Training Graph Neural Networks with 1000 Layers – Supplementary Material –

Guohao Li [1 2] Matthias Müller [1] Bernard Ghanem [2] Vladlen Koltun [1]

A. Grouping of RevGNN

Grouped convolution is an effective way to reduce the parameter complexity in CNNs. We provide an ablation study to show how grouping reduces the number of parameters of RevGNNs. We conduct experiments on the *ogbn-proteins* dataset with different group sizes and report the results in Table 1. The number of hidden channels for all of these models is set to 224. We find that a larger group size reduces the number of parameters. As the group size increases from 2 to 4, the number of parameters reduces by more than 30%. The performance of models with 3 to 56 layers decreases slightly. The 112-layer networks achieve the same performance while the model with group size 4 uses only around 67% parameters compared to the model with group size 2. However, we observe that the GPU memory usage increases from 7.30 GB to 11.05 GB as the group size increases from 2 to 4 with our current implementation. We conjecture that this is due to our inefficient implementation. Optimizing our code for larger group sizes and conducting a more rigorous analysis is an interesting avenue for future investigation.

Table 1. **Ablation of the group size of group reversible GNNs on the *ogbn-protein* dataset.** L is the number of layers. The number of hidden channels is 224 for all the models.

	Group=2		Group=4	
#L	Params	ROC-AUC ↑	Params	ROC-AUC ↑
3	490k	85.09	339k	84.86
7	1.1M	85.68	750k	85.25
14	2.2M	86.62	1.5M	85.79
28	4.3M	86.68	2.9M	86.30
56	8.6M	86.90	5.8M	86.76
112	17.2M	87.02	11.5M	87.09

[1]Intel Labs [2]King Abdullah University of Science and Technology. Correspondence to: Guohao Li <guohao.li@kaust.edu.sa>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

B. Experimental Details and More Ablations

B.1. Datasets and Frameworks

We conduct experiments on three OGB datasets (Hu et al., 2020) including *ogbn-proteins*, *ogbn-arxiv* and *ogbn-products*. We follow the standard data splits and evaluation protocol of OGB 1.2.4. Please refer to the OGB website[1] for more details. Our code implementation relies on the deep learning framework Pytorch 1.6.0. We use Pytorch Geometric 1.6.1 (Fey & Lenssen, 2019) for all experiments except for the experiments with GATs where we use DGL 0.5.3 (Wang et al., 2019). The reversible module is implemented based on MemCNN (Leemput et al., 2019). The deep equilibrium module is implemented based on DEQ (Bai et al., 2019).

B.2. Hyperparameters and Experimental Settings

We describe all important hyperparameters and training settings that were not mentioned in the main paper for reproducibility. The settings are slightly different for each dataset.

Ogbn-proteins. The node features are initialized through aggregating connected edge features by a *sum* aggregation at the first layer. We use random partitioning for mini-batch training. The number of partitions is set to 10 for training and 5 for validation for all the ablated models. One subgraph is sampled at each SGD step. One layer normalization is used in the GNN block. Dropout with a rate of 0.1 is used for each layer. We use *max* as the message aggregator. Each model is trained for 2000 epochs using the Adam optimizer with a learning rate of 0.001.

Ablations on Ogbn-proteins. A detailed comparison of ResGNN, Weight-tied ResGNN, DEQ-GNN, RevGNN and Weight-tied RevGNN is shown in Table 2. Except for the DEQ-GNN, all the other models have an explicit depth of 112 layers. The reversible connections reduce the memory consumption significantly and enable training of wider RevGNNs. A 112-layer RevGNN achieves the best performance (87.02 ROC-AUC) among the compared models. DEQ-GNN with 64 channels and WT-RevGNN with 80

[1]https://ogb.stanford.edu/

Table 2. **Results on the *ogbn-proteins* dataset for various 112-layer networks.** Note that DEQ-GNN always has only a single layer that approximates an infinitely deep network. Each network is trained on one V100 GPU with 32GB of memory. The column *Mem* reports the GPU memory in GB, *Params* reports the number of model parameters, and *Time* reports the training time in days. *Baselines* are in italic.

