Topology-Aware Distributed Ensemble Learning Using the vCube Hierarchical Hypercube Overlay

Charles Giovane Salles, André Luiz Brun, and Luiz Antonio Rodrigues(✉)

Western Parana State University, Cascavel, Brazil
{charles.salles,andre.brun,luiz.rodrigues}@unioeste.br

Abstract. The increasing volume of distributed data poses challenges for scalable machine learning, including communication overhead, privacy concerns, and system robustness. Distributed learning approaches address these issues by allowing models to be trained locally while exchanging knowledge among nodes. However, the efficiency and resilience of such systems strongly depend on the communication topology used for model dissemination. This paper proposes a topology-aware distributed ensemble learning architecture based on the vCube hierarchical hypercube-like overlay. In this approach, nodes independently train local classifiers using private datasets and disseminate the resulting models through the vCube communication structure. This mechanism enables efficient and fault-tolerant model sharing without transferring raw data, allowing each node to construct a distributed ensemble classifier. The architecture is evaluated using a dataset with more than 1.4 million samples distributed across multiple nodes. Experiments consider heterogeneous data distributions and node failures. Results show that the proposed approach improves classification accuracy compared to a decentralized tree-based dissemination strategy while maintaining robustness under node failures and significantly reducing communication costs.

Keywords: Distributed Machine Learning · Fault Tolerance · vCube

1 Introduction

The rapid growth of data generated by digital systems has increased the demand for scalable machine learning solutions [20]. In many domains such as mobile sensing, healthcare, smart cities, and monitoring systems, data is generated and stored across geographically distributed locations. Although centralizing data can simplify model training, it introduces challenges including high communication costs, privacy concerns, and potential single points of failure [18].

Distributed learning has emerged as an alternative paradigm that enables models to be trained using decentralized data sources [19]. In this setting, each node performs local training using its own dataset and collaborates with other

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027
O. Gervasi et al. (Eds.): ICCSA 2026, LNCS 16767, pp. 321–339, 2027.
https://doi.org/10.1007/978-3-032-30491-9_21

nodes to share knowledge without transferring raw data. This strategy reduces communication overhead while preserving data privacy and enabling knowledge integration across the network.

Several architectures have been proposed to support distributed learning [10,14]. Centralized approaches rely on parameter servers to aggregate models, which may introduce communication bottlenecks and single points of failure. Hierarchical structures such as tree-based architectures improve scalability but remain vulnerable to failures in upper-level nodes. Peer-to-peer approaches provide greater robustness by distributing responsibilities among nodes, but require efficient mechanisms to coordinate model dissemination and integration.

Ensemble learning is another strategy commonly used in distributed environments, where multiple classifiers trained on different datasets are combined to produce a final prediction [2,17]. Distributed ensemble methods are particularly attractive when nodes train models using local data and exchange them for collaborative inference. However, the efficiency and robustness of these systems strongly depend on the communication topology used for disseminating models.

In this work, we explore the use of the vCube topology [3], a hierarchical hypercube-based communication structure designed for scalable and fault-tolerant distributed systems. The vCube topology organizes nodes into clusters that grow logarithmically, enabling efficient information dissemination while maintaining resilience to node failures. Although vCube has been applied in several distributed system scenarios, its potential in distributed machine learning environments remains largely unexplored.

This paper proposes a distributed ensemble learning architecture that leverages the vCube topology to disseminate locally trained classifiers across nodes. Each node trains a model using its local dataset and shares the resulting classifier following the vCube communication protocol. After dissemination, nodes obtain the classifiers generated across the network and construct a distributed ensemble model for prediction. This strategy enables collaborative knowledge integration while maintaining decentralized training and robustness to node failures.

The proposed architecture is evaluated using a human activity recognition dataset containing more than 1.4 million samples. Experiments consider multiple data distribution scenarios, including balanced distributions, heterogeneous dataset sizes, and non-stratified class distributions. We also evaluate system behavior under node failures to analyze the fault-tolerance properties of the vCube topology. The results are compared with a decentralized tree-based architecture proposed in [19].

This paper makes the following contributions:

1. A topology-aware distributed ensemble learning architecture based on the vCube hierarchical hypercube overlay.
2. A fault-tolerant model dissemination strategy that tolerates node failures without interrupting ensemble construction.
3. An experimental evaluation under heterogeneous data distributions and node failures.

4. An empirical comparison with a decentralized tree topology, demonstrating improved robustness and accuracy.

The remainder of this paper is organized as follows. Section 2 presents background concepts and related work. Section 3 describes the proposed methodology. Section 4 details the configuration of the experiments performed while Sect. 5 presents and discusses the experimental results. Finally, Sect. 6 concludes the paper and outlines directions for future work.

2 Background and Related Work

This section presents the main concepts related to distributed machine learning, ensemble learning, and communication topologies used in decentralized systems, including vCube. In addition, we review previous work addressing distributed model training and knowledge sharing in large-scale environments.

2.1 Distributed Machine Learning and Ensemble Learning

The rapid growth of data volumes has motivated the development of distributed machine learning techniques capable of processing data stored across multiple locations. In many real-world scenarios, data is generated and maintained in geographically distributed environments, making centralized training impractical due to communication costs, privacy concerns, and scalability limitations.

