# VQT: Vector Quantization Tuning for Efficient Fine-tuning and Compression of Pre-trained Vision Transformers

Yinglong Li, Jiyuan Xia, Jingcheng Xie, and Zhiwei Xiong(✉)

University of Science and Technology of China, China
{yllee, jyxia, icarus0131}@mail.ustc.edu.cn, zwxiong@ustc.edu.cn

**Abstract.** Pre-trained Vision Transformers (ViTs) have shown strong performance across diverse tasks, creating increasing demand for efficient fine-tuning and compression for downstream deployment. Existing Vector Quantization (VQ) methods achieve high compression ratios, yet struggle to balance fine-tuning efficiency and downstream performance. In this work, we propose Vector Quantization Tuning (VQT), a novel framework for efficient fine-tuning and compression of pre-trained ViTs. We identify Coupled Vector Quantization Noise (CVQN) as a key factor limiting performance. It arises from the interaction between Low-Rank Adaptation (LoRA) and the Straight-Through Estimator (STE), which degrades optimization effectiveness under vector quantization. To address this, VQT employs a VQ-aware fine-tuning process that stochastically replaces vector-quantized weights with continuous counterparts to reduce CVQN, while progressively aligning the model toward fully vector-quantized representations to ensure consistency between fine-tuning and deployment. Furthermore, VQT enables efficient storage for multi-task deployment by storing only lightweight task-specific parameters in the cloud, with less than 1 bit per parameter needed to reconstruct compressed models. Extensive experiments show the advantages of VQT over state-of-the-art baselines in performance and efficiency at high compression ratios. For example, at 1-bit compression, VQT surpasses the advanced baseline by 9.5% on the VTAB-1K benchmark while fine-tuning only 0.8% of the parameters. The code is available: https://github.com/leenas233/VQT.

**Keywords:** Vector quantization · Parameter-efficient fine-tuning · Network compression

## 1 Introduction

Vision Transformers (ViTs) pre-trained on large-scale datasets exhibit strong generalization capability across a wide range of downstream tasks. Empirically, increasing model parameters can consistently improve representational power and performance, making large ViTs particularly effective. To efficiently fine-tune such models to task-specific scenarios, Parameter-Efficient Fine-Tuning (PEFT) methods [16, 17, 20, 27] have become a practical and widely adopted solution.

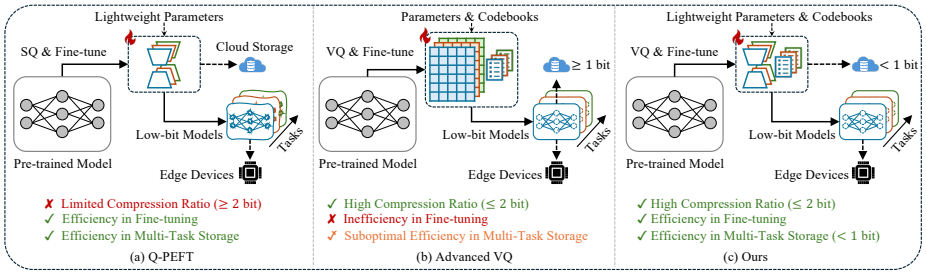



**Fig. 1:** (a) Quantized PEFT (Q-PEFT) achieves efficient fine-tuning but limited compression. (b) Advanced Vector Quantization (VQ) attains high compression at the cost of fine-tuning and multi-task storage efficiency. (c) Our VQT delivers both high compression and efficient fine-tuning with lightweight multi-task storage.

Despite the fine-tuning efficiency of PEFT, fine-tuned models still retain the original size, limiting their deployment on resource-limited edge devices.

Building on PEFT, Quantized Parameter-Efficient Fine-Tuning (Q-PEFT) methods [2, 8, 22, 40] combine quantization with task-specific fine-tuning. They enable both model quantization (*i.e.*, compression) and efficient fine-tuning, allowing task-specific compressed models to be obtained efficiently within a unified training procedure (*i.e.*, single-stage process), without requiring a separate post-training compression procedure after fine-tuning a pre-trained model on a downstream task. Nevertheless, as shown in Fig. 1(a), these methods focus on Scalar Quantization (SQ), which struggles to maintain performance at extremely low bit-widths (*e.g.*, below 2 bits), *i.e.*, high compression ratios.

Unlike SQ, Vector Quantization (VQ) can achieve extremely high compression ratios. Early VQ methods [33] typically cluster continuous weight sub-vectors to the nearest codeword in a codebook and then fine-tune the codebook using task-specific loss. However, updating only the codebook limits task-specific fine-tuning, as sub-vectors sharing a codeword are constrained to identical update directions [26]. Codeword reassignment can unleash fine-tuning potential. Advanced VQ methods [5–7,26] achieve this via soft assignments or splitting sign bits, but, as shown in Fig. 1(b), they rely on a large number of independently learnable parameters or on knowledge distillation that can only be applied in a post-training procedure, lacking efficiency in fine-tuning and multi-task storage. Designing VQ methods that support efficient fine-tuning and compression within a unified training procedure for downstream tasks remains challenging.

In this work, we propose **V**ector **Q**uantization **T**uning (VQT), a novel framework for efficient fine-tuning and compression of pre-trained ViTs, as shown in Fig. 1(c). We first analyze the compatibility between Low-Rank Adaptation (LoRA) and VQ, where both LoRA parameters and codebooks are fine-tuned using task loss, followed by codeword reassignment via nearest-neighbor selection for each LoRA-updated sub-vector. Since this codeword reassignment process is discrete and non-differentiable, the Straight-Through Estimator (STE) [1] is commonly used to approximate gradients [10, 18]. However, we observe that the

interdependence among LoRA parameters amplifies the gradient approximation error introduced by STE, producing an accumulated disturbance termed Coupled Vector Quantization Noise (CVQN). CVQN perturbs LoRA updates and leads to incorrect codeword reassignments, degrading downstream performance. To address this issue, we design a VQ-aware fine-tuning process which jointly fine-tunes LoRA parameters and codebooks. Instead of directly optimizing under fully vector-quantized representations, VQT uses a stochastic sub-vector replacement strategy that partially restores continuous representations during fine-tuning, mitigating CVQN to enable stable LoRA updates and codeword reassignments. Meanwhile, VQT progressively aligns the intermediate representations with their fully vector-quantized counterparts, preventing optimization mismatch between fine-tuning and deployment. Furthermore, VQT stores only lightweight task-specific parameters rather than full compressed models for each downstream task, substantially reducing multi-task storage overhead.

We evaluate VQT on various pre-trained ViTs, including ViT-B [11], ViT-L [11], and SAM-H [24], across diverse downstream tasks. Extensive experiments show that VQT consistently outperforms state-of-the-art methods at high compression ratios while fine-tuning and storing only lightweight task-specific parameters for multiple downstream tasks.

