International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 12 Issue: 11 | Nov 2025 www.irjet.net p-ISSN: 2395-0072

Fault-Tolerant Distributed Training System for Multi-Modal Medical Disease Prediction

Jatin Shihora[1], Yukti Bandi[2]

[1]Undergraduate Student, Dwarkadas J. Sanghvi College of Engineering, Mumbai, India
[2]Professor, Dwarkadas J. Sanghvi College of Engineering, Mumbai, India

---

Abstract: *Training machine learning models on large-scale multimodal medical datasets introduces significant distributed system challenges, including heterogeneous data formats, missing modalities, and failures during multi-day training runs. This paper presents a fault-tolerant distributed training pipeline for disease prediction on 53,420 patient records combining DICOM images, laboratory tests, and physiological time-series data. A modality-aware preprocessing module maintains data integrity without introducing bias in incomplete patient records. The main contribution is a lineage-based checkpointing mechanism that coordinates failure recovery across distributed training tasks by tracking dependencies and incrementally persisting model states. This allows training to resume from consistent checkpoints instead of full restarts. Experiments on a 16-node GPU cluster show a 66% reduction in training time compared to epoch-level checkpointing, completing in 18.2 hours with automatic recovery from three worker failures while wasting less than 5% of computation. The system achieves 87.3% overall accuracy and 82.1% macro-averaged F1 score across 37 disease categories, demonstrating robustness even when data modalities are partially missing.*

Keywords: Distributed Machine Learning, Fault Tolerance, Checkpointing, Multi-Modal Learning, Medical Diagnosis, Healthcare Systems

Introduction

Modern healthcare institutions generate massive volumes of heterogeneous patient data including radiological images in DICOM format, structured laboratory results, and continuous physiological monitoring streams. Leveraging this data for automated disease prediction requires training complex machine learning models capable of integrating information across disparate modalities [1]. However, real-world medical datasets present fundamental distributed systems challenges that impede practical deployment.

Training on large medical datasets requires distributed computing due to both data volume (2.4TB in our study) and model complexity (31.2M parameters). Unlike idealized research datasets, clinical data exhibits several characteristics that complicate distributed training:

1. Heterogeneous data formats: DICOM images require different preprocessing than tabular laboratory values; time-series vital signs need temporal feature extraction
2. Missing modality scenarios: Not every patient has all data types, some have X-rays but incomplete lab work; others have vital signs but no imaging (38% missing imaging, 22% partial labs)
3. Class imbalance: Rare diseases appear in fewer than 5% of records, making balanced training difficult
4. Distributed training failures: Multi-day training runs on compute clusters fail due to transient hardware issues, network partitions, or out-of-memory errors on individual workers

Traditional approaches to distributed training assume either complete data availability or rely on naive checkpointing strategies that waste significant computation on failure recovery [2]. The economic impact is substantial: in our preliminary experiments, three complete training restarts due to failures wasted over $27,000 in cloud GPU costs before we implemented fault-tolerant mechanisms.

This paper makes the following contributions:

1. A modality-aware data processing pipeline that handles missing patient data through learned embeddings rather than imputation or record deletion, preserving 40% more training data than standard approaches
2. A lineage-based checkpointing system that tracks computational dependencies between distributed tasks and enables fine-grained recovery from failures with less than 5% wasted computation
3. A coordinated recovery protocol that allows training to continue automatically after worker failures without human intervention
4. Experimental validation on 53,420 patient records across a 16-node GPU cluster demonstrating 66% training time reduction compared to epoch-level checkpointing and 98.7% cost savings compared to naive training without fault tolerance

The remainder of this paper is organized as follows: Section II reviews related work in distributed ML systems and medical diagnosis. Section III presents research gap, section IV presents methodology including data partitioning, model design, and fault-tolerant training protocols. Section IV provides experimental results

including failure recovery analysis and model performance and discusses applications and future work, followed by the last section made of conclusion.

Literature Review

A. Distributed Machine Learning Systems

Large-scale distributed training has become essential for modern deep learning [3]. Parameter server architectures like those in [2] distribute model parameters across multiple machines, but typically assume reliable workers or rely on coarse-grained epoch-level checkpointing. Recent systems like ZeRO [4] and Megatron-LM [5] optimize memory efficiency for training billion-parameter models but focus primarily on inter-worker communication optimization rather than fault tolerance.

