OUT-OF-DISTRIBUTION DETECTION

Evgeny Osipov

LULEÅ UNIVERSITY OF TECHNOLOGY

Problem Formulation

Out-of-distribution (OOD) detection in artificial neural networks addresses the critical challenge of ensuring model reliability when encountering inputs that deviate from the distribution on which the model was trained.

LULEÅ UNIVERSITY OF TECHNOLOGY

Formal definition of the problem

Formally, consider a neural network model $f: \mathcal{X} \rightarrow \mathcal{Y}$ trained on a dataset $\mathcal{D}_{train} = \{(x_i, y_i)\}_{i=1}^{N}$ drawn from an in-distribution P_{in} .The goal of OOD detection is to distinguish between in-distribution samples $x \sim P_{in}$ and out-of-distribution samples $x \sim P_{ood}$,where $P_{ood} \neq P_{in}$.

The problem can be formulated as a binary classification task where the model must assign a confidence score $s(x)$ to each input x, such that:

- For in-distribution samples: $s(x)$ is high
- For out-of-distribution samples: $s(x)$ is low

OOD Challenges

- Neural networks tend to produce overconfident predictions even for OOD inputs
- The decision boundary between in- and out-of-distribution is not well-defined
- OOD detection must be performed without access to OOD data during training
- The method should be computationally efficient and applicable to various network architectures

LULEÅ UNIVERSITY OF TECHNOLOGY

State-of-the-Art Methods

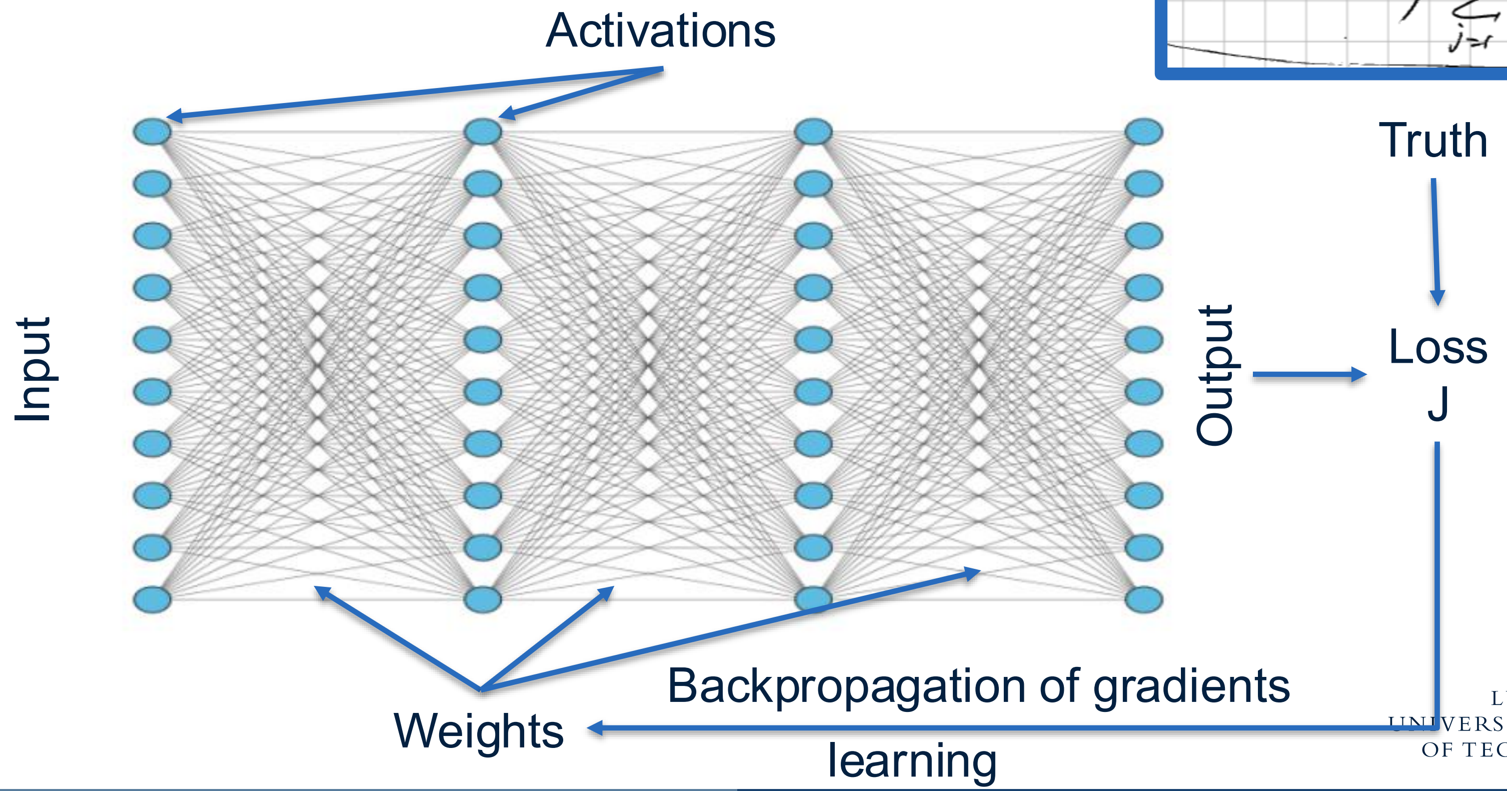
1. Maximum Softmax Probability (MSP)
2. ODIN (Out-of-Distribution Detector for Neural Networks)
3. Mahalanobis Distance-based Detection
4. Energy-based Out-of-Distribution Detection
5. Union of 1-Dimensional Subspaces (U1D)
6. Hyperdimensional Feature Fusion

