A Appendix

A.1 Quantization kinetics in the continuous time domain

The asymptotic quantization of weights $\boldsymbol{W}$ using BDMM with a Lagrangian function $\mathcal{L}$ follows the discrete updates,

$$\begin{aligned}\boldsymbol{W} &\leftarrow \boldsymbol{W} - \eta_W \nabla_{\boldsymbol{W}} \mathcal{L}(\boldsymbol{W}, \boldsymbol{\lambda}) \\ \boldsymbol{\lambda} &\leftarrow \boldsymbol{\lambda} + \eta_\lambda \nabla_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}),\end{aligned}$$

which can be expressed in the continuous time domain as follows.

$$\frac{d\boldsymbol{W}}{dt} = -\tau_W^{-1} \nabla_{\boldsymbol{W}} \mathcal{L}, \tag{1}$$

and

$$\frac{d\boldsymbol{\lambda}}{dt} = \tau_\lambda^{-1} \nabla_{\boldsymbol{\lambda}} \mathcal{L}, \tag{2}$$

where the reciprocal time constants τ_W^{-1} and τ_λ^{-1} are proportional to learning rates η_W and η_λ, respectively. The Lagrangian function $\mathcal{L}$ is a Lyapunov function of $\boldsymbol{W}$ and $\boldsymbol{\lambda}$.

$$\frac{d\mathcal{L}}{dt} = \nabla_{\boldsymbol{W}} \mathcal{L} \cdot \frac{d\boldsymbol{W}}{dt} + \nabla_{\boldsymbol{\lambda}} \mathcal{L} \cdot \frac{d\boldsymbol{\lambda}}{dt}. \tag{3}$$

Plugging Eqs. (1) and (2) into Eq. (3) yields

$$\frac{d\mathcal{L}}{dt} = -\tau_W^{-1} \left|\nabla_{\boldsymbol{W}} \mathcal{L}\right|^2 + \tau_\lambda^{-1} \left|\nabla_{\boldsymbol{\lambda}} \mathcal{L}\right|^2. \tag{4}$$

The gradients in Eq. (4) can be calculated from the Lagrangian function $\mathcal{L}$, given by

$$\mathcal{L} = C\left(\boldsymbol{y}^{(i)}, \hat{\boldsymbol{y}}^{(i)}; \boldsymbol{W}\right) + \boldsymbol{\lambda}^{\mathrm{T}} \boldsymbol{cs}(\boldsymbol{W}),$$

as follows.

$$\begin{aligned}\left|\nabla_{\boldsymbol{W}} \mathcal{L}\right|^2 &= \sum_{i=0}^{n_w} \left(\frac{\partial C}{\partial w_i} + \lambda_i \frac{\partial cs_i}{\partial w_i}\right)^2, \\ \left|\nabla_{\boldsymbol{\lambda}} \mathcal{L}\right|^2 &= \sum_{i=0}^{n_w} cs_i^2.\end{aligned} \tag{5}$$

Therefore, the following equation holds.

$$\frac{d\mathcal{L}}{dt} = -\tau_W^{-1} \sum_{i=0}^{n_w} \left(\frac{\partial C}{\partial w_i} + \lambda_i \frac{\partial cs_i}{\partial w_i}\right)^2 + \tau_\lambda^{-1} \sum_{i=0}^{n_w} cs_i^2. \tag{6}$$

The Lagrange multiplier λ_i at time t is evaluated using Eq. (2).

