Semiconductor Research Corporation

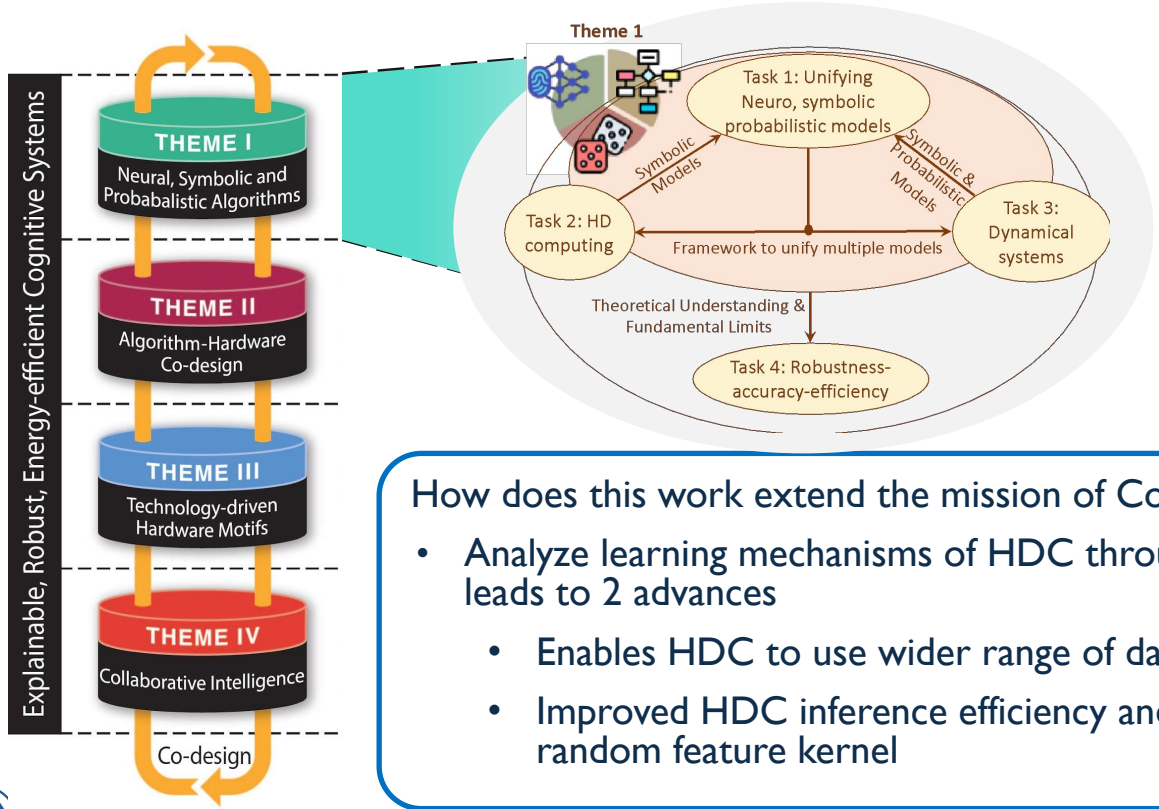
CoCoSys
CENTER FOR THE CO-DESIGN OF COGNITIVE SYSTEMS

Connecting Kernel Methods and Hyperdimensional Computing for Capable and Efficient Learning

Quanling Zhao, PI Tajana Rosing

University of California, San Diego (UCSD)

Relevance to the Center Goals



Theme 1 Mission Statement

A principled approach to unified neuro-symbolic-probabilistic algorithms, supported by new information representations, computing models and analysis of fundamental limits, has the potential to radically advance the next generation of cognitive algorithms

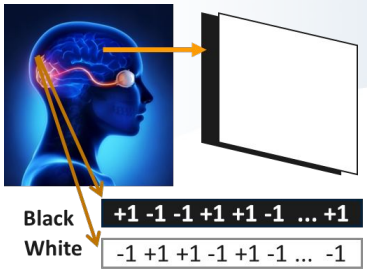
How does this work extend the mission of CoCoSys?

- Analyze learning mechanisms of HDC through the lens of kernel method, leads to 2 advances
  - Enables HDC to use wider range of data appropriate kernel
  - Improved HDC inference efficiency and throughput via multiplexed random feature kernel

SRC

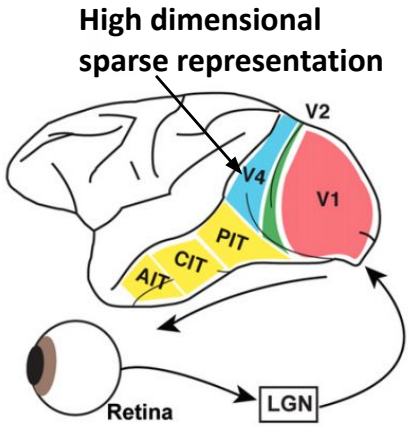
Hyperdimensional Computing (HDC)

Dense sensory input is mapped to
high-dimensional sparse representation
on which brain operates [Neuron'14]



