Lab: Brain-inspired Computing for AI

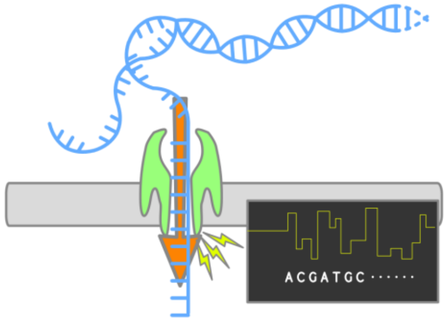
Theory Session: Hyperdimensional Computing

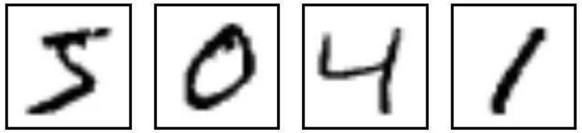
Prof. Dr.-Ing. Hussam Amrouch
Chair of AI Processor Design

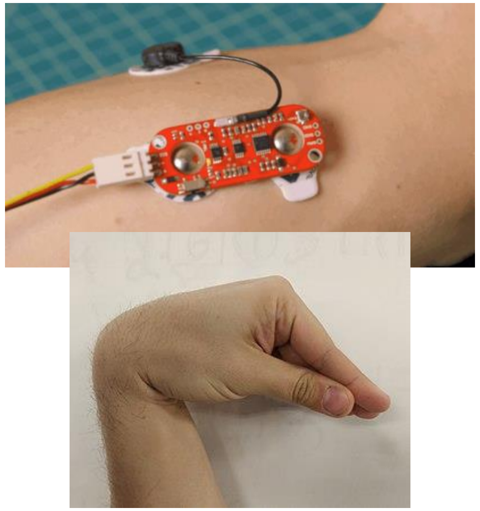
Technical University of Munich

Brain-Inspired Hyperdimensional Computing

- Computing with patterns
- Hyperdimensional (HD) vectors as basic symbols
- No exact match needed: approximation based on similarity
- Pattern matching and search
- Reasoning about the learned data







Hyperdimensional Computing

Randomness and similarity are core concepts.