Model	#Ch	ROC-AUC ↑	Mem ↓	Params	Time ↓
ResGNN	64	85.94	27.1	2.37M	1.3
ResGNN	224	-	OOM	28.4M	-
WT-ResGNN	64	83.30	27.4	51.2k	1.2
WT-ResGNN	224	-	OOM	537k	-
DEQ-GNN	64	83.17	2.22	51.3k	1.3
DEQ-GNN	224	85.84	7.60	537k	2.9
RevGNN	64	85.48	2.09	1.46M	1.8
RevGNN	80	85.97	2.56	2.25M	2.2
RevGNN	224	87.02	7.30	17.1M	4.9
WT-RevGNN	64	82.89	1.60	35.0k	1.7
WT-RevGNN	80	83.46	2.08	51.4k	2.0
WT-RevGNN	224	85.28	5.55	337k	4.8

channels have a similar number of parameters and memory consumption and also perform similarly. However, training DEQ-GNN is significantly faster than training WT-RevGNN (1.3 days *vs.* 2 days).

Multi-view Inference on Ogbn-proteins. To further improve the evaluation results, we propose multi-view inference which reduces the negative effects of random partitioning and noisy neighbors. During different inference passes, each vertex will see a different set of neighbors. We refer to this as multi-view inference and implement it by partitioning the graphs into different subgraphs in each inference pass. In Table 3, we find that performing inference with more views yields better results. We observe a substantial improvement with increasing number of views for both the RevGNN-Deep and RevGNN-Wide models. The results increase by about 0.4% in terms of ROC-AUC going from 1 view to 10 views. We also observe that a smaller number of partitions is favorable for evaluation. To reduce memory cost, automatic mixed precision[2] by NVIDIA is used for inference.

Ogbn-arxiv. The directed graph is converted into an undirected graph and self-loops are added. We use the full-batch setting for both training and testing. For the GCN (Kipf & Welling, 2017), SAGE (Hamilton et al., 2017) and GEN (Li et al., 2020) models, batch normalization and dropout with a rate of 0.5 is applied to each layer and the Adam optimizer with a learning rate of 0.001 is used to train the

[2]https://developer.nvidia.com/automatic-mixed-precision

Table 3. **Ablations for multi-view inference with RevGNN-Deep and RevGNN-Wide on the *ogbn-proteins* dataset.** *L*, *Ch*, *Views* and *Parts* denote the numbers of layers, channels, views and parts respectively. Doing inference with more views and less parts is favorable.

Model	#L	#Ch	#Views	#Parts	ROC-AUC ↑
RevGNN-Deep	1001	80	1	3	87.29 ± 0.16
RevGNN-Deep	1001	80	5	3	87.68 ± 0.13
RevGNN-Deep	1001	80	10	3	**87.74** ± 0.13
RevGNN-Wide	448	224	1	3	87.84 ± 0.21
RevGNN-Wide	448	224	5	3	88.20 ± 0.16
RevGNN-Wide	448	224	10	3	**88.24** ± 0.15
RevGNN-Wide	448	224	1	3	**87.84** ± 0.21
RevGNN-Wide	448	224	1	5	87.62 ± 0.18
RevGNN-Wide	448	224	1	10	87.23 ± 0.22

models for 2000 epochs. The GAT-based (Veličković et al., 2018) models are implemented based on the OGB leaderboard submission *GAT + norm. adj. + label reuse*[3]. The RevGAT models with self-knowledge distillation are implemented based on the submission *GAT + label reuse + self KD*[4]. The teacher models and student models have the same architecture. A knowledge distillation loss is added to the student model to minimize the Kullback–Leibler divergence between the teacher's predictions and the student's predictions during training. Please refer to the Github repositories for more details about the implementation.

Ogbn-products. Self-loops are added to the graph. We compare RevGNNs with full-batch training and mini-batch training. For mini-batch training, the graph is randomly partitioned into 10 subgraphs and one subgraph is sampled at each SGD step. We use full-batch testing in both scenarios. Batch normalization and dropout with a rate of 0.5 are used for each GNN block. The model is trained using the Adam optimizer with a learning rate of 0.001 for 1000 epochs.

