Graph Structure from Point Clouds: Geometric Attention is All You Need

Daniel Murnane
Scientific Data Division
Lawrence Berkeley National Laboratory
Berkeley, CA USA
dtmurnane@lbl.gov

Abstract

The use of graph neural networks has produced significant advances in point cloud problems, such as those found in high energy physics. The question of how to produce a graph structure in these problems is usually treated as a matter of heuristics, employing fully connected graphs or K-nearest neighbors. In this work, we elevate this question to utmost importance as the Topology Problem. We propose an attention mechanism that allows a graph to be constructed in a learned space that handles geometrically the flow of relevance, providing one solution to the Topology Problem. We test this architecture, called GravNet++, on the task of top jet tagging, and show that it is competitive in tagging accuracy, and uses far fewer computational resources than all other comparable models.

1 Introduction

Relational neural networks such as transformers and graph neural networks (GNNs) have pushed the limits of ML performance on many tasks, and the attention mechanism has been shown to be a key ingredient for achieving these state-of-the-art (SotA) results [1]. In natural language processing for example, attention-based transformers treat sentences as graphs, where words are represented by nodes and are "fully connected" (FC) - that is, all nodes are connected to all other nodes [2]. Attention-based GNNs have also been successfully employed in high energy physics (HEP) [3–12], a domain where data is often represented by point clouds of objects in space. The choice of how to connect point-cloud nodes into a graph is often non-obvious. A FC topology scales poorly with the complexity of the problem, possibly being prohibited by hardware constraints. Additionally, attention in many models is handled separately from the construction of the graph (the "choice of topology"), and is usually obtained as a learned function of pairs of node features, which can be computationally expensive. In short, if not handled carefully, an attention mechanism applied to a fully-connected point cloud scales as $O(N^2_{nodes})$.

In this work, we seek to address both of these hurdles - the choice of topology and the cost of attention - with a single solution. By adapting an existing architecture called GravNet [13], we propose an attention mechanism that is entirely dependent on a learned embedding space, and in doing so construct the topology of the graph in that space, at each iteration of message passing. The resulting network is called Gravnet++ as it "corrects" a subtle shortcoming of the original implementation, such that the relevance of neighboring nodes was diffused through a mixture of geometry and node features, thus requiring the use of a K-nearest-neighbor graph construction to function well. Our updated model instead learns the appropriate neighborhood size node-by-node, and in doing so uses fewer computational resources and performs with better accuracy than the original Gravnet. Additionally, we apply Gravnet++ to a classic point cloud problem - jet flavor tagging - and show it is competitive with SotA methods, while taking an order of magnitude less memory, and a factor of

Machine Learning and the Physical Sciences workshop, NeurIPS 2022.

four less time. We propose several extensions to this model that may improve accuracy further, while still retaining the learned geometric attention that makes it desirable for point cloud applications.

2 Geometric Attention and the Topology Problem

2.1 Constructing a Graph

Much work has been done in applying machine learning techniques to point cloud problems[14], and in particular attention models, typically for 3D points [15–18]. We take as a case-study the problem of tagging jets of reconstructed particles as coming from either a top quark or a lighter hadronic particle[1] In this case as in most point cloud problems, we are given only a set of points (herein called "nodes"), each with a feature vector, but without any notion of inter-node connections or relationships (herein called "edges"). To apply a GNN to these problems, there are two limiting approaches. The first is to treat the nodes as unconnected - that is, as a set. The DeepSets architecture [20] has been used in jet tagging with, at the time, SotA results [21, 22]. The other limit is to to treat the point cloud as fully connected, and this is the approach taken in transformer models, such as the Particle Transformer [23], which outperforms the set-limit approach in top tagging, although with significant computational overhead. A happy medium is struck by ParticleNet [24], a model that applies a GNN to neighborhoods of K=16 neighbors and achieves very good results[2]. Given these three working points (unconnected, fully-connected, and sparsely connected), we therefore suggest that including graph structure benefits a model's predictive power, but that most node-pair connections are not relevant to the prediction task.

