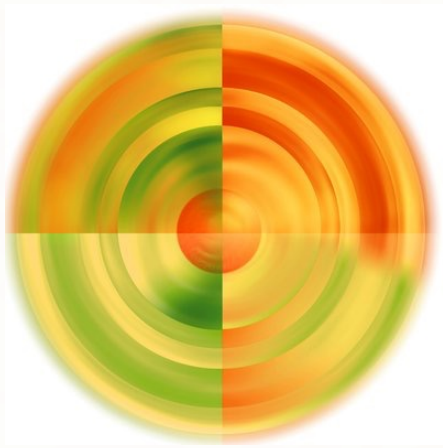
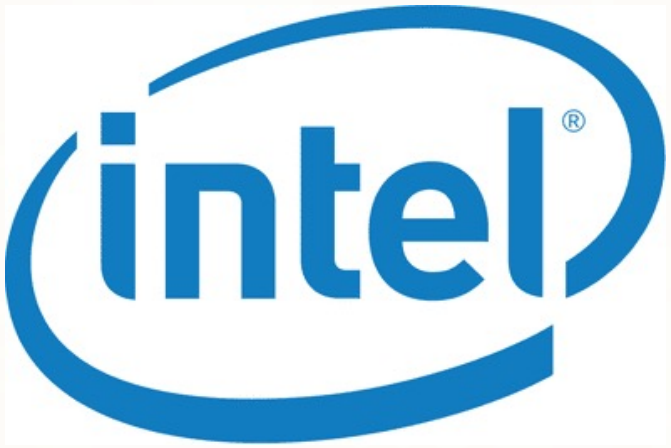
Integer Factorization with Compositional Distributed Representations

Denis Kleyko, Connor Bybee, Christopher J. Kymn, Bruno A. Olshausen, Amir Khosrowshahi, Dmitri E. Nikonov, Friedrich T. Sommer, E. Paxon Frady



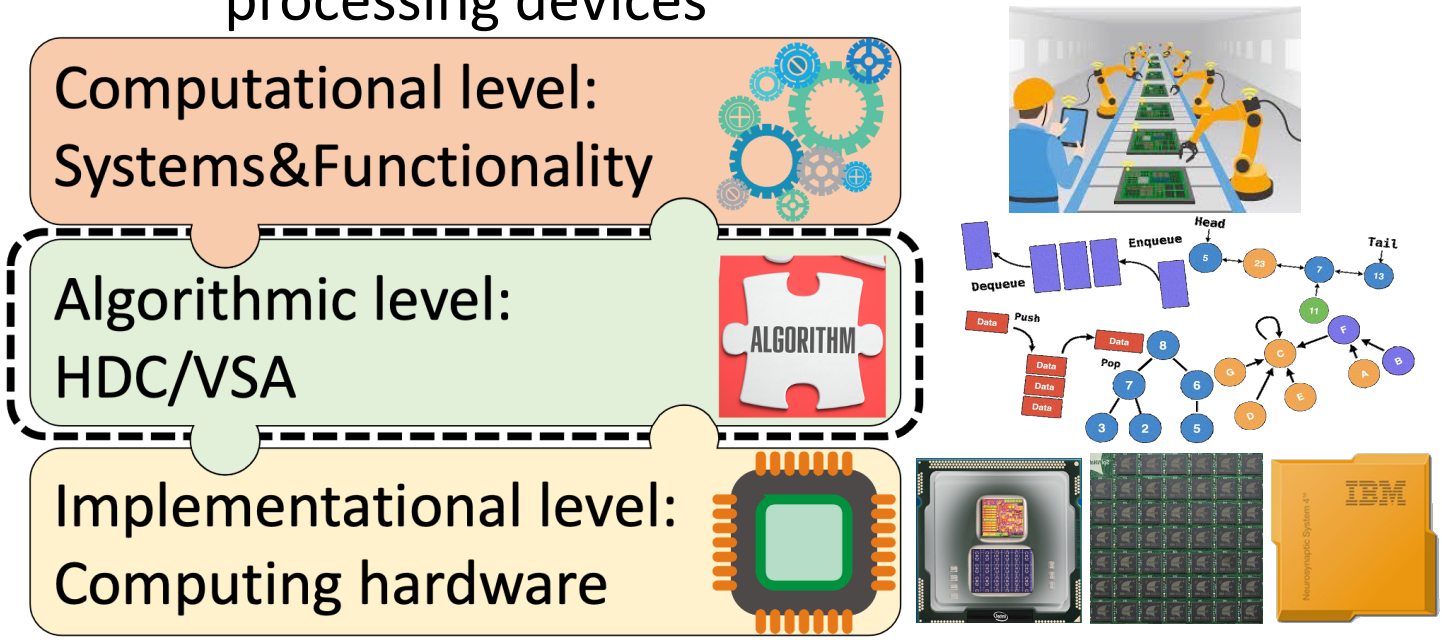


9th Annual Neuro-Inspired Computational Elements Workshop, March 29, 2022

Neuromorphic engineering

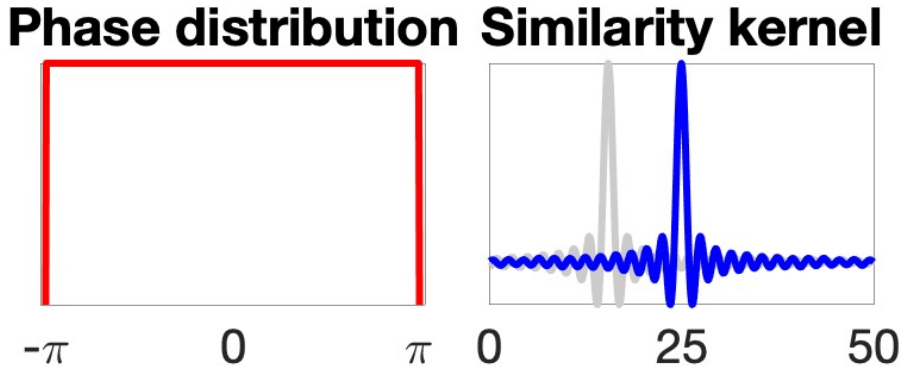
- Programming neuromorphic hardware can be challenging
  - Algorithm development is not straightforward
- HDC/VSA framework for distributed computing
  - VFA = HDC/VSA + KLPE
  - Abstraction for programming neuromorphic hardware