Distributed machine learning enables multiple nodes to collaboratively train models without transferring raw datasets. In this setting, each node processes its local data and contributes to the learning process through model updates or knowledge exchange. Several architectures have been proposed to support distributed learning, including centralized parameter servers, hierarchical aggregation structures, and fully decentralized peer-to-peer systems [19]. While centralized approaches simplify coordination, they may introduce communication bottlenecks and single points of failure. Decentralized architectures mitigate these issues by allowing nodes to communicate directly, often using structured or unstructured communication graphs.

Ensemble learning provides a complementary strategy for distributed environments by combining multiple classifiers to improve prediction accuracy and robustness [2]. In this approach, nodes independently train classifiers using their local datasets, and the resulting models are aggregated to produce a global prediction system. This strategy is particularly attractive when datasets cannot be centralized due to privacy or communication constraints. Several ensemble combination strategies have been proposed, including majority voting, sum rules, and product rules [17].

2.2 The vCube Topology

The vCube [3] is a hierarchical virtual topology derived from distributed diagnose algorithms [5]. It relies on an active failure detection mechanism in which

each process periodically tests its peers. If a tested process responds within the expected time window, it is considered correct; otherwise, it is marked as suspect.

The topology considers a distributed system composed of n processes identified from 0 to $n-1$. Each pair of nodes can communicate directly, forming a fully connected network. Nodes may fail and recover, and in our scenario each node represents a site that stores data used for training classification models.

vCube organizes nodes into clusters $s = 1, ..., \log_2 n$, where cluster s contains 2^{s-1} processes. The function $c_{i,s}$ (Eq. 1) defines the ordered list of processes in each cluster, where $\oplus$ denotes the bitwise exclusive OR (XOR) operator.

$$c_{i,s} = \{i \oplus 2^{s-1}, c_{i\oplus 2^{s-1},1}, ..., c_{i\oplus 2^{s-1},s-1}\} \tag{1}$$

Figure 1 illustrates the results of $c_{i,s}$ for a system with eight processes. For each node i, an edge (j, i) exists such that j is the first correct node in $c_{i,s}$ for $s = 1... \log_2 n$. When a node failure or recovery is detected, the testing edges are recalculated. In the example, node n_0 initially tests n_4 in cluster 3, but after n_4 fails it starts testing n_5, the next correct node in $c_{0,3}$.

s	$c_{0,s}$	$c_{1,s}$	$c_{2,s}$	$c_{3,s}$	$c_{4,s}$	$c_{5,s}$	$c_{6,s}$	$c_{7,s}$
1	1	0	3	2	5	4	7	6
2	2,3	3,2	0,1	1,0	6,7	7,6	4,5	5,4
3	4,5,6,7	5,4,7,6	6,7,4,5	7,6,5,4	0,1,2,3	1,0,3,2	2,3,0,1	3,2,1,0

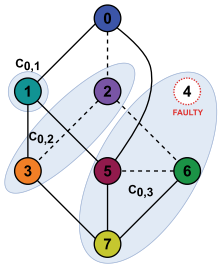

Fig. 1. Clusters of a vCube with $2^3 = 8$ nodes; n_4 is fault.

One key advantage of vCube is its ability to dynamically adapt to node failures. When a node fails, communication can be redirected to alternative nodes within the same cluster, preserving the dissemination process. vCube performs testing rounds with logarithmic latency $\log_2 n$, while the maximum number of tests is bounded by $n \log_2 n$ [5].

Due to these properties, vCube has been applied in several distributed system contexts, including failure detection, monitoring, and scalable information dissemination [1,7,15,16].

2.3 Related Work

Several studies have investigated distributed learning architectures designed to train machine learning models across multiple nodes without centralizing datasets. These approaches differ mainly in how models are trained, exchanged, and aggregated.

Federated learning is one of the most widely adopted paradigms for decentralized training. Clients train models locally and periodically share updates with a central aggregator that produces a global model. The Federated Averaging (FedAvg) algorithm is a well-known example and has been widely applied in mobile and edge computing environments [13]. Subsequent studies addressed challenges such as heterogeneous data distributions, communication efficiency, and system robustness [8]. However, many implementations rely on centralized aggregation, which may introduce bottlenecks and single points of failure.

To address these limitations, decentralized learning approaches allow nodes to communicate directly without a central coordinator. These methods typically rely on peer-to-peer communication graphs to propagate model updates. For instance, decentralized stochastic gradient descent enables nodes to exchange parameters with neighbors and gradually converge to a shared solution [10]. Gossip-based learning follows a similar idea, spreading updates through randomized communication patterns.

Another research direction explores distributed ensemble learning, where nodes independently train classifiers and combine them during inference. Ensemble methods improve predictive performance by aggregating multiple models [2]. In distributed environments, nodes exchange trained classifiers instead of model parameters or training data, enabling collaborative inference across datasets [17].

Communication topology plays a key role in distributed learning systems. Centralized architectures rely on parameter servers, while hierarchical approaches aggregate models at multiple levels [19]. Although hierarchical structures reduce communication overhead, failures near the root may degrade system performance. Fully decentralized architectures distribute communication responsibilities across nodes, improving resilience.

The vCube topology was originally proposed as a scalable failure detection protocol for distributed systems [3]. By organizing nodes into logarithmic clusters, it enables efficient information dissemination while maintaining resilience to failures. Despite its advantages, its application to distributed machine learning remains largely unexplored.

