Learning to Understand: Identifying Interactions via the Möbius Transform

Justin S. Kang[1] Yigit E. Erginbas[1] Landon Butler[1] Ramtin Pedarsani[2] Kannan Ramchandran[1]

Berkeley[1] UCSB[2]

Problem

Deep learning models are getting better, but not any easier to understand.

- A popular approach for building explanations of models involves looking at first-order approximations, like the well known Shapley Value.
- First order models can miss important structures critical for explanation.
- Example: A sentiment analysis LLM trained on the IMDB dataset:

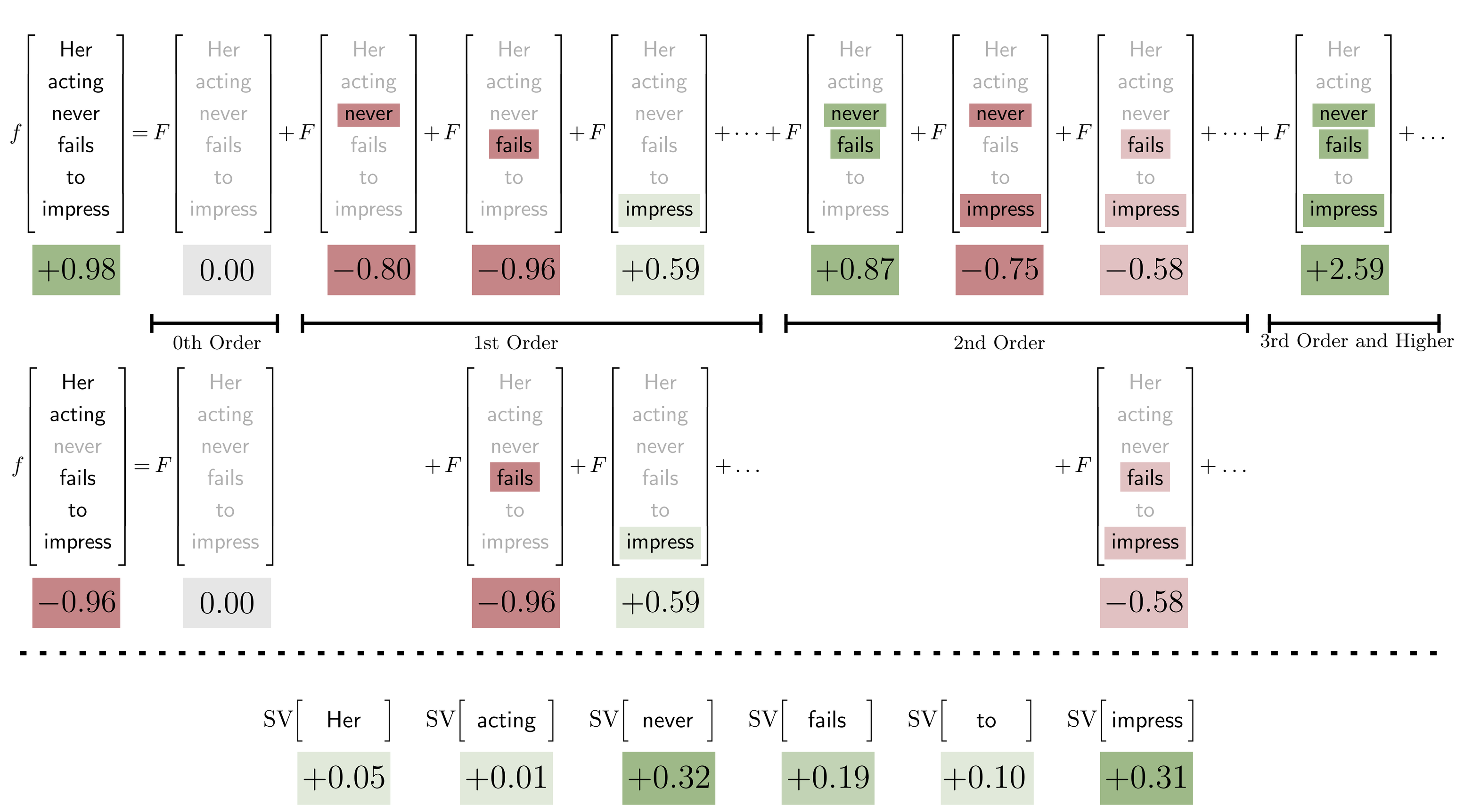


Figure 1: Presented are 1^{st}, 2^{nd} and 3^{rd} order Möbius coefficients. While *never* and *fails* have negative sentiments, combined they are strongly positive. In the second row, the word *never* is deleted, changing overall sentiment. The Shapley values $\mathrm{SV}(\cdot)$ are less informative.