LULEÅ UNIVERSITY OF TECHNOLOGY

Maximum Softmax Probability (MSP)

Recall the output architecture in Artificial neural networks

SoftMax $S(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{M} e^{x_j}}$



Maximum Softmax Probability (MSP)

Uses the maximum value of the softmax output as a confidence score. OOD samples typically have lower maximum probabilities due to the model’s uncertainty.

Example: For a 3-class classification, suppose an in-distribution sample has softmax outputs [0.05, 0.85, 0.10], yielding MSP = 0.85 (high confidence). An OOD sample might have [0.30, 0.35, 0.35], with MSP = 0.35 (lower confidence), indicating potential OOD.

ODIN (Out-of-Distribution Detector for Neural Networks)

ODIN improves upon MSP by applying **temperature** scaling to the softmax outputs and adding small perturbations to the input. This enhances the separation between in-distribution and OOD samples.

Effect of temperature:

$$S_i(x;T) = \frac{e^{f_i(x)/T}}{\sum_j e^{f_j(x)/T}}$$

When T≫1:

- Logits are scaled down
- Small logit differences matter more
- ID and OOD confidence gaps increase

LULEÅ
UNIVERSITY
OF TECHNOLOGY

ODIN Example

Example: Using temperature T=1000, an in-distribution sample’s scaled softmax might be [0.001, 0.998, 0.001], while an OOD sample after perturbation could have [0.332, 0.334, 0.334]

A challenge

Traditional OOD methods:

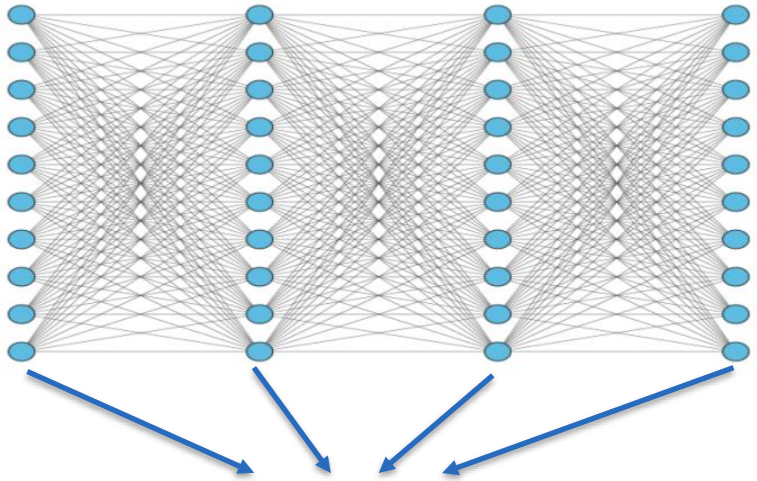
- MSP / ODIN operate on softmax outputs
- Mahalanobis operate on a single feature layer
- Energy-based operate on logits

Problem:

- Different layers encode **different types of information (for images)**
    - Early layers - low-level textures
    - Mid layers - shapes
    - Deep layers - semantics
- Single-layer OOD detection ignores this richness.