In contrast to traditional federated learning methods that rely on iterative parameter aggregation, this work investigates a distributed ensemble learning architecture in which nodes independently train local models and disseminate them using the vCube communication structure. By leveraging the fault-tolerant properties of vCube, the proposed approach enables scalable model sharing and robust classifier integration in distributed environments, reducing synchronization overhead.

3 Proposed Architecture

This section describes the distributed learning architecture proposed in this work. The architecture consists of three components: (i) local model training at each node, (ii) model dissemination using the vCube topology, and (iii) distributed ensemble construction. The objective is to enable collaborative knowledge sharing among distributed nodes without transferring raw data.

3.1 System Model

We consider a distributed system composed of N nodes interconnected through a communication network. Each node n_i stores a local dataset D_i, which remains private and is not shared with other nodes. Nodes can independently train models, exchange trained models with peers, and may fail during execution. Communication follows the hierarchical hypercube structure defined by the vCube protocol.

Each node trains a local classification model using its dataset:

$$M_i = Train(D_i) \tag{2}$$

After training, models are disseminated across the network using the vCube communication structure. Once dissemination is completed, each node obtains the models trained by all nodes and constructs a distributed ensemble classifier.

Model dissemination occurs through the vCube topology [3], described in Sect. 2.2. The topology organizes nodes into clusters of increasing size, enabling efficient information propagation.

For a system with N nodes, the dissemination process requires $\log_2 N$ communication rounds. During round s, node i exchanges its current models with the first active node in cluster $C_{i,s}$.

Algorithm 1 summarizes the dissemination procedure.

Algorithm 1. Distributed Model Dissemination using vCube

1: $ModelSet \leftarrow \{M_i\}$
2: **for** $s = 1$ to $\log_2 N$ **do**
3: $\quad j \leftarrow$ first active node in cluster $C_{i,s}$
4: $\quad$ send $ModelSet$ to node j
5: $\quad$ receive $ModelSet_j$ from node j
6: $\quad ModelSet \leftarrow ModelSet \cup ModelSet_j$
7: **end for**
8: **return** $ModelSet$

Figure 2 illustrates the communication procedure in a system with $N = 8$ processes. Initially (step A), each node trains its classification model locally, relying on its own data and not considering models from other nodes.

In the next phase (step B), the model exchange process begins. Following the vCube topology, each node transmits its trained model to a predefined counterpart, as determined by the function $c_{i,s}$. In this example, communication follows a structured pairing scheme in which nodes exchange models within predefined groups: node pairs (0,1), (2,3), (4,5), and (6,7) share their respective models trained in step A.

In step B, each node shares its respective models with its neighbor in the first cluster defined by vCube. In the example, the nodes 0 and 1 exchange their models, as do the pairs $(2, 3)$, $(4, 5)$ and $(6, 7)$.

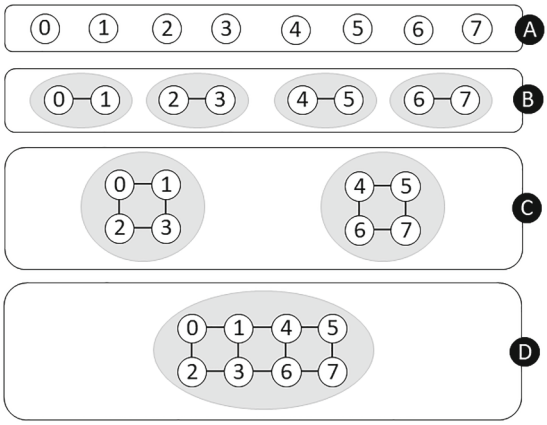

Fig. 2. Communication between 8 nodes based on the vCube topology.

In the third step (C), the pairs formed in the previous step are combined so that the models are now shared by all nodes in each group consisting of four elements (clusters $0-3$ and $4-7$). In the last step (step D), all elements are part of the same group, which means that all nodes in the network know all eight models.

3.2 Distributed Ensemble Construction

After dissemination, each node stores the set of models:

$$\mathcal{M} = \{M_1, M_2, ..., M_N\} \tag{3}$$

For an input sample x, each classifier produces a prediction:

$$y_i = M_i(x) \tag{4}$$

Predictions are combined using ensemble rules. In this work, we evaluate three strategies:

Majority Voting

$$y = \text{mode}(y_1, y_2, ..., y_N) \tag{5}$$

Sum Rule

$$y = \arg\max_{c} \sum_{i=1}^{N} P_i(c|x) \tag{6}$$

Product Rule

$$y = \arg\max_{c} \prod_{i=1}^{N} P_i(c|x) \tag{7}$$

This ensemble allows nodes to integrate knowledge learned from datasets distributed across the network.

4 Experimental Setup

This section presents the classifier, the dataset, and the three test scenarios used to evaluate our decentralized ensemble learning proposal.

4.1 Implementation Details

The classification algorithms were implemented using the WEKA platform [6], which provides a Java library with several machine learning algorithms and tools for data preprocessing and performance evaluation.

We adopted a Multi-Layer Perceptron (MLP) as the base classifier due to its suitability for multi-class problems. The network consists of 12 input neurons, three hidden layers with 16, 32, and 64 neurons, and an output layer with six neurons corresponding to the target classes. Hidden layers use sigmoid activation functions. The model was trained using WEKA default parameters, with learning rate 0.3, momentum 0.2, and 500 epochs.

