Project Proposal - BaryGNN

MATAN FAINZILBER - 209635762, AMIT RABINOVICH - 315857847, and IDO ROSINER - 209617000

TL;DR. We propose a geometry-aware graph–level pooling scheme that treats every node as a *distribution* in latent space and represents the whole graph by the 2-Wasserstein barycenter of those node distributions, yielding fixed-size, uncertainty-preserving embeddings for graph classification.

1 BACKGROUND & MOTIVATION

Graph–level classification underlies tasks from molecular property prediction to program-analysis security checks. Current Graph Neural Networks (GNNs) first encode every node as a latent vector h_v and then apply a permutation-invariant *pooling* operator such as mean, sum, max, attention, or hierarchical coarsening - to obtain a single graph vector g [Hamilton et al. 2017; Kipf and Welling 2017; Veličković et al. 2018]. Although effective, these point-based summaries overlook two key facts: node embeddings often carry *uncertainty* (e.g. due to stochastic encoders), and latent spaces may have non-Euclidean *geometry*.

1.1 Problem Statement

Standard graph pooling maps each node to a latent vector $h_v \in \mathbb{R}^d$ and then reduces the whole set $\{h_v\}_{v \in V}$ with a simple Euclidean operator—typically MEAN, SUM, or MAX. This averaging step erases any *distributional information* produced by stochastic encoders or dropout, so node-level uncertainty and multi-modes structure are lost. It also forces the pooled representation to lie in flat Euclidean space, ignoring the curved or clustered geometry that latent features may inhabit. We therefore aim to design a pooling layer that preserves the full distribution learned for every node and aggregates these distributions via the 2-Wasserstein barycenter, yielding a fixed-size graph embedding that respects the underlying geometry instead of collapsing it.

1.2 Why It Matters

Accurate graph embeddings directly improve predictive performance, robustness to noisy nodes, and sample efficiency (allowing models to reach a target performance with *fewer labelled graphs*). All vital for domains where graphs are small but information-rich, such as drug discovery and protein-interaction networks. Moreover, retaining per-graph uncertainty enables calibrated confidence estimates, a prerequisite for risk-aware downstream decision making.

1.3 Limitations of Prior Work

- **Point-based pooling** (mean/sum/max, attention) assumes Euclidean averaging and discards uncertainty [Kipf and Welling 2017; Veličković et al. 2018; Xu et al. 2019a].
- **Hierarchical coarsening** methods (e.g. DiffPool, SAGPool) learn soft clusters but still average point vectors inside clusters, losing distributional richness [Lee et al. 2019; Ying et al. 2018].
- **OT-based graph methods** focus on graph matching or edge-level transport [Vayer et al. 2020; Xu et al. 2019b]. none provide a fixed-dimension, barycentric *graph* embedding that integrates seamlessly with stochastic node encoders.

Authors' address: Matan Fainzilber - 209635762; Amit Rabinovich - 315857847; Ido Rosiner - 209617000.

Hence, a gap remains: a pooling operator that embeds graphs via the Wasserstein barycenter of node-level *distributions*, preserving uncertainty and latent geometry while guaranteeing a constant output size. Our project addresses this gap.

2 PROPOSED APPROACH

2.1 Core Idea

We model *each node* as a **latent distribution** rather than a point embedding, then summarise the whole graph by the **2-Wasserstein barycenter** of those node distributions. To obtain a *fixed-size* graph vector, we introduce a *global, learnable code-book* $C = \{c_1, \dots, c_m\} \subset \mathbb{R}^d$ (shared by *all* graphs). During training, every graph solves for a weight vector $w^{(G)} \in \Delta^{m-1}$ that minimises the OT cost between C and its node distributions, while the code-book atoms $\{c_j\}$ themselves are updated by back-propagation. Thus graphs differ *only* in the mass they place on the common support.

