

---

# Efficient Multi-bit Quantization Network Training via Weight Bias Correction and Bit-wise Coreset Sampling

---

Jinhee Kim<sup>1,2\*</sup> Jae Jun An<sup>1\*</sup> Kang Eun Jeon<sup>1,3†</sup> Jong Hwan Ko<sup>1†</sup>

<sup>1</sup> Department of Electrical and Computer Engineering, Sungkyunkwan University

<sup>2</sup> Department of Electrical and Computer Engineering, Duke University

<sup>3</sup> Kim Jaechul Graduate School of AI, Korea Advanced Institute of Science and Technology (KAIST)

{a2jinhee, ajj8061, kejeon, jhko}@skku.edu

## Abstract

Multi-bit quantization networks enable flexible deployment of deep neural networks by supporting multiple precision levels within a single model. However, existing approaches suffer from significant training overhead as full-dataset updates are repeated for each supported bit-width, resulting in a cost that scales linearly with the number of precisions. Additionally, extra fine-tuning stages are often required to support additional or intermediate precision options, further compounding the overall training burden. To address this issue, we propose two techniques that greatly reduce the training overhead without compromising model utility: (i) *Weight bias correction* enables shared batch normalization and eliminates the need for fine-tuning by neutralizing quantization-induced bias across bit-widths and aligning activation distributions; and (ii) *Bit-wise coreset sampling strategy* allows each child model to train on a compact, informative subset selected via gradient-based importance scores by exploiting the implicit knowledge transfer phenomenon. Experiments on CIFAR-10/100, TinyImageNet, and ImageNet-1K with both ResNet and ViT architectures demonstrate that our method achieves competitive or superior accuracy while reducing training time up to 7.88×. Our code is released at this link.

## 1 Introduction

With the explosion of highly capable yet computationally demanding deep learning models, quantization has emerged as an effective strategy for balancing performance and efficiency [1, 2, 3, 4]. Despite its advantages, most existing quantization methods are optimized for a single fixed quantization precision configuration, which limits their ability to adapt dynamically to changing resource availability and deployment across various platforms of diverse memory, compute, and power specifications. This has led to a line of recent work focused on training a single model capable of supporting multiple precisions [5, 6, 7, 8], thereby enabling instant adaptation to varying resource budgets at runtime without the need for further training. In such multi-bit quantization networks, henceforth *multi-bit networks*, a single full-precision parent model generates multiple reduced-precision child models, thereby neutralizing the overhead of maintaining separate models for inference. By supporting multiple quantization precisions- referred to as the model’s *switchable bit range*, these networks enable adaptive deployment across a wide range of compute-constrained devices [9, 10, 11, 12].

---

\*Equal contribution.

†Corresponding authors.

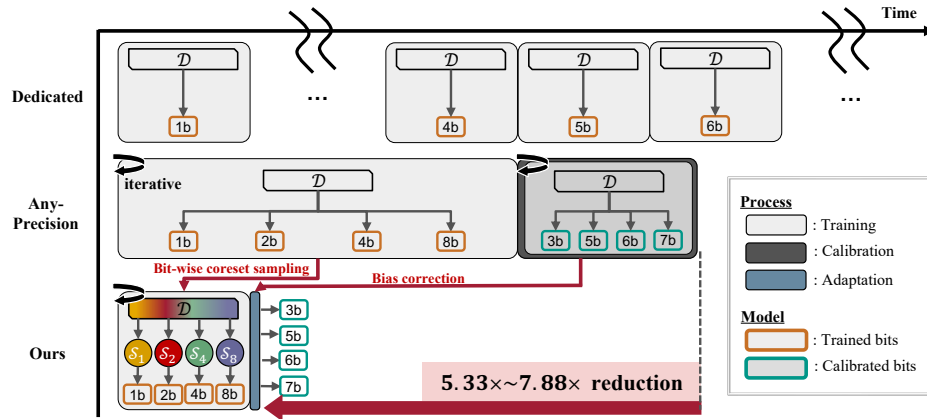


Figure 1: A conceptual diagram of i) Dedicated, ii) Any-Precision, and iii) our training pipelines.  $\mathcal{D}$  and  $\mathcal{S}$  indicate the full training dataset and coreset, respectively.

Although the multi-bit networks provide flexibility during inference/deployment, this advantage comes at the cost of substantial training overhead, limiting their practical adoption. The prevalent multi-bit training approach, known as *Any-Precision* [13], jointly optimizes the model across a small subset of selected bit-widths, termed the *trained bit range*. While this approach is more efficient than training individual models across the switchable range (*Dedicated*), it still introduces considerable overhead due to the additional calibration required to enable inference at bit-widths outside the trained range (*calibrated bit range*), as shown in Fig. 1. Specifically, calibration demands extensive computation using large amounts of training data to preserve the accuracy of untrained bit models in the calibrated range, by aligning their mismatched activation distributions. We identify that these activation mismatches across different bit-width models stem from biases in the weight distribution induced by quantization. Based on this observation, we propose a novel bias correction technique that directly controls the shift and scaling biases in the quantized weights to align distributions across the entire switchable bit-widths. This alignment enables multiple bit-width sub-networks to share a common set of batch normalization (BN) parameters, effectively eliminating the need for costly post-training calibration.

Another major source of the significant training overhead is the use of the entire dataset for updating models in the trained bit range. Although coreset selection methods have been introduced to reduce the training overhead by identifying a subset of important data samples [14, 15, 16, 17], these approaches have primarily targeted single-precision model training with fixed coresets. Extending this idea to a multi-bit quantization setting, we observe that each bit-width child model can benefit from training on distinct and smaller data subsets due to implicit gradient alignment across bit-widths. Leveraging this insight, we propose a bit-wise coreset sampling method that dynamically selects informative samples individually for each child model, based on the gradients computed per bit-width. Furthermore, since sample importance changes throughout training, we periodically re-sample these coresets to reflect evolving model dynamics. The proposed sampling approach effectively reduces per-epoch computational costs while preserving strong performance through implicit cross-bit-width knowledge transfer, a phenomenon we discover for the first time.

To summarize, our contributions are as follows:

- **Weight bias correction for activation alignment:** We correct quantization-induced biases in the weight space instead of the activation space, enabling multiple child models to share normalization parameters, and in turn eliminating the need for an extra training stage.
- **Bit-wise coreset sampling:** We propose a novel per-bit-width coreset sampling strategy that computes bit-wise importance scores using gradient-based methods, thereby reducing training redundancy in multi-bit quantization networks.
- **Extensive empirical validation:** We demonstrate that our method consistently improves or maintains accuracy while significantly reducing training cost across diverse datasets (e.g., CIFAR-10, CIFAR-100, TinyImageNet, and ImageNet-1K) and architectures (e.g., ResNet, DeiT, Swin). Our method achieves up to 7.88× GPU hour reduction without sacrificing model utility, validating the scalability and generality of our approach.

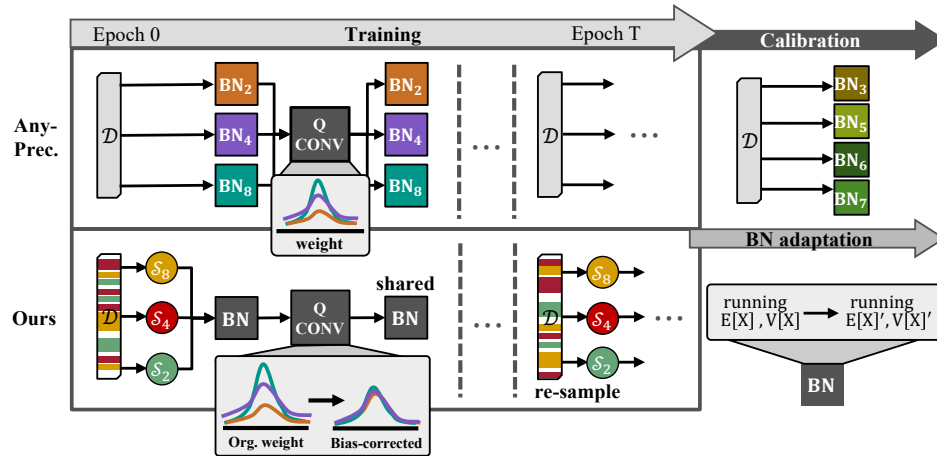


Figure 2: A summary of Any-Precision's and our training pipeline.

## 2 Backgrounds & Related Works

**Multi-bit quantization networks.** Unlike traditional quantized networks that are optimized for a single reduced numerical precision, multi-bit quantization networks [13, 5, 7, 18, 8, 19] are capable of supporting multiple quantization precisions, enabling adaptive and versatile inference/deployment across a wide spectrum of compute-constrained devices [9, 10, 11, 12]. The mainstream approach to training these networks involves optimizing the model for multiple precisions simultaneously, typically by the sum of loss functions corresponding to each bit-width. Formally, this is stated as shown below:

$$\min_{\theta} \sum_{(\mathbf{x}, y) \in \mathcal{S}} \sum_{b \in \mathcal{B}} \mathcal{L}(\mathbf{x}, y, Q(\theta, b)), \quad (1)$$

where  $\theta \in \mathbb{R}^d$  denotes the learnable model parameter which is shared across multiple precisions;  $\mathcal{L}(\mathbf{x}, y, Q(\theta, b))$  is the loss on training sample  $(\mathbf{x}, y)$  in training set  $\mathcal{S}$ ;  $Q(\theta, b) \in \mathbb{Z}^d$  is the quantized version of  $\theta$  at  $b$ -bit precision; and  $\mathcal{B}$ , referred to as the **trained range**, represents the set of all **trained bit-widths**. To perform this optimization in practice, the batch-wise training scheme, which interleaves parameter updates across child models corresponding to different bit-widths in a batch-wise manner (see Algorithm 1), is commonly adopted to promote generalization across the entire training range.

**Training overhead of multi-bit networks.** While multi-bit training is generally more efficient than training multiple single-precision networks independently, it still incurs significant computational overhead—particularly as the training range  $\mathcal{B}$  expands. To mitigate this cost, recent approaches minimize the number of bit-widths included in the training range and instead introduce **calibrated bit-widths/range** to expand precision support. Specifically, the model is first trained on a small set of bit-widths (the trained range), after which a large portion of its parameters are frozen. The remaining parameters are then calibrated or lightly fine-tuned to support additional bit-widths (the calibrated range). The union of the trained and calibrated ranges defines the model's **switchable range**,  $\mathcal{R}$ —i.e., the full set of bit-widths supported by the multi-bit network.

### Algorithm 1 Batch-wise training scheme

**Input:** Data  $\mathbf{X}$ , label  $\mathbf{Y}$

**Output:** Multi-bit network  $G$

```

1: for epoch = 1, ..., T do
2:   for batch from  $\mathbf{X}, \mathbf{Y}$  do
3:     for bit  $b$  in  $\mathcal{B}$  do
4:       Set all layers in  $G$  to  $b$ -bit
5:       Compute forward pass of  $G$ 
6:       Calculate gradients of  $G$ 
7:     end for
8:     Update parameters with  $\sum_{b \in \mathcal{B}} \mathcal{L}_b$ 
9:   end for
10: end for
```

**Challenges in training multi-bit networks.** One major challenge in multi-bit network training research is the accuracy degradation due to activation distribution mismatch between different bit-widths. To address this mismatch, Any-Precision [13], along with CoQuant [18] and MBQuant [8], leverages the 'switchable batch normalization' approach first proposed by [20]. While effective, assigning separate batch normalization layers to each bit-width incurs additional overhead: obtaining

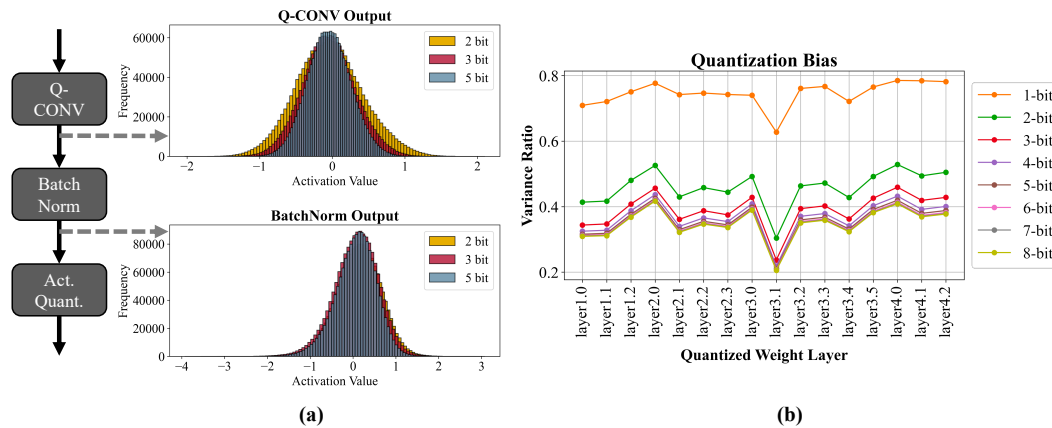


Figure 3: (a) Mismatch in activation distributions between different bit-widths, and (b) variance ratio between quantized weights and original weights in ResNet-50.

these parameters for unseen bit-widths typically requires an extra training phase as shown in Fig. 2. Some recent works [7, 5, 21] avoid this mechanism altogether, but often suffer from degraded performance at lower bit-widths or resort to computationally expensive strategies to reduce interference between the training objectives of different child models. In our proposed method, rather than relying on multiple batch normalization layers or other costly techniques, we correct the weight distribution directly before and after quantization. Our key insight is that quantization introduces bit-width-specific shifts and scaling in the weight distribution, and aligning these distributions helps reduce conflicts across child models during training.