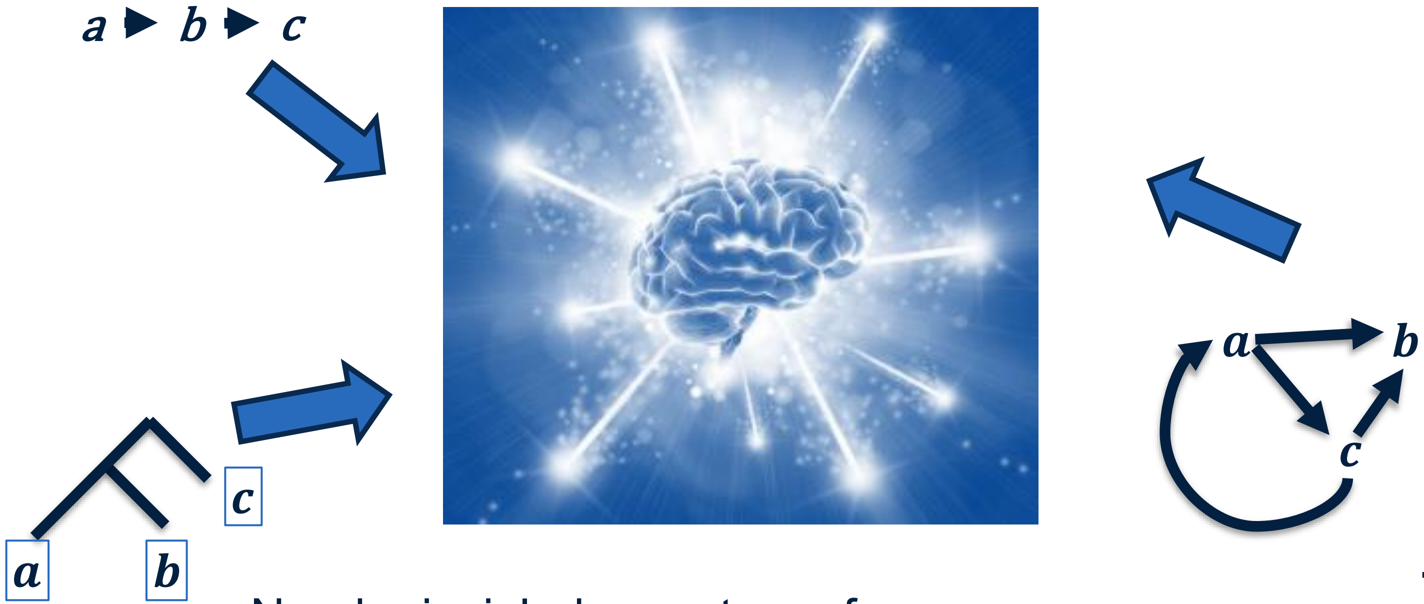
Hyperdimensional Feature Fusion

Idea: fuse features from multiple layers of the neural network into a single representation robust OOD detection.



Single vector representation

… based on methods of hyperdimensional computing



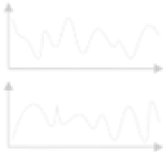
Need principled ways to perform computations on data structures in a brain-like manner

Principle 1: (Hyper)vectors

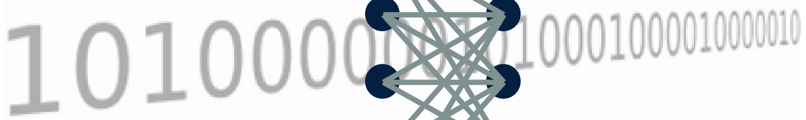
High-dimensional vectors appear everywhere in AI/ML pipelines

