Degree-Quant: Quantization-Aware Training for Graph Neural Networks

Shyam A. Tailor [*1] **Javier Fernandez-Marques** [*1] **Nicholas D. Lane** [1 2]

Abstract

Graph neural networks have demonstrated strong performance modelling non-uniform structured data. However, there exists little research exploring methods to make them more efficient at inference time. In this work, we explore the viability of training quantized GNNs models, enabling the usage of low precision integer arithmetic for inference. We propose a method, *Degree-Quant*, to improve performance over existing quantization-aware training baselines commonly used on other architectures, such as CNNs. Our work demonstrates that it is possible to train models using 8-bit integer arithmetic at inference-time with similar accuracy to their full precision counterparts.

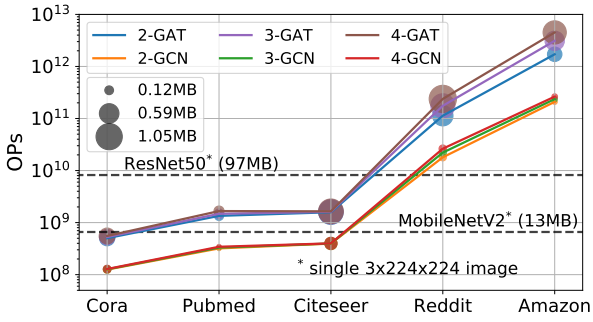

Figure 1. Despite model sizes for GNNs rarely exceeding 1MB, the OPs needed for inference grows at least linearly with dataset and node features dimensions. GNNs models $100\times$ smaller than popular CNNs require many more OPs to process large graphs.

1. Introduction

Graph neural networks (GNNs) have received substantial attention in recent years due to their ability to model irregularly structured data. As a result, they are extensively used for applications as diverse as molecular interactions (Duvenaud et al., 2015; Wu et al., 2017), social networks (Hamilton et al., 2017), recommendation systems (van den Berg et al., 2017) or program understanding (Allamanis et al., 2018). Recent advancements have centered around building more sophisticated graph models, including new types of layers (Kipf & Welling, 2017; Velickovic et al., 2018; Xu et al., 2019) and better aggregation functions (Corso et al., 2020). However, despite GNNs being small in terms of number of parameters, the compute required for each application remains tightly coupled to the input graph size. A 2-layer GCN model with 32 hidden units would result in a model size of just 81KB but requires 19 GigaOPs to process the entire Reddit graph. We illustrate this growth in Figure 1.

One major challenge with graph architectures is therefore performing inference efficiently, which limits the applications they can be deployed for. For example, GNNs have been combined with CNNs for SLAM feature matching (Sarlin et al., 2019), however it is not possible to deploy this technique on smartphones, or even smaller devices, where accelerators often do not implement floating point arithmetic, and instead favour more efficient integer arithmetic. Integer quantization is one way to lower the compute budget required to perform inference. This is achieved by employing fewer bits to represent each element involved in the forward pass, significantly reducing memory consumption and, compute and data movement costs.

Although quantization has been well studied for CNNs and language models (Jacob et al., 2017; Wang et al., 2018; Zafrir et al., 2019; Prato et al., 2019), there remains little work addressing GNN efficiency (Mukkara et al., 2018; Jia et al., 2020). To the best of our knowledge, there is no work studying quantization for GNNs, and explicitly characterising the issues that arise. The recent work of Wang et al. (2020) explores the use of binarized graph embeddings but limits the study to citation networks. Our work enables GNN quantization of any bit-width, which we evaluate under 8-bit and 4-bit settings (due to the availability of native hardware support) and under six datasets including citation networks, superpixel image classification, molecular regression and social graph classification. We discover and explain the sources of accuracy degradation in GNNs when using lower precision arithmetic, and propose a technique, *Degree-Quant*, to enable networks performing inference with 8-bit arithmetic to achieve similar accuracy to FP32 models.

