Prototype-Based vs. Traditional Embeddings in Federated Graph Neural Networks: Inference Efficiency at Scale

Assignee Research

June 1, 2026

Abstract

This report synthesises findings from 11 peer-reviewed papers addressing the following research question: What is the comparative inference efficiency of prototype-based embeddings against traditional embeddings in federated graph neural networks under varying client scalability on standard benchmarks. Graph Neural Networks (GNNs) have experienced rapid advancements in recent years due to their ability to learn meaningful representations from graph data structures. However, in most real-world settings, such as financial transaction networks and healthcare networks, this data. 19 claims were extracted from source literature; 0 were independently verified against retrieved documents. An automated multi-reviewer quality assessment produced a score of 3.1/10. This report is a machine-generated literature synthesis and does not constitute original research.

1 Introduction

This paper examines: OptimES: Optimizing Federated Learning Using Remote Embeddings for Graph Neural Networks. Research question: What is the comparative inference efficiency of prototype-based embeddings against traditional embeddings in federated graph neural networks under varying client scalability on standard benchmarks?.

2 Methodology

Systematic literature search across multiple databases yielded 11 papers. Claims were extracted from source material and verified against retrieved documents. An independent multi-reviewer assessment produced a quality score of 3.1/10.

3 Results

11 papers retrieved. 19 claims extracted; 0 independently verified. Quality review score: 3.1/10.

4 Limitations

This report is a machine-generated literature synthesis and does not constitute original research. Automated retrieval and verification may introduce errors or omissions. Review scores reflect automated assessment, not human peer review. Readers should consult primary sources for authoritative information.

5 Extracted Claims

Claim	Verified	Confidence
The implementation uses DGL, PyTorch, and the Flotilla framework for federated learning.	×	0.08
DGL provides APIs for local training of GNN models, and PyTorch enables tensor operations like maintaining local embeddi	×	0.06
DGL leverages PyTorch as its backend for forward and backward passes during model training.	×	0.02
Flotilla framework is used for orchestration of model aggregation and local training.	×	0.10
The embedding server is implemented as a Redis server storing key, value pairs with node ID as key and embedding as va	×	0.03
Separate databases are maintained for each layer's embeddings to allow scoped updates.	×	0.02
Batched requests combined with pipelining are used to reduce latency for RPC requests for each remote node.	×	0.03
The experimental setup consists of eight GPU workstations, each with an Nvidia RTX 4090 GPU and an AMD Ryzen 9 7900X CPU	×	0.00
Each client workstation runs a Flotilla client instance to manage local training.	×	0.05
The Redis store with embeddings and the Flotilla aggregation service run on a server with an Nvidia RTX A5000 GPU and an	×	0.02
All machines are connected using a 1 Gbps Ethernet interface.	×	0.01
OptimES reduces communication, computation, and memory costs for the shared embeddings method.	×	0.06
OptimES leads to faster training time per round, faster training time to convergence, and minimum to no reduction in acc	×	0.12
OptimES does not share any additional data that may compromise privacy.	×	0.07
Pruning the neighbourhood of remote vertices reduces embeddings transferred, in-memory footprint, and compute costs for	×	0.07
Assigning scores to remote vertices and limiting the expansion of the local subgraph to high-scoring nodes reduces train	×	0.07
Overlapping the transfer of embeddings from clients to the embedding server with the final epoch of training hides netwo	×	0.08
Limiting clients to pull only the embeddings they will use at the start of each round reduces network communication time	×	0.05
Fetching a set of high-scoring nodes at the beginning of each round and retrieving remaining	×	0.03

References

- http://arxiv.org/abs/2507.09805v1
- http://arxiv.org/abs/2509.22922v1
- http://arxiv.org/abs/1909.10086v3