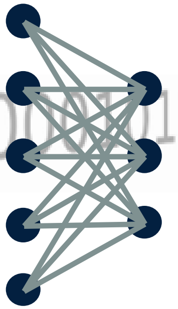
Data Representation → Model Representation → Similarity Measurement

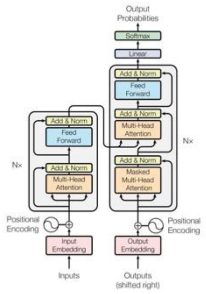
Images **Signals** **Text**

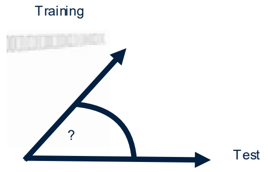


„Robot“









Hyperdimensional Computing (**HDC)** provides a **great toolbox** for handling them.

Hypervectors

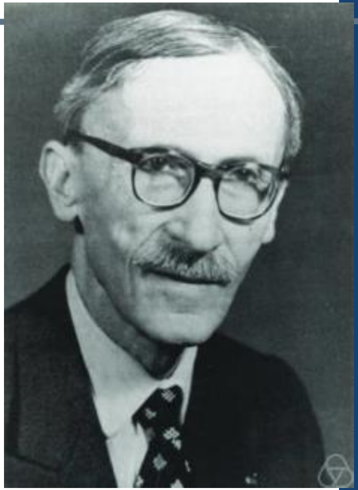
- Let $\mathrm{a, b, c} \ldots \in \{\mathbb{B}^{\mathrm{D}}, \mathbb{R}^{\mathrm{D}}, \mathbb{C}^{\mathrm{D}}\}$,are vectors in D dimensional space
- D is high ($\sim 10^2, \sim 10^3, \sim 10^6, \ldots$)

- Operations:
- $a+b$:B$undling$ ($superposition$) (element-wise sum)
- $a \odot b$:B$inding$ (Hadamard product, XOR, Circular convolution)
- $a \oslash b$:U$nbinding\ operation$
- $\rho(a)$:O$rdering\ operation$ (permutation cyclic shift)