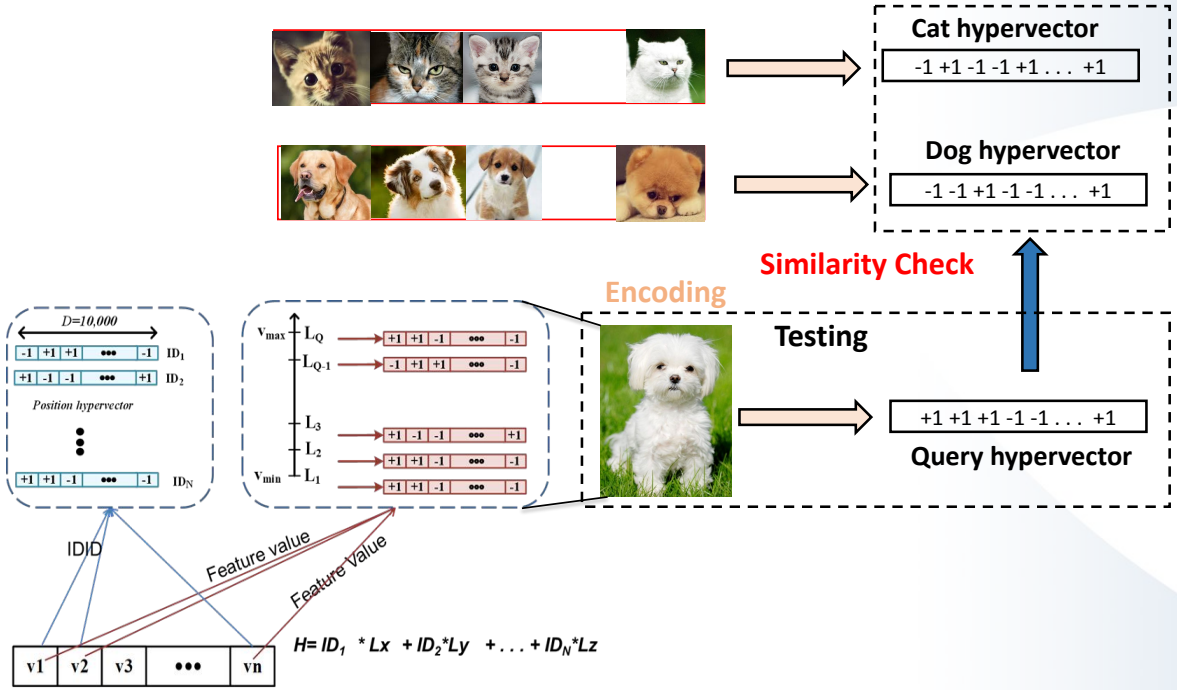
Hyperdimensional (HD) computing:

- Encodes data into hypervectors
- Leverages full algebra and works on well-defined set of operations that are easy to parallelize
- Fast single-pass training
- Supports symbolic reasoning & explainability
- Energy-efficient & robust to noise

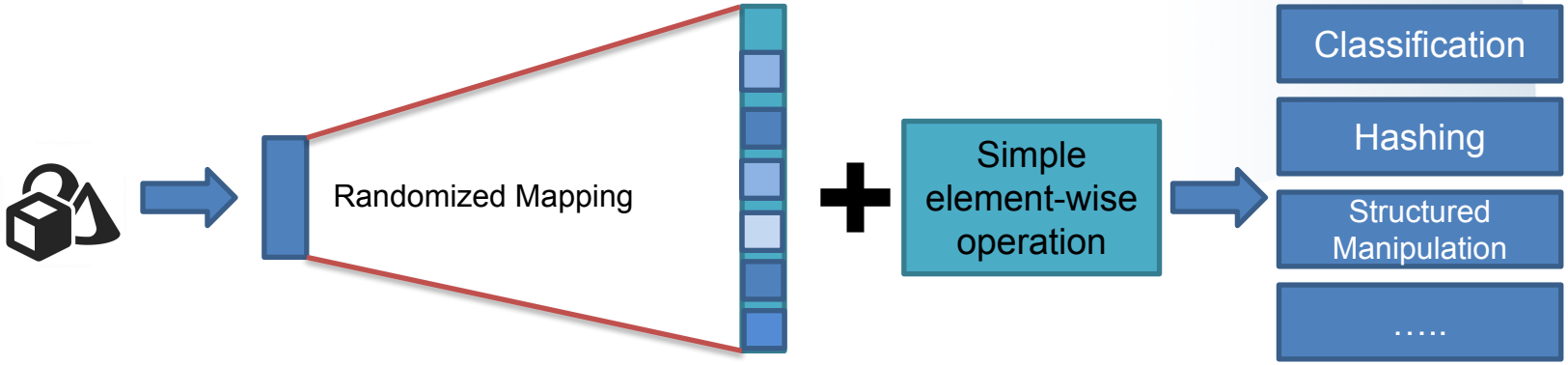


SRC

A “Canonical” Classification Example



Properties of HDC



- Diverse encoding function
- Simple, robust computation

Rich representation

Algorithmic Efficiency

Hardware Awareness

HDC can be viewed as a single, unified framework that delivers rich representations, algorithmic efficiency, and hardware-friendly operations together as one package.

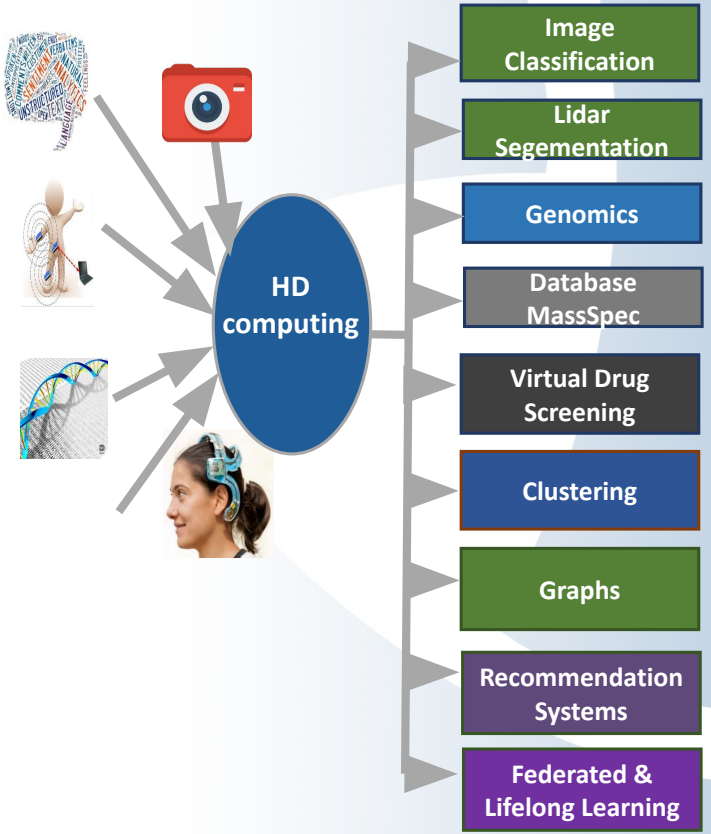
Success of HDC

Efficient and low-cost approach for various ML tasks

- Classification, regression
- Learning and reasoning
- AI applications under resource constraints
- Much more ….