**Coreset selection.** Coreset selection—also known as dataset pruning—aims to reduce model training cost by selecting a small yet representative subset of the training data, while preserving model utility. The core challenge lies in accurately identifying the most informative samples. Feature-space based methods select subsets that preserve the geometry of the data distribution—for example, Herding [22] and Moderate [15] select data points with distance in feature space. Uncertainty-based methods prioritize ambiguous or hard-to-classify samples; for example, Entropy [23] and Cal [24] select samples near decision boundaries. Gradient-based methods leverage training loss gradients. GraND/EL2N [14] rank samples by their gradient magnitude (or prediction error), while Craig [25] and GradMatch [26] select subsets that best match/mimic the full dataset’s gradient signals. Training-dynamics based methods consider samples’ behavior over many epochs. Forgetting [16] counts how often a sample is forgotten during training, and AUM [27] averages the confidence gap across all epochs. Finally, hybrid approaches fuse multiple criteria: TDDS [17] integrates gradient information with training dynamics by measuring each sample’s variability in its epoch-wise contribution to the overall training gradient.

**Shortcomings of existing coreset selection research.** Despite this breadth of approaches, most coreset selection methods are investigated under the assumption of fixed, full-precision floating-point models, and their applicability to quantized neural networks remains largely unexplored. In particular, the integration of coreset selection into quantization-aware training (QAT) has received little attention, let alone its extension to the more complex setting of *multi-bit quantization*, where cross-bit interactions can significantly affect saliency estimation and the underlying training dynamics.

### 3 Weight Bias Correction for Activation Alignment

**Activation distribution mismatch in multi-bit networks.** As discussed in Section 2, multi-bit networks often suffer from mismatched activation distributions across bit-widths. To isolate the source of this mismatch, we decompose the post-convolutional activation into two components: the quantized input activation and the quantized weight. To simplify the analysis, we fix the input activation to a specific precision (e.g., 4-bit). Under this setup, any observed variation in the output can be attributed solely to the quantized weights, thereby reducing the problem to a single source of quantization noise.

In Fig. 3(a), we visualize the post-convolutional activations of ResNet-50 for bit-widths  $b \in \mathcal{B} = \{2, 3, 5\}$  using a batch of ImageNet examples. Although the input activation is fixed, the output distributions vary noticeably across bit-widths. This indicates that the differences can be attributed to quantization-induced bias in the weights. Fig.3(b) supports this explanation by showing that the quantized weights exhibit clear scale distortions compared to the original weights. This observation is consistent with prior observations of [2, 28], which highlight the presence of systematic bias introduced during quantization.

Many multi-bit networks [13, 18, 8] address this activation mismatch problem by training separate BN parameters for each bit-width to independently correct activation distributions. As illustrated in Fig. 3(a), this approach successfully aligns BN outputs across different precisions. While effective, aligning output activations typically requires access to the training data and additional forward/backward passes, which incurs additional training overhead.

**Bias correction for quantized weights.** Instead of rectifying the output activations, we address the bias at its source by aligning the quantized weights prior to convolution. By doing this, we can match activation outputs across bit-widths just by correcting the weights during the initial training stage, without having to explicitly match the activations themselves. As a result, BN layers can be shared across all bit-widths, as shown in Fig. 2, avoiding the additional overhead of calibrating separate BN layers. It is important to note that this correction is performed under a fixed activation bit-width (e.g., 4 bits), meaning that aligning the weights directly translates to more consistent activation outputs across different bit-widths. Specifically, we adjust the quantized weight vector  $\mathbf{w}_q$  with respect to their full-precision counterpart  $\mathbf{w}$ , and compute the corrected weights  $\mathbf{w}_q'$  as follows:

$$\mathbf{w}_q' = \sqrt{\frac{\mathbb{V}[\mathbf{w}]}{\mathbb{V}[\mathbf{w}_q]}} (\mathbf{w}_q + (\mathbb{E}[\mathbf{w}] - \mathbb{E}[\mathbf{w}_q])), \quad (2)$$

where  $\mathbb{E}[\cdot]$  and  $\mathbb{V}[\cdot]$  denote the expectation and variance, respectively. This weight alignment enables multiple child models to share a single set of BN parameters with minimal interference. To compensate for residual discrepancies not fully addressed by bias correction, we additionally apply BN adaptation [29]. While adjusting running statistics has been proven to be effective in fixed-quantized networks [30], its use in multi-bit networks [5] remains limited and often lacks clarity on when it is applied (e.g., applied at every epoch in [5], which is unnecessary). Applying BN adaptation once at the final training stage as shown in Fig. 2, is sufficient to correct the running mean and variance for each bit-width, achieving optimal performance without additional overhead.

## 4 Bit-Wise Coreset Sampling

To translate the benefits of coreset selection into multi-bit quantization networks, we propose two techniques tailored to this setting: (i) a coreset sampling strategy that accounts for variations in sample importance across bit-widths and training epochs; and (ii) a bit-wise training scheme for accurate per-bit-width importance score evaluation. Together, these techniques enable more efficient and adaptive training across a range of quantization levels while maintaining strong model utility.

### 4.1 Coreset sampling strategy

The central idea behind our coreset sampling method is to dynamically redraw training subsets along two axes: bit-width and training time. Rather than using a static, global coreset, we select bit-wise coresets that evolve throughout training via sampling as shown in Fig. 2. This design is motivated by two key observations: (i) gradient alignment across bit-widths, and (ii) temporal drift in sample importance.

**Observation 1 – Gradient alignment across bit-widths.** We find that gradients computed from different bit-widths using the same data sample are highly aligned. In Fig. 4, we visualize the angles between the gradients of the 8-bit and 2-bit child models across several layers of ResNet-20 at various training epochs. It can be seen that the angle between the two gradients stays consistently below  $28^\circ$ , with alignment improving in deeper layers. This implies that, without loss of generality, parameter updates based on 2-bit gradients positively influence 8-bit child model (and also that of other precisions in the trained range), and vice versa. We refer to this phenomenon as *cross-bit-width implicit knowledge transfer*, where shared parameters act as conduits for the transfer of learning signals between child models.

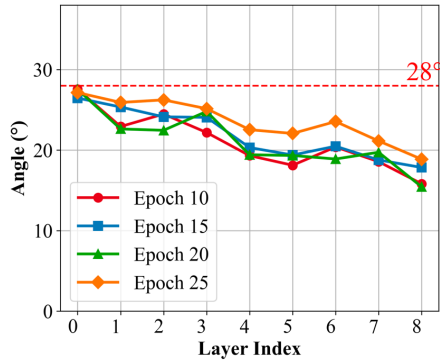


Figure 4: Angle between the 8-bit and 2-bit gradients across layers.

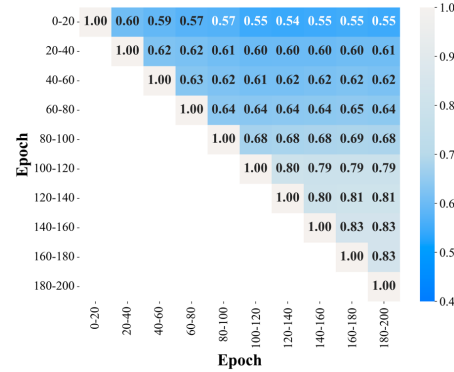


Figure 5: Spearman correlation between ranks at different epochs.

This observation leads to a key insight: *it is unnecessary to feed the same sample to all bit-widths during training*. Accordingly, we construct bit-wise coresets—separate subsets tailored for each bit-width. These bit-wise coresets not only reflect the variation in sample importance across bit-widths but also exploit the implicit gradient transfer phenomena to benefit from a collective learning signal without accessing the full dataset. Therefore, this design significantly reduces per-epoch computational cost while preserving strong performance across the trained range.

**Observation 2 – Temporal drift in sample importance.** We also observe that a sample’s importance evolves as training progresses. More specifically, samples influential in early epochs often become less relevant in later stages, where the model nears convergence and the loss landscape flattens. In Figure 5, we visualize this effect using Spearman correlation of TDDS-based importance scores measured at different training stages on ResNet-18 trained with CIFAR-100 dataset. Correlations between early and late-stage scores may drop as low as 0.54, indicating substantial shifts in sample influence over time.

To account for the temporal drift in sample importance, we periodically re-sample each bit-wise coreset throughout training. Although high-score samples may be informative in the early stages, they often lose relevance as the model’s learning dynamics evolve. Continually refreshing the coresets helps prevent overfitting to outdated importance estimates, especially critical when the true sample importance landscape is dynamic and only partially observable.

**Sampling method.** To construct the bit-wise coresets via sampling, we first convert the sample importance scores into *sampling probabilities* by applying min-max normalization. We then further shape the sampling probabilities using temperature-based sampling [31, 32, 33], which simultaneously reduces overfitting to high-scoring samples and promotes diversity, effectively balancing exploitation and exploration. The sampling probability  $p_i^{(b)}$  for sample  $i$  at bit-width  $b$  is defined as:

$$p_i^{(b)}(\tau) = \frac{(s_i^{(b)})^{1/\tau}}{\sum_{j \in \mathcal{S}} (s_j^{(b)})^{1/\tau}}, \quad (3)$$

where  $s_i^{(b)}$  denotes the min-max normalized importance score for sample  $i$  at bit-width  $b$ ; and  $\tau > 0$  denotes the temperature parameter. Note that importance scores are computed once prior to coreset sampling and remain fixed throughout training.

