EdgeNet: Hyperdimensional Computing for Efficient Distributed Classification with Randomized Neural Networks

Arindham Samarth Letitia Grisha Ved

3rd January 2026

1 Introduction and Problem Setting

This paper focuses on **distributed classification**, where:

- A dataset T is spread across multiple agents $\{1, 2, \ldots, N\}$.
- Data cannot be centrally shared (purely distributed scenario).
- Each agent independently trains a **randomized neural network**.
- Agents exchange only **compressed classifier information**.

The goal is to improve classification accuracy at each agent by combining classifier information from neighbors, without sharing raw data.

2 Hyperdimensional Computing (HDC) Framework

2.1 Hypervectors and High-Dimensional Space

A **hypervector** is defined as $\mathbf{v} \in \mathbb{R}^D$ where D is large (e.g., thousands of dimensions). Components of $\mathbf{v}$ are typically bipolar, $\{-1, +1\}$, or normalized real values. Random hypervectors are approximately orthogonal with high probability.

The following key operations are used in HDC:

2.2 Binding

Binding associates two hypervectors into one:

$$\mathbf{z} = \mathbf{x} \star \mathbf{y}.$$

Two common implementations are:

1. **Component-wise multiplication:**

$$z_i = x_i \cdot y_i, \quad i = 1, 2, \ldots, D$$

2. **Circular convolution** (denoted ˜): For hypervectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^D$:

$$z_j = \sum_{k=0}^{D-1} y_k \, x_{(j-k) \bmod D}, \quad j = 0, 1, \ldots, D-1.$$

   This operation mimics a compressed outer product and preserves associativity.

2.3 Superposition (Summation)

Superposition (or bundling) combines multiple hypervectors:

$$\mathbf{s} = \sum_i \mathbf{v}_i.$$

This operation increases similarity to all included hypervectors. To prevent unbounded growth of components, a clipping function $f_\kappa(x)$ can be applied:

$$f_\kappa(x) = \begin{cases} -\kappa, & x \le -\kappa \\ x, & -\kappa < x < \kappa \\ +\kappa, & x \ge \kappa \end{cases}.$$

3 Randomized Neural Networks (RVFL)

The paper uses **Random Vector Functional Link (RVFL)** networks to map input features into hyperdimensional space.

3.1 Feature Encoding into Hypervectors

Given an input vector

$$\mathbf{x} = [x_1, x_2, \ldots, x_K]^\top$$

with K features:

- Quantize each feature.
- Encode each feature as a bipolar hypervector using a **thermometer code**.
- Collect all resulting hypervectors into a matrix

$$F \in \mathbb{R}^{D \times K}.$$

Random fixed weights are assigned as

$$W_{\text{in}} \in \mathbb{R}^{D \times K},$$

where each column $W_{\text{in},j}$ is a random hypervector corresponding to feature j.

3.2 Hidden Layer Activation

The hidden layer activations are computed as:

$$\mathbf{h} = f_\kappa\left(\sum_{j=1}^{K} W_{\text{in},j} \star F_j\right),$$

where:

- F_j is the j-th feature hypervector.
- $f_\kappa(\cdot)$ is the activation/clipping function.
- $\star$ denotes the binding operation (component-wise multiplication or circular convolution).

4 Classifier Definitions

4.1 Regularized Least Squares Classifier

Collect M training samples into:

- Hidden activation matrix: $H \in \mathbb{R}^{M \times D}$
- Class one-hot labels: $Y \in \mathbb{R}^{M \times L}$

The optimal classifier weight matrix $W_{\text{out}} \in \mathbb{R}^{D \times L}$ is computed as:

$$W_{\text{out}} = \left(H^\top H + \lambda I_D\right)^{-1} H^\top Y,$$

where $\lambda > 0$ is a regularization parameter and I_D is the identity matrix of size D.

4.2 Centroid Classifier (HDC)

For class i, define the centroid as:

$$W_{\text{out},i} = \frac{1}{|C_i|} \sum_{t:h(t)\in C_i} h(t),$$

where C_i is the set of samples belonging to class i and $h(t)$ is the hidden layer activation for sample t.

5 Distributed Classification

Consider a network of N agents with adjacency matrix Ω:

- Each agent p has its local dataset T_p.
- Each agent trains a local classifier $W_{\mathrm{out}}^{(p)}$.
- Each agent aggregates neighbor classifiers:

  $$W_{\mathrm{dist}}(p) = \sum_{s \in \mathcal{N}(p)} W_{\mathrm{out}}^{(s)},$$

  where $\mathcal{N}(p)$ are the neighbors of agent p.

This aggregation is **one-shot** — no iterative consensus is computed.

6 Compression via Hyperdimensional Encoding

To reduce communication bandwidth, the classifier W_{out} can be compressed into a single hypervector using **key-value binding**.

6.1 Key Generation

For each class i, generate a random key hypervector:

$$K_i \in \mathbb{R}^D.$$

Bind the class weight with the key using circular convolution:

$$Z_i = K_i \tilde{} W_{\mathrm{out},i},$$

where $\tilde{}$ denotes circular convolution.

6.2 Superposition of All Classes

The compressed classifier hypervector is:

$$\mathbf{w} = \sum_{i=1}^{L} Z_i = \sum_{i=1}^{L} K_i \tilde{} W_{\mathrm{out},i}.$$

This hypervector $\mathbf{w}$ is what agents exchange.

6.3 Approximate Decompression

Given $\mathbf{w}$ and class key K_i, the approximate class weight can be recovered as:

$$\hat{W}_{\mathrm{out},i} \approx \mathbf{w} \tilde{} K_i^{-1}.$$

Since superposition is lossy, some **crosstalk noise** is introduced, but combining multiple reconstructed classifiers from neighbors helps average out the noise.