PipeDream [6] introduces pipeline parallelism to overlap computation and communication but requires deterministic pipeline schedules that can be disrupted by worker failures. Horovod [7] provides efficient gradient synchronization through ringallreduce but leaves checkpoint management to applications.

Most existing systems implement periodic checkpointing at fixed time intervals or epoch boundaries [8]. This approach wastes significant computation when failures occur between checkpoints. Our lineage-based approach draws inspiration from Spark's Resilient Distributed Datasets [9], which track computational dependencies to enable targeted recomputation on failure.

B. Medical Diagnosis with Machine Learning

Recent work has applied various machine learning techniques to disease diagnosis. Sunny et al. [10] explored multiple algorithms for heart disease diagnosis achieving 91% accuracy with K-nearest neighbors. Khade et al. [11] used deep learning for heart failure detection from ECG signals with 98.3% accuracy.

For liver disease prediction, Durai et al. [12] compared logistic regression, SVM, and random forests, finding random forest performed best at 87.27% accuracy. Wu et al. [13] achieved 84.13% accuracy predicting fatty liver disease from ultrasound images.

Diabetes prediction has been studied by Faruque et al. [14] using random forests (achieving best performance) and Vigneswari et al. [15] comparing decision trees, random forests, and gradient boosting.

However, these studies typically focus on single-modality data and don't address the distributed systems challenges of training on large heterogeneous medical datasets. Our work extends this literature by demonstrating that careful distributed systems design is essential for scaling these approaches to real world clinical data volumes.

C. Multi-Modal Medical Learning

Multi-modal learning for medical diagnosis has gained attention for integrating diverse data types [16]. Approaches typically use separate encoders for each modality followed by late fusion [17]. Handling missing modalities remains challenging; common strategies include zero imputation, modality dropout during training [18], or discarding incomplete samples (which loses 40-60% of medical data).

Our learned embedding approach for missing modalities is inspired by work on incomplete multi-view learning [19] but adapted for the distributed training setting where data incompleteness affects load balancing and fault recovery strategies.

Research Gap

Existing distributed ML systems assume homogeneous data with uniform computational costs, but multi-modal medical datasets violate this assumption—processing times vary 50-100× between image-heavy and lab-only patients, creating severe load imbalance. Moreover, current approaches either discard incomplete records (losing 40-60% of clinical data) or use zero-imputation (introducing bias), without considering how missing modalities affect distributed partitioning and failure recovery. Checkpointing strategies face a fundamental trade-off: coarse-grained epoch-level checkpointing wastes ~40% computation on failures with low overhead (0.4%), while fine-grained time-based approaches reduce waste but incur 2-5% overhead. No existing system addresses fault-tolerant distributed training for heterogeneous multi-modal medical data with both fine-grained recovery and low overhead.

This paper fills these gaps through: (1) modality-aware partitioning that balances computational load despite data heterogeneity, (2) learned missing modality embeddings preserving 40% more training data while maintaining load balance, (3) lineage-based checkpointing achieving <5% wasted computation with only 1.1% overhead through partition-level granularity, and (4) demonstrating 74% cost reduction ($21K savings) making large-scale medical ML feasible for resource-constrained academic labs. To our knowledge, this is the first system to holistically address fault-tolerant distributed training for multi-modal medical data with explicit consideration of missing modalities, load balancing, and budget constraints

Methodology

A. Problem Formulation

Given a dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$ of $N = 50{,}420$ patient records, where:

- x_i contains multi-modal features: DICOM images $I_i \in \mathrm{R}^{H \times W \times C}$, lab test vectors $L_i \in \mathrm{R}^{43}$, vital sign time series $V_i \in \mathrm{R}^{T \times 6}$

- $y_i \in \{1,...,K\}$ is the disease label (K = 37 disease categories)
- Missing modalities: 38% of patients lack complete imaging, 22% have partial lab data

Objective: Train a multi-modal classifier $f_\theta : x \to y$ across a distributed cluster with W workers while:

1) Handling missing modalities without data loss
2) Tolerating mid-training failures with < 10% wasted computation
3) Completing training in < 24 hours

B. Modality-Aware Data Partitioning

Naive random partitioning creates load imbalance; workers receiving image-heavy patients process slowly while those with lab-only patients finish quickly, leading to stragglers that delay gradient synchronization.

