A. Gradient of Quantization Mapping

In this section, we present the formula for the gradient of the quantization mapping with respect to d, t, and q_m.

$$\nabla_d x^Q = \operatorname{sgn}(x) \cdot \begin{cases} \left(\lfloor \frac{|x|^t}{d} \rceil - \frac{|x|^t}{d}\right), & |x| \leq q_m, \\ \left(\lfloor \frac{(q_m)^t}{d} \rceil - \frac{(q_m)^t}{d}\right), & |x| > q_m, \end{cases} \tag{15}$$

$$\nabla_t x^Q = \operatorname{sgn}(x) \cdot \begin{cases} |x|^t \log(|x|), & |x| \leq q_m, \\ (q_m)^t \log(q_m), & |x| > q_m, \end{cases} \tag{16}$$

$$\nabla_{q_m} x^Q = \begin{cases} 0, & |x| \leq q_m, \\ \operatorname{sgn}(x) t (q_m)^{t-1}, & |x| > q_m. \end{cases} \tag{17}$$

Remark. The computation involving x in this section represents element-wise operations.

B. Proof for Proposition 5.1

In this section, we present the proof for Proposition 5.1. For convenience, we restate the proposition as follows.

Proposition B.1. *Let $\hat{\nabla}_x f$ be the full gradient of function $f(x, d, q_m, t)$ with respect to x. With forget rate γ selection rule Eq. (13) and quantization step size d selection rule Eq. (14), the search direction $s(x)$ is a descent direction for the function f with respect to x at x.*

Proof. Denote the full gradient of function $f(x, d, q_m, t)$ with respect to x as $\nabla_x f$. The search direction $s(x)$ is rewritten as

$$s(x) = \begin{cases} -\alpha[\nabla_x f]_g, & g \in \mathcal{G}_I, \\ -\alpha[\nabla_x f]_g - \gamma[x^Q]_g, & g \in \mathcal{G}_R. \end{cases} \tag{18}$$

Since $-\alpha[\nabla_x f]_g^T[\nabla_x f]_g < 0$ for $g \in \mathcal{G}_I$, it suffices to show that for $g \in \mathcal{G}_R$,

$$[\nabla_x f]_g^T \left[-\alpha[\nabla_x f]_g - \gamma[x^Q]_g\right] < 0.$$

It follows from (9) that for $g \in \mathcal{G}_R$,

$$-\alpha[\nabla_x f]_g - \gamma[x^Q]_g = \underbrace{-\alpha[\nabla_x f]_g - \gamma[\operatorname{sgn}(x) \cdot \text{clip}_{q_m}^t(|x|)]_g}_{[s_{\text{clip}}(x)]_g} - \gamma \cdot d[\operatorname{sgn}(x) \cdot R(x)]_g.$$

Denote the angle between $-[\nabla_x f]_g$ and $-[\operatorname{sgn}(x) \cdot \text{clip}_{q_m}^t(|x|)]_g$ as θ_γ. It follows that the vector $[s_{\text{clip}}(x)]_g$ can be decomposed into two orthogonal vectors, i.e.,

$$[s_{\text{clip}}(x)]_g = [\hat{s}_{\text{clip}}(x)]_g + [\tilde{s}_{\text{clip}}(x)]_g,$$

where $[\hat{s}_{\text{clip}}(x)]_g$ is orthogonal to vector $[\nabla_x f]_g$ and $[\tilde{s}_{\text{clip}}(x)]_g$ is parallel to vector $[\nabla_x f]_g$. Since $[\hat{s}_{\text{clip}}(x)]_g^T[\nabla_x f]_g = 0$, we have that

$$\|[\hat{s}_{\text{clip}}(x)]_g\| = \gamma \sin\theta_\gamma \|[\operatorname{sgn}(x) \cdot \text{clip}_{q_m}^t(|x|)]_g\|.$$

Using the orthogonality between vector $[\hat{s}_{\text{clip}}(x)]_g$ and vector $[\tilde{s}_{\text{clip}}(x)]_g$, we have that

