MITIGATING STATE DILUTION IN GRAPH NEURAL NETWORKS FOR PREMISE RERANKING VIA BLOCK ATTENTION RESIDUALS

Levente Rácz
Eötvös Loránd University
raczlevente67879@gmail.com

1 Implemented Task: Reranking Premises, Not Tactics

Automated theorem proving combines a search procedure with a trusted symbolic checker, such as the Lean proof assistant. At one step of a Lean proof, the system knows the current goal and the local assumptions. Lean advances by executing *tactics*: commands such as `apply`, `rw`, or `exact`. A tactic may use an existing theorem or definition as an argument; in this paper, such a theorem or definition is called a *premise*.

Our experiment does not choose or rank tactics. It studies a narrower supporting task with three fixed parts:

1. **Input:** one current Lean proof state.
2. **Initial list:** for that state, ReProver returns an ordered list of 100 candidate premises.
3. **Output:** our model keeps exactly those same 100 premises and returns them in a new order.

ReProver repeats this retrieval for each proof state; the candidates are not one global list chosen once at the beginning of the proof. The experiment therefore evaluates **proof-state-conditioned premise reranking over fixed ReProver top-100 candidates**. The model cannot add a missing premise, choose a Lean tactic, generate Lean code, or prove the theorem by itself [6].

Proof search and neural-network depth are also different sequences. Proof search moves from one verified Lean state to another. Inside one state, however, the GNN processes the same input through several layers. Every earlier representation used below is an earlier processing stage of the *same current state*, not an earlier proof-tree node.

Deep message passing can make node representations less distinguishable (over-smoothing) [3] and can weaken information inherited from the input (over-dilution) [4]. We test whether access to earlier internal representations improves premise ranking and information retention across GNN depth.

2 Model: What Changes Across Depth

The main comparison uses two eight-layer relational GNNs. They receive the same proof-state graph and share the same node encoder, message-passing transformations, graph summary, premise representations, scorer, loss, initialization of common parameters, and training schedule. The intended difference is only how information crosses the eight layers.

The matched residual model passes the latest representation forward through ordinary skip connections. Block Attention Residuals group the layers into four two-layer blocks [5]. Before a layer is evaluated, the model forms a weighted average of the available earlier summaries,

$$c^{(\ell)} = \sum_i \alpha_i V_i^{(\ell)}, \qquad \alpha_i \geq 0, \quad \sum_i \alpha_i = 1. \tag{1}$$

The available values are the initial state encoding, completed block summaries, and the current partial block when one exists. Learned weights decide which summaries to use.

This idea is related to Highway Networks [7]: both create a learned route around several transformations. Highway Networks choose how much transformed and carried information to mix; BlockAttnRes chooses among several earlier block summaries. The attention is over at most five depth sources, not over every graph node. It adds 2,304 trainable parameters and some extra computation and memory. Runtime and total memory were not benchmarked, so no end-to-end efficiency advantage is claimed.

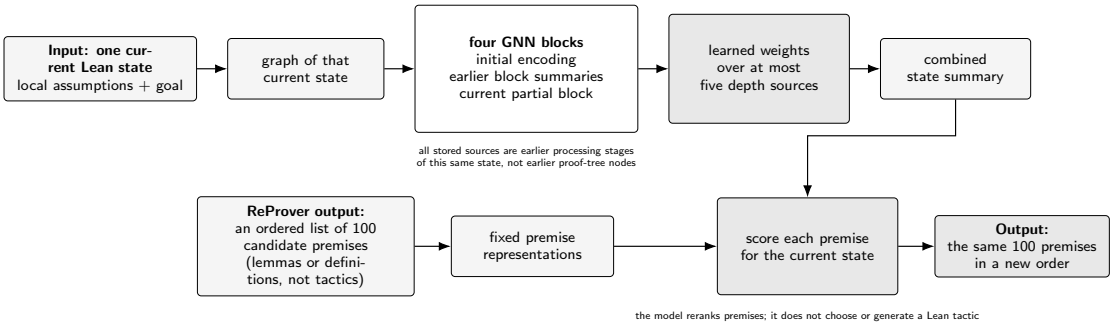

Figure 1: Implemented workflow. ReProver supplies 100 candidate premises—lemmas or definitions, not tactics—for the current Lean state. BlockAttnRes reranks only those same candidates. Its stored history contains earlier neural processing stages of the same state.

