Quantization-Aware Training and Knowledge Distillation: Theory and Implementation

Abstract

Quantization-Aware Training (QAT) is a powerful technique for optimizing neural networks for deployment on resource-constrained devices by simulating low-precision computations during training. This paper provides a comprehensive theoretical analysis of QAT, QAT combined with Knowledge Distillation (QAT+KD), and QAT with Entropy-Aware Knowledge Distillation (QAT+EntKD). We explore the mathematical foundations, including quantization mappings, gradient approximations, and loss functions, with detailed explanations to ensure accessibility. We then discuss practical implementations using PyTorch, highlighting key components from provided codebases. Finally, we analyze experimental results on the CIFAR-100 dataset, demonstrating the efficacy of these methods in improving accuracy and reducing latency compared to full-precision models. This work aims to bridge theoretical rigor with practical deployment considerations.

1 Introduction

Deep neural networks (DNNs) have achieved remarkable success across various domains, from image classification to natural language processing. However, their computational complexity and memory requirements often make them impractical for deployment on resource-constrained edge devices, such as mobile phones or embedded systems. Optimization techniques like quantization, pruning, and knowledge distillation address these challenges by reducing model size and computational overhead while striving to maintain accuracy.

Quantization-Aware Training (QAT) is a leading approach that simulates low-precision arithmetic during training to produce models robust to quantization errors. By integrating quantization effects into the training process, QAT minimizes accuracy degradation when deploying models in low-precision formats, such as 8-bit integers (INT8). Knowledge Distillation (KD) enhances this by leveraging a high-capacity teacher model to guide a smaller student model, improving generalization. Entropy-Aware Knowledge Distillation (EntKD) further refines KD by dynamically adjusting the distillation process based on the teacher's output uncertainty, as measured by entropy.

This paper provides a rigorous theoretical framework for QAT, QAT+KD, and QAT+EntKD, emphasizing mathematical foundations and their implications. We then detail practical implementations using PyTorch, drawing on provided codebases, and discuss experimental results on CIFAR-100.

2 Theoretical Foundations

This section explores the mathematical underpinnings of QAT, QAT+KD, and QAT+EntKD, with detailed explanations to make the concepts accessible to readers with varying levels of expertise.

2.1 Quantization-Aware Training (QAT)

QAT simulates low-precision arithmetic during training to produce models that perform well when quantized for inference. The goal is to map high-precision floating-point values (typically FP32) to low-precision integers (e.g., INT8) while minimizing accuracy loss.

2.1.1 Quantization Mapping

Quantization involves mapping a floating-point value $x \in \mathbb{R}$ to a b-bit integer $q \in \{q_{\min}, \dots, q_{\max}\}$, where typically $q_{\min} = -2^{b-1}$ and $q_{\max} = 2^{b-1} - 1$ for signed integers (e.g., $[-128, 127]$ for 8-bit). The affine quantization mapping is defined as:

$$x = s(q - z),$$

where $s > 0$ is the scale factor, and $z \in \mathbb{Z}$ is the zero-point, which maps the floating-point value 0 to an integer. The quantization process is the inverse:

$$q = \text{clip}\left(\left\lfloor \frac{x}{s} \right\rceil + z, q_{\min}, q_{\max}\right),$$

where $\text{clip}(v, a, b) \equiv \max(a, \min(v, b))$ ensures q stays within the valid integer range, and $\lfloor - \rceil$ denotes rounding to the nearest integer.

To determine s and z, we map the floating-point range $[x_{\min}, x_{\max}]$ to the integer range $[q_{\min}, q_{\max}]$. Solving the linear system:

$$\begin{aligned} x_{\min} &= s\left(q_{\min} - z\right), \\ x_{\max} &= s\left(q_{\max} - z\right), \end{aligned}$$

yields:

$$\begin{aligned} s &= \frac{x_{\max} - x_{\min}}{q_{\max} - q_{\min}}, \\ z &= \text{round}\left(q_{\min} - \frac{x_{\min}}{s}\right). \end{aligned}$$

For example, for a tensor with $x_{\min} = -1.0, x_{\max} = 1.0$, and 8-bit quantization ($q_{\min} = -128, q_{\max} = 127$)

$$\begin{aligned} s &= \frac{1.0 - (-1.0)}{127 - (-128)} = \frac{2.0}{255} \approx 0.007843, \\ z &= \text{round}\left(-128 - \frac{-1.0}{0.007843}\right) = \text{round}(-128 + 127.5) \approx 0. \end{aligned}$$

This mapping ensures that floating-point values are approximated by discrete integer levels, reducing memory and computational requirements.