The contributions of this work are summarized as follows:

1) We analyze the compatibility between LoRA and VQ and identify CVQN, which perturbs LoRA updates and causes incorrect codeword reassignments.

2) We propose VQT, a novel framework for efficient fine-tuning and compression of pre-trained ViTs, which employs stochastic sub-vector replacement to mitigate CVQN and progressive alignment to prevent optimization mismatch.

3) We design an efficient storage strategy for multi-task deployment based on VQT, storing only a lightweight set of necessary parameters instead of full compressed models for each downstream task.

4) Extensive experiments across diverse downstream tasks and pre-trained ViTs show the effectiveness of VQT in efficient fine-tuning and compression.

## 2 Related Works

**Pre-trained vision transformers.** ViTs [11] pre-trained on large-scale datasets have become powerful foundation models for visual representation learning. Representative models such as ViT [11], SAM [24], CLIP [36], and DINOv2 [35] show strong generalization across downstream tasks, where fine-tuning often achieves state-of-the-art performance [4, 20, 27, 43] without training task-specific models from scratch. However, their large parameter sizes make fine-tuning and deployment costly, motivating efficient fine-tuning and compression methods.

**Parameter-efficient fine-tuning.** PEFT methods [3, 17, 20, 27, 30, 44] adapt large-scale pre-trained models to downstream tasks by updating only a small subset of parameters. Among them, LoRA [17] fine-tunes lightweight low-rank matrices and merges them into pre-trained weights to reduce training cost. However, standard PEFT methods do not reduce the model size at deployment, lim-

iting their application on resource-constrained edge devices. Recent Quantized PEFT (Q-PEFT) methods [2,8,22,37,40] combine PEFT with quantization techniques [9, 14, 28]. These approaches typically quantize pre-trained weights while fine-tuning lightweight parameters, such as LoRA parameters or quantization scaling factors, in a unified training procedure. Nevertheless, these methods primarily rely on Scalar Quantization (SQ) and consequently suffer from significant performance degradation at extremely low bit-widths (*e.g.*, 2 bits or below).

**Vector quantization.** VQ compresses models by clustering weight sub-vectors into codebook codewords and storing their assignments. PQF [33] reorders weight dimensions to reduce clustering errors and fine-tuning the codebook with respect to the target loss, but it keeps codeword assignments fixed. VQ4DiT [6] and ViM-VQ [7] leverage soft assignments over candidate codewords to improve codeword reassignment, but they rely on post-training procedures following the knowledge distillation paradigm and introduce many learnable parameters. SSVQ [26] splits and fine-tunes the sign bits independently from the codebook, allowing sub-vectors to update in different directions during fine-tuning, which essentially achieves codeword reassignment through sign changes, but its large number of independent learnable parameters still limits fine-tuning efficiency. In this work, we focus on using VQ to achieve efficient fine-tuning and compression.

## 3 Preliminaries and Motivation

### 3.1 Preliminaries

**Low-rank adaptation.** LoRA [17] is a representative PEFT method with many variants [29, 30, 42]. The basic LoRA is widely used and defined as:

$$W = W_0 + \alpha \cdot BA, \tag{1}$$

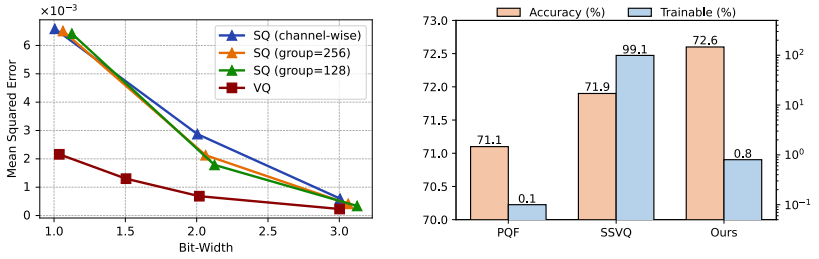
where $B \in \mathbb{R}^{M\times r}$ and $A \in \mathbb{R}^{r\times N}$ are trainable LoRA parameters with $r \ll \min(M, N)$, $W_0 \in \mathbb{R}^{M\times N}$ is the frozen pre-trained weight, $W \in \mathbb{R}^{M\times N}$ is the fine-tuned weight, and $\alpha$ is a scalar. The weight update at epoch $(t+1)$ is:

$$\begin{aligned} W_{t+1} &= W_0 + \alpha \cdot (B_t - \eta \cdot \alpha \cdot G_{W_t} A_t^T)(A_t - \eta \cdot \alpha \cdot B_t^T G_{W_t}) \\ &= W_t \underbrace{-\eta\alpha^2(B_t B_t^T G_{W_t} + G_{W_t} A_t^T A_t - \eta\alpha G_{W_t} A_t^T B_t^T G_{W_t})}_{\Delta W_t}, \end{aligned} \tag{2}$$

where $\eta$ is the learning rate, $G_{W_t}$ is the gradient *w.r.t.* $W_t$, and $\Delta W_t$ denotes the weight update at $t$-th epoch during fine-tuning.

**Codebook and assignments in vector quantization.** Let $W \in \mathbb{R}^{M\times N}$ denote a continuous uncompressed weight matrix of a ViT linear layer, partitioned into $\frac{MN}{d}$ sub-vectors $w_i \in \mathbb{R}^{1\times d}$, forming $\mathcal{W} = \{w_i\}_{i=1}^{\frac{MN}{d}}$. VQ constructs a codebook $\mathcal{C} = \{c_k\}_{k=1}^{K}$ by clustering these sub-vectors (*e.g.*, K-Means [15]) into $K$ centroids (*i.e.*, codewords). Each sub-vector is assigned to its nearest codeword, producing the assignment set $\mathcal{A}$ and the vector-quantized sub-vectors $\hat{\mathcal{W}}$:

$$\mathcal{A} = \{a_i \mid a_i = \arg\min_k \|w_i - c_k\|^2\}_{i=1}^{\frac{MN}{d}}, \quad \hat{\mathcal{W}} = \{\hat{w}_i \mid \hat{w}_i = c_{a_i}\}_{i=1}^{\frac{MN}{d}}. \tag{3}$$




**(a)** MSE of SQ and VQ **(b)** Trade-off of VQ methods

**Fig. 2:** (a) VQ consistently achieves lower MSE than SQ, especially at very low bitwidths, where SQ (group=128/256) applies finer-grained partitioning along input dimensions [14] than channel-wise SQ. (b) Evaluated on ViT-B [11] with VTAB-1K [41], reporting average accuracy at 2 bits. "Train (%)" denotes the ratio of trainable parameters. Our method surpasses VQ baselines with lightweight fine-tuned parameters.