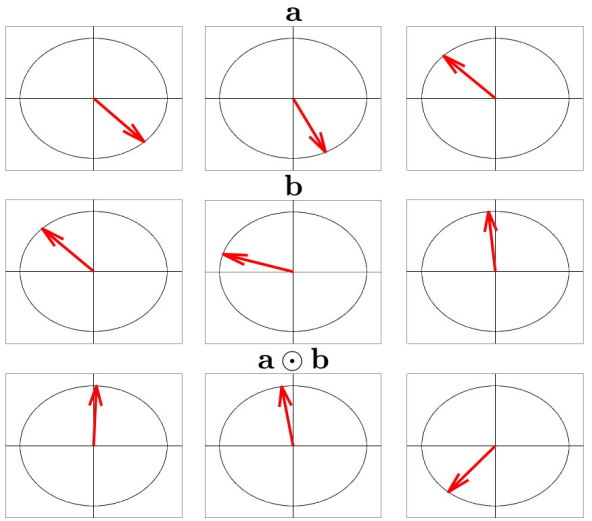
Marr's levels for information-processing devices



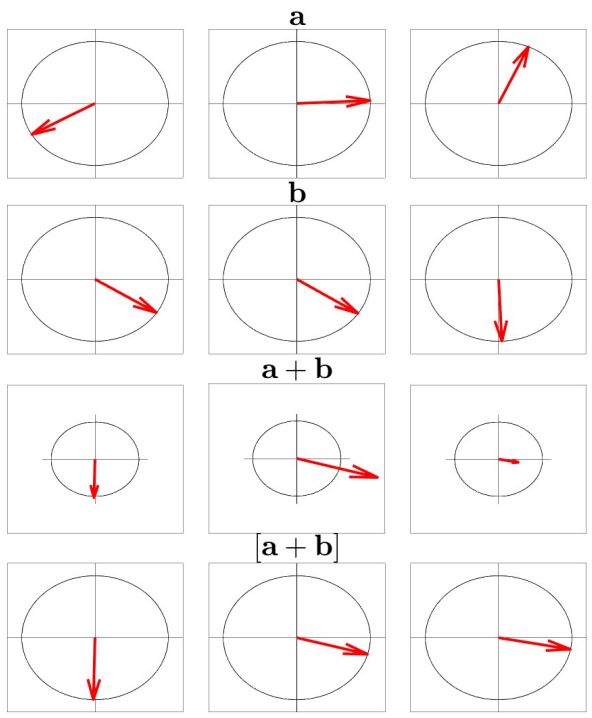
Integers as HD vectors: Vector Function Architecture

- **C. Kymn "Computing on Functions Using Randomized Vector Representations"**
- Choose random base HD vector - **z**
  - **z** ~ phasors $e^{i\varphi}$ $\varphi \sim U(-\pi,+\pi]$
  - VFA of x: $\mathbf{z}(x) = \mathbf{z}^x$
- Binding of VFA representations corresponds to addition
  - $\mathbf{z}(x) \odot \mathbf{z}(y) = \mathbf{z}^x \odot \mathbf{z}^y = \mathbf{z}^{x+y} = \mathbf{z}(x+y)$
- Superposition:
  - Component-wise addition
  - Normalizes each component to have unit magnitude





Unit magnitude normalization



T. A. Plate, "Holographic Reduced Representations," IEEE Transactions on Neural Networks, 1995.
E. P. Frady, et al., "Computing on Functions Using Randomized Vector Representations," arXiv, 2021.

More applications of VFA?

- Focus: integer factorization
  - **Semiprimes** factorization: $s = xy$

- Transformation of problem into HDC/VSA terms
  - $\mathbf{z}(x) \odot \mathbf{z}(y) = \mathbf{z}^{x} \odot \mathbf{z}^{y} = \mathbf{z}^{x+y} = \mathbf{z}(x + y)$
  - Factorization needs multiplication!

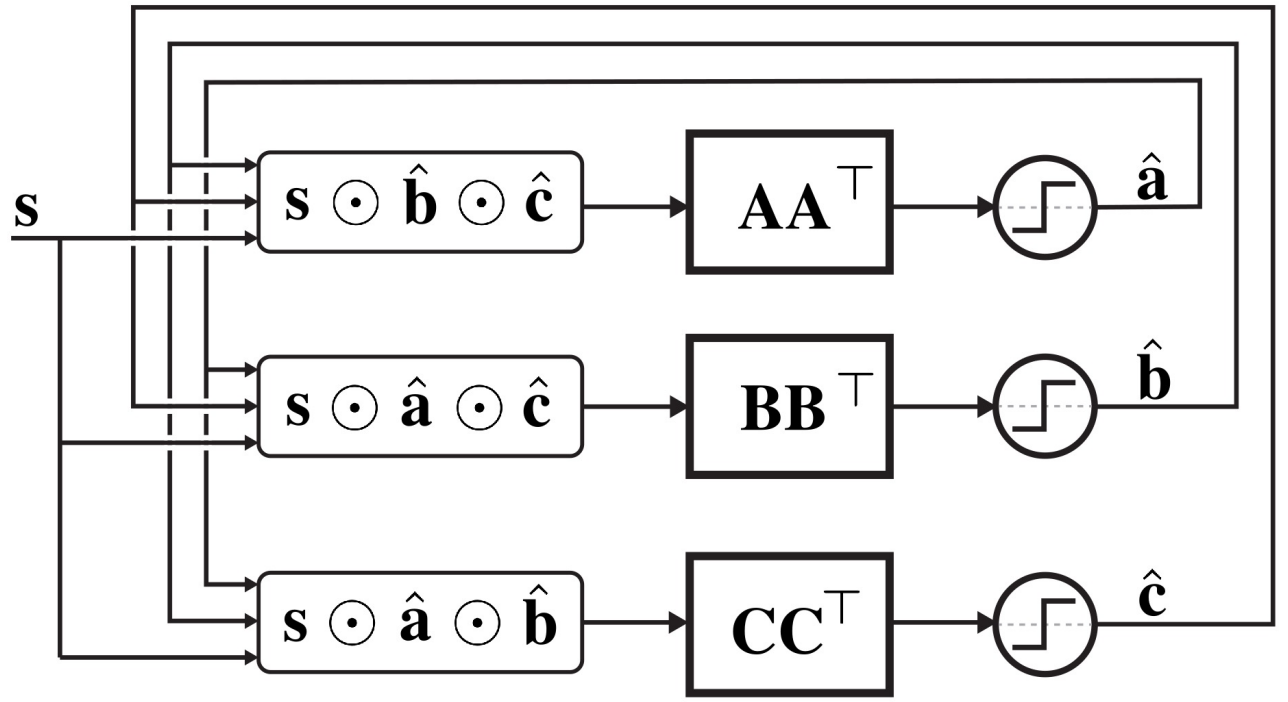
Log transformation:

$$\mathbf{z}(\log(s)) = \mathbf{z}^{\log(s)} = \mathbf{z}^{\log(xy)} = \mathbf{z}^{(\log(x)+\log(y))} = \mathbf{z}^{\log(x)} \odot \mathbf{z}^{\log(y)} = \mathbf{z}(\log(x)) \odot \mathbf{z}(\log(y))$$

- "Stress test" for VFA

Factorization process: Resonator network

- Problems involving vector factorization are common in HDC/VSA
  - $\mathbf{s} = \mathbf{a} \odot \mathbf{b} \odot \mathbf{c}$
- Resonator network
  - Solves vector factorization
  - Iterative
  - Hopfield-like CAM
  - Inverting binding: $\mathbf{a} \odot \mathbf{b} \odot \bar{\mathbf{b}} = \mathbf{a}$
  - Strategy: search in superposition



E. P. Frady, et al., "Resonator Networks, 1: An Efficient Solution for Factoring High-Dimensional, Distributed Representations of Data Structures," Neural Computation, 2020.

Resonator network for semiprimes factorization

- Define the set $\mathcal{P}(s)$
  - Includes all primes that are potential factors of s
- Populate codebook $\mathbf{\Phi}$
  - Generate random base vector **z**
  - Contains VFA representations for every prime in $\mathcal{P}(s)$
- Setup resonator network to factorize **z**(log(s))

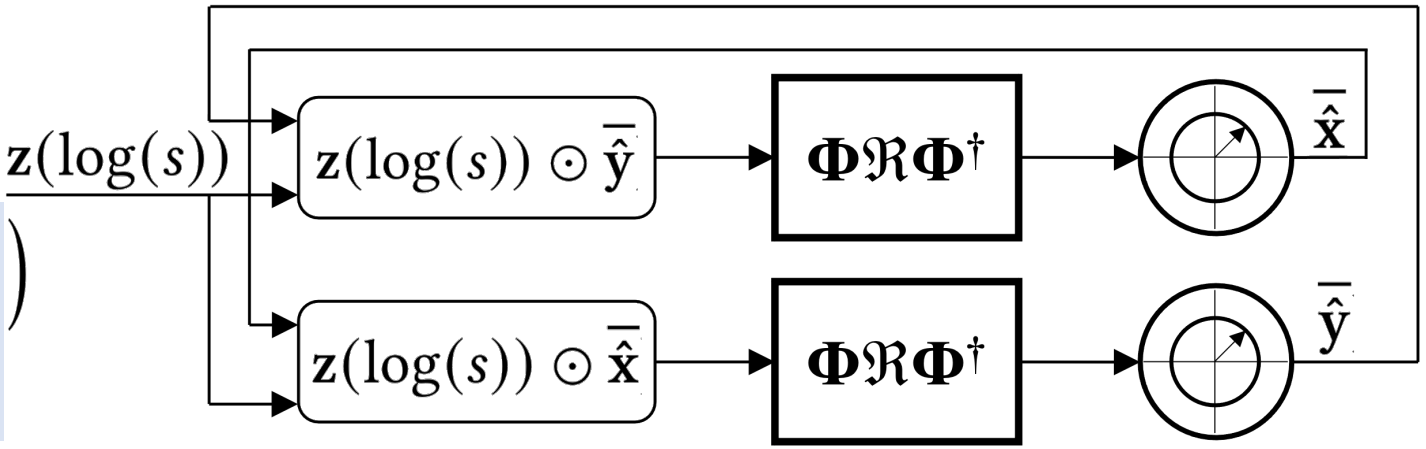
Crucial: $\hat{\mathbf{x}}(t)$ & $\hat{\mathbf{y}}(t)$ are superpositions of entries of $\mathbf{\Phi}$

$$\hat{\mathbf{x}}(t+1) = f_n\left(\mathbf{\Phi}\Re\left(\mathbf{\Phi}^{\dagger}\left(\mathbf{z}(\log(s)) \odot \bar{\hat{\mathbf{y}}}(t)\right)\right)\right)$$

$$\hat{\mathbf{y}}(t+1) = f_n\left(\underbrace{\mathbf{\Phi}\Re\left(\mathbf{\Phi}^{\dagger}}\left(\mathbf{z}(\log(s)) \odot \underbrace{\bar{\hat{\mathbf{x}}}(t+1)}\right)\right)\right)$$