$$\begin{aligned} \|[\tilde{s}_{\text{clip}}(x)]_g\|^2 &= \|[s_{\text{clip}}(x)]_g\|^2 - \|[\hat{s}_{\text{clip}}(x)]_g\|^2 \\ &= \| -\alpha[\nabla_x f]_g - \gamma[\operatorname{sgn}(x) \cdot \text{clip}_{q_m}^t(|x|)]_g\|^2 - \gamma^2 \sin^2\theta_\gamma \|[\operatorname{sgn}(x) \cdot \text{clip}_{q_m}^t(|x|)]_g\|^2 \\ &= \alpha^2\|[\nabla_x f]_g\|^2 + 2\alpha\gamma[\nabla f(x)]_g^T[\operatorname{sgn}(x) \cdot \text{clip}_{q_m}^t(|x|)]_g + \gamma^2\cos^2\theta_\gamma\|[\operatorname{sgn}(x) \cdot \text{clip}_{q_m}^t(|x|)]_g\|^2 \\ &= \left[\alpha\|[\nabla_x f]_g\| + \gamma\cos\theta_\gamma\|[\operatorname{sgn}(x) \cdot \text{clip}_{q_m}^t(|x|)]_g\|\right]^2. \end{aligned}$$

Given the norm and direction of the vector $[\tilde{s}_{\text{clip}}(x)]_g$, we have $[\tilde{s}_{\text{clip}}(x)]_g$ expressed as, for $g \in \mathcal{G}_R$,

$$[\tilde{s}_{\text{clip}}(x)]_g = -\frac{\alpha\|[\nabla_x f]_g\| + \gamma\cos(\theta_\gamma)\|[\operatorname{sgn}(x) \cdot \text{clip}_{q_m}^t(|x|)]_g\|}{\|[\nabla_x f]_g\|}[\nabla_x f]_g. \tag{19}$$

Combining the forget rate selection rule (13) and the expression (19) allows us to have that for $g \in \mathcal{G}_R$,

$$\begin{aligned} [\nabla_x f]_g^T [s_{\text{clip}}(x)]_g &= [\nabla_x f]_g^T \left[[\hat{s}_{\text{clip}}(x)]_g + [\tilde{s}_{\text{clip}}(x)]_g\right] \\ &= [\nabla_x f]_g^T [\tilde{s}_{\text{clip}}(x)]_g \\ &= -\alpha \|[\nabla_x f]_g\|^2 - \gamma \cos(\theta_\gamma) \|[\nabla_x f]_g\| \|[\text{sgn}(x) \cdot [\text{clip}_{q_m}^t(|x|)]_g]\| \\ &< -\eta\alpha \|[\nabla_x f]_g\|^2. \end{aligned} \tag{20}$$

Further, our quantization step size d selection rule (14) guarantees that

$$-\eta\alpha\|[\nabla_x f]_g\|^2 - \gamma d [\nabla_x f]_g^T [\text{sgn}(x) \cdot R(x)]_g < 0. \tag{21}$$

Combining Eq. (20) and Eq. (21) allows us to have that

$$\begin{aligned} [\nabla_x f]_g^T \left[-[\nabla_x f]_g - \gamma [x^Q]_g\right] &= [\nabla_x f]_g^T \left[[s_{\text{clip}}(x)]_g - \gamma \cdot d[\text{sgn}(x) \cdot R(x)]_g\right] \\ &< -\eta\alpha\|[\nabla_x f]_g\|^2 - \gamma d [\nabla_x f]_g^T [\text{sgn}(x) \cdot R(x)]_g < 0, \end{aligned}$$

which completes the proof. □

C. Joint Stage Implementation Details

The update rule in Eq. (14) alone is insufficient to ensure that the bit width constraint in Eq. (7b) is consistently satisfied. To address this issue, we introduce an algorithm, outlined in Algorithm 4, to adaptively adjust the forget rate γ and quantization step size d such that the computed bit width stays within the target bit width range $[b_l, b_u]$. Meanwhile, with the adaptive algorithm in place, the search direction $s(x)$ continues to be a descent direction when stochastic gradient $\hat{\nabla}_x f$ is assumed to be full gradient, as demonstrated in Proposition C.1. In addition, there are three hyperparameters that appear in Eq. (13) and Eq. (14) and they are selected as $\eta = 0.9$, $\xi = 0.999$, and $\epsilon =$1e-8.

Proposition C.1. *Let $\hat{\nabla}_x f$ be the full gradient of function $f(x, d, q_m, t)$ with respect to x. With the Algorithm 4 in place (applied immediately after Line 17 in Algorithm 1), the search direction $s(x)$ is still a descent direction with respect to function f at the point x.*

Proof. Denote the full gradient of function $f(x, d, q_m, t)$ with respect to x as $\nabla_x f$. Let's first consider the following three simple cases. When *clip* $> \epsilon$, the forget rate is equal to 0 and therefore, $s(x) = -\nabla f(x)$, which is a descent direction with respect to function $f(x, d, q_m, t)$ with respect to x at x. When $\cos(\theta_\gamma) \geq 0$ or $\cos(\theta_d) \geq 0$, $s(x)$ is guaranteed to be descent direction with respect to function $f(x, d, t, q_m)$ with respect to x when γ and d are positive values.

