Correlation-Aware Joint-Pruning-Quantization using Graph Neural Networks

Muhammad Nor Azzafri Nor-Azman, Usman Ullah Sheikh, Mohammed Sultan Mohammed, Jeevan Sirkunan, Muhammad Nadzir Marsono

UTM
UNIVERSITI TEKNOLOGI MALAYSIA

Abstract

Deep learning excels in image classification but is constrained by its complexity. While joint pruning-quantization offers improvements, it can be further enhanced by considering layer correlations. This ❶ exposes redundant computations across layers, ❷ facilitates faster convergence in finding optimal pruning-quantization configurations, and ❸ achieves better or comparable complexity reduction compared to other works. This paper introduces Graph Neural Networks (GNNs) to aggregate these inter-layer relationships.

Methodology

Algorithm 1 Joint pruning-quantization using GNN and RL

Input: E_{max}, R_{solve}, N_{max}, C_{target}, I_{update}, K, Q

Output: $model_{pruned\text{-}quantized}$

1: $e \leftarrow 0$
2: **while** $e \leq E_{max}$ **and** $R \leq R_{solve}$ **do**
3: Reset $model_{base}$ and state
4: Set state through graph representation of $model_{base}$
5: $n \leftarrow 0$
6: **while** $n \leq N_{max}$ **and** $C_{current} \leq C_{target}$ **do**
7: Set $C_{current}$ based on agent's action
8: Prune $model_{base}$ based on agent's action
9: Quantize $model_{pruned}$ based on agent's action
10: Finetune $model_{pruned\text{-}quantized}$ for F epochs
11: Update state through graph representation of $model_{pruned\text{-}quantized}$
12: Get reward from $model_{pruned\text{-}quantized}$
13: Increment n
14: **end while**
15: **if** e mod I_{update} = 0 **then**
16: Train agent for K epochs
17: **end if**
18: Increment e
19: **end while**

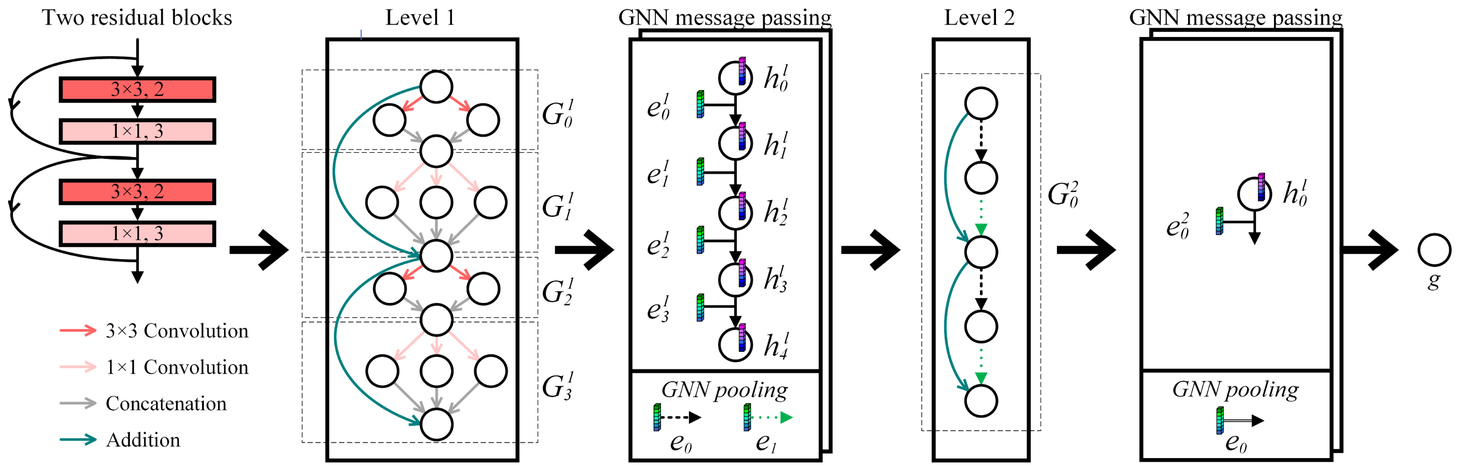

Fig. 1 Example of graph transformation for two residual block.