- The word "never" has a negative first-order sentiment, but is involved in critical second order interactions, making its net effect positive.

The Möbius Transform

- The model for higher order interactions is called the Möbius Transform:

Inverse: $f(\mathbf{m}) = \sum_{\mathbf{k} \leq \mathbf{m}} F(\mathbf{k})$, Forward: $F(\mathbf{k}) = \sum_{\mathbf{m} \leq \mathbf{k}} (-1)^{\mathbf{1}^{\mathrm{T}}(\mathbf{k}-\mathbf{m})} f(\mathbf{m})$

Naïve computation is exponential in number of features n.

- The Shapley Values $\mathrm{SV}(\cdot)$ and Banzhaf Values $\mathrm{BZ}(\cdot)$ can be written as:

$$\mathrm{SV}(i) = \sum_{\mathbf{k}:k_i=1} \frac{1}{|\mathbf{k}|} F(\mathbf{k}), \qquad \mathrm{BZ}(i) = \sum_{\mathbf{k}:k_i=1} \frac{1}{2^{|\mathbf{k}|-1}} F(\mathbf{k}).$$

- A small number of interactions dominate the function overall.

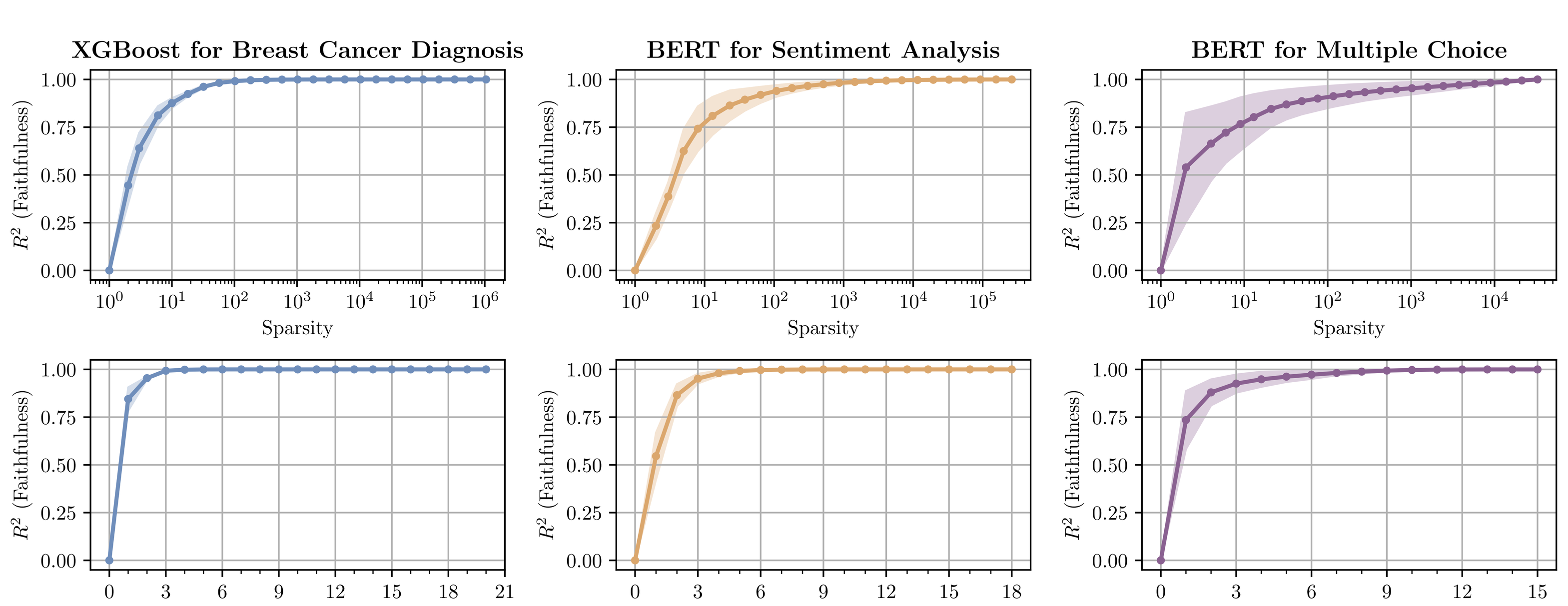