$$\lambda_i(t) = \lambda_i(0) + \tau_\lambda^{-1} \int_0^t cs_i dt. \tag{7}$$

A.2 Pseudocode

Algorithm 1: CBP algorithm. N denotes the number of training epochs in aggregate. M denotes the number of mini-batches of the training set $\boldsymbol{Tr}$. The function $minibatch\left(\boldsymbol{Tr}\right)$ samples a mini-batch of training data and their targets from $\boldsymbol{Tr}$. The function $model\left(x, \boldsymbol{W}\right)$ returns the output from the network for a given mini-batch $\boldsymbol{x}$. The function clip($\boldsymbol{W}$) denotes the clipping weight, and η_W and η_λ denote the weight- and multiplier-learning rates, respectively.

```
Result: Updated weight matrix W
Pre-training using conventional backprop;
Initialization such that λ ← 0, p ← 0, g ← 1;
Initial update of λ;
for epoch = 1 to N do
    L_sum ← 0;
    /* Update of weight W                                              */
    for i = 1 to M do
        x^(i), ŷ^(i) ← minibatch(Tr);
        y^(i) ← model(x^(i); W);
        L ← C(ŷ^(i), y^(i); W) + λ^T cs(W; Q, M, g);
        L_sum ← L_sum + L;
        W ← clip(W − η_W ∇_W L);
    end
    /* Update of window variable g and Lagrange multiplier λ           */
    p ← p + 1;
    if L_sum ≥ L_sum^pre or p = p_max then
        g ← g + Δg;
        λ ← λ + η_λ cs(W, g);
        p ← 0;
        L_sum^pre ← L_sum^max;
    else
        L_sum^pre ← L_sum;
    end
end
```

A.3 Quantization kinetics with gradually vanishing unconstrained-weight window

We consider the gradually vanishing unconstrained-weight window in addition to the kinetics of update of weights and lagrange multipliers in Eqs. (1) and (2). Given that the update frequency of the unconstrained-weight window variable g is equal to that of the Lagrange multipliers, its time constant equals τ_λ.

$$\frac{dg}{dt} = \tau_\lambda^{-1} g_0, \tag{8}$$

where $g_0 = 1$ when $g < 10$, and $g_0 = 10$ otherwise. Regarding the Lagrangian function $\mathcal{L}$ as a Lyapunov function of $\boldsymbol{W}$, $\boldsymbol{\lambda}$, and g, Eq. (3) should be modified as follow.

$$\frac{d\mathcal{L}}{dt} = \nabla_{\boldsymbol{W}} \mathcal{L} \cdot \frac{d\boldsymbol{W}}{dt} + \nabla_{\boldsymbol{\lambda}} \mathcal{L} \cdot \frac{d\boldsymbol{\lambda}}{dt} + \frac{\partial \mathcal{L}}{\partial g} \frac{dg}{dt}. \tag{9}$$

Plugging Eqs. (1), (2), and (8) into Eq. (9) yields

$$\frac{d\mathcal{L}}{dt} = -\tau_W^{-1} \left|\nabla_{\boldsymbol{W}} \mathcal{L}\right|^2 + \tau_\lambda^{-1} \left|\nabla_{\boldsymbol{\lambda}} \mathcal{L}\right|^2 + \tau_\lambda^{-1} g_0 \frac{\partial \mathcal{L}}{\partial g}. \tag{10}$$

The gradients in Eq. (10) can be calculated using Eqs. (8), (9), and (10) as follows.

$$\begin{aligned}
\left|\nabla_{\boldsymbol{W}}\mathcal{L}\right|^2 &= \sum_{i=0}^{n_w}\left[\frac{\partial C}{\partial w_i}+\lambda_i\left(ucs_i\frac{\partial Y_i}{\partial w_i}+Y_i\frac{\partial ucs_i}{\partial w_i}\right)\right]^2, && (11)\\
\left|\nabla_{\boldsymbol{\lambda}}\mathcal{L}\right|^2 &= \sum_{i=0}^{n_w}\left(ucs_iY_i\right)^2,\\
\frac{\partial\mathcal{L}}{\partial g} &= \frac{1}{2g^2}\sum_{i=0}^{n_w}\lambda_iY_i\sum_{j=1}^{n_q-1}\left(q_{j+1}-q_j\right)\delta\left(\frac{1}{2g}\left(q_{j+1}-q_j\right)-\left|w_i-m_j+\epsilon\right|\right). && (12)
\end{aligned}$$

Given that $\partial ucs_i/\partial w_i = 0$ holds for any w_i value because of $\epsilon \to 0^+$, $\left|\nabla_{\boldsymbol{W}}\mathcal{L}\right|^2$ is simplified as

$$\left|\nabla_{\boldsymbol{W}}\mathcal{L}\right|^2=\sum_{i=0}^{n_w}\left(\frac{\partial C}{\partial w_i}+\lambda_iucs_i\frac{\partial Y_i}{\partial w_i}\right)^2. \tag{13}$$