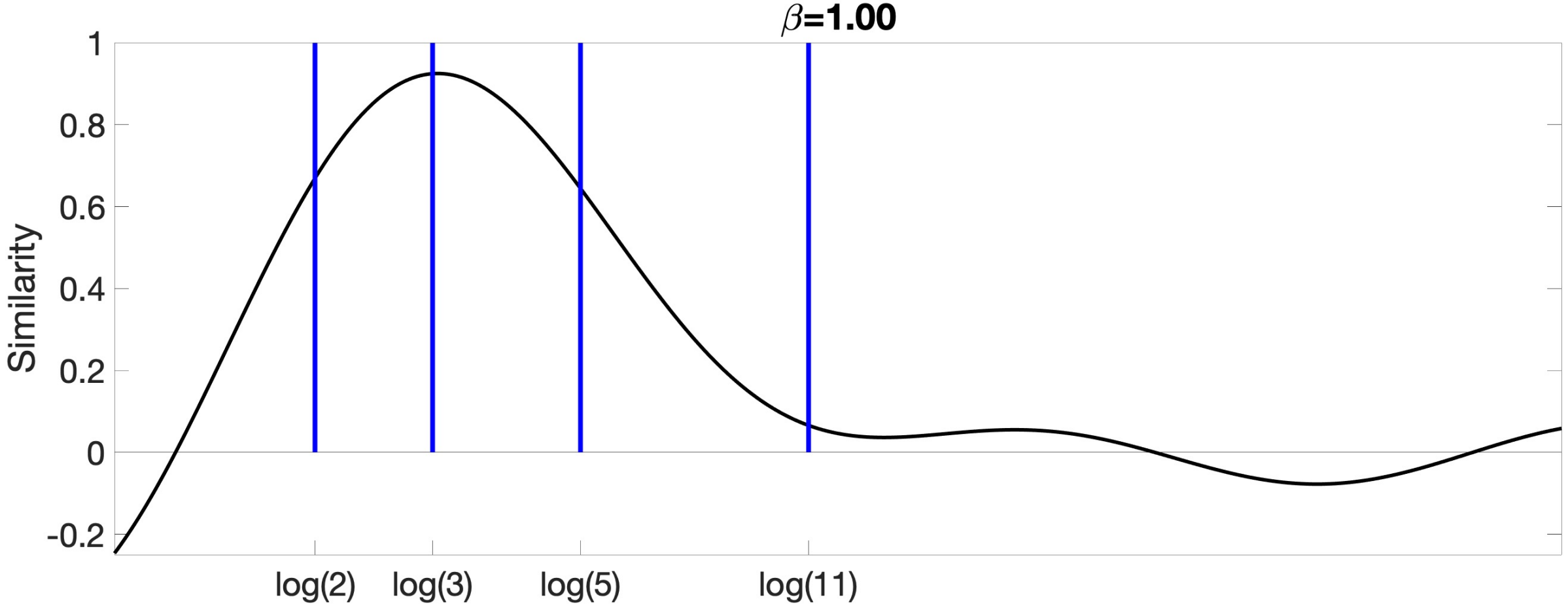
Constrain $\hat{\mathbf{y}}$ guesses to codebook Φ

Use guess in $\hat{\mathbf{x}}$ to infer $\hat{\mathbf{y}}$



VFA representations in superposition

- Example: {log(2), log(3), log(5), log(11)} $\mathbf{s} = \mathbf{z}^{\log(2)} + \mathbf{z}^{\log(3)} + \mathbf{z}^{\log(5)} + \mathbf{z}^{\log(11)}$
- Let's measure cosine similarity between **s** and VFA representations of scalars