Two random binary vectors A and B with dimension 5:

```
A=[10101]
B=[00111]
```

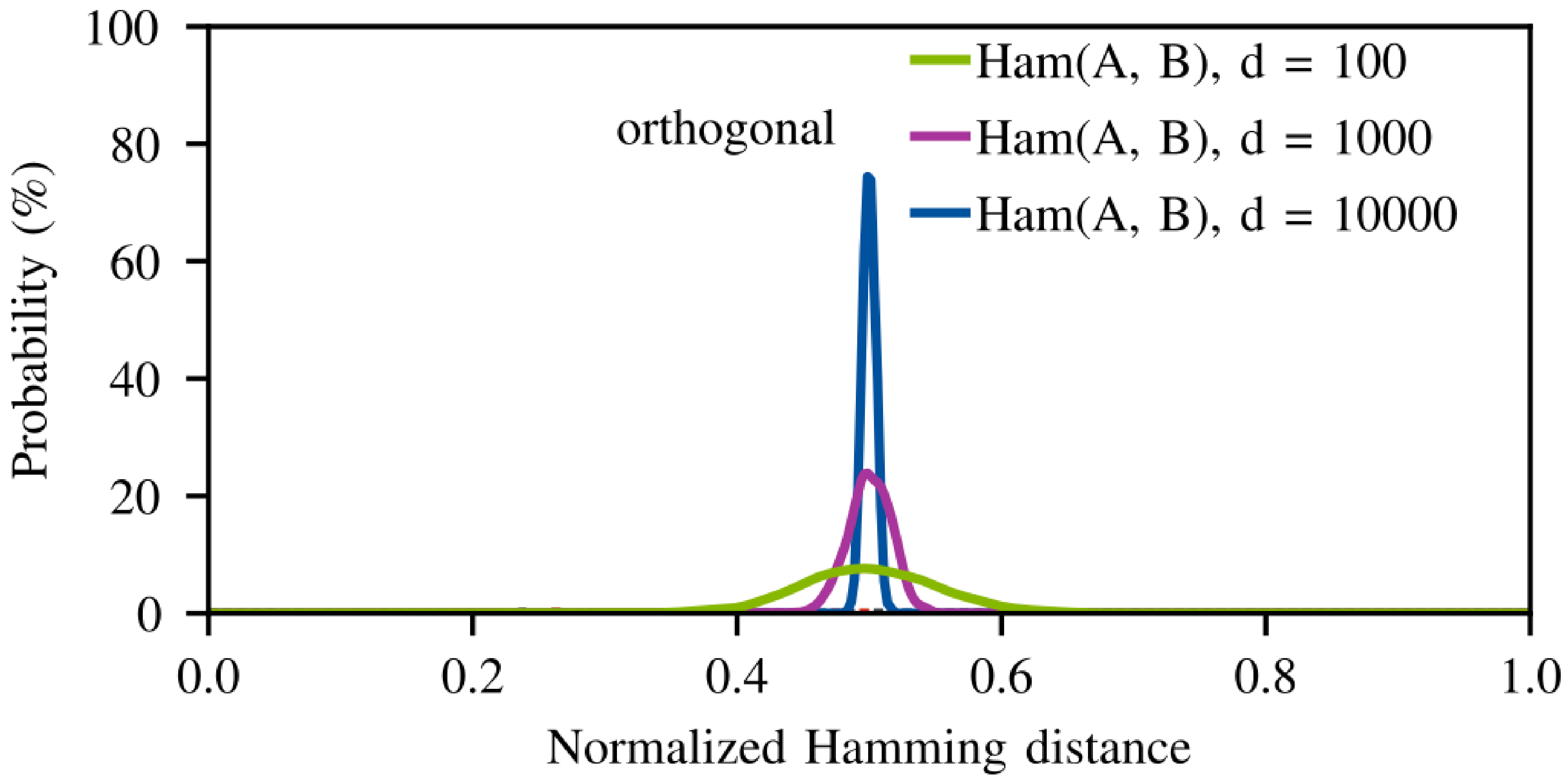
Hamming distance of A and B is 2.

In practice, we will use *much* larger vectors with dimensions in the thousands.