B.3. GPU Memory Measurement

In all the experiments, the GPU memory usage is measured as the peak GPU memory during the first training epoch. Note that the measured GPU memory is larger than the GPU memory for storing node features due to the intermediate computation and network parameters. We consider the peak GPU memory usage as a practical metric since it is the bottleneck for training neural networks. As is common practice, we use `torch.cuda.max_memory_allocated()` for the memory measurement. However, note that

[3]https://github.com/Espylapiza/dgl/tree/master/examples/pytorch/ogb/ogbn-arxiv

[4]https://github.com/ShunliRen/dgl/tree/master/examples/pytorch/ogb/ogbn-arxiv

the measured peak GPU memory obtained using `torch.cuda.max_memory_allocated()` is usually smaller than the actual peak GPU memory obtained with NVIDIA-SMI.

B.4. Correlation of Model Predictions

We perform a correlation analysis on model predictions of RevGNN, Weight-tied RevGNN and DEQ-GNN. The pearson correlations of RevGNN with 1000 layers, WT-RevGNN-224 with 7 layers and DEQ-GNN-224 with 56 iterations are: 0.8571 (RevGNN *vs.* WT-RevGNN), 0.8565 (RevGNN *vs.* DEQ-GNN) and 0.8948 (WT-RevGNN *vs.* DEQ-GNN).

C. More Discussion

C.1. Gradient Checkpointing and Model Parallelism

By saving checkpoints every $\sqrt{L}$ steps, gradient checkpointing can achieve a memory complexity of $\mathcal{O}(\sqrt{L}ND)$, which is still higher than the memory complexity $\mathcal{O}(ND)$ of RevGNN. For 112-layer models with 64 hidden channels, ResGNN with gradient checkpointing consumes 2.5X the memory compared to RevGNN (5.22 G *vs.* 2.09 G) while reaching similar performance on *ogbn-proteins* with similar training time. Model parallelism is orthogonal to our approach. It would be interesting to investigate model parallelism to make RevGNN even wider with multiple GPUs.

C.2. Going Deeper and Datasets

In our experiments, we find that going deeper is very effective on *ogbn-proteins*. It would probably be beneficial to pre-train overparameterized GNNs on larger-scale protein datasets and then apply the pre-trained models to scientific applications such as drug discovery, protein structure prediction and gene-disease associations. For the other datasets such as *ogbn-products* and *ogbn-arxiv*, we observe less improvement when going very deep. It is still unclear what kind of datasets benefit more from depth and overparameterization. Investigating the relationship between overparameterization and factors such as dataset size, graph modality and graph learning task is an important direction of future work to better understand when overparameterized models are beneficial. We also anticipate that overparameterized GNNs will be a promising solution to even larger datasets such as OGB-LSC (Hu et al., 2021).

References

Bai, S., Kolter, J. Z., and Koltun, V. Deep equilibrium models. In *Advances in Neural Information Processing Systems*, 2019.

Fey, M. and Lenssen, J. E. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pp. 1024–1034, 2017.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. In *Advances in Neural Information Processing Systems*, volume 33, pp. 22118–22133, 2020.

Hu, W., Fey, M., Ren, H., Nakata, M., Dong, Y., and Leskovec, J. Ogb-lsc: A large-scale challenge for machine learning on graphs. *arXiv preprint arXiv:2103.09430*, 2021.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Leemput, S. C. v., Teuwen, J., Ginneken, B. v., and Manniesing, R. Memcnn: A python/pytorch package for creating memory-efficient invertible neural networks. *Journal of Open Source Software*, 4(39):1576, 7 2019. ISSN 2475-9066.

Li, G., Xiong, C., Thabet, A., and Ghanem, B. Deepergcn: All you need to train deeper gcns. *arXiv preprint arXiv:2006.07739*, 2020.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Wang, M., Zheng, D., Ye, Z., Gan, Q., Li, M., Song, X., Zhou, J., Ma, C., Yu, L., Gai, Y., Xiao, T., He, T., Karypis, G., Li, J., and Zhang, Z. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*, 2019.