Syracuse University
College of Engineering
& Computer Science

Scalable External Memories for Neural Networks

Ruipeng Liu

Syracuse University

rliu02@syr.edu

Overview

Outline

Differentiable Data Structures and Memory

One approach to neurosymbolic AI: Vector/matrix emulations of symbolic data structures and memory

- Neural Turing Machines (NTMs) [1],
- Neural Virtual Machines (NVMs) [2],
- Associative Memory [3], Vector-Symbolic Architectures (VSAs) [4]

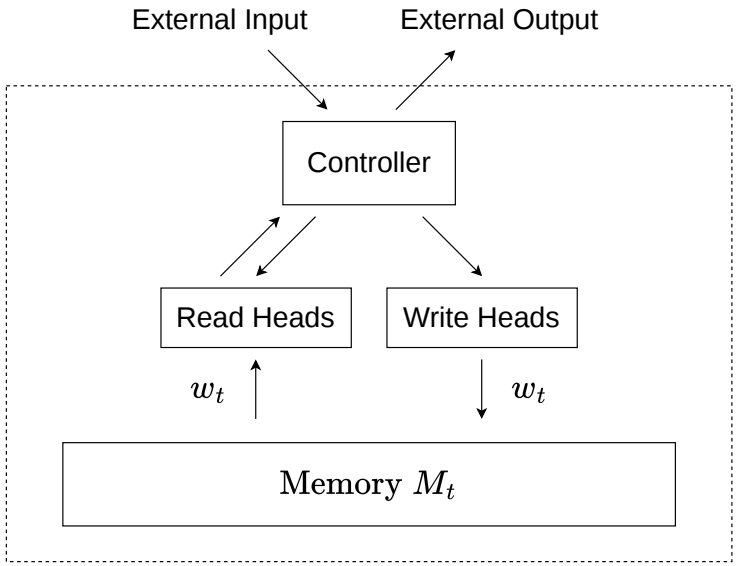


Figure: High Level Structure of NTM

Neural Virtual Machines (NVMs)

- Neural networks that emulate symbolic program execution
- RAM reads and writes are emulated by single-pass linear associative memory
- Reading a "value" $v \in \{-1, +1\}^N$ from "address" $a \in \{-1, +1\}^N$ in "memory" $M \in \mathbb{R}^{N \times N}$:

$$v \leftarrow \sigma(Ma)$$

  where σ is sign or sigmoid
- Writing (or overwriting):

$$M \leftarrow M - \underbrace{\sigma(Ma)a^\top/N}_{\text{Erase}} + \underbrace{va^\top/N}_{\text{Write}}$$

- Main challenges: Storage capacity scales linearly with N and computational complexity is $\mathcal{O}(N^2)$

Motivation

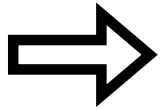
- **Goal**: Better Memory Structure
  - Larger capacity
  - Lower computational complexity
- **Directions**:
  - Neural network as memory: single-pass full-capacity learning rule
  - Alternative key-value memory structures: **V**ector **S**ymbolic **A**rchitectures (VSAs)

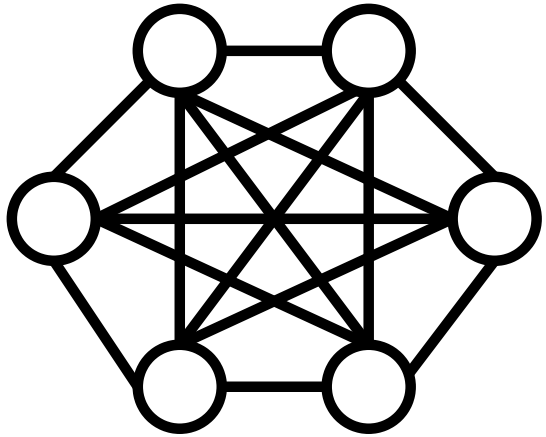
Outline

Motivation

- Replace matrix-like memory with neural networks
  - Hopfield network [3] like structure

a_1	v_1
a_2	v_2
a_3	v_3
$\dots$	$\dots$
a_n	v_n





Problem Definition

- Model: Linear threshold function

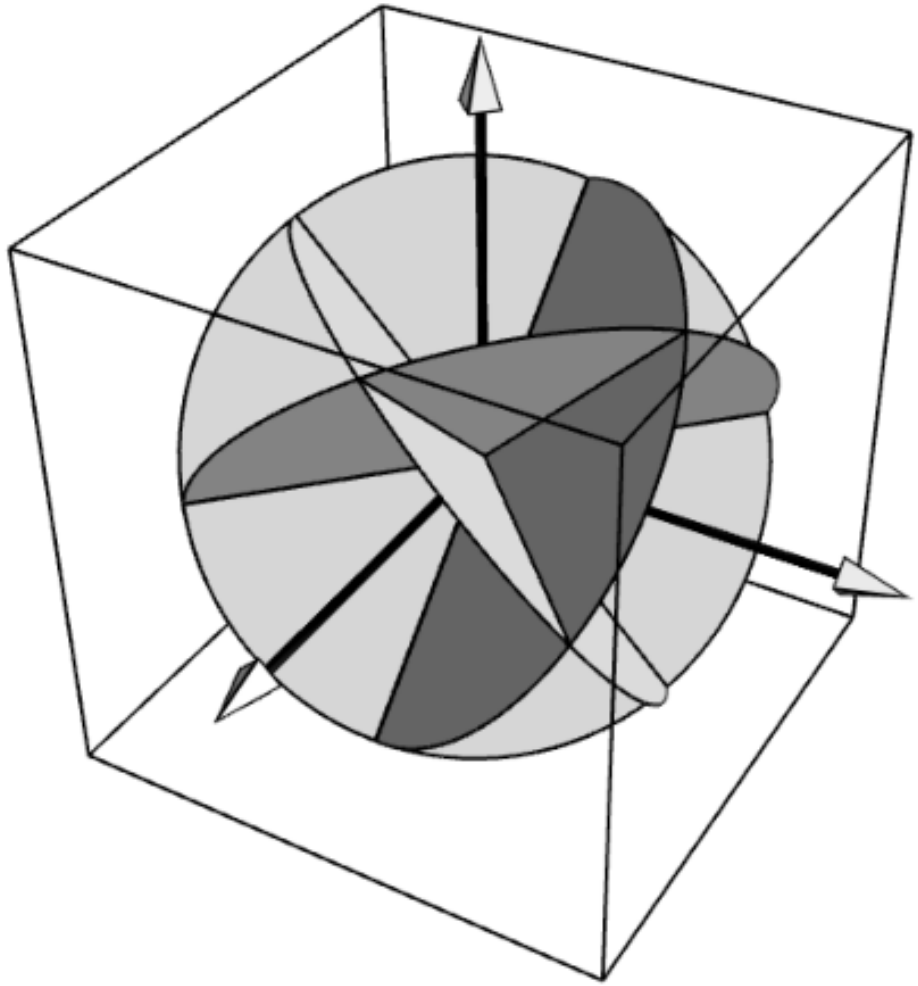
$$\phi(w, x) = \text{sign}(w^T x) \tag{1}$$