Both the MLP training and the vCube communication strategy were implemented in Java 8 (64-bit). Experiments were executed on a dedicated machine running Windows 11 with an Intel Core i7 (8th generation) processor, 12 GB RAM, and 1 TB HDD storage.

In the proposed architecture, nodes exchange trained models without performing weight aggregation. Model integration occurs through prediction-based combination, where classifiers share their outputs while keeping their original parameters unchanged.

Execution time is used as a performance metric to compare centralized and distributed scenarios. Since models are trained independently on local data, the distributed approach aims to reduce training time while maintaining coordination through opinion-based model integration.

4.2 The Dataset

We use the dataset "MotionSense: Sensor-based human activity and attribute recognition" [12] to evaluate the performance of our distributed learning proposal. This dataset consists of time series data generated by accelerometer and gyroscope sensors (altitude, gravity, user acceleration, and rotation rate). Data were collected from 6 activities performed by 24 participants of different genders, ages, weight and height. Each participant performed 3 trials of the Downstairs

activity, 3 trials of the Upstairs activity, 3 trials of the Walking activity, 2 trials of the Jogging activity, 2 trials of the Sitting activity, and 2 trials of the Standing activity. All these tests were conducted in the same environment and under the same conditions. A total of 1,412,865 data set items were generated and distributed as shown in Table 1.

Table 1. Distribution of instances among the six classes present in the dataset

Class	Number of Instances	Proportion
Downstairs	131,856	9.33%
Jogging	134,231	9.5%
Upstairs	157,285	11.13%
Standing	306,427	21.69%
Sitting	338,778	23.98%
Walking	344,288	24.37%

In this way, the dataset is suitable for the proposed work as it contains a considerable number of records, which is common in distributed data scenarios.

4.3 Test Scenarios

The first experiment consists of a centralized classification process, referred to as *global classification*. In this setting, the entire dataset is available at a single node and split into training and test sets, preserving the original class proportions. The training set contains 70% of the original dataset (989,004 samples), while the remaining 30% (423,861 samples) is used for evaluation.

In the distributed setting, eight nodes (numbered from 0 to 7) receive partitions of the training data to build local models. Data distribution follows three different scenarios.

In the first scenario, data is evenly distributed among nodes. The dataset contains approximately 990,000 records, so each node receives about 123,000 samples while preserving the original class proportions.

In the second scenario, nodes receive different numbers of samples, but class proportions remain unchanged. Thus, reductions in dataset size are applied proportionally across all classes.

In the third scenario, training data is distributed without preserving class stratification, creating an unbalanced distribution. This scenario simulates situations in which nodes contain more information about certain classes than others, a common characteristic of non-IID data distributions [11].

These scenarios allow us to evaluate the impact of different levels of data heterogeneity on the proposed architecture.

In the distributed setting, nodes share only trained models rather than raw data. Model integration is therefore performed through classifier combination

strategies [9], including *Sum Rule* (aggregation of probability outputs), *Product Rule* (multiplication of class probabilities), and *Majority Voting* (selection of the most frequent prediction).

To evaluate performance, the same test dataset (30% of the original data) is used for both centralized and distributed experiments.

For the distributed scenarios, the vCube topology is used to disseminate trained models among nodes, allowing each node to obtain models trained across the network. The resulting system forms a system of multiple classifiers (SMC) composed of eight classifiers, labeled C_i, where i indicates the node where the classifier was trained (C_0 to C_7).

Experiments consider both fault-free and faulty conditions. Nodes may become temporarily unavailable due to network issues, hardware faults, or maintenance, reflecting realistic distributed environments. Because the decision process is decentralized, the vCube architecture remains operational even when some nodes are unavailable. In such cases, predictions are computed using the remaining models, which may slightly reduce ensemble diversity but does not interrupt system operation.

5 Results

In this section, we present the results obtained for each of the three scenarios outlined in the previous section. The results are first analyzed individually and then consolidated. We conducted tests under both fault-free and faulty conditions. For comparison, we initially implemented a centralized approach using the full dataset, followed by simulations with a decentralized tree-based approach.

In all simulations, only the training instances were loaded, as the test data (representing 30% of the original dataset) was excluded. In distributed tests, we allocated all classes present in the input file (Stairs Down, Stairs Up, Walking, Jogging, Sitting, and Standing) among the nodes based on the specific scenario.

For each isolated class, the percentage of data allocated to each node was defined. For instance, when distributing a class across eight nodes, each node received 12.5% of the available instances.

Although the experiments were conducted with eight nodes, the vCube topology is defined for networks of size 2^d and scales logarithmically with respect to the number of communication rounds. Therefore, the dissemination mechanism remains applicable to larger systems, where the benefits of hierarchical communication structures become even more significant.

5.1 Global Classification Approach

The global classification, which simulates the centralization of training data, was initially performed as a baseline for comparison. We used 70% of the dataset for training and 30% for testing.

This approach offers the advantage of having all available training examples in a single location, allowing the model to learn from the complete dataset.

However, it comes with drawbacks, including the need to consolidate data at a central point, which can create network bottlenecks and expose potentially confidential information to eavesdropping attacks.

In this scenario, the classifier required 20 h and 13 min to train, achieving an accuracy of 82.98%. This is a strong result, considering the problem involves six classes, where random classification would yield an accuracy of only 16%.