When deployed on edge devices, the compressed model consists of codebook $\mathcal{C}$ and the assignments $\mathcal{A}$. Let $b_w$ and $b_c$ denote the bit-widths of the original weights and codewords, respectively. The compression ratio ($CR$) and the average bit-width per weight value ($Bit$) are given as follows:

$$CR = \frac{MNb_w}{M\frac{N}{d}\log_2 K + Kdb_c}, \quad Bit = \frac{b_w}{CR} = \frac{\log_2 K}{d} + \frac{Kdb_c}{MN}. \tag{4}$$

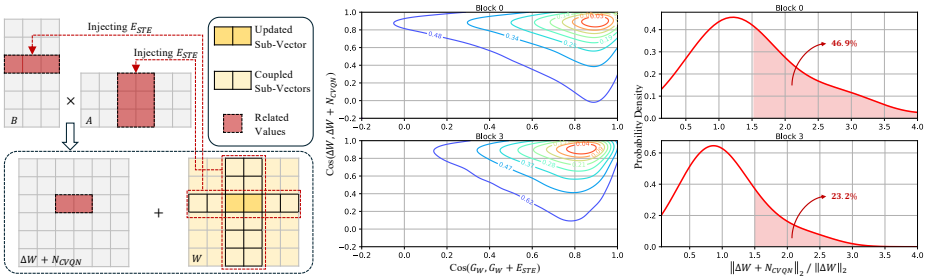
At inference, $\hat{W}$ is reconstructed by indexing $\mathcal{C}$ with $\mathcal{A}$.

In this work, we update both the continuous $W$ (with LoRA) and the codebook $\mathcal{C}$ by minimizing the task loss, and then reassign codewords by selecting the nearest codeword to each LoRA-updated continuous sub-vector of $\mathcal{W}$ according to Eq. 3. This guides codeword reassignment toward the task-specific objective.

### 3.2 Motivation

Efficient fine-tuning and model compression facilitate the use of pre-trained ViTs on resource-limited devices across diverse downstream scenarios. Q-PEFT methods [2, 22, 40] perform fine-tuning and quantization of pre-trained models in a unified training procedure, *i.e.*, single-stage process without a separate post-training procedure. By fine-tuning only lightweight learnable parameters, these methods incur low overhead (*e.g.*, GPU memory usage) and enable efficient storage for multiple tasks: only task-specific lightweight parameters need to be stored in the cloud and be loaded to reconstruct the quantized model for different downstream tasks. However, these methods typically rely on SQ, which incurs large errors at very low bit-widths (*e.g.*, 2 bits or below), as shown in Fig. 2a, limiting compression ratios. In contrast, VQ yields much lower errors in extreme low bit-widths, which is more suitable for high-compression scenarios.

To enable effective downstream fine-tuning, VQ must balance the fine-tuning efficiency and the capacity of weight sub-vector updates, corresponding to fine-tuning overhead and downstream performance. As shown in Fig 2b, existing VQ




**(a)** Coupled LoRA updates **(b)** Distortion of update directions (left) and magnitudes (right)

**Fig. 3:** (a) LoRA updates for a sub-vector are affected by STE-induced errors ($E_{STE}$) from all its coupled sub-vectors. (b) LoRA updates amplify STE-induced errors, producing Coupled Vector Quantization Noise ($E_{CVQN}$), which distorts the update directions and magnitudes. **Left:** Contour plots of the joint probability distribution of update direction distortions, with numbers indicating cumulative probability of enclosed regions, where $G_W^{\text{approx}} \coloneqq G_W + E_{STE}$. **Right:** Probability density of update magnitude distortions under negligible STE-induced errors ($\|E_{STE}\|_2/\|G_W\|_2 < 0.001$). Observations are from the 1st and 4th blocks of ViT-B fine-tuned on VTAB-1K [41].

methods struggle to achieve this balance. Early VQ methods, such as PQF [33], use fixed assignments and only fine-tune codebooks, which is parameter-efficient but restricts sub-vector flexibility, limiting performance. Advanced VQ methods [6,7,26], such as VQ4DiT [6] or SSVQ [26], employ soft assignments or split sign bits to support codeword reassignment, increasing the flexibility of weight sub-vector updates. However, these designs inevitably introduce a large number of independently learnable parameters and complex fine-tuning strategies, substantially increasing overhead and reducing efficiency. In addition, these advanced VQ methods are less efficient for multi-task deployment, as they require storing a full task-specific compressed model for each downstream task in the cloud, resulting in higher storage overhead compared to the lightweight parameters used in Q-PEFT methods. Integrating PEFT techniques into VQ offers a more balanced solution, preserving fine-tuning efficiency and capacity while enabling lighter storage for multi-task deployment via task-specific parameters.

## 4 Coupled Vector Quantization Noise

To achieve efficient fine-tuning and compression in a unified training procedure, we incorporate LoRA into VQ by jointly fine-tuning the codebooks and LoRA parameters. During fine-tuning, the LoRA-updated weights $W$ are vector-quantized into $\hat{W}$ via Eq. 3 to enable task-specific codeword reassignments.

Since VQ is inherently non-differentiable and blocks gradient propagation to the LoRA parameters, we adopt the Straight-Through Estimator (STE) [1] to approximate gradients for the LoRA-updated continuous weights $W$. STE is a highly efficient and widely adopted solution for handling non-differentiable

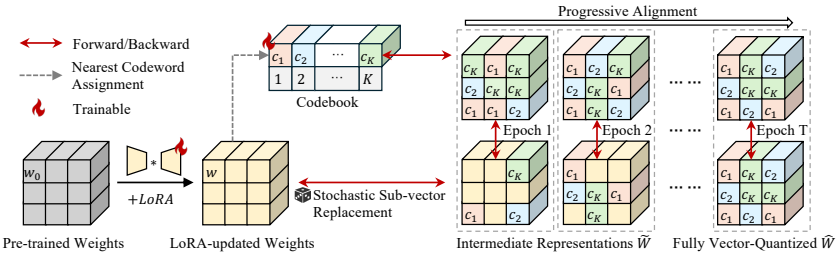



**Fig. 4:** Overview of VQT. It performs VQ-aware fine-tuning with stochastic sub-vector replacement to mitigate the CVQN and progressive alignment to prevent optimization mismatch between fine-tuning and deployment.