Now, it remains to consider the following two cases: $\cos(\theta_\gamma) < 0, clip > \epsilon$ and $\cos(\theta_d) < 0$. As indicated in Eq. (13) and Eq. (14), we have that $s(x)$ is a descent direction with respect to function $f(x, d, q_m, t)$ with respsect to x if when $\cos(\theta_\gamma) < 0$ and *clip* $> \epsilon$,

$$\gamma \in (0, -\frac{\alpha\|[\nabla_x f]_g\|}{\cos(\theta_\gamma)\|[\text{sgn}(x) \cdot \text{clip}_{q_m}^t(|x|)]_g\|}), \tag{22}$$

and when $\cos(\theta_d) < 0$,

$$d \in (0, -\frac{\xi\eta\alpha\|[\nabla_x f]_g\|}{\gamma\cos(\theta_d)\|[\text{sgn}(x) \cdot R(x)]_g\|}). \tag{23}$$

When $\cos(\theta_\gamma) < 0$, we guarantee that with the Algorithm 4 in place, the forget rate γ always lie in the range specified in Eq. (22) since γ either decreases by a factor of β (see Line 4) or remains the same (see Line 6). When $\cos(\theta_d) < 0$, we consider two cases based on if the forget rate decreases. If forget rate decreases (see Line 4), then the range given in Eq. (14) is changed to

$$(0, -\frac{\xi\eta\alpha\|[\nabla_x f]_g\|}{\beta\gamma\cos(\theta_d)\|[\text{sgn}(x) \cdot R(x)]_g\|}) \tag{24}$$

It follows that increasing d by a factor of $\frac{1}{\beta}$ guarantees that d lies within the range Eq. (24). If forget rate remains the same, then d always lie in the range Eq. (23) since d decreases by a factor of β. □

Algorithm 4 Adaptive update rule for γ and d.

```
Inputs. Variables: $\gamma$, $d$, bit width range: $[b_l, b_u]$, $\beta \in (0,1)$, fixed quantization variables: $q_m, t$.
while $\log_2\left(\frac{(q_m)^t}{d}+1\right)+1 \notin [b_l, b_u]$ do
    if $\log_2\left(\frac{(q_m)^t}{d}+1\right)+1 > b_u$ then
        $\gamma \leftarrow \beta\gamma$, $d \leftarrow d/\beta$.
    else if $\log_2\left(\frac{(q_m)^t}{d}+1\right)+1 < b_l$ then
        $\gamma \leftarrow \gamma$, $d \leftarrow \beta d$.
    end if
end while
Outputs. $\gamma$, $d$.
```

D. Numerical Experiment Setup

First, we provide details on how we initialize quantization parameters. For each layer that contain quantization parameters, the exponential $t = 1$ and the maximum of quantization range q_m is set to the layerwise maximum of the weight tensor. For experiments on ResNet20, VGG7, and ResNet50, the quantization step size d is chosen such that the resulting bit width is 32 bits while for Bert, the quantization step size d is selected to achieve a bit width of 8 bits.

Next, we provide details on how we select the optimizer and the learning rate for different experiments. For ResNet20, we use the SGD optimizer and the initial learning rate is set to 1e-1 with StepLR learning rate scheduler. For experiments of VGG7, we use the optimizer ADAM and the learning rate is set to 1e-3 with StepLR learning rate scheduler. For ResNet50, we use the optimizer SGD and the learning rate is set to 1e-1 with StepLR learning rate scheduler. For Bert, we use AdamW with learning rate as constant 3e-5. For all four experiments, the learning rate for quantization parameters is set as constant 1e-4. For details on how we set hyperparameters related to projection stage and pruning stage, one can find them in Tab. 7.

Table 7. Experiment setup for all four experiments. In the following table, the unit for projection steps and pruning steps is the number of epochs. As for Bert, the experiment setups are same under all sparsity ratios (10%, 30%, 50%, 70%).