H. Amrouch, E-mail: amrouch@tum.de
Chair of AI Processor Design (AI-Pro)

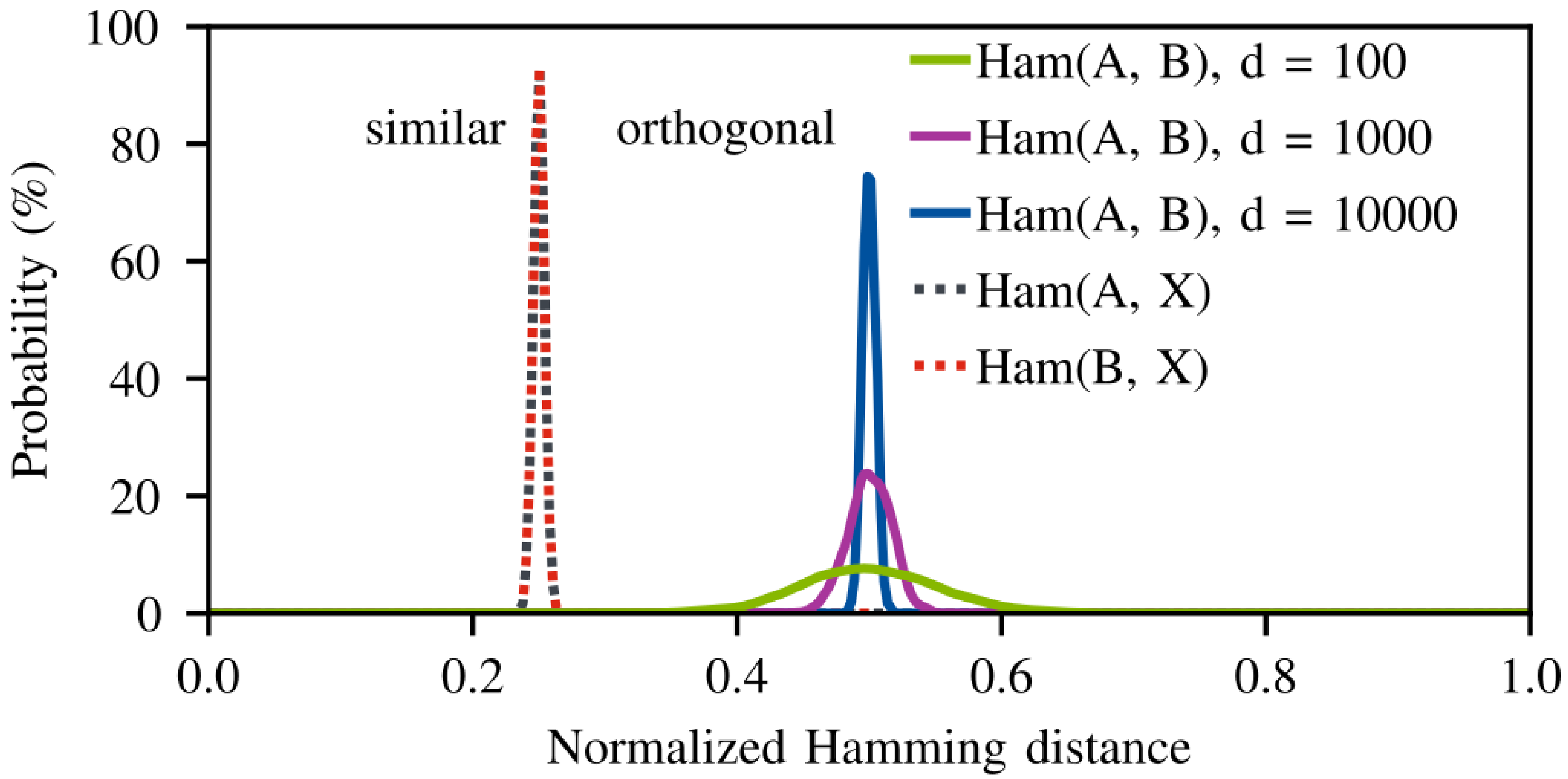
Hyperdimensional Computing

Two random vectors A and B



Hyperdimensional Computing

Two random vectors A and B, and X = A+B



Hyperdimensional Computing

Core operations:

- ❑ Permutation Π
- ❑ Binding $\otimes$
- ❑ Bundeling $\oplus$
- ❑ Similarity

Permutation

Symbol: Π

Random permutation of the vector elements

Generates orthogonal vector based on input

Binary implementation: Shift elements in vector (cyclic shift)

Example:

$$x = [0011]$$
$$\Pi(x) = [1001]$$

Binding

Symbol: $\otimes$

"Multiplies" inputs

Result is orthogonal to both inputs

Invertible

Binary implementation: XOR

Example:

$$a = [0100]$$
$$b = [0101]$$
$$a \otimes b = [0001] = c$$
$$c \otimes b = a \otimes b \otimes b = [0100] = a$$

Bundeling

Symbol: $\oplus$

"Addition" of two vectors

Result is **similar** to the inputs

Binary implementation: E.g., majority of popcount

Example:

$$
\begin{gathered}
[1101] \oplus \\
[0101] \oplus \\
[0001] = \\
[1203] \rightarrow \text{Binarize} \\
[0101]
\end{gathered}
$$

H. Amrouch, E-mail: amrouch@tum.de
Chair of AI Processor Design (AI-Pro)

„What is the Dollar of Mexico?“

- Human Brain recognizes Dollar as currency → Currency of Mexico?
- Build model of countries
  1. Random vectors for properties: "currency" V_C, "population" V_P
  2. Random vectors for values: "Dollar" $V_\$$, "Peso" $V_{₱}$, "350M“ V_{350}, "120M" V_{120}
  3. “Learn” countries: $V_{USA} = V_C \otimes V_\$ \oplus V_P \otimes V_{350}$

