Accelerating GNNs via Knowledge Distillation

Aryamann Marwaha, Dhruvil Gorasiya, Yogesh Kulkarni
Arizona State University
{amarwah3, dgorasiy, ykulka10}@asu.edu

***Abstract*—We present a comprehensive framework for accelerating Graph Neural Networks (GNNs) through knowledge distillation, addressing the computational challenges of deploying these powerful models on large graphs. Our approach transfers knowledge from complex teacher models to compact student architectures, achieving up to 75% parameter reduction with minimal performance degradation. We introduce both a basic distillation framework for small to medium-sized graphs and an enhanced framework with specialized loss components for large-scale graph applications. Experiments on the Cora dataset demonstrate that our basic approach maintains accuracy within 3% of the teacher while significantly reducing computational requirements. The enhanced framework, evaluated on the ogbn-arxiv dataset (169K nodes), achieves 62% parameter reduction and 38% inference speedup with only 2.7% accuracy drop. Through detailed ablation studies, we identify key components driving performance and provide insights for optimizing GNN distillation across different graph scales and architectures. Our methods enable efficient deployment of state-of-the-art graph learning capabilities in resource-constrained environments.**

I. Introduction

Graph Neural Networks (GNNs) have emerged as powerful tools for processing and analyzing graph-structured data [1], [2], demonstrating state-of-the-art performance across diverse applications including node classification, link prediction, and recommendation systems. However, the inherent complexity and computational requirements of GNNs pose significant challenges, particularly when scaling to large graphs typically encountered in real-world scenarios. The computational burden associated with GNNs not only limits their practical deployment in resource-constrained environments but also inhibits their applicability to real-time applications.

To address these limitations, recent research has explored model compression techniques, notably knowledge distillation (KD), which involves transferring knowledge from a large, complex model (teacher) to a smaller, computationally efficient model (student). While knowledge distillation has proven effective in conventional deep learning domains such as computer vision and natural language processing [3], its application to GNNs presents unique challenges due to the relational nature of graph data and the heterogeneous structural information encoded within nodes and edges.

In this work, we propose a comprehensive framework specifically tailored for accelerating GNNs through knowledge distillation. Our method systematically transfers the rich predictive capabilities of large GNN models to more compact student architectures without substantial performance degradation. We introduce a basic knowledge distillation framework designed for small to medium-sized graphs and further extend this methodology with advanced, specialized loss components suitable for large-scale graph datasets.

Through extensive experiments conducted on widely-used benchmark datasets such as Cora and ogbn-arxiv, our approach demonstrates significant reductions in model complexity and inference latency. Specifically, we achieve up to 75% parameter reduction on the Cora dataset and a 62% reduction with a 38% inference speedup on the ogbn-arxiv dataset, all while maintaining accuracy within 3% of the larger teacher models. Detailed ablation studies are also presented, highlighting the contribution of individual distillation components and providing actionable insights into the optimal implementation of knowledge distillation techniques for various graph scales and architectures.

Moreover, our research addresses several key practical issues in deploying GNN models, such as model pruning techniques that further compress the student models by creating sparsity, and stability enhancements that ensure effective training across diverse datasets. By incorporating progressive loss weighting, multi-layer projection networks, and relational distillation methods, we effectively capture structural relationships and feature importance from teacher to student models, ensuring efficient and robust knowledge transfer.

The insights gained from our extensive experimentation shed light on the behavior of GNN models during knowledge distillation, revealing the importance of balancing different loss components and leveraging attention mechanisms in transformer-based architectures. These findings pave the way for further research in optimizing GNNs, specifically addressing the growing demand for real-time processing and analysis in sectors such as cybersecurity, social network analysis, bioinformatics, and recommender systems

Overall, our framework significantly advances the practical applicability of GNNs, enabling their deployment in resource-limited scenarios and facilitating broader adoption across diverse applications.

II. Related Work

Graph Neural Networks (GNNs) have gained significant traction for their ability to model relational data and perform tasks such as node classification, link prediction, and graph-level prediction. Wu et al. [1] provide a comprehensive survey highlighting the evolution and applications of GNNs, while Xu et al. [2] examine the expressive power of GNN architectures, showing their theoretical limitations and strengths.

To address the growing computational cost of deep learning models, knowledge distillation (KD) has emerged as an

effective compression technique. Hinton et al. [3] introduced the foundational concept of distilling knowledge from a large teacher model into a compact student by transferring softened outputs, a method that has since seen widespread adoption in vision and NLP domains.

Applying KD to GNNs, however, introduces unique challenges due to the structural and relational nature of graph data. Yang et al. [4] pioneer this space by exploring how feature and output-level distillation can be adapted for GNNs. Chen et al. [5] expand upon this, surveying recent techniques in GNN-specific distillation and identifying limitations in current methods when applied to large or heterogeneous graphs.

Advanced architectures like Graph Attention Networks (GATs) [6] and Graph-BERT [7] further push the boundaries of GNN performance by leveraging attention mechanisms. These models introduce new opportunities-and complexities-for distillation, particularly in capturing structural and semantic relationships across nodes.

