A. Supplementary Materials

We introduced our quantization method in Sections 2 and 3 of the main paper. Here, we will cover various sweeps, sensitivity analysis, and ablations that we have performed to more comprehensively study our quantization scheme.

A.1. Calibration Methods

Model calibration refers to the process of determining a good step size (d) and dynamic range (q_+) for the quantizer (see eq. (1) for how we define these terms). Previous studies have shown that not only does the calibration method impact the final quantization outcome, the same method might not be optimal across different bitwidths [9]. Additionally, weights and activations have shown different sensitivities to quantization and dynamic range in computation, notably recent mixed precision work [11] seem to employ higher precision on activations. Consequently, we study how different calibration schemes might impact quantization performance through a coarse grid-search studying the quantization performance on a 3-bit homogeneous EfficientNet-Lite0 followed by quantization aware training for 50 epochs. Our results are summarized in Table 2, which show: (i) calibration method impact for weights vs. activations and (ii) calibration method impact at different bit precisions. The first phase of our training implements a homogeneously quantized model, with the dynamic range and step size being determined through the data of the first batch. We implement the following baselines to determine the clipping value (q_{max}): (i) the maximum value in the data, (ii) $2\times$ the mean of the data, (iii) $\max(\mu + 3\sigma, |\mu - 3\sigma|)$ (Gaussian in table 2), and (iv) calibrating to the X^{th} percentile of the absolute value of the data over a range [9]. The choice of calibration scheme results in a 2.05% improvement over alternatives. The effect of different calibration schemes gets less pronounced at the higher precision (4 bits vs. 3 bits — see Table 3 for 4 bits).

A.2. Comparison of Gradient Scaling Methods

In the main text we showed results for gradient scaling when using different gradient scaling methods for weights and activations, in Figure 7 we show the baseline results when the scaling methods are the same for weights and activations. The mean for STE and EWGS (best method in this experiment) also serve as orientations in Figure 4. We further show the effect of adding noise to the gradient instead of meaningful gradient scaling (Gaussian and uniform noise), both methods boost the accuracy beyond STE accuracy on average. Further, we show the computational overhead measured in wall-clock time compare to the STE method, which does not perform any gradient modification. The methods which employ random noise are the most computationally expensive compared to the STE. Interestingly, PBGS only incurs minimal overhead while the other methods show a substantial increase in compute time. We conjecture that this can be primarily be attributed to the trigonometric operation in the backward pass.

We also analyzed the behaviour of several hand-picked scaling methods on lower and higher bit widths comparable to what we have done in Figure 4. The results can be seen in Figure 8, showing that at 2 bits no method delivered better than 30% accuracy. The differences between these methods are less apparent ($\leq 0.4\%$) at 4 bits.

A.3. Effect of Granularity and Gradient Scaling

We demonstrate the effectiveness of our proposed quantization (see config. j in Figure 8), through ablation trials where we remove gradient scaling, fine-grained quantization, and their combination (base in A.3). Across different models and target budgets, we see that our proposed fine-grained scheme consistently occupies the Pareto, although not all trials perform as well (A.3).

A.4. Assumption for Comparison to Other Work

To facilitate our comparisons in (Figure 1), we computed the memory footprint for weights and activations for different networks, assuming floating point (bfloat16) parameters for batch norm. If the references stated that they did not quantize the first and last layer ([3, 8, 13, 26, 28]), we assume they can use bfloat16 datatypes without any loss of accuracy. Table 4 states our assumptions about networks sizes in terms of how many parameters models have in total, how many of those parameters are for matrix-vector multiplications, how many parameters are batch norm (BN) parameters and how many parameters belong to the first and last layers.