- w can be viewed as one row of an associative memory matrix
- Data (x, y): $x \in \{-1, +1\}^N$, $y \in \{-1, +1\}$
- Data came as a stream: $\langle (x^{(1)}, y^{(1)}) ... (x^{(T)}, y^{(T)}) \rangle$
  - Assume linear separability
- Objective: find a learning rule Λ such that at time $t + 1$, Equation 1 is satisfied for all $1, ..., t + 1$ by

$$w^{(t+1)} = \Lambda(w^{(t)}, x^{(t+1)}, y^{(t+1)})$$

Visualize in 3D

- Wire frame: N dimensional hypercube
- Arrow: axis
- Shaded disk: null planes
- Dichotomy: one partition of dataset into 2 parts



Span Rule

- Hypothesis: Λ exists with the form of a span rule:

$$\begin{aligned} w^{(t+1)} &= \Lambda(w^{(t)}, x^{(t+1)}, y^{(t+1)}) \\ &= \alpha^{(t+1)} w^{(t)} + \beta^{(t+1)} x^{(t+1)} \end{aligned}$$

- Goal: Find the following formulae:

$$\begin{aligned} \alpha^{(t+1)} &= f_\alpha(w^{(t)}, x^{(t+1)}, y^{(t+1)}) \\ \beta^{(t+1)} &= f_\beta(w^{(t)}, x^{(t+1)}, y^{(t+1)}) \end{aligned}$$

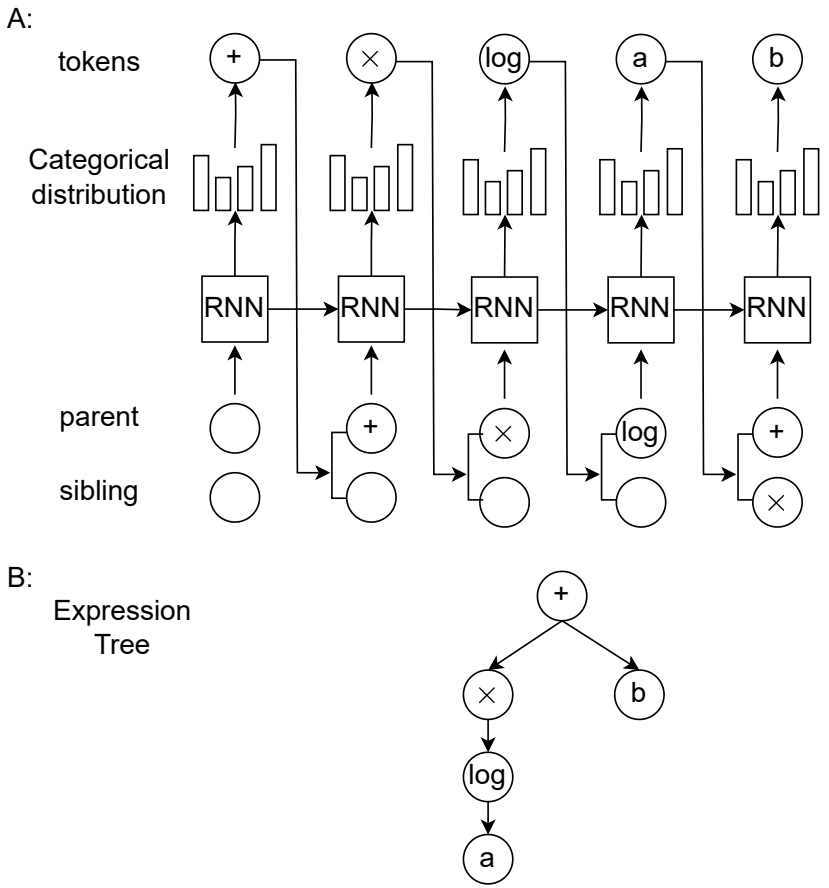
Deep Symbolic Regression [6]

Reinforcement Learning for Symbolic Regression

- REINFORCE [5]
- RNN-based policy
- Risk-seeking policy gradient
- Sparse reward signal

Design Choices

- Pre-order traversal for expression generation
- Domain-agnostic constraints
- Out task related reward definition:
  - Negative reward penalizing invalid expressions
  - Cosine similarity between target and output



Deep Symbolic Regression [6]