- similarity measure :Cosine similarity, Dot product, Hamming distance

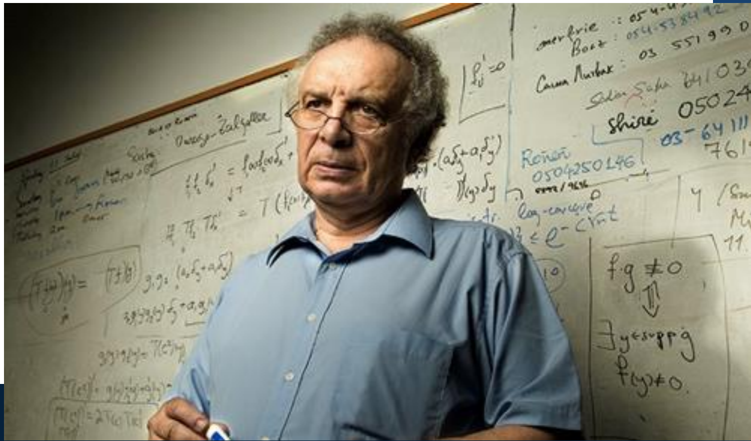
UNIVERS
OF TE

Principle 2: Concentration of Measure Theorems in VSA



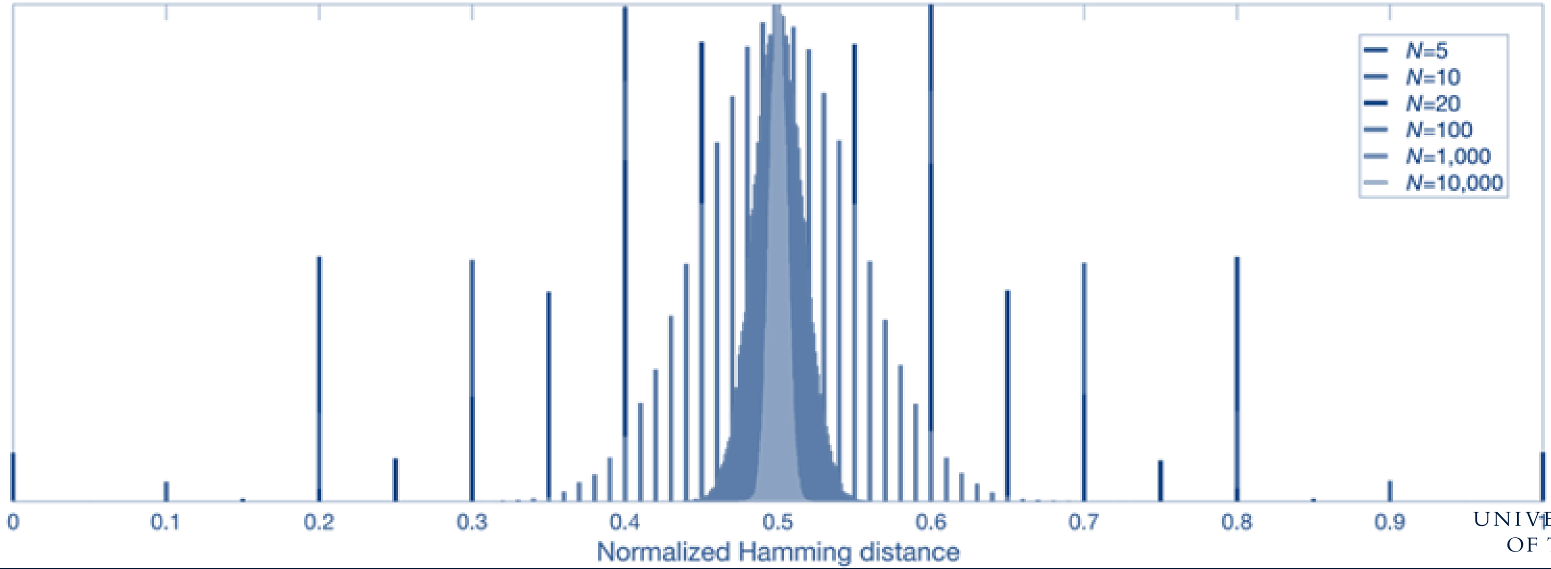
- Concentration of measure: In high dimensions, kernel of random vectors concentrate around their means
  - Paul Lévy
  - Vitali Milman

- Lévy's lemma: For unit vectors x, y on sphere,
  - $P(| < x, y > | \geq t) \leq 2\exp(-dt^2/2)$
- Implication: Random vectors are nearly orthogonal
- "Symbols" can be encoded as random vectors "for free"



Quasi-orthogonality is native in highdimensional random spaces – the case of B^N

- Higher is the dimensionality *N* the more stable is quasi-orthogonality between two random vectors
    - Gets thinner and thinner with increased *N*
    - Distribution of Hamming distance $\Delta_H(\mathbf{A},\mathbf{B})$ between randomly chosen HD vectors

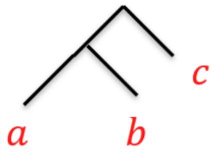


Principle 3: We can encode data structures with algebra on hypervectors

Sequence: $a \rightarrow b \rightarrow c$

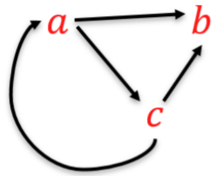
$$s = pos1 \odot a + pos2 \odot b + pos3 \odot c$$

Tree:



$$\boldsymbol{s} = \boldsymbol{r} \odot \boldsymbol{c} + \boldsymbol{l} \odot \boldsymbol{\rho}(\boldsymbol{l} \odot \boldsymbol{a}) + \boldsymbol{l} \odot \boldsymbol{\rho}(\boldsymbol{r} \odot \boldsymbol{b})$$

Graph:



$$\boldsymbol{s} = \boldsymbol{a} \odot \boldsymbol{\rho}(\boldsymbol{b}) + \boldsymbol{a} \odot \boldsymbol{\rho}(\boldsymbol{c}) + \boldsymbol{c} \odot \boldsymbol{\rho}(\boldsymbol{b}) + \boldsymbol{c} \odot \boldsymbol{\rho}(\boldsymbol{a})$$