2.1.2 Fake Quantization in QAT

During QAT, 'fake quantization' simulates the effect of quantization in the forward pass while maintaining full-precision weights for gradient updates. For a weight or activation x, the fake quantization operation is:

$$x_{\text{fake}} = s\left(\text{clip}\left(\left\lfloor \frac{x}{s} \right\rceil + z, q_{\min}, q_{\max}\right) - z\right).$$

This operation ensures x_{fake} lies on the same discrete lattice as an INT8 tensor but remains in floating-point for compatibility with training. The backward pass uses the Straight-Through Estimator (STE) to handle the non-differentiable rounding function:

$$\frac{\partial L}{\partial x} \approx \frac{\partial L}{\partial x_{\text{fake}}},$$

where L is the loss. The STE treats the quantization as an identity function for gradients, allowing standard optimizers like SGD or Adam to update weights.

2.1.3 Loss Function

The loss function in QAT is typically the standard task-specific loss, such as cross-entropy for classification:

$$L = -\sum_{i=1}^{C} y_i \log\left(\hat{y}_i\right),$$

where y_i is the true label, $\hat{y}_i$ is the predicted probability, and C is the number of classes. The quantization noise introduced in the forward pass influences the loss, and the optimizer adjusts weights to minimize both task error and quantization error.

2.2 QAT with Knowledge Distillation (QAT+KD)

Knowledge Distillation (KD) enhances QAT by using a pre-trained, high-capacity teacher model to guide the quantized student model. The teacher provides 'soft' predictions, which encode richer information about class similarities compared to hard labels.

2.2.1 KD Loss Function

The KD loss combines the standard cross-entropy loss with a Kullback-Leibler (KL) divergence term to align the student's predictions with the teacher's:

$$L_{\text{KD}} = \alpha L_{\text{CE}}\left(p_s, y\right) + (1-\alpha)T^2 \text{KL}\left(\text{softmax}\left(\frac{p_s}{T}\right) \parallel \text{softmax}\left(\frac{p_t}{T}\right)\right),$$

where:

- p_s and p_t are the student and teacher logits, respectively,
- y is the true label,
- T is the temperature parameter that softens the softmax distributions,
- $\alpha \in [0, 1]$ balances the cross-entropy (L_{CE}) and KL terms,
- The factor T^2 scales the KL divergence to account for the softened distributions.

The softmax function is:

$$\text{softmax}\left(z_i\right) = \frac{\exp\left(z_i/T\right)}{\sum_{j=1}^{T} \exp\left(z_j/T\right)}.$$

A higher T produces softer probabilities, emphasizing inter-class relationships. For example, if a teacher predicts probabilities $[0.7, 0.2, 0.1]$ for classes A, B, and C, the student learns not only that A is the most likely but also that B is more similar to A than C is.

2.2.2 Why KD Helps QAT

Quantization introduces noise that can degrade model performance. KD mitigates this by providing a regularizing effect: the teacher's soft labels guide the student to focus on the most informative features, reducing overfitting to quantization noise. Mathematically, the KL divergence minimizes the difference in probability distributions:

$$\mathrm{KL}\left(p_s \| p_t\right) = \sum_{i=1}^{C} p_s(i) \log\left(\frac{p_s(i)}{p_t(i)}\right).$$

This encourages the student to mimic the teacher's behavior, improving robustness to low-precision constraints.