Algorithm DSR training loop

while not converged **do**

Sample expressions $\tau \sim p_\theta(\tau)$ with constraints

Compute rewards $\mathcal{R}(\tau)$

Select top $(1-\epsilon)$ quantile: $\mathcal{R}_\epsilon$

Apply risk-seeking selection

Compute policy gradient $\hat{g}_1$

Compute entropy regularization $\hat{g}_2$

Update: $\theta \leftarrow \theta + \eta(\hat{g}_1 + \hat{g}_2)$

end while

Deep Symbolic Regression [6]

Algorithm Reward definition

Require: traversal τ, target weight value w^{t+1}, invalid expression punish $L_i < 0$, dimension mismatch punish $L_d < 0$, reward scale $C_r \geq 1$;
$penalty \leftarrow 0$, $r \leftarrow 0$;
if τ is invalid **then**
    $penalty \leftarrow$ penalty $+L_i$;
else if dimension not match for any operation in τ **then**
    $penalty \leftarrow$ penalty $+L_d$;
end if
$\hat{w}^{t+1} \leftarrow$ compile_traversal(τ);
if $penalty == 0$ **then**
    $r \leftarrow C_r \times$ cosine_similarity($\hat{w}^{t+1}, w^{t+1}$);
end if
return r, $penalty$;

Experiment Setup

- **Training setup**
  - Train the model on $N = \{3, 4, 5\}$
  - Extract best-performing equations
  - Test those equations on $N = \{6, 7, 8\}$
- **Transition Test**
  - Select a dichotomy d from higher dimension
  - Initialize w such that it separates d
  - Iteratively sample adjacent d' and compute corresponding w'
- **Consistency check**
  - Verify whether w' satisfies constraints for d'

Transition Test Algorithm

Algorithm Weight transition test

Choose a dichotomy d randomly, and initialize w such that w separates d.

repeat ▷ for T transitions

Randomly select one d's adjacent dichotomy d'.

Select the pair $(x_m^{[d']}, y_m^{[d']})$ where d and d' disagree

$\hat{w}' \leftarrow \Lambda(w, x_m^{[d']}, y_m^{[d']})$

if $\hat{w}'$ does not satisfy all data in d' **then**

return Fail

end if

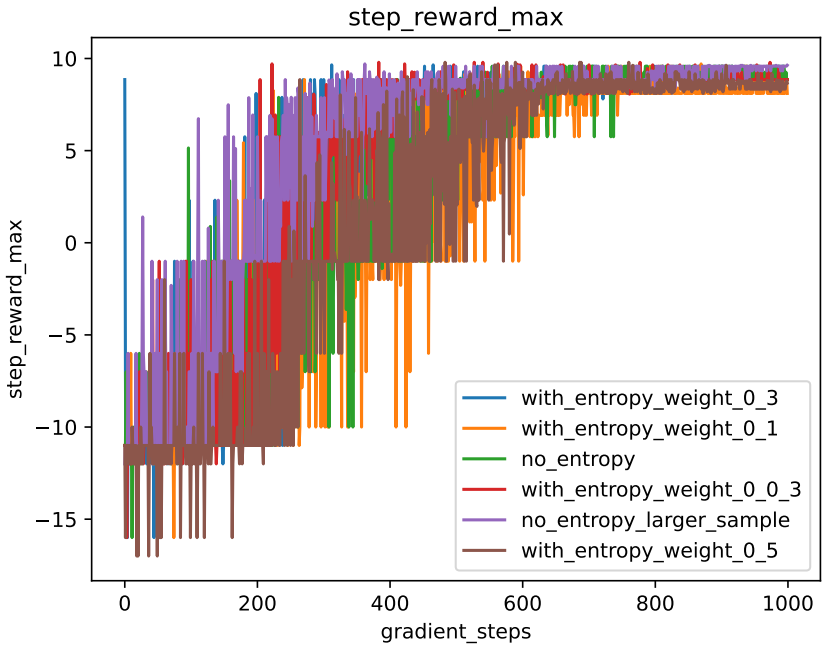
$d \leftarrow d'$

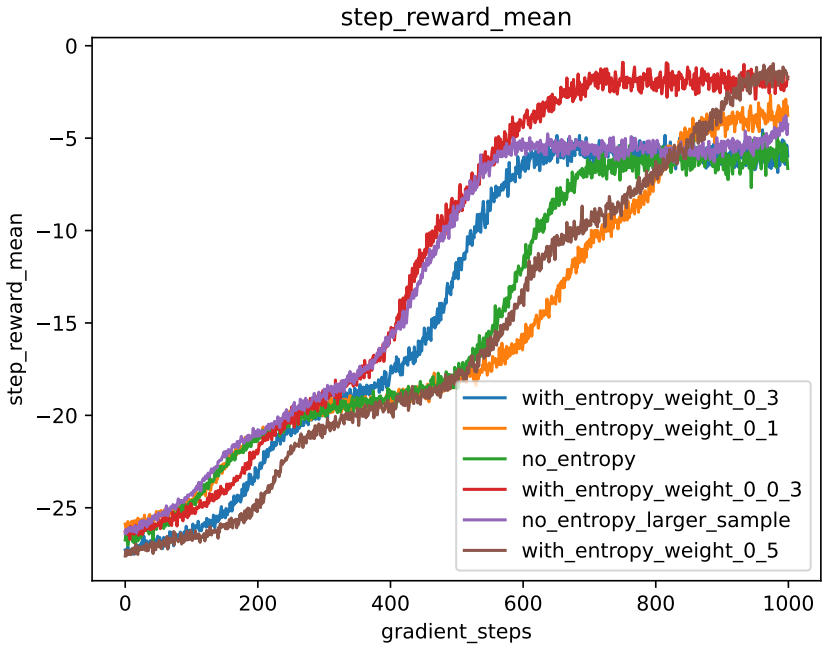
$w \leftarrow \hat{w}'$

until Done

return Success

Training Curves





Conclusion

Empirical finding

Models learned in low dimensions do not generalize to higher dimensions.