Model	Sparsity	Total	Projection	Projection	Pruning	Pruning	Bit width	Bit width
	level	epochs	periods B	steps K_b	periods P	steps K_p	reduction b_r	range $[b_l, b_u]$
ResNet20	0.35	350	7	35	5	30	2	[4,16]
VGG7	0.7	200	5	20	10	30	2	[4,16]
ResNet50	0.4,0.5	120	5	5	10	10	2	[4,16]
Bert	0.1,0.3,0.5,0.7	10	4	1	6	6	2	[4,16]

E. Ablation Study

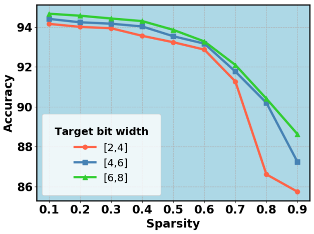
Our proposed QASSO consists of four distinct stages: warm-up stage, projection stage, joint stage, and cool-down stage. To evaluate the contribution of each stage, we conduct an ablation study on two benchmarks, ResNet56 trained from scratch on CIFAR10 and Phi2 fine-tuned from a pre-trained model on the Common-Sense task. The results demonstrate that each stage positively contributes to the model's performance, as measured by test accuracy. As shown in Fig. 4a, removing any of the four stages, especially the joint stage and cool-down stage, results in a noticeable decline in test accuracy. The significance of the joint stage and cool-down stage stems from the fact that a significant knowledge transfer is conducted to retain the information lost when applying pruning and quantization.

Moreover, each stage's contribution varies over downstream applications. For instance, the joint stage plays a more critical role when fine-tuning a pre-trained model compared to training from scratch. This can be attributed to the fact that pre-trained models inherently possess a wealth of useful knowledge, and the joint stage helps preserve performance by effectively transferring this knowledge under quantization constraints.

In addition, we perform an ablation study (See Fig. 4b) using ResNet56 on CIFAR10 benchmark to study the limit of each compression technique within **GETA** framework. As highlighted in [27], structured pruning methods typically achieve sparsity greater than 80%. However, under joint setup, accuracy begins to degrade significantly beyond 60% sparsity. This suggests quantization error constrains aggressive pruning, lowering the achievable sparsity threshold from 80% to 60% for ResNet56-CIFAR10. For quantization, satisfactory accuracy is typically retained with bit width $\geq$ 2bits when sparsity $\leq$ 60%. When sparsity exceeds 60%, model becomes less tolerant to lower bit width, requiring at least 4-bit to retain performance.

Warmup	Projection	Joint	Cool Down	ResNet56 (%)	Phi2 (%)
✓	✓	✓	✓	94.61	58.64
✗	✓	✓	✓	94.11	56.32
✓	✗	✓	✓	94.10	55.17
✓	✓	✗	✓	93.63	52.81
✓	✓	✓	✗	91.32	51.24

(a)



(b)

Figure 4. The Fig. 4a presents an ablation study evaluating the necessity of the four distinct stages of the QASSO optimizer using ResNet56 on the CIFAR10 benchmark and Phi2-2.7B on a common-sense task. The last two columns indicate the model's test accuracy. The Fig. 4b illustrates the limits and boundaries of various compression techniques applied to ResNet56 on the CIFAR10 dataset.

F. Quantization-Aware Dependency Graph

For more intuitive illustration, we present quantization-aware dependency graphs of Bert1 (mini-Bert with one transformer block) and VGG7. Both the original and post-analysis versions of these graphs are shown. To enhance readability of the graph's finer details, we recommend zooming in to a scale of 1500% or higher using Adobe PDF Reader.

Bert.