The gradient $\partial\mathcal{L}/\partial g$ is non-zero only if a given weight w_i satisfies $\left|w_i-m_j+\epsilon\right|=\frac{1}{2g}\left(q_{j+1}-q_j\right)$ The probability that w_i at a given time satisfies the equality for a given g should be very low. Additionally, regarding the discrete change in g in the actual application of the algorithm, the probability is negligible. Thus, this gradient can be ignored hereafter. Therefore, Eq. (10) can be re-expressed as

$$\frac{d\mathcal{L}}{dt}=-\tau_W^{-1}\sum_{i=0}^{n_w}\left(\frac{\partial C}{\partial w_i}+\lambda_iucs_i\frac{\partial Y_i}{\partial w_i}\right)^2+\tau_\lambda^{-1}\sum_{i=0}^{n_w}\left(ucs_iY_i\right)^2. \tag{14}$$

Distinguishing the weights belonging to the unconstrained-weight window D_{ucs} from the others at a given time t, Eq. (14) can be written by

$$\frac{d\mathcal{L}}{dt}=-\tau_W^{-1}\sum_{w_i\in D_{ucs}}\left(\frac{\partial C}{\partial w_i}\right)^2-\sum_{w_i\notin D_{ucs}}\left[\tau_W^{-1}\left(\frac{\partial C}{\partial w_i}+\lambda_i\frac{\partial Y_i}{\partial w_i}\right)^2-\tau_\lambda^{-1}Y_i^2\right]. \tag{15}$$

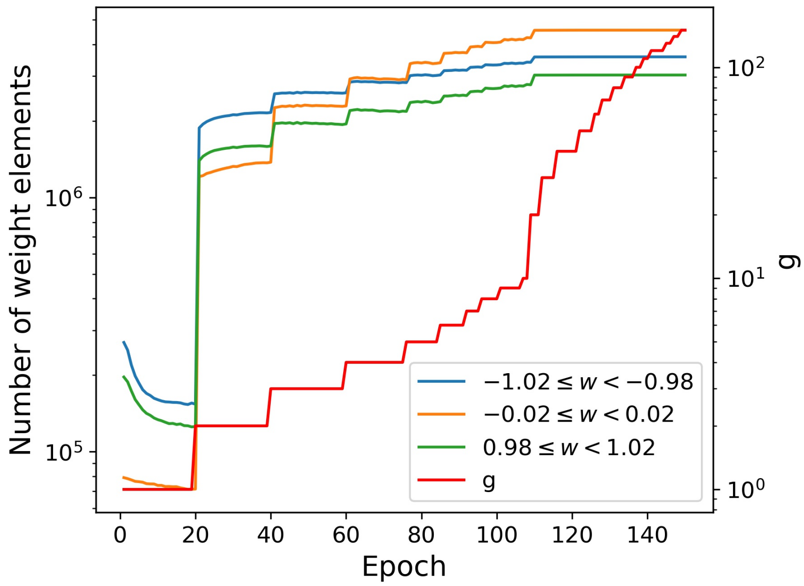

Figure 1: Weight-ternarization kinetics of ResNet-18 on ImageNet

A.4 Quantization kinetics in the discrete time domain

We monitored the population changes of weights near given quantized weight values for ResNet-18 on ImageNet with ternary-weight constraints. Fig. 1 shows the population changes of weights near -1, 0, and 1 upon the update of the unconstrained-weight window variable g. As such, the variable g was updated such that $\Delta g = 1$ when $g < 10$, and $\Delta g = 10$ otherwise. Step-wise increases in populations upon the increase of g are seen, indicating the obvious effect of the unconstrained-weight window on weight-quantization kinetics.

A.5 Hyperparameters

The hyperparameters used are listed in Table 1. The weight- and multiplier-learning rates are denoted by η_W and η_λ, respectively. The weight decay rate (L2 regularization) is denoted by wd.