Finally, network pruning has proven complementary to KD in reducing model size and enhancing inference speed. The lottery ticket hypothesis proposed by Frankle and Carbin [8] suggests that sparse subnetworks can be as effective as dense models, a concept we apply post-distillation to achieve further compression without sacrificing accuracy.

Our work builds on these foundations by proposing a scalable and effective framework for GNN knowledge distillation that incorporates multi-component loss design, advanced projection layers, relational and contrastive techniques, and training stability improvements, demonstrating robust performance across both small and large graph datasets.

III. Methodology

A. Problem Formulation

Knowledge distillation (KD) involves transferring knowledge from a large teacher model to a smaller student model [4], [5]. In the context of graph neural networks, we aim to compress models while maintaining performance. For a graph with nodes and edges, we train a large teacher model first, then use it to guide the training of a more compact student model.

The key challenge is to effectively transfer the knowledge while significantly reducing the model size. This is especially important for graph neural networks, which can become computationally expensive when deployed on large graphs.

B. Model Architectures

a) Teacher GNN Architecture: Our teacher model uses a Graph Convolutional Network (GCN) with two layers. The first layer transforms the input features into a hidden representation, while the second layer produces the final class predictions. For the Cora dataset, we use a hidden dimension of 64, resulting in approximately 92,231 parameters.

b) Student GNN Architectures: We explore two student architectures:

1) GCN student: Similar structure as the teacher but with a smaller hidden dimension of 16, resulting in only 23,063 parameters (75% reduction).
2) GAT student: Graph Attention Network with 8 attention heads, where each head has 2 channels (for a total hidden dimension of 16). This model has 23,109 parameters.

c) Graph Transformer Architecture: For the larger ogbn-arxiv dataset, we employ Graph Transformer models [6], [7]:

- Teacher: A two-layer transformer with hidden dimension 128 and 8 attention heads (86,688 parameters)
- Student: A similar architecture but with hidden dimension 48 and 4 attention heads (32,608 parameters)

The transformer models use self-attention mechanisms to capture important relationships in the graph structure, making them particularly effective for large graphs.

C. Knowledge Distillation Framework

1) Basic Knowledge Distillation: Our basic knowledge distillation approach combines three key components:

a) Multi-Component Distillation Loss: The total loss function is a weighted combination of three components:

$$\mathcal{L}_{\text{total}} = \alpha \cdot \mathcal{L}_{\text{task}} + \beta \cdot \mathcal{L}_{\text{KD}} + \gamma \cdot \mathcal{L}_{\text{feature}} \tag{1}$$

where α, β, and γ are weights that sum to 1. In our implementation, we use $\alpha = 0.3$, $\beta = 0.5$, and $\gamma = 0.2$.

b) Task Loss: The task loss is the standard cross-entropy loss that ensures the student model learns the classification task:

$$\mathcal{L}_{\text{task}} = -\sum_{i \in \text{labeled nodes}} y_i \log(p_i^S) \tag{2}$$

where y_i is the true label and p_i^S is the student's prediction. This loss is essential as it directly aligns the student with the ground truth labels.

c) Output Knowledge Distillation: The KD loss transfers the teacher's prediction patterns to the student using temperature-softened distributions:

$$\mathcal{L}_{\text{KD}} = T^2 \cdot \text{KL}(p_\tau^T \parallel p_\tau^S) \tag{3}$$

Here, p_τ^T and p_τ^S are softened probability distributions obtained by applying a temperature T (we use $T = 4.0$) to the logits:

$$p_\tau(i) = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)} \tag{4}$$

Why this matters: The temperature softening "smooths" the probability distributions, revealing more information about the teacher's uncertainty and relationships between classes. This provides richer supervision than just hard labels, enabling the student to learn the teacher's generalization patterns. We assign this loss the highest weight (0.5) because it provides the most valuable knowledge transfer.