[*]Equal contribution [1]Department of Computer Science, University of Oxford. [2]Samsung AI Center, Cambridge (UK). Correspondence to: Shyam A. Tailor <shyamatailor@gmail.com>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020. Copyright 2020 by the author(s).

2. Background

2.1. Message Passing Neural Networks (MPNNs)

Many popular GNN architectures may be viewed as generalizations of CNN architectures to an irregular domain: at a high level, graph architectures attempt to build representations based on a node's neighborhood. Unlike CNNs, however, this neighborhood does not have a fixed ordering or size. This work considers GNN architectures conforming to the MPNN paradigm (Gilmer et al., 2017). A graph $\mathcal{G}$ has node features $\mathbf{X} \in \mathbb{R}^{N \times F}$, an incidence matrix $\mathbf{I} \in \mathbb{N}^{2 \times E}$, and optionally D-dimensional edge features $\mathbf{E} \in \mathbb{R}^{E \times D}$. We focus on three popular architectures with update rules:

1. Graph Convolution Network (GCN) (Kipf & Welling, 2017): $\mathbf{x}'_v = \sum_{u \in \mathcal{N}(v) \cup \{v\}} [\frac{1}{\sqrt{\deg(u)\deg(v)}} \mathbf{\Theta} \mathbf{x}_u]$
2. Graph Attention Network (GAT) (Velickovic et al., 2018): $\mathbf{x}'_v = \alpha_{v,v} \mathbf{\Theta} \mathbf{x}_v + \sum_{u \in \mathcal{N}(v)} [\alpha_{v,u} \mathbf{\Theta} \mathbf{x}_u]$, where α represent attention coefficients.
3. Graph Isomorphism Network (GIN) (Xu et al., 2019): $\mathbf{x}'_v = h_{\mathbf{\Theta}}[(1+\epsilon)\mathbf{x}_v + \sum_{u \in \mathcal{N}(v)} \mathbf{x}_u]$, where h is a function (e.g. MLP) and ϵ is a constant, both learnable.

2.2. Quantization for Non-Graph Neural Networks

Quantization allows for model size reduction and inference speedup without changing the model architecture. While there exists extensive studies of the impact of quantization at different bit-widths (Courbariaux et al., 2015; Han et al., 2015; Louizos et al., 2017) and data formats (Carmichael et al., 2018; Kalamkar et al., 2019), it is 8-bit integer (INT8) quantization that has attracted the most attention. This is due to INT8 models reaching comparable accuracy levels to full-precision (FP32) models (Krishnamoorthi, 2018; Jacob et al., 2017), offer a $4\times$ model compression, and result in inference speedups on off-the-shelf hardware.

Quantization-aware training (QAT) has become the *de facto* approach towards designing robust quantized models (Wang et al., 2018; Zafrir et al., 2019; Wang et al., 2018). In their simplest forms, QAT schemes involve exposing the numerical errors introduced by quantization by simulating it on the forward pass and make use of straight-through estimator (STE) (Bengio et al., 2013) to compute the gradients. For integer QAT, the quantization of a tensor x during the forward pass is often implemented as: $x_q = \min(q_{\max}, \max(q_{\min}, \lfloor x/s + z \rceil))$, where $q_{\min}$ and $q_{\max}$ are the minimum and maximum representable values at a given bit-width and signedness, s is the scaling factor making x span the $[q_{\min}, q_{\max}]$ range and, z is the *zero-point*, which allows for the real value 0 to be representable in x_q. Both s and z are scalars obtained at training time. Then, the tensor is *dequantized* as: $\hat{x} = (x_q - z)s$. Other variants of integer QAT are presented in Jacob et al. (2017).