## 4.2 Bit-wise training scheme for score evaluation

Extracting accurate, bit-wise importance scores is particularly challenging in the context of training dynamics-based coreset selection methods. These mainstream approaches estimate sample importance over multiple training epochs to capture the intricate training dynamics and improve score reliability (see Section 2 for an overview). A representative example is TDDS [17], which accumulates intermediate gradient signals—referred to as context vectors—throughout training to capture the evolving contribution of each sample. While effective in single-precision settings, we find that applying such methods directly under the standard batch-wise training scheme (Algorithm 1) fails to produce meaningful bit-wise importance estimates. The core issue lies in the interleaved update

Table 1: ResNet on CIFAR-10 and 100. Pruning rate of *Coreset Sampling* is 80%.

| Dataset   | Framework | Coreset Sampling | Test Accuracy    |                  |                  |                  |                  | GPU hours (Speed up) |
|-----------|-----------|------------------|------------------|------------------|------------------|------------------|------------------|----------------------|
|           |           |                  | 1bit             | 2bit             | 4bit             | 8bit             | 32bit            |                      |
| CIFAR-10  | Dedicated | -                | 92.42            | 93.04            | 92.99            | 93.08            | 94.11            | 93.10                |
|           | Any-Prec. | -                | 92.85 $\pm 0.21$ | 93.28 $\pm 0.17$ | 93.61 $\pm 0.07$ | 93.64 $\pm 0.04$ | 93.77 $\pm 0.12$ | 93.31                |
|           | Ours      | -                | 93.11 $\pm 0.07$ | 93.46 $\pm 0.11$ | 93.57 $\pm 0.08$ | 93.53 $\pm 0.05$ | 93.60 $\pm 0.05$ | 93.46                |
|           |           | ✓                | 92.60 $\pm 0.14$ | 93.01 $\pm 0.10$ | 93.03 $\pm 0.04$ | 93.00 $\pm 0.13$ | 93.08 $\pm 0.12$ | 92.97                |
| CIFAR-100 | Dedicated | -                | 67.52            | 70.21            | 70.17            | 70.50            | 72.63            | 70.21                |
|           | Any-Prec. | -                | 70.54 $\pm 0.31$ | 71.54 $\pm 0.27$ | 71.60 $\pm 0.27$ | 71.58 $\pm 0.39$ | 72.23 $\pm 0.29$ | 71.47                |
|           | Ours      | -                | 70.95 $\pm 0.09$ | 71.92 $\pm 0.17$ | 71.96 $\pm 0.11$ | 71.91 $\pm 0.09$ | 71.91 $\pm 0.08$ | 71.84                |
|           |           | ✓                | 69.14 $\pm 0.08$ | 70.12 $\pm 0.11$ | 70.35 $\pm 0.17$ | 70.43 $\pm 0.11$ | 70.41 $\pm 0.11$ | 70.26                |

Table 2: ResNet on CIFAR-10. Comparison against previous methods at 80% pruning rate.

| Method     | Test Accuracy    |                  |                  |                  |                  |       |
|------------|------------------|------------------|------------------|------------------|------------------|-------|
|            | 1bit             | 2bit             | 4bit             | 8bit             | 32bit            | Avg.  |
| Random     | 88.97 $\pm 0.47$ | 89.99 $\pm 0.22$ | 90.21 $\pm 0.40$ | 90.12 $\pm 0.28$ | 89.68 $\pm 0.32$ | 89.94 |
| Entropy    | 85.62 $\pm 0.04$ | 86.20 $\pm 0.39$ | 86.31 $\pm 0.24$ | 86.25 $\pm 0.22$ | 86.36 $\pm 0.15$ | 86.21 |
| Forgetting | 76.57 $\pm 1.10$ | 78.18 $\pm 0.97$ | 78.46 $\pm 0.99$ | 78.36 $\pm 1.00$ | 78.49 $\pm 0.90$ | 78.14 |
| EL2N       | 80.21 $\pm 0.29$ | 81.16 $\pm 0.14$ | 81.21 $\pm 0.20$ | 81.15 $\pm 0.43$ | 81.22 $\pm 0.12$ | 81.07 |
| Moderate   | 87.63 $\pm 0.22$ | 88.18 $\pm 0.18$ | 88.35 $\pm 0.11$ | 88.43 $\pm 0.21$ | 88.27 $\pm 0.08$ | 88.26 |
| TDDS       | 87.67 $\pm 0.55$ | 88.35 $\pm 0.05$ | 88.72 $\pm 0.09$ | 88.75 $\pm 0.07$ | 88.57 $\pm 0.19$ | 88.54 |
| Ours       | 92.60 $\pm 0.14$ | 93.01 $\pm 0.10$ | 93.03 $\pm 0.04$ | 93.00 $\pm 0.13$ | 93.08 $\pm 0.12$ | 92.97 |

pattern: gradients from all bit-widths are aggregated before a shared parameter update, resulting in a single unified context vector that masks the distinct training dynamics of each child model.

To address this issue, we introduce a bit-wise training scheme for score evaluation, as shown in Algorithm 2. In this setup, each child model corresponding to a trained bit-width is trained on the entire dataset before proceeding to the next bit-width. This scheduling isolates the gradient updates for each precision, enabling the computation of distinct context vectors and more accurate, bit-wise importance scores.

It is important to note that the proposed bit-wise training scheme is used *exclusively* for importance score extraction. Once the scores are computed, we revert to the standard batch-wise training scheme for actual multi-bit network training. This hybrid approach allows coreset construction to benefit from accurate, bit-wise decoupled importance evaluation while preserving the generalization advantages of batch-wise training.

#### Algorithm 2 Bit-wise training scheme

**Input:** Data  $\mathbf{X}$ , label  $\mathbf{Y}$

**Output:** Multi-bit network  $G$

```

1: for epoch = 1, ..., T do
2:   for bit  $b$  in  $\mathcal{B}$  do
3:     for batch from  $\mathbf{X}$ ,  $\mathbf{Y}$  do
4:       Set all layers in  $G$  to  $b$ -bit
5:       Compute forward pass of  $G$ 
6:       Calculate gradients of  $G$ 
7:       Update parameters with  $\mathcal{L}_b$ 
8:     end for
9:   end for
10: end for

```

## 5 Experiments

### 5.1 Setup

**Evaluation metrics and baselines.** We evaluate our method in terms of per-bit-width accuracy and total GPU hours. Comparisons are made against: (1) the dedicated framework, (2) the standard multi-bit framework (e.g., Any-Precision [13]), and (3) our method which augments the standard framework with Bias Correction (*Ours*). Within *Ours*, we evaluate our proposed *coreset sampling* strategy, which uses *bit-wise scores*, against six baseline coreset selection methods: Random, Entropy [23], Forgetting [16], EL2N [14], Moderate [15], and TDDS [17]. Since most existing coreset selection techniques are designed for dedicated training, we adapt each baseline to our multi-bit framework to ensure a fair comparison.

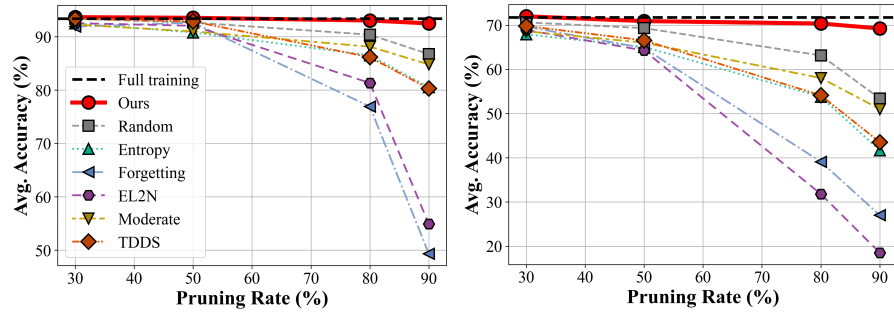


Figure 6: Accuracy comparison across different pruning rates. *Left*: CIFAR-10, *Right*: CIFAR-100

Table 3: ResNet on ImageNet-1K. Pruning rate is 80%.

| Framework | Coreset Sampling | Test Accuracy |       |       |       |       |       | GPU hours (Speed up) |
|-----------|------------------|---------------|-------|-------|-------|-------|-------|----------------------|
|           |                  | 1bit          | 2bit  | 4bit  | 8bit  | 32bit | Avg.  |                      |
| Dedicated | -                | 57.93         | 68.74 | 74.12 | 74.96 | 75.95 | 72.04 | 39.91 (1.00×)        |
| Any-Prec. | -                | 68.77         | 71.66 | 73.84 | 74.07 | 74.63 | 73.01 | 33.94 (1.18×)        |
| Ours      | -                | 68.12         | 72.34 | 73.97 | 74.20 | 74.32 | 73.22 | 27.21 (1.47×)        |
|           | ✓                | 67.22         | 71.93 | 73.29 | 73.40 | 73.92 | 72.51 | 6.94 (5.75×)         |

Table 4: Accuracy comparison against previous methods at 80% pruning.

| Method     | Test Accuracy |       |       |       |       |       |
|------------|---------------|-------|-------|-------|-------|-------|
|            | 1bit          | 2bit  | 4bit  | 8bit  | 32bit | Avg.  |
| Random     | 66.13         | 70.68 | 72.07 | 72.05 | 72.97 | 71.36 |
| Entropy    | 65.32         | 69.28 | 70.66 | 70.77 | 71.68 | 70.03 |
| Forgetting | 55.79         | 60.70 | 64.87 | 65.48 | 66.78 | 63.73 |
| EL2N       | 63.19         | 68.17 | 69.64 | 69.75 | 70.72 | 68.90 |
| Moderate   | 64.57         | 68.88 | 70.39 | 70.63 | 71.58 | 69.74 |
| TDDS       | 65.24         | 69.21 | 70.56 | 70.71 | 71.36 | 69.42 |
| Ours       | 67.22         | 71.93 | 73.29 | 73.40 | 73.92 | 72.51 |

**Datasets and networks.** We evaluate our method on four canonical datasets—CIFAR-10, CIFAR-100 [34], TinyImageNet [35], and ImageNet-1K [36]—with a diverse set of networks. These include three ResNet models: PreActResNet-20 [37], ResNet-18, and ResNet-50 [38], as well as three Vision Transformers (ViTs): DeiT-T, DeiT-S [39], and Swin-T [40].

**Implementation details.** All experiments are conducted on a single NVIDIA A100 GPU. Each experimental setting is as follows: (i) *Dedicated* trains the model with a single weight and activation bit-width. (ii) *Any-Precision* uses a training range of  $\mathcal{B} = 1, 2, 4, 8, 32$  for ResNet models (and  $\mathcal{B} = 2, 8$  for ViTs). After training, independent parameters for the remaining bit-widths are calibrated for approximately one-third of the training epochs to ensure convergence. (iii) *Bias Correction* adopts the same training range as (ii) but skips the calibration phase and instead performs BN adaptation. Specifically, we assign a separate BN layer for 1-bit and share BN layers for all other bit-widths. Although inference for the calibrated range is supported and achieves accuracy comparable to the trained range in both (ii) and (iii), those results are omitted here due to the page limit and are provided in the Appendix.

## 5.2 Results

**ResNet on CIFAR-10 and 100.** Table 1 presents the results on PreActResNet-20 for CIFAR-10 and ResNet-18 for CIFAR-100, highlighting both performance and training time reduction achieved by our method. Compared to existing baselines, *Ours* achieves competitive performance with reduced training time by eliminating the calibration phase, while *coreset sampling* further improves efficiency by reducing data usage without compromising accuracy. Additional results for CIFAR-10 with PreActResNet-20, including comparisons between our coreset sampling method and six baselines (see Section 5.1), are presented in Table 2. Random selection has been shown to excel at high pruning rates in prior studies [17], and we observe the same trend in our experimental setup. Our method shows consistent improvements in both accuracy and efficiency across bit-widths. Figure 6 shows the trade-off between training cost and accuracy by plotting average accuracy against pruning rate. Our method consistently outperforms all baselines across the entire pruning spectrum and maintains high accuracy even at a 90% pruning rate.

**ResNet on ImageNet-1K.** We further demonstrate the effectiveness of our method for a bigger dataset like ImageNet-1K. Table 3 and Table 4 summarize our results for ResNet-50 on ImageNet-1K with respect to the baseline and previous methods. For these experiments, we finetune from a pretrained Any-Precision model, where specific implementation details are provided in the Appendix.