The attention mechanism addresses exactly this hypothesis. A multilayer perceptron (MLP), applied to pairs of nodes, learns which neighboring nodes carry relevant features and up-weight them in the message passing aggregation. The catch-22 is that nodes must be connected somehow in order to apply the weighted aggregation. The question of how to form edges we refer to as the Topology Problem:

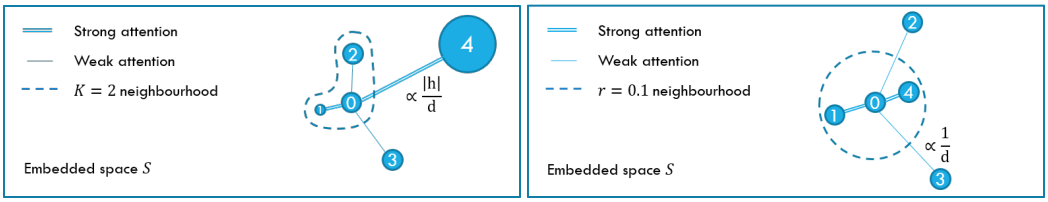
Given a variable-sized set of nodes and a loss function, then aside from a set of optima achieved by the learned attention MLP weights, there is also a set of optima achieved by the topology of the attention-based message passing. A rigorous approach to finding these optima is a solution to the Topology Problem.

For a sparsely connected GNN, for a particular message passing step, it is non-obvious which nodes are most informational or relevant to other nodes. Many construction approaches, such as that used in ParticleNet, assume the best topology to be homophilic - that is, nodes with similar latent features should be topologically close. However, this is an arbitrary constraint, and some message steps may benefit from connections with dissimilar nodes[3]. The solution is partly provided by having a second, independent latent space in which the graph is constructed. This is the mechanism adopted by GravNet and GarNet, two models proposed for GNN learning on point clouds.

In the elegant approach suggested by the authors of GravNet, two latent spaces are learned for each node update step. The first is the hidden features to be aggregated, h_i. The second is an embedding space vector $\vec{s}_i \in S$ to be used to calculate the KNN neighborhood K and attention weights A_{ij}. Both latent spaces are learned by MLPs applied independently to the input features of each GravNet convolution layer. The aggregated node features are thus given as

$$h'_i = \sum_{j \in K} A(d_{ij}, h_j) \cdot h_j, \qquad \text{where} \qquad A(d_{ij}, h_j) = e^{-Gd_{ij}^2}, \qquad d_{ij} = |\vec{s}_i - \vec{s}_j|_{p=2} \quad (1)$$

where G is a hyperparameter that acts like a gravitational constant[4].



(a) Original GravNet (b) GravNet++

Figure 1: Sketch of the GravNet attention mechanisms. The original GravNet node update propagates features h proportionally to $|h|/d$, such that a node is affected by nearby (in embedded space S) *and* heavy nodes. GravNet++ constrains information to flow only through a function of distance, and therefore the geometry fully captures the attention mechanism. Thus only *nearby* nodes need to be considered in the node update function.

2.2 Geometry as Attention: GravNet++

We are motivated to refine the GravNet architecture by the Topology Problem: Is the attention given to each neighboring node completely captured by the embedding space, and thus is an optimal topology constructed? Intuitively, we look at which information or relevance flows from one node to the next in message passing[5]. In the original GravNet model, nodes are influenced proportionally to both the closeness of a neighbor d_{ij} and the *size* of a neighbor $|h_j|_1$. This is sketched in fig. 1a. The latter $|h_j|_1$ factor means that a distant neighbor may still have an oversized influence if it is an "important" node (whatever this may mean in the problem being considered). Thus, a graph constructed according to nearness in S will not necessarily reflect the attention function, leading to important connections possibly being missed, and a suboptimal solution to the Topology Problem. Flow of information as a function of both neighbor size and distance is well-defined in a FC graph, hence the excellent performance of transformers. However if we require a sparse topology, we need to know which neighbors to connect. In the GravNet case, they will be connections that maximize $\frac{\text{size}}{\text{distance}}$ - an expensive calculation. Instead, if the weighting of information is only a function of distance, we only need to consider neighbors within a radius r in S, which can be calculated efficiently and scales well with graph size.