2.2 Technical Outline

(1) Stochastic node encoder. A message-passing GNN outputs k latent samples per node $\{z_v^{(1)}, \dots, z_v^{(k)}\} \subset \mathbb{R}^d$ (or, equivalently, a Gaussian $\mathcal{N}(\mu_v, \Sigma_v)$), yielding an empirical distribution $\mu_v = \frac{1}{k} \sum_\ell \delta_{z_v^{(\ell)}}$.

(2) Shared code-book initialisation. We treat the support atoms $\{c_j\}_{j=1}^m \subset \mathbb{R}^d$ as *learnable parameters* and initialise them in one of two ways:

- **Random Gaussian init (default)**: We sample each $c_j \sim \mathcal{N}(0, \sigma^2 I)$ with small variance. This allows the code-book to be learned end-to-end from scratch, even before any meaningful node embeddings are available.
- **Data-driven warm start (optional)**: After one warmup epoch, we collect latent node samples from the GNN and run k-means++ over them to re-initialise $\{c_j\}$ in a data-informed region of the latent space. This may improve convergence and stability by anchoring the support where node distributions actually live.

In both cases, the support atoms $\{c_j\}$ are refined throughout training via back-propagation.

(3) Per-graph barycenter weights. For graph G we solve the entropic Sinkhorn barycenter problem

$$w^{(G)} = \arg \min_{w \in \Delta^{m-1}} \sum_{v \in V_G} \lambda_v W_2^2\Big(\mu_v, \sum_{j=1}^m w_j \delta_{c_j}\Big),$$

running T differentiable Sinkhorn iterations (λ_v uniform or learned attention).

(4) Fixed-length graph embedding. Given the shared atoms C and graph-specific weights $w^{(G)}$ we build a vector $g^{(G)}$ in **one of two explicit ways**:

- **Weighted mean**

$$g^{(G)} \;=\; \sum_{j=1}^m w_j^{(G)}\, c_j \;\in\; \mathbb{R}^d.$$

  Each $c_j \in \mathbb{R}^d$, so the convex combination preserves dimension d.

- **Flattened atoms**

$$g^{(G)} \;=\; \big[\, w_1^{(G)} c_1 \,\|\, w_2^{(G)} c_2 \,\|\, \dots \,\|\, w_m^{(G)} c_m \big] \;\in\; \mathbb{R}^{md},$$

  where "$\|$" denotes concatenation. Each block contributes d numbers and we have m blocks, giving total length $m\,d$.

(5) Classifier and loss. An MLP maps $g^{(G)}$ to logits, cross-entropy gradients flow through the classifier, Sinkhorn solver, node encoder, and code-book atoms.

2.3 Novelty

- **Distributional nodes:** Unlike point-based GNNs, each node is a stochastic object, preserving uncertainty and multi-modes potential hidden structure.
- **Global barycentric pooling:** We pool graphs through a *shared-support* Wasserstein barycenter, ensuring a constant output size while respecting optimal-transport geometry.
- **End-to-end learnability:** Code-book atoms and barycenter weights are optimised jointly with the GNN via differentiable Sinkhorn—prior OT graph work either fixes supports or applies OT only at inference time.

3 EXPERIMENTAL PLAN

We outline the datasets, baselines, and evaluation protocol that will demonstrate the advantages of BaryGNN.

Datasets

- **TU-Datasets:** small-scale graphs like MUTAG, PROTEINS, IMDB-BINARY [Morris et al. 2020] for fast ablations.
- **OGBG-MolHIV:** medium-scale molecular property dataset with heavy class imbalance, evaluated by ROC-AUC [Hu et al. 2020].

Baselines / Comparisons

- **DiffPool** [Ying et al. 2018] and **SAGPool** [Lee et al. 2019].
- **Wasserstein Embedding for Graph Learning (WEGL)** [Kolouri et al. 2021].
- **Template-based Fused Gromov-Wasserstein GNN (TFGW)** [Vincent-Cuaz et al. 2022].
- **ROT-Pooling (ROT-GNN):** the Regularized Optimal Transport pooling layer of Xu and Cheng [2023], which generalizes many pooling schemes by solving a learnable ROT problem.
- **Comprehensive Graph Pooling Benchmark (GPB):** we will also report BaryGNN inside the evaluation harness of Wang et al. [Wang et al. 2024], which compares 17 pooling methods across 28 datasets.