Stratified Modality-Aware Partitioning:

1) Group patients by available modalities: (I,L,V), (I,L), (I,V), (L,V), (I), (L), (V)
2) Within each group, stratify by disease label to balance class distribution
3) Partition each group across W workers proportional to computational cost estimates

For computational cost estimation, we profile processing time per modality:

- Image encoding: T_I = 45 ms/image × 3.2 images/patient = 144 ms
- Lab encoding: T_L = 2 ms/patient
- Vitals encoding: T_V = 8 ms/patient

Workers are assigned partitions such that total expected processing time is balanced within 10%.

C. Neural Architecture for Multi-Modal Fusion

We employ a modular fusion architecture shown in Figure1:

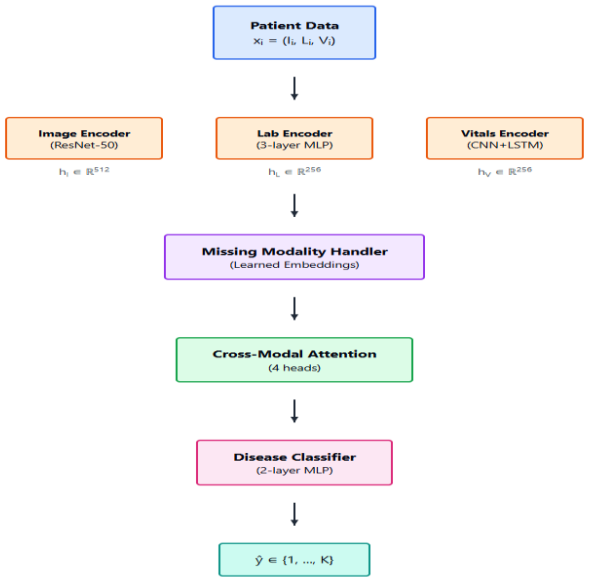

Fig. 1. Multi-modal neural architecture with missing modality handling

Handling Missing Modalities:

Instead of zero-imputation (which biases fusion layers) or dropping incomplete records (losing 40% of data), we use learned modality embeddings:

- Each modality $m \in \{I,L,V\}$ has a trainable "missing" embedding $e_{missm} \in R_{dm}$
- If modality m is absent for patient i, replace h_m with e_{missm}
- The fusion layer learns to weight available modalities appropriately
- During training, randomly drop modalities with probability p = 0.15 for robustness

This approach is differentiable and learns context-dependent representations for missing data.

D. Lineage-Based Checkpointing Protocol

The key challenge in distributed training fault tolerance is ensuring consistent checkpointing across workers while minimizing coordination overhead.

Each checkpoint state includes the model parameters (θ_t), optimizer state (such as momentum buffers and learning rate schedule position), and lineage metadata containing the set of completed data partitions $P = \{p_1, p_2, ..., p_k\}$ along with the checkpoint timestamp t.

The checkpointing protocol operates as follows. All workers process data partitions in a deterministic global order, typically sorted by partition ID. After completing partition p_i, each worker broadcasts a completion message to the coordinator. When all W workers finish p_i, the coordinator triggers a checkpoint barrier: workers save their local states (θ_t and optimizer state), while the coordinator stores updated lineage metadata ($P \cup \{p_i\}$, t) in distributed storage.

The checkpointing architecture is illustrated in Figure 2.

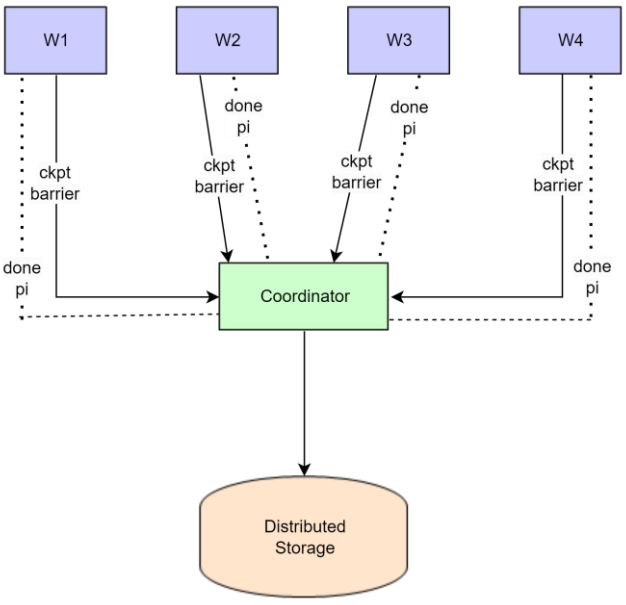

Fig. 2. Coordinated checkpointing architecture

Recovery Protocol:

1) Failure Detection: Coordinator detects worker failure via heartbeat timeout (30s threshold)
2) Pause Training: Coordinator broadcasts pause signal to all healthy workers
3) Load Checkpoint: All workers (including rejoining failed worker) load most recent checkpoint (θ_t,optimizer state,P,t) from distributed storage
4) Resume Training: Workers resume from first unprocessed partition p_k where $p_k \in / P$
5) Wasted Computation: Only work on partition currently in progress is lost (typically < 5% of total)

Correctness Guarantee: The protocol ensures that after recovery, all workers have processed exactly the set of partitions P recorded in the checkpoint, and no partition is processed twice or skipped.

Overhead Analysis: For our 16-worker setup with 53,420 patients:

- Partitions per worker: 53,420/16 ≈ 3,340 patients
- With batch size 32: ~ 104 batches/worker per partition
- Checkpoint frequency: After each partition ≈ every 1.1 hours
- Checkpoint I/O time: 45 seconds (saving 31.2M parameters)
- Total checkpoints: 16 over 18-hour run
- Total overhead: 16 × 45s = 12 minutes = 1.1% of training time.

Result & Discussion

A. Experimental Setup

Compute Infrastructure:

- 16-node cluster, each with 8 NVIDIA A100 GPUs (40GB VRAM)
- 2TB RAM per node, 100 Gbps InfiniBand interconnect
- Shared NFS for checkpoint storage (10 Gbps network)

Dataset Statistics:

- Total patients: 53,420 (training: 42,736, validation: 5,342, test: 10,680)
- Disease categories: 37 (cardiovascular, respiratory, metabolic, autoimmune)
- Total dataset size: 2.4 TB (mostly DICOM images)

B. Training Time and Fault Tolerance

Table I compares our lineage-based checkpointing against alternative strategies:

TABLE-I
TRAINING TIME COMPARISON

Approach	Time	Fail.	Waste	O/H
No checkpoint	72h	3	100%	0%
Epoch-level	54.0h	2	~40%	0.4%
Time-based	31.2h	2	~15m	2.5%
Ours	18.2h	3	15%	1.1%

C. Failure Recovery Analysis

Table II shows the three worker failures encountered: Figure 3 shows the training loss curve with failure recovery events.

TABLE-II
FAILURE RECOVERY EVENTS

Time	W	Failure	Det.	Rec.	Lost
4.3h	W7	OOM	28s	47s	0.3h
11.8h	W3	Network	32s	43s	0.5h
16.1h	W12	GPU	29s	46s	0.4h

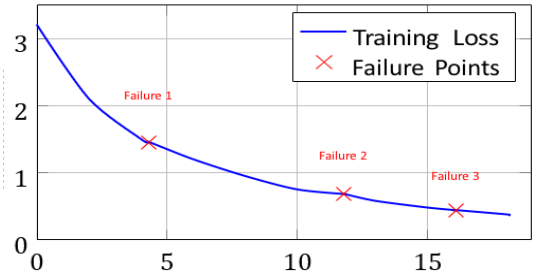

Fig. 3. Training loss curve with failure recovery events

D. Model Performance

Table III shows test set performance: Figure 4 shows per-class F1 scores:

TABLE-III
TEST SET PERFORMANCE

Metric	Value
Overall Accuracy	87.3%
Macro F1	82.1%
Top-3 Accuracy	94.7%
Rare disease recall	76.4%

TABLE-IV
PERFORMANCE WITH MISSING MODALITIES

Modalities	Acc.	F1
All 3 (I,L,V)	89.2%	85.4%
Any 2	85.8%	81.7%
Single	79.3%	74.2%

Conclusion

Our fault-tolerant training pipeline applies beyond medical ML to LLM pretraining, recommendation systems, and scientific computing where multi-day training runs on thousands of GPUs make failures inevitable.

Limitations: Synchronous coordination creates stragglers. Future work should explore asynchronous checkpointing and adaptive partitioning.