quantization in prior works [10, 12, 18] and introduces no additional computational overhead. The approximate gradient of LoRA-updated continuous weights $W$ via STE is estimated as: $G_W^{\text{approx}} := G_{\hat{W}}$, where $G_{\hat{W}}$ is the true gradients of vector-quantized weights $\hat{W}$. The STE-induced error $E_{STE}$ between the approximate gradient $G_W^{\text{approx}}$ and the true gradient $G_W$ of $W$ can be formalized using a Taylor expansion of the loss $\mathcal{L}$ around $\hat{W}$ followed by differentiation:

$$\begin{aligned} \mathcal{L}_W &\approx \mathcal{L}_{\hat{W}} + G_{\hat{W}}(W - \hat{W}) + 0.5 \cdot (W - \hat{W})^T H_{\hat{W}}(W - \hat{W}), \\ G_W^{\text{approx}} &:= G_{\hat{W}} \approx G_W \underbrace{- H_{\hat{W}}(W - \hat{W})}_{E_{STE}}, \end{aligned} \tag{5}$$

where $H_{\hat{W}}$ is the hessian of loss $\mathcal{L}$ with respect to $\hat{W}$. Since $G_W$ is approximated as $G_W^{\text{approx}}$, substituting it into Eq. 2 injects noise in LoRA updates:

$$\begin{aligned} W_{t+1} &= W_t + \Delta W_t + N_{CVQN}, \\ N_{CVQN} &= \eta\alpha^2(B_t B_t^T E_{STE} + E_{STE} A_t^T A_t - \eta\alpha E_{STE} A_t^T B_t^T E_{STE}), \end{aligned} \tag{6}$$

where $N_{CVQN}$ are defined as the Coupled Vector Quantization Noise (CVQN). As illustrated in Fig. 3a, CVQN for each sub-vector arises from $E_{STE}$ propagated from other sub-vectors in the same row or column (coupled sub-vectors). Therefore, the impact of STE-induced error $E_{STE}$ on sub-vector updates is amplified. As shown in Fig. 3b, even when $E_{STE}$ of each sub-vector is small, for example when the cosine similarity between $G_W^{\text{approx}}$ and $G_W$ exceeds 0.9, LoRA amplifies this error, producing $N_{CVQN}$ that distorts update directions, making them deviate more severely than the original directional difference between $G_W^{\text{approx}}$ and $G_W$. This perturbs LoRA updates, causes incorrect codeword reassignments, and ultimately degrades the performance of the compressed model.

## 5 Vector Quantization Tuning

To address the limitations of CVQN, we propose VQT for efficient fine-tuning and high-ratio compression via VQ. We first apply the K-Means [15] to the pre-trained linear-layer weights to initialize the codebook $\mathcal{C}_0$ and assignments $\mathcal{A}_0$. As

shown in Fig. 4, VQT then performs VQ-aware fine-tuning for each downstream task $\tau$ by jointly updating codebooks and LoRA parameters with codeword reassignment, producing a task-specific codebook $\mathcal{C}^{(\tau)}$ and assignments $\mathcal{A}^{(\tau)}$.

### 5.1 Stochastic Sub-vector Replacement

As analyzed above, CVQN arises from STE-induced errors propagated among coupled sub-vectors. Therefore, mitigating CVQN requires restricting the accumulation and propagation of these errors across coupled sub-vectors.

A crucial insight is that STE-induced errors occur only when a sub-vector is vector-quantized. If a sub-vector remains continuous during fine-tuning, its gradients propagate exactly, thereby avoiding STE-induced errors entirely. Based on this observation, we propose a stochastic sub-vector replacement strategy that selectively breaks the coupling causing CVQN. Specifically, during fine-tuning, each weight sub-vector is stochastically represented as either a vector-quantized sub-vector or its continuous counterpart:

$$\tilde{w}_i = z_i\big(w_i + \underbrace{\hat{w}_i - \mathrm{sg}(\hat{w}_i)}_{\text{Continuous-anchored}}\big) + (1 - z_i)\big(\hat{w}_i + \underbrace{w_i - \mathrm{sg}(w_i)}_{\text{VQ-anchored}}\big), \tag{7}$$

where $w_i$ is the LoRA-updated continuous sub-vector, $\hat{w}_i$ is the vector-quantized sub-vector, $\tilde{w}_i$ denotes a sub-vector of the intermediate $\tilde{W}_i$ used during forward and backward passes in fine-tuning, $\mathrm{sg}(\cdot)$ denotes the stop-gradient operator, $z_i \sim \mathrm{Bernoulli}(p)$ indicates whether the $i$-th sub-vector uses the continuous-anchored or vector-quantized-anchored mode, and $p$ is the replacement probability.

The continuous-anchored and VQ-anchored modes play complementary roles. In the continuous-anchored mode, the sub-vector $\tilde{w}_i$ receives the true gradient of $w_i$ to optimize LoRA. A large proportion of sub-vectors in this mode reduces the STE-induced errors from coupled sub-vectors, mitigating the impact of CVQN. Meanwhile, the term $\hat{w}_i - \mathrm{sg}(\hat{w}_i)$ propagates an approximate gradient to the corresponding codeword. Although this gradient still contains STE-induced errors, it only updates the codebook and does not affect LoRA. In the VQ-anchored mode, the $\tilde{w}_i$ receives the true gradient of $\hat{w}_i$ to optimize the codebook. Moreover, its vector-quantized state also allows sub-vectors in the continuous-anchored mode to sense vector quantization effects, enabling VQ-aware fine-tuning.

### 5.2 Progressive Alignment

Although stochastic sub-vector replacement mitigates CVQN, the optimization objective is minimizing $\mathcal{L}_{\tilde{W}}$, where $\tilde{W}$ is composed of the sub-vectors $\tilde{w}_i$ in Eq. 7. According to the descent lemma [34], minimizing $\mathcal{L}_{\tilde{W}}$ still leaves a gap relative to the ultimate goal of minimizing $\mathcal{L}_{\hat{W}}$:

$$\mathcal{L}_{\hat{W}} - \mathcal{L}_{\tilde{W}} \leq G_{\tilde{W}}(\hat{W} - \tilde{W}) + 0.5 \cdot L\|\hat{W} - \tilde{W}\|^2, \tag{8}$$

where $L > 0$ is the smoothness constant and $G_{\tilde{W}}$ denotes the gradient of $\tilde{W}$. By converting the sub-vectors $\tilde{w}_i$ to the fully vector-quantized ones $\hat{w}_i$ during fine-tuning, the optimization process of minimizing $\mathcal{L}_{\tilde{W}}$ can be aligned with the goal

of minimizing $\mathcal{L}_{\hat{W}}$. To realize this, we propose a progressive alignment strategy by annealing the replacement probability $p$ toward zero with a cosine schedule:

$$p_t = p_0 \cdot 0.5 \cdot (1 + \cos(\pi t/T)), \tag{9}$$

where $p_0$ is the initial replacement probability, $t$ is the current epoch, and $T$ is the total number of epochs. This progressive alignment mitigates CVQN and enables sufficient and correct codeword reassignment early in fine-tuning, while stabilizing later stages as sub-vectors transition to the fully vector-quantized ones, thereby aligning the optimization objective with minimizing $\mathcal{L}_{\hat{W}}$.