Figure 2: $F(\mathbf{k})$ generally has a sparse structure. The functions are well-approximated with only a small number of coefficients (sparsity), and these coefficients also have small $|\mathbf{k}|$ (low degree). Can we compute the Möbius transform more efficiently under these settings?

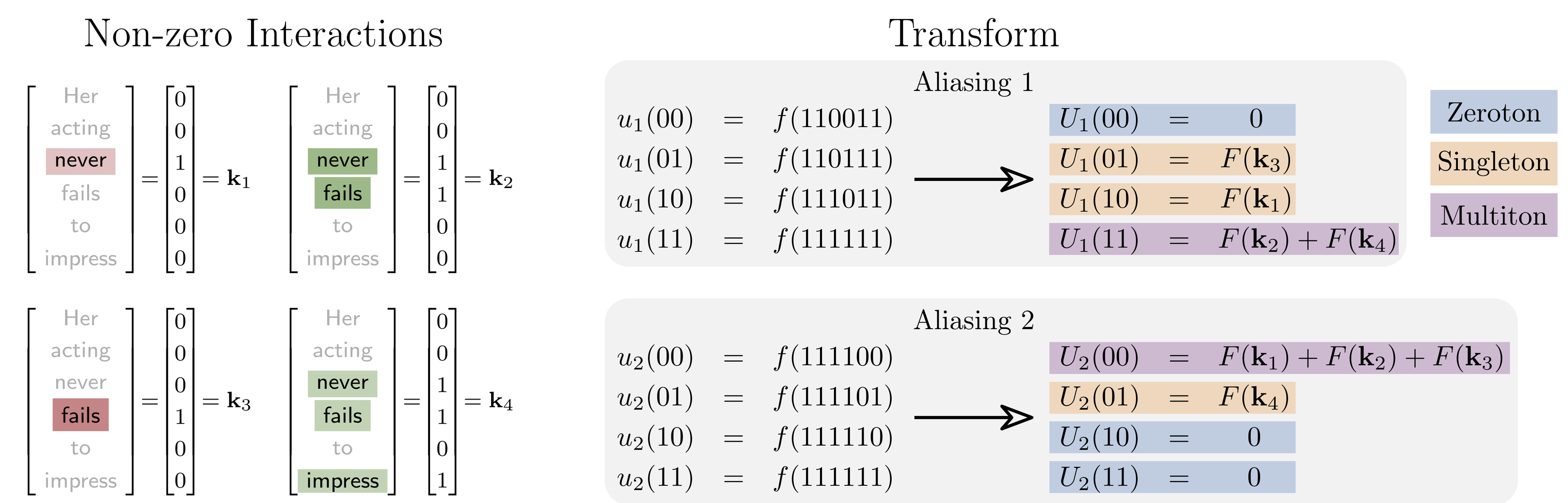
The Algorithm

Step 1: Aliasing Informed Masking Design

- Construct the function u from samples of f with $b \ll n$, and take the Transform of u, denoted U in $b2^b$ time:

$$u_c(\boldsymbol{\ell}) = f\left(\overline{\mathbf{H}_c^{\mathrm{T}}\overline{\boldsymbol{\ell}}}\right) \quad \forall \boldsymbol{\ell} \in \mathbb{Z}_2^b \iff U_c(\mathbf{j}) = \sum_{\mathbf{H}_c\mathbf{k}=\mathbf{j}} F(\mathbf{k}) \quad \forall \mathbf{j} \in \mathbb{Z}_2^b.$$

- Aliasing effectively hashes the coefficients $F(\mathbf{k})$ into one of 2^b bins:



- The singleton coefficients can be detected, and their $\mathbf{k}$ index identified.