Table 1: Hyperparameters used.

<table>
<tr><th></th><th colspan="4">AlexNet</th><th colspan="4">ResNet-18</th></tr>
<tr><th></th><th>η_W</th><th>η_λ</th><th>wd</th><th>batch size</th><th>η_W</th><th>η_λ</th><th>wd</th><th>batch size</th></tr>
<tr><td>Binary</td><td rowspan="2">10^{-3}</td><td rowspan="4">10^{-4}</td><td rowspan="4">5×10^{-4}</td><td rowspan="4">256</td><td rowspan="4">10^{-3}</td><td rowspan="4">10^{-4}</td><td rowspan="4">10^{-4}</td><td rowspan="4">256</td></tr>
<tr><td>Ternary</td></tr>
<tr><td>One-bit shift</td><td rowspan="2">10^{-4}</td></tr>
<tr><td>Two-bit shift</td></tr>
<tr><th></th><th colspan="4">ResNet-50</th><th colspan="4">GoogLeNet</th></tr>
<tr><th></th><th>η_W</th><th>η_λ</th><th>wd</th><th>batch size</th><th>η_W</th><th>η_λ</th><th>wd</th><th>batch size</th></tr>
<tr><td>Binary</td><td rowspan="2">10^{-3}</td><td rowspan="4">10^{-4}</td><td rowspan="4">10^{-4}</td><td rowspan="4">128</td><td rowspan="4">10^{-4}</td><td rowspan="4">10^{-4}</td><td rowspan="4">10^{-4}</td><td rowspan="4">256</td></tr>
<tr><td>Ternary</td></tr>
<tr><td>One-bit shift</td><td rowspan="2">10^{-4}</td></tr>
<tr><td>Two-bit shift</td></tr>
</table>

A.6 Computational complexity

CBP is a post-training method so that this number of FLOPs is an additional computational complexity to the pre-training using backprop.

#FLOPs for CBP = (#FLOPs for weight update) + (#FLOPs for Lagrange multiplier update), where

#FLOPs for weight update = (#FLOPs for loss evaluation) + (#FLOPs for error-backpropagation).

#FLOPs for loss evaluation = (#FLOPs for forward propagation) + (#FLOPs for constraint contribution calculation $\boldsymbol{\lambda}^T \mathbf{cs}$).

The number of FLOPs for the latter scales with the number of parameters in total (n_w) because each parameter is given a set of λ and cs. The number of multiplication $\lambda \times cs_i\ (w_i)$ is the same as the number of parameters (n_w).The calculation of cs_i for a given w_i involves six FLOPs according to Eqs. (8)-(10). Therefore,

#FLOPs for loss evaluation = (#FLOPs for forward propagation) + $6n_w$.

As for conventional backprop, the number of FLOPs for weight update (using error-backpropagation) approximately equals the number of FLOPs for forward propagation. Therefore,

#FLOPs for weight update = $2\times$(#FLOPs for forward propagation) + $6n_w$

The Lagrange multiplier update for each multiplier involves one multiplication ($\eta_\lambda \times cs_i$) and one addition ($\lambda_i \leftarrow \lambda_i + \eta_\lambda cs_i$), but uses cs_i that has been calculated already when calculating the loss function. Therefore,

#FLOPs for Lagrange multiplier update = $2n_w$.

It should be noted that the multiplier is updated merely a few times during the entire training period: less than 20 percent of the training epochs, which is parameterized by p.

Therefore, we have

#FLOPs for CBP = 2(#FLOPs for forward propagation) + $2(p+3)n_w$

The number of FLOPs for CBP for three models (for $p = 0.2$) is shown below.

AlexNet: #FLOPs for CBP $\approx$ 1.82G, and #FLOPs for BP $\approx$ 1.45G (i.e., 25% increase in #FLOPs)

ResNet18: #FLOPs for CBP $\approx$ 3.69G, and #FLOPs for BP $\approx$ 3.62G (i.e., 2% increase in #FLOPs)

ResNet50: #FLOPs for CBP $\approx$ 7.89G, and #FLOPs for BP $\approx$ 7.74G (i.e., 2% increase in #FLOPs)

B Additional Data

B.1 Extra Data

Processes of learning quantized weights in AlexNet, ResNet-18, ResNet-50, and GoogLeNet are shown in Fig. 2, 3, 4, and 5, respectively.