### 5.3 Efficient Multi-Task Storage

VQT also enables efficient storage for multiple downstream tasks. Since codeword reassignments in VQ-aware fine-tuning use minimum Euclidean distance, task-specific assignments for each task $\tau$ can be generated after fine-tuning as:

$$\mathcal{A}^{(\tau)} = \{a_i^{(\tau)} \mid a_i^{(\tau)} = \arg\min_k \|w_i^{(\tau)} - c_k^{(\tau)}\|^2\}_{i=1}^{\frac{MN}{d}}, \tag{10}$$

where $w_i^{(\tau)} \in \mathcal{W}^{(\tau)}$ is a LoRA-updated sub-vector for task $\tau$. Unlike existing advanced VQ methods [6, 26] that store a full compressed model (*i.e.*, $\mathcal{A}^{(\tau)}$ and $\mathcal{C}^{(\tau)}$) per task, VQT stores only lightweight task-specific LoRA parameters and codebooks in the cloud. Their storage is often smaller than the equivalent storage of a fully compressed 1-bit model. The effective bit-width and compression ratio of these task-specific parameters (TP) relative to the uncompressed model are:

$$CR_{\text{TP}} = MNb_w/((M+N)rb_w + Kdb_c), \quad Bit_{\text{TP}} = b_w/CR_{\text{TP}}, \tag{11}$$

where $r$ is LoRA rank. For edge deployment, the task-specific compressed model can be reconstructed rapidly from the stored LoRA parameters and codebooks using Eq. 10, eliminating the need to store multiple full compressed models.

## 6 Experiments

### 6.1 Experimental Settings

**Datasets, metrics, and pre-trained ViTs.** We evaluate VQT on several representative pre-trained ViTs, including ViT-Base (ViT-B) and ViT-Large (ViT-L) from [11], and SAM-Huge (SAM-H) from [24], using their official pre-trained weights. Experiments covers multiple vision benchmarks, including image classification and semantic segmentation. For image classification, following the widely adopted VTAB-1K protocol [20,27], we evaluate ViT-B and ViT-L on the VTAB-1K benchmark [41], which consists of 19 downstream datasets grouped into *Natural*, *Specialized*, and *Structured* categories. Top-1 accuracy is reported as the evaluation metric. For semantic segmentation, following prior works [4,43], we evaluate SAM-H on four representative downstream scenarios: camouflaged

**Table 1:** Comparison on ViT-B over the VTAB-1K benchmark. "Bit" denotes the target bit-width according to Eq. 4. "Param" denotes the number of trainable parameters. "Average All" represents the average accuracy of all 19 downstream tasks. The best results are highlighted in **bold**. The gray background represents ours.

| | | Natural | | | | | | | | Specialized | | | | | Structured | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Methods | Bit | Cifar100 | Caltech101 | DTD | Flower102 | Pets | SVHN | Sun397 | Average | Camelyon | EuroSAT | Resisc45 | Retinopathy | Average | Clevr-Count | Clevr-Dist | DMLab | KITTI-Dist | dSpr-Loc | dSpr-Ori | sNORB-Azim | sNORB-Ele | Average | Average All | Param (M) |
| Full fine-tuning | FP | 74.3 | 89.2 | 72.8 | 99.3 | 91.2 | 89.2 | 57.7 | 81.9 | 87.1 | 96.0 | 87.0 | 74.1 | 86.1 | 68.1 | 61.5 | 52.9 | 81.4 | 80.7 | 54.5 | 31.2 | 36.7 | 58.4 | 72.9 | 85.80 |
| Linear probing | FP | 64.4 | 85.0 | 63.2 | 97.0 | 86.3 | 36.6 | 51.0 | 69.1 | 78.5 | 87.5 | 68.5 | 74.0 | 77.1 | 34.3 | 30.6 | 33.2 | 55.4 | 12.5 | 20.0 | 9.6 | 19.2 | 26.9 | 53.0 | 0.04 |
| VPT [20] | FP | 78.8 | 90.8 | 65.8 | 98.0 | 88.3 | 78.1 | 49.6 | 78.5 | 81.8 | 96.1 | 83.4 | 68.4 | 82.4 | 68.5 | 60.0 | 46.5 | 72.8 | 73.6 | 47.9 | 32.9 | 37.8 | 55.0 | 69.4 | 0.60 |
| LoRA [17] | FP | 74.9 | 89.2 | 72.1 | 99.2 | 91.4 | 88.7 | 57.3 | 81.8 | 88.6 | 96.4 | 87.7 | 75.2 | 87.0 | 78.3 | 61.4 | 53.1 | 79.3 | 84.7 | 54.6 | 31.1 | 40.4 | 60.4 | 73.9 | 0.59 |
| PEQA [22] | 2 | 60.3 | 90.0 | 68.2 | 98.3 | 87.3 | 88.2 | 46.8 | 77.0 | 80.4 | 95.2 | 83.6 | 71.2 | 82.6 | 72.1 | 60.6 | 47.0 | 78.3 | 80.5 | 52.5 | 30.4 | 39.0 | 57.6 | 70.0 | 0.66 |
| EfficientQAT [2] | 2 | 65.6 | 89.6 | 69.8 | 98.2 | 88.8 | 87.8 | 49.0 | 78.4 | 81.6 | 95.5 | 85.7 | 73.5 | 84.1 | 75.2 | 60.8 | 50.5 | 80.6 | 81.5 | **53.5** | 30.8 | 38.7 | 59.0 | 71.4 | 0.66 |
| PQF [33] | 2 | 66.5 | **90.1** | 71.1 | **99.1** | 90.1 | 83.2 | 51.8 | 78.8 | 84.5 | 95.0 | 84.4 | 74.3 | 84.6 | 75.2 | **64.0** | 47.4 | 79.7 | 77.8 | 51.5 | 27.9 | 38.0 | 57.7 | 71.1 | 0.05 |
| SSVQ [26] | 2 | 67.4 | 89.9 | **72.0** | 98.9 | **90.4** | 86.7 | 52.0 | 79.6 | 84.5 | **95.8** | 84.7 | **74.9** | 85.0 | **76.8** | 63.8 | 48.7 | **81.7** | 79.2 | 52.5 | 28.7 | 38.3 | 58.7 | 71.9 | 85.03 |
| VQT (Ours) | 2 | **70.1** | 89.9 | 71.3 | **99.1** | 89.8 | **88.6** | **53.7** | **80.4** | **86.2** | 95.6 | **86.9** | 74.7 | **85.9** | 73.2 | 60.8 | **50.8** | 81.6 | **82.6** | 52.9 | **31.9** | **39.1** | **59.1** | **72.6** | 0.64 |
| PQF [33] | 1.5 | 43.8 | 87.2 | 64.7 | 97.2 | 85.7 | 75.7 | 36.9 | 70.2 | 78.2 | 94.0 | 79.1 | 72.8 | 81.0 | 71.7 | **63.5** | 39.4 | 74.7 | 73.8 | 45.9 | 26.8 | 37.7 | 54.2 | 65.7 | 0.01 |
| SSVQ [26] | 1.5 | 44.4 | 85.6 | 64.0 | 95.1 | 85.1 | 85.2 | 30.6 | 70.0 | 82.9 | 94.6 | 79.7 | 71.8 | 82.3 | **74.3** | 61.6 | 42.0 | 78.3 | 77.0 | 47.3 | 29.4 | 36.9 | 55.8 | 66.6 | 84.94 |
| VQT (Ours) | 1.5 | **61.5** | **89.4** | **69.1** | **98.2** | **86.9** | **87.5** | **45.9** | **76.9** | **84.2** | **95.2** | **84.6** | **74.0** | **84.5** | 71.0 | 60.4 | **49.0** | **81.4** | **84.4** | **50.5** | **31.6** | **40.5** | **58.6** | **70.8** | 0.60 |
| SSVQ [26] | 1.25 | 26.5 | 66.0 | 53.2 | 76.9 | 47.6 | 81.2 | 14.5 | 52.3 | 79.6 | 92.4 | 73.1 | **74.2** | 79.8 | **70.0** | **60.5** | 41.6 | 65.0 | 76.2 | 44.7 | 27.1 | 31.5 | 52.1 | 58.0 | 84.95 |
| VQT (Ours) | 1.25 | **56.3** | **87.4** | **64.6** | **95.6** | **81.8** | **88.8** | **40.1** | **73.5** | **83.5** | **95.3** | **82.8** | 73.9 | **83.8** | 68.5 | 59.5 | **49.9** | **79.7** | **83.1** | **49.5** | **32.3** | **38.7** | **57.6** | **69.0** | 0.60 |
| PEQA [22] | 1 | 8.9 | 25.9 | 23.2 | 38.6 | 12.5 | 19.6 | 5.2 | 19.1 | 72.9 | 80.6 | 41.7 | 58.0 | 63.3 | 45.3 | 57.3 | 34.0 | 41.5 | 68.6 | 7.6 | 9.0 | 28.0 | 36.4 | 35.7 | 0.66 |
| EfficientQAT [2] | 1 | 16.7 | 49.2 | 27.5 | 50.7 | 23.8 | 67.6 | 10.4 | 35.1 | 74.2 | 84.1 | 53.2 | 65.1 | 69.2 | 44.4 | 57.6 | 33.3 | 58.9 | 70.6 | 11.8 | 16.6 | 34.9 | 41.0 | 44.8 | 0.66 |
| PQF [33] | 1 | 24.5 | 63.5 | 50.2 | 74.5 | 41.2 | 75.7 | 13.9 | 49.1 | 75.6 | 91.7 | 69.8 | **74.0** | 77.8 | 65.5 | **61.6** | 37.9 | 64.7 | 76.4 | 41.3 | 21.6 | 32.3 | 50.2 | 55.6 | 0.10 |
| VQT (Ours) | 1 | **52.3** | **85.1** | **60.6** | **93.5** | **78.2** | **86.8** | **35.3** | **70.3** | **82.0** | **94.5** | **80.5** | 73.8 | **82.7** | **66.8** | 59.2 | **49.2** | **77.8** | **84.9** | **50.0** | **33.7** | **38.0** | **57.5** | **67.5** | 0.69 |