     $V_{MEX} = V_C \otimes V_{₱} \oplus V_P \otimes V_{120}$

❑ Currency of USA $= V_{USA} \otimes V_C$

$$= \underbrace{V_C \otimes V_\$ \otimes V_C}_{V_\$} \oplus \underbrace{V_P \otimes V_{350} \otimes V_C}_{\oplus \text{ unknown vector } \triangleq \text{ noise}}$$

$\approx V_\$$

Remember:
Multiplication is invertible

H. Amrouch, E-mail: amrouch@tum.de
Chair of AI Processor Design (AI-Pro)

„What is the Dollar of Mexico?“

$$V_{USA} = V_C \otimes V_{\$} \oplus V_P \otimes V_{350}$$
$$V_{MEX} = V_C \otimes V_{₱} \oplus V_P \otimes V_{120}$$

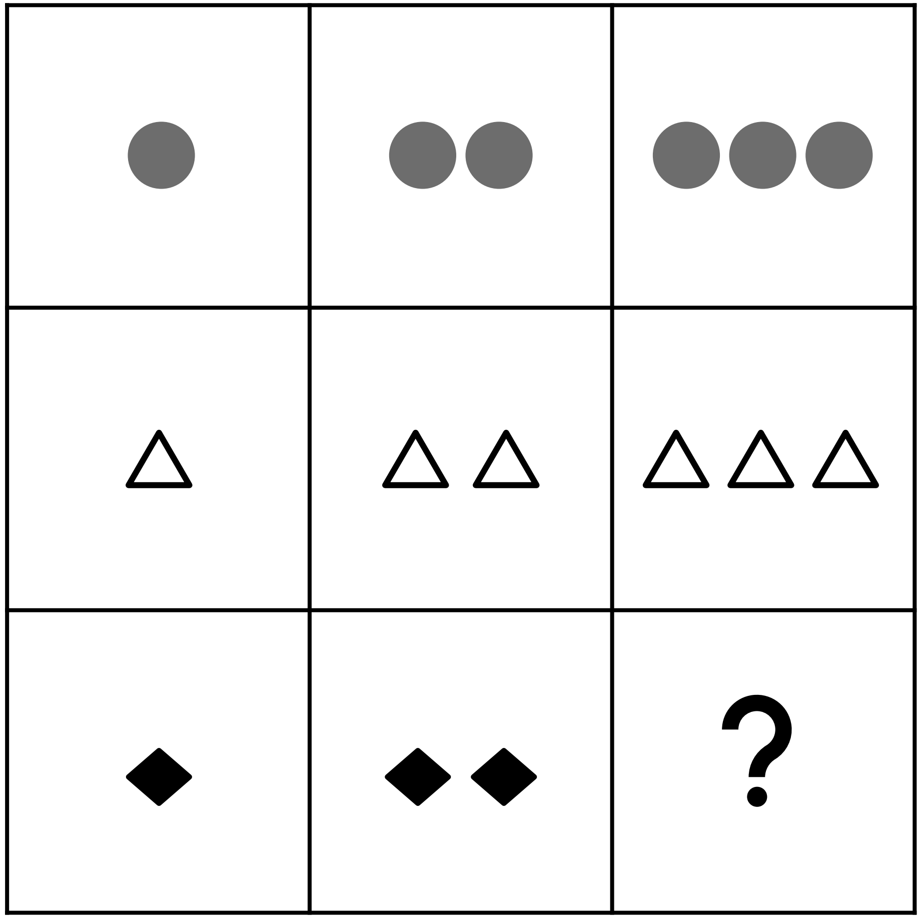
What is the Dollar of Mexico?

$$\text{Dollar of Mexico} = V_{MEX} \otimes \underbrace{(V_{USA} \otimes V_{\$})}_{V_C}$$
$$\approx V_{MEX} \otimes V_C$$
$$\approx V_{₱}$$

H. Amrouch, E-mail: amrouch@tum.de
Chair of AI Processor Design (AI-Pro)

Example: Composition of Elements

- HDC can combine properties and values
- Classical ML methods struggle with composition



General HDC Flow

1. Prepare: Encode real-world data into hyperspace

Generate random item vectors representing basic real-world values → Store in *Item Memory (IM)* → Combine item vectors to encode complex real-world data into a vector