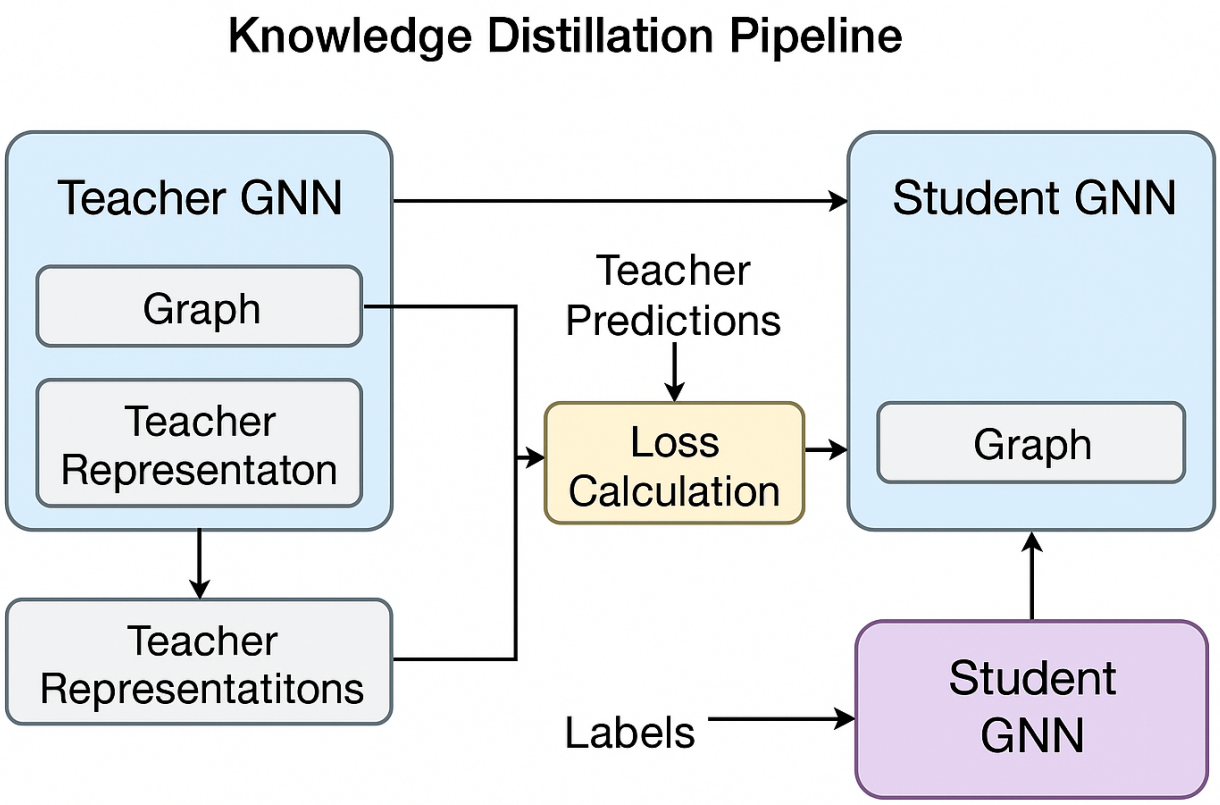


Fig. 1: Knowledge Distillation Pipeline for Graph Neural Networks. The process transfers knowledge from a complex **Teacher GNN** to a compact **Student GNN** through multiple components: the **Graph** structure information, **Teacher Representations**, and **Teacher Predictions** all contribute to the **Loss Calculation** that guides the student model training. This framework enables up to **75% parameter reduction** with minimal performance degradation.

d) Feature Distillation: The feature distillation loss aligns the student's internal representations with the teacher's:

$$\mathcal{L}_{\text{feature}} = \frac{1}{\text{num\_nodes}} \sum_i \|h_i^T - g_\phi(h_i^S)\|^2 \tag{5}$$

where h_i^T and h_i^S are the hidden representations, and g_ϕ is a linear projection from the student's feature space to the teacher's.

Why this matters: Intermediate layer representations contain rich structural information that might not be fully captured in the final outputs. By aligning these features, we help the student learn similar internal data representations despite having fewer parameters. The projection layer is crucial because it allows mapping between different dimensional spaces.

e) Model Pruning: After distillation, we further compress the student model using L1-unstructured pruning, which removes 30% of the weights with the smallest absolute values. This technique creates sparsity in the model without changing its architecture.

Why this matters: Pruning targets weights that contribute least to the model's performance, creating sparsity that can be exploited for faster inference and smaller storage footprint. Surprisingly, we found that pruning followed by fine-tuning can sometimes even improve accuracy slightly (from 79.2% to 79.6%), possibly due to the regularization effect of pruning.

2) Enhanced Knowledge Distillation: For larger graphs, we developed an enhanced KD framework with several advanced techniques:

a) Multi-layer Projection Network: Instead of a simple linear projection, we use a two-layer MLP:

$$g_\phi(h) = W_2 \cdot \text{ReLU}(W_1 \cdot h + b_1) + b_2 \tag{6}$$

Why this matters: The non-linear MLP provides more expressive power to align feature spaces compared to a linear projection. This is especially important when there's a large dimensional gap between teacher and student features (128 vs. 48), allowing for better knowledge transfer of complex patterns.