A.5. Latency Considerations

To highlight the possible effects on latency we simulated latency numbers from synthesized single MAC units on a commercially available advanced node. Realistic latency numbers require a comprehensive co-design between the hardware and accelerator, including parameters like technology node, hardware, compiler configurations etc. Note that these are weak approximations and a more comprehensive evaluation will require full architectural simulation. Consider that data movement between memory hierarchies and computing units has a major impact on latency. Minimizing this through quantization will impact the end-to-end latency. For example, consider a hypothetical accelerator with a bandwidth of 128 bits/cycle to transfer data between the global buffer and local register. A convolution layer of size 3x3x128 will require 4x the number of cycles to transfer 16bits (144 cycles) vs. 4 bits (36 cycles). While some amount of this can be hidden through communication-computation overlap, quantization will still reduce the number of cycles for which computing might stall while waiting for additional data. In our simulations each MAC unit can have different bit-width inputs. To obtain a full model latency we added up different

Table 2. Effect of different calibration methods on homogeneous bit-width training (3 bits) for a Efficient-Lite0 on the ImageNet dataset.

W/A	Max	2× Mean	Gaussian	P99.9	P99.99	P99.999	P99.9999
Max	63.85%	63.78%	63.38%	63.24%	63.59%	63.73%	63.54%
2× Mean	64.57%	0.10%	63.02%	64.51%	64.55%	64.72%	64.22%
Gaussian	64.72%	64.73%	64.90%	**65.07%**	64.75%	64.67%	64.79%
P99.9	64.67%	64.87%	64.64%	64.71%	64.64%	64.86%	64.88%
P99.99	64.56%	64.81%	64.44%	64.27%	64.45%	64.33%	64.51%
P99.999	64.29%	64.59%	64.50%	64.38%	63.96%	64.03%	64.25%
P99.9999	64.14%	64.23%	63.78%	63.73%	63.95%	64.34%	64.26%

Table 3. Effect of different calibration methods on homogeneous bit-width training (4 bits) for a Efficient-Lite0 on the ImageNet dataset.

W/A	Max	2× Mean	Gaussian	P99.9	P99.99	P99.999	P99.9999
Max	72.08%	71.96%	71.89%	72.00%	71.87%	71.95%	72.08%
2× Mean	72.25%	0.73%	72.09%	72.22%	72.15%	72.29%	72.30%
Gaussian	72.24%	72.27%	72.15%	72.26%	72.29%	72.29%	72.38%
P99.9	72.29%	72.44%	72.20%	72.31%	72.17%	72.12%	72.44%
P99.99	72.10%	72.10%	72.10%	72.32%	72.21%	72.27%	72.27%
P99.999	72.21%	72.11%	72.03%	71.95%	72.16%	72.25%	72.19%
P99.9999	72.09%	72.01%	72.10%	72.05%	71.97%	72.25%	72.02%

latencies of single MAC units per layer given our trained bit-width. Table 5 shows the results for our frontier results (Efficient-Lite0 and MobileNetV2). The latency results are relative to a 16 bit model.

A.6. ADMM for Heterogenous Quantization

To apply ADMM to quantization outlined in ***[sj: ()]***, we reformulate the problem as:

$$\begin{aligned}\min_{x_1,x_2}\quad & CE(x_1,y)\\ &+\beta\max\left(\left(\sum_{l=1}^{L}\sum_{c=1}^{C}b^w_{lc,x_2}\cdot s^w_{lc}\right)-t^w,0\right)^2\\ &+\beta\max\left(\left(\sum_{l=1}^{L}b^a_{l,x_2}\cdot s^a_{l}\right)-t^a,0\right)^2\\ \text{s.t.}\quad & x_1-x_2=0,\end{aligned}\tag{3}$$

where the notation is the same as in eq. (2). However, due to ADMM's alternating phase formulation, we now have two sets of model weights x_1 and x_2. The first set of model weights are used only for the accuracy optimization (see CE loss) and the second part only for size penalties. Given the constraint, we can form an augmented Lagrangian:

$$\begin{aligned}F(x_1,x_2,y)=\quad & CE(x_1,y)\\ &+\beta\max\left(\left(\sum_{l=1}^{L}\sum_{c=1}^{C}b^w_{lc,x_2}\cdot s^w_{lc}\right)-t^w,0\right)^2\\ &+\beta\max\left(\left(\sum_{l=1}^{L}b^a_{l,x_2}\cdot s^a_{l}\right)-t^a,0\right)^2\\ &+y^T(x_1-x_2)+\frac{\rho}{2}\|x_1-x_2\|_2^2.\end{aligned}\tag{4}$$

To which we can apply the ADMM method consisting of the following steps:

1. $\bar{x_1}=\arg\min_{x_1}F(x_1,x_2,y)$ minimize x_1 while x_2 and y kept constant.
2. $\bar{x_2}=\arg\min_{x_2}F(\bar{x_1},x_2,y)$ minimize x_2 while y and $\bar{x_1}$ kept constant.
3. $\bar{y}=y+\rho h(\bar{x_1},\bar{x_2})$ update multiplier.