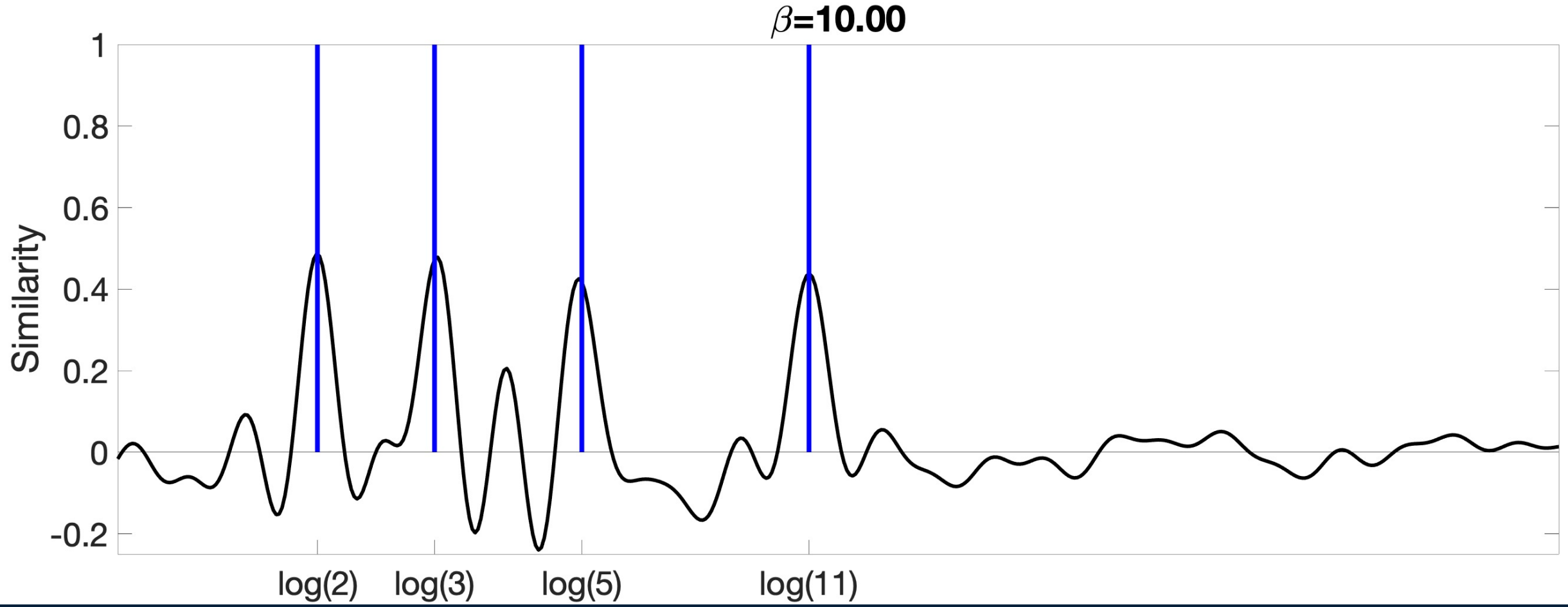


VFA representations in superposition: role of β

- Let's consider β

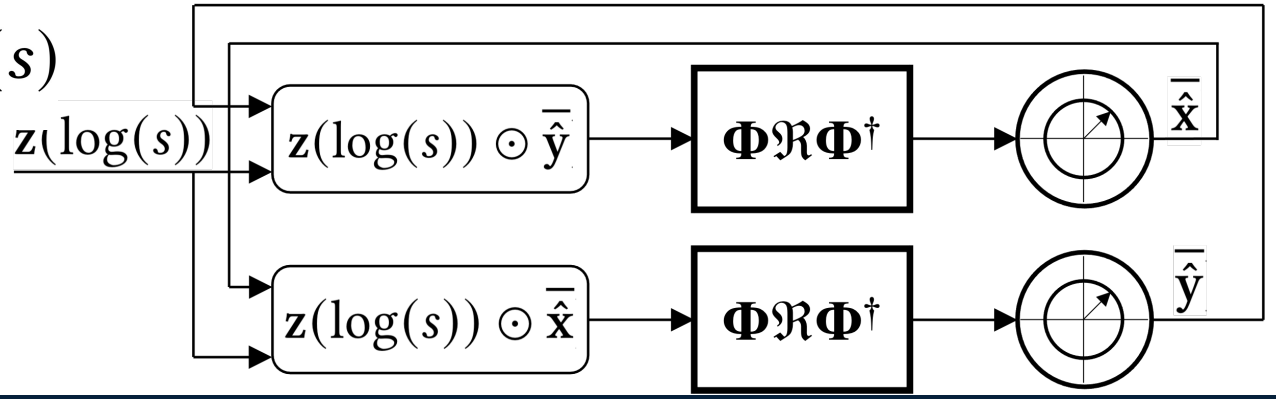
$$\mathbf{s} = \mathbf{z}^{\beta \log(2)} + \mathbf{z}^{\beta \log(3)} + \mathbf{z}^{\beta \log(5)} + \mathbf{z}^{\beta \log(11)}$$

- Symbolic vs. sub-symbolic mode



Example: Setup

- Let's factorize $s = xy = 603{,}329$
  - $x = 757$ & $y = 797$
- Define the set $\mathcal{P}(s)$
  - Primes, $\mathcal{P}(s)$ = {743, 751, 757, 761, 769, 773, 787, 797, 809, 811}
  - $|\mathcal{P}(s)|$ = 10
- Choose $\beta = 400$, n = 15
- Generate n-dim. base vector **z**
- Populate $\mathbf{\Phi}$ using **z**, β, and $\mathcal{P}(s)$
- Form VFA repr. of s: $\mathbf{z}(\log(s)) = \mathbf{z}^{\beta \log(s)}$
- Setup and run the resonator network