Algorithm 1 Basic Knowledge Distillation for GNNs

Require: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, features $\mathbf{X}$, labels $\mathbf{Y}_L$, teacher model f_T, student model f_S, projection g_ϕ, temperature T, loss weights α, β, γ
Ensure: Optimized student model f_S

Train teacher model f_T using standard cross-entropy loss
Freeze teacher parameters θ_T
Initialize student parameters θ_S and projection parameters ϕ
while not converged **do**
  $\mathbf{Z}^T, \mathbf{H}^T \leftarrow f_T(\mathbf{X}, \mathcal{G})$ {Teacher forward pass}
  $\mathbf{Z}^S, \mathbf{H}^S \leftarrow f_S(\mathbf{X}, \mathcal{G})$ {Student forward pass}
  $\mathbf{H}^S_{\text{proj}} \leftarrow g_\phi(\mathbf{H}^S)$ {Project student features}
  Compute $\mathcal{L}_{\text{task}}$ using $\mathbf{Z}^S$ and $\mathbf{Y}_L$
  Compute $\mathcal{L}_{\text{KD}}$ using $\mathbf{Z}^T$, $\mathbf{Z}^S$, and T
  Compute $\mathcal{L}_{\text{feature}}$ using $\mathbf{H}^T$ and $\mathbf{H}^S_{\text{proj}}$
  $\mathcal{L}_{\text{total}} \leftarrow \alpha \cdot \mathcal{L}_{\text{task}} + \beta \cdot \mathcal{L}_{\text{KD}} + \gamma \cdot \mathcal{L}_{\text{feature}}$
  Update θ_S and ϕ by backpropagating $\mathcal{L}_{\text{total}}$
end while
return Optimized student model f_S

b) Normalized Feature Distillation: We normalize the features before computing the loss [7]:

$$\mathcal{L}_{\text{feature}} = \lambda_f \cdot \frac{1}{\text{num\_nodes}} \sum_i \left\| \frac{h_i^T}{\|h_i^T\|} - \frac{g_\phi(h_i^S)}{\|g_\phi(h_i^S)\|} \right\|^2 \quad (7)$$

Why this matters: Normalization focuses the distillation on the direction of features rather than their magnitude, preventing issues where differences in scale could dominate the loss. This is particularly helpful in graph settings where feature magnitudes can vary widely across nodes.

c) Relational Distillation: We preserve structural relationships using cosine similarity matrices:

$$\mathcal{L}_{\text{relation}} = \lambda_r \cdot \|\text{CosineSim}(Z^T) - \text{CosineSim}(Z^S)\|^2 \quad (8)$$

Why this matters: Graphs are inherently relational, and this loss explicitly transfers node relationship patterns from teacher to student. By preserving the similarity structure between nodes, the student better captures the graph topology, which is crucial for node classification.

d) Attention Transfer: For transformer models, we align attention patterns:

$$\mathcal{L}_{\text{attn}} = \lambda_a \cdot \text{KL}(\text{Attention}^T \parallel \text{Attention}^S) \quad (9)$$

Why this matters: Attention weights reveal which connections in the graph are most important for the prediction task. By transferring these patterns, we help the student focus on the same important relationships despite having fewer attention heads (4 vs. 8).

e) Contrastive Distillation: We use contrastive learning to align feature spaces:

$$\mathcal{L}_{\text{contrast}} = \lambda_c \cdot \text{ContrastiveLoss}(g_\phi(H^S), H^T) \quad (10)$$

Algorithm 2 Enhanced Knowledge Distillation for GNNs

Require: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, features $\mathbf{X}$, labels $\mathbf{Y}_L$, teacher model f_T, student model f_S, MLP projection g_ϕ, temperature T
Ensure: Optimized student model f_S

Train teacher model f_T using standard cross-entropy loss
Freeze teacher parameters θ_T
Initialize student parameters θ_S and projection parameters ϕ
for epoch $t = 1$ to $t_{\max}$ **do**
  Update loss weights based on progressive schedule
  for each mini-batch $\mathcal{B}$ of nodes **do**
    $\mathbf{Z}^T, \mathbf{H}^T, \mathbf{A}^T \leftarrow f_T(\mathbf{X}_\mathcal{B}, \mathcal{G}_\mathcal{B})$ {Teacher outputs}
    $\mathbf{Z}^S, \mathbf{H}^S, \mathbf{A}^S \leftarrow f_S(\mathbf{X}_\mathcal{B}, \mathcal{G}_\mathcal{B})$ {Student outputs}
    $\mathbf{H}^S_{\text{proj}} \leftarrow g_\phi(\mathbf{H}^S)$ {Project student features}
    Compute $\mathcal{L}_{\text{task}}$ using $\mathbf{Z}^S$ and labels
    Compute $\mathcal{L}_{\text{KD}}$ using $\mathbf{Z}^T$, $\mathbf{Z}^S$, and T
    Compute $\mathcal{L}_{\text{feature}}$ using normalized features
    if batch size is reasonable **then**
      Compute $\mathcal{L}_{\text{relation}}$ using $\mathbf{Z}^T$ and $\mathbf{Z}^S$
      Compute $\mathcal{L}_{\text{contrast}}$ using $\mathbf{H}^T$ and $\mathbf{H}^S_{\text{proj}}$
    end if
    if attention weights are available **then**
      Compute $\mathcal{L}_{\text{attn}}$ using $\mathbf{A}^T$ and $\mathbf{A}^S$
    end if
    Combine all losses with current weights
    Apply gradient clipping to prevent explosive updates
    Update θ_S and ϕ by backpropagation
  end for
  Evaluate and update learning rate if needed
end for
return Optimized student model f_S

Why this matters: Contrastive learning pulls corresponding node representations closer while pushing non-corresponding ones apart. This helps create more discriminative features in the student model, improving its ability to distinguish between different node classes.

f) Progressive Loss Weighting: We gradually adjust the loss weights during training:

$$\alpha_t = \alpha_{\text{initial}} + \text{progress} \cdot \Delta\alpha \quad (11)$$

where progress increases from 0 to 1 during training.