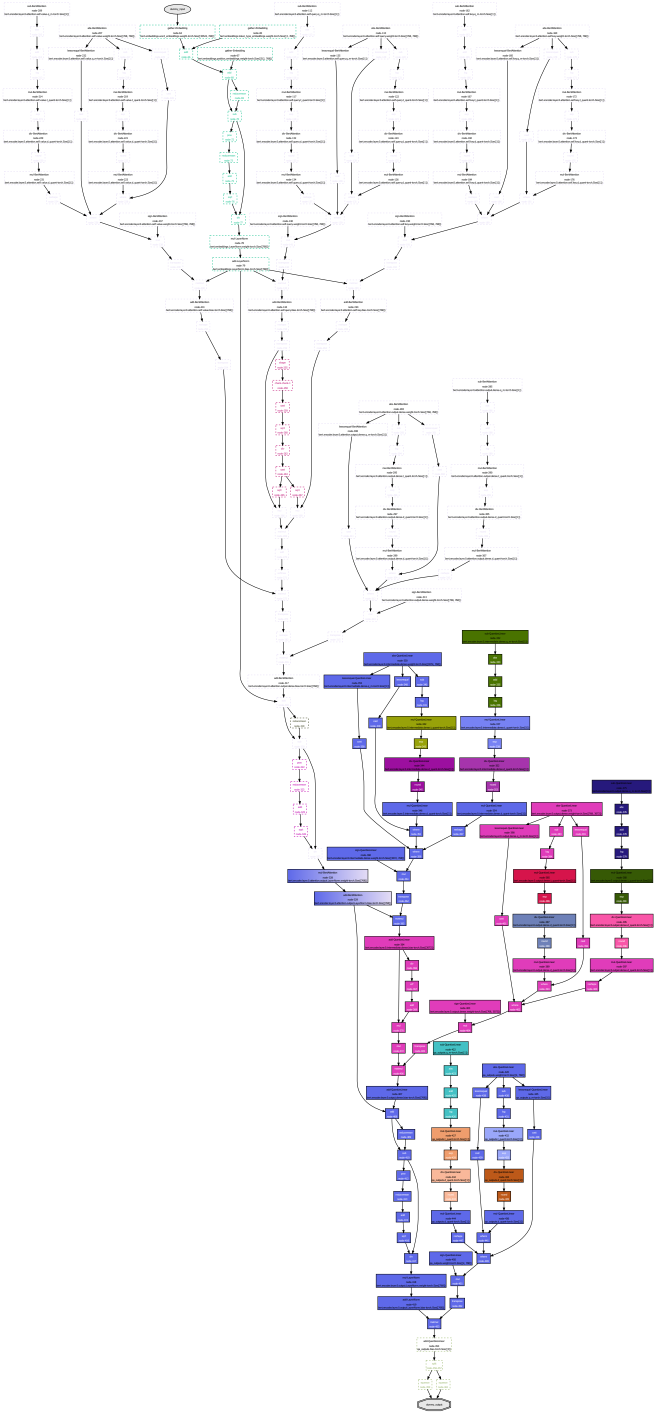
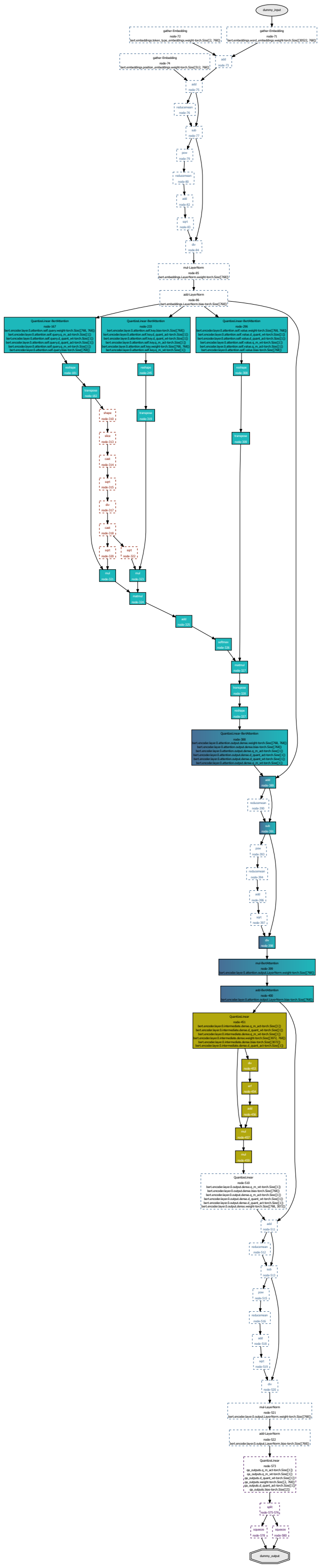


Figure 5. Bert1 before performing quantization-aware dependency graph analysis.

BERT.