5.2 Tests Without Failure Nodes

We first analyze the results in a scenario where nodes operate independently, without communication. Table 2 presents the training time and individual accuracy of each node across the three scenarios described in Sect. 4.

In the first scenario, each node received an equal amount of data from each class. Training times ranged from 2 h and 29 min to 2 h and 55 min, which was expected since all nodes processed the same volume of data. Compared to the global training strategy (20 h and 13 min), this configuration significantly reduces the training time due to the parallel execution of local training processes.

Regarding classification performance, most classifiers (five in total) achieved similar accuracy values, around 43%, while classifiers C_6 and C_7 obtained higher accuracies. Although the datasets maintain the same class proportions, the actual instances assigned to each node differ due to the random partitioning of the training data. As a result, the test set may contain patterns that are closer to the training data of certain nodes, which can lead to variations in classification performance.

Table 2. Individual training on each node in the three scenarios

Classifier	Scenario 1		Scenario 2		Scenario 3	
	Time	Accuracy (%)	Time	Accuracy (%)	Time	Accuracy (%)
C_0	2h50	43.21	4h11	51.80	1h52	32.83
C_1	2h36	43.96	4h16	52.92	1h56	39.16
C_2	2h29	45.41	3h57	47.34	2h57	45.00
C_3	2h29	45.57	4h09	49.94	2h39	47.72
C_4	2h40	38.74	1h02	42.40	2h43	38.37
C_5	2h29	42.53	0h53	42.30	2h04	33.24
C_6	2h38	56.62	0h54	32.45	3h15	58.77
C_7	2h55	54.89	1h01	33.14	3h01	55.64

In scenario 2, the distribution was performed as follows: nodes 0, 1, 2 and 3 received 80% of the training data volume, i.e., each received 20% of the datasets, while nodes 4, 5, 6, and 7 shared the other 20% of the data, i.e., each one received 5% of the datasets. Despite the imbalance in the number of instances present at each node in the network, the relationship between the classes remains following

the original set. As expected, the more information concentrated in a node of the network, the higher the training timing and the accuracy is also better.

In scenario 3, the data was randomly distributed in such a way that both the number of data sets and the proportion of classes present in each node can vary greatly. The configuration of the instances between the eight nodes is shown in Table 3. The values correspond to the percentage of each class present in each node. The Set size column shows the number of instances in each node.

Table 3. Distribution of instances across the nodes for the third scenario

Node	Classes (%)						Set Size
	Downstairs	Upstairs	Walking	Jogging	Sitting	Standing	
0	30	8	8	8	8	8	99,429
1	8	30	8	8	8	8	103,343
2	8	8	30	8	8	8	131,293
3	8	8	8	30	8	8	126,311
4	8	8	8	8	30	8	132,141
5	8	8	8	8	8	30	99,793
6	15	15	15	15	15	15	148,448
7	15	15	15	15	15	15	148,448

The accuracy and the time spent training the models are also shown in Table 2. It can be seen from the values that half of the nodes performed below 40%, with C_0 at 32.83%, C_1 at 39.16%, C_4 at 38.37% and C_5 at 33.24%. The classifiers C_2 and C_3 had a classification accuracy of 45% and 47.72%. Finally, the models C_6 and C_7, which received the most data, had an accuracy of over 55% predicting correctly 58.77% and 55.64% of the instances, respectively.

As expected and demonstrated in the three scenarios, nodes with a similar number of records had a similar training time, again confirming a direct relationship between the number of samples and the learning time. We can see that classifiers with low accuracy trained their models faster, while models with higher accuracy took longer.

Once each model is trained locally, the nodes start to communicate with neighbors, according to the vCube topology, and combine the models hierarchically. Table 4 shows the accuracy rounds to each scenario and combination rule (sum, product and vote).

Step 1. In the first round, each node exchanges messages with its nearest neighbor and shares its classification model. In the first scenario, the accuracy of the approach increased with the sum rule and vote rules. When the product rule is used, the increase in accuracy is smaller and, in some cases, the percentage of correct answers decreases. The group formed by nodes 6 and 7, achieves a

higher performance compared to the others. This can be explained by the fact that these nodes obtained the highest accuracies during individual training.

In the second scenario, the first two groups achieve better results in all approaches than the last two groups. This is because in this scenario, nodes 1, 2, 3 and 4 contain 80% of the data sets, while nodes 5, 6, 7 and 8 only had 20% of the samples in the training set.

In scenario 3, the first group achieved a hit rate of 48.21% with the sum rule, reflecting the low individual performance of the nodes in this group. Similarly, the third group achieved a hit rate of 43.21%. The second group had a success rate of 59.91%, which was higher than the previous groups. Finally, the last group achieved a hit rate of 67.61%, reflecting the high performance of the nodes in this group (mainly due to the sets containing the largest amount of training instances). Concerning other classifier combination approaches in this scenario, a similar behavior as for the sum rule can be observed, i.e. for both majority voting and the product rules, groups 1 and 3 perform worse than the others. The second group achieved an average performance, while the fourth group achieved the best performance.