Specialized hardware design for large scale processing

- Bio-sequence processing and search
- Large scale knowledge graph inference
- Accelerators with in/near memory Computation
- Much more ….

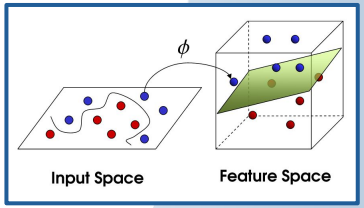


SRC

Roadmap of this talk

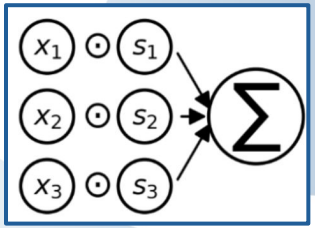
1: Kernel method view of HDC and Learning more expressive HDC model

Bridging the Gap Between Hyperdimensional Computing and Kernel Methods via the Nyström Method *[AAAI'25]*



2: Improved HDC inference efficiency via approximate kernel inference

Superposed Inference for Hyperdimensional Computing via Multiplexed Random-Feature Kernels *[ICLR'27 WIP]*

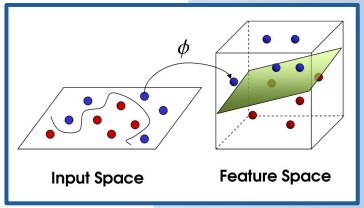


SRC

Roadmap of this talk

1: Kernel method view of HDC and Learning more expressive HDC model

Bridging the Gap Between Hyperdimensional Computing and Kernel Methods via the Nyström Method *[AAAI'25]*



2: Improved HDC inference efficiency via approximate kernel inference

Superposed Inference for Hyperdimensional Computing via Multiplexed Random-Feature Kernels *[ICLR'27 WIP]*

SRC

Learning with HDC representation

- **Class are represented as linear combination of encoded training samples**

Training Data


$$\theta_{dog} = \sum_{dogs} w_i \phi(dog_i)$$


$$\theta_{cat} = \sum_{cats} w_i \phi(cat_i)$$

Class vector

Sample weight

Encoding function

Inference

$$\text{argmax}_{l \in \{cat, dog\}} (\phi(\text{image}) \cdot \theta_l)$$

Encode unknown image

Compare with class representation

- Weight can be tuned via perceptron algorithm
- HDC encoding maps “similar” data close to each other
  - **Encoding function needs to capture some salient notion of similarity**

SRC

HDC from Kernel Method Perspective

- Kernel functions: how to measure similarity between data
- Kernel machines (like SVMs, Gaussian process) learn a linear function feature space $\mathcal{H}$
  - HDC learns a linear function in the space of encodings

Sample weight

Encoding of reference samples

$$f(x) = \sum_{i=1}^{n} \alpha_i \, k(x_i, x) \;\Rightarrow\; f(x) = \langle w, \phi(x) \rangle_{\mathcal{H}}$$

$$w = \sum_{i-1}^{n} \alpha_i \phi(x_i)$$

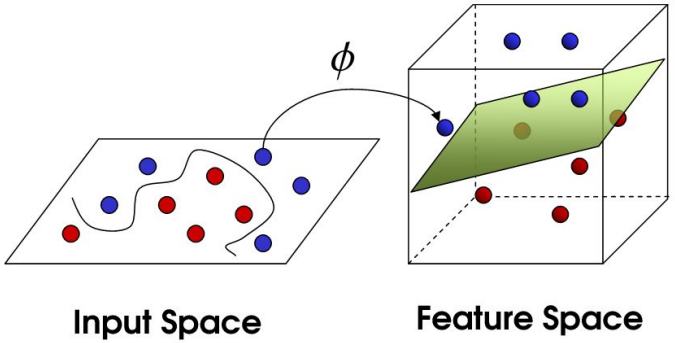
Kernel machine Through **Mercer's theorem**

Exact form HDC class prototype takes

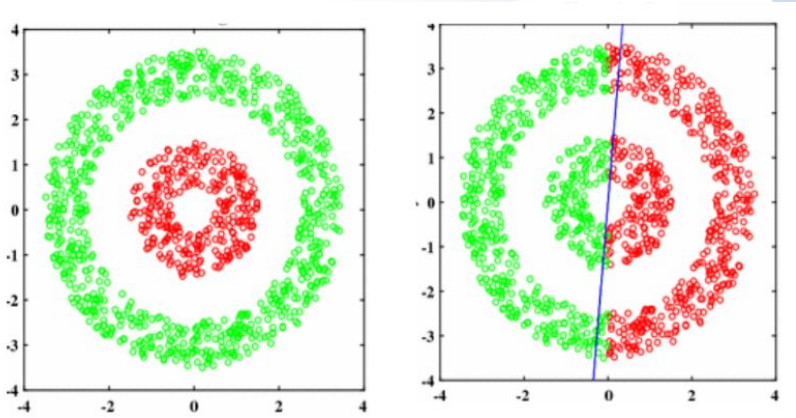
A "richer" kernel (encoding function) corresponds to a feature space where a **simple linear separator** can solve harder problems