Table 5: DeiT-T, DeiT-S, Swin-T on ImageNet-1K for different pruning rates.

| Network | Framework | Pruning Rate | Test Accuracy |       |       |       |       |       |       |              | GPU hours  |         |
|---------|-----------|--------------|---------------|-------|-------|-------|-------|-------|-------|--------------|------------|---------|
|         |           |              | 2             | 3     | 4     | 5     | 6     | 7     | 8     | Avg.         | (Speed up) |         |
| DeiT-T  | Any-Prec. | -            | -             | 69.72 | 69.97 | 70.59 | 70.77 | 70.86 | 70.91 | 70.47        | 25.77      | (1.00×) |
|         | Ours      | 50%          | -             | 69.30 | 69.42 | 69.93 | 70.15 | 70.22 | 70.16 | <b>69.86</b> | 10.00      | (2.58×) |
|         |           | 60%          | -             | 69.03 | 68.87 | 69.65 | 69.99 | 69.93 | 70.00 | <b>69.58</b> | 7.55       | (3.41×) |
| DeiT-S  | Any-Prec. | -            | 76.34         | 76.93 | 78.19 | 78.25 | 78.30 | 78.32 | 78.37 | 77.81        | 27.07      | (1.00×) |
|         | Ours      | 50%          | 76.05         | 76.45 | 78.04 | 78.21 | 78.16 | 78.18 | 78.15 | <b>77.61</b> | 13.33      | (2.03×) |
|         |           | 60%          | 76.22         | 76.59 | 78.05 | 78.25 | 78.36 | 78.27 | 78.31 | <b>77.72</b> | 11.67      | (2.32×) |
| Swin-T  | Any-Prec. | -            | 78.68         | 79.14 | 79.86 | 79.97 | 79.96 | 79.94 | 79.96 | 79.64        | 27.10      | (1.00×) |
|         | Ours      | 50%          | 78.49         | 78.90 | 79.79 | 79.92 | 79.88 | 79.93 | 79.96 | <b>79.55</b> | 13.27      | (2.04×) |
|         |           | 60%          | 78.53         | 78.92 | 79.67 | 79.88 | 79.85 | 79.94 | 79.90 | <b>79.53</b> | 10.57      | (2.56×) |

Table 6: The effect of Bias Correction and BN Adaptation.

| Dataset   | Bias Correction | BN Adaption | Test Accuracy |       |       |       |       |       |       |       |       |              |
|-----------|-----------------|-------------|---------------|-------|-------|-------|-------|-------|-------|-------|-------|--------------|
|           |                 |             | 1bit          | 2bit  | 3bit  | 4bit  | 5bit  | 6bit  | 7bit  | 8bit  | 32bit | Avg.         |
| CIFAR-10  | -               | -           | 92.95         | 87.72 | 93.53 | 93.32 | 92.73 | 92.47 | 92.08 | 91.87 | 93.53 | <b>92.24</b> |
|           | ✓               | -           | 93.58         | 91.98 | 93.50 | 93.68 | 93.62 | 93.51 | 93.47 | 93.40 | 93.70 | <b>93.38</b> |
|           | -               | ✓           | 92.87         | 93.36 | 93.59 | 93.55 | 93.56 | 93.52 | 93.56 | 93.63 | 93.65 | <b>93.48</b> |
|           | ✓               | ✓           | 93.61         | 93.72 | 93.88 | 93.89 | 93.92 | 93.84 | 93.88 | 93.83 | 93.92 | <b>93.83</b> |
| CIFAR-100 | -               | -           | 70.23         | 53.18 | 70.48 | 70.88 | 69.52 | 68.19 | 67.25 | 66.88 | 70.21 | <b>67.42</b> |
|           | ✓               | -           | 71.12         | 69.03 | 71.58 | 72.03 | 71.63 | 71.39 | 71.23 | 71.10 | 71.83 | <b>71.22</b> |
|           | -               | ✓           | 70.36         | 70.95 | 71.53 | 71.56 | 71.40 | 71.46 | 71.52 | 71.47 | 71.45 | <b>71.30</b> |
|           | ✓               | ✓           | 71.37         | 72.10 | 72.31 | 72.37 | 72.27 | 72.34 | 72.38 | 72.33 | 72.26 | <b>72.19</b> |

Compared to the dedicated training setting, our method substantially reduces training time by  $5.75\times$  with minimal impact on performance.

**ViTs on ImageNet-1K.** We also evaluate our method on larger transformer-based models to demonstrate the generality and scalability of our method to other architectures. The results of three different ViTs (i.e., DeiT-T, DeiT-S, and Swin-T) are summarized in Table 5. To the best of our knowledge, there is not yet a standard multi-bit framework such as Any-Precision for vision transformers. To this end, we implement our own framework with similar configurations as Any-Precision. We compare our method with two dataset pruning ratios- 50% and 60% respectively. With a bigger dataset, our method shows even more significant reduction in training time (as large as 18.22 GPU hours reduction in DeiT-T), while showing consistent accuracy compared to the baselines.

### 5.3 Ablation

**Effect of Bias Correction and BN Adaptation.** As shown in Table 6, we conduct an ablation study to quantify the individual and combined effects of Bias Correction and BN adaptation in the final training stage. The results show that both components play distinct yet complementary roles in achieving stable alignment and high accuracy across bit widths. Bias Correction primarily compensates for systematic deviations introduced during quantization, restoring the representational balance in the weight space. However, since it does not modify the batch normalization statistics used at inference, it alone cannot fully align the activation distributions. BN adaptation, applied at the final stage, addresses this limitation by recalibrating the running mean and variance through a small number of forward passes, thereby aligning post-quantization activations with their floating-point counterparts. Together, these two procedures act on different levels of the model—weight and activation—resulting in consistent improvements across all bit widths. Quantitatively, the combination yields the highest accuracy, confirming that the final BN adaptation provides additional activation alignment beyond what bias correction alone can achieve. These findings clarify the respective contributions of both components and highlight the importance of performing BN adaptation at the last training stage for precise activation calibration in quantized models.

Table 7: The effect of the bit-wise training scheme.

| Pruning | CIFAR-10   |                        | CIFAR-100  |                        |
|---------|------------|------------------------|------------|------------------------|
|         | Batch-wise | Bit-wise               | Batch-wise | Bit-wise               |
| 70%     | 90.89      | 91.17 ( <b>0.29↑</b> ) | 61.13      | 64.06 ( <b>2.93↑</b> ) |
| 80%     | 88.55      | 88.91 ( <b>0.36↑</b> ) | 54.23      | 59.51 ( <b>5.28↑</b> ) |
| 90%     | 80.27      | 82.79 ( <b>2.52↑</b> ) | 43.52      | 48.87 ( <b>5.35↑</b> ) |

**Effect of bit-wise schedule for score extraction.** To quantify the benefit of isolating per-bit-width dynamics, we perform a fixed-coreset ablation under the multi-bit framework (i.e., Any-Precision [13]) with Bias Correction setting. We select the dataset once—either by (i) the conventional batch-wise 1 TDDS scores [17] or by (ii) our bit-wise 2 scores. We then train the full multi-bit schedule on these reduced sets. As reported in Table 7, bit-wise scoring yields higher accuracy at every pruning ratio on both datasets. This improvement stems from our bit-wise extraction design 2, which enables the collection of separate intermediate gradients per bit-width at each epoch—allowing us to compute distinct context vectors for every sub-model.

## 6 Conclusion

In this work, we introduce two techniques to reduce the training overhead of multi-bit quantization networks. First, we correct quantization-induced bias in the weight space, removing the need for an additional training stage. Second, we design a bit-wise coreset sampling strategy that leverages implicit knowledge transfer, allowing each child model to train on a compact, informative subset selected via gradient-based importance scores. Our approach preserves model utility while reducing training costs across various architectures such as ResNets and ViTs, offering a scalable solution for efficient multi-bit quantization training. By enabling more efficient multi-precision learning, our method contributes to the broader goal of sustainable and energy-efficient AI, helping make high-performance models more accessible, affordable, and ubiquitous to everyone.

Despite strong empirical results, our evaluations are limited to computer vision tasks due to the high computational cost of training multi-bit networks—a challenge shared by every prior work in this space. By significantly reducing this overhead, our method paves the way for applying multi-bit quantization to more demanding applications such as generative AI and large-scale language tasks. Extending our approach to these domains will be the focus of our future work, advancing the broader applicability and impact of multi-bit quantization networks across diverse tasks.

## 7 Acknowledgment

We thank the anonymous reviewers for their constructive comments. This work was partly supported by the National Research Foundation of Korea (NRF) grant (RS-2024-00345732, RS-2025-02216217); the Institute for Information & communications Technology Planning & Evaluation (IITP) grants (RS-2020-II201821, RS2019-II190421, RS-2021-II212052, RS-2021-II212068, RS2025-02217613, RS-2025-10692981, RS-2025-25442569); the Technology Innovation Program (RS-2023-00235718, 23040-15FC) funded by the Ministry of Trade, Industry & Energy (MOTIE, Korea) grant (1415187505).

## References

- [1] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2704–2713, 2018.
- [2] R. Banner, Y. Nahshan, and D. Soudry, "Post training 4-bit quantization of convolutional networks for rapid-deployment," *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [3] M. Nagel, R. A. Amjad, M. Van Baalen, C. Louizos, and T. Blankevoort, "Up or down? adaptive rounding for post-training quantization," in *International Conference on Machine Learning*, pp. 7197–7206, PMLR, 2020.
- [4] Y. Li, R. Gong, X. Tan, Y. Yang, P. Hu, Q. Zhang, F. Yu, W. Wang, and S. Gu, "Brecq: Pushing the limit of post-training quantization by block reconstruction," *arXiv preprint arXiv:2102.05426*, 2021.
- [5] K. Xu, L. Han, Y. Tian, S. Yang, and X. Zhang, "Eq-net: Elastic quantization neural networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1505–1514, 2023.
- [6] B. Chmiel, R. Banner, G. Shomron, Y. Nahshan, A. Bronstein, U. Weiser, *et al.*, "Robust quantization: One model to rule them all," *Advances in neural information processing systems*, vol. 33, pp. 5308–5317, 2020.
- [7] K. Xu, Q. Feng, X. Zhang, and D. Wang, "Multiquant: Training once for multi-bit quantization of neural networks.," in *IJCAI*, pp. 3629–3635, 2022.
- [8] Y. Zhong, Y. Zhou, F. Chao, and R. Ji, "Mbquant: A novel multi-branch topology method for arbitrary bit-width network quantization," *arXiv preprint arXiv:2305.08117*, 2023.
- [9] F. Tahmasebi, Y. Wang, B. Y. Huang, and H. Kwon, "Flexibit: Fully flexible precision bit-parallel accelerator architecture for arbitrary mixed precision ai," *arXiv preprint arXiv:2411.18065*, 2024.
- [10] S. Ryu, H. Kim, W. Yi, and J.-J. Kim, "Bitblade: Area and energy-efficient precision-scalable neural network accelerator with bitwise summation," in *Proceedings of the 56th Annual Design Automation Conference 2019*, pp. 1–6, 2019.
- [11] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "14.5 envision: A 0.26-to-10tops/w subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28nm fdsoi," in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 246–247, 2017.
- [12] V. Camus, L. Mei, C. Enz, and M. Verhelst, "Review and benchmarking of precision-scalable multiply-accumulate unit architectures for embedded neural-network processing," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 4, pp. 697–711, 2019.
- [13] H. Yu, H. Li, Haoxiang opand Shi, T. S. Huang, and G. Hua, "Any-precision deep neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 10763–10771, 2021.
- [14] M. Paul, S. Ganguli, and G. K. Dziugaite, "Deep learning on a data diet: Finding important examples early in training," in *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 20596–20607, 2021.
- [15] X. Xia, J. Liu, J. Yu, X. Shen, B. Han, and T. Liu, "Moderate coreset: A universal method of data selection for real-world data-efficient deep learning," in *International Conference on Learning Representations (ICLR)*, 2023.
- [16] M. Toneva, A. Sordoni, R. Tachet des Combes, A. Trischler, Y. Bengio, and G. J. Gordon, "An empirical study of example forgetting during deep neural network learning," in *International Conference on Learning Representations (ICLR)*, 2019.

- [17] X. Zhang, J. Du, Y. Li, W. Xie, and J. T. Zhou, "Spanning training progress: Temporal dual-depth scoring (tdds) for enhanced dataset pruning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- [18] X. Sun, R. Panda, C.-F. R. Chen, N. Wang, B. Pan, A. Oliva, R. Feris, and K. Saenko, "Improved techniques for quantizing deep networks with adaptive bit-widths," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 957–967, 2024.
- [19] P. Nair, P. Datta, J. Dean, P. Jain, and A. Kusupati, "Matryoshka quantization," *arXiv preprint arXiv:2502.06786*, 2025.
- [20] J. Yu, L. Yang, N. Xu, J. Yang, and T. Huang, "Slimmable neural networks," *arXiv preprint arXiv:1812.08928*, 2018.
- [21] C. Tang, Y. Meng, J. Jiang, S. Xie, R. Lu, X. Ma, Z. Wang, and W. Zhu, "Retraining-free model quantization via one-shot weight-coupling learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15855–15865, 2024.
- [22] M. Welling, "Herding dynamical weights to learn," in *Proceedings of the 26th International Conference on Machine Learning (ICML)*, pp. 1121–1128, 2009.
- [23] C. Coleman, C. Yeh, S. Mussmann, B. Mirzasoleiman, P. Bailis, P. Liang, J. Leskovec, and M. Zaharia, "Selection via proxy: Efficient data selection for deep learning," *arXiv preprint arXiv:1906.11829*, 2019.
- [24] K. Margatina, D. Tsipras, M. Sheehan, and N. Aletras, "Active learning by acquiring contrastive examples," in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021.
- [25] B. Mirzasoleiman, J. Bilmes, and J. Leskovec, "Coresets for data-efficient training of machine learning models," in *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pp. 6950–6960, 2020.
- [26] K. Killamsetty, S. Durga, G. Ramakrishnan, A. De, and R. Iyer, "Grad-match: Gradient matching based data subset selection for efficient deep model training," in *International Conference on Machine Learning*, pp. 5464–5474, PMLR, 2021.
- [27] G. Pleiss, T. Zhang, E. R. Elenberg, and K. Q. Weinberger, "Identifying mislabeled data using the area under the margin ranking," in *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 17044–17056, 2020.
- [28] M. Nagel, M. v. Baalen, T. Blankevoort, and M. Welling, "Data-free quantization through weight equalization and bias correction," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1325–1334, 2019.
- [29] B. Li, B. Wu, J. Su, and G. Wang, "Eagleeye: Fast sub-net evaluation for efficient neural network pruning," in *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pp. 639–654, Springer, 2020.
- [30] M. Nagel, M. Fournarakis, Y. Bondarenko, and T. Blankevoort, "Overcoming oscillations in quantization-aware training," in *International Conference on Machine Learning*, pp. 16318–16330, PMLR, 2022.
- [31] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for boltzmann machines," *Cognitive Science*, vol. 9, no. 1, pp. 147–169, 1985.
- [32] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015.
- [33] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," 2017.
- [34] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.
- [35] Y. Le and X. Yang, "Tiny imagenet visual recognition challenge," *CS 231N*, vol. 7, no. 7, p. 3, 2015.