Table 4. Accuracy (%) of the vCube-based SMC for the first scenario (without failures) over the three phases of the process

Step	Groups	Scenario 1			Scenario 2			Scenario 3		
		Sum	Prod.	Vote	Sum	Prod.	Vote	Sum	Prod.	Vote
1	C_0, C_1	53.96	44.62	51.61	63.81	51.66	61.53	48.21	31.76	47.37
	C_2, C_3	58.37	48.26	54.15	61.33	51.31	55.04	59.91	46.07	56.04
	C_4, C_5	50.44	39.41	47.95	50.12	43.94	47.51	43.21	35.74	40.22
	C_6, C_7	69.33	64.29	67.51	37.71	31.71	37.18	67.61	64.71	64.41
2	C_0, ... C_3	60.02	48.46	58.46	66.79	56.22	65.01	52.31	42.49	51.76
	C_4, ..., C_7	61.35	51.51	60.84	45.19	37.48	42.93	61.04	52.16	58.11
3	C_0, ..., C_7	66.23	51.65	64.66	61.21	46.71	61.12	66.04	50.15	64.93

Step 2. In the second round, there are two groups, each consisting of four models. In the first scenario, when combining two classifiers, the accuracy of the first group is 60.02% according to the sum rule, a value slightly higher than the group formed from classifiers C_3 and C_4 (in the previous round), i.e. a small gain was achieved. For the majority voting, the gain was bigger, as the group obtained an accuracy of 58.46%, while the accuracy of classifiers C_3 and C_4 in the previous round was 54.15%. For the product rule, the accuracy of the first group was 48.46%, i.e. a minimal accuracy gain was achieved compared to the previous round. In this case, the approach has a much lower performance than the product and the majority rules. The second group ($C_4 - C_7$) achieves the best accuracy in the three combination scenarios. This is because the nodes with

the best individual performances (classifiers C_6 and C_7) are in this group. For the sum rule, the accuracy was 61.35%, achieving a lower performance than the group of models C_6 and C_7 (in the previous round), which was 69.33%, i.e. the performance of the models trained on nodes 4 and 5 had a negative impact on the accuracy of the second group. For the majority rule, the accuracy of the group was 60.84%, which also represents a decrease in accuracy compared to the groups formed in the previous round. Finally, for the product rule, the accuracy of the group was 51.51%, significantly lower than in the previous round.

In scenario 2, the accuracy of the first group based on the sum, majority voting and product rules have significant increase in accuracy when the four classification models were combined. For the second group, when classification models with low performance were combined, the resulting classification model tended to have lower performance.

In the third scenario, the accuracy of the first group was 52.31% for the sum rule, 51.76% for the majority rule and 42.49% for the product rule. In this group, nodes that performed poorly pulled the overall success rate. When the classifiers from the second group were combined, the accuracy was 61.04% for the sum rule, 58.11% for the majority rule and 52.16% for the product rule. As with the previous group, it was found that despite the nodes with good individual performance, the overall accuracy of the group tended to decrease due to the influence of some weaker classifiers.

Step 3. In the last round of execution, a single group is formed from all classifiers, so that each node has all classification models in the network, allowing the creation of a combination model from all models. In Scenario 1, the sum rule achieved the best accuracy with 66.23% correct answers, followed by the majority rule with an accuracy of 64.66% and the product rule with the worst performance. In Scenario 2, the sum rule also achieved the best performance with an accuracy of 61.21%. Despite the better performance, the result was very similar to that of the majority rule, whose accuracy was 61.12%. Finally, the product rule achieved a performance of 46.71%, mainly penalized by nodes with low performance. In Scenario 3, despite the low hit rate of most nodes in the network, the accuracy increases significantly when combining classification models. This is because the nodes can offer complementary information about the data of the entire system. This fact is confirmed by the fact that the hit rate of the SMC was 66.04% for the sum rule, 64.93% for the majority rule and 50.15% for the product rule.

It is important to note that the proposed solution supports varying the number of distributed nodes, which can lead to reduced training time due to parallelization. However, when the overall dataset remains fixed, increasing the number of nodes may result in smaller local datasets per node, potentially degrading the accuracy of local models and consequently, the performance of the aggregated global model.

Tests with Node Failures. In these tests, one node is considered fault. For a more robust evaluation, we conducted eight executions of the vCube algorithm, systematically removing one node per execution to simulate a failure in model transmission. This left seven nodes available for the combination of classifiers.

Table 5 presents the accuracy values obtained after the third iteration of vCube. Each row represents a scenario where a specific network node is considered faulty. The best performance for each of the eight cases is highlighted in bold, while the worst-case value is underlined.

Table 5. Performance considering each node in the network as a point of failure

Fault Node	Scenario 1			Scenario 2			Scenario 3		
	Sum	Prod.	Vote	Sum	Prod.	Vote	Sum	Prod.	Vote
0	66.43	65.39	51.88	59.01	58.62	45.99	65.82	64.41	52.43
1	64.32	63.39	52.69	57.45	56.77	45.30	**67.75**	66.28	50.61
2	64.31	63.34	51.79	58.28	58.31	44.89	65.37	64.88	52.47
3	64.30	62.86	50.70	60.14	58.99	45.32	61.93	60.65	51.09
4	**68.03**	67.32	55.10	61.72	61.03	46.50	64.38	63.58	50.91
5	65.89	63.99	51.85	61.82	60.36	46.36	66.10	64.73	50.10
6	63.22	61.77	48.62	61.22	61.34	49.91	63.41	62.02	45.46
7	64.04	62.40	48.95	**62.18**	61.20	48.61	62.11	60.97	47.55

Although the results are comparable to Scenario 1 without failure, it was found that the accuracy of the combined model increases when a low-performing node is excluded, as can be seen when Node 4 was excluded and the model accuracy was 68.03%. Similarly, the performance of the combined model decreases when a node with good assertiveness is excluded, as can be seen when Node 6 was excluded and the performance of the combined model dropped to 63.22%.