SRC

Importance of Encoding Function



Gaussian Kernel



- HDC relies the underlying kernel of encoding function to expose salient structure in data
- Choose the correct encoding function is crucial in any HDC models

SRC

Related Works

- For euclidean data [NeurIPS'22, DAC'24]
  - Random Projection / Random Fourier Feature
  - Inner product preserves linear / shift-invariant kernel

Euclidean data is limiting
Not applicable to many useful kernel

- For temporal data [DAC'24,IJCAI'23]
  - Use permutation for temporal data
  - Inner product preserves how much two time-series "overlaps"

Will underperform if time-series misalign

- For symbolic data (graphs, strings) [DATE'22,ACS'23]
  - Usually Ad-hoc approaches that does not generalize
  - Existing approaches captures fairly simple similarities

Often too simple to capture complex relations

Problem Definition

How to **import** broader class of kernel functions into HDC setting?

Given a suitable notion of similarity (kernel) for a certain task, is it possible to generate HDC encoding $\phi : \mathcal{X} \rightarrow \mathcal{H}$ such that:

$$\phi(x) \cdot \phi\left(x'\right) \approx k\left(x, x'\right)$$

HDC encoding function

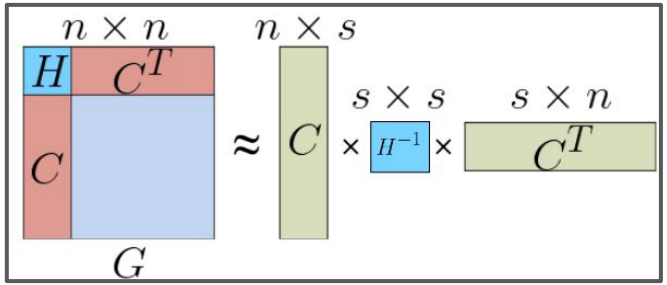
User specified Kernel function

Also satisfies:

- Poses minimal restrictions to the kernel function.
- The representations continue to be amenable to practical applications of HDC
  - Low precision & distributed (noise-resilient)

NysHD: Main Idea

- Generate HDC encodings from Nyström method
  - Nyström method - low rank matrix approximation technique



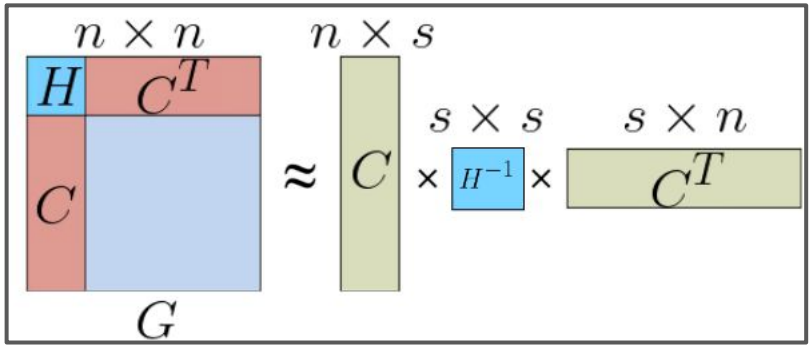
$$CH^{-1}C^T = \hat{G} \approx G$$

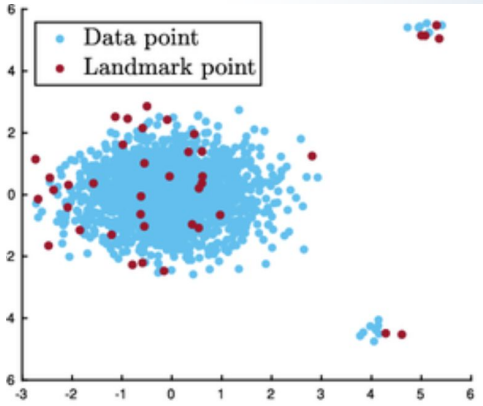
(Pseudo-) inverse

Real matrix

NysHD: Nyström Method for Kernel Matrices

- Generate embeddings by decompose Nyström approximation





- Directly sample from kernel matrix
- Full matrix can be reconstructed
  - Works well with kernel matrices in practice

SRC

Compose with Signed Random Projection

- In general, Nystrom embedding do not need to be high-dimensional
  - Lack other desiderata of HDC like noise robustness
- High dimension ⟺ large landmark set - Problematic
  - Rectify by composing with another technique that preserves angular similarities.

$$sign(Px)$$ ⟵ **Sign-thresholded random projection**

- Similar to random hyperplane hashing / rounding
- Preserves a version of angular kernel (angular distance)

NysHD: Generate Nyström Random Projection

Compose Nystrom method and sign-thresholded random projection

- Input: Dataset and kernel function
- Output: Landmark set and Nystrom Random Projection

Step 1: $\mathcal{Z} \leftarrow$ uniform sampling of s data points from $\mathcal{D}$

Step 2: $(H_{\mathcal{Z}})_{ij} = K(z_i, z_j) \quad Q\Lambda Q^T = H_{\mathcal{Z}}$