Metrics

- **Balanced datasets:** Accuracy and macro-F_1.
- **Imbalanced (MolHIV):** ROC-AUC (OGB protocol) [Hu et al. 2020].

BaryGNN Variants and Ablations

- **Read-out type:** *Weighted mean* ($\mathbb{R}^d$) vs. *Flattened atoms* ($\mathbb{R}^{md}$).
- **Support size:** $m \in \{8, 16, 32, 64\}$ to study capacity/speed trade-offs.
- **Initialization strategy:** random Gaussian vs. k-means++ warm-start (Sec. 2.2).
- **Sinkhorn regimes:** entropy $\varepsilon \in \{0.01, 0.05, 0.1\}$ and iterations $T \in \{10, 20\}$ to test robustness of the OT solver.

Protocol. All models are implemented in PyTorch Geometric. We will adopt a *single message-passing backbone* (e.g., GIN-EPS or GraphSAGE) chosen during an initial pilot study. This backbone will remain identical for every pooling variant and dataset, while all parameters are trained *fully* on each dataset. Training runs for at most 300 epochs with early stopping (20-epoch patience). Learning-rate, weight-decay, and dropout are selected from the standard hyper-parameter grids recommended for each benchmark using the validation split—no bespoke tuning per model. Every experiment is repeated with **three** random seeds. We report mean ± standard deviation. All code and configuration files will be released for full reproducibility.

REFERENCES

William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Neural Information Processing Systems conference (NIPS)*.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open Graph Benchmark: Datasets for Machine Learning on Graphs. In *Conference on Neural Information Processing Systems (NeurIPS)*.

Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

Soheil Kolouri, Navid Naderializadeh, Gustavo K. Rohde, and Heiko Hoffmann. 2021. Wasserstein Embedding for Graph Learning. In *International Conference on Learning Representations (ICLR)*. arXiv:2006.09430

Junhyun Lee, Inyeop Lee, and Jaewoo Kang. 2019. Self-Attention Graph Pooling. In *International Conference on Machine Learning (ICML)*.

Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. 2020. TUDataset: A Collection of Benchmark Datasets for Learning with Graphs. In *ICML Workshop on Graph Representation Learning and Beyond (GRL+)*. arXiv:2007.08663

Titouan Vayer, Laetita Chapel, Rémi Flamary, Romain Tavenard, and Nicolas Courty. 2020. Fused Gromov-Wasserstein distance for structured objects. *Algorithms* (2020).

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lió, and Yoshua Bengio. 2018. Graph Attention Networks. In *International Conference on Learning Representations (ICLR)*.

Cédric Vincent-Cuaz, Rémi Flamary, Marco Corneli, Titouan Vayer, and Nicolas Courty. 2022. Template based Graph Neural Network with Optimal Transport Distances. In *Conference on Neural Information Processing Systems (NeurIPS)*. arXiv:2205.15733

Pengyun Wang, Junyu Luo, Yanxin Shen, Ming Zhang, Siyu Heng, and Xiao Luo. 2024. A Comprehensive Graph Pooling Benchmark: Effectiveness, Robustness and Generalizability. *arXiv preprint arXiv:2406.09031* (2024).

Hongteng Xu and Minjie Cheng. 2023. Regularized Optimal Transport Layers for Generalized Global Pooling Operations. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2023), 1–18. https://doi.org/10.1109/TPAMI.2023.3314661 arXiv:2212.06339

Hongteng Xu, Dixin Luo, Hongyuan Zha, and Lawrence Carin. 2019b. Gromov–Wasserstein Learning for Graph Matching and Node Embedding. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*. 6932–6941. arXiv:1901.06003

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019a. How Powerful are Graph Neural Networks?. In *International Conference on Learning Representations (ICLR)*.

Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. 2018. Hierarchical Graph Representation Learning with Differentiable Pooling. In *Advances in Neural Information Processing Systems (NeurIPS)*.