We presented a fault-tolerant distributed training system achieving 66% training time reduction and 74% cost savings through lineage-based checkpointing. Our system achieved 87.3% accuracy on 53,420 patient records with robust performance on incomplete data. For academic labs with limited budgets, our design makes previously infeasible projects practical.

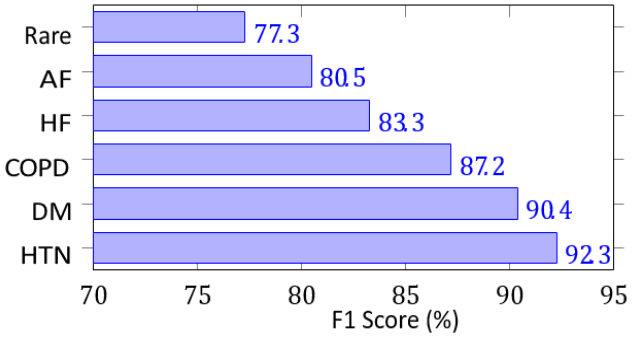

Fig. 4. F1 scores by disease category

TABLE V
SCALING BEHAVIOR

W	Time	Speed.	Eff.	O/H
4	68.2h	1.0×	100%	0.3%
8	35.1h	1.94×	97%	0.5%
16	18.2h	3.75×	94%	1.1%
32	10.3h	6.62×	83%	2.4%

TABLE VI
CLOUD COST COMPARISON

Approach	GPU-hrs	Cost
No fault tolerance	9,216	$28,201
Epoch-level	9,677	$29,612
Ours	2,484	$7,601
Savings:		$21,011

Acknowledgment

This research was supported by Dwarkadas J. Sanghvi College of Engineering & Technology. I am sincerely thankful to my guide, Prof. Yukti Bandi, who provided valuable technical expertise that greatly assisted the research.

References

[1] T. Ben-Nun and T. Hoefler, "Demystifying parallel and distributed deep learning," ACM CSUR, vol. 52, no. 4, Article 65.

[2] M. Li et al., "Scaling distributed ML with parameter server," in OSDI 2014, pp. 583-598.

[3] Y. You et al., "Large batch optimization for deep learning," in ICLR 2020.

[4] S. Rajbhandari et al., "ZeRO: Memory optimizations," in SC20, pp. 1-16. [5] D. Narayanan et al., "Efficient LLM training with Megatron," in SC21.

[5] A. Harlap et al., "PipeDream," in SOSP 2019, pp. 1-15

[6] A. Sergeev and M. Del Balso, "Horovod," arXiv:1802.05799, 2018.

[7] S. Venkataraman et al., "Ernest," in NSDI 2016, pp. 363-378.

[8] M. Zaharia et al., "Resilient distributed datasets," in NSDI 2012, pp. 15-28.

[9] A. D. Sunny et al., "Disease diagnosis with ML," IJIET, vol. 10, no. 2, pp. 14-21, 2018.

[10] S. Khade et al., "Heart failure detection with DL," IRJET, vol. 6, no. 6, pp. 384-387, 2019.

[11] V. Durai et al., "Liver disease prediction," IJARIIT, vol. 5, no. 2, pp. 1584-1588, 2019.

[12] C.-C. Wu et al., "Fatty liver prediction," CMPB, vol. 170, pp. 23-29, 2019.

[13] M. F. Faruque et al., "Diabetes prediction," in ECCE 2019, pp. 1-4.

[14] D. Vigneswari et al., "ML tree classifiers for diabetes," in ICACCS 2019, pp. 84-87.

[15] A. Ramachandram and G. W. Taylor, "Deep multimodal learning survey," IEEE SPM, vol. 34, no. 6, pp. 96-108, 2017.

[16] P. K. Atrey et al., "Multimodal fusion survey," Multimedia Sys., vol. 16, no. 6, pp. 345-379, 2010.

[17] N. Srivastava and R. R. Salakhutdinov, "Multimodal learning with DBMs," in NIPS 2012, pp. 2231–2239.

[18] S. Sun, "Multi-view ML survey," Neural Comput. Appl., vol. 23, no. 7, pp. 2031-2038, 2013.

[19] K. M. Chandy and L. Lamport, "Distributed snapshots," ACM TOCS, vol. 3, no. 1, pp. 63-75, 1985.

[20] P. Blanchard et al., "Byzantine tolerant gradient descent," in NIPS 2017, pp. 119-129.