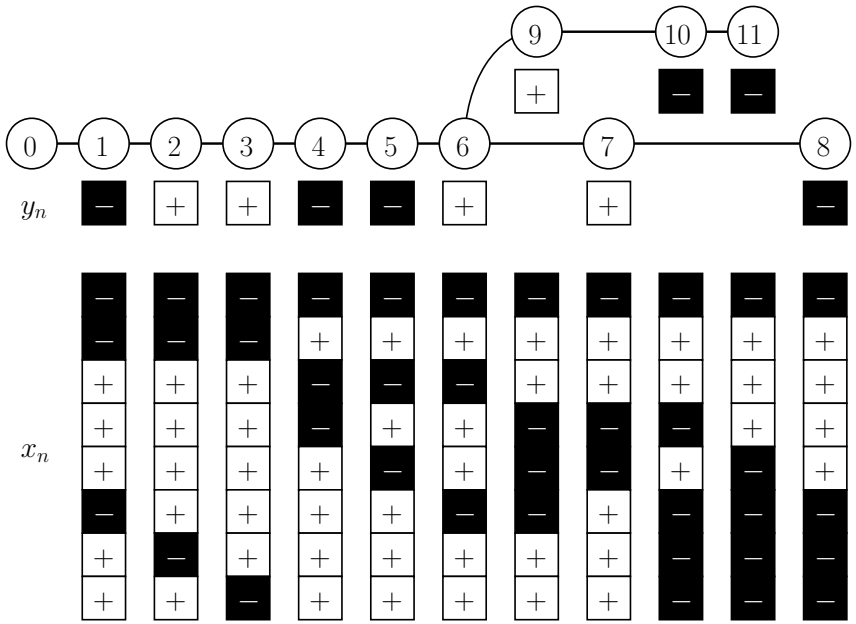
Theoretical finding

A subsequent theoretical analysis showed that such a rule does not exist [7].

Proof Idea

Considering all possible streams with length T, arrange these stream in a tree $\mathcal{T}$. Denote the stream of all samples "seen so far" at node n by $\mathcal{D}_n$, i.e.,

$$\mathcal{D}_n = \langle (x_{n_1}, y_{n_1}), ..., (x_{n_T}, y_{n_T}) \rangle,$$



Proof Idea

- **Supposition 4.1**: There exist a set of vectors w_n and scalars $\alpha_n > 0$, $\beta_n \in \mathbb{R}$, such that for every non-root node n in $\overline{\mathcal{T}}$,

$$\begin{aligned} &\forall (x,y) \in \mathcal{D}_n \quad w_n^\top xy > 0, \\ &w_n = \alpha_n w_{p(n)} + \beta_n x_n \end{aligned} \quad \Rightarrow \quad \begin{aligned} &\forall (x,y) \in \mathcal{D}_n \quad u_n^\top xy > 0, \\ &u_n = u_{p(n)} + \gamma_n x_n \end{aligned}$$

- Feasibility corresponds to a linear program with objective

$$\min_n \min_{(x,y)\in\mathcal{D}_n} u_n^\top xy.$$

Proof Idea

- **Proof by induction through contrapositive**:
  Consider any node n in $\overline{\mathcal{T}}_N$ with training stream $\mathcal{D}_n = \langle z_1, z_2, ..., z_n \rangle$, where $z_t = x_t \cdot y_t$.
  Proposition 4.2: For any arbitrary $N \in \mathbb{N}$, if $\overline{\mathcal{T}}_N$ is infeasible, then $\overline{\mathcal{T}}_{N+1}$ is infeasible.

$$\mathcal{D}_{\hat{n}} = \left\langle \begin{bmatrix} z_1 \\ -1 \end{bmatrix}, \begin{bmatrix} z_1 \\ +1 \end{bmatrix}, ..., \begin{bmatrix} z_n \\ -1 \end{bmatrix}, \begin{bmatrix} z_n \\ +1 \end{bmatrix} \right\rangle,$$

Proof Idea

- Suppose $\overline{\mathcal{T}}_{N+1}$ is feasible, we have:

$$u_{\hat{n}} = u_{\hat{p}} + \gamma_{\hat{n}_-} \begin{bmatrix} z_n \\ -1 \end{bmatrix} + \gamma_{\hat{n}_+} \begin{bmatrix} z_n \\ +1 \end{bmatrix}$$

which also satisfy:

$$u_{\hat{n}}^\top \begin{bmatrix} z_t \\ -1 \end{bmatrix} > 0 \quad \text{and} \quad u_{\hat{n}}^\top \begin{bmatrix} z_t \\ +1 \end{bmatrix} > 0$$

$$\Downarrow$$

$$\check{u}_{\hat{n}}^\top z_t > 0$$

where $\check{u}_{\hat{n}} \in \mathbb{R}^N$ is the vector containing the first N entries of $u_{\hat{n}}$

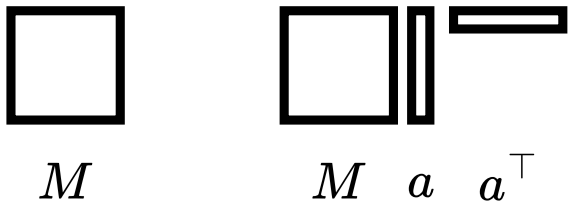
$\therefore$ if $\overline{\mathcal{T}}_{N+1}$ is feasible then $\overline{\mathcal{T}}_N$ feasible $\Rightarrow$ if $\overline{\mathcal{T}}_N$ infeasible, then $\overline{\mathcal{T}}_{N+1}$ is infeasible. Counterexample found in $N = 8$

Outline

Motivation

- *Flatten* the memory using vector symbolic architectures(VSAs)

	Matrix	VSA
Read:	$v \leftarrow \sigma(Ma)$	$v \leftarrow a^{(m)} \ \textcircled{\#} \ \mathcal{M}$
Write:	$M \leftarrow M - \sigma(Ma)a^\top/N + va^\top/N$	$\mathcal{M} \leftarrow \mathcal{M} - (a \circledast v^{\text{old}}) + (a \circledast v^{new})$



Vector Symbolic Architecture (VSA)

- Hypervectors (HVs): random vectors with very large dimensions.
  - **Codebook** is needed, mapping symbols to HVs
- Binding and unbinding:
  - **Binding** associates two vectors, analogous to forming a key-value pair.
  - **Unbinding** retrieves the value associated with a given key.
- Superposition: combining multiple pairs of vectors, analogous to storing multiple key-value pairs in an associative array.
- Permutation: reordering elements in the HVs.