- [36] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.
- [37] K. He, X. Zhang, S. Ren, and J. Sun, “Identity mappings in deep residual networks,” in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pp. 630–645, Springer, 2016.
- [38] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
- [39] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, “Training data-efficient image transformers & distillation through attention,” in *International conference on machine learning*, pp. 10347–10357, PMLR, 2021.
- [40] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, “Swin transformer: Hierarchical vision transformer using shifted windows,” in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10012–10022, 2021.
- [41] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, “Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients,” *arXiv preprint arXiv:1606.06160*, 2016.
- [42] S.-Y. Liu, Z. Liu, and K.-T. Cheng, “Oscillation-free quantization for low-bit vision transformers,” in *International conference on machine learning*, pp. 21813–21824, PMLR, 2023.
- [43] S. K. Esser, J. L. McKinstry, D. Bablani, R. Appuswamy, and D. S. Modha, “Learned step size quantization,” *arXiv preprint arXiv:1902.08153*, 2019.
- [44] Z. Yuan, C. Xue, Y. Chen, Q. Wu, and G. Sun, “Pttq4vit: Post-training quantization for vision transformers with twin uniform quantization,” in *European conference on computer vision*, pp. 191–207, Springer, 2022.
- [45] M. Phuong and C. Lampert, “Towards understanding knowledge distillation,” in *International conference on machine learning*, pp. 5142–5151, PMLR, 2019.

## Appendix

### Contents

|          |                                                                            |           |
|----------|----------------------------------------------------------------------------|-----------|
| <b>1</b> | <b>Introduction</b>                                                        | <b>1</b>  |
| <b>2</b> | <b>Backgrounds &amp; Related Works</b>                                     | <b>3</b>  |
| <b>3</b> | <b>Weight Bias Correction for Activation Alignment</b>                     | <b>4</b>  |
| <b>4</b> | <b>Bit-Wise Coreset Sampling</b>                                           | <b>5</b>  |
| 4.1      | Coreset sampling strategy . . . . .                                        | 5         |
| 4.2      | Bit-wise training scheme for score evaluation . . . . .                    | 6         |
| <b>5</b> | <b>Experiments</b>                                                         | <b>7</b>  |
| 5.1      | Setup . . . . .                                                            | 7         |
| 5.2      | Results . . . . .                                                          | 8         |
| 5.3      | Ablation . . . . .                                                         | 9         |
| <b>6</b> | <b>Conclusion</b>                                                          | <b>10</b> |
| <b>7</b> | <b>Acknowledgment</b>                                                      | <b>10</b> |
| <b>A</b> | <b>Implementation Details</b>                                              | <b>14</b> |
| <b>B</b> | <b>Additional Experiments</b>                                              | <b>15</b> |
| <b>C</b> | <b>Theoretical Analysis of Cross-bit-width Implicit Knowledge Transfer</b> | <b>21</b> |

### A Implementation Details

**Quantization function.** For CNN experiments, we quantize the weights and activations following the baseline settings of Any-Precision [13], where the activation bit-width is fixed at 4-bits. Any-Precision employs the DoReFa [41] quantizer into its framework. The quantization process is as follows:

$$s = \mathbb{E}(|\mathbf{W}|) \quad (4)$$

$$\mathbf{W}' = \frac{\tanh(\mathbf{W})}{2 \max(|\tanh(\mathbf{W})|)} + 0.5 \quad (5)$$

$$\bar{\mathbf{Q}}_n = \lceil M_n \mathbf{W}' \rceil \quad (6)$$

$$\mathbf{Q}_n = s(2\bar{\mathbf{Q}}_n/M_n - 1) \quad (7)$$

where  $\mathbf{W}'$  is the normalized weight tensor;  $\bar{\mathbf{Q}}_n$  is the quantized bin; and  $\mathbf{Q}_n$  is the quantized floating-point weight.  $M_n = 2^n - 1$  is the maximum bin value with respect to the target bit-width  $n$ .

For ViT experiments, we adopt the StatsQ quantizer proposed in [42] for weight quantization. Activation quantization is performed using the LSQ quantizer [43], fixed at 8-bits. StatsQ applies uniform quantization, using the channel-wise (or head-wise) mean of the absolute weight values as

the scaling factor. The quantization process is defined as follows:

$$s = 2\mathbb{E}_{\mathcal{A}}(|\mathbf{W}|) \quad (8)$$

$$\mathbf{W}' = \text{clip}(\mathbf{W}/s, -1, 1) \quad (9)$$

$$\mathbf{W}'' = \frac{\mathbf{W}'}{2 \max(|\mathbf{W}'|)} + 0.5 \quad (10)$$

$$\bar{\mathbf{Q}}_n = \lceil M_n \mathbf{W}'' \rceil \quad (11)$$

where the remaining steps for mapping the quantized values back to the floating-point range follow the same procedure as in DoReFa. Here, the expectation in the scaling factor  $s$  is taken over dimension(s)  $\mathcal{A}$ , which depend on the shape of  $\mathbf{W}$ : for 2D weights,  $\mathcal{A}$  corresponds to the column dimension; for 3D weights, it includes both the first and last dimensions.

**Implementation of Any-Precision ViTs.** To the best of our knowledge, there is currently no standardized multi-bit framework like Any-Precision [13] available for vision transformers (ViTs). To address this, we develop our own framework, adopting configurations similar to those used in Any-Precision. Our implementation is based on the QAT framework from the OFQ [42] codebase, with minor modifications to the quantizer to match the quantization function described above and without the query-key reparameterization proposed in [42].

**Configurations of baseline coreset selection methods.** To evaluate the effectiveness of our method, we compare it against several existing coreset selection methods. While prior approaches typically focus on selecting a single subset optimized for a single model or configuration, our method targets bit-wise unique subsets tailored for multi-bit training. Accordingly, we apply each baseline’s scoring strategy independently to each sub-model, constructing distinct coresets, and train each sub-model on its respective subset. For CIFAR-10 and CIFAR-100, we train for 200 epochs; for ImageNet-1K, the number of epochs varies depending on the model. For ImageNet-1K experiments, we initialize from pretrained models: ResNet models use the Any-Precision weights [13], while ViTs are initialized from the 8-bit quantized checkpoints provided by PTQ4ViT [44]. The specific configurations for each coreset selection method are as follows:

- **Entropy** [23], **EL2N** [14]: Scores are computed over 20 epochs for CIFAR-10/100 and 5 epochs for ImageNet-1K. Entropy identifies samples near decision boundaries, while EL2N ranks samples based on the magnitude of their gradients.
- **Forgetting** [16]: Scores are computed over 200 epochs for CIFAR-10/100 and 10 epochs for ImageNet-1K. A forgetting score is defined by the number of times a sample is misclassified after being learned correctly, capturing how often a sample is “forgotten” during training.
- **Moderate** [15]: Scores are computed using features extracted by a pre-trained model for 200 epochs on CIFAR-10/100, and for 70 epochs on ImageNet-1K, as an importance proxy. Moderate quantifies importance by measuring distances between samples in the resulting feature space.
- **TDDS** [17]: Scores are computed over 10 epochs for CIFAR-10, 20 epochs for CIFAR-100, and 5 epochs for ImageNet-1K. TDDS combines gradient information with training dynamics to estimate sample importance.
- **Bit-wise Unique Scores (Ours)**: Our method computes per-bit-width importance scores over 10 epochs for CIFAR-10, 20 epochs for CIFAR-100, and 5 epochs for ImageNet-1K, enabling dynamic coreset selection tailored to each sub-network.

## B Additional Experiments

**Comparison of score uniqueness in bit-wise vs. batch-wise training scheme.** To analyze the effect of the proposed *bit-wise training scheme*, we examine the uniqueness of the importance scores it produces for each bit-width sub-model, compared to those from the standard *batch-wise training scheme*. Figure 7 shows a heatmap of Spearman correlation values, which quantify the similarity between importance scores produced by different bit-width sub-models. On both CIFAR-10 and CIFAR-100, the bit-wise scheme results in consistently lower correlation values, indicating that it yields more distinct and representative importance scores for each bit-width. This suggests that separating the training dynamics by bit-width is essential for accurately capturing per-sample

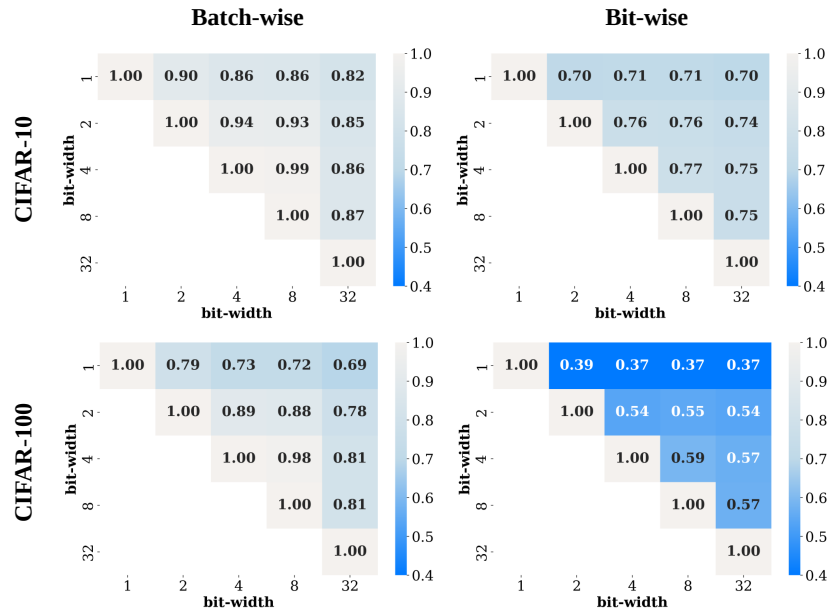


Figure 7: Comparison of score similarity across sub-models under batch-wise and bit-wise training schemes, measured by Spearman correlation. *The top row* shows PreActResNet-20 on CIFAR-10 results and *the bottom row* shows PreActResNet-18 on CIFAR-100; in each row, *the left* is batch-wise and *the right* is bit-wise.

importance in multi-bit settings, as it avoids the gradient aggregation seen in batch-wise training and allows each sub-model to maintain distinct context vectors. This isolation is critical for deriving meaningful, bit-wise importance estimates, which are otherwise masked under the standard batch-wise update pattern.