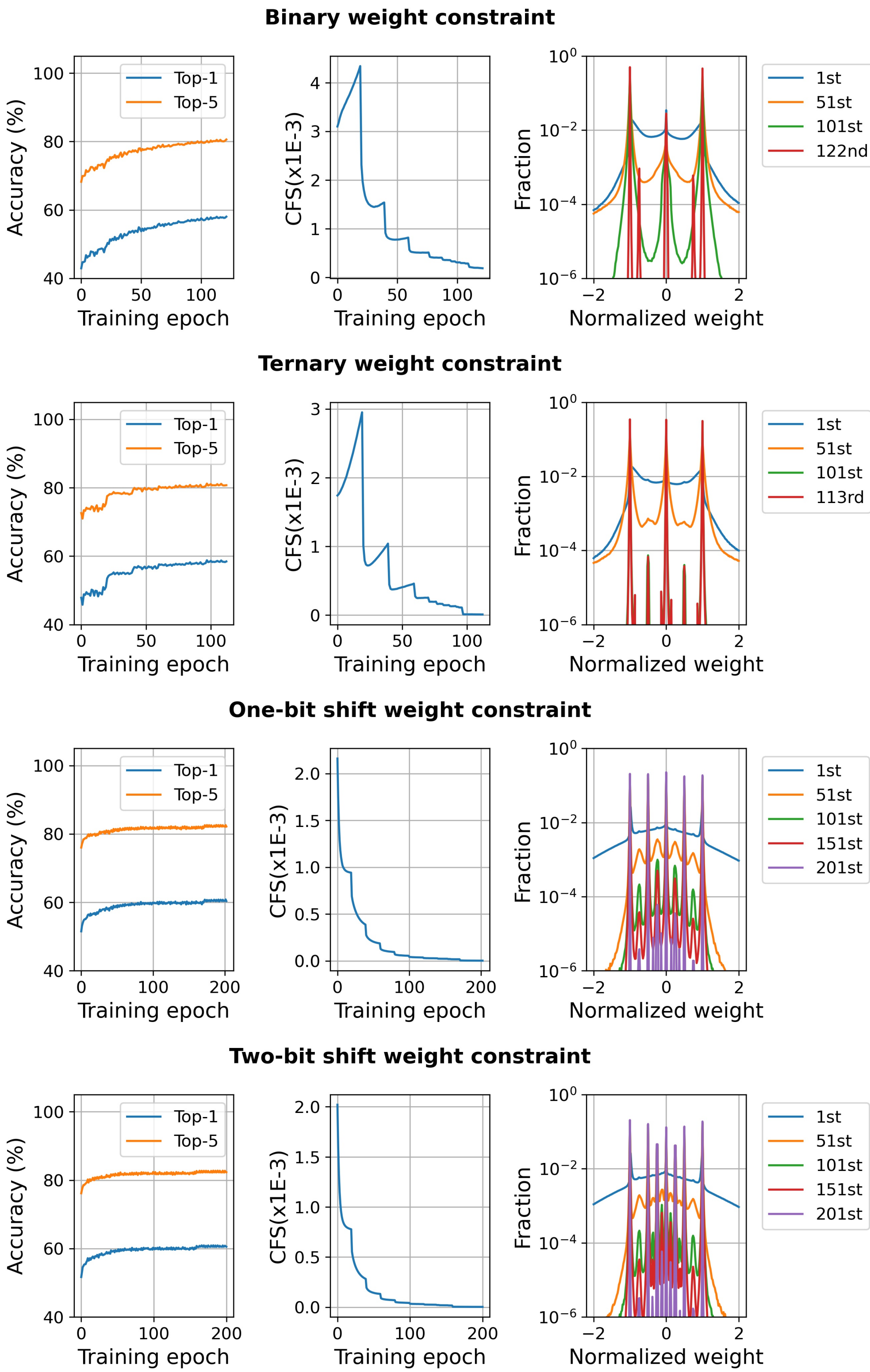

Figure 2: Learning quantized weights in AlexNet