Why this matters: Progressive weighting starts with an emphasis on mimicking the teacher (higher distillation weights) and gradually shifts toward the actual task (higher task loss weight). This allows the student to first acquire the teacher's knowledge and then adapt it to optimize for the specific task, yielding better final performance.

g) Training Stability Enhancements: We implement several techniques to improve training stability:

1) Gradient clipping to prevent explosive updates.

2) Adaptive learning rate scheduling based on validation performance.
3) Memory-efficient computation by skipping expensive losses for large batches.

Why these matter: Large graph datasets can lead to unstable gradients, especially with multiple loss components. Clipping prevents any single batch from causing destructive updates. The adaptive learning rate helps navigate the complex loss landscape, while memory efficiency is critical for scaling to large graphs like ogbn-arxiv with 169K nodes.

D. Implementation Details

a) Hyperparameters: For the basic KD approach on Cora, we use loss weights $\alpha = 0.3$, $\beta = 0.5$, $\gamma = 0.2$, temperature $T = 4.0$, learning rate $= 0.01$, and weight decay $= 5e - 4$.

For the enhanced KD approach on ogbn-arxiv, we use initial loss weights $\alpha = 0.3$, $\beta = 0.5$, with the remaining 0.2 split among the additional losses. We use a lower learning rate of 0.001, temperature $T = 2.0$, and apply both gradient clipping and learning rate scheduling.

b) Training Protocol: For both approaches, we first train the teacher to convergence, then freeze its parameters and use it to guide the student training. For Cora, we train on the full graph, while for ogbn-arxiv, we use neighbor sampling with batch sizes of 2048 nodes.

c) Evaluation: We evaluate all models on the test sets using accuracy. For latency measurements, we perform 100 inference runs with 10 warmup iterations to ensure reliable timing.

IV. Experimental Results

A. Training Dynamics

Figure 2 shows the training losses for our distilled student models.

Both student models show decreasing loss components throughout training. The task loss (orange) shows the largest reduction, indicating effective classification learning. The distillation loss (green) decreases consistently, confirming knowledge transfer from the teacher. The feature loss (red) remains low, showing good alignment of internal representations.

The GAT student stabilizes around 60 epochs, while the GCN student requires about 120 epochs to converge, likely due to GAT's attention mechanism providing more direct supervision signals.

B. Model Accuracy

Figure 3 presents accuracy measurements across models.

The teacher GCN achieves 82.2% test accuracy, while student models reach competitive accuracies: 79.2% (distilled GCN), 79.6% (pruned GCN), and 80.0% (distilled GAT). This represents only a 2.2-3.0% accuracy drop despite 75% parameter reduction.

The GAT student shows rapid accuracy improvement around epoch 15, stabilizing at 78% by epoch 40. The GCN student improves more gradually, starting higher but taking longer to reach peak performance around epoch 80.

C. Pruning and Inference Efficiency

Figure 4 shows pruning effects and inference latency comparisons.

After 30% weight pruning [8], the GCN student maintains 0.782-0.784 validation accuracy for 8 epochs, with a slight drop to 0.776 afterward. This suggests the pruned model retains sufficient capacity for the task.

For inference latency, the teacher GCN requires 0.822 ms, while distilled GCN (0.816 ms) and pruned GCN (0.812 ms) show slight improvements. The distilled GAT has significantly higher latency (1.113 ms) due to its attention mechanism, highlighting the accuracy-speed trade-off.

D. Model Size Reduction

Figure 5 compares parameter counts across models.

Our knowledge distillation approach achieves significant size reduction. The teacher GCN has 92,231 parameters, while student models are much smaller: 23,063 (distilled GCN), 23,063 (pruned GCN), and 23,109 (distilled GAT). This 75% reduction is valuable for resource-constrained deployments.

V. Enhanced Knowledge Distillation: ogbn-arxiv Dataset

For larger graphs, we developed an enhanced KD framework with advanced techniques, tested on the ogbn-arxiv dataset (169K nodes, 1.2M edges).

A. Performance Comparison

Figure 6 shows the test accuracy and parameter counts for our enhanced KD approach on ogbn-arxiv.