ADMM performance is sensitive to the learning rate in steps 1 and 2, and the hyperparameter ρ. To overcome this, we conducted an extensive hyperparameter search using a BBO algorithm [16, 37]. Based on this search. ρ was set to 18.1179 and the learning rate was set to $8.95062e^{-6}$ for an EfficientNet-Lite0 with a budget equivalent to a 4-bit model.

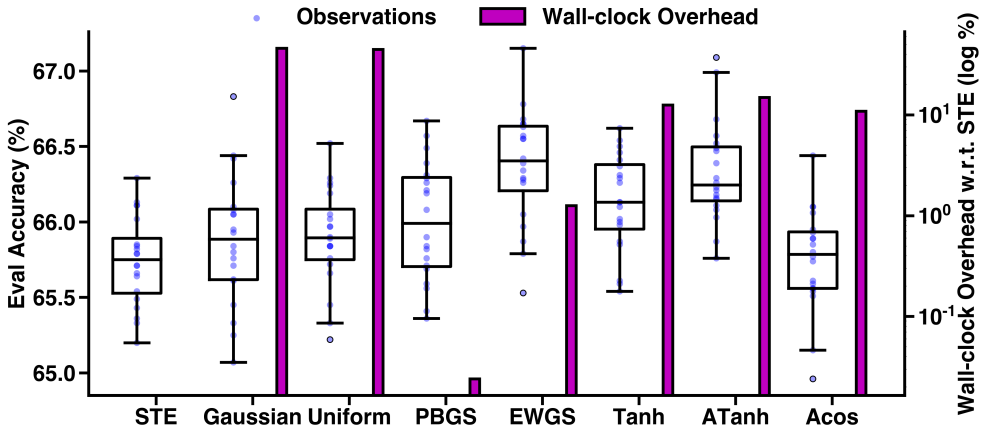

Figure 7. Comparison of gradient scaling methods on a EfficientNet-Lite0 with weights and activation both quantized to 3 bits with a gradient scale factor of 5e-3. STE ([2]) stands for the straight-through-estimator which does not modify the gradient meanwhile Gaussian and Uniform add random noise to the gradient in the backward pass. PBGS ([27]), EWGS ([28]) and Acos ([31]) scale the gradient based on the distance from the quantization point. We added two additional position based scaling methods Tanh and InvTanh. We also display the wall-clock time overhead of gradient scaling method in log percent compared to STE.

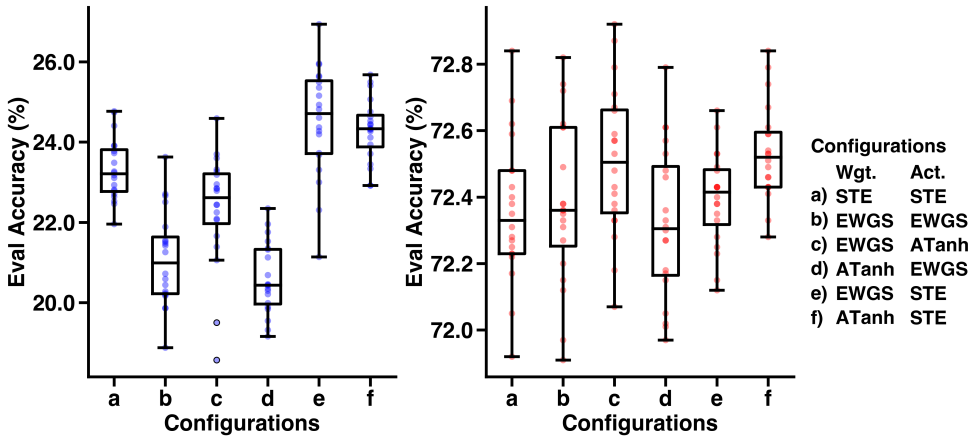

Figure 8. Comparison of selected gradient scaling methods on a homogeneously quantized 2 and 4 bit EfficientNet-Lite0 network. Accuracy numbers drop drastically for 2 bit networks and gradient scaling methods on activations show lower performance compared to the straight-through estimator method.