Table 8: PreActResNet-20 on CIFAR-10 and PreActResNet-18 on CIFAR-100. Breakdown of GPU hours.

| Dataset   | Framework                                 | GPU hours   |             |              |                | Total GPU hours (Speed up) |
|-----------|-------------------------------------------|-------------|-------------|--------------|----------------|----------------------------|
|           |                                           | Training    | Calibration | Adaptation   | Scoring        |                            |
| CIFAR-10  | Dedicated                                 | 11.97       | -           | -            | -              | 11.97 (1.00×)              |
|           | Any-Prec.                                 | 7.51        | 1.25        | -            | -              | 8.76 (1.36×)               |
|           | <b>Bias Correction</b>                    | 7.51        | -           | <b>0.004</b> | -              | 7.52 ( <b>1.59×</b> )      |
|           | <b>Bias Correction + Coreset Sampling</b> | <b>1.52</b> | -           | <b>0.004</b> | 0.37 (offline) | <b>1.52 (7.88×)</b>        |
| CIFAR-100 | Dedicated                                 | 11.19       | -           | -            | -              | 11.19 (1.00×)              |
|           | Any-Prec.                                 | 7.17        | 1.10        | -            | -              | 8.27 (1.36×)               |
|           | <b>Bias Correction</b>                    | 7.17        | -           | <b>0.004</b> | -              | 7.17 ( <b>1.56×</b> )      |
|           | <b>Bias Correction + Coreset Sampling</b> | <b>1.47</b> | -           | <b>0.004</b> | 0.74 (offline) | <b>1.47 (7.61×)</b>        |

**Breakdown of GPU hours.** We report GPU hours for all ResNet experiments, broken down by training stage, including: score evaluation, coreset training, calibration (if applicable), and adaptation. Table 8 also help clarify the computational cost structure of our framework. Specifically, coreset sampling significantly reduces training GPU hours for multi-bit quantization models, and this efficiency gain increases with the pruning rate. However, coreset sampling alone does not eliminate the cost of the calibration phase, which is typically needed to support additional bit-widths. To address this, we apply Bias Correction and BN adaptation, which allow us to remove the calibration step entirely without sacrificing accuracy.

**Experiments on alignment achieved by Bias Correction.** To demonstrate the effectiveness of our Bias Correction method, we conducted an analysis of the output activations, comparing how well activations from different bit-widths align with one another. In all experiments, the activation

Table 9: MAE between different bit-widths and full precision models’ activation with vs. without Bias Correction.

| Bias<br>Correction | MAE   |       |       |       |       |       |       |       |
|--------------------|-------|-------|-------|-------|-------|-------|-------|-------|
|                    | 2bit  | 3bit  | 4bit  | 5bit  | 6bit  | 7bit  | 8bit  | Avg.  |
| -                  | 0.744 | 0.756 | 0.754 | 0.750 | 0.761 | 0.756 | 0.766 | 0.755 |
| ✓                  | 0.667 | 0.670 | 0.671 | 0.654 | 0.683 | 0.668 | 0.659 | 0.667 |

precision is fixed to 4 bits, and we measure the mean absolute error (MAE) between the activations of quantized and full-precision models, with and without applying Bias Correction. Table 9 show that applying the correction consistently improves alignment, reducing the avg. MAE from 0.755 to 0.667. This improvement holds across all bit-widths, demonstrating our method’s effectiveness. We will include these results in the revised version to highlight the impact of Bias Correction on activation alignment.

Table 10: PreActResNet-20 on CIFAR-10. BN sharing with and without 1-bit quantization.

| Framework         | Test Accuracy |       |       |       |       |       |       |       |       |              |
|-------------------|---------------|-------|-------|-------|-------|-------|-------|-------|-------|--------------|
|                   | 1bit          | 2bit  | 3bit  | 4bit  | 5bit  | 6bit  | 7bit  | 8bit  | 32bit | Avg.         |
| Share BN w/ 1bit  | 46.38         | 91.90 | 92.57 | 91.48 | 90.98 | 90.53 | 90.24 | 90.14 | 89.90 | <b>86.01</b> |
| Share BN w/o 1bit | 92.49         | 89.44 | 92.43 | 92.67 | 92.43 | 92.15 | 92.00 | 91.97 | 91.65 | <b>91.91</b> |

**Experiments on effectiveness of BN sharing for 1-bit quantization.** In multi-bit settings, weight distributions tend to approximate a normal distribution, where our Bias Correction mechanism works effectively. However, in the 1-bit case, the weight distribution collapses into a near-uniform or binary form, causing a significant distribution shift that Bias Correction alone struggles to adequately address. For that reason, prior works on multi-bit networks often omit the 1-bit setting from their analysis [5, 18, 8]. In contrast, our method explicitly incorporates this case and demonstrates that assigning a separate BN layer for 1-bit quantization is both a simple and effective solution.

To further support this, we conduct ablation studies comparing two configurations: one where BN is shared across all bit-widths (including 1-bit), and another where the 1-bit case has its own BN, and the remaining bit-widths share a single BN. Table 10 clearly show that including 1-bit in BN sharing significantly degrades 1-bit quantization performance. On the other hand, using a separate BN for 1-bit achieves strong performance with negligible additional overhead, that is the additional parameters for the separate BN layer, which accounts for less than 0.01% of the total number of parameters. This performance degradation stems from unstable BN statistics. The 1-bit weights produce activation distributions that are markedly different from those of higher bit-widths, leading to significant fluctuations in shared BN statistics. These fluctuations negatively impact the normalization of other bit-widths, ultimately harming overall model performance.

Table 11: PreActResNet-20 on CIFAR-10. Impact of only coreset sampling.

| Coreset Sampling<br>(Pruning Ratio) | Bias<br>Correction | Test Accuracy |       |       |       |       |       |       |       |       |              | GPU hours<br>(Speed up) |
|-------------------------------------|--------------------|---------------|-------|-------|-------|-------|-------|-------|-------|-------|--------------|-------------------------|
|                                     |                    | 1bit          | 2bit  | 3bit  | 4bit  | 5bit  | 6bit  | 7bit  | 8bit  | 32bit | Avg.         |                         |
| - (0.0)                             | -                  | 92.95         | 87.72 | 93.53 | 93.32 | 92.73 | 92.47 | 92.08 | 91.87 | 93.53 | <b>92.24</b> | 7.51 (1.00×)            |
| ✓ (0.7)                             | -                  | 92.46         | 87.42 | 92.92 | 93.05 | 92.51 | 91.95 | 91.69 | 91.49 | 92.98 | <b>91.83</b> | 2.06 (3.65×)            |
|                                     | ✓                  | 92.62         | 93.03 | 92.98 | 93.08 | 93.05 | 93.05 | 93.08 | 93.10 | 93.15 | <b>93.02</b> |                         |
| ✓ (0.8)                             | -                  | 92.29         | 87.61 | 92.84 | 92.27 | 91.51 | 90.99 | 90.68 | 90.46 | 92.84 | <b>91.28</b> | 1.52 (4.94×)            |
|                                     | ✓                  | 92.60         | 93.01 | 92.96 | 93.03 | 93.02 | 92.99 | 93.01 | 93.00 | 93.08 | <b>92.97</b> |                         |
| ✓ (0.9)                             | -                  | 91.47         | 87.07 | 91.61 | 91.22 | 90.42 | 89.62 | 89.23 | 89.00 | 91.62 | <b>90.14</b> | 0.83 (9.05×)            |
|                                     | ✓                  | 92.04         | 92.66 | 92.61 | 92.66 | 92.47 | 92.50 | 92.51 | 92.63 | 92.38 | <b>92.50</b> |                         |

**Experiments on impact of coreset sampling alone.** To examine the standalone contribution of coreset sampling to both accuracy and training speedup, we provide results on a CIFAR-10 baseline where only coreset sampling is applied, with all bit-widths except for 1-bit sharing BN layers. As shown in Table 11, while coreset sampling contributes most to the speedup, it is not sufficient on its

own to maintain strong accuracy at across every bit-width. In contrast, when Bias Correction and BN adaptation are applied alongside coreset sampling, we observe accuracy improvements everywhere, with practically no additional GPU hours. The results indicate that although coreset sampling is the main driver of compute efficiency, Bias Correction and BN adaptation is essential for best accuracy.

Table 12: PreActResNet-20 on CIFAR-10. Influence of varying fixed temperatures and scheduling methods.

| Temperature<br>scheme     |        | Test Accuracy |       |       |       |       |       |
|---------------------------|--------|---------------|-------|-------|-------|-------|-------|
|                           |        | 1bit          | 2bit  | 4bit  | 8bit  | 32bit | Avg   |
| Fixed                     | 0.1    | 90.08         | 92.11 | 92.35 | 92.55 | 92.21 | 91.86 |
|                           | 0.5    | 90.61         | 92.61 | 92.78 | 92.85 | 92.61 | 92.29 |
|                           | 1.0    | 90.33         | 92.41 | 92.51 | 92.62 | 91.87 | 91.95 |
| Scheduling<br>(0.1 ~ 1.0) | Linear | 90.40         | 92.29 | 92.07 | 92.22 | 91.55 | 91.71 |
|                           | Exp.   | 90.57         | 92.39 | 92.56 | 92.63 | 91.93 | 92.02 |
|                           | Log    | 90.77         | 92.64 | 92.65 | 92.89 | 92.06 | 92.20 |

**Experiments with varying fixed temperature settings and scheduling methods.** Table 12 presents the test accuracies of our method, evaluated in the Any-Precision setting, trained across bit-widths  $b \in \{1, 2, 4, 8, 32\}$  using fixed sampling temperatures  $\tau \in \{0.1, 0.5, 1.0\}$  and three temperature scheduling strategies (e.g., linear, exponential, logarithmic). We observe that among the fixed settings, a moderate temperature of  $\tau = 0.5$  consistently achieves the highest accuracy, outperforming both the low ( $\tau = 0.1$ ) and high ( $\tau = 1.0$ ) extremes. Dynamically increasing the temperature from 0.1 to 1.0 over training—regardless of the scheduling scheme—yields performance that is comparable to or worse than using a fixed  $\tau = 0.5$ . These results suggest that a single, well-chosen temperature is sufficient to balance the sampling distribution—favoring informative samples while maintaining diversity. In contrast, dynamically adjusting the temperature throughout training introduces additional complexity without delivering clear performance benefits. Based on these observations, all coreset sampling experiments were conducted with the temperature fixed at  $\tau = 0.5$ .

Table 13: PreActResNet-20 on CIFAR-10 and PreActResNet-18 on CIFAR-100. Performance of calibrated bit-widths when pruning rate for *Coreset Sampling* is 80%.

| Dataset   | Framework | Coreset Sampling | Test Accuracy    |                  |                  |                  |
|-----------|-----------|------------------|------------------|------------------|------------------|------------------|
|           |           |                  | 3bit             | 5bit             | 6bit             | 7bit             |
| CIFAR-10  | Any-Prec. | -                | 93.17 $\pm 0.26$ | 93.19 $\pm 0.18$ | 93.16 $\pm 0.25$ | 93.10 $\pm 0.24$ |
|           | Ours      | -                | 93.46 $\pm 0.14$ | 93.54 $\pm 0.02$ | 93.50 $\pm 0.07$ | 93.43 $\pm 0.10$ |
|           |           | ✓                | 92.96 $\pm 0.09$ | 93.02 $\pm 0.04$ | 92.99 $\pm 0.13$ | 93.01 $\pm 0.13$ |
| CIFAR-100 | Any-Prec. | -                | 71.28 $\pm 0.26$ | 71.43 $\pm 0.24$ | 71.53 $\pm 0.15$ | 71.45 $\pm 0.09$ |
|           | Ours      | -                | 71.93 $\pm 0.07$ | 71.96 $\pm 0.11$ | 71.96 $\pm 0.12$ | 71.95 $\pm 0.14$ |
|           |           | ✓                | 70.41 $\pm 0.15$ | 70.42 $\pm 0.10$ | 70.47 $\pm 0.08$ | 70.57 $\pm 0.13$ |

**Evaluation of calibrated bit-widths.** In this section, we present the performance of calibrated bit-widths, which were omitted from the main experimental results in the main paper. Although we refer to these as *calibrated bit-widths*, it is important to clarify that, in our method, these bit-widths are not explicitly trained or fine-tuned. Instead, we obtain their accuracy using bias correction and batch normalization (BN) adaptation, without additional training or calibration stages. In contrast, Any-Precision [13] recovers calibrated bit-width performance by performing a separate post-training BN calibration procedure. As shown in Table 13 and Table 14, the calibrated bit-widths in our method achieve accuracy comparable to trained bit-widths, confirming that our proposed weight bias correction effectively aligns activation distributions without the need for costly calibration.