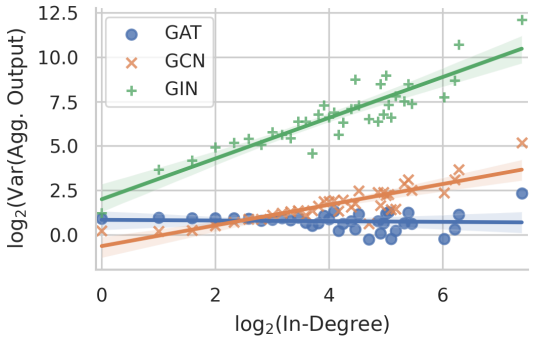

Figure 2. Analysis of values collected after aggregation at the final layer of FP32 GNNs trained on Cora. Generated using channel data collected from 100 runs. As in-degree grows, so does the mean and variance of channel values after aggregation.

Reaching performance comparable to FP32 models at lower bit-widths is not trivial. As a result, QAT schemes often rely on additional techniques to close the performance gap (Fan et al., 2020; Sheng et al., 2018; Alizadeh et al., 2019).

3. Quantization for GNNs

In this section, we build an intuition for why GNNs would fail with low precision arithmetic. Then, we propose our novel technique for QAT with GNNs, *Degree-Quant*.

3.1. Sources of Error

QAT relies upon the STE to make an estimate of the gradient despite the non-differentiable rounding in the forward pass. If this approximation is inaccurate, however, then poor performance will be obtained. In GNN layers, we identify the aggregation phase as a source of numerical error, especially at nodes with high in-degree due to two sources:

1. Outputs from aggregation have magnitudes that vary significantly depending on a node's in-degree: as it increases, the variance of output values will increase. Over the course of training $q_{\min}$ and $q_{\max}$ may become severely distorted by infrequent outliers, reducing the resolution for the vast majority of values observed.
2. Accumulation at large in-degree nodes where errors compound leads to the error being backpropagated to a large number of nodes, exacerbating the gradient error.

We can derive how the mean and variance of aggregation output values vary as node in-degree, n, increases for each of the three GNN layers, explaining the source (1) errors. Suppose we model incoming message values for a single output dimension with identically distributed random variables X_i, while making no assumptions on their exact distribution or independence. Further, we use Y_n as the random variable representing the value of node output after the aggregation step. With GIN layers, we have $Y_n = (1+\epsilon)X_0 + \sum_{i=1}^{n} X_i$. It can be proven that $\mathbb{E}(Y_n) = \mathcal{O}(n)$. The variance is also proportion to n in the case that we assume that

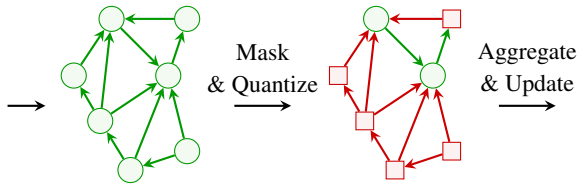

Figure 3. High-level view of the stochastic element of Degree-Quant. Masked (high in-degree) nodes, in green, operate at full precision, while unmasked nodes (red) operate at reduced precision. High in-degree nodes contribute most to poor gradient estimates, hence they are stochastically masked more often.

$\sum_{i \neq j} \text{Cov}(X_i, X_j) \ll \sum_i \text{Var}(X_i)$. This assumption is sensible: if $\sum_{i \neq j} \text{Cov}(X_i, X_j)$ is large then it implies that the network has learned highly redundant features, and may be a sign of over-fitting. Similar arguments can be made for GCN and GAT layers; we would expect GCN aggregation values to grow with $\mathcal{O}(\sqrt{n})$, and GAT aggregation values to remain constant due to the attention coefficients. We empirically validate these predictions on networks trained on the Cora dataset; results are plotted in fig. 2. We observe that the aggregation values do follow the trends predicted, and that for the values of in-degree in the plot (up to 168) the covariance terms can be neglected.

3.2. Our Method: Degree-Quant

To address these sources of error we propose *Degree-Quant* (DQ), a method for QAT with GNNs. Motivated by our observation that quantization error accumulates most at high in-degree nodes, our method targets these nodes specifically.