object detection (COD) [13, 25], shadow detection (SD) [39], leaf disease segmentation (LDS) [43], and polyp segmentation (PS) [19, 38]. We report mean Intersection over Union (IoU), mean Dice coefficient (Dice), weighted F-measure ($F_\beta^w$), S-measure ($S_\alpha$), mean E-measure ($E_\phi$), and balanced error rate (BER). More details of datasets are provided in the supplementary material.

**Implementation details.** For image classification, we use the AdamW optimizer [31] with cross-entropy loss, a batch size of 32, and fine-tune the models for 100 epochs. For semantic segmentation, we use the Adam optimizer [23] with structure loss, defined as a combination of weighted IoU loss and binary cross-entropy loss, unless otherwise specified. The batch size is set to 2, and the models are fine-tuned for 20 epochs. In all experiments, we adopt a cosine annealing learning rate schedule [32]. We primarily evaluate three high compression configurations: $(d = 4, K = 256)$, $(d = 4, K = 64)$, and $(d = 8, K = 256)$, which correspond approximately to 2-bit, 1.5-bit, and 1-bit settings, respectively, according to Eq. 4. In all settings, both the codebooks and the pre-trained weights are in 32-bit precision. We set the LoRA rank $r$ to 4 and the initial replacement probability $p_0$ to 0.8 by default. More details are in the supplementary material.

**Comparison methods.** We compare our method with three categories of baselines: full-precision (FP) methods, *i.e.*, uncompressed models, Q-PEFT methods, and VQ methods. 1) FP methods include Full fine-tuning and representative PEFT methods (Linear probing, Decoder-only, VPT [20], and LoRA [17]), where Linear probing or Decoder-only denotes fine-tuning only the classification head or segmentation decoder, respectively. For full fine-tuning, we use a smaller learning rate for the backbone than for the classification head or segmentation decoder, which makes the results more stable. FP methods serve as performance

**Table 2:** Comparison with baselines of post-training procedure on the VTAB-1K using ViT-B. "+baseline": applying the method to the fine-tuned model (first row). †: enhanced with task loss.

| Methods | Bit | Natural | Specialized | Structured | Average All | Param |
|---|---|---|---|---|---|---|
| LoRA | FP | 81.8 | 87.0 | 60.4 | 73.9 | 0.59 M |
| Post-training procedure | | | | | | |
| +VQ4DiT [6] | 2 | 79.4 | 86.0 | 59.5 | 72.4 | 42.52 M |
| | 1.5 | 73.7 | 84.0 | 56.6 | 68.7 | 42.48 M |
| | 1 | 57.0 | 80.6 | 54.5 | 60.9 | 21.33 M |
| +VQ4DiT† | 2 | 79.9 | 85.9 | 60.2 | 72.9 | 42.52 M |
| | 1.5 | 76.2 | 84.4 | 58.7 | 70.6 | 42.48 M |
| | 1 | 68.3 | 79.8 | 58.0 | 66.4 | 21.33 M |
| Unified training procedure | | | | | | |
| VQT (Ours) | 2 | 80.4 | 85.9 | 59.1 | 72.6 | 0.64 M |
| | 1.5 | 76.9 | 84.5 | 58.6 | 70.8 | 0.60 M |
| | 1 | 70.3 | 82.7 | 57.5 | 67.5 | 0.69 M |

**Table 3:** Performance comparisons on the VTAB-1K benchmark using ViT-L. We report the average accuracy over 19 tasks. The best results are highlighted in **bold**. The gray background represents ours.