Holographic Reduced Representations (HRRs) [8]

- Codebooks: Gaussian ($\mathcal{N}(0, \frac{1}{N})$) or Binary ($\pm\frac{1}{\sqrt{N}}$)
- Binding and unbinding: circular convolution($\circledast$) and circular correlation ($\circledcirc$).
  - Implementation: Fast Fourier Transform

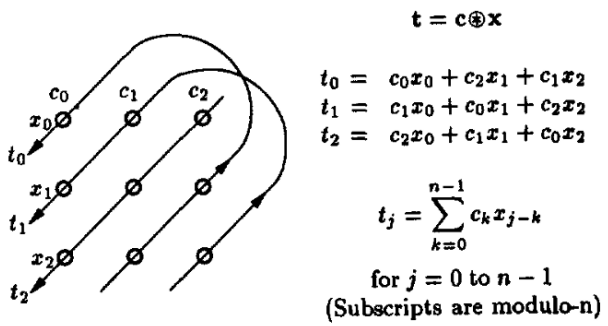


Fig. 4. Circular convolution represented as a compressed outer product for $n = 3$.

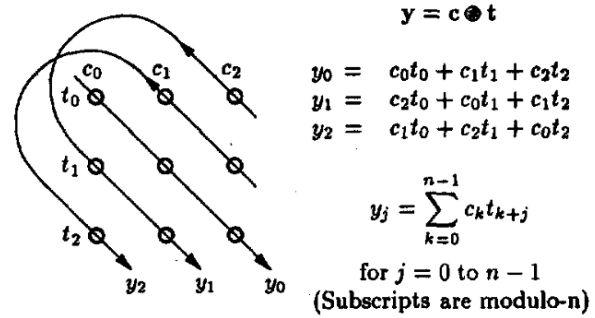


Fig. 5. Circular correlation represented as a compressed outer product for $n = 3$.

- Superposition: element-wise addition.
  - E.g. the vector $t = (a \circledast u) + (b \circledast v)$ represents an associative array containing two key-value pairs: (a, u) and (b, v)

Problem: Quadratic Complexity of Cleanup

- Retrieval of a vector from a binding of K pairs of vectors:

$$\hat{u}_k = a_k \circledast \sum_{i=0}^{K-1} (a_i \circledast u_i)$$

  Note that $\hat{u}$ is **noisy**.

- A standard way to clean-up $\hat{u}$ is comparison with the codebook (V):

$$u^* = V_{i*} \quad \text{where} \quad i = \arg\max_{i^*} (V\hat{u})_i$$

1. Dot-product similarity, cosine similarity and Hamming distance for binary vectors. $\Rightarrow$ $\mathcal{O}(N^2)$ [4, 9, 10].
2. Random linear codes [11] $\Rightarrow$ $\mathcal{O}(N \cdot |V|)$ or $\mathcal{O}(N^2 - N \log|V|)$,

Methodology: `krop`

Codebook based on **Hadamard Matrices** using Sylvester's construction and **Kroneker Rotation Products**:

$$\tilde{H}^{(0)} = [1], \quad \tilde{H}^{(k+1)} = \begin{bmatrix} \tilde{H}^{(k)}\cos(\theta_k) & \tilde{H}^{(k)}\sin(\theta_k) \\ \tilde{H}^{(k)}\sin(\theta_k) & -\tilde{H}^{(k)}\cos(\theta_k) \end{bmatrix}$$

$$\Rightarrow \quad \tilde{H}^{(K)} = \bigotimes_{k=1}^{K} \begin{bmatrix} \cos(\theta_{K-k}) & \sin(\theta_{K-k}) \\ \sin(\theta_{K-k}) & -\cos(\theta_{K-k}) \end{bmatrix}$$

where θ_k is sampled uniformly from $(0, 2\pi)$.

- Rows and columns are mutually orthogonal (as original Hadamard Matrices)

Methodology: `krop`

Divide and Conquer: Time complexity: $\mathcal{O}(N \log N)$.

$$
\begin{aligned}
\tilde{H}^{(K)} u &= \begin{bmatrix} \tilde{H}^{(K-1)} c_{K-1} & \tilde{H}^{(K-1)} s_{K-1} \\ \tilde{H}^{(K-1)} s_{K-1} & -\tilde{H}^{(K-1)} c_{K-1} \end{bmatrix} \begin{bmatrix} \hat{u} \\ \check{u} \end{bmatrix} \\
&= \begin{bmatrix} \tilde{H}^{(K-1)} (c_{K-1} \hat{u} + s_{K-1} \check{u}) \\ \tilde{H}^{(K-1)} (s_{K-1} \hat{u} - c_{K-1} \check{u}) \end{bmatrix} \\
&= \begin{bmatrix} \tilde{H}^{(K-1)} U_0^{(K-1)} \\ \tilde{H}^{(K-1)} U_1^{(K-1)} \end{bmatrix}
\end{aligned}
\quad \rightarrow \quad
\begin{bmatrix} \tilde{H}^{(K-2)} U_0^{(K-2)} \\ \tilde{H}^{(K-2)} U_1^{(K-2)} \end{bmatrix}, \quad
\begin{bmatrix} \tilde{H}^{(K-2)} U_2^{(K-2)} \\ \tilde{H}^{(K-2)} U_3^{(K-2)} \end{bmatrix}
\quad \rightarrow \quad \cdots \quad \rightarrow \quad
\begin{bmatrix} \tilde{H}^{(0)} U_0^{(0)} \\ \vdots \\ \tilde{H}^{(0)} U_{N-1}^{(0)} \end{bmatrix}
$$

