Evaluating Ensembling Approaches for Node Class Prediction

Christopher Elchik, Harsh Manoj More, Teague McCracken

December 2025

1 Introduction

The node classification task on the ogbn-products dataset from the Open Graph Benchmark (OGB) is a famous benchmarking task used to compare novel approaches in machine learning on graphs. In this dataset, each node represents an Amazon product, and edges indicate that two products are frequently bought together. The objective of the node classification task is to predict the product category (47 classes) for each product based on its own features and neighborhood context. Each node is described by a 100-dimensional feature vector obtained from product word embeddings extracted from text descriptions. The graph is large and sparse, containing 2.4 million nodes and 61.9 million undirected edges, which makes scalability and memory efficiency key challenges.

The ogbn-products graph exhibits strong homophily, which means that neighboring nodes tend to belong to the same product category. This property makes message-passing GNNs, such as GCNs and GraphSAGE, particularly well-suited to this dataset, as they assume homophily and exploit this during message passing. In recent years, there have been several alternative models with high performance on the OGBN leaderboard with diverse architectures, but no ensembling methods have been submitted to the leaderboard. **This leads us to question the extent to which existing model predictions overlap and if a scheme can be devised to take advantage of each model's point of view or bias to achieve a higher test accuracy.** In this course project, we explore the agreement between different model architectures' predictions and test multiple simple ensemble schemes to evaluate if an ensemble prediction can achieve an accuracy greater than any individual model's accuracy for the ogbn-products task.

2 Related Works

Graph representation learning has witnessed significant advances in recent years, with methods ranging from simple feature-based models to complex neural architectures that exploit graph topology. A notable line of work demonstrates that shallow predictors, when combined with graph-based post-processing, can achieve competitive performance. Huang et al. [1] introduced the Correct and Smooth (C&S) framework, which refines the outputs of base models such as linear classifiers or MLPs using two post-processing steps: residual error propagation and label smoothing across the graph. The study shows that such approaches can match or even outperform sophisticated Graph Neural Networks (GNNs) on benchmark datasets, including OGBN-Products, while using orders of magnitude fewer parameters and computational resources. Similarly, GraphHop [2] leverages iterative label propagation with feature updates to improve semi-supervised node classification, illustrating the efficacy of combining label structure with feature-based models.

While post-processing methods focus on prediction refinement, other works target scalability and efficiency through feature precomputation and propagation. GAMLP (Graph Attention Multi-Layer Perceptron) [3] decouples feature propagation from model learning by precomputing multi-hop propagated features and applying a learned attention mechanism to combine them. This approach reduces training complexity while preserving performance on large-scale graphs. Extensions of GAMLP incorporate pseudo-labeling and consistency regularization techniques, such as Reliable Label Utilization (RLU) and Self-Consistency Regularization (SCR) [4], to enhance supervision and generalization, demonstrating strong performance on semi-supervised node classification tasks.

Ensembling multiple models has also emerged as an effective strategy to improve robustness and predictive performance in graph learning. Approaches such as E2GNN [5] and Boosting-GNN [6] illustrate that combining predictions from diverse base learners, whether via explicit ensembling or iterative boosting, mitigates weaknesses of individual models and leverages complementary strengths. In parallel, Luo

et al. [7] revisit classic GNNs, showing that with careful hyperparameter tuning (e.g., normalization, dropout, residual connections, and network depth), traditional models like GCN, GAT, and GraphSAGE can match or surpass modern Graph Transformer architectures on multiple benchmarks. This finding highlights that even simple GNNs can provide strong baselines for ensemble methods.

Our ensemble approach integrates these strands of research by combining six complementary models: shallow predictors with C&S post-processing (Plain Linear, Linear, and MLP), scalable GAMLP-based models enhanced with RLU, SCR, and C&S, and a classic GCN. By unifying feature-based, propagation-based, regularized, and deep message-passing models, the ensemble leverages diverse learning biases to improve robustness and predictive accuracy on the large-scale OGBN-Products dataset. This design reflects a convergence of insights from prior work, demonstrating how post-processing, pseudo-labeling, consistency regularization, and careful model tuning can be synergistically combined in a single framework.