Step 2: Identifying Interactions via Group Testing

- The key to identifying a singletons is to construct "delayed" versions of u:

$$u_{cp}(\boldsymbol{\ell}) = f\left(\overline{\mathbf{H}_c^{\mathrm{T}}\overline{\boldsymbol{\ell}} + \mathbf{d}_p}\right) \iff U_c(\mathbf{j}) = \sum_{\substack{\mathbf{H}_c\mathbf{k}=\mathbf{j} \\ \mathbf{k} \leq \overline{\mathbf{d}}_p}} F(\mathbf{k}).$$

- A "delay" is a membership test on $\mathbf{k}$. Repeating, we construct $\mathbf{y} = \mathbf{D}\mathbf{k}$.
- When $\mathbf{k}$ is arbitrary we take $\mathbf{D} = \mathbf{I}$, and require n delays $\mathbf{d}_p$.
- When $|\mathbf{k}| < t$ for some t, we choose $\mathbf{D}$ as a group testing matrix:

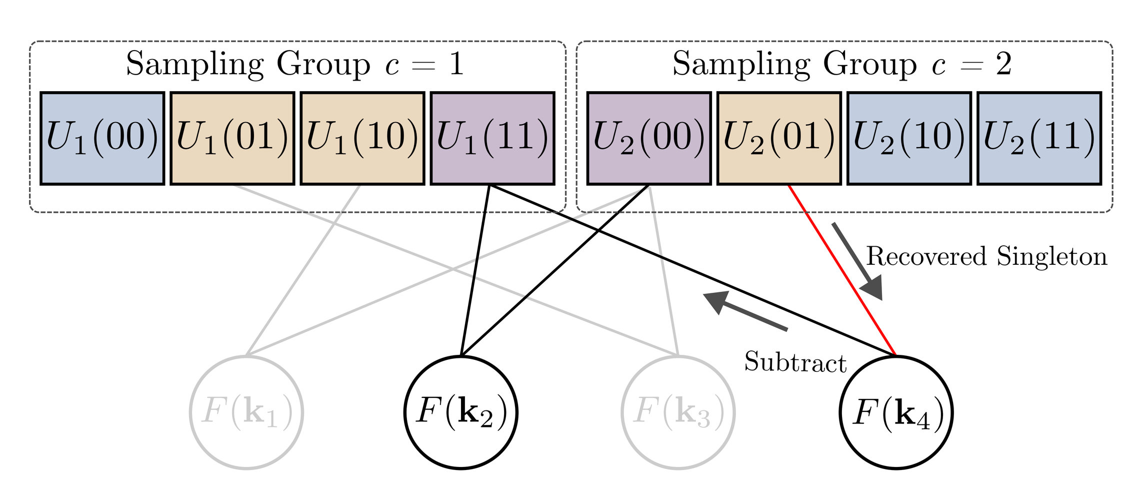
	Her	acting	never	fails	to	impress	$\mathbf{y}$
$\mathbf{k}_1 =$							
$\mathbf{D} =$	0	0	0	1	1	1	0
	0	1	1	0	0	1	1
	1	0	1	0	1	0	1

$\mathbf{y} \xrightarrow{\text{Decode}} \mathbf{k}_1$

- Theory says we only require $O(t\log(n))$ delays to ensure recovery.