2.3 QAT with Entropy-Aware Knowledge Distillation (QAT+EntKD)

Entropy-Aware KD (EntKD) builds on QAT+KD by dynamically adjusting the temperature T based on the teacher's output entropy, which measures prediction uncertainty.

2.3.1 Teacher Entropy

The entropy of the teacher's softmax probabilities p_t is:

$$H\left(p_t\right) = -\sum_{i=1}^{C} p_t(i) \log\left(p_t(i) + \varepsilon\right),$$

where $\varepsilon = 10^{-10}$ prevents numerical issues. High entropy indicates uncertainty (e.g., uniform probabilities $[0.33, 0.33, 0.33]$), while low entropy indicates confidence (e.g., $[0.9, 0.05, 0.05]$).

2.3.2 Dynamic Temperature

EntKD adjusts the temperature as:

$$T = \frac{T_{\text{base}}}{1 + \beta H\left(p_t\right)},$$

where T_{base} is the base temperature (e.g., 3.0), and $\beta > 0$ (e.g., 0.1) controls the sensitivity to entropy. The temperature is clamped to $[1.0, 10.0]$ to avoid extreme values. When entropy is high, T decreases, sharpening the teacher's probabilities to focus the student on the most likely classes, reducing the impact of quantization noise in uncertain cases. When entropy is low, T increases, softening the teacher's probabilities to provide more generalized guidance, enhancing inter-class learning.

2.3.3 Loss Function

The EntKD loss is similar to KD but uses the dynamic temperature:

$$L_{\text{EntKD}} = \alpha L_{\text{CE}}\left(p_s, y\right) + (1-\alpha) T^2 \mathrm{KL}\left(\text{softmax}\left(\frac{p_s}{T}\right) \| \text{softmax}\left(\frac{p_t}{T}\right)\right).$$

The dynamic temperature adapts the distillation process to the teacher's confidence, improving the student's ability to handle ambiguous cases.

2.3.4 Why EntKD Improves Performance

EntKD enhances QAT+KD by tailoring the distillation process to the teacher's uncertainty. High-entropy outputs indicate cases where the teacher is less certain, and a lower temperature sharpens the teacher's probabilities, helping the student focus on reliable predictions and avoid overfitting to noisy teacher outputs. This is particularly beneficial in QAT, where quantization noise can exacerbate errors in uncertain regions.

3 Implementation

This section details the key components of the provided PyTorch implementations for QAT, QAT+KD, and QAT+EntKD, applied to a ResNet50 model on the CIFAR-100 dataset with 32×32 images.

3.1 Data Loading and Preprocessing

All three implementations use the same data loading function to prepare CIFAR-100 data:

- Transforms: Training includes random cropping, horizontal flipping, and normalization with mean $\mu = (0.5071, 0.4867, 0.4408)$ and standard deviation $\sigma = (0.2675, 0.2565, 0.2761)$. Testing uses only normalization.
- Splitting: 10% of the training set is used for validation via random shuffling with a fixed seed (42).
- DataLoaders: Batch sizes are 128 (training), 256 (validation), and 8 (testing), with 4 worker threads and pinned memory for efficiency.

3.2 QAT Implementation

The QAT implementation (Code 1) includes:

- Model Setup: ResNet50 is modified for 32×32 inputs by replacing the first convolutional layer (Conv2d $(3, 64, 7, 2, 3) \rightarrow$ Conv2d $(3, 64, 3, 1, 0)$) and removing maxpooling. The model is fused for quantization compatibility and configured with the flogemm QAT qconfig.
- Calibration: Observers are enabled for 200 batches to estimate activation ranges, after which fake quantization is activated.
- Training: The model is trained for up to 200 epochs with AdamW ($lr = 10^{-3}$, weight decay $= 10^{-4}$), cosine annealing LR scheduling, and cross-entropy loss with label smoothing (0.1). Early stopping is triggered after 20 epochs without validation accuracy improvement.
- Conversion: Post-training, the model is converted to INT8 using torch.quantization.convert.
- Evaluation: Accuracy and latency are measured for both fake-quantized (GPU) and INT8 (CPU) models.