3 Methodology

3.1 Model Selection

For the experimental purpose we tried our ensemble approach on the following set of 6 different models:

1. Plain Linear + C&S
2. Linear + C&S
3. MLP + C&S
4. GAMLP+RLU+SCR+C&S
5. GAMLP+RLU+SCR
6. GCN

Here most of the models are using C&S, the post-processing method, and some models are using SCR and RLU Training-time regularization methods.

1. **C&S (Correct and Smooth)**

   Correct step (residual propagation): It starts with a base predictor (linear model, MLP, etc.). It makes errors on labeled nodes, those residuals encode where the model's feature-based reasoning fails. Since errors often correlate along edges, we propagate residuals across the graph to adjust predictions. Mathematically, residuals are diffused using powers of a normalized adjacency matrix or Laplacian operator. The intuition is that if the predictor underestimated a class region, nodes connected to that region should also receive a correction.

   Smooth step (label-probability propagation): After corrections, predictions are refined by smoothing over graph structure, i.e. nodes connected through edges tend to share labels. This runs a classical graph diffusion over the corrected soft labels (e.g., repeated multiplication by a diffusion operator or solving a convex Laplacian smoothing system). The final prediction blends (i) feature-driven signal from the base model and (ii) structural consistency from the graph, all without retraining the base model.

2. **RLU (Robust Label Update)**

   Update step (pseudo-label expansion with selection): RLU extends semi-supervised learning by expanding supervision beyond the original labeled nodes, but only when confident. A base predictor generates soft predictions on unlabeled nodes. Nodes with high confidence are treated as pseudo-label sources. These pseudo-labels enlarge the training set and allow information to flow deeper into the graph.

Learning step (joint self-training with graph structure): The key is robustness, pseudo-labels are not blindly trusted. RLU selectively filters them (e.g., confidence thresholds, teacher-student consistency, soft weighting) before feeding them back into a learning loop. The model is retrained iteratively where the true labels anchor reliability, and pseudo-labels expand learning signal. The math typically couples cross-entropy with regularizers ensuring consistency over the graph (similar in spirit to Laplacian learning). Over iterations, labels are updated, smoothing graph structure and reinforcing confident regions while suppressing noise, effectively bootstrapping structure-aware supervision.

3. **SCR (Selective Correction and Refinement)**

   Selective correction step (targeted residual propagation): SCR generalizes C&S by recognizing that not all nodes should be corrected equally. Instead of uniformly diffusing residual error, it selectively applies correction only to regions where mistakes matter, e.g. nodes with high uncertainty, inconsistency with neighbors, or belonging to ambiguous graph neighborhoods. Mathematically, this looks like residual propagation multiplied by attention or gating weights that select where corrections are allowed to influence. This guards against over-correction and error amplification.

   Refinement step (adaptive smoothing / consistency enforcement): After selective correction, predictions undergo refinement similar to smoothing, but again selectively. Rather than globally enforce homophily, SCR selectively refines where graph consistency is meaningful. This yields a constrained Laplacian smoothing or weighted proximal optimization where smoothness penalties vary across the graph. Conceptually, it prevents flattening across heterophilous regions (where neighbors have different labels), while maximizing label consistency within coherent communities. Thus, SCR preserves structural nuance that classical C&S cannot fully capture.

Following base models were used:

1. **Plain Linear:** The Plain Linear + C&S model begins with the simplest possible base predictor: a linear classifier trained only on node features (logistic regression), without looking at the graph connectivity. Formally, this base model computes logits as

$$Z = XW \tag{1}$$

   where X is the node feature matrix and W is a learned weight matrix optimized using labeled nodes. This stage yields initial soft predictions over classes. The model does not incorporate adjacency information during learning, making it computationally cheap and easy to optimize.