Decompose landmark kernel matrix

Step 3: $P_{rp} = [w_1, w_1, \cdots, w_d]^T \in \mathbb{R}^{d \times s}$

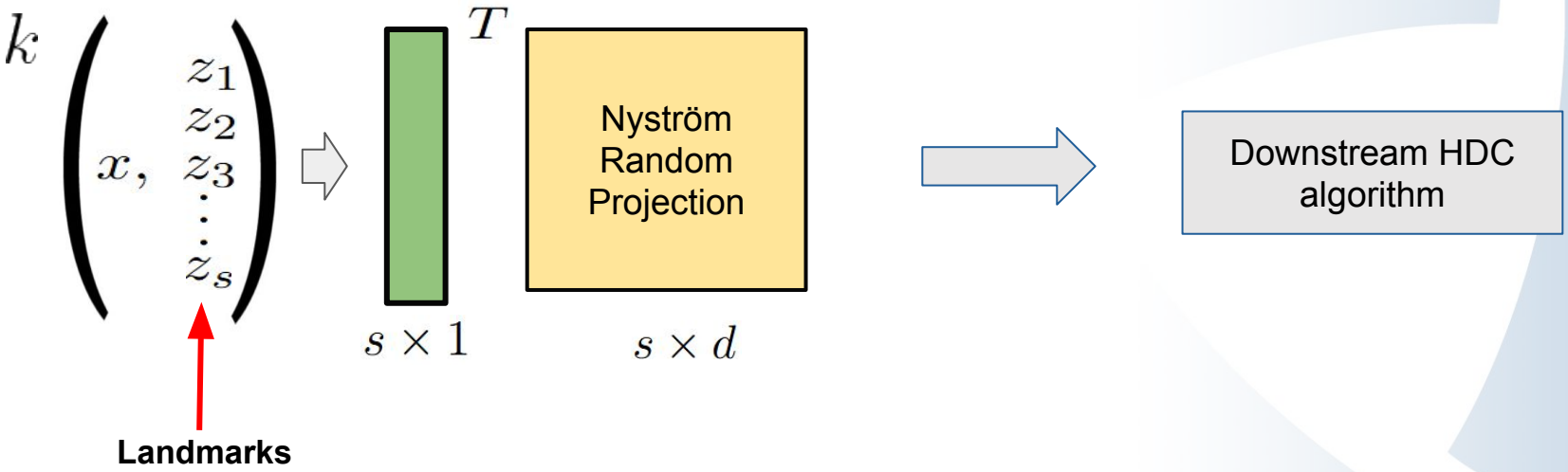
Random hyperplane hashing - sample for s dimensional unit sphere

Step 4: $P_{nys} = P_{rp}\Lambda^{-\frac{1}{2}}Q^T$

Nyström Random Projection

SRC

NysHD: Encoding data



- Can be seen as feature extraction via kernel function
- Any downstream HDC algorithm can then uses the this kernel function implicitly

NysHD: Preserving Normalized Kernel

$$\mathbb{E}\left[\langle \phi(x_i), \phi(x_j) \rangle\right] \approx \frac{\hat{G}_{ij}}{\sqrt{\hat{G}_{ii}\hat{G}_{jj}}}$$

Normalized kernel

Nyström approximation of $k(x_i, x_j)$

- Dot product between NysHD encodings preserves normalized kernel

Complexity

With kernel of complexity of $O(m)$

- Overhead: Kernel evaluations and one-time eigen-decomposition

$$O(\max(s^3, snm, snd))$$

Eigen-decomp

Kernel evaluation

Proj

In typical case where: $s \ll n$

- Second or third term can dominate the complexity
- Core trade-off: Landmark size and efficiency

s - size of landmark

n - size of dataset

d - dimension of hypervector

Evaluation Setup

- NysHD: general-purpose and can be applied to many tasks
  - Import well-designed kernel function into HDC setting
- Evaluate on two classes of tasks
  - Graph classification: protein structure, social graphs
    - 8 graph datasets from TUDataset [Arxiv'2020]
  - String classification: spam detection, Bio-sequences
    - 4 string datasets [BMC bioinfo'23, UCI ML repo]

Propagation kernel for graphs [ML'16]

- Similarity function over attributed graphs
- Based on graph diffusion, use random walk to measures structural similarity.

Gappy kernel for strings [JMLR'04]

- N-gram based similarity measure
- Allows some mismatch within each N-grams

Graph Dataset Results

Best accuracy

Second best accuracy

		NCI1	ENZYMES	D&D	BZR	MUTAG	COX2	NCI109	Mutagenicity	Average
DNN	DGCNN [AAAI'18]	70.2	36.9	74.5	81.5	82.9	78.3	71.1	75.0	71.3
	GCN [ICLR'17]	79.9	60.7	74.8	84.2	85.5	83.1	80.2	79.9	78.5
	GIN [Arxiv'18]	73.4	28.8	67.5	76.4	76.8	78.1	70.8	78.0	68.7
HDC	GIUNet [ESA'24]	72.4	29.9	63.4	78.3	85.0	77.4	68.8	76.5	68.9
	GraphHD [DATE'22]	60.0	23.2	67.6	74.9	85.3	81.9	59.9	59.8	64.1
NysHD	NysHD	73.8	61.3	76.2	82.0	85.5	74.6	71.8	75.2	75.1

- On average **11%** better in accuracy than previous HDC method
- **52%** more time efficient than GCN

SRC

String Dataset Results (Accuracy)

Best accuracy

Second best accuracy

	%	Protein	SMS	Promoter	Splice	Average
DNN	LM Finetune [NAACL'19,Sci'23]	96	99	82	92	92.5
HDC	HDC N-gram [TCAS'24]	96	97	52	43	72
Ours	NysHD	98	97	89	72	89

- On average **17%** better in accuracy than previous HDC method [TCAS'24]
- Near-optimal accuracy even compared with DNN baselines [NAACL'19,Sci'23]

SRC

String Dataset Results (Speed)