3.3 QAT+KD Implementation

The QAT+KD implementation (Code 2) extends QAT by incorporating KD:

- Teacher Model: A pre-trained FP32 ResNet50, modified similarly for 32×32 inputs, is loaded from a checkpoint.
- Loss Function: A custom KDLoss combines cross-entropy ($\alpha = 0.7$) with KL divergence ($T = 3.0$):

$$L = \alpha L_{\mathrm{CE}} + (1-\alpha)T^2 \mathrm{KL}\left(\log \mathrm{softmax}\left(\frac{p_s}{T}\right) \| \, \mathrm{softmax}\left(\frac{p_t}{T}\right)\right).$$

- Training: The student model follows the same QAT pipeline, but the loss incorporates teacher outputs computed in a no-grad context.

3.4 QAT+EntKD Implementation

The QAT+EntKD implementation (Code 3) further refines QAT+KD:

- Entropy-Aware Loss: A custom EntKDLoss computes teacher entropy and adjusts the temperature dynamically:

$$T = \frac{T_{\mathrm{base}}}{1 + \beta H\left(p_t\right)}, \quad H\left(p_t\right) = -\sum_i p_t(i) \log\left(p_t(i) + 10^{-10}\right).$$

  Parameters are $T_{\mathrm{base}} = 3.0, \alpha = 0.3, \beta = 0.1$, with T clamped to $[1.0, 10.0]$.

- Training: The pipeline mirrors QAT+KD, but the loss uses the dynamic temperature to adapt distillation based on teacher uncertainty.

3.5 Evaluation Function

All implementations use a common evaluate accuracy and latency function:

- Accuracy is computed as the percentage of correct predictions.
- Latency is measured as the average time (ms) per batch.
- Fake-quantized models are evaluated on GPU, and INT8 models are evaluated on CPU.

4 Results

The experimental results compare QAT, QAT+KD, QAT+EntKD, and the baseline FP32 model on CIFAR-100, with additional experiments incorporating CutMix data augmentation to assess the robustness of these methods under enhanced training conditions.

4.1 Analysis

- Accuracy without Augmentation: QAT+EntKD achieves the highest accuracy (74.78%) among non-augmented methods, surpassing QAT+KD (73.90%), QAT (73.67%), and the FP32 baseline (72.05%). The +2.73% improvement over FP32 underscores the effectiveness of combining quantization with entropy-aware distillation. QAT alone improves accuracy by 1.62%, indicating that quantization noise can act as a regularizer. KD adds a further 0.23% by leveraging teacher guidance, and EntKD boosts this by 0.88% through adaptive temperature adjustment, demonstrating its ability to adapt to teacher uncertainty.

Table 1: Performance Comparison on CIFAR-100

Method	Top-1 Accuracy (%)	Δ Accuracy vs FP32 (%)	INT8 Latency (ms/batc
FP32	72.05	–	261.00
QAT	73.67	+1.62	130.00
QAT+KD	73.90	+1.85	130.00
QAT+EntKD	74.78	+2.73	130.00
FP32 + CutMix	76.69	–	267.59
QAT+EntKD + CutMix	78.40	+1.71	130.78

- Accuracy with CutMix Augmentation: Incorporating CutMix significantly boosts the FP32 baseline to 76.69%, reflecting the power of data augmentation. Remarkably, QAT+EntKD + CutMix achieves the highest overall accuracy of 78.40%, a +1.71% improvement over FP32 + CutMix. This result highlights the consistent superiority of QAT+EntKD, as it maintains a significant accuracy gain even when paired with advanced augmentation techniques.

- Consistency of QAT+EntKD: QAT+EntKD delivers top performance both with and without CutMix, achieving the highest accuracy in both settings (74.78% without augmentation and 78.40% with CutMix). The dynamic temperature adjustment in EntKD allows the student model to effectively handle teacher uncertainty, making it robust to quantization noise and complementary to data augmentation. This consistency underscores QAT+EntKD's versatility for resource-constrained environments, where both quantization and augmentation may be employed.