The solution is simple: Normalize hidden features such that all nodes have a total size of 1, and therefore constrain the GNN to pass all relevance through the geometry of S alone. That is, we take $A(d_{ij}, h_j) = exp(-G\frac{d_{ij}^2}{r^2})/|h_j|_{p=1}$. Although a seemingly minor alteration, this produces a most-minimal implementation of geometry-constrained attention mechanism. We also introduce a factor $\frac{1}{r^2}$ in the attention function. This new hyperparameter r appears in the following training procedure: Assuming now that all attention is constrained to the neighbourhood of each node in S, we should train and inference our model using topology built from that neighbourhood only. That is, we construct a radius graph in each message passing step, with radius r. Once this r hyperparameter is set, e.g. to $r = 1$, then the gravitational constant G can then be used to tune the sparsity of the topology. E.g. a choice of $G = 3$ means that nodes at distance $r = 1$ will be given an attention weight of around 0.05. For the problem considered here, this seems to be the choice of G above which performance plateaus. The effect of normalizing node sizes is sketched in fig. 1b. The details of the implementation and the training procedure are available in a public Github repository [28].

[1]This task is well-established, and many good descriptions are available, for example in [19].

[2]Apples-to-apples comparisons are subtle, as training dataset size is a large factor in performance. See [23] for a comprehensive analysis.

[3]See [25] for a review of heterophily in graph neural networks.

[4]Observe the notation $|\vec{s}|_p = \sqrt[p]{\sum s_a^p}$

[5]We can formalize this intuition using Layerwise Relevance Propogation (LRP) analysis. An introduction to this is given in [26] and an application to GNNs developed in [27]. The full calculation of LRP in geometry-constrained attention will be provided in an upcoming study.

Model	Acc	AUC	$\epsilon_B^{-1}\vert_{30\%}$
P-CNN	0.936	0.9837	1174 ± 58
PFN	0.932	0.9819	888 ± 17
Gravnet	0.937	0.9844	1340 ± 69
ParticleNet	0.940	0.9858	1615 ± 93
Gravnet++	0.939	0.9850	1438 ± 35

Table 1: Comparison of top tagging physics performance for a selection of DNNs [19, 30, 22]. The performance of the first three models is quoted from [24], and all results are averaged across five training runs[6]. Variation across these runs is given for background rejection, while variation of accuracy and AUC is negligible. Other high-performing taggers ([31, 23]) are not compared here as they contain features orthogonal to geometric attention, such as equivariance. A future study will seek to combine these mechanisms.

3 Results

3.1 Top Tagging Problem

The dataset used in this study comes from [29], which contains a set of 1.4m training jets, and 400k each of validation and test jet samples. A jet contains up to 200 constituent reconstructed particle 4-vectors, which we take as nodes. A further 17 hand-engineered features are attached to each node, taken to match those described in [24]. The task of top tagging is to classify each jet as either originating from the decay of a top quark, or from the decay of a lighter quark or gluon. We thus treat this as a graph-level binary classification problem, where the GNN must output a classification score between 0 and 1 for each graph, which is used in a binary cross entropy loss function, with no positive weighting as the dataset is well-balanced.

3.2 Physics Performance

An initial study of the physics performance of the original GravNet and GravNet++ is presented in table 1, along with several other well-performing deep neural networks. Both the accuracy and area under the ROC curve (AUC) are given, as well as the background rejection rate ϵ_B^{-1} (where ϵ_B is the false positive rate) at a working point of 30% efficiency.

One can see that GravNet++ outperforms all other models, except for ParticleNet. This shortcoming in performance can be attributed to several factors. The first is that layer sizes are heuristically taken from existing models, and may not be optimally suited to this new architecture. Additionally, in training, we note significant overfitting even on the full training set of 1.2 million jets and with a dropout of 0.2. Performance plateaus above this dropout rate. As such, we propose in an upcoming work to use a larger dataset such as that created in [23], to fully explore the predictive power of GravNet++. One can also see in the table that the original GravNet performs well, but not equivalently with the updated variant.