A.7. Limitations

The main limitation of our work is that to achieve a specific model size we use standard gradient descent optimization with penalties on the loss function which does not guarantee that the model size constraint will be fulfilled. The size constraints are part of the loss function and with a sufficiently large penalty factor (β) the chance of fulfilling the constraint can substantially increase but simultaneously performance might be sacrificed. Furthermore, our method does not guarantee a solution on the efficient frontier. See A.3 in the supplementary materials where we summarize the sensitivity of different elements of our technique across multiple training runs. Typical solutions are within 1% or on the Pareto frontier however some memory constraints prove especially hard to quantize efficiently on some networks, e.g. EfficientLite-0 around 5 MB yields sub-optimal solutions at two different accuracy levels both more than 1% away from the Pareto frontier. Due to convergence limitations, we did not examine techniques for binarization and terenerization which could further improve the memory and energy footprint of these models.

Like related work, we do not quantize batch normalization parameters which contribute significantly to the model size post-quantization. Techniques to "fold" these parameters into the convolution or affine layers have been proposed that could be leveraged to minimize this [24].

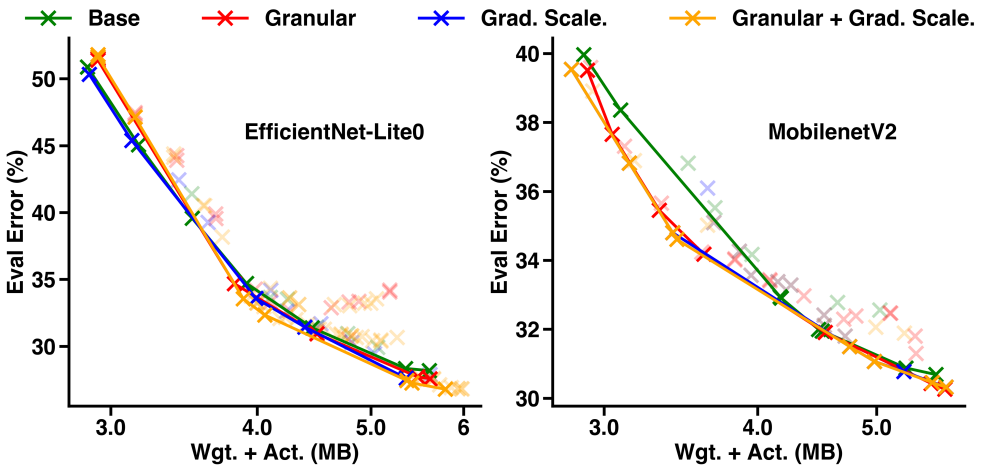

Figure 9. EfficientNet-Lite0 and MobileNetV2 mixed precision results showing all obtained data points, even those which are not Pareto optimal.