| Methods | Bit | Natural | Specialized | Structured | Average All | Param |
|---|---|---|---|---|---|---|
| Full fine-tuning | FP | 78.7 | 85.2 | 55.2 | 70.2 | 303.30 M |
| Linear probing | FP | 70.9 | 69.1 | 25.8 | 51.5 | 0.05 M |
| VPT | FP | 82.5 | 83.9 | 54.1 | 70.8 | 0.49 M |
| LoRA | FP | 81.4 | 86.8 | 56.7 | 72.1 | 1.57 M |
| EfficientQAT | 2 | 75.6 | 84.7 | 54.9 | 68.8 | 2.36 M |
| SSVQ | 2 | 76.0 | 84.9 | **56.2** | 69.5 | 302.19 M |
| VQT (Ours) | 2 | **78.5** | **85.6** | 55.1 | **70.1** | 1.72 M |
| EfficientQAT | 1 | 47.4 | 75.3 | 46.8 | 53.0 | 2.36 M |
| SSVQ | 1.25 | 57.5 | 80.2 | 50.2 | 59.2 | 302.01 M |
| VQT (Ours) | 1 | **69.9** | **83.2** | **55.0** | **66.4** | 1.82 M |

**Table 4:** Efficiency comparison with advanced VQ on ViT-B under 2/1.5/1-bit settings. "Param", "Memory", and "Time" denote trainable parameters, training memory (batchsize = 32), and per-task fine-tuning time on VTAB-1K. $\text{Bit}_{\text{TP}}$ and $\text{CR}_{\text{TP}}$ denote the effective bit-width and compression ratio of the task-specific parameters required for storage for each downstream task, relative to the uncompressed 32-bit model.

| Methods | Param (M) | | | Memory (GB) | | | Time (min) | | | $\text{Bit}_{\text{TP}}$ | | | $\text{CR}_{\text{TP}}$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 1.5 | 1 | 2 | 1.5 | 1 | 2 | 1.5 | 1 | 2 | 1.5 | 1 | 2 | 1.5 | 1 |
| VQ4DiT [6] | 42.52 | 42.48 | 21.33 | 7.68 | 6.79 | 7.67 | 16.5 | 16.9 | 16.5 | 2.02 | 1.50 | 1.04 | 15.8× | 21.2× | 30.8× |
| SSVQ [26] | 85.03 | 84.94 | 84.95 | 7.31 | 7.31 | 7.21 | 22.9 | 23.4 | 22.8 | 1.05 | 1.03 | 1.01 | 30.4× | 31.06× | 31.68× |
| VQT (Ours) | 0.64 | 0.60 | 0.69 | 4.82 | 4.40 | 4.40 | 14.0 | 13.2 | 13.1 | 0.24 | 0.23 | 0.26 | 132.9× | 141.0× | 123.4× |

references without model compression. 2) Q-PEFT methods are based on SQ, including PEQA [22] and EfficientQAT [2]. All Q-PEFT methods use weight-only group quantization protocol [14] with a group size of 128, where every 128 input-channel weight elements share one scaling factor. 3) VQ methods include PQF [33], SSVQ [26], and VQ4DiT [6]. VQ4DiT is evaluated in a post-training procedure. Since SSVQ requires retaining at least 1 bit for sign information, we report its 1.25-bit results for a comprehensive comparison. Implementation details of comparison methods are in the supplementary material.

### 6.2 Comparisons on Image Classification

**Comparisons with baselines in unified training procedure.** We report comparison results on ViT-B over the VTAB-1K benchmark in Table 1. VQT achieves substantial performance gains while fine-tuning only a small fraction of parameters. For example, at the 1.5-bit compression, SSVQ fine-tunes a parameter scale comparable to full fine-tuning and attains 66.6% average accuracy, whereas VQT updates only 0.7% of parameters (0.6M/85.80M) and improves accuracy by 4.2%. Although PQF fine-tunes only the codebooks with a small number of parameters, it keeps the codeword assignments fixed and performs 5.1% worse than VQT (65.7% *vs.* 70.8%). These results highlight the importance of codeword reassignment for downstream fine-tuning. The advantage of

**Table 5:** Semantic segmentation results on SAM-H over four downstream scenarios. "Bit" denotes the target bit-width setting. "Param" denotes trainable parameters. The best results are highlighted in **bold**. The gray background represents ours.

| Methods | Bit | Param (M) | SD | LDS | PS | | COD | |
|---|---|---|---|---|---|---|---|---|
| | | | SBU BER↓ | Leaf Dice↑/IoU↑/$F_\beta^w$↑ | Kvasir Dice↑/IoU↑/$F_\beta^w$↑ | ETIS Dice↑/IoU↑/$F_\beta^w$↑ | CAMO $S_\alpha$↑/$E_\phi$↑/$F_\beta^w$↑ | COD10K $S_\alpha$↑/$E_\phi$↑/$F_\beta^w$↑ |
| Full fine-tuning | FP | 641.09 | 0.037 | 0.835/0.737/0.812 | 0.915/0.870/0.911 | 0.722/0.655/0.690 | 0.889/0.930/0.852 | 0.915/0.953/0.870 |
| Decoder-only | FP | 4.06 | 0.196 | 0.614/0.483/0.566 | 0.814/0.744/0.795 | 0.441/0.387/0.405 | 0.814/0.856/0.733 | 0.842/0.888/0.736 |
| LoRA [17] | FP | 6.71 | 0.034 | 0.839/0.741/0.818 | 0.912/0.861/0.901 | 0.802/0.732/0.760 | 0.893/0.937/0.866 | 0.922/0.961/0.883 |
| EfficientQAT [2] | 2 | 9.88 | 0.045 | 0.828/0.727/0.798 | 0.919/0.869/0.910 | 0.719/0.651/0.681 | 0.861/0.904/0.816 | 0.883/0.928/0.814 |
| PQF [33] | 2 | 5.10 | 0.043 | 0.825/0.723/0.799 | 0.909/0.859/0.899 | 0.713/0.645/0.671 | 0.866/0.909/0.826 | 0.896/0.941/0.839 |
| SSVQ [26] | 2 | 634.45 | 0.042 | 0.836/0.735/**0.817** | 0.907/0.857/0.900 | 0.777/0.706/0.733 | 0.868/0.914/0.827 | 0.903/0.946/0.851 |
| VQT (Ours) | 2 | 7.80 | **0.039** | **0.838/0.739/0.817** | **0.923/0.876/0.916** | **0.786/0.718/0.752** | **0.875/0.916/0.837** | **0.908/0.950/0.858** |
| EfficientQAT [2] | 1 | 9.88 | 0.080 | 0.713/0.585/0.672 | 0.702/0.598/0.662 | 0.311/0.245/0.269 | 0.625/0.698/0.440 | 0.649/0.721/0.399 |
| PQF [33] | 1 | 5.23 | 0.055 | 0.775/0.662/0.737 | 0.878/0.822/0.870 | 0.642/0.572/0.600 | 0.784/0.837/0.698 | 0.829/0.887/0.723 |
| SSVQ [26] | 1.25 | 634.14 | 0.052 | 0.797/0.688/0.770 | 0.882/0.826/0.871 | 0.625/0.561/0.592 | 0.803/0.845/0.730 | 0.848/0.902/0.752 |
| VQT (Ours) | 1 | 7.85 | **0.047** | **0.818/0.710/0.789** | **0.903/0.848/0.897** | **0.710/0.637/0.663** | **0.818/0.863/0.742** | **0.853/0.903/0.755** |