Further improvements are being studied, and will be presented in a near-future work, to boost the physics performance of GravNet++. These include dividing the spatial vector to use as a multi-headed attention (a mechanism implicit in the ParticleNet architecture), and learning dynamically the *number* of message passing steps each node requires, just as we do with the number of topological neighbors. These will both add expressiveness without losing the geometry-constrained attention mechanism.

3.3 Computational Performance

Inference performance is here measured by both the peak memory usage (taken as a proxy for the kind of hardware limitation these models may impose), and the average jet inference time in microseconds. Presented in table 2, we see that GravNet++ is by far the most computationally efficient. Despite

[6]A note on ParticleNet performance: This is the published performance. We were not able to obtain this result. The training techniques used in that work could also be used to improve GravNet++ performance.

Model	# Parameters	Max. memory (Gb)	Time (μs per jet)
P-CNN	348k	-	110
PFN	82k	-	120
ParticleNet	467k	3.1	88
Gravnet	545k	0.87	37
Gravnet++	545k	**0.23**	**22**

Table 2: Comparison of memory and time requirements of top taggers. Best performances are given in bold. Performance is measured on an Nvidia 40Gb A100, with batch size 1000. Timings are given *per jet*, that is $t_{jet} = t_{batch}/1000$. The first two model timings are quoted from [31].

having a comparable number of parameters to other DNNs, this model has two features that allow superior performance. The first is the geometric attention mechanism. Since attention is learned node-wise in embedded space, the embedding step (i.e. the forward pass from $h_i \to \vec{s}_i$) scales as $O(N_{nodes})$. We see that both GravNet variants benefit from this. Compare this with the standard edge-wise attention, such as that employed in ParticleNet, which scales as $O(N_{edges})$.

The second feature is that the topology is completely learned, so neighborhoods are only as large as required for good performance[7]. This allows GravNet++ to consume fewer resources than GravNet. In particular, a radius graph construction scales naively as $O(kN_{nodes})$ (where k is the average neighborhood size), while a KNN construction requires neighbors to be sorted and scales naively as $O(N^2_{nodes})$[32]. Additionally, K values are set arbitrarily by hand, but GravNet++ learns to build neighborhoods of mean size [3, 8, 13] (in the top tagging case, in order of node update step), significantly improving the throughput of both the graph-building and aggregation operations.

4 Conclusion

In this work, we have explored a long-standing obstacle in the application of graph neural networks to point clouds, which we term the Topology Problem. We present one set of solutions to this, in the form of a geometry-constrained attention. In particular, we alter the pre-existing GravNet architecture to construct a minimal geometric attention model, and show how it intuitively leads to a topology that captures the node connections with highest attention. We have taken an example use case to be graph-level top tagging, however the use of geometric attention could be applied to node-level or edge-level prediction tasks, and we will present results on those tasks in upcoming work. We show that our GravNet++ variation is competitive in tagging accuracy with other state-of-the-art taggers, while requiring far fewer computational resources. As this is the "most-minimal" geometric attention model, we will explore ways to further boost tagging accuracy in upcoming work. The codebase is available on Github [28].

[7] It is the case that the attention varies smoothly with the geometry, so some arbitrary choice of radius still needs to be made. However, we can quantify exactly the relevance of nodes outside this radius by e^{-G}, which is less than 5% for $G = 3$

5 Impact Statement

In this work, we propose several ideas that we hope will stimulate further discussion and research directions. These include:

- A presentation of the **Topology Problem** - an oft-overlooked issue that is usually solved ad hoc in graph neural network application to point clouds. In reality, as high energy physics datasets grow in size and complexity, a careful analysis of how graph topology is constructed will be essential to scaling up production-ready models in collider and astroparticle experiments.
- A **geometry-constrained attention operator**, as applied in an amended version of the GravNet architecture. This can be seen as a most-minimal construction of a GNN that propagates all relevance entirely through geometry, and may open the door to more sophisticated attention geometries. Regardless, the operation as presented here can be dropped into existing architectures to greatly improve computational efficiency.
- Some suggestions are given of further exploration of geometric-constrained attention, including multi-headed attention and learned number of message passing iterations.