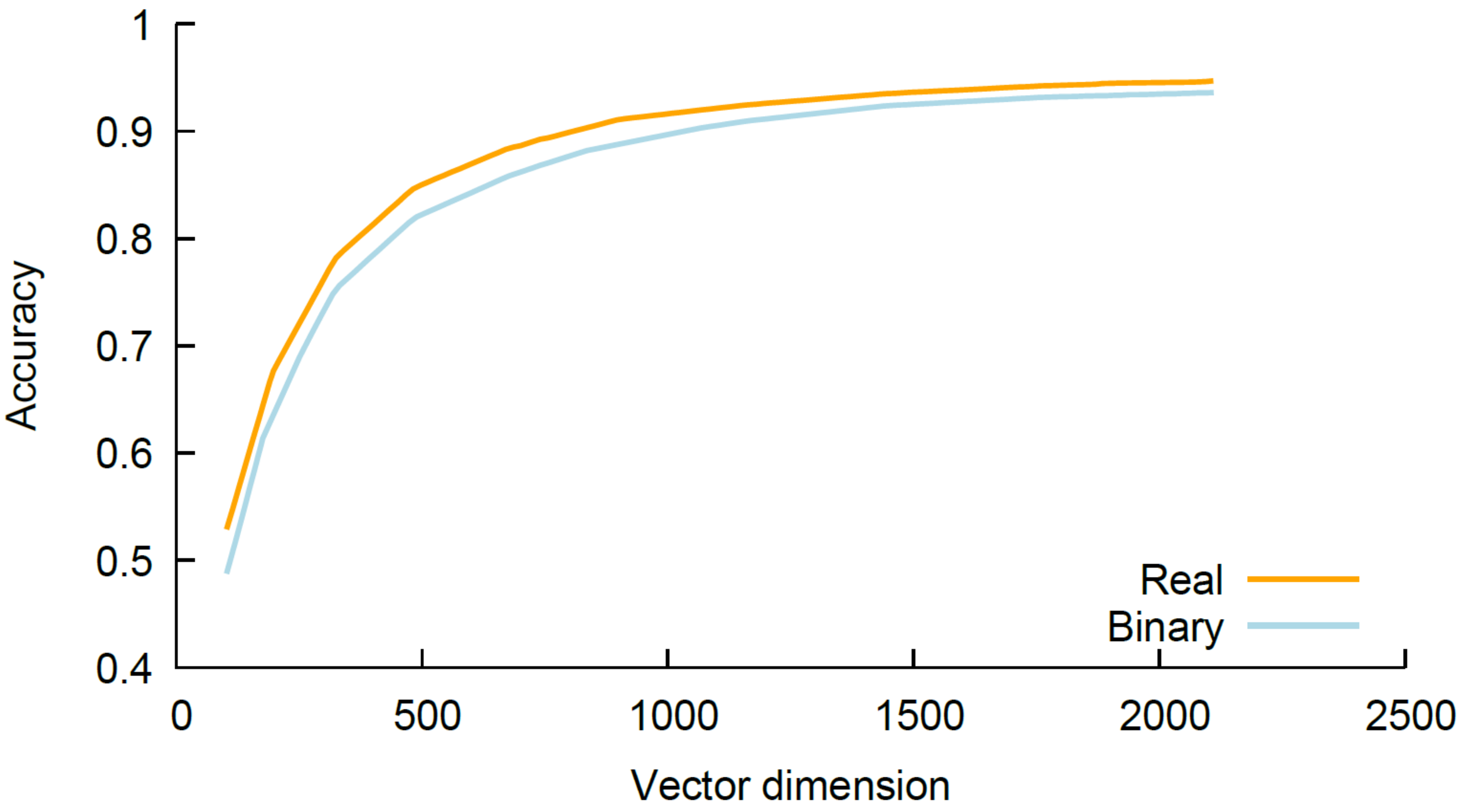
2. Learn: Train the model

Encode training data into one vector per class → Store label ↔ vector in *Associative Memory (AM)*