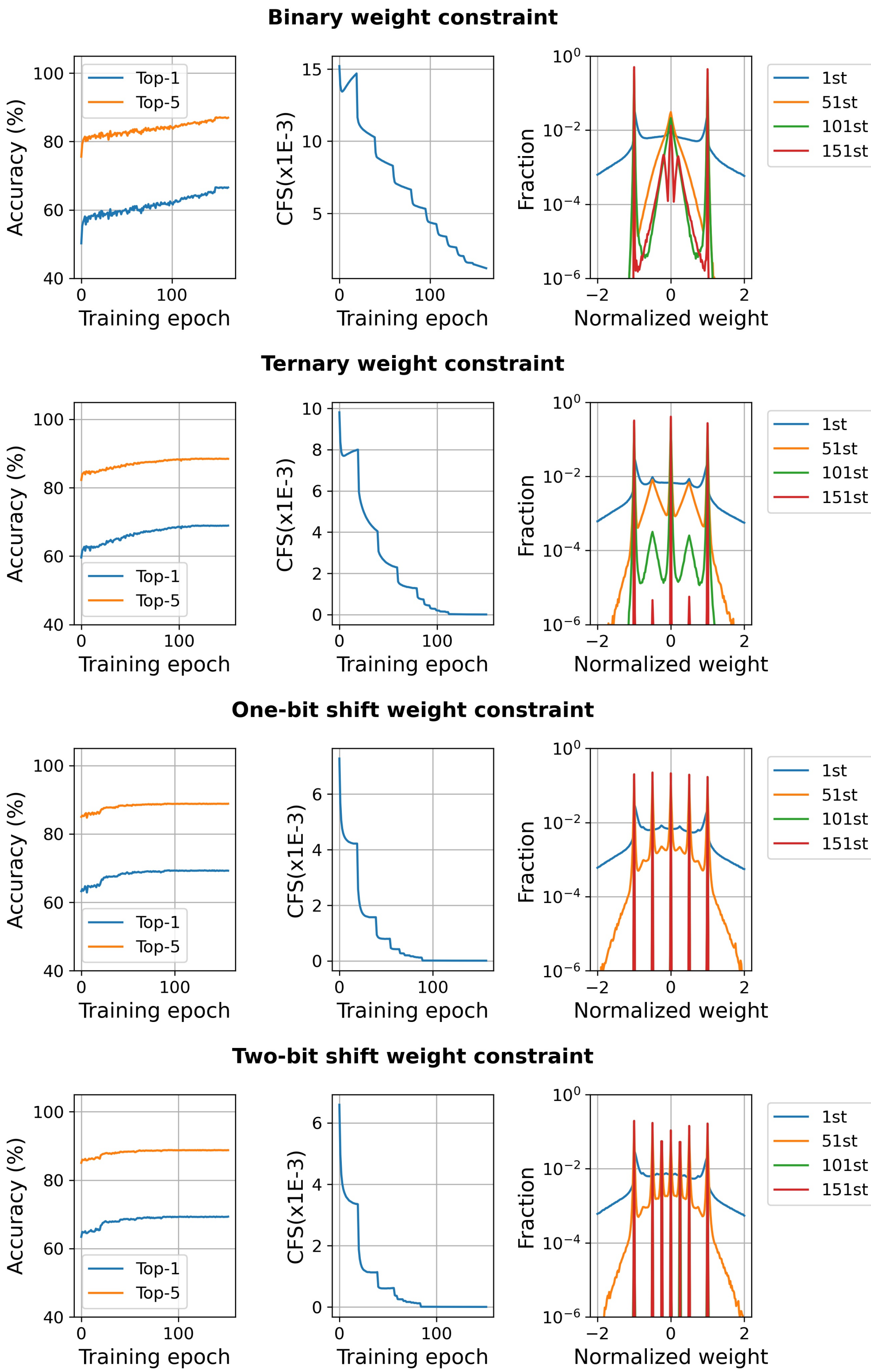

Figure 3: Learning quantized weights in ResNet-18