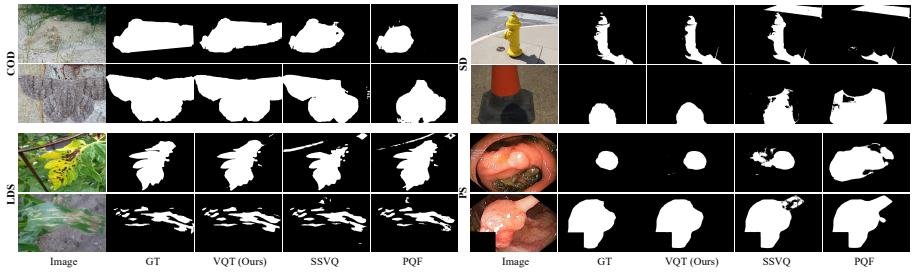



**Fig. 5:** Visual comparison of SAM-H under the 1-bit vector quantization setting across four semantic segmentation scenarios, where SSVQ is at 1.25 bit-widths.

VQT becomes increasingly significant as the compression ratio increases (lower bit-widths), particularly at the 1-bit setting. Notably, compared to the FP methods, at the 2-bit compression, VQT incurs only a 1.3% average accuracy drop relative to LoRA, while achieving a 16× compression (from 32 bits to 2 bits), making it more suitable for deployment. Moreover, under low bit-widths, VQT also consistently outperforms SQ-based Q-PEFT methods like PEQA and EfficientQAT. Overall, our VQT achieves a favorable balance between fine-tuning efficiency, high compression ratios, and downstream performance.

**Comparisons with baselines of post-training procedure.** Table 2 highlights the advantages of our method over post-training VQ baselines. "+VQ4DiT" is the vanilla design with block-wise calibration. "+VQ4DiT†" is our modified version augmented with a task loss. Both "+VQ4DiT" and "+VQ4DiT†" perform VQ on already fine-tuned models, requiring an additional post-training compression procedure after fine-tuning. In contrast, our VQT performs VQ-aware fine-tuning on pre-trained models in a unified training procedure for efficient fine-tuning and compression, eliminating the need for any extra post-training procedure and achieving superior performance at low bit-widths, such as 1-bit, compared with the "+VQ4DiT" baseline. While VQT achieves comparable performance to "+VQ4DiT†" at 2 bits, it is more efficient with no extra post-training procedure and only a small fraction of fine-tuned parameters.

**Table 6:** Ablation on each component under three different bit-widths. We report the average accuracy on VTAB-1K using ViT-B. The deep gray background means the baseline directly combining LoRA with VQ. The gray background means final design.

| VQ | LoRA & STE | Stochastic Sub-vector Replacement | Progressive Alignment | Continuous Anchored | VQ Anchored | 2-bit | 1.5-bit | 1-bit |
|---|---|---|---|---|---|---|---|---|
| ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | 71.2 | 64.4 | 55.1 |
| ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | 72.5 | 69.0 | 57.6 |
| ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | 72.3 | 69.4 | 63.9 |
| ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | 72.6 | 70.8 | 66.2 |
| ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | 72.5 | 70.7 | 66.6 |
| ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | 72.6 | 70.8 | 67.5 |

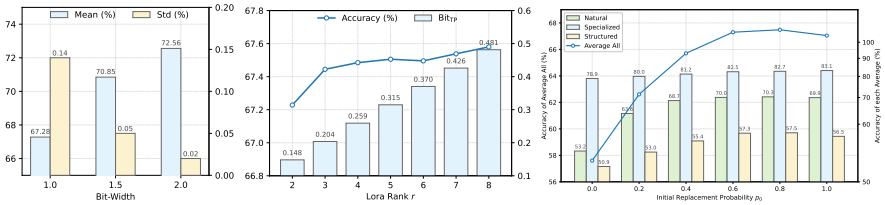
**Comparisons on larger ViT.** We evaluate VQT on larger-scale ViT-L. As shown in Table 3, 1-bit VQT outperforms 1.25-bit SSVQ by 7.2% while fine-tuning only 0.6% of the parameters (1.82M/303.30M), demonstrating its effectiveness at extreme compression. More results are in the supplementary material.
**Efficiency comparison with advanced VQ methods.** The efficiency comparisons in Table 4 are conducted on ViT-B under 2/1.5/1-bit settings. Here, SSVQ is quantized to 1.25 bits in practice under the nominal 1-bit setting. As shown in Table 4, VQT exhibits clear advantages over VQ4DiT and SSVQ during fine-tuning. It fine-tunes only a small fraction of parameters, which reduces both memory usage and fine-tuning time. For multi-task deployment, VQT stores only task-specific LoRA parameters and codebooks, resulting in an equivalent storage cost of less than 1 bit per parameter. In contrast, VQ4DiT stores the full vector-quantized model, including assignments and codebooks, while SSVQ requires at least 1-bit sign information. Consequently, their equivalent storage cost is no less than 1 bit per parameter. These results highlight VQT's efficiency in both fine-tuning and multi-task storage. The additional inference-time measurement of VQ is provided in the supplementary material.

### 6.3 Comparisons on Semantic Segmentation

We report the comparison results of SAM-H across four downstream scenarios in Table 5. Both SSVQ and our proposed VQT outperform PQF across different scenarios, highlighting the importance of codeword index reassignment for effective downstream adaptation. Moreover, our VQT consistently demonstrates superior performance over SSVQ. For instance, under the 1-bit setting, VQT outperforms SSVQ (1.25-bit) on the CAMO dataset by 0.015 $S_\alpha$, 0.018 $E_\phi$, and 0.012 $F_\beta^w$, while fine-tuning only 1.2% of the parameters (7.85/641.09), whereas SSVQ requires fine-tuning nearly all parameters. We also provide the visual results of semantic segmentation in Fig. 5, which further verify the effectiveness of our VQT in the qualitative performance. More quantitative and qualitative comparison results are provided in the supplementary material.

### 6.4 Ablation Studies and Analysis

In this section, we conduct ablation studies on ViT-B over VTAB-1K to evaluate each component, analyze the effects of the LoRA rank $r$ and initial replacement




**(a)** Robustness analysis **(b)** LoRA Rank $r$ **(c)** Initial replacement probability $p_0$

**Fig. 6:** Ablation studies on random seeds (a) and hyperparameters (b, c).

probability $p_0$, and assess robustness across different random seeds with average accuracy as the evaluation metric. We also add some further analysis.