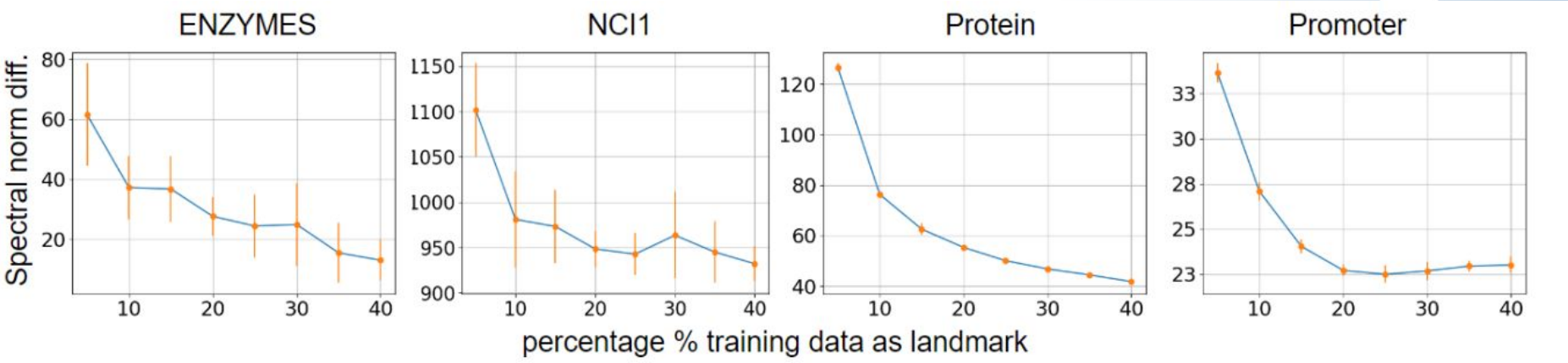
Best accuracy

Second best accuracy

	Seconds	Protein	SMS	Promoter	Splice
DNN	LM Finetune [NAACL'19,Sci'23]	2000	2800	6	142
HDC	HDC N-gram [TCAS'24]	13	43	1	25
Ours	NysHD	19	34	1	21

- Retained HDC efficiency, significantly faster than DNN approaches

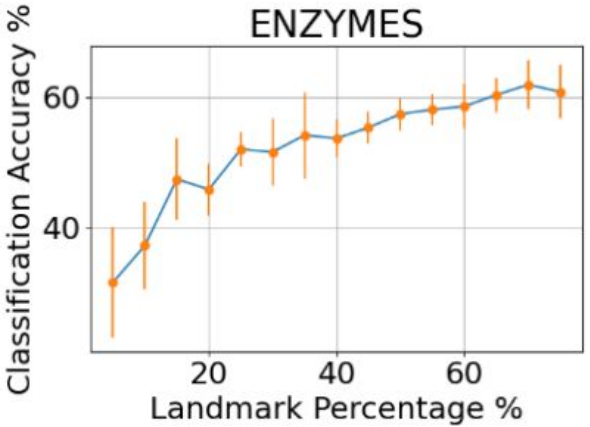
Approximating normalized kernel matrix

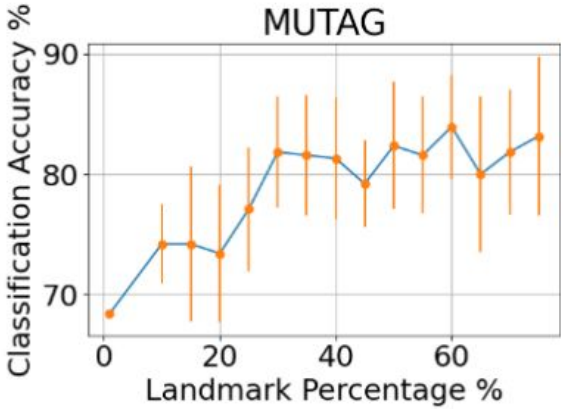


- We measure the difference between HDC approximated kernel matrix and the actual kernel matrix
- Empirically shows successful transfer of kernel function to HDC
  - **Better** normalized kernel approximation as landmark set grow

SRC

Effect of landmark set size





Larger landmark set ➡ Better kernel approximation ➡ Higher accuracy

SRC

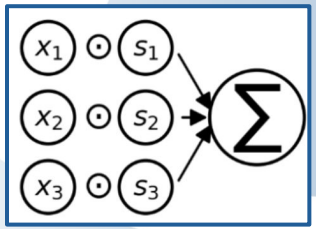
Roadmap of this talk

1: Kernel method view of HDC and Learning more expressive HDC model

Bridging the Gap Between Hyperdimensional Computing and Kernel Methods via the Nyström Method *[AAAI'25]*

2: Improved HDC inference efficiency via approximate kernel inference

Superposed Inference for Hyperdimensional Computing via Multiplexed Random-Feature Kernels *[ICLR'27 WIP]*



SRC

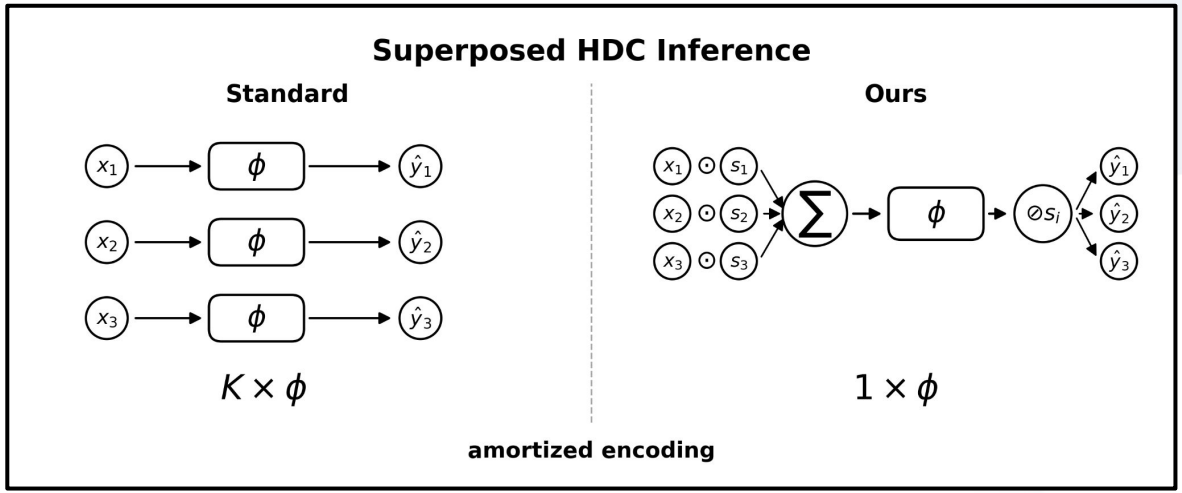
HDC Efficiency Bottleneck

- HDC inference is already efficient, but standard inference still processes each input separately:

$$
\begin{aligned}
x_1 &\rightarrow \phi(x_1) \rightarrow \hat{y}_1 \\
x_2 &\rightarrow \phi(x_2) \rightarrow \hat{y}_2 \\
x_3 &\rightarrow \phi(x_3) \rightarrow \hat{y}_3
\end{aligned}
$$