Table 1: Validation results. R@k values are percentages; higher is better. ReProver is one fixed, seedless reference. Learned entries are mean ± sample standard deviation over seeds 42/43/44. Bold values are the best result in each metric column. R@100 is 68.80% for every condition.

Condition	R@1 (%)	R@10 (%)	R@15 (%)	MRR
ReProver	**13.61**	**40.00**	45.22	**0.332**
Residual RGCN	10.80 ± 0.31	36.97 ± 0.17	44.22 ± 0.45	0.290 ± 0.005
Latest source only	7.99 ± 0.29	32.71 ± 0.64	39.54 ± 0.34	0.244 ± 0.002
Uniform history	10.95 ± 0.35	37.25 ± 0.61	44.14 ± 0.39	0.293 ± 0.001
Learned BlockAttnRes	11.55 ± 0.44	38.84 ± 0.37	45.67 ± 0.21	0.304 ± 0.004

3 Experiment and Measures

We use the LeanDojo Benchmark 4 random split and the untouched ReProver top-100 candidate lists [6]. The current goal and assumptions are converted into a graph of goals, declarations, tokens, and typed relations. Each candidate uses the same frozen 1,472-dimensional ReProver representation. A shared scorer compares the current-state representation with every candidate premise.

All learned conditions use seeds 42, 43, and 44 and the same candidates, data populations, starting values for common parameters, batches, AdamW settings, stopping rule, and checkpoint rule. No test-set result is used. The validation metrics cover 2,672 states with at least one resolved relevant premise. In 2,189 states, at least one relevant premise is present among ReProver's 100 candidates and can contribute to training. In the other 483 states, every relevant premise is missing from the candidate list, so no reranker can recover it. Separately, 131 of 6,514 validation annotations (2.01%) could not be matched to the corpus; these are data-alignment limitations rather than model errors.

R@k is the average fraction of known relevant premises appearing in the first k positions. R@15 is the primary measure: it asks how much useful material reaches the first 15 suggestions. Mean reciprocal rank (MRR) rewards placing at least one relevant premise very early. R@100 remains 68.80% for every condition because every method reorders the same 100 candidates.

4 What Improved?

The primary architectural comparison is BlockAttnRes versus the matched residual GNN. Mean R@15 rises from 44.22% to 45.67%, a gain of 1.45 percentage points, and MRR rises from 0.290 to 0.304. BlockAttnRes is better on R@1, R@10, R@15, and MRR in all three paired seeds. Because the candidates and shared training choices are fixed, this is the cleanest evidence for the depth-retrieval mechanism.

The ReProver comparison is mixed. BlockAttnRes is slightly better on R@15 (45.67% versus 45.22%), while ReProver remains better on R@1, R@10, and MRR. Thus BlockAttnRes places slightly more relevant material somewhere in the first 15 positions, whereas ReProver more often places the first useful premise very near the top.

5 Which Part Mattered?

Two controls isolate the mechanism. *Latest source only* keeps the four-block structure but uses only the newest summary; its R@15 falls to 39.54%. *Uniform history* can use every earlier summary but averages them equally; it reaches 44.14%. The learned model reaches 45.67%.

Earlier summaries therefore recover most of the failure of latest-only block processing. Equal averaging, however, does not beat the residual model on R@15. The clear gain appears when the network learns which earlier summaries are useful for the current state. This weakens a block-structure-only explanation and supports learned selection over same-state computational history.