**Evaluation against baseline coreset selection methods on CIFAR-100.** Table 15 presents additional experimental results on CIFAR-100 using the PreActResNet-18 architecture, comparing our bit-wise coreset sampling method against several baseline coreset selection strategies. Consistent with prior findings in TDDS [17] and our CIFAR-10 experiments reported in the main paper, we observe that

Table 14: PreActResNet-20 on CIFAR-10 and PreActResNet-18 on CIFAR-100. Performance of calibrated bit-widths compared to previous methods at an 80% pruning rate.

| Dataset   | Method      | Test Accuracy    |                  |                  |                  |
|-----------|-------------|------------------|------------------|------------------|------------------|
|           |             | 3bit             | 5bit             | 6bit             | 7bit             |
| CIFAR-10  | Random      | 90.14 $\pm 0.25$ | 90.15 $\pm 0.30$ | 90.13 $\pm 0.32$ | 90.10 $\pm 0.29$ |
|           | Entropy     | 86.29 $\pm 0.37$ | 86.30 $\pm 0.29$ | 86.27 $\pm 0.27$ | 86.28 $\pm 0.25$ |
|           | Forgetting  | 78.12 $\pm 0.95$ | 78.43 $\pm 1.01$ | 78.37 $\pm 1.00$ | 78.24 $\pm 1.02$ |
|           | EL2N        | 81.15 $\pm 0.22$ | 81.21 $\pm 0.38$ | 81.20 $\pm 0.31$ | 81.13 $\pm 0.43$ |
|           | Moderate    | 88.36 $\pm 0.11$ | 88.35 $\pm 0.15$ | 88.39 $\pm 0.22$ | 88.40 $\pm 0.19$ |
|           | TDDS        | 88.72 $\pm 0.13$ | 88.72 $\pm 0.07$ | 88.70 $\pm 0.05$ | 88.71 $\pm 0.01$ |
|           | <b>Ours</b> | 92.96 $\pm 0.09$ | 93.02 $\pm 0.04$ | 92.99 $\pm 0.13$ | 93.01 $\pm 0.13$ |
| CIFAR-100 | Random      | 63.01 $\pm 0.50$ | 63.00 $\pm 0.44$ | 63.02 $\pm 0.42$ | 63.09 $\pm 0.34$ |
|           | Entropy     | 53.46 $\pm 0.31$ | 53.45 $\pm 0.42$ | 53.54 $\pm 0.51$ | 53.53 $\pm 0.34$ |
|           | Forgetting  | 38.71 $\pm 0.46$ | 39.40 $\pm 0.55$ | 39.49 $\pm 0.60$ | 39.27 $\pm 0.72$ |
|           | EL2N        | 31.00 $\pm 0.59$ | 31.44 $\pm 0.62$ | 31.57 $\pm 0.62$ | 31.49 $\pm 0.55$ |
|           | Moderate    | 58.54 $\pm 0.21$ | 58.63 $\pm 0.19$ | 58.67 $\pm 0.17$ | 58.66 $\pm 0.13$ |
|           | TDDS        | 54.40 $\pm 0.27$ | 54.40 $\pm 0.40$ | 54.40 $\pm 0.44$ | 54.35 $\pm 0.40$ |
|           | <b>Ours</b> | 70.41 $\pm 0.15$ | 70.42 $\pm 0.10$ | 70.47 $\pm 0.08$ | 70.57 $\pm 0.13$ |

Table 15: PreActResNet-18 on CIFAR-100. Comparison with previous methods at 80% pruning rate.

| Method      | Test Accuracy    |                  |                  |                  |                  |              |
|-------------|------------------|------------------|------------------|------------------|------------------|--------------|
|             | 1bit             | 2bit             | 4bit             | 8bit             | 32bit            | Avg.         |
| Random      | 60.32 $\pm 0.62$ | 62.45 $\pm 0.51$ | 63.11 $\pm 0.49$ | 62.87 $\pm 0.53$ | 61.39 $\pm 0.60$ | 62.47        |
| Entropy     | 52.65 $\pm 0.37$ | 53.28 $\pm 0.20$ | 53.55 $\pm 0.37$ | 53.53 $\pm 0.47$ | 53.22 $\pm 0.62$ | 53.36        |
| Forgetting  | 35.21 $\pm 0.37$ | 38.00 $\pm 0.34$ | 39.18 $\pm 0.50$ | 39.45 $\pm 0.53$ | 37.94 $\pm 0.68$ | 38.52        |
| EL2N        | 30.49 $\pm 0.91$ | 31.07 $\pm 0.57$ | 31.38 $\pm 0.62$ | 31.50 $\pm 0.57$ | 29.43 $\pm 0.29$ | 31.04        |
| Moderate    | 57.05 $\pm 0.40$ | 58.24 $\pm 0.42$ | 58.69 $\pm 0.16$ | 58.67 $\pm 0.10$ | 57.83 $\pm 0.37$ | 58.33        |
| TDDS        | 53.65 $\pm 0.36$ | 54.07 $\pm 0.43$ | 54.40 $\pm 0.40$ | 54.36 $\pm 0.41$ | 54.04 $\pm 0.55$ | 54.23        |
| <b>Ours</b> | 69.14 $\pm 0.08$ | 70.12 $\pm 0.11$ | 70.35 $\pm 0.17$ | 70.43 $\pm 0.11$ | 70.41 $\pm 0.11$ | <b>70.26</b> |

random coreset selection performs surprisingly well at high pruning rates. This trend persists in the CIFAR-100 setting, where random sampling remains a strong baseline under severe data reduction. Nonetheless, our proposed method consistently outperforms the baselines across different bit-widths, demonstrating its effectiveness in selecting informative samples even under high pruning constraints.

**Evaluation of DeiT-S on CIFAR-100 and TinyImageNet** We further evaluate our method on a transformer-based architecture, specifically DeiT-S, using smaller datasets such as CIFAR-100 and TinyImageNet. The results are presented in Table 16. To the best of our knowledge, there is currently no standardized multi-bit framework like Any-Precision for vision transformers. To address this, we implement our own framework following configurations similar to Any-Precision, with a slight modification to the StatsQ quantizer—details of which are provided in Section A. Our method demonstrates consistently strong performance compared to the basic Any-Precision setup, even when pruning the dataset by 60%, achieving up to an  $8.41\times$  reduction in GPU hours on TinyImageNet.

**Evaluation of storage-constrained coreset sampling.** Coreset-based approaches in multi-bit networks consistently reduce training time; however, due to varying data importance across sub-models, it remains challenging to impose a uniform constraint on the total number of training samples used by the entire model. To address this, we first discard samples that are consistently considered unimportant across all sub-models, and then apply our coreset sampling method with bias correction. To identify and remove consistently uninformative samples before applying coreset sampling, we first compute the importance of each training sample for every sub-model over a single epoch, following our bit-wise training scheme. These importance values are then summed across sub-models, and their variability is assessed—similar to training dynamics approaches [17, 16]—to obtain a

Table 16: DeiT-S on CIFAR-100 and TinyImageNet.

| Dataset      | Framework | Pruning Rate | Test Accuracy |       |       |       |              | GPU hours (Speed up)   |
|--------------|-----------|--------------|---------------|-------|-------|-------|--------------|------------------------|
|              |           |              | 2bit          | 4bit  | 6bit  | 8bit  | Avg.         |                        |
| CIFAR-100    | Dedicated | -            | 87.14         | 87.92 | 87.88 | 88.03 | 87.74        | 41.01 (1.00×)          |
|              | Any-Prec. | -            | 87.52         | 88.30 | 88.20 | 88.21 | 88.08        | 10.47 (3.92×)          |
|              | Ours      | 50%          | 87.83         | 88.56 | 88.68 | 88.59 | <b>88.43</b> | 6.05 ( <b>6.78×</b> )  |
|              |           | 60%          | 87.61         | 88.45 | 88.54 | 88.65 | <b>88.31</b> | 5.20 ( <b>7.89×</b> )  |
| TinyImageNet | Dedicated | -            | 82.61         | 85.60 | 85.68 | 85.86 | 84.94        | 74.00 (1.00×)          |
|              | Any-Prec. | -            | 82.10         | 84.61 | 84.47 | 84.70 | 84.07        | 19.32 (3.83×)          |
|              | Ours      | 50%          | 82.54         | 84.95 | 85.33 | 85.17 | <b>84.59</b> | 10.50 ( <b>7.05×</b> ) |
|              |           | 60%          | 82.89         | 84.39 | 84.95 | 84.86 | <b>84.26</b> | 8.80 ( <b>8.41×</b> )  |

Table 17: PreActResNet-20 on CIFAR-10. Comparison across varying storage reduction rates.

| Storage Reduction | Test Accuracy |       |       |       |       |              |
|-------------------|---------------|-------|-------|-------|-------|--------------|
|                   | 1bit          | 2bit  | 4bit  | 8bit  | 32bit | Avg.         |
| 0%                | 92.60         | 93.01 | 93.03 | 93.00 | 93.08 | <b>92.97</b> |
| 20%               | 92.39         | 93.17 | 93.26 | 93.32 | 93.32 | <b>93.20</b> |
| 30%               | 92.28         | 92.70 | 92.68 | 92.62 | 93.10 | <b>92.65</b> |
| 40%               | 92.09         | 92.61 | 92.75 | 92.77 | 92.55 | <b>92.64</b> |
| 50%               | 91.97         | 92.15 | 92.19 | 92.29 | 92.61 | <b>92.25</b> |

final importance score. As shown in Table 17, our coreset sampling method performs comparably to existing approaches, even under a 50% dataset storage constraint. Moreover, Table 18 shows that training performance can be further enhanced by tuning the pruning rate (i.e., training time), highlighting the adaptability of our method to varying resource budgets in multi-bit network training.

**Experiments on Influence of coreset sampling frequency.** In practice, the overhead of bit-wise coreset resampling is extremely small compared to the overall training cost. For example, even on an ImageNet-scale dataset, performing 100 resamplings takes only about 3.36 minutes. Given this negligible cost, resampling at every epoch is a practical and effective choice.

To quantitatively demonstrate this, we conducted experiments with different resampling intervals and measured both validation accuracy and total sampling time. Table 19 show that resampling every epoch improves average accuracy by 1.33%p compared to resampling every 30 epochs, while adding just 53 seconds of overhead to a multi-hour training process. This demonstrates that frequent resampling can offer meaningful accuracy gains at virtually no additional cost.

**Experiments on dynamic score re-evaluation.** We conducted additional experiments where importance scores are dynamically re-evaluated every 10, 30, 50, or 100 epochs, and coresets are resampled accordingly. We evaluated how it impacts accuracy and GPU hours using PreActResNet-20 on CIFAR-10 and PreActResNet-18 on CIFAR-100 under both 80% and 90% data pruning. Table 20 and Table 21 reveal a consistent pattern: while dynamic score re-evaluation leads to only marginal accuracy changes, it incurs a substantial increase in computational cost. In many settings, our one-time scoring strategy already matches or even outperforms more frequent re-evaluation in terms of final accuracy, while consuming significantly fewer GPU hours. This empirical finding validates our design choice, and shows that a single, well-computed importance estimate—when paired with stochastic sampling—offers an effective and efficient balance, capturing most of the benefits of dynamic importance tracking without incurring its heavy cost. Looking ahead, with the observation that dynamic re-evaluation yields modest gains on the more challenging CIFAR-100, we believe dynamic sampling techniques could be the key to boosting performance on complex, high-variability

Table 18: PreActResNet-20 on CIFAR-10 and PreActResNet-18 on CIFAR-100. Comparison across pruning rates when dataset storage is fixed at 30K out of 50K samples. Since retaining 30K out of 50K samples represents a 40% reduction, a pruning rate of 40% corresponds to the full-training scenario in this context.

| Dataset   | Pruning Rate | Test Accuracy |       |       |       |       |              |
|-----------|--------------|---------------|-------|-------|-------|-------|--------------|
|           |              | 1bit          | 2bit  | 4bit  | 8bit  | 32bit | Avg.         |
| CIFAR-10  | 40% (full)   | 93.01         | 93.39 | 93.61 | 93.58 | 93.65 | <b>93.50</b> |
|           | 50%          | 92.81         | 93.07 | 93.11 | 93.15 | 93.39 | <b>93.13</b> |
|           | 60%          | 92.66         | 93.07 | 93.04 | 93.09 | 93.48 | <b>93.07</b> |
|           | 70%          | 92.52         | 92.91 | 93.07 | 93.11 | 93.17 | <b>92.97</b> |
|           | 80%          | 92.09         | 92.61 | 92.75 | 92.77 | 92.55 | <b>92.64</b> |
| CIFAR-100 | 40% (full)   | 66.01         | 66.35 | 66.85 | 66.90 | 66.90 | <b>66.65</b> |
|           | 50%          | 66.32         | 66.98 | 67.60 | 67.93 | 67.49 | <b>67.52</b> |
|           | 60%          | 65.65         | 65.99 | 66.48 | 66.38 | 66.20 | <b>66.24</b> |
|           | 70%          | 66.10         | 66.86 | 66.69 | 66.90 | 67.03 | <b>66.78</b> |
|           | 80%          | 65.29         | 66.30 | 66.33 | 66.39 | 65.83 | <b>66.19</b> |

Table 19: PreActResNet-20 on CIFAR-10. Influence of sampling frequency.

| Resampling Frequency | Test Accuracy |       |       |       |       |              | Total Sampling Time (% of Total GPU time) |
|----------------------|---------------|-------|-------|-------|-------|--------------|-------------------------------------------|
|                      | 1bit          | 2bit  | 4bit  | 8bit  | 32bit | Avg.         |                                           |
| 1                    | 92.60         | 93.01 | 93.03 | 93.00 | 93.08 | <b>92.97</b> | 53.02s (0.96%)                            |
| 10                   | 91.96         | 92.40 | 92.65 | 92.66 | 92.75 | <b>92.55</b> | 5.09s (0.09%)                             |
| 20                   | 91.40         | 91.73 | 92.04 | 92.03 | 92.34 | <b>91.98</b> | 2.77s (0.05%)                             |
| 30                   | 91.29         | 91.53 | 91.63 | 91.73 | 91.68 | <b>91.64</b> | 1.40s (0.03%)                             |

datasets where importance scores drift more drastically throughout training. The main hurdle is the high cost of score re-evaluation during training, which currently limits the practicality of dynamic methods. In future work, we will explore lightweight techniques to reduce score-evaluation overhead while maintaining the quality of importance estimates.