We do not expect this work to have any negative societal or ethical impacts.

6 Acknowledgements

This work is supported by the US DoE's Office of Science, under contract DE-AC02-05CH11231 (CompHEP Exa.TrkX) and the Exascale Computing Project (17-SC-20-SC). This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility located at Lawrence Berkeley National Laboratory, operated under Contract No. DE-AC02-05CH11231. I am grateful to Paolo Calafiura for comments on this work, as well as Ryan Liu, Gage DeZoort and Tuan Pham for discussions.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[2] Chaitanya Joshi. Transformers are graph neural networks. *The Gradient*, page 5, 2020.

[3] Jonathan Shlomi, Peter Battaglia, and Jean-Roch Vlimant. Graph neural networks in particle physics. *Machine Learning: Science and Technology*, 2(2):021001, jan 2021.

[4] Yogesh Verma and Satyajit Jena. Particle track reconstruction using geometric deep learning, 2020.

[5] Shah Rukh Qasim, Nadezda Chernyavskaya, Jan Kieseler, Kenneth Long, Oleksandr Viazlo, Maurizio Pierini, and Raheel Nawaz. End-to-end multi-particle reconstruction in high occupancy imaging calorimeters with graph neural networks. *The European Physical Journal C*, 82(8), aug 2022.

[6] Joosep Pata, Javier Duarte, Jean-Roch Vlimant, Maurizio Pierini, and Maria Spiropulu. MLPF: efficient machine-learned particle-flow reconstruction using graph neural networks. *The European Physical Journal C*, 81(5), may 2021.

[7] Jun Guo, Jinmian Li, Tianjun Li, and Rao Zhang. Boosted higgs boson jet reconstruction via a graph neural network. *Physical Review D*, 103(11), jun 2021.

[8] Xiangyang Ju, Steven Farrell, Paolo Calafiura, Daniel Murnane, Prabhat, Lindsey Gray, Thomas Klijnsma, Kevin Pedro, Giuseppe Cerati, Jim Kowalkowski, Gabriel Perdue, Panagiotis Spentzouris, Nhan Tran, Jean-Roch Vlimant, Alexander Zlokapa, Joosep Pata, Maria Spiropulu, Sitong An, Adam Aurisano, Jeremy Hewes, Aristeidis Tsaris, Kazuhiro Terao, and Tracy Usher. Graph neural networks for particle reconstruction in high energy physics detectors, 2020.

[9] Xiangyang Ju, Daniel Murnane, Paolo Calafiura, Nicholas Choma, Sean Conlon, Steven Farrell, Yaoyuan Xu, Maria Spiropulu, Jean-Roch Vlimant, Adam Aurisano, et al. Performance of a geometric deep learning pipeline for hl-lhc particle tracking. *The European Physical Journal C*, 81(10):1–14, 2021.

[10] Savannah Thais, Paolo Calafiura, Grigorios Chachamis, Gage DeZoort, Javier Duarte, Sanmay Ganguly, Michael Kagan, Daniel Murnane, Mark S. Neubauer, and Kazuhiro Terao. Graph neural networks in particle physics: Implementations, innovations, and challenges, 2022.

[11] Jeremy Hewes, Adam Aurisano, Giuseppe Cerati, Jim Kowalkowski, Claire Lee, Wei-keng Liao, Alexandra Day, Ankit Agrawal, Maria Spiropulu, Jean-Roch Vlimant, et al. Graph neural network for object reconstruction in liquid argon time projection chambers. In *EPJ Web of Conferences*, volume 251, page 03054. EDP Sciences, 2021.

[12] Yutaro Iiyama, Gianluca Cerminara, Abhijay Gupta, Jan Kieseler, Vladimir Loncar, Maurizio Pierini, Shah Rukh Qasim, Marcel Rieger, Sioni Summers, Gerrit Van Onsem, et al. Distance-weighted graph neural networks on fpgas for real-time particle reconstruction in high energy physics. *Frontiers in big Data*, 3:598927, 2021.