dummy_input
gather-Embedding
node-72
bert.embeddings.token_type_embeddings.weight-torch.Size([2, 768])
gather-Embedding
node-71
bert.embeddings.word_embeddings.weight-torch.Size([30522, 768])
gather-Embedding
node-74
bert.embeddings.position_embeddings.weight-torch.Size([512, 768])
add
node-73
add
node-75
reducemean
node-76
sub
node-77
pow
node-79
reducemean
node-80
add
node-82
sqrt
node-83
div
node-84
mul-LayerNorm
node-85
bert.embeddings.LayerNorm.weight-torch.Size([768])
add-LayerNorm
node-86
bert.embeddings.LayerNorm.bias-torch.Size([768])
QuantizeLinear-BertAttention
node-167
bert.encoder.layer.0.attention.self.query.weight-torch.Size([768, 768])
bert.encoder.layer.0.attention.self.query.q_m_act-torch.Size([1])
bert.encoder.layer.0.attention.self.query.d_quant_wt-torch.Size([1])
bert.encoder.layer.0.attention.self.query.d_quant_act-torch.Size([1])
bert.encoder.layer.0.attention.self.query.q_m_wt-torch.Size([1])
bert.encoder.layer.0.attention.self.query.bias-torch.Size([768])
QuantizeLinear-BertAttention
node-233
bert.encoder.layer.0.attention.self.key.bias-torch.Size([768])
bert.encoder.layer.0.attention.self.key.d_quant_act-torch.Size([1])
bert.encoder.layer.0.attention.self.key.d_quant_wt-torch.Size([1])
bert.encoder.layer.0.attention.self.key.q_m_act-torch.Size([1])
bert.encoder.layer.0.attention.self.key.weight-torch.Size([768, 768])
bert.encoder.layer.0.attention.self.key.q_m_wt-torch.Size([1])
QuantizeLinear-BertAttention
node-296
bert.encoder.layer.0.attention.self.value.weight-torch.Size([768, 768])
bert.encoder.layer.0.attention.self.value.d_quant_wt-torch.Size([1])
bert.encoder.layer.0.attention.self.value.d_quant_act-torch.Size([1])
bert.encoder.layer.0.attention.self.value.q_m_wt-torch.Size([1])
bert.encoder.layer.0.attention.self.value.q_m_act-torch.Size([1])
bert.encoder.layer.0.attention.self.value.bias-torch.Size([768])
reshape
node-181
reshape
node-245
reshape
node-308
transpose
node-182
shape
node-310
transpose
node-319
transpose
node-309
slice
node-313
cast
node-314
sqrt
node-315
div
node-317
cast
node-318
sqrt
node-320
sqrt
node-322
mul
node-321
mul
node-323
matmul
node-324
add
node-325
softmax
node-326
matmul
node-327
transpose
node-328
reshape
node-337
QuantizeLinear-BertAttention
node-388
bert.encoder.layer.0.attention.output.dense.weight-torch.Size([768, 768])
bert.encoder.layer.0.attention.output.dense.bias-torch.Size([768])
bert.encoder.layer.0.attention.output.dense.q_m_act-torch.Size([1])
bert.encoder.layer.0.attention.output.dense.d_quant_act-torch.Size([1])
bert.encoder.layer.0.attention.output.dense.d_quant_wt-torch.Size([1])
bert.encoder.layer.0.attention.output.dense.q_m_wt-torch.Size([1])
add
node-389
reducemean
node-390
sub
node-391
pow
node-393
reducemean
node-394
add
node-396
sqrt
node-397
div
node-398
mul-BertAttention
node-399
bert.encoder.layer.0.attention.output.LayerNorm.weight-torch.Size([768])
add-BertAttention
node-400
bert.encoder.layer.0.attention.output.LayerNorm.bias-torch.Size([768])
QuantizeLinear
node-451
bert.encoder.layer.0.intermediate.dense.q_m_act-torch.Size([1])
bert.encoder.layer.0.intermediate.dense.d_quant_wt-torch.Size([1])
bert.encoder.layer.0.intermediate.dense.q_m_wt-torch.Size([1])
bert.encoder.layer.0.intermediate.dense.weight-torch.Size([3072, 768])
bert.encoder.layer.0.intermediate.dense.bias-torch.Size([3072])
bert.encoder.layer.0.intermediate.dense.d_quant_act-torch.Size([1])
div
node-453
erf
node-454
add
node-456
mul
node-457
mul
node-459
QuantizeLinear
node-510
bert.encoder.layer.0.output.dense.q_m_wt-torch.Size([1])
bert.encoder.layer.0.output.dense.bias-torch.Size([768])
bert.encoder.layer.0.output.dense.q_m_act-torch.Size([1])
bert.encoder.layer.0.output.dense.d_quant_wt-torch.Size([1])
bert.encoder.layer.0.output.dense.d_quant_act-torch.Size([1])
bert.encoder.layer.0.output.dense.weight-torch.Size([768, 3072])
add
node-511
reducemean
node-512
sub
node-513
pow
node-515
reducemean
node-516
add
node-518
sqrt
node-519
div
node-520
mul-LayerNorm
node-521
bert.encoder.layer.0.output.LayerNorm.weight-torch.Size([768])
add-LayerNorm
node-522
bert.encoder.layer.0.output.LayerNorm.bias-torch.Size([768])
QuantizeLinear
node-573
qa_outputs.q_m_act-torch.Size([1])
qa_outputs.q_m_wt-torch.Size([1])
qa_outputs.d_quant_wt-torch.Size([1])
qa_outputs.weight-torch.Size([2, 768])
qa_outputs.d_quant_act-torch.Size([1])
qa_outputs.bias-torch.Size([2])
split
node-575-576
squeeze
node-578
squeeze
node-580
dummy_output

Figure 6. Bert1 after performing quantization-aware dependency graph analysis.