Similarity between hypervectors captures similarity between data structures

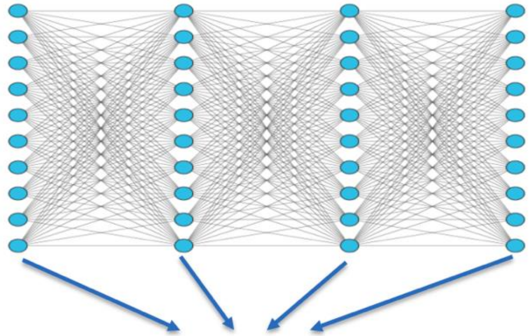
Layer Encoding via Binding

Each layer gets a unique random hypervector key: k_l

Then bind:

$$z_l = k_l \odot h_l$$

($\odot$ = element-wise multiplication)



Single vector representation

- This preserves layer identity.
- Dimensionality of layer representations h_l should be aligned – random projection

Bundling (Superposition)

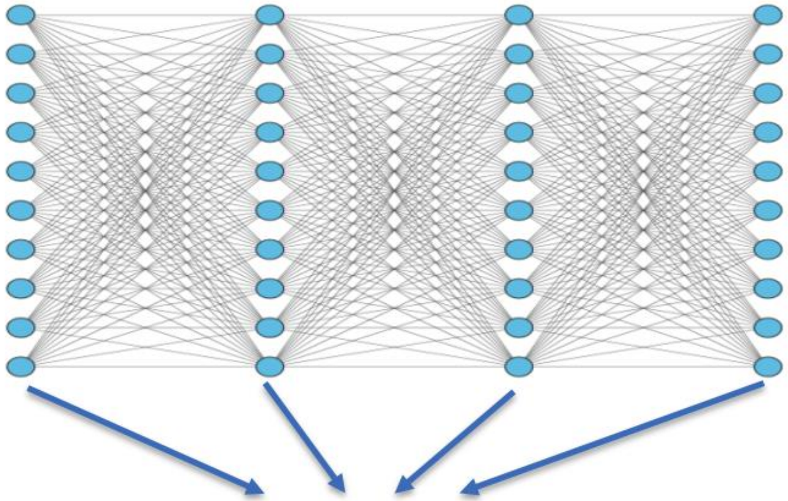
Final input descriptor:

$$y = \sum_{l=1}^{L} z_l$$

Optionally apply sign function:

y=sign(y)