In the second scenario, the best performance over the eight executions (62.18%) was achieved with the sum combination strategy when we excluded node 7, which was already expected as it is one of the nodes that only received 5% of the training data. The worst hit rate occurred when node 2 was excluded. Considering that this is one of the elements that received the most training data (20%), it was expected that it could lead to a lower performance of the system when it is removed. With this in mind, it should be re-emphasized how important it is that a fault-tolerant system such as vCube ensures that the process only runs when all nodes in the network can submit their models, as the absence of a model can have a detrimental effect on the accuracy of the entire network.

In scenario 3, as noted in the previous scenarios, the best performance is achieved when a node with low representation is present. In this case, this was the case when node 1 was faulty, resulting in an accuracy of 67.75% for the sum rule. The worst performance was achieved when node 7 was excluded, resulting in a hit rate of 45.46% for the combination based on the product rule.

The results after combining the remaining models showed that the system can handle the failure of a network node. However, the individual performance of the classes directly depends on the distribution of the classes among the nodes. If the instances of a less frequent class are stored in a failed node, the system may not be able to handle that class.

By comprehensively analyzing the obtained results, the combination strategy with the product rule always obtained the worst results. The fusion of classifiers based on the sum rule, on the other hand, showed the best performance and only just outperformed the combination based on the majority rule. In addition to the performance of the combination techniques, they always perform worse than strategies using vCube, regardless of the scenario. The only exception is the use of the product rule, which performed worse than the individual classification.

Furthermore, it was observed that even in cases where one of the nodes fails, the vCube performance was always better than the best individual performance, regardless of which node had a communication failure. This shows that vCube is suitable as a strategy for model integration of isolated trained models and is a good alternative for the management of distributed learning systems.

It should also be emphasized that the classification model generated in each node was on average 45KB in size, i.e., in a scenario with 8 nodes in the system, each node can receive 7 classification models simultaneously, generating a total load of 315 KB. On the other hand, if it were possible to move the data for training the classification model to a specific node, 113 MB of information would have to be moved, an amount of data that is approximately 360 times larger.

5.3 Comparison with Baseline Approaches

To compare the approach proposed in this paper, a centralized ensembling and a decentralized tree [19] simulation were performed to evaluate the average accuracy when each of the network nodes fails.

Table 6 presents a comparison of the results obtained by the decentralized and vCube approaches in Scenario 1. In general, the distributed learning system with vCube has a higher average accuracy than the decentralized tree topology because in this strategy, more than one classification model is lost when a node fails, especially the nodes closest to the root of the tree.

Table 6. Comparison between the decentralized tree and the vCube approach

Topology	Scenario 1			Scenario 2			Scenario 3		
	Sum	Vote	Prod.	Sum	Vote	Prod.	Sum	Vote	Prod.
vCube	**65.07**	63.81	51.45	60.23	59.58	46.61	64.61	63.44	50.08
Decentralized Tree	54.89	53.70	44.57	49.92	48.94	40.12	52.45	50.94	42.46

The comparison of the results achieved with the two strategies for Scenario 2 is also shown in Table 6. As in Scenario 1, the vCube-based topology has a higher

average accuracy compared to the decentralized tree topology, again due to the loss of the classification models contained in the nodes of the latter approach.

Finally, Table 6 outlines the results of the two approaches in Scenario 3. As observed in the first two scenarios, the peer-to-peer topology has a higher average accuracy than the centralized tree topology.

In a centralized model (omitted in the Table 6), the accuracy result would be equivalent to vCube, but the failure of the central node would render the unified model unavailable. An all-to-all strategy would guarantee the sharing of the remaining models in the event of a failure of one or more nodes. However, this requires a quadratic number of model shares, whereas in vCube, the number of shares is at most $n \log n$.

6 Conclusion

This work presented a distributed ensemble learning architecture based on the vCube topology, a hierarchical hypercube structure designed to support scalable and fault-tolerant communication among distributed nodes. The proposed architecture allows nodes to train models locally and disseminate them efficiently without transferring raw data.

The experimental evaluation used a dataset with approximately 1.4 million samples distributed across eight nodes under three scenarios that varied in data balance and class stratification. We evaluated three configurations: (i) independent classifiers without communication, (ii) distributed model sharing using the vCube topology, and (iii) scenarios involving node failures.

Results show that the vCube-based architecture improves classification accuracy compared to standalone classifiers and outperforms a decentralized tree-based communication topology. Moreover, the vCube structure demonstrated resilience to node failures, allowing the system to continue operating even when nodes become temporarily unavailable.

Another important finding is that distributed training significantly reduces computational time compared to centralized training while minimizing communication overhead. Across all scenarios, classifier combination using summation and majority voting rules consistently outperformed individual classifiers, highlighting the benefits of model dissemination and ensemble integration.

These results emphasize the importance of communication topology in distributed learning systems. By leveraging the logarithmic communication structure of vCube, the proposed approach enables scalable model dissemination while maintaining robustness in dynamic environments. Although experiments were conducted with eight nodes, the topology supports networks of size 2^d, allowing logarithmic scaling with respect to communication rounds.