Advantages for `krop`

- **No explicit codebook storage**

$$\text{store only } \theta \to \mathcal{O}(\log N) \text{ space}$$

- **Efficient vector reconstruction**

$$\mathcal{O}(N) \text{ time}$$

- **Faster clean-up**

$$\mathcal{O}(N^2) \quad \longrightarrow \quad \mathcal{O}(N \log N)$$

Vector Reconstruction in $\mathcal{O}(N)$

Algorithm Codebook vector reconstruction procedure

Require: Codebook parameters $\theta = [\theta_0, \ldots, \theta_{K-1}]$, reconstruction index i
Ensure: Reconstructed codebook vector v

$v \leftarrow [1]$
for $k = 0$ to $K - 1$ **do**
    $b \leftarrow (i \gg k)\ \&\ 1$
    if $b = 0$ **then**
        $v \leftarrow [\cos(\theta_k)v,\ \sin(\theta_k)v]$
    else
        $v \leftarrow [\sin(\theta_k)v,\ -\cos(\theta_k)v]$
    end if
end for
return v

krop Clean-up Algorithm

Algorithm krop clean-up procedure

Require: Codebook parameters $\theta = [\theta_0, \ldots, \theta_{K-1}]$, vector u

Ensure: Reconstructed index

$U_0^{(K)} \leftarrow u$

for $k = K$ down to 1 **do**

  for $i = 0$ to $N/2^k - 1$ **do**

    $U_{2i}^{(k-1)} \leftarrow c_{k-1}\hat{U}_i^{(k)} + s_{k-1}\check{U}_i^{(k)}$

    $U_{2i+1}^{(k-1)} \leftarrow s_{k-1}\hat{U}_i^{(k)} - c_{k-1}\check{U}_i^{(k)}$

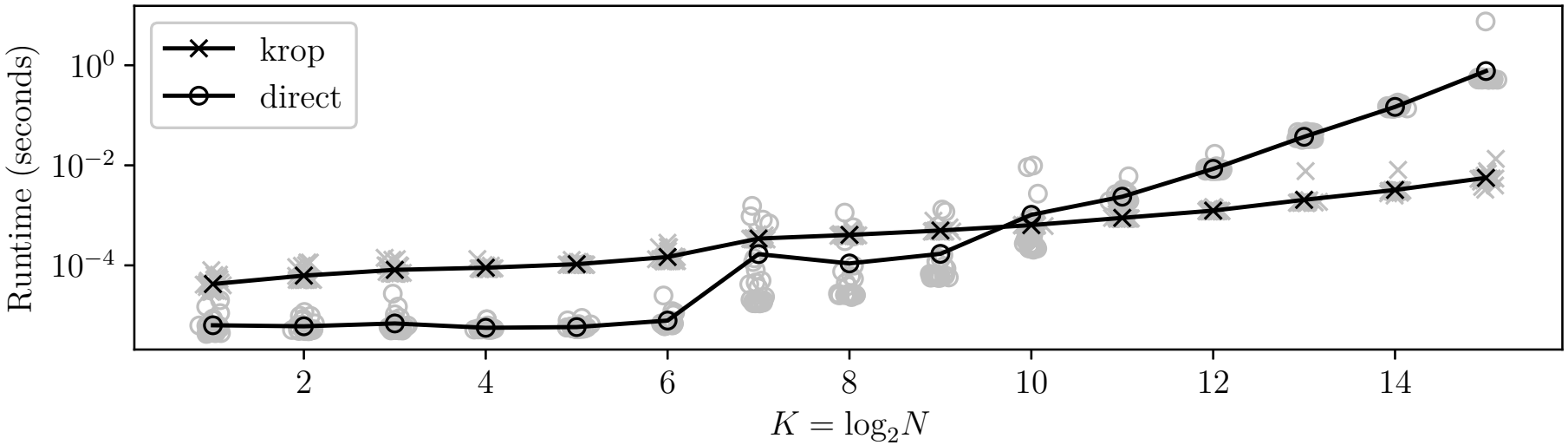
  end for

end for

return $\arg\max_i U_i^{(0)}$

Clean-up Efficiency

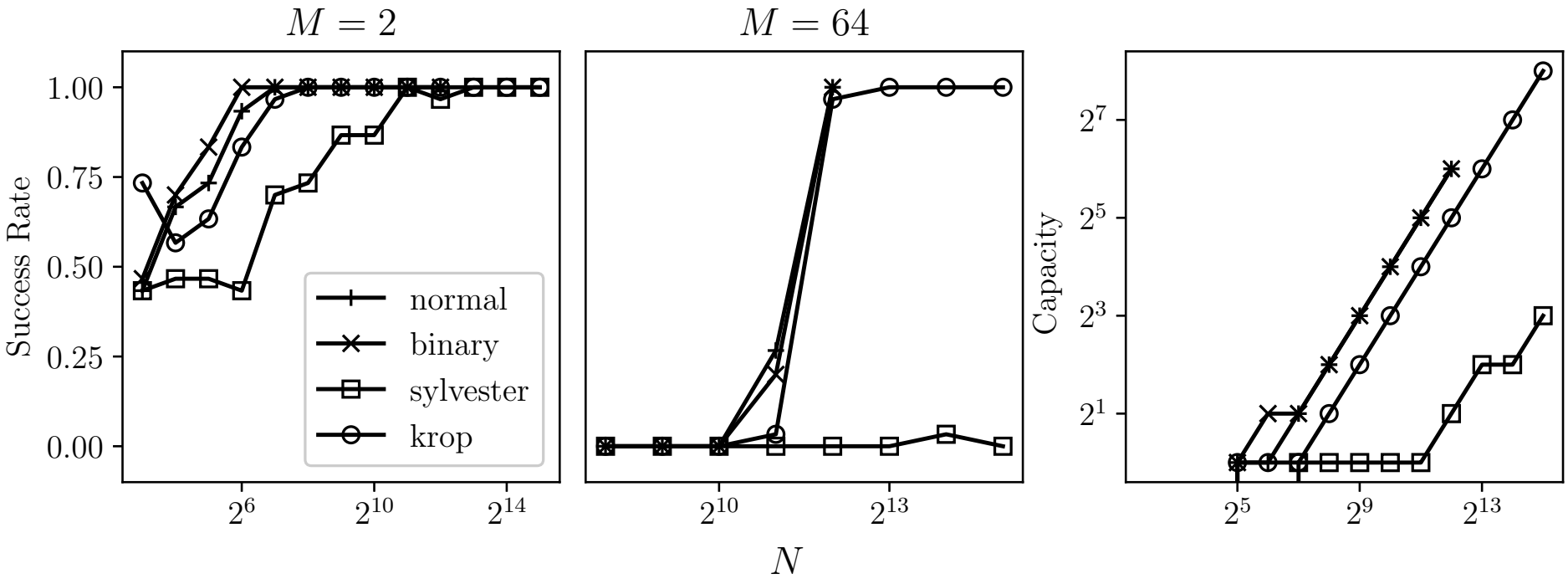
We timed the runtime of `krop` and direct matrix-vector multiplication for cleaning-up noisy vectors retrieved from HRR. The experiment for each K was repeated for **30** times.