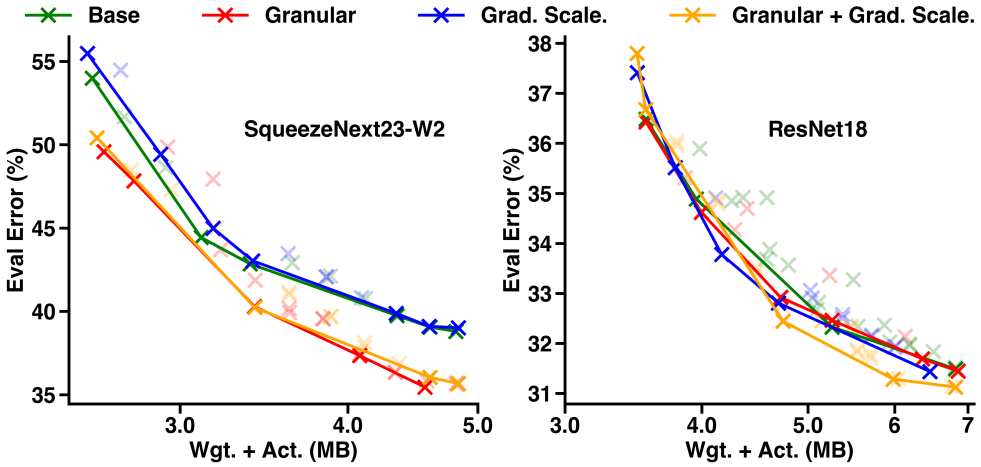

Figure 10. SqueezeNext23-W2 and ResNet18 mixed precision results showing all obtained data points, even those which are not Pareto optimal.

A.8. Societal Impact

Our work describes a general technique to train mixed precision neural networks and demonstrates its performance with image classification on the ImageNet dataset. In principle, our techniques can be applied to other domains as well. Our quantization method trades off accuracy for model size. We have, however, not analyzed what the model forgets when its size constraints are tightened. Other research has recently investigated this issue for model compression [22]. A similar analysis for mixed precision quantization remains open. We foresee the primary impact on society being a reduction in the cost of deploying machine learning systems on edge devices, reducing energy consumption and carbon footprint. However, widespread deployment of ML on such devices could see negative uses, e.g., surveillance. Additional negative impacts might arise due to the need for mixed-precision accelerators which might increase the barrier to entry for deploying such models.

A.9. Effect of Different Training Phases

Table 6 shows Pearson's correlation coefficient of the final accuracy and parameters from the BBO (note we only included runs that actually met our size constraints). The correlation coefficients give insights into the importance of the training phases and settings. For example, the activation penalty has a high correlation coefficient, hinting at its importance in training. For further phase ablation studies, we would like to note that ablating the phase itself (i.e., removing the phase itself) might not be particularly meaningful. In our initial studies, removing phase 1 (homogeneous net-

Table 4. Numbers used to compute effective network sizes for other works.

	ResNet18	MobileNetV2	SqNxt23-W2	ENet-B0	ENet-Lite0
# Total Wgt.	11,689,512	3,504,872	20,670,016	5,288,548	4,652,008
# MVM Wgt.	11,679,912	3,470,760	20,485,184	5,246,532	4,609,992
# First Layer Wgt.	9,408	864	18,816	864	864
# Last Layer Wgt.	513,000	1,281,000	256,000	1,281,000	1,281,000
# BN Wgt.	9,600	34,112	184,832	42,016	42,016
# Total Sum Act.	2,032,640	6,678,112	3,962,208	8,982,784	6,676,256
# Last Layer Act.	512	1,280	256	1,280	1,280

Table 5. Relative simulated latency numbers based on single mixed precision MAC units.

EfficientNet-Lite0									
Size (MB)	22.66	-	3.01	3.23	3.98	4.14	5.45	5.50	5.87
Accuracy (%)	75.53	-	48.37	52.87	66.46	67.66	72.56	72.75	73.21
Norm. Latency (%)	100.00	-	31.04	31.38	33.69	33.86	38.91	38.74	42.16
MobileNetV2									
Size (MB)	20.25	2.89	3.21	3.48	3.51	4.82	5.05	5.62	5.76
Accuracy (%)	71.46	60.72	63.18	65.20	65.39	68.50	68.93	69.54	69.68
Norm. Latency (%)	100.00	33.45	33.06	34.66	35.74	37.38	38.15	43.94	41.63

Table 6. Correlation coefficients of BBO search space parameters to accuracy.

Parameter	Corr. Coef.
Activation Penalty	-0.0756
Weight Penalty	-0.0567
Penalty Ramp-up Length	-0.0491
Update Frequency	+0.0408

work training) resulted in near-random accuracy. Removing phase 2 removes the bit-width learning mechanism, thereby preventing adaptation to size targets. Removing phase 3 prevents performance recovery from finetuning, which generally yielded a 2-3% improvement in final accuracy.