Future work will extend the evaluation to larger networks, additional datasets, and more diverse data distributions to better understand the impact of data heterogeneity. Another direction is to investigate communication costs and scalability when disseminating models in larger and more dynamic environments [4]. We also plan to explore adaptive model integration strategies and alternative

learning algorithms, including deep learning models, to further improve ensemble diversity and prediction performance.

References

1. de Araujo, J.P., Arantes, L., Duarte, E.P., Rodrigues, L.A., Sens, P.: VCube-PS: a causal broadcast topic-based publish/subscribe system. JPDC **125**, 18–30 (2019). https://doi.org/10.1016/j.jpdc.2018.10.011
2. Dietterich, T.G.: Ensemble methods in machine learning. In: Kittler, J., Roli, F. (eds.) MCS 2000. LNCS, vol. 1857, pp. 1–15. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45014-9_1
3. Duarte, E.P., Bona, L.C.E., Ruoso, V.K.: VCube: a provably scalable distributed diagnosis algorithm. In: 5th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems, pp. 17–22 (2014). https://doi.org/10.1109/ScalA.2014.14
4. Duarte, E.P., Weber, A., Fonseca, K.V.: Distributed diagnosis of dynamic events in partitionable arbitrary topology networks. IEEE Trans. Parallel Distrib. Syst. **23**(8), 1415–1426 (2011). https://doi.org/10.1109/TPDS.2011.284
5. Duarte, E.P., Jr., Rodrigues, L.A., Camargo, E.T., Turchetti, R.C.: The missing piece: a distributed system-level diagnosis model for the implementation of unreliable failure detectors. Computing **105**(12), 2821–2845 (2023). https://doi.org/10.1007/s00607-023-01211-8
6. Frank, E., Hall, M.A., Witten, I.H.: The WEKA Workbench. Online Appendix for Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, San Francisco, California, Burlington (2016)
7. Jeanneau, D., Rodrigues, L.A., Arantes, L., Duarte, E.P., Jr.: An autonomic hierarchical reliable broadcast protocol for asynchronous distributed systems with failure detection. JBCS **23**, 1–14 (2017). https://doi.org/10.1186/s13173-017-0064-9
8. Kairouz, P., et al.: Advances and open problems in federated learning. Found. Trends Mach. Learn. **14**(1–2), 1–210 (2021). https://doi.org/10.1561/2200000083
9. Kittler, J., Hatef, M., Duin, R., Matas, J.: On combining classifiers. IEEE Trans. Pattern Anal. Mach. Intell. **20**(3), 226–239 (1998). https://doi.org/10.1109/34.667881
10. Lian, X., Zhang, C., Zhang, H., Hsieh, C.J., Zhang, W., Liu, J.: Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, pp. 5336–5346. Curran Associates Inc., Red Hook, NY, USA (2017). https://dl.acm.org/doi/10.5555/3295222.3295285
11. Lu, Z., Pan, H., Dai, Y., Si, X., Zhang, Y.: Federated learning with non-IID data: a survey. IEEE Internet Things J. **11**(11), 19188–19209 (2024). https://doi.org/10.1109/JIOT.2024.3376548
12. Malekzadeh, M., Clegg, R.G., Cavallaro, A., Haddadi, H.: Mobile sensor data anonymization. In: Proceedings of the International Conference on Internet of Things Design and Implementation, IoTDI 2019, pp. 49–58. ACM, New York (2019). https://doi.org/10.1145/3302505.3310068
13. McMahan, H.B., Moore, E., Ramage, D., Hampson, S., Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: International Conference on Artificial Intelligence and Statistics, pp. 1273–1282. PMLR (2017). https://proceedings.mlr.press/v54/mcmahan17a

14. Peteiro-Barral, D., Guijarro-Berdinas, B.: A survey of methods for distributed machine learning. Prog. Artif. Intell. **2**(1), 1–11 (2013). https://doi.org/10.1007/s13748-012-0035-5
15. Rodrigues, L.A., Arantes, L., Duarte, E.P.: An autonomic majority quorum system. In: IEEE 30th AINA, pp. 524–531. IEEE (2016). https://doi.org/10.1109/AINA.2016.73
16. Rodrigues, L.A., Duarte, E.P., de Araujo, J.P., Arantes, L., Sens, P.: Bundling messages to reduce the cost of tree-based broadcast algorithms. In: LADC, pp. 115–124. IEEE (2018). https://doi.org/10.1109/LADC.2018.00022
17. Rokach, L.: Ensemble-based classifiers. Artif. Intell. Rev. **33**(1–2), 1–39 (2009). https://doi.org/10.1007/s10462-009-9124-7
18. Saeed, N., Ashour, M., Mashaly, M.: Comprehensive review of federated learning challenges: a data preparation viewpoint. J. Big Data **12**(1) (2025). https://doi.org/10.1186/s40537-025-01195-6
19. Verbraeken, J., Wolting, M., Katzy, J., Kloppenburg, J., Verbelen, T., Rellermeyer, J.S.: A survey on distributed machine learning. ACM Comput. Surveys (CSUR) **53**(2), 1–33 (2020). https://doi.org/10.1145/3377454
20. Zaharia, M., et al.: Apache spark: a unified engine for big data processing. Commun. ACM **59**(11), 56–65 (2016). https://doi.org/10.1145/2934664