3. Inference: Recognize unknown data

Encode test data into a vector → Find most similar vector in AM → Inferred class

Using HDC for Language Recognition



Encode Text for HDC

- ❑ Random vectors for each letter of the alphabet

```
a=[10110000010000110101]
b=[10100011011010000001]
      ⋮
␣=[00101011100110111011]
!=[10101111000111100101]
.=[11001010110111001011]
      ⋮
```

- ❑ Store in Item Memory (IM)
- ❑ Constant over training and inference

H. Amrouch, E-mail: amrouch@tum.de
Chair of AI Processor Design (AI-Pro)

N-Grams – Encoding Text Chunks

- Transform substring of *N* characters to HD vector
- Construction of 4-Grams for word "gear"

$$V_{gear} = V_g \otimes \Pi V_e \otimes \Pi\Pi V_a \otimes \Pi\Pi\Pi V_r$$

- Permutation preserves order of characters

$$4\mathrm{-Gram}(\text{"gear"}) \neq 4\mathrm{-Gram}(\text{"rage"})$$

- Encoding longer words like "awesome" → use multiple 4-Grams

$$V_{awesome} = V_{awes} \oplus V_{weso} \oplus V_{esom} \oplus V_{some}$$

Encoding Text with HDC

"To be, or not to be, that is the question:
Whether 'tis nobler in the mind to suffer…"

T o ␣ b e , ␣ o r ␣ n o t ␣ t o ␣ b e …

H. Amrouch, E-mail: amrouch@tum.de
Chair of AI Processor Design (AI-Pro)

Encoding Text with HDC

“To be, or not to be, that is the question:
Whether 'tis nobler in the mind to suffer…”

To␣be,␣or␣not␣to␣be...

[00100100000111110001]

Sliding window stepping 1 character at a time, and using the next 4 characters (4-Gram)

Encoding Text with HDC

“To be, or not to be, that is the question:
Whether 'tis nobler in the mind to suffer…”

To␣be,␣or␣not␣to␣be...

[00100100000111110001]

$\oplus$

[01110100110101111111]

Sliding window stepping 1 character at a time, and using the next 4 characters (4-Gram)

Encoding Text with HDC

"To be, or not to be, that is the question:
Whether 'tis nobler in the mind to suffer…"

T o ␣ b e , ␣ o r ␣ n o t ␣ t o ␣ b e …

[00100100000111110001]

$\oplus$

[01110100110101111111]

$\oplus$

[10100100010111100010]

Sliding window stepping 1 character at a time,
and using the next 4 characters (4-Gram)

Encoding Text with HDC

“To be, or not to be, that is the question:
Whether 'tis nobler in the mind to suffer…”

To␣be,␣or␣not␣to␣be…

[00100100000111110001]

$\oplus$

[01110100110101111111]

$\oplus$

[10100100010111100010]

⋮

Count 1‘s

[3, 6, 10, 9, 13, 4, 19, …]

Majority gate

Class hypervector representing English

[10100110001100111001]

Language Recognition with HDC

1. Prepare: Encode real-world text into hyperspace

Generate random item vectors for letters and symbols → Store in *Item Memory (IM)* → Combine item vectors to encode a text into a vector

2. Learn: Train the language model

Encode training text into one vector per language → Store language ↔ vector mapping in AM

3. Inference: Recognize language of unknown text

Encode unknown text into a vector → Find most similar vector in AM → Inferred language

References

- Kanerva, Pentti. "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors." Cognitive computation 1.2 (2009): 139-159.

- Ge, Lulu, and Keshab K. Parhi. "Classification using hyperdimensional computing: A review." IEEE Circuits and Systems Magazine 20.2 (2020): 30-47.

Image Sources

- https://www.freesion.com/article/67001042007
- https://www.cosademarketing.com/2016/07/herramientas-del-neuromarketing.html
- https://glossary.madaportal.org/glossary/voice-recognition-2/
- https://commons.wikimedia.org/wiki/File:202001_nanopore_sequencing.svg
- https://en.wikipedia.org/wiki/Trousseau_sign_of_latent_tetany