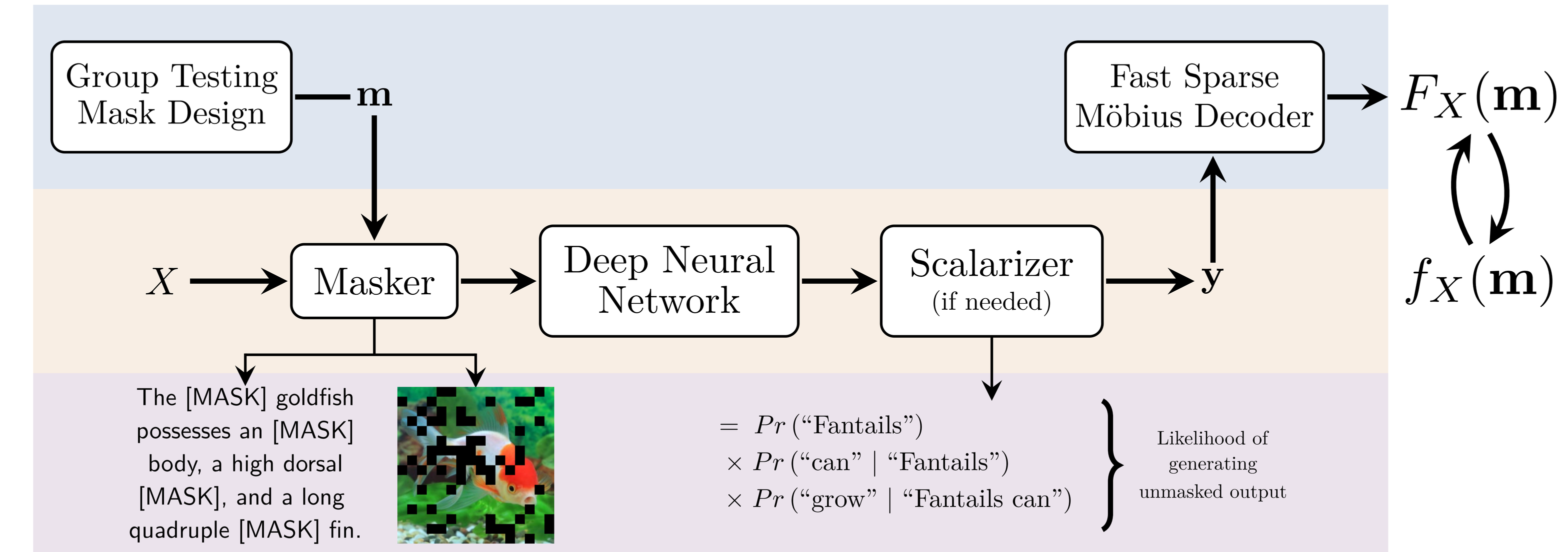
Step 3: Message Passing to Resolve Collisions

- Defines a bipartite graph connecting the non-zero $F(\mathbf{k})$ and U.
- Use a message passing algorithm (peeling decoder) to resolve multitons. This is inspired by sparse graph codes for robust communication.



- Choosing $\mathbf{H}, \mathbf{D}$ correctly ensures we are likely to peel all non-zero $F(\mathbf{k})$.
- Density evolution theory can prove the performance of the algorithm.

Overview



We design masking patterns according to a group testing design, and perform inference of the masked inputs. If needed, the output is converted to a scalar, and the output is used to compute the Möbius Transform.

Our algorithm is non-adaptive and has rigorous performance guarantees.

Theorems

1. (Sparse) With K non-zero interactions among all 2^n interaction, our algorithm exactly computes the Mobius transform $F(\mathbf{k})$ in $O(Kn)$ samples and $O(Kn^2)$ time with probability $1 - O(1/K)$.
2. (Sparse, Low Degree) When there are K non-zero interactions all with $|\mathbf{k}| \leq t$, our algorithm computes the Mobius transform in $O(Kt\log(n))$ samples and $O(K\,\mathrm{poly}(n))$ time with probability $1 - O(1/K)$, even under the presence of noise at any fixed SNR.