Memory Capacity

- Forming memory $\mathcal{M} = \sum_{m=1}^{M} a^{(m)} \circledast v^{(m)}$ using HRR, for M key-value pairs $(a^{(m)}, v^{(m)})$
- 4 codebooks for value vector, address vectors from $\mathcal{N}(0, \frac{1}{N})$
- Success Rate:

$$\frac{1}{M} \sum_{m=1}^{M} \mathbb{1} \left[\text{clean-up}(a^{(m)} \circledcirc \mathcal{M}) = v^{(m)} \right]$$



Mutable Key-Value Memory

- Testing HRR-formed memory in a read-write task with 3 different memory forms:
  - `krop`: `krop` embeddings and clean-up

$$v^{\text{old}} \leftarrow \texttt{krop\_cleanup}(a^{(t)} \circledast \mathcal{M}_t^{\texttt{krop}})$$
$$\mathcal{M}_{t+1}^{\texttt{krop}} \leftarrow \mathcal{M}_t^{\texttt{krop}} - (a^{(t)} \circledast v^{\text{old}}) + (a^{(t)} \circledast v^{(t)})$$

  - sign: $\pm\frac{1}{\sqrt{N}}$ embeddings and sign-based clean-up

$$v^{\text{old}} \leftarrow \text{sign}(a^{(t)} \circledast \mathcal{M}_t^{\text{sign}})/\sqrt{N}$$
$$\mathcal{M}_{t+1}^{\text{sign}} \leftarrow \mathcal{M}_t^{\text{sign}} - (a^{(t)} \circledast v^{\text{old}}) + (a^{(t)} \circledast v^{(t)})$$

  - none: $\mathcal{N}(0, \frac{1}{N})$ embeddings and no clean-up

$$v^{\text{old}} \leftarrow a^{(m)} \circledast \mathcal{M}_t^{\text{none}}$$
$$\mathcal{M}_{t+1}^{\text{none}} \leftarrow \mathcal{M}_t^{\text{none}} - (a^{(t)} \circledast v^{\text{old}}) + (a^{(t)} \circledast v^{(t)})$$

Mutable Key-Value Memory

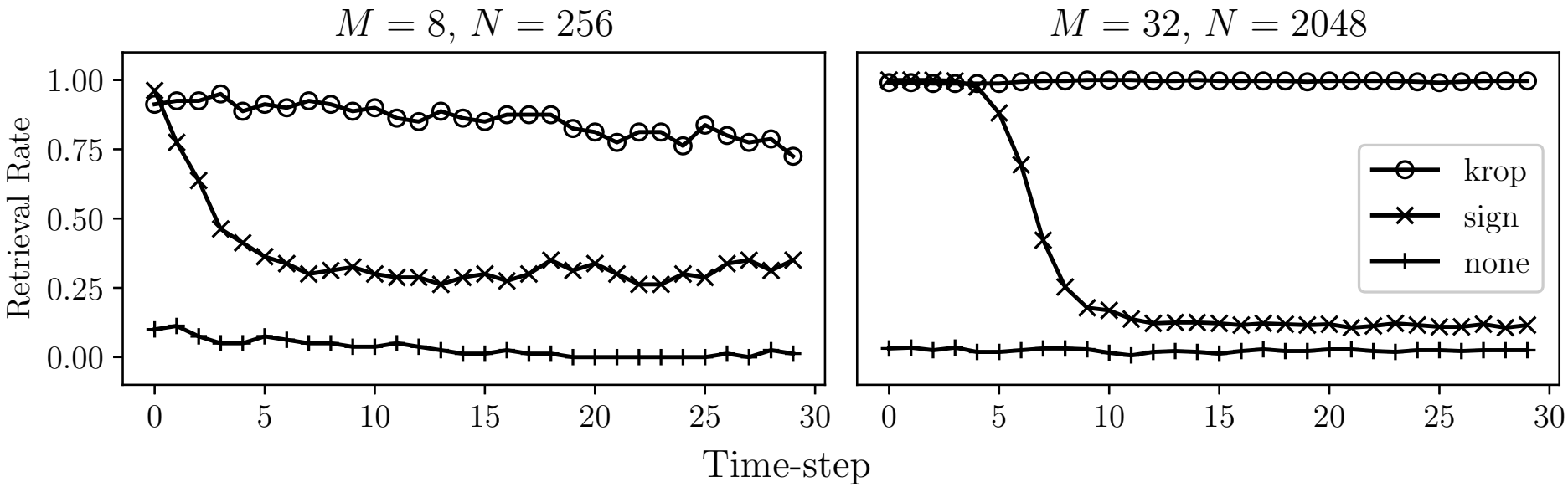


Figure: Examples of mutable VSA memory retrieval rates by time-step.