VGG7.

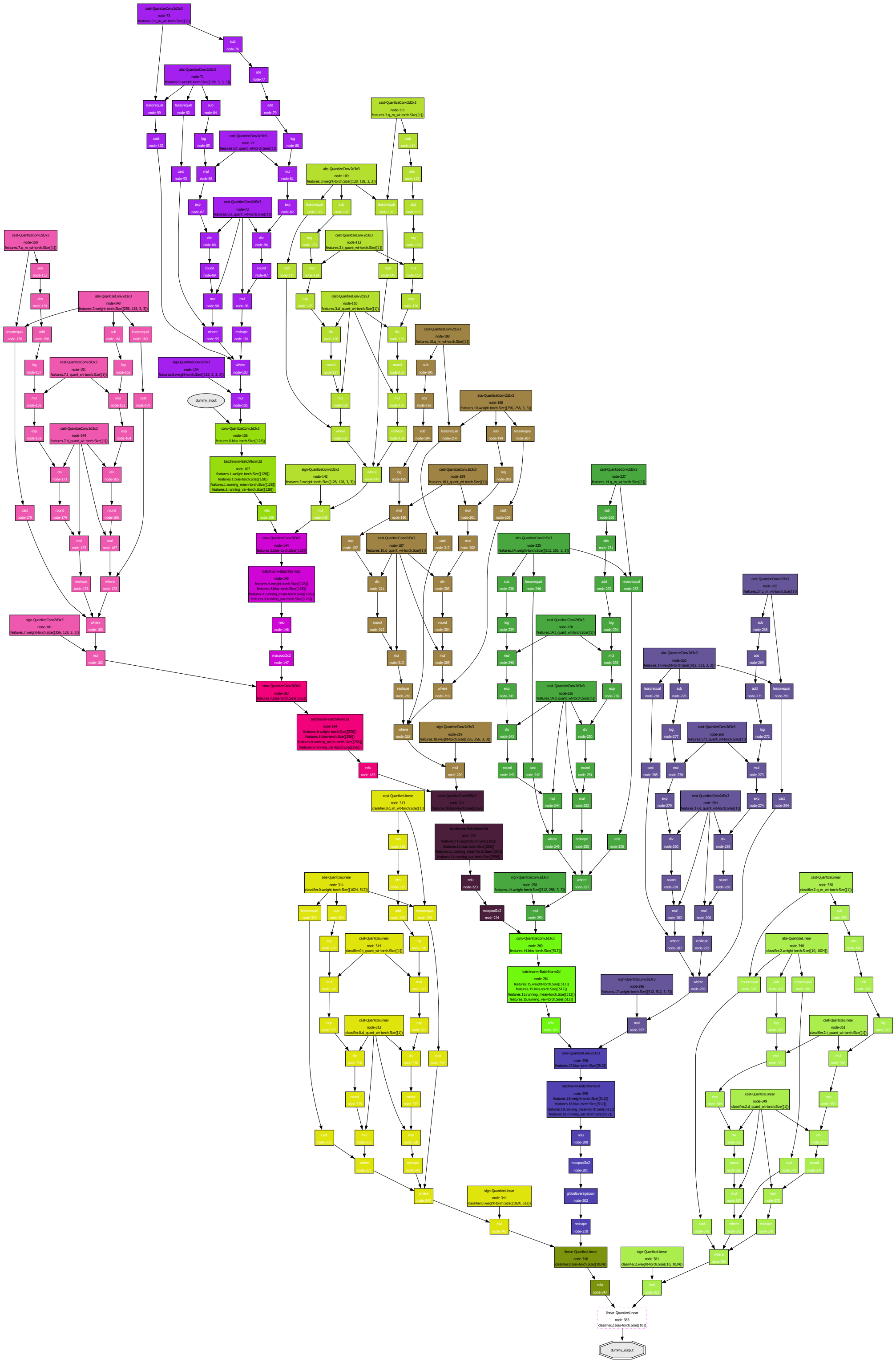


Figure 7. VGG7 after performing quantization-aware dependency graph analysis.

VGG7.