Experiments

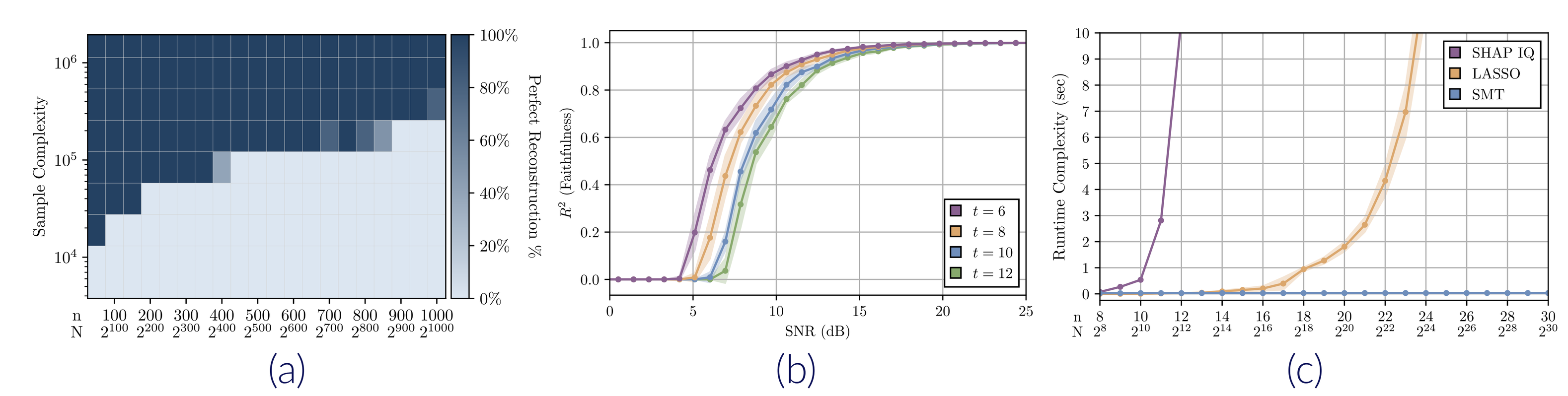


Figure: (a) Sample complexity of our algorithm. Clear phase transition, with the threshold scaling linearly in n is visible. (b) Shows our algorithm under a noise model where $U(\mathbf{j})$ are corrupted by Gaussian noise at different SNR.

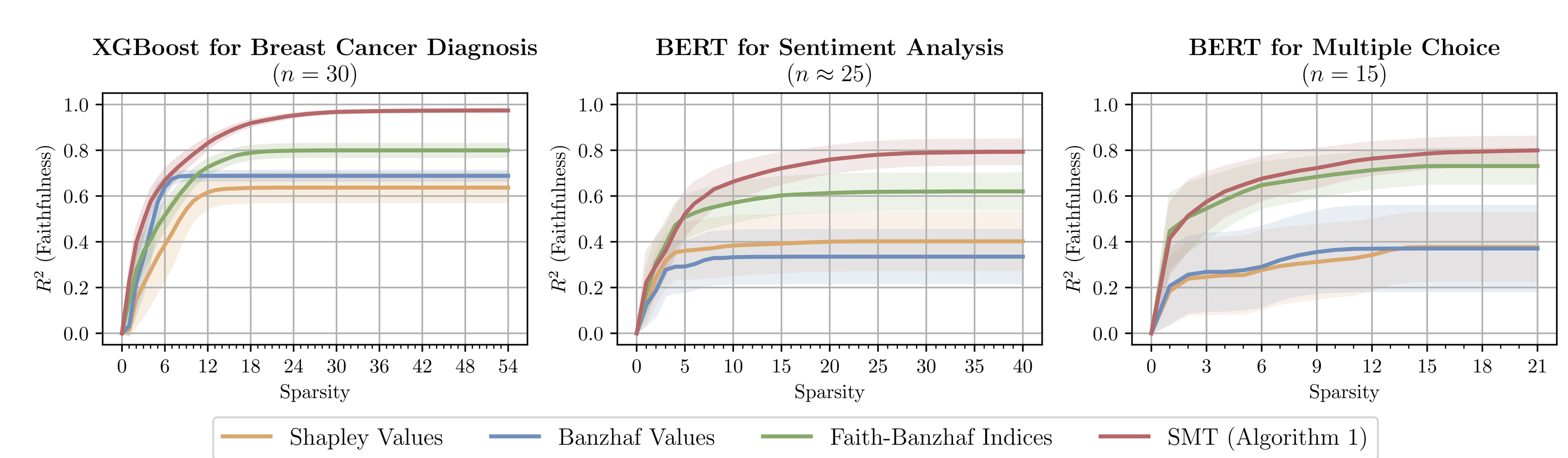


Figure: Using only a small number of coefficients (sparsity), the Möbius transform computed by our method outperforms first order methods in faithfulness (R^2) to the underlying network. The gap is larger in problems with non-linear feature relationships.

Further Reading

[1] Kang JS, et al. "Learning to Understand: Identifying Interactions via the Möbius Transform". NeurIPS (2024).

[2] Erginbas, YE, Kang, JS et al.. "Efficiently Computing Sparse Fourier Transforms of q-ary Functions." IEEE ISIT (2023).