Algorithm 1 Training forward pass for Degree-Quant. Any function accepting the mask parameter $\mathbf{m}$ is understood to perform only the masked computations at full precision: intermediate tensors are *not* quantized. At test time, all operations are at low precision.

Input: Graph $\mathcal{G}$, masking probabilities $\mathbf{p}$
Create mask: $\mathbf{m} \leftarrow \text{Bernoulli}(\mathbf{p})$
Quantize weights: $\Theta \leftarrow \text{Quantize}(\Theta)$
$\mathcal{M} \leftarrow \text{MessageCalculate}(\mathcal{G}, \Theta', \mathbf{m})$
$X \leftarrow \text{Quantize}(\text{Aggregate}(\mathcal{M}, \Theta', \mathbf{m}), \mathbf{m})$
return $\text{Update}(X, \Theta', \mathbf{m})$

DQ aims to encourage more accurate gradients to flow through high in-degree nodes by probabilistically performing the forward pass at those nodes at full precision. At each layer a binary node mask is generated; masked nodes have the phases of the message passing, aggregation and update performed at full precision. This includes messages sent by masked nodes to other nodes, as shown in Figure 3. It is important to note that the weights used at all nodes are the quantized weights; this is motivated by the fact that our method is used to encourage more accurate gradients to flow back to the weights through high in-degree nodes. At test time masking is disabled: all nodes operate at low precision.

To generate the mask, we pre-process each graph before training and create a vector of probabilities $\mathbf{p}$ with length equal to the number of nodes. At training time, mask $\mathbf{m}$ is generated by sampling using the Bernoulli distribution: $\mathbf{m} \sim \text{Bernoulli}(\mathbf{p})$. In our scheme, the p_i is higher if the in-degree of node i is large. We use a simple scheme with two hyperparameters, p_{min} and p_{max}, to tune; nodes with the maximum in-degree are assigned p_{max} as their masking probability, with all other nodes assigned a probability calculated by linearly interpolating between p_{min} and p_{max} based on their in-degree ranking in the graph.

Figure 2 also demonstrates large fluctuations in variance as in-degree increases. Since these fluctuations can disproportionately affect the q_{min} and q_{max} found by commonly used methods, we propose using *percentiles*. While percentiles have been used for post-training quantization (Wu et al., 2020), we are the first (to the best of our knowledge) to propose making it a core part of QAT; we find it to be a key contributor to achieving good results with graphs. q_{min} and q_{max} are more representative of the vast majority of values in this scheme, resulting in greater precision as bits are not wasted for encoding infrequently observed values.

4. Experiments

In this section we evaluate Degree-Quant against strong FP32 and INT8-QAT baselines. Our study evaluates performance on six datasets and includes both transductive and inductive tasks. The datasets used were Cora, CiteSeer (Yang et al., 2016), ZINC (Jin et al., 2018), MNIST and CIFAR-10 superpixels (Knyazev et al., 2019), and REDDIT-BINARY (Yanardag & Vishwanathan, 2015). Across all datasets INT8 models trained with Degree-Quant manage to recover most of the accuracy lost as a result of quantization.

We use models described in Fey & Lenssen (2019) and Dwivedi et al. (2020). We frequently improved upon the results reported in these publications after extensive hyperparameter tuning. Significant gains were observed for GIN models for MNIST and CIFAR10 and large gains for all models on the ZINC dataset. For citation networks, our tuning resulted in models with considerably lower validation loss, however that did not translate into higher test accuracy.

To obtain quantized baselines, we first evaluated each type of layer and dataset with two variants of STE, *vanilla* STE and STE with gradient clipping, and two ways of tracking min/max ranges of the tensors to be quantized, using absolute values and using momentum. We found the choice of STE configuration to be highly dependent on the model architecture and type of problem to be solved: we see a much larger variance than is observed with CNNs. All QAT baselines use STE configurations informed by this analysis.