2. **Linear Model:** The Linear + C&S variant is similar in spirit but incorporates a slightly richer linear base model than the plain linear version. Instead of a single weight matrix operating directly on features, one may include regularization, normalization, or additional linear layers, effectively improving expressiveness while still avoiding nonlinear transformations. Agriculturally, its inference can be seen as

$$Z = flinear(X), \tag{2}$$

   where flinear is a deeper or regularized linear mapping, such as ridge-penalized logistic regression or feature pre-projection before classification.

3. **MLP:** The MLP + C&S model replaces the base linear classifier with a non-linear multilayer perceptron operating on features. Instead of Z = XW, the MLP computes a transformation like

$$H_1 = \sigma(XW_1), Z = H_1W_2, \tag{3}$$

   where $\sigma(.)$ is a nonlinear activation function. This allows hierarchical feature interaction and representation learning even though the graph is still ignored during training. Compared to linear models, an MLP learns more complex decision boundaries using hidden units, offering richer signal for graph diffusion.

Conceptually, MLP + C&S is similar to adding a feature extractor before harmonic smoothing. It preserves computational efficiency relative to GNNs (because message passing is not learned), yet improves representation capacity relative to linear models.

4. **GAMLP (Graph Attention Multi-Layer Perceptron):** GAMLP is a model designed to leverage multi-hop graph structure without expensive message passing, avoiding the full complexity of GNN propagation. Instead, it augments raw node features with precomputed hop-wise adjacency aggregations

$$AX, AX, \ldots, AkX, \tag{4}$$

where A is the normalized adjacency operator. These multiple neighborhood signals are fused using attention weights learned over hops, and fed into an MLP classifier. Mathematically, GAMLP learns

$$h = \sum_i \alpha_i f(A_i X) \tag{5}$$

where α_i are trainable hop attention coefficients, and f is an MLP feature projection. Thus, GAMLP approximates GNN expressiveness by decoupling propagation (preprocessing) from learning (attention + MLP), leading to scalable training.

5. **GCN (Graph Convolutional Network):** A Graph Convolutional Network (GCN) is a fundamental model for learning from graph-structured data. It updates a node's representation by combining its features with information from neighboring nodes, allowing the model to capture structural relationships such as homophily and label similarity. This makes GCNs effective for semi-supervised node classification, where label information can naturally propagate through the graph.

   Mathematically, a GCN layer applies normalized neighborhood aggregation followed by a linear transformation and nonlinearity, expressed as

$$H^{l+1} = \sigma(D^{-1/2}\hat{A}D^{-1/2}H^l W^l), \tag{6}$$

   where $\hat{A} = A + I$ includes self-loops, D is the degree matrix, H^l the input embeddings, W^l the learnable weights, and σ an activation like ReLU. This normalization ensures balanced message passing and stable training. With appropriate tuning, GCNs remain strong baselines and can rival more complex graph models.

The above pre-processing method, training-time regularization method and base models were used for the experiments.

3.2 Ensembling

Traditional ensemble models for classification tasks have typically used voting systems to determine the final prediction. These voting systems have usually been either a basic majority vote by all the models, or the highest average probability across all model outputs.

In this project, there were 47 categories for the classification task in the ogbn-products dataset. All 6 models were ran, and the final output tensors were collected for each, containing the probability distribution across each of the 47 categories for each node in the dataset. For aggregating the votes across the 6 models (for each node), 9 unique voting methods were tested, using these output tensors. All ties were broken with random choice.

1. **Basic Majority Voting**

   Each model votes one time for the category with the highest probability in its respective output tensor. The category with the highest number of votes "wins" and is chosen as the final ensemble prediction.

2. **Ranked Choice Voting**

   Each model gets three votes, where the first-choice vote is worth 3 points, second-choice vote is worth 2 points, and third-choice vote is worth 1 point. These choices correspond to the top 3 categories with highest probabilities in each model's output tensor. The category with the most points is chosen as the final ensemble prediction.