- In this way y is a single vector representing a list of layers L



Single vector representation

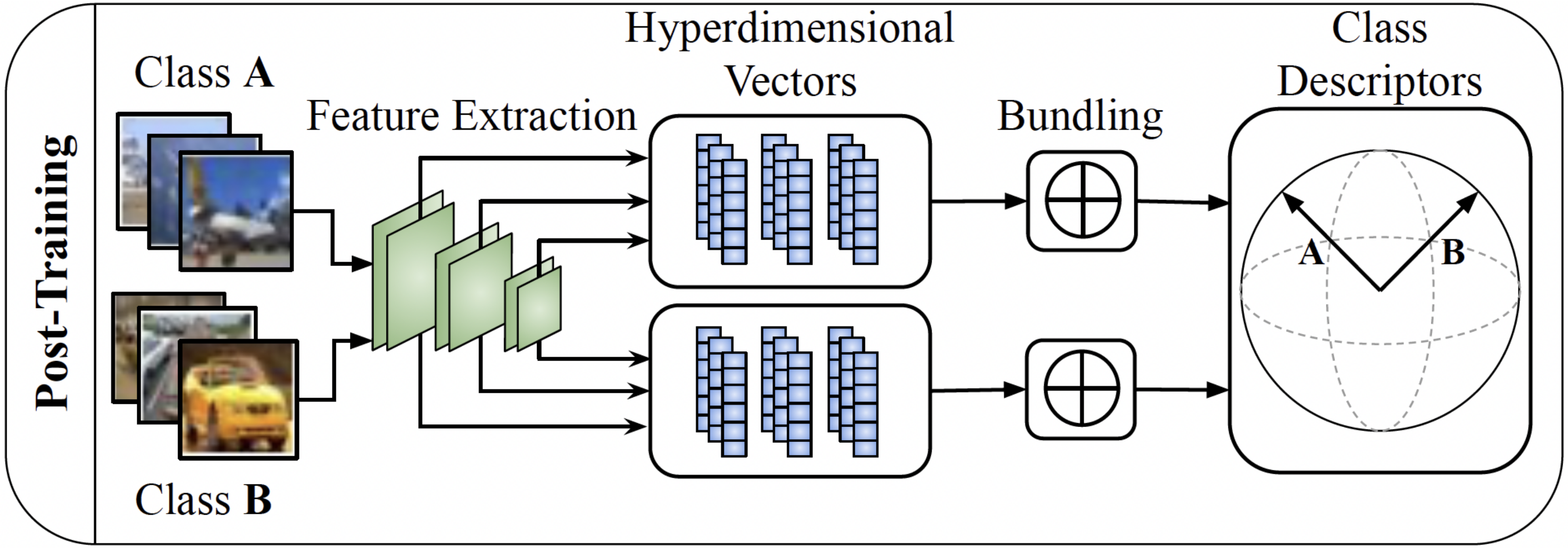
OOD scoring

For test sample:

1. Compute fused hypervector y
2. Compute similarity to prototypes: $s_c = \cos(y, \mu_c)$
3. Define OOD score: $\mathrm{Score}(\mathrm{x}) = \max_c s_c$

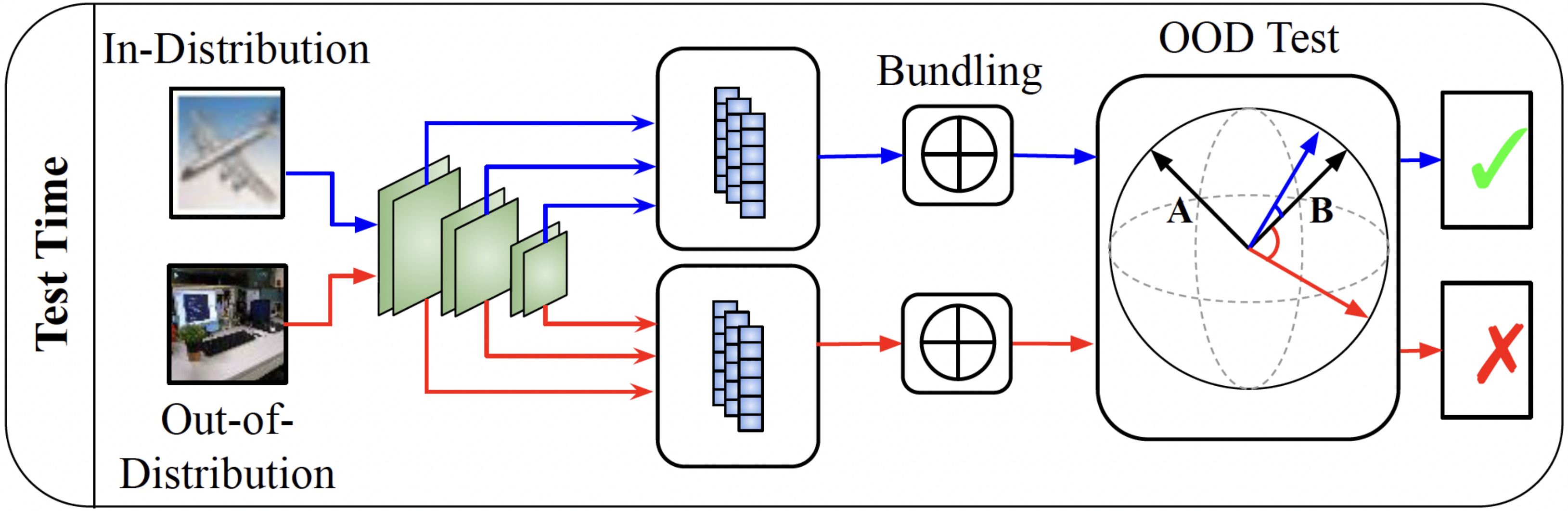
- Low similarity means OOD.

Post-training phase



LULEÅ UNIVERSITY OF TECHNOLOGY

Test time



LULEÅ UNIVERSITY OF TECHNOLOGY

References

- Wilson et al. (2022). Hyperdimensional Feature Fusion for Out-of-Distribution Detection. arXiv:2112.05341.