QAT-INT8 results in table 1, with the exception of MNIST (an easy to classify dataset), corroborate our hypothesis that

Table 1. Results for DQ at INT8, with percent points improvements over baseline QAT in bold (for ZINC we show relative improvement). While GCN and GAT models retain most accuracy for INT8, models with GIN layers result in larger degradations. Models trained with Degree-Quant (DQ), result in performances comparable to those of their FP32 counterparts, and surpassing FP32 baselines in some cases.

Quant. Scheme	Model Arch.	Node Classification (Accuracy %) Cora ↑	Citeseer ↑	Graph Classification (Accuracy %) MNIST ↑	CIFAR-10 ↑	Graph Regression (Loss) ZINC ↓
FP32	GCN	80.9 ± 0.7	71.4 ± 0.9	90.9 ± 0.4	58.4 ± 0.5	0.450 ± 0.008
	GAT	82.3 ± 0.8	70.5 ± 0.9	95.8 ± 0.4	65.1 ± 0.8	0.455 ± 0.006
	GIN	77.9 ± 1.1	65.1 ± 2.1	96.4 ± 0.4	57.4 ± 0.7	0.334 ± 0.024
QAT-INT8	GCN	81.0 ± 0.7	70.9 ± 0.7	90.9 ± 0.2	56.4 ± 0.5	0.481 ± 0.029
	GAT	81.9 ± 0.7	70.5 ± 0.9	95.8 ± 0.3	66.3 ± 0.4	0.460 ± 0.005
	GIN	75.6 ± 1.2	63.0 ± 2.6	96.7 ± 0.2	52.4 ± 1.2	0.386 ± 0.025
DQ-INT8	GCN	81.7 ± 0.7 (**+0.7**)	70.7 ± 0.9 (**-0.2**)	90.9 ± 0.1 (**+0.0**)	56.3 ± 0.1 (**-0.1**)	0.434 ± 0.009 (**+9.8**)
	GAT	82.1 ± 0.1 (**+0.2**)	70.8 ± 1.0 (**+0.3**)	95.8 ± 0.4 (**+0.0**)	67.7 ± 0.5 (**+1.4**)	0.456 ± 0.005 (**+0.9**)
	GIN	77.2 ± 1.2 (**+1.6**)	67.4 ± 1.4 (**+4.4**)	96.6 ± 0.4 (**-0.1**)	55.5 ± 0.6 (**+3.1**)	0.357 ± 0.014 (**+7.5**)

Table 2. Results for DQ-INT8 GIN models perform nearly as well as at FP32. For INT4, DQ offers a significant increase in accuracy. We focus on GIN as it is most susceptible to degradation.

Quantization	Model	REDDIT-BIN (Acc. %) ↑
FP32	GIN	92.0 ± 1.5
QAT-INT8	GIN	76.1 ± 7.5
DQ-INT8	GIN	91.8 ± 2.3 (**+15.7**)
QAT-INT4	GIN	54.4 ± 6.6
DQ-INT4	GIN	81.3 ± 4.4 (**+26.9**)

GIN layers are less resilient to quantization. This was first observed in fig. 2. In the case of ZINC, while all models results in noticeable degradation, GIN sees a more severe 16% increase of regression loss compared to our FP32 baseline. We present further results with GIN on REDDIT-BINARY in table 2, including a study of performance at INT4.

Citation networks trained with DQ manage to recover most of the accuracy lost as a results of QAT-INT8. In some instances DQ-INT8 models outperform the reference FP32 baselines. We see DQ being more effective for GIN layers, outperforming INT8 baselines for Citeseer (4.4%) and REDDIT-BINARY (15.7%) by large margins. For DQ, ratios of p_{min} and p_{max} in $[0.0, 0.2]$ were the most common.