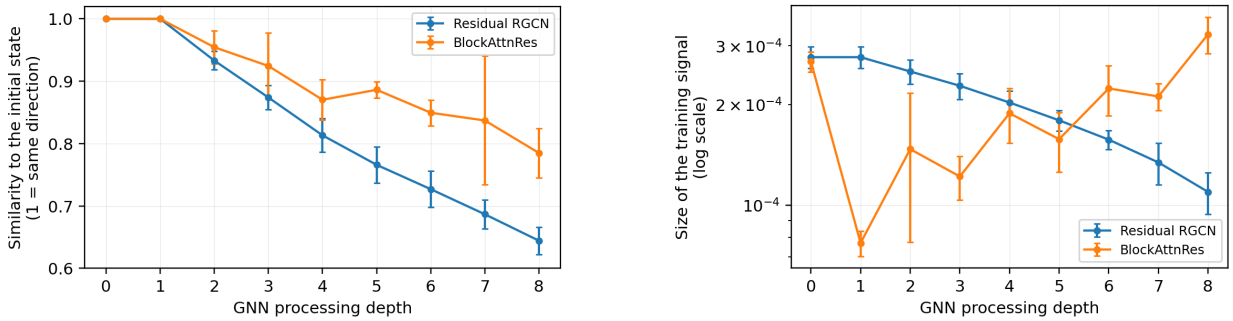

Figure 2: Depth diagnostics averaged over three seeds. Left: similarity in direction to the initial state. Right: size of the training signal. Error bars show variation across seeds.

6 Internal Measurements

Two indirect measurements examine how information and training signals behave across the eight GNN layers. At the final layer, the direction of the BlockAttnRes representation remains more similar to the initial state representation than in the residual model (0.785 versus 0.644). Its size nevertheless falls to 35% of its initial value, so the input is not preserved unchanged.

The training signal is weaker for BlockAttnRes near the first layer but almost three times stronger at the final layer. We therefore describe it as *redistributed toward later layers*, not uniformly improved. The learned final selector also assigns 46.6% of its average weight to the initial encoding and earlier blocks, confirming that it uses more than the newest source. These observations support the mechanism, but attention weights alone are not causal importance scores.

7 Limitations and Conclusion

This is a validation-set study with three paired seeds. It evaluates premise reranking, not tactic generation, complete Lean proof search, or theorem-proving success. It does not directly quantify node over-smoothing, and the internal measurements cannot prove that state dilution has been eliminated. We did not compare with a Transformer, and runtime and total memory remain unmeasured.

Within these limits, the behavioral results, internal measurements, and controlled ablations tell a consistent story. Access to earlier summaries repairs much of the failure of latest-only processing, and learned selection adds the improvement over uniform averaging. BlockAttnRes therefore improves fixed-candidate Lean premise reranking over a matched residual GNN and provides evidence consistent with mitigating information loss across GNN depth. Future work should evaluate a held-out test split, benchmark runtime and memory, compare broader encoder families, and integrate the reranker into end-to-end Lean proof search.

References

[1] Ambati, M. *ProofNet++: A Neuro-Symbolic System for Formal Proof Verification with Self-Correction.* arXiv:2505.24230, 2025.

[2] Trinh, T. H. et al. *Solving olympiad geometry without human demonstrations.* Nature, 2024.

[3] Rusch, T. K., Bronstein, M. M., and Mishra, S. *A Survey on Oversmoothing in Graph Neural Networks.* arXiv:2303.10993, 2023.

[4] Lee, J., Thost, V., and Kim, B. *Understanding and Tackling Over-Dilution in Graph Neural Networks.* arXiv:2508.16829, 2025.

[5] Kimi Team. *Attention Residuals: Technical Report.* arXiv:2603.15031, 2026.

[6] Yang, K. et al. *LeanDojo: Theorem Proving with Retrieval-Augmented Language Models.* NeurIPS, 2023.

[7] Srivastava, R. K., Greff, K., and Schmidhuber, J. *Training Very Deep Networks.* NeurIPS, 2015.