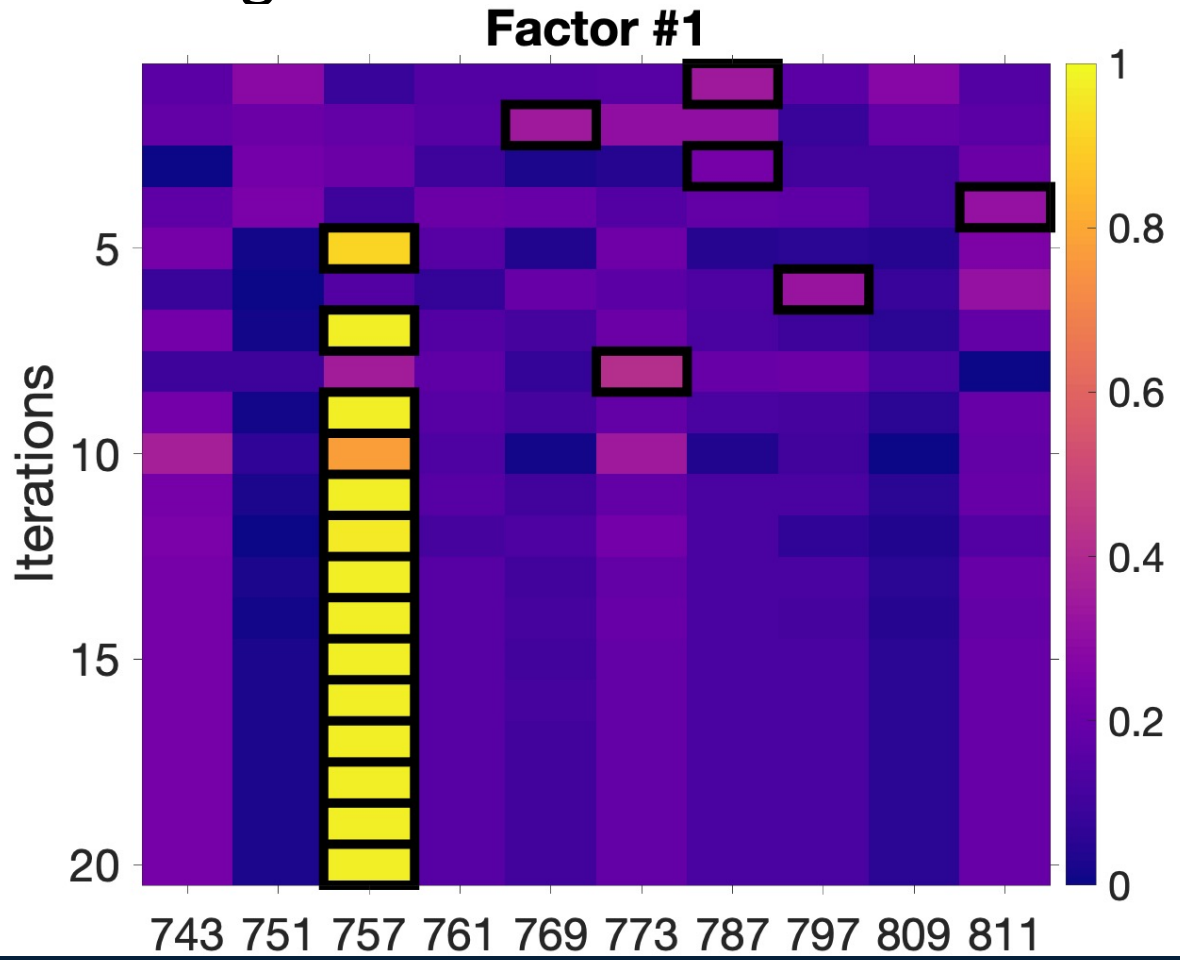


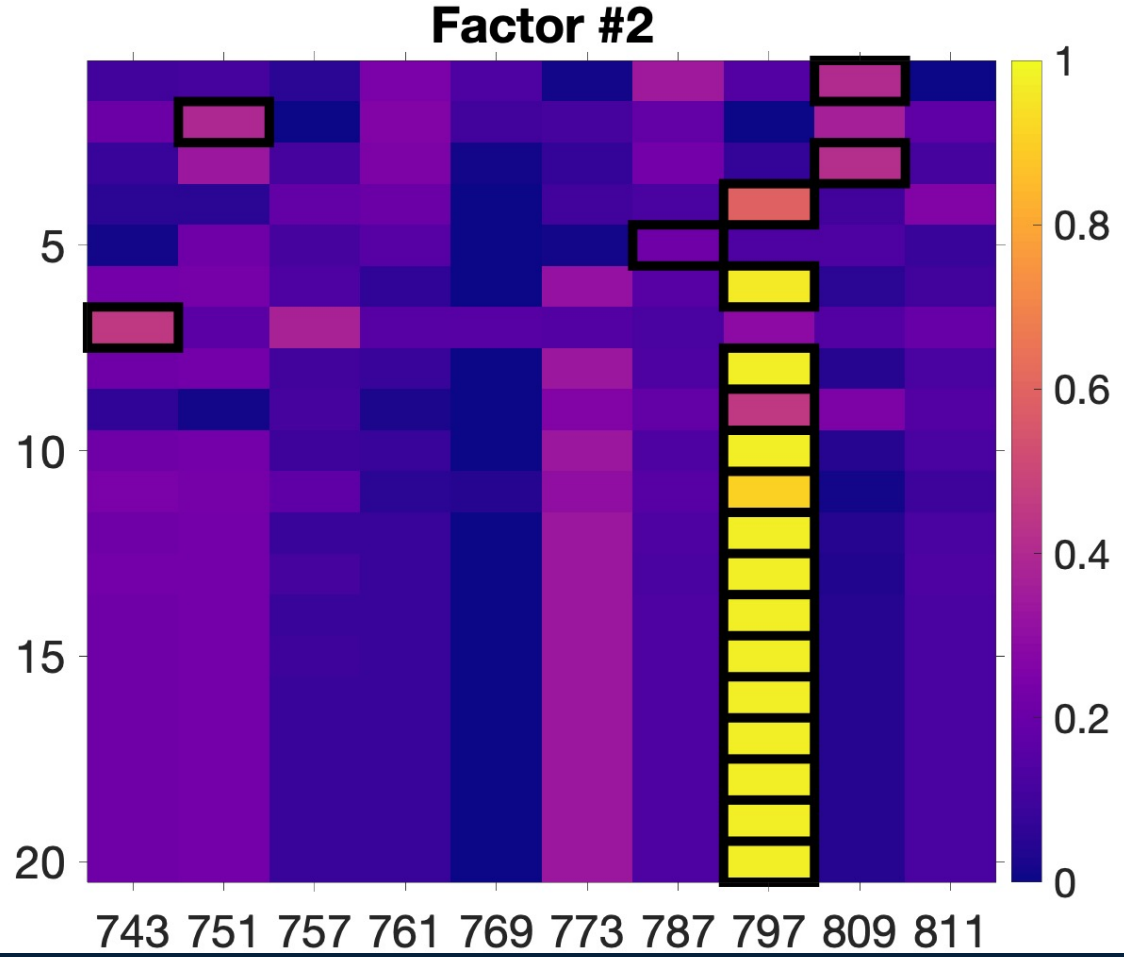
Example: Resonator network dynamics

- Many solutions compete
- Converges to the correct factors: 757 & 797

Factor #1: $\hat{\mathbf{x}}(t+1) = f_n\left(\Phi\Re\left(\Phi^{\dagger}(\mathbf{z}(\log(s)) \odot \bar{\hat{\mathbf{y}}}(t))\right)\right)$

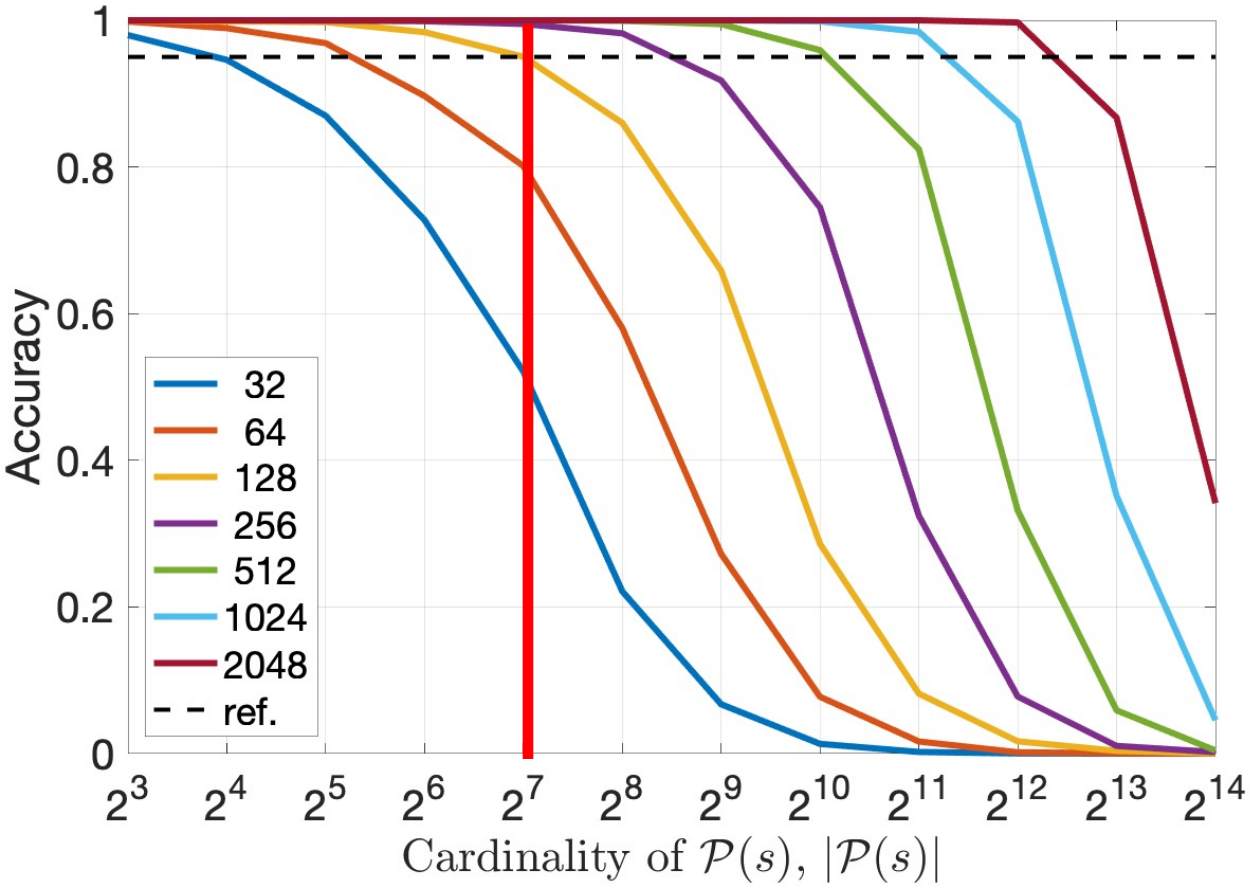
Factor #2: $\hat{\mathbf{y}}(t+1) = f_n\left(\Phi\Re\left(\Phi^{\dagger}(\mathbf{z}(\log(s)) \odot \bar{\hat{\mathbf{x}}}(t+1))\right)\right)$