3. **Probability Sum Voting**

   For each of the 47 categories, the corresponding probability values across each of the 6 output tensors are summed. The category with the highest value is chosen as the final ensemble prediction.

4. **Proportional Top-5 Voting**

   Each model has 5 total points that it will only distribute across its top 5 predicted classes. The points are split in proportion to the class probabilities. Models that are more confident in their predictions will allocate more of the 5 points to their top categories, thus making the confident votes more impactful. All point allocations are summed, and the category with the most points is chosen as the final prediction.

5. **Performance-Weighted Voting**

   Each model's probability vector gets multiplied by a weight proportional to its test accuracy, so the higher performing models get more influence. These weighted probabilities are summed across all models, where the class with the highest value is chosen as the final prediction.

6. **Confidence-Weighted Voting**

   Each model's probability vector gets multiplied by its maximum probability value, and these confidence-weighted probabilities are summed across all the models. The more confident models have more influence in this case. The class with the highest value is chosen as the final prediction.

7. **Geometric Mean Voting**

   For each of the 47 categories, the corresponding probability values across each of the 6 output tensors are multiplied, and the class with the highest product is chosen as the final prediction. This method emphasizes agreement across all the models.

8. **Exponential Weighted Voting**

   Each model's class probabilites are raised to a certain power before summing them, which amplifies the more confident predictions and suppresses the less confident ones. The ensemble chooses the class with the highest sum of exponentiated probabilities. We tested exponents from 1.5-3.0, and 3.0 was the only one that performed well, so we're omitting the rest from this paper.

9. **Hybrid Weighted Voting**

   This method combines methods 5 and 6, taking the probability vector for each model and multiplying it by both the maximum probability value and its corresponding performance weight. The weighted probabilities are then summed, and the class with the highest value is chosen as the final prediction.

4 Experiment

4.1 Experimental Design

The best individual model in the ensemble has a test accuracy of 85.19% on the ogbn-products dataset, and the average performance across all the models is 85.38%. The goal of ensembling this model alongside lower-performing models is to see if aggregating predictions across a group will perform better than this individual model. All 6 models were pre-ran, and the output tensors were collected separately. Using jupyter notebooks, all 9 of the voting methods were ran using these 6 output tensors, specifically on the nodes in the test split of the ogbn-products dataset. The performance of each voting method was then recorded.

Additionally, an agreement analysis was conducted across the 6 models to identify whether different models make the same mistakes, as ensembles work best when comprised of a diverse set of models. This agreement analysis used the same output tensors for each model as the voting method analysis.

Generally, we found the GCN to be the least agreeable and lowest performing (individually) across all the models, so we conducted another voting analysis that excludes the GCN from the ensemble. This was added to examine whether or not the GCN brings down the ensemble performance or improves it, despite being the worst performing model.

4.2 Results

Table 1 contains the results of each voting method.

Method	Test Accuracy	Imp. over Best	Imp. over Avg
Basic Majority Voting	0.8442	-0.77%	1.02%
Ranked Choice Voting	0.8489	-0.30%	1.49%
Probability Sum	0.8524	0.05%	1.84%
Proportional Top 5	0.8538	0.19%	1.98%
Performance Weighted Voting	0.8524	0.05%	1.84%
Confidence Weighted Voting	0.8512	-0.07%	1.72%
Geometric Mean Voting	0.8539	0.20%	1.99%
Exponential Weighted Voting	0.8508	-0.11%	1.68%
Hybrid Weighted Voting	0.8512	-0.07%	1.72%

Table 1: Comparison of Ensemble Voting Methods

From these results, the following observations were drawn:

- Four of the nine voting methods for ensembling resulted in improved performance over the best model: Basic Voting, Probability Sum Voting, Proportional Top-5 Voting, and Geometric Mean Voting.
- Geometric Mean Voting and Proportional Top 5 were the best, lifting the ensemble to 0.19% and 0.20% above the best individual model.
- Every single voting method resulted in a better performance than the average performance of the individual models.
- Voting methods that prioritized specific attributes (i.e., performance, confidence, agreement) saw better performances than voting methods that treated all models equally.