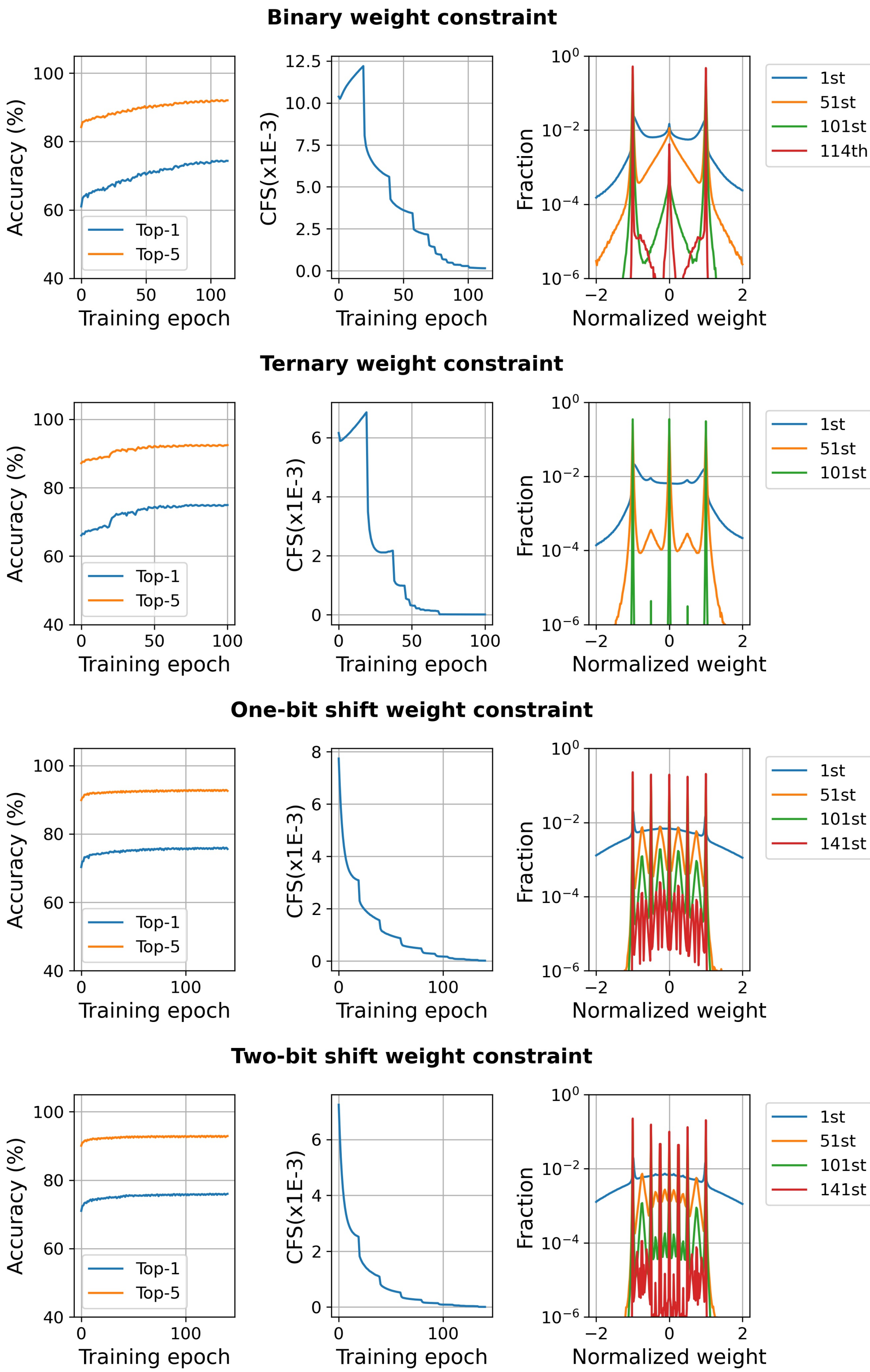

Figure 4: Learning quantized weights in ResNet-50