Line 8 – $P_d \in [0,1]$: Action determines the pruning ratio for channels with the lowest L_2-norm.

Line 9 – $Q_d \in [0,1]$: Action determines the bitwidth using the formula:
$b_d = \text{round}(b_{min} - 0.5 + Q_d \times (b_{max} - b_{min} + 1))$

Line 12 – R: Accuracy on validation dataset.

Line 16 – All GNN learnable weights are updated within the RL's proximal policy optimization.

Symbol	Representation
b_d	Bitwidth at layer d
C	Complexity reduction
d	Layer of $model_{base}$
e, E	Episode
F	Finetuning epochs
G	Computational graph
h^l	Hidden state at layer l
I	Interval for policy update
K	Epochs for policy update
l	Graph layer
n, N	Timesteps
P_d	Pruning action
Q_d	Quantization action
R	Reward

Overview of underlying process

The GNN is integrated into a reinforcement learning (RL) framework. The baseline or pruned-quantized model is transformed into a graph to serve as the RL **state**, where the agent proposes **actions** to prune and quantize the model, with the resulting accuracy as the **reward**. As episodes progress, the agent, leveraging the GNN to capture layer correlations, learns to propose optimal pruning-quantization actions that maximize accuracy.

Experimental setup

- CIFAR10 (5:1 training-to-validation ratio)
- ResNet20/56
- Batch size 256
- Pytorch's augmentation