[13] Shah Rukh Qasim, Jan Kieseler, Yutaro Iiyama, and Maurizio Pierini. Learning representations of irregular particle-detector geometry with distance-weighted graph networks. *The European Physical Journal C*, 79(7), jul 2019.

[14] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[15] Shi Qiu, Yunfan Wu, Saeed Anwar, and Chongyi Li. Investigating attention mechanism in 3d point cloud object detection, 2021.

[16] Xu Wang, Yi Jin, Yigang Cen, Tao Wang, and Yidong Li. Attention models for point clouds in deep learning: A survey, 2021.

[17] Lei Wang, Yuchun Huang, Yaolin Hou, Shenman Zhang, and Jie Shan. Graph attention convolution for point cloud semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10296–10305, 2019.

[18] Mingtao Feng, Liang Zhang, Xuefei Lin, Syed Zulqarnain Gilani, and Ajmal Mian. Point attention network for semantic segmentation of 3d point clouds, 2019.

[19] Gregor Kasieczka, Tilman Plehn, Anja Butter, Kyle Cranmer, Dipsikha Debnath, Barry Dillon, Malcolm Fairbairn, Darius Faroughy, Wojtek Fedorko, Christophe Gay, Loukas Gouskos, Jernej Kamenik, Patrick Komiske, Simon Leiss, Alison Lister, Sebastian Macaluso, Eric Metodiev, Liam Moore, Benjamin Nachman, and Sreedevi Varma. The machine learning landscape of top taggers. *SciPost Physics*, 7, 07 2019.

[20] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander Smola. Deep sets, 2017.

[21] Nicole Michelle Hartman, Michael Kagan, and Rafael Teixeira De Lima. Deep Sets for Flavor Tagging on the ATLAS Experiment. Technical report, CERN, Geneva, 2020.

[22] Patrick T. Komiske, Eric M. Metodiev, and Jesse Thaler. Energy flow networks: deep sets for particle jets. *Journal of High Energy Physics*, 2019(1), jan 2019.

[23] Huilin Qu, Congqiao Li, and Sitian Qian. Particle transformer for jet tagging, 2022.

[24] Huilin Qu and Loukas Gouskos. Jet tagging via particle clouds. *Physical Review D*, 101(5), mar 2020.

[25] Xin Zheng, Yixin Liu, Shirui Pan, Miao Zhang, Di Jin, and Philip S. Yu. Graph neural networks for graphs with heterophily: A survey, 2022.

[26] Grégoire Montavon, Alexander Binder, Sebastian Lapuschkin, Wojciech Samek, and Klaus-Robert Müller. Layer-wise relevance propagation: an overview. *Explainable AI: interpreting, explaining and visualizing deep learning*, pages 193–209, 2019.

[27] Farouk Mokhtar, Raghav Kansal, Daniel Diaz, Javier Duarte, Joosep Pata, Maurizio Pierini, and Jean-Roch Vlimant. Explaining machine-learned particle-flow reconstruction, 2021.

[28] Daniel Murnane. Geometric attention. `https://github.com/murnanedaniel/GeometricAttention`, 2022.

[29] Gregor Kasieczka, Tilman Plehn, Jennifer Thompson, and Michael Russel. Top quark tagging reference dataset. *Version v0 (2018_03_27). Mar*, 2019.

[30] Boosted jet identification using particle candidates and deep neural networks. 2017.

[31] Shiqi Gong, Qi Meng, Jue Zhang, Huilin Qu, Congqiao Li, Sitian Qian, Weitao Du, Zhi-Ming Ma, and Tie-Yan Liu. An efficient lorentz equivariant graph neural network for jet tagging. *Journal of High Energy Physics*, 2022(7), jul 2022.

[32] Yan-Ming Zhang, Kaizhu Huang, Guanggang Geng, and Cheng-Lin Liu. Fast knn graph construction with locality sensitive hashing. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 660–674. Springer, 2013.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [No] These will be provided in the same public repository as the model data files. These are already being prepared and will be released shortly once stable.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]