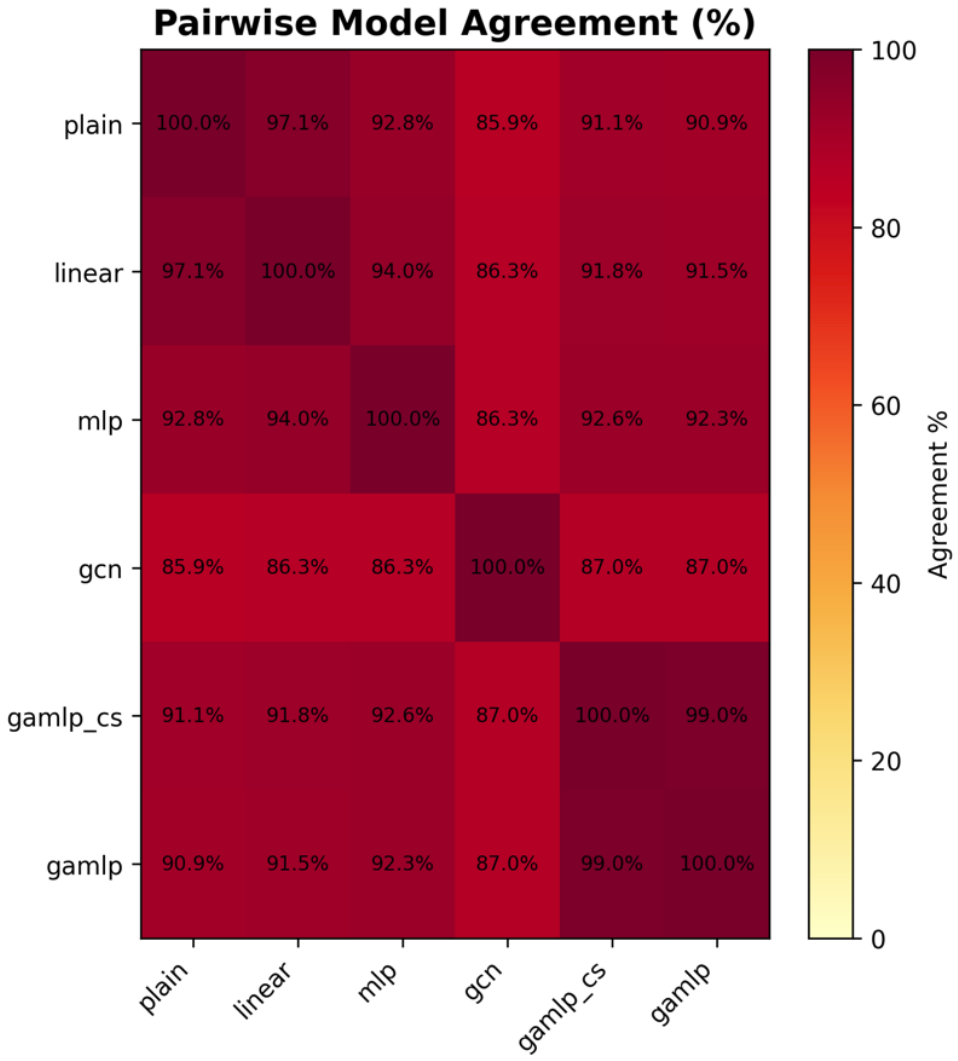

Figure 1: Pairwise Model Agreement Analysis

The following observations were drawn regarding model agreements:

- The GCN is the least agreeable model out of the entire ensemble, only matching the predictions of other models 85.9%-87.0% of the time.
- The two GAMLP-based models make the same predictions 99% of the time, showing the minimal difference C&S makes in this specific circumstance. This may suggest that the GAMLP models have very heavy influence over the voting systems.
- The two linear models (linear + C&S, plain linear + C&S) make the same predictions 97.1% of the time, showing another dominant pair in the ensemble.
- Agreement percentages are always higher than accuracy percentages, showing how these models generally make similar mistakes.