Results ①

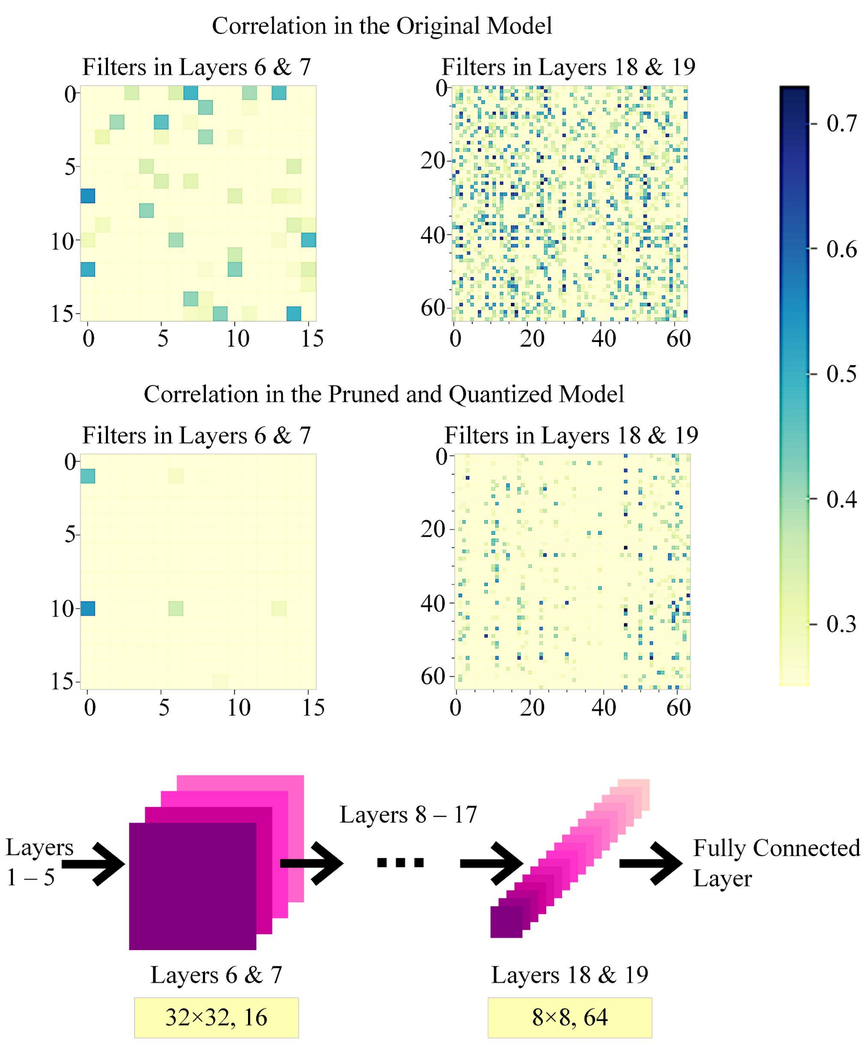

Fig. 2 Pearson correlation matrices

❶ Lower correlation in the joint pruning-quantization model suggests that each layer contributes unique computations.

Results ②

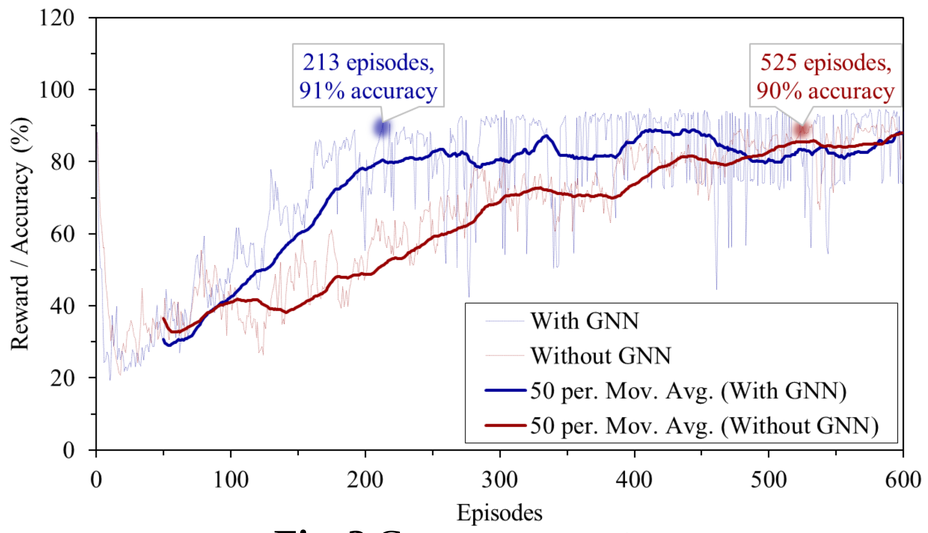

Fig. 3 Convergence rate

❷ The ablation study demonstrates that the GNN reduces the number of iterations by an average of 2.46x.

Results ③

DL Model	Approach	Complexity Reduction	Top-1 Accuracy
ResNet20	Baseline	0%	91.73%
	AGMC [1]	50%	91.42%
	GNNRL [2]	49%	91.31%
	DBNN [3]	99.1%	91.60%
	Proposed	**96.36%**	**91.62%**
ResNet56	Baseline	0%	93.39%
	AGMC [1]	50%	92.76%
	GNNRL [2]	50%	93.49%
	Proposed	**98.51%**	**92.80%**

❸ 96–98% complexity reduction with 0.1–1.1% accuracy trade-offs.

Future works

❶ Investigate how other joint pruning-quantization methods impact inter-layer correlations to gain further insights.

❷ Optimize the depth for the best trade-off, as fewer iterations increase per-iteration time.

❸ Validate with additional models and datasets.

References

[1] Sixing Yu, Arya Mazaheri, and Ali Jannesari, "Auto graph encoder-decoder for neural network pruning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 6362–6372.

[2] Sixing Yu, Arya Mazaheri, and Ali Jannesari, "Topology-aware network pruning using multi-stage graph embedding and reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2022, pp. 25656–25667.

[3] Yuhan Lin, Lingfeng Niu, Yang Xiao, and Ruizhi Zhou, "Diluted binary neural network," *Pattern Recognition*, vol. 140, pp. 109556, 2023.