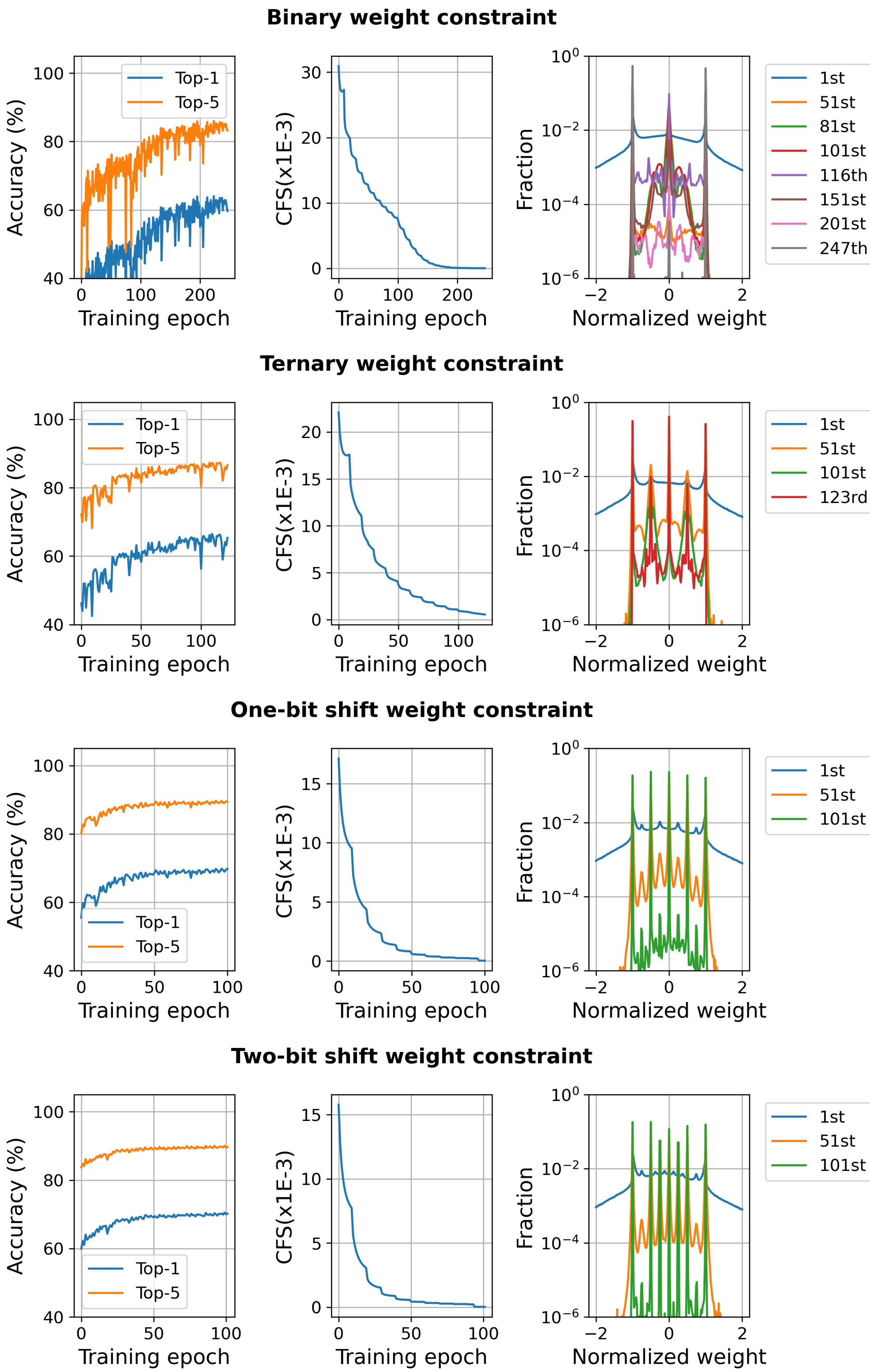

Figure 5: Learning quantized weights in GoogLeNet