- Encoding became a major overhead, dominating the inference cost

Superposed Inference for HDC



- Instead of encoding each sample independently, can we pack multiple inputs into one composite representation and perform inference jointly?
  - Inference on N samples in the cost of (almost) 1

Key Intuition — Superposition Preserves Kernel Scores

- Consider an general class of HDC encoder based on random fourier feature.

$$\phi(x) = \frac{1}{\sqrt{D}} \exp(iWx)$$

Changing sampling distribution of ***W*** changes underlying kernel

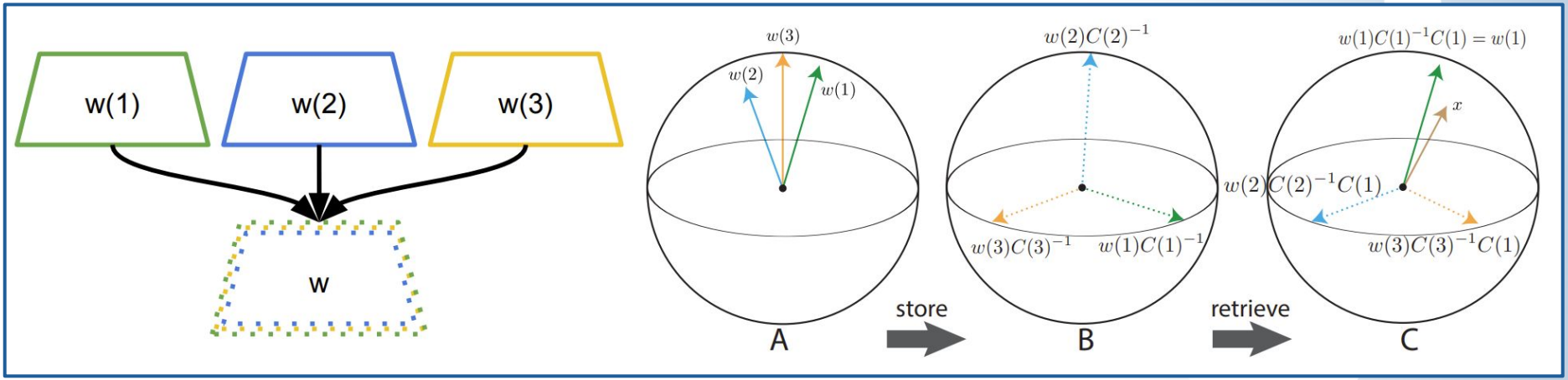
$$\phi(a+b) \propto \phi(a) \odot \phi(b)$$

Superposition in ambient become structured noise

- Why this works?
  - Uses the high-dimensionality of input data themselves: multiple signals can be bundled together while remaining approximately recoverable or separable under some random transformation

Related Works

- Model superpositioning [NeurIPS'19]

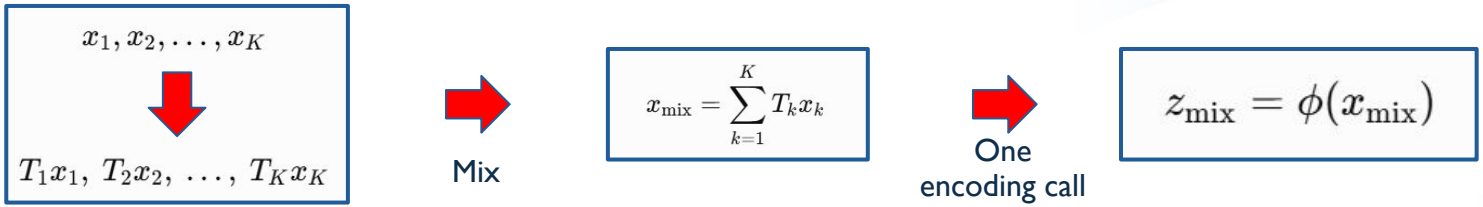


- Store model weight in different mutually orthogonal subspaces
- Can call different model at inference time given context

Slot Keys — Assigning Each Input a Role

- If multiple inputs are simply added together, their roles are ambiguous.
- The method assigns each packed input to a slot.
- For slot k, apply a random signed permutation

$$T_k x = S_k \Pi_k x$$



- Because each slot has a different transform, the model learns slot-specific class readouts.