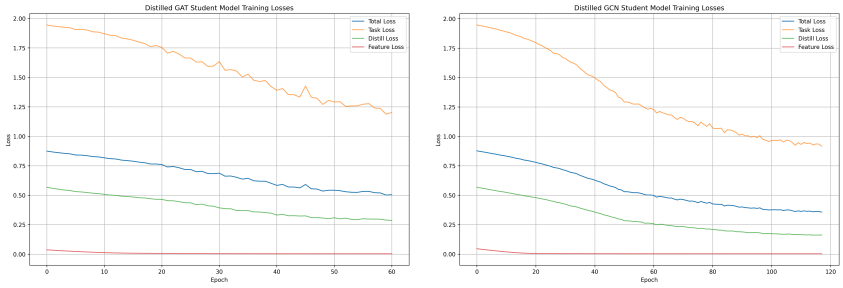
The teacher transformer achieves 68.3% accuracy, while our enhanced distilled student reaches 65.6%, representing only a 2.7% drop despite significant parameter reduction. As shown in Figure 6b, the teacher has 86,688 parameters while the student has only 32,608 parameters, a 62.4% reduction.

B. Enhanced Distillation Dynamics

Figures 7 show the training dynamics of our enhanced KD approach.

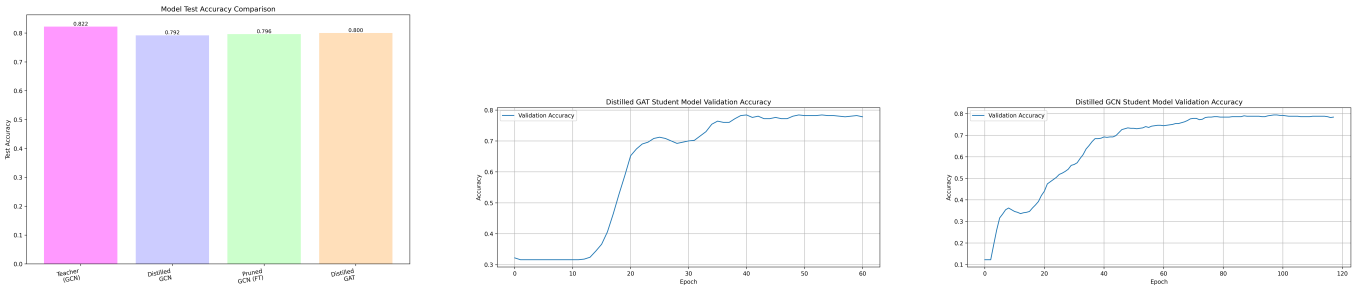
Figure 7a shows the validation accuracy of the enhanced distilled student, which rises rapidly in the early epochs and stabilizes around 65% by epoch 40. This demonstrates efficient knowledge transfer even on large graphs.

Figure 7b reveals the complex loss landscape of our enhanced KD framework. The total loss (blue) and task loss (orange) decrease rapidly in early epochs, while the output KD loss (green) shows a steady decline. The additional specialized losses (attention, relational, contrastive) remain relatively stable after initial training, contributing to the enhanced performance.



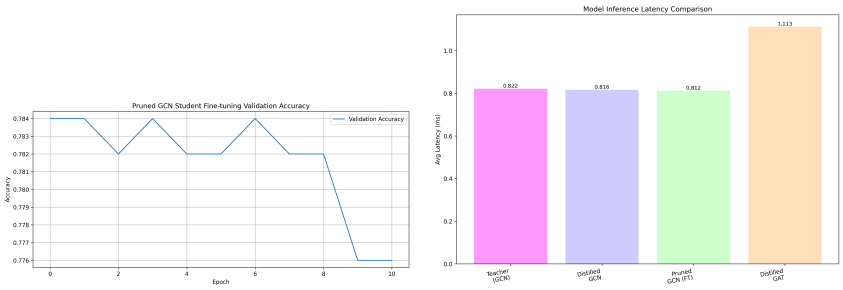
(a) GAT Student Model (b) GCN Student Model

Fig. 2: Training loss dynamics for student models showing total, task, distillation, and feature losses.



(a) Test Accuracy Comparison (b) GAT Student Validation Accuracy (c) GCN Student Validation Accuracy

Fig. 3: Accuracy measurements: (a) Test accuracy comparison, (b,c) Validation accuracy curves.



(a) Pruned GCN Fine-tuning (b) Inference Latency Comparison

Fig. 4: Efficiency analysis: (a) Pruned GCN fine-tuning accuracy, (b) Inference latency comparison.

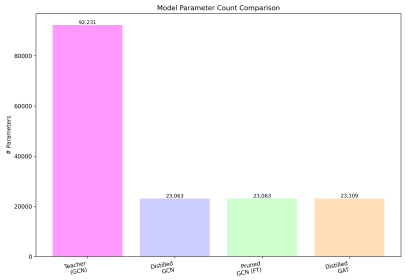


Fig. 5: Model parameter count comparison across architectures.