## C Theoretical Analysis of Cross-bit-width Implicit Knowledge Transfer

In this section, we use a simple linear classifier to examine how shared weights in multi-bit networks can *implicitly* transfer knowledge between sub-networks. We consider a setting where an 8-bit model and a 2-bit model share the same real-valued parameter vector  $w$ , with weights quantized using the DoReFa quantizer [41]. Training alternates iteratively: the 8-bit model is trained on batch  $X_8$  with hard labels, followed by the 2-bit model trained on a separate batch  $X_2$ , also with hard labels. These batches are drawn independently from the data matrix  $X \in \mathbb{R}^{d \times n}$  and do not overlap. The shared parameter  $w$  is updated in-place using the cross-entropy loss and is continuously modified by both models. The key question is: *can we formally argue that, despite no explicit soft-label distillation and no shared data examples, the 2-bit model benefits from the 8-bit model’s training- and vice versa- through the shared parameter?*

**Gradient update within combined data subspaces.** When the 8-bit model observes batch  $B_8 = (X_8, y_8)$ , it performs a gradient step using the cross-entropy loss. The model is blind to any component of the optimum that is orthogonal to the plane that spans the  $n_8$ -dimensional subspace of  $X_8$  [45]. That is, under an asymptotic assumption, the direction of the update is fully constrained to the subspace spanned by the input vectors in the batch, as the gradient is a linear combination of the each data point  $x_i$ . Therefore, the gradient of the 8-bit model lies within the subspace spanned by its input batch, and thus the corresponding update step is bound as follows:  $\Delta_8 \in \text{span}(X_8)$ . Similarly, the 2-bit model performs its update with batch  $X_2$ . By induction, the net update to  $w$  lies within a sum

Table 20: PreActResNet-20 on CIFAR-10. Impact of score re-evaluation.

| Pruning Ratio | Re-eval Frequency | Test Accuracy |       |       |       |       |              | Re-eval GPU hours | Total GPU hours |
|---------------|-------------------|---------------|-------|-------|-------|-------|--------------|-------------------|-----------------|
|               |                   | 1bit          | 2bit  | 4bit  | 8bit  | 32bit | Avg.         |                   |                 |
| 0.8           | -                 | 92.60         | 93.01 | 93.03 | 93.00 | 93.08 | <b>92.97</b> | -                 | <b>1.52</b>     |
|               | 100               | 92.08         | 92.64 | 92.80 | 92.82 | 92.88 | <b>92.75</b> | <b>0.38</b>       | <b>5.31</b>     |
|               | 50                | 92.45         | 92.83 | 92.96 | 93.04 | 92.96 | <b>92.97</b> | <b>1.13</b>       | <b>2.65</b>     |
|               | 30                | 92.64         | 92.90 | 93.03 | 92.88 | 92.69 | <b>92.84</b> | <b>2.27</b>       | <b>3.79</b>     |
|               | 10                | 92.21         | 92.56 | 92.73 | 92.86 | 92.75 | <b>92.73</b> | <b>7.33</b>       | <b>8.85</b>     |
| 0.9           | -                 | 92.04         | 92.66 | 92.66 | 92.63 | 92.38 | <b>92.38</b> | -                 | <b>0.84</b>     |
|               | 100               | 91.43         | 92.09 | 92.09 | 92.09 | 91.89 | <b>92.09</b> | <b>0.37</b>       | <b>1.21</b>     |
|               | 50                | 91.37         | 91.98 | 92.22 | 92.29 | 92.32 | <b>92.09</b> | <b>1.15</b>       | <b>1.99</b>     |
|               | 30                | 91.94         | 92.13 | 92.12 | 92.11 | 92.38 | <b>92.10</b> | <b>2.31</b>       | <b>3.15</b>     |
|               | 10                | 91.24         | 91.64 | 91.94 | 92.03 | 91.77 | <b>91.82</b> | <b>7.33</b>       | <b>8.17</b>     |

Table 21: PreActResNet-18 on CIFAR-100. Impact of score re-evaluation.

| Pruning Ratio | Re-eval Frequency | Test Accuracy |       |       |       |       |              | Re-eval GPU hours | Total GPU hours |
|---------------|-------------------|---------------|-------|-------|-------|-------|--------------|-------------------|-----------------|
|               |                   | 1bit          | 2bit  | 4bit  | 8bit  | 32bit | Avg.         |                   |                 |
| 0.8           | -                 | 69.14         | 70.12 | 70.35 | 70.43 | 70.41 | <b>70.26</b> | -                 | <b>1.47</b>     |
|               | 100               | 69.16         | 70.24 | 70.41 | 70.39 | 70.04 | <b>70.18</b> | <b>0.77</b>       | <b>2.24</b>     |
|               | 50                | 69.13         | 70.23 | 70.60 | 70.82 | 70.41 | <b>70.45</b> | <b>2.30</b>       | <b>3.78</b>     |
|               | 30                | 68.91         | 70.17 | 70.18 | 70.05 | 70.09 | <b>70.02</b> | <b>4.57</b>       | <b>6.04</b>     |
|               | 10                | 68.95         | 70.45 | 70.76 | 70.78 | 70.87 | <b>70.66</b> | <b>14.57</b>      | <b>16.04</b>    |
| 0.9           | -                 | 67.53         | 69.40 | 69.75 | 69.25 | 69.24 | <b>69.32</b> | -                 | <b>0.83</b>     |
|               | 100               | 67.34         | 69.17 | 69.23 | 69.29 | 69.28 | <b>69.02</b> | <b>0.76</b>       | <b>1.59</b>     |
|               | 50                | 67.77         | 69.12 | 69.43 | 69.20 | 69.11 | <b>69.11</b> | <b>2.32</b>       | <b>3.15</b>     |
|               | 30                | 67.82         | 69.30 | 69.11 | 69.34 | 69.02 | <b>69.15</b> | <b>4.63</b>       | <b>5.46</b>     |
|               | 10                | 67.28         | 69.14 | 69.41 | 69.50 | 69.44 | <b>69.23</b> | <b>14.68</b>      | <b>15.51</b>    |

of subspaces as follows:  $\Delta_{\text{net}} \in \sum_{j=1}^{\mathcal{B}} \text{span}(X_j)$ , where  $\mathcal{B} = \{2, 8\}$  in our case. Thus, the shared weight vector evolves within the union of data subspaces:  $\text{span}(X_8) \cup \text{span}(X_2)$ . This shows that each sub-network updates its parameters based on a broader subspace that includes data from other sub-networks, thereby increasing its effective data exposure.

**Gradient alignment between quantized sub-networks.** Given that updates occur within a shared subspace, we analyze whether the gradients from different quantized sub-networks are sufficiently aligned to enable mutual benefit. We assume that the optimum value for both quantized model is similar [41]. In our setting, when the 8-bit model receives batch  $X_8$ , this updates  $w$  towards minimizing  $\mathcal{L}_8$ . Since  $\Delta w_8 \in \text{span}(X_8)$ , this update steers  $w$  toward the optimum  $w^*$  within  $\text{span}(X_8)$ . The 8-bit model moves  $w$  to a point where the 2-bit loss cannot be worse—and is often better (i.e.,  $\theta < 90$ ). This is functionally equivalent to soft distillation: the 8-bit model’s higher-capacity updates are immediately used by the 2-bit model, enabling generalization benefits without soft targets. This provides a theoretical basis for implicit knowledge transfer as the shared parameter acts as a channel of indirect supervision.

Motivated by these observations, we propose a bit-wise coreset sampling method that trains each sub-network with its own compact, informative subset of the full dataset. As the multi-bit network implicitly accesses the collective data seen by all sub-networks, each can prioritize important samples by directly training them, while also benefiting from diverse data exposure through indirect supervision. This not only reduces the overall training cost but also preserves model performance by ensuring sufficient coverage of the dataset across bit-widths.

## NeurIPS Paper Checklist

### 1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [Yes]

Justification: The abstract clearly states the two primary contributions (the gradient-based bit-wise coresampling and the bias correction method), which align with the methods and results presented in the paper, accurately reflecting its contributions and scope.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

### 2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: Due to page constraints, we placed our discussion of limitations in the Conclusion section.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

### 3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: The methodology section simply outlines the observations that inspired our core idea.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

#### 4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper thoroughly documents the experimental setup, including descriptions of the datasets used, model architecture details, training procedures. These details provide all the information needed to reproduce the key results.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

## 5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [\[Yes\]](#)

Justification: We will release the code and necessary data along with documentation and instructions, enabling others to reproduce the primary experimental results.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

## 6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [\[Yes\]](#)

Justification: Our paper specifies all relevant training details, including dataset, and tuning ranges, model architectures. Information that could not be included due to length can be found in the code. This comprehensive reporting ensures that readers have all information needed to replicate the experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

## 7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [\[Yes\]](#)

Justification: For the CIFAR-10 and CIFAR-100 experiments, we report error bars representing the standard deviation over three independent runs, capturing variability due to initialization and data ordering.

Guidelines:

- The answer NA means that the paper does not include experiments.

- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

#### 8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [\[Yes\]](#)

Justification: Our paper discusses computational costs by reporting GPU hours for training and highlights that our approach substantially reduces this cost compared to baselines. It also specifies the hardware environment (noting the GPUs used for training), giving a clear sense of the resources required for reproduction.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

#### 9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [\[Yes\]](#)

Justification: Our work adheres to ethical guidelines. Our work involves only widely-used public datasets and standard methodology with no human subjects or privacy issues. We confirm compliance with the NeurIPS Code of Ethics, as there are no aspects of the research that conflict with ethical standards.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

#### 10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: Our paper's contribution is a general method for improving training efficiency, with no obvious negative societal impact. Thus, no specific societal impacts are identified, beyond the positive aspect of reducing computational resource usage.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

## 11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: Our research does not involve releasing any high-risk model. It introduces a training framework, so no special safeguards are necessary or applicable.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

## 12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: Our paper uses only established public datasets and tools, and it properly cites the sources. All assets are used in accordance with their licenses and terms of use, with appropriate references given to the original creators.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, [paperswithcode.com/datasets](https://paperswithcode.com/datasets) has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

### 13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: Our work does not introduce new datasets or standalone model assets; it focuses on a training algorithm and uses existing benchmark datasets. Therefore, this question is not applicable, aside from the planned code release which is documented and covered under reproducibility above.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

### 14. Crowdsourcing and research with human subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: No human subjects or crowdworkers were involved in our research. Our study exclusively uses pre-existing machine learning datasets, so no participant instructions or compensation details are applicable.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

**15. Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: Our study did not involve any human participants or personal data, so IRB approval was not required.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

**16. Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [NA]

Justification: LLMs were used only for sentence-level editing and grammar checking during manuscript preparation and did not influence or form part of the core technical methodology.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.