5. Discussion

Benefits of Percentile Ranges. Figure 4 shows the importance of using percentiles during training. When using standard min/max that the upper range grows to over double the range for 99.9% of values: this effectively halves the quantization resolution for most values. We found that gradient clipping had no clear benefits when combined with percentiles: all results used the STE, with the exception of REDDIT-BINARY. DQ was also more stable, and we obtained strong results with an order of magnitude less tuning relative to the QAT baselines.

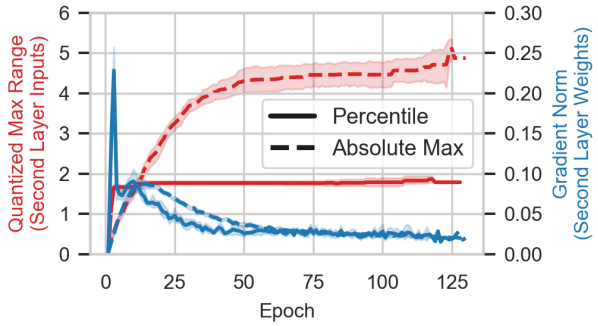

Figure 4. q_{max} with absolute min/max and percentile ranges, applied to INT8 GCN training on Cora. Percentile max is half that of the absolute, *doubling* resolution for the majority of values.

Performance Implications. The performance benefits of INT8 arithmetic have been studied for CNNs and other *regular* architectures (Horowitz, 2014; Jacob et al., 2017; Bhandare et al., 2019). However, we note that to obtain peak performance for GNNs it is necessary to process nodes with a cache-friendly ordering: the true speedup will be dataset and hardware dependent. It is worth emphasizing that quantized networks are necessary to use accelerators in smartphones and smaller devices as they primarily accelerate integer arithmetic. Quantized networks are also smaller and require less memory at inference-time.

6. Conclusion

This work has presented Degree-Quant, a method for training a diverse set of GNN architectures to obtain close to FP32 performance while using only 8-bit integer arithmetic. Our work is a first step towards enabling GNNs to be deployed more widely, including to resource constrained devices such as smartphones. We believe that our insights pave the way for research into mixed-precision training and further techniques for efficient inference with GNNs.

ACKNOWLEDGMENTS

This work was supported by Samsung AI, Arm and, by the UK's Engineering and Physical Sciences Research Council (EPSRC) with grants EP/M50659X/1 and EP/S001530/1.

References

Alizadeh, M., Fernández-Marqués, J., Lane, N. D., and Gal, Y. A empirical study of binary neural networks' optimisation. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJfUCoR5KX.

Allamanis, M., Brockschmidt, M., and Khademi, M. Learning to represent programs with graphs. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=BJOFETxR-.

Bengio, Y., Léonard, N., and Courville, A. Estimating or propagating gradients through stochastic neurons for conditional computation, 2013.

Bhandare, A., Sripathi, V., Karkada, D., Menon, V., Choi, S., Datta, K., and Saletore, V. Efficient 8-bit quantization of transformer neural machine language translation model, 2019.

Carmichael, Z., Langroudi, H. F., Khazanov, C., Lillie, J., Gustafson, J. L., and Kudithipudi, D. Deep positron: A deep neural network using the posit number system, 2018.

Corso, G., Cavalleri, L., Beaini, D., Liò, P., and Veličković, P. Principal neighbourhood aggregation for graph nets, 2020.

Courbariaux, M., Bengio, Y., and David, J.-P. Binaryconnect: Training deep neural networks with binary weights during propagations, 2015.

Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pp. 2224–2232, 2015.

Dwivedi, V. P., Joshi, C. K., Laurent, T., Bengio, Y., and Bresson, X. Benchmarking graph neural networks, 2020.

Fan, A., Stock, P., Graham, B., Grave, E., Gribonval, R., Jegou, H., and Joulin, A. Training with quantization noise for extreme model compression, 2020.

Fey, M. and Lenssen, J. E. Fast graph representation learning with pytorch geometric, 2019.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. *CoRR*, abs/1704.01212, 2017. URL http://arxiv.org/abs/1704.01212.

Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs, 2017.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, 2015.

Horowitz, M. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pp. 10–14, Feb 2014. doi: 10.1109/ISSCC.2014.6757323.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., and Kalenichenko, D. Quantization and training of neural networks for efficient integer-arithmetic-only inference, 2017.

Jia, Z., Lin, S., Gao, M., Zaharia, M., and Aiken, A. Improving the accuracy, scalability, and performance of graph neural networks with roc. In *Proceedings of Machine Learning and Systems 2020*, pp. 187–198. 2020.

Jin, W., Barzilay, R., and Jaakkola, T. Junction tree variational autoencoder for molecular graph generation, 2018.

Kalamkar, D., Mudigere, D., Mellempudi, N., Das, D., Banerjee, K., Avancha, S., Vooturi, D. T., Jammalamadaka, N., Huang, J., Yuen, H., Yang, J., Park, J., Heinecke, A., Georganas, E., Srinivasan, S., Kundu, A., Smelyanskiy, M., Kaul, B., and Dubey, P. A study of bfloat16 for deep learning training, 2019.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Knyazev, B., Taylor, G. W., and Amer, M. R. Understanding attention and generalization in graph neural networks, 2019.

Krishnamoorthi, R. Quantizing deep convolutional networks for efficient inference: A whitepaper, 2018.

Louizos, C., Ullrich, K., and Welling, M. Bayesian compression for deep learning, 2017.

Mukkara, A., Beckmann, N., Abeydeera, M., Ma, X., and Sanchez, D. Exploiting locality in graph analytics through hardware-accelerated traversal scheduling. In *Proceedings of the 51st Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-51, pp. 1–14. IEEE Press, 2018. ISBN 9781538662403. doi: 10.1109/MICRO.2018.00010. URL https://doi.org/10.1109/MICRO.2018.00010.

Prato, G., Charlaix, E., and Rezagholizadeh, M. Fully quantized transformer for machine translation, 2019.

Sarlin, P.-E., DeTone, D., Malisiewicz, T., and Rabinovich, A. Superglue: Learning feature matching with graph neural networks. *arXiv preprint arXiv:1911.11763*, 2019.

Sheng, T., Feng, C., Zhuo, S., Zhang, X., Shen, L., and Aleksic, M. A quantization-friendly separable convolution for mobilenets. *2018 1st Workshop on Energy Efficient Machine Learning and Cognitive Computing for Embedded Applications (EMC2)*, Mar 2018. doi: 10.1109/emc2.2018.00011. URL http://dx.doi.org/10.1109/emc2.2018.00011.

van den Berg, R., Kipf, T. N., and Welling, M. Graph convolutional matrix completion, 2017.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Wang, H., Lian, D., Zhang, Y., Qin, L., He, X., Lin, Y., and Lin, X. Binarized graph neural network, 2020.

Wang, K., Liu, Z., Lin, Y., Lin, J., and Han, S. Haq: Hardware-aware automated quantization with mixed precision, 2018.

Wu, H., Judd, P., Zhang, X., Isaev, M., and Micikevicius, P. Integer quantization for deep learning inference: Principles and empirical evaluation, 2020.

Wu, Z., Ramsundar, B., Feinberg, E. N., Gomes, J., Geniesse, C., Pappu, A. S., Leswing, K., and Pande, V. Moleculenet: A benchmark for molecular machine learning, 2017.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

Yanardag, P. and Vishwanathan, S. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pp. 1365–1374, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450336642. doi: 10.1145/2783258.2783417. URL https://doi.org/10.1145/2783258.2783417.

Yang, Z., Cohen, W. W., and Salakhutdinov, R. Revisiting semi-supervised learning with graph embeddings, 2016.

Zafrir, O., Boudoukh, G., Izsak, P., and Wasserblat, M. Q8bert: Quantized 8bit bert, 2019.