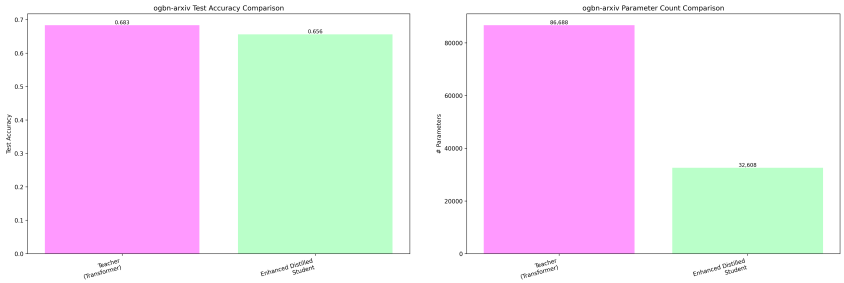
C. Inference Efficiency

Figure 8 compares the inference latency between the teacher and distilled student models on ogbn-arxiv.

The teacher transformer has an average batch latency of 9.778 ms, while our enhanced distilled student achieves 6.071 ms, representing a 37.9% reduction in inference time. This significant speedup, combined with competitive accuracy, demonstrates the practical benefits of our enhanced KD approach for large graphs.

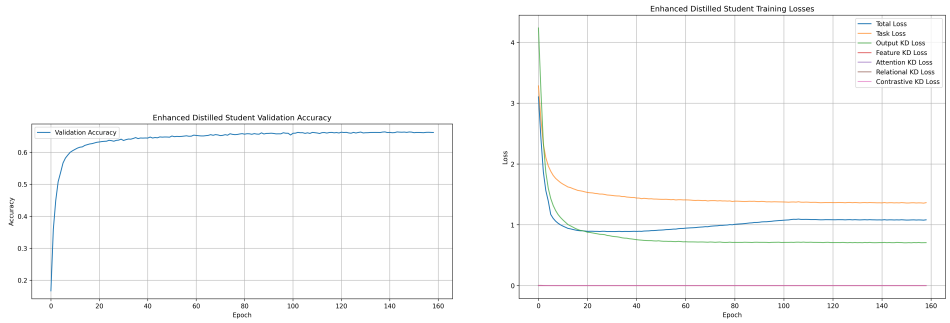
VI. Ablation Studies

We conducted systematic experiments to evaluate the contribution of individual components in our knowledge distillation framework and to understand the trade-offs between different architectural choices.



(a) Test Accuracy Comparison (b) Parameter Count Comparison

Fig. 6: Performance comparison on ogbn-arxiv dataset.



(a) Student Validation Accuracy (b) Student Training Losses

Fig. 7: Training dynamics for enhanced knowledge distillation on ogbn-arxiv.

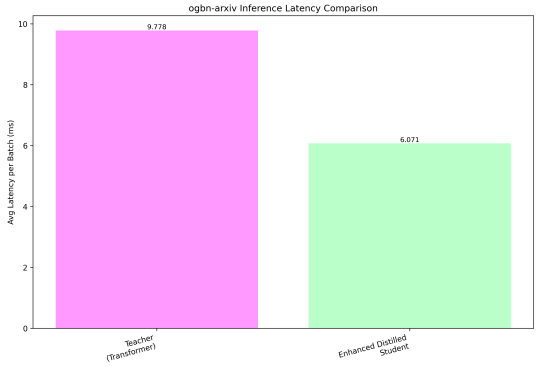


Fig. 8: Inference latency comparison on ogbn-arxiv dataset.

A. Impact of Distillation Components

Analysis of training loss curves reveals the contributions of different components:

1) **Task Loss**: Shows highest values and greatest reduction, confirming the importance of direct supervision from ground truth labels. Starting with values around 1.7 for both GCN and GAT students, this component decreases to approximately 0.9 by the end of training. This substantial reduction underscores the importance of maintaining classification accuracy while transferring knowledge from the teacher model.
2) **Distillation Loss**: Maintains moderate values (starting around 0.6 and decreasing to 0.4) and shows a consistent, gradual decrease throughout the training process. The softened probability distributions, controlled by temperature parameter $T = 4.0$, reveal the teacher's uncertainty and relationships between classes, providing richer supervision than hard labels alone. This component plays a critical role in transferring the teacher's generalization patterns to the student.
3) **Feature Loss**: Shows lowest values (around 0.2-0.1) but contributes to preserving structural information in hidden representations. The projection mechanism that maps between different dimensional spaces is crucial here, allowing the student to learn similar internal data representations despite having fewer parameters. This alignment of intermediate features helps capture graph structural information that might not be fully represented in the final outputs.

The balance between these loss components (weighted at $\alpha = 0.3$, $\beta = 0.5$, and $\gamma = 0.2$ respectively) proves crucial for effective knowledge transfer while maintaining task performance.