Mutable Key-Value Memory

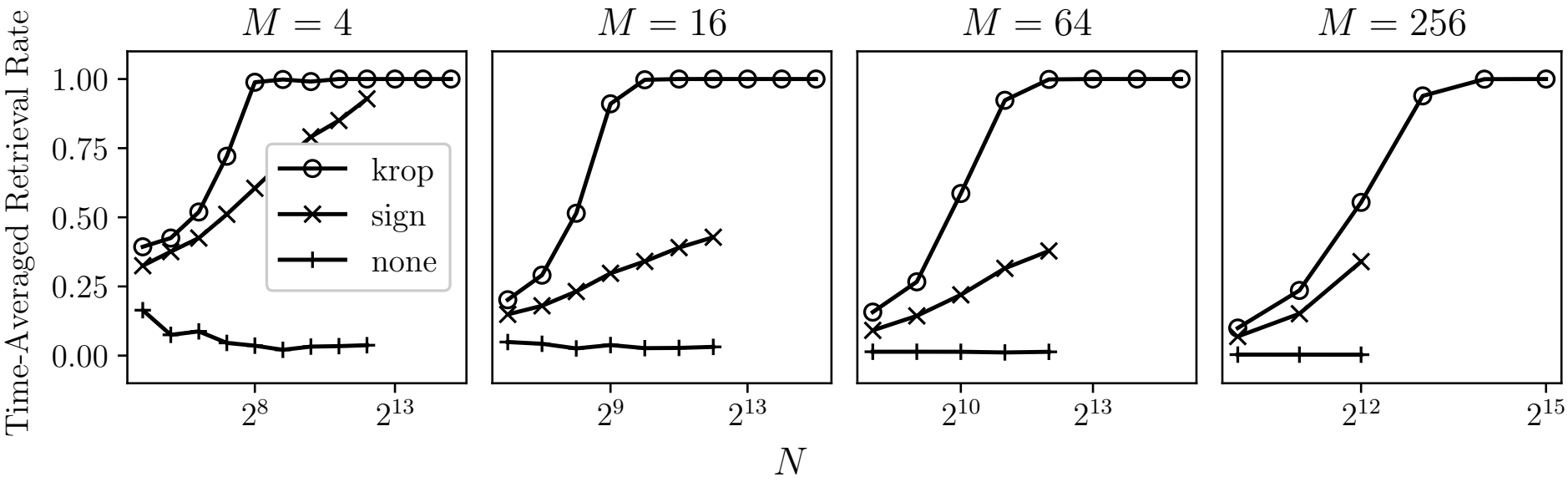


Figure: Mutable VSA memory retrieval rates averaged over 10 independent trials and 30 read-write steps.

Limitations

- **Distribution mismatch**
  - Real-world data may differ from the `krop` codebook assumptions
- **Non-differentiability**
  - Relys on the argmax operator
  - Limits compatibility with gradient-based optimization
- **High dimensional requirement**
  - Sufficient dimension is very large
  - Problematic for iterative training settings
- **Pointer Issue**
  - Data stored in the memory have no types.

Thank You

Questions and Discussion

Ruipeng Liu

Garrett Katz

References

A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," 2014. [Online]. Available: https://arxiv.org/abs/1410.5401

G. E. Katz, G. P. Davis, R. J. Gentili, and J. A. Reggia, "A programmable neural virtual machine based on a fast store-erase learning rule," *Neural Networks*, vol. 119, pp. 10–30, 2019.

J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities." *Proceedings of the National Academy of Sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

D. Kleyko, D. A. Rachkovskij, E. Osipov, and A. Rahimi, "A survey on hyperdimensional computing aka vector symbolic architectures, part I: Models and data transformations," *ACM Computing Surveys*, vol. 55, no. 6, pp. 1–40, 2022.

R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, pp. 229–256, 1992.

B. K. Petersen, M. L. Larma, T. N. Mundhenk, C. P. Santiago, S. K. Kim, and J. T. Kim, "Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=m5Qsh0kBQG

References


R. Liu, B. He, N. Tahir, and G. E. Katz, "On the feasibility of single-pass full-capacity learning in linear threshold neurons with binary input vectors," in *Proceedings of the 41st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp, Eds., vol. 235. PMLR, 21–27 Jul 2024, pp. 31 119–31 130. [Online]. Available: https://proceedings.mlr.press/v235/liu24x.html


T. A. Plate, "Holographic reduced representations," *IEEE Transactions on Neural Networks*, vol. 6, no. 3, pp. 623–641, 1995.


A. Thomas, S. Dasgupta, and T. Rosing, "A theoretical perspective on hyperdimensional computing," *Journal of Artificial Intelligence Research*, vol. 72, pp. 215–249, 2021.


T. Yu, Y. Zhang, Z. Zhang, and C. M. De Sa, "Understanding hyperdimensional computing for parallel single-pass learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 1157–1169, 2022.


N. Raviv, "Linear codes for hyperdimensional computing," *Neural Computation*, vol. 36, no. 6, pp. 1084–1120, 2024.