Since the GCN is the least agreeable model as well as the lowest performing, voting analysis was conducted for an ensemble that excludes only the GCN. The same voting methods were used.

Method	Test Accuracy (No GCN)	Test Accuracy (With GCN)
Basic Majority Voting	84.57%	84.42%
Ranked Choice Voting	84.85%	84.89%
Probability Sum	85.18%	85.24%
Proportional Top 5	85.24%	85.38%
Performance Weighted Voting	85.18%	85.24%
Confidence Weighted Voting	85.10%	85.12%
Geometric Mean Voting	85.18%	85.39%
Exponential Weighted Voting	85.08%	85.08%
Hybrid Weighted Voting	85.10%	85.12%

Table 2: Comparison of Ensemble Voting Methods With and Without GCN in the Ensemble

The following observations were drawn:

- Without the GCN, the only voting method that saw an improvement over the best model (85.19%) was the Proportional Top 5 method at 85.24%. Geometric Mean and Probability Sum were both close at 85.18%.
- The Basic Majority Vote was the only voting method that did better without the GCN involved, with a 0.15% difference. Exponential Weighted Voting saw no difference.
- All other voting methods generally saw improvements when the GCN was included, showing that ensemble methods can benefit from less agreeable models, even if they're lower performing.

5 Conclusion and Future Directions

The best ensemble approach achieved a higher test accuracy than any individual model on the ogbn-leaderboard, which did not rely on external data, where it achieved 85.39% test accuracy, using geometric mean voting, and the highest performing single model achieved 85.19% accuracy. In this project, we tested various voting methods for creating an ensemble prediction. Of the nine methods tested, four outperformed the best model, where geometric mean voting and proportional top five voting were the top two methods. The high performance of these top two methods can be partially explained by their ability to leverage confidence attributes of each prediction in the ensemble, where the continuous probabilities assigned to each class were leveraged to generate the final ensemble. Through investigating the pairwise model agreement and ensemble performance with and without the least consistent model (GCN model), we identify that adding models with diverse prediction sets explains the high performance of our ensemble approaches. This is to be expected as it is the main motivation behind ensembling approaches; we demonstrate this phenomenon in our analysis. Interestingly, the GCN did not have the highest performance, but it still contributed to an improvement in most of the voting methods, demonstrating that prediction diversity is more important than exceptional performance for improving ensemble methods.

Future work on this topic could explore unsupervised machine learning based approaches to deriving a final prediction from each model's individual prediction sets. Additionally, future research into ensembling on graphic predictions should explore model bias by node properties such as degree, eigen centrality, etc., as well as node feature classes. This exploration would enable a characterization of each model's bias and help to explain why ensemble approaches are able to achieve a higher performance than a high-performing individual model.

6 Code Availability

See the code at: https://github.com/teegman123/GNN_project.git

References

1. Huang, Q., He, H., Singh, A., Lim, S.-N., Benson, A.R. Combining Label Propagation and Simple Models Outperforms Graph Neural Networks, 2020.

2. Yang, J., Liu, D., et al. GraphHop: An Enhanced Label Propagation Method for Node Classification, 2021.

3. Zhang, Y., et al. GAMLP: Graph Attention Multi-Layer Perceptron for Large Graphs, 2021.

4. Zhang, C., He, Y., et al. SCR: Training Graph Neural Networks with Consistency Regularization, 2021.

5. Zhou, H., et al. E2GNN: Efficient Graph Neural Network Ensembles for Semi-Supervised Classification, 2022.

6. Chen, X., et al. Boosting-GNN: Boosting Algorithm for Graph Networks on Imbalanced Node Classification, 2021.

7 Luo, Y., Shi, L., Wu, X.-M. Classic GNNs are Strong Baselines: Reassessing GNNs for Node Classification, 2021.