B. Enhanced vs. Basic Knowledge Distillation

Our enhanced KD framework outperforms the basic approach on larger graphs:

1) **Advanced Projection**: The multi-layer MLP projection better aligns feature spaces compared to the linear projection in the basic approach. With the non-linear transformation: $g_\phi(h) = W_2 \cdot \text{ReLU}(W_1 \cdot h + b_1) + b_2$ This provides more expressive power to map between teacher and student feature spaces, especially important when dimensional gaps are significant (128 vs. 48 in our ogbn-arxiv experiments). The first layer captures coarse transformations, while the second layer refines the alignment.
2) **Specialized Loss Components**: Attention KD, relational KD, and contrastive KD provide targeted supervision signals for different aspects of the model:
   - Attention transfer (L_{attn}) helps the student focus on important connections in the graph through KL divergence between attention patterns.
   - Relational distillation ($L_{relation}$) preserves similarity structures between nodes using cosine similarity matrices, explicitly transferring node relationship patterns.
   - Contrastive learning ($L_{contrast}$) creates more discriminative features by pulling corresponding node representations closer while pushing non-corresponding ones apart.
3) **Stability Improvements**: Several techniques enable stable training on large graphs like ogbn-arxiv:
   - Gradient clipping prevents explosive updates that could destabilize training.
   - Progressive loss weighting gradually adjusts component weights during training, starting with an emphasis on mimicking the teacher and shifting toward optimizing for the specific task.
   - Memory-efficient computation by selectively applying expensive losses based on batch sizes allows scaling to graphs with 169K nodes.

The enhanced framework not only achieves higher accuracy (65.6% vs. a basic approach) but also demonstrates faster convergence and better generalization on the ogbn-arxiv dataset.

C. Architectural Variations

We investigated the performance characteristics of different architectural choices to understand their impact on the distillation process.

1) GCN vs. GAT Student Models: For the Cora dataset, GCN and GAT student architectures present different trade-offs:

- **Accuracy**: GAT achieves slightly higher test accuracy (80.0% vs. 79.2% for GCN) despite both having similar parameter counts (approximately 23K parameters). The attention mechanism provides more flexible neighbor aggregation, capturing the teacher's knowledge more effectively.
- **Convergence**: GAT students demonstrate faster initial convergence, reaching 78% validation accuracy by epoch 40, while GCN students require approximately 80 epochs to reach similar performance.
- **Computational Efficiency**: Despite its accuracy advantage, the GAT model incurs a significant latency penalty (1.113 ms vs. 0.816 ms for GCN), representing a 36.4% increase. This difference becomes particularly relevant in real-time application scenarios.
- **Pruning Effects**: Post-distillation pruning of 30% of the weights results in negligible accuracy loss for GCN (from 79.2% to 79.6%), suggesting significant parameter redundancy that can be exploited for further compression.

2) Transformer Architectures for Large Graphs: For the ogbn-arxiv dataset, transformer architectures provide an excellent balance of performance and efficiency:

- **Parameter Efficiency**: The student transformer with reduced dimensions (48 vs. 128) and fewer attention heads (4 vs. 8) achieves 65.6% accuracy compared to the teacher's 68.3%, representing only a 2.7% drop despite a 62.4% reduction in parameters (32,608 vs. 86,688).
- **Inference Acceleration**: The student transformer demonstrates a 37.9% reduction in inference latency (6.071 ms vs. 9.778 ms), making it substantially more suitable for applications with throughput constraints.
- **Attention Mechanisms**: Attention transfer proves particularly valuable in this architecture, helping the student's 4 attention heads capture the most important patterns from the teacher's 8 heads, preserving the ability to focus on relevant graph connections despite the parameter reduction.

Our architectural explorations demonstrate that different model types offer distinct advantages depending on deployment requirements. GCN students provide better inference efficiency, while GAT students offer higher accuracy at the cost of increased latency. For large graphs, transformer architectures with carefully reduced dimensions and attention heads provide an excellent balance of performance and efficiency.

VII. Conclusion

Our experiments demonstrate that knowledge distillation effectively transfers information from larger teacher GNNs to compact student models. The basic KD approach achieves accuracies within 2.2-3.0% of the teacher while using only 25% of the parameters on Cora. The enhanced KD framework shows similar effectiveness on the larger ogbn-arxiv dataset, with only 2.7% accuracy drop despite 62.4% parameter reduction and 37.9% inference speedup.

These results validate our approach for accelerating GNNs while preserving performance across different graph scales and architectures. The enhanced KD framework, with its specialized loss components and training stability improvements, is particularly valuable for large-scale graph applications. This advancement facilitates the broader adoption of sophisticated graph learning in latency-critical and resource-limited environments. Future research could explore automating the selection of distillation components and extending these methods to emerging GNN architectures and other graph-related tasks.

REFERENCES

[1] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2021.

[2] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *International Conference on Learning Representations*, 2019.

[3] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[4] C. Yang, M. Wang, J. Pei, S. Bai, Y. Yuan, and W. Xu, "Distilling knowledge from graph convolutional networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 7074–7083.

[5] J. Chen, Y. Shen, S. Qiu, and B. W. Yang, "Knowledge distillation on graph neural networks: A survey," in *IJCAI*, 2022, pp. 5397–5405.

[6] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[7] V. P. Dwivedi, C. K. Joshi, T. Laurent, Y. Bengio, and X. Bresson, "Graph-bert: Only attention is needed for learning graph representations," in *International Conference on Learning Representations*, 2021.

[8] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," *arXiv preprint arXiv:1803.03635*, 2018.