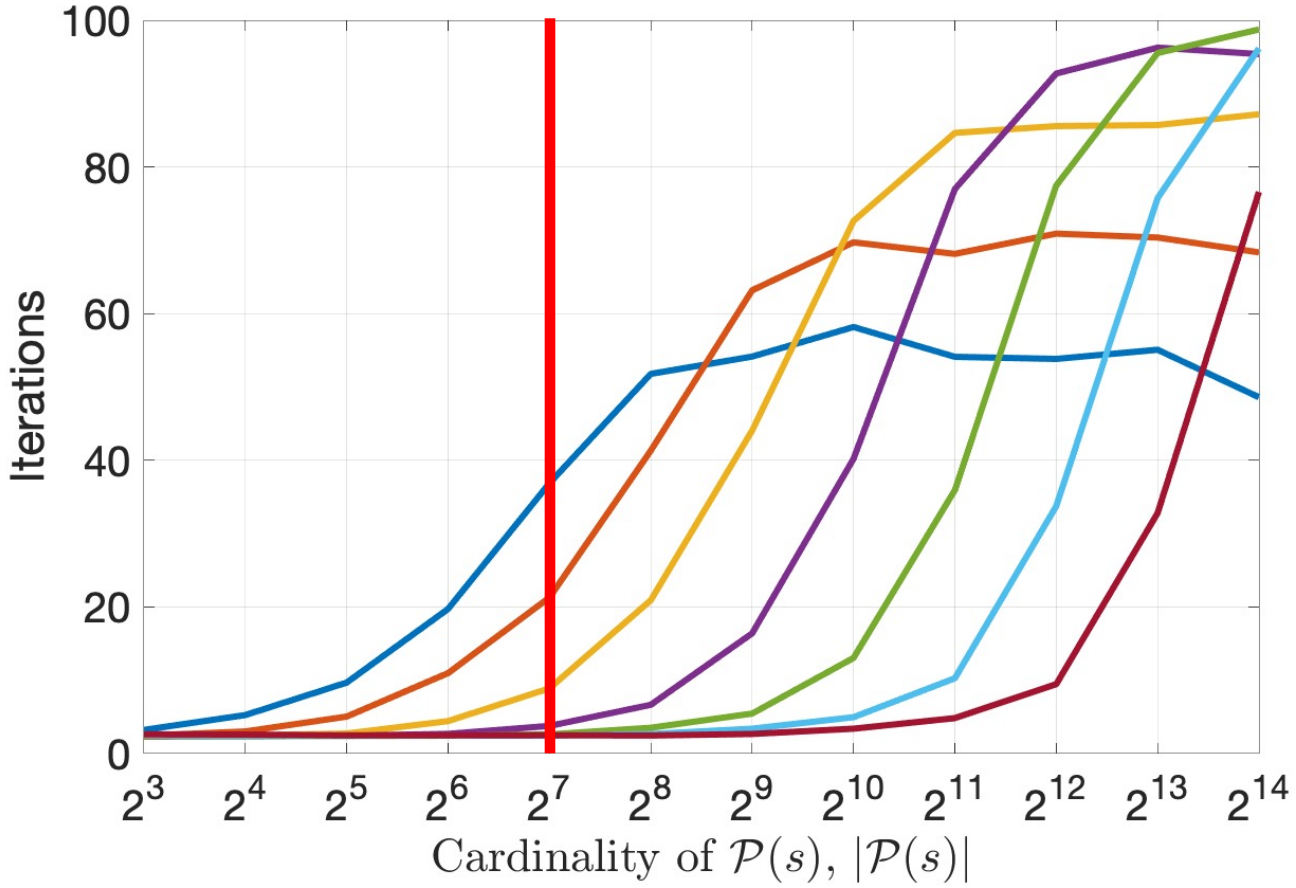




Empirical evaluation: Fixed dimensionality

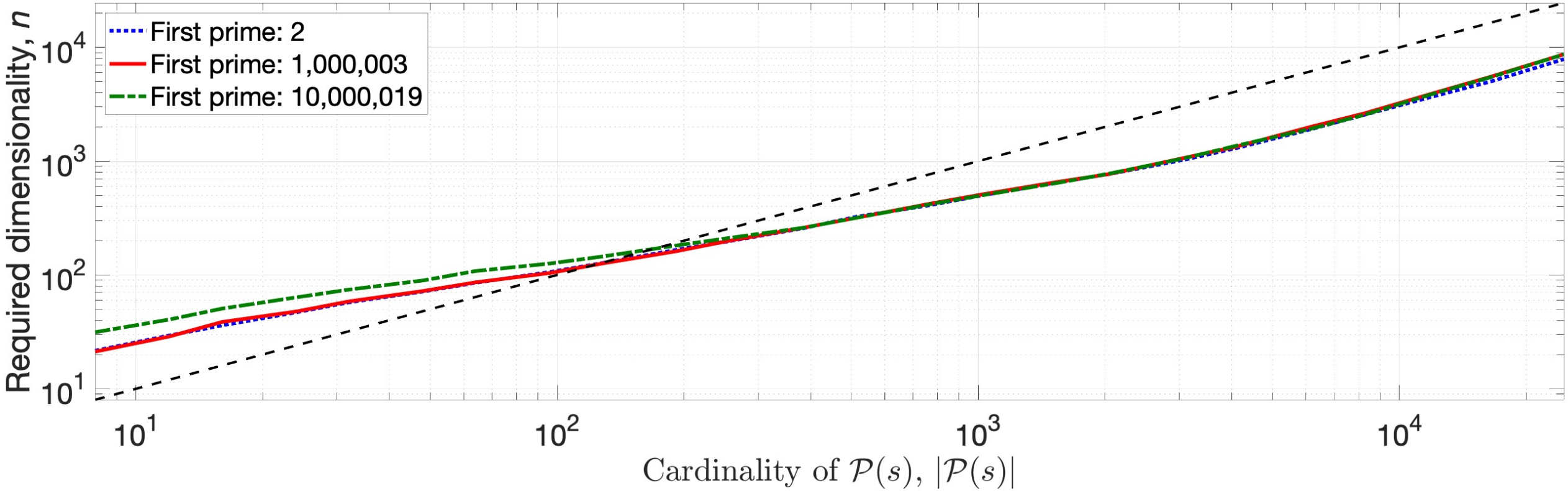
- Factorization accuracy and iterations for different dimensionalities of HD vectors
- Size of the search space: $|\mathcal{P}(s)|^2$
- # of iterations is indicative of factorization performance





Scaling

- For a given $|\mathcal{P}(s)|$ find *n* such that acc. ≥ 95%



- Search space scales as $|\mathcal{P}(s)|^2$ while the required dimensionality scales linearly
- Required *n* depends on the search space but not on prime numbers themselves

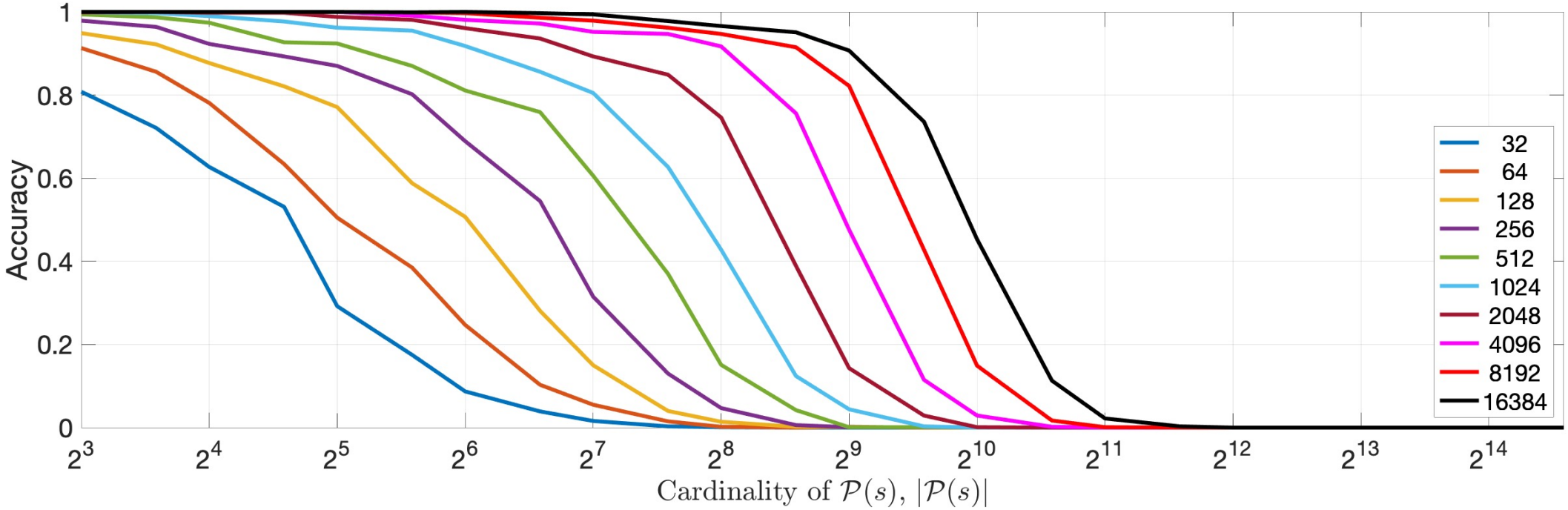
Extension: *k*-almost primes

- 3-almost primes: $s = wxy$
- Factorization accuracy and iterations for different dimensionalities of HD vectors
- Size of the search space: $|\mathcal{P}(s)|^3$

$$\hat{\mathbf{w}}(t+1) = f_n\left(\Phi\Re\left(\Phi^{\dagger}(\mathbf{z}(\log(s)) \odot \bar{\hat{\mathbf{x}}}(t) \odot \bar{\hat{\mathbf{y}}}(t))\right)\right)$$

$$\hat{\mathbf{x}}(t+1) = f_n\left(\Phi\Re\left(\Phi^{\dagger}(\mathbf{z}(\log(s)) \odot \bar{\hat{\mathbf{w}}}(t) \odot \bar{\hat{\mathbf{y}}}(t))\right)\right)$$

$$\hat{\mathbf{y}}(t+1) = f_n\left(\Phi\Re\left(\Phi^{\dagger}(\mathbf{z}(\log(s)) \odot \bar{\hat{\mathbf{w}}}(t) \odot \bar{\hat{\mathbf{x}}}(t))\right)\right)$$



Summary

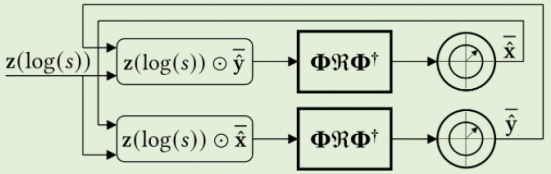
- HDC/VSA/VFA as an abstraction level
- Vector factorization ⇔ classic CS factorization
- Promising new way of approaching a broad class of factorization problems in CS
  - Seamlessly distributed
  - Implementable on parallel hardware

Marr's levels

Computational:
Systems&Functionality

$s = xy$

Algorithmic:
HDC/VSA



Implementational:
Computing hardware