- Latency: All quantized models (QAT, QAT+KD, QAT+EntKD, QAT+EntKD + CutMix) exhibit significantly reduced latency ($\sim$ 130 ms/batch) compared to FP32 models (261–267 ms/batch), reflecting the efficiency of INT8 operations. The slight increase in latency for QAT+EntKD + CutMix (130.78 ms) is negligible and likely due to minor computational overhead from augmentation.

- Impact of EntKD: The dynamic temperature in EntKD enhances generalization by adapting to the teacher's prediction confidence. The lower $\alpha = 0.3$ in QAT+EntKD prioritizes the KL divergence term, emphasizing teacher knowledge, which proves effective in both standard and augmented settings.

- Trade-offs: QAT+EntKD and QAT+EntKD + CutMix offer the best accuracy but require a pre-trained teacher and additional computational overhead for entropy calculation. QAT alone is simpler but less accurate. The addition of CutMix increases training complexity but significantly boosts accuracy, making QAT+EntKD + CutMix ideal for scenarios where maximum performance is prioritized.

5 Conclusion

QAT, QAT+KD, and QAT+EntKD offer powerful methods for optimizing neural networks for edge deployment. QAT leverages fake quantization and STE to produce robust low-precision models. KD enhances this by incorporating teacher guidance, and EntKD further improves performance by adapting to teacher uncertainty. Experimental results on CIFAR-100 demonstrate that QAT+EntKD achieves the highest accuracy (74.78% without augmentation, 78.40% with CutMix) with significantly reduced latency, making it ideal for resource-constrained environments. The consistent performance of QAT+EntKD across both standard and augmented settings highlights its robustness and versatility. Future work could explore combining these methods with pruning or advanced quantization schemes to further enhance efficiency.

6 Suggested Reading

This section provides a curated list of academic papers and blog posts for further reading on Quantization-Aware Training (QAT), Knowledge Distillation (KD), and Entropy-Aware Knowledge Distillation (EntKD).

6.1 Academic Papers

1. QKD: Quantization-aware Knowledge Distillation Jangho Kim, et al. arXiv:1911.12491, 2019. Proposes a three-phase approach (Self-studying, Co-studying, Tutoring) to combine QAT and KD, improving performance on datasets like ImageNet and CIFAR-100.

2. Knowledge From the Dark Side: Entropy-Reweighted Knowledge Distillation arXiv:2311.13621, 2023. Introduces Entropy-Reweighted KD, which uses teacher prediction entropy to reweight the KD loss, emphasizing challenging samples.

3. Understanding and Improving Knowledge Distillation for Quantization-Aware Training of Large Transformer Encoders Minsoo Kim, et al. ACL Anthology, 2022. Analyzes KD in QAT for Transformer encoders like BERT, proposing attention-map and attention-output losses to enhance quantized model performance.

4. A survey on knowledge distillation: Recent advancements ScienceDirect, 2024. Provides a comprehensive overview of KD methods, categorizing them into traditional and novel paradigms, and discussing integration with quantization.

5. EntKD: Entropy-Aware Knowledge Distillation this work, 2025. Introduces EntKD, a novel method that dynamically adjusts the KD temperature based on teacher entropy, achieving superior accuracy in quantized settings.

6.2 Blog Posts

1. Quantization for Neural Networks Lei Mao. https://leinao.github.io/article/Neural-Networks-Qu Explains the basics of neural network quantization, including uniform and non-uniform quantization techniques.

2. Neural Network Quantization in PyTorch Arik Poznanski. https://medium.com/@arikpoznanski/neural-network-quantization-in-pytorch-52. Provides a practical guide to implementing quantization in PyTorch, with examples and code snippets.

3. Inside Quantization Aware Training Towards Data Science. https://towardsdatascience.com/inside-quantization-aware-training-4f91c8837. Discusses the mechanics of QAT, including fake quantization and its impact on model training.

4. TinyML - Quantization Aware Training Thomas Kevin. https://medium.com/@thomaskevin/tinyml-quantization-aware-training-b4f29odd. Explores QAT in the context of TinyML, focusing on deployment for resource-constrained devices.