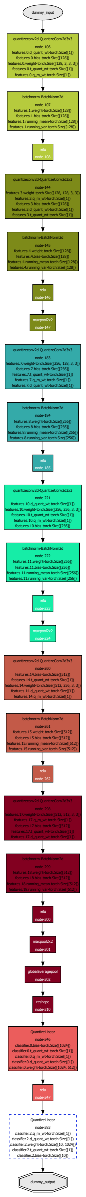

dummy_input
quantizeconv2d-QuantizeConv2d3x3
node-106
features.0.d_quant_wt-torch.Size([1])
features.0.bias-torch.Size([128])
features.0.weight-torch.Size([128, 3, 3, 3])
features.0.t_quant_wt-torch.Size([1])
features.0.q_m_wt-torch.Size([1])
batchnorm-BatchNorm2d
node-107
features.1.weight-torch.Size([128])
features.1.bias-torch.Size([128])
features.1.running_mean-torch.Size([128])
features.1.running_var-torch.Size([128])
relu
node-108
quantizeconv2d-QuantizeConv2d3x3
node-144
features.3.weight-torch.Size([128, 128, 3, 3])
features.3.q_m_wt-torch.Size([1])
features.3.bias-torch.Size([128])
features.3.d_quant_wt-torch.Size([1])
features.3.t_quant_wt-torch.Size([1])
batchnorm-BatchNorm2d
node-145
features.4.weight-torch.Size([128])
features.4.bias-torch.Size([128])
features.4.running_mean-torch.Size([128])
features.4.running_var-torch.Size([128])
relu
node-146
maxpool2x2
node-147
quantizeconv2d-QuantizeConv2d3x3
node-183
features.7.weight-torch.Size([256, 128, 3, 3])
features.7.bias-torch.Size([256])
features.7.t_quant_wt-torch.Size([1])
features.7.q_m_wt-torch.Size([1])
features.7.d_quant_wt-torch.Size([1])
batchnorm-BatchNorm2d
node-184
features.8.weight-torch.Size([256])
features.8.bias-torch.Size([256])
features.8.running_mean-torch.Size([256])
features.8.running_var-torch.Size([256])
relu
node-185
quantizeconv2d-QuantizeConv2d3x3
node-221
features.10.d_quant_wt-torch.Size([1])
features.10.weight-torch.Size([256, 256, 3, 3])
features.10.t_quant_wt-torch.Size([1])
features.10.q_m_wt-torch.Size([1])
features.10.bias-torch.Size([256])
batchnorm-BatchNorm2d
node-222
features.11.weight-torch.Size([256])
features.11.bias-torch.Size([256])
features.11.running_mean-torch.Size([256])
features.11.running_var-torch.Size([256])
relu
node-223
maxpool2x2
node-224
quantizeconv2d-QuantizeConv2d3x3
node-260
features.14.bias-torch.Size([512])
features.14.t_quant_wt-torch.Size([1])
features.14.weight-torch.Size([512, 256, 3, 3])
features.14.d_quant_wt-torch.Size([1])
features.14.q_m_wt-torch.Size([1])
batchnorm-BatchNorm2d
node-261
features.15.weight-torch.Size([512])
features.15.bias-torch.Size([512])
features.15.running_mean-torch.Size([512])
features.15.running_var-torch.Size([512])
relu
node-262
quantizeconv2d-QuantizeConv2d3x3
node-298
features.17.weight-torch.Size([512, 512, 3, 3])
features.17.q_m_wt-torch.Size([1])
features.17.bias-torch.Size([512])
features.17.t_quant_wt-torch.Size([1])
features.17.d_quant_wt-torch.Size([1])
batchnorm-BatchNorm2d
node-299
features.18.weight-torch.Size([512])
features.18.bias-torch.Size([512])
features.18.running_mean-torch.Size([512])
features.18.running_var-torch.Size([512])
relu
node-300
maxpool2x2
node-301
globalaveragepool
node-302
reshape
node-310
QuantizeLinear
node-346
classifier.0.bias-torch.Size([1024])
classifier.0.t_quant_wt-torch.Size([1])
classifier.0.q_m_wt-torch.Size([1])
classifier.0.d_quant_wt-torch.Size([1])
classifier.0.weight-torch.Size([1024, 512])
relu
node-347
QuantizeLinear
node-383
classifier.2.q_m_wt-torch.Size([1])
classifier.2.d_quant_wt-torch.Size([1])
classifier.2.weight-torch.Size([10, 1024])
classifier.2.t_quant_wt-torch.Size([1])
classifier.2.bias-torch.Size([10])
dummy_output

Figure 8. VGG7 after performing quantization-aware dependency graph analysis.