$$P_{k,c} = \sum_i \alpha_{c,i} \phi(T_k x_i)$$

SRC

Interference under Superposition

Clean score if no superposition is used

$$\tilde{s}_{k,c} \approx \rho s^{\text{plain}}_{k,c} + \epsilon_{k,c}$$

superposed score ≈ scaled plain score + controlled interference

ρ = class-independent attenuation from other packed inputs

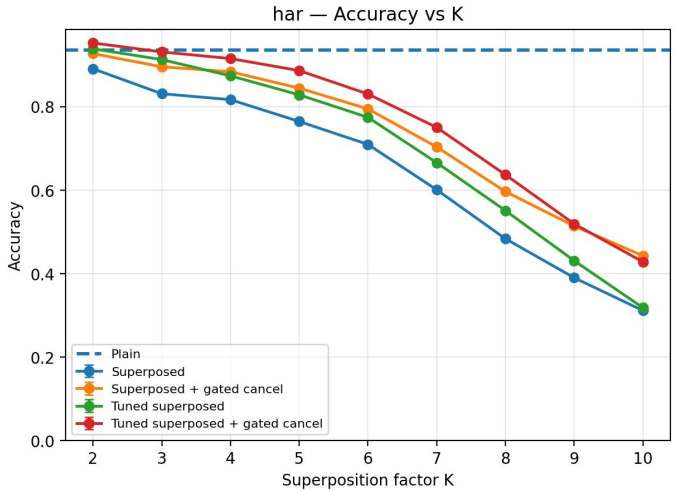
$\epsilon_{k,c}$ = class-dependent residual interference

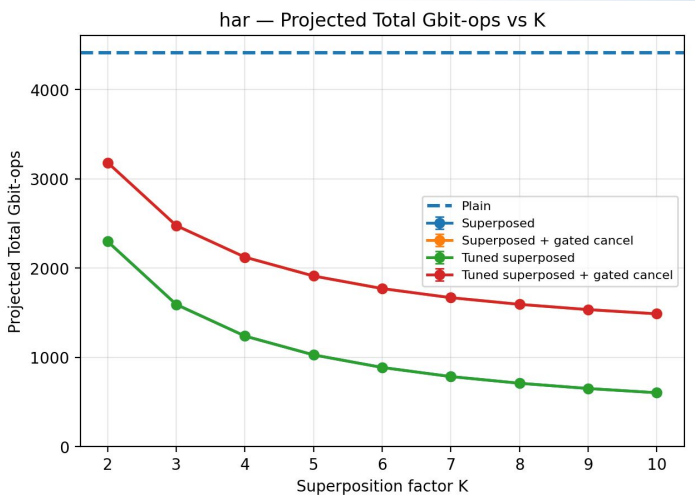
- First scale term class-independent - does not lead to mis-classification
- Second term is class dependent, it becomes smaller with more random/orthogonal slot keys controllable via K and D

Experiment Setup

- Evaluate whether superposed RFF-HDC can reduce inference cost while preserving accuracy.
- Consider Complex RFF as HDC encoder
  - Generalizable to any shift-invariant kernels
- Datasets:
  - HAR, ISOLET, CIFAR-10 embeddings, Airline Sentiment, AG News, IMDb, 20 Newsgroups, SECOM
- Compared methods using different K:
  - Plain RFF-HDC: Baseline model w/o superposition
  - Superposed inference
  - Superposed inference + margin-gated selective cancellation
  - Superposed inference + Superposition-aware fine-tuning
  - Superposed inference + margin-gated selective cancellation + Superposition-aware fine-tuning
- Metrics: Accuracy and Project Gbit-ops

Preliminary Results

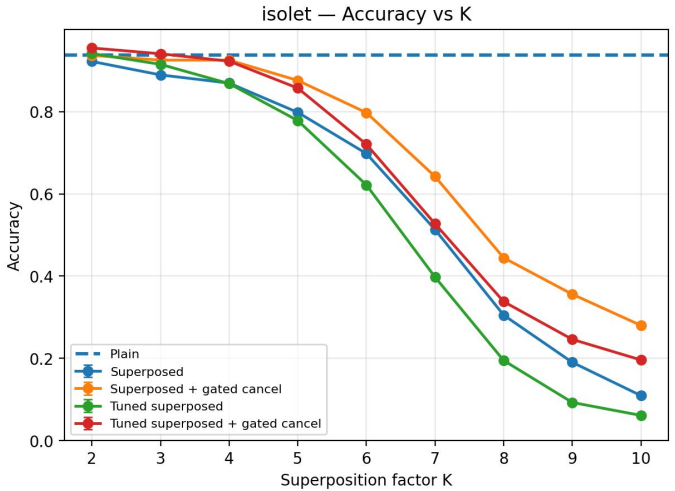


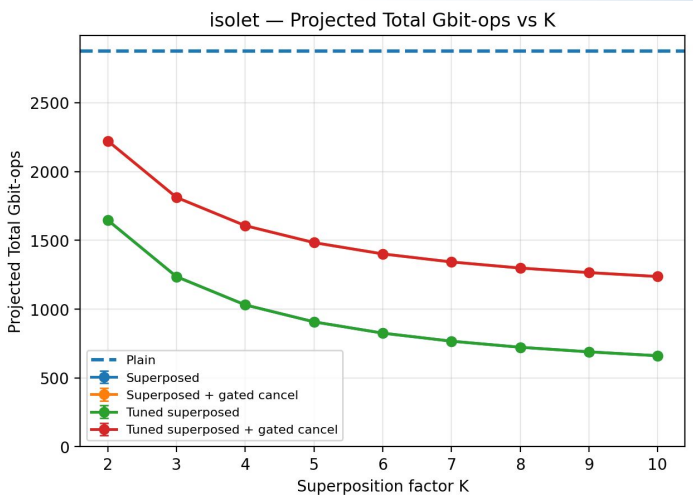


- Superposition gives a clear efficiency–accuracy tradeoff
- Best operating region is small K, especially K=2–4
- At K=3, reduce computation almost 3 times with minimal loss of accuracy

SRC

Preliminary Results





- Almost no accuracy upto K=4, same efficiency gain pattern
- Gated cancellation is important for preserving accuracy, but with additional cost

SRC

Conclusion

- Kernel methods provide a principled lens for understanding HDC.

HDC similarity $\approx$ kernel similarity

Better encoders $\Rightarrow$ more capable HDC models

Superposed inference $\Rightarrow$ more efficient HDC computation

- Kernel method makes HDC more expression
- HDC makes kernel method-inspired learning more efficient