S\subset T\subseteq R,|T|>\ell}\frac{\binom{|T|-1}{\ell}}{\binom{|T|+\ell-1}{\ell+|S|}}(-2)^{|T|}.$$

Fourier to Shapley-Taylor Interaction Indices: Shapley-Taylor Interactions Indices Sundararajan et al. (2020) of up to degree ℓ are related to Möbius Coefficients in the following way:

$$I^{STII}(S,\ell)=\begin{cases}I^M(S), & \text{if } |S|<\ell\\ \sum_{T\supseteq S}\binom{|T|}{\ell}^{-1}I^M(T), & \text{if } |S|=\ell.\end{cases} \tag{50}$$

From an application of Eq. 22,

$$I^{STII}(S,\ell)=\begin{cases}(-2)^{|S|}\sum_{T\supseteq S}F(T), & \text{if } |S|<\ell\\ \sum_{T\supseteq S}\binom{|T|}{\ell}^{-1}(-2)^{|T|}\sum_{R\supseteq T}F(R), & \text{if } |S|=\ell.\end{cases} \tag{51}$$

Simplifying the sum in the $|S|=\ell$ case:

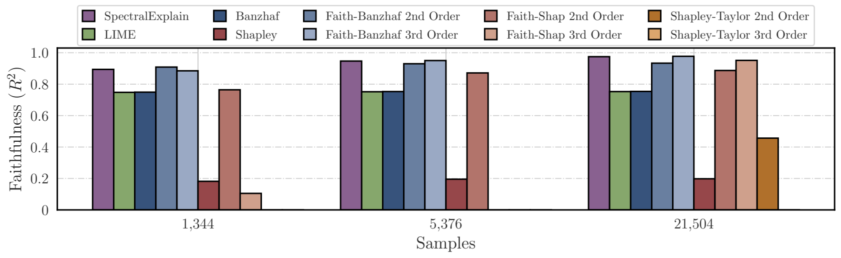
$$\sum_{T\supseteq S}\binom{|T|}{\ell}^{-1}(-2)^{|T|}\sum_{R\supseteq T}F(R)=\sum_{R\supseteq S}F(R)\sum_{S\subseteq T\subseteq R}\binom{|T|}{\ell}^{-1}(-2)^{|T|} \tag{52}$$

$$=\sum_{R\supseteq S}F(R)\sum_{k=\ell}^{|R|}\binom{k}{\ell}^{-1}(-2)^k\binom{|R|-\ell}{k-\ell} \tag{53}$$

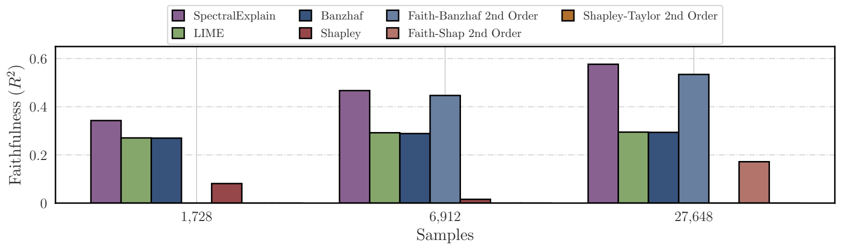
$$\tag{54}$$

Hence,

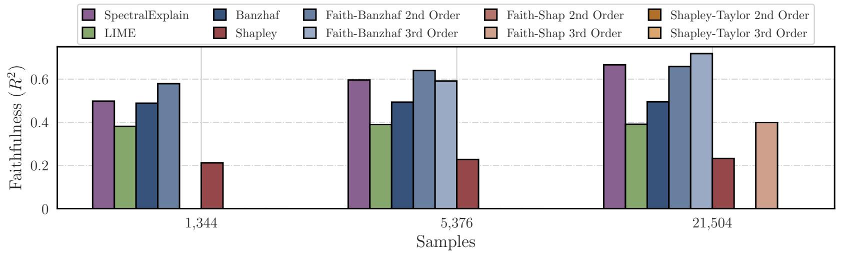
$$I^{STII}(S,\ell)=\begin{cases}(-2)^{|S|}\sum_{T\supseteq S}F(T), & \text{if } |S|<\ell\\ \sum_{T\supseteq S}F(T)\sum_{k=\ell}^{|T|}\binom{k}{\ell}^{-1}(-2)^k\binom{|T|-\ell}{k-\ell}, & \text{if } |S|=\ell.\end{cases} \tag{55}$$



(a) *Sentiment* $n \in [16,31]$

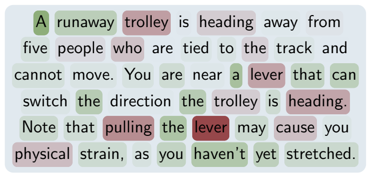


(b) *DROP* $n \in [32,63]$



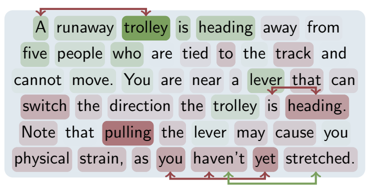
(c) *HotpotQA* $n \in [16,31]$

Figure 7: Faithfulness across the three datasets for all methods that can be feasibly ran. Methods appearing in the legend, but not in the plot had a faithfulness below 0 for the given number of samples.



A runaway trolley is heading away from five people who are tied to the track and cannot move. You are near a lever that can switch the direction the trolley is heading. Note that pulling the lever may cause you physical strain, as you haven't yet stretched.

(a) SHAP values



A runaway trolley is heading away from five people who are tied to the track and cannot move. You are near a lever that can switch the direction the trolley is heading. Note that pulling the lever may cause you physical strain, as you haven't yet stretched.

(b) Interactions computed via SPEX

Figure 8: SHAP and SPEX-computed interactions computed for `Llama 3.2 3B-Instruct`'s answering of the modified trolley problem. Words and interactions highlighted in green contribute positively to producing the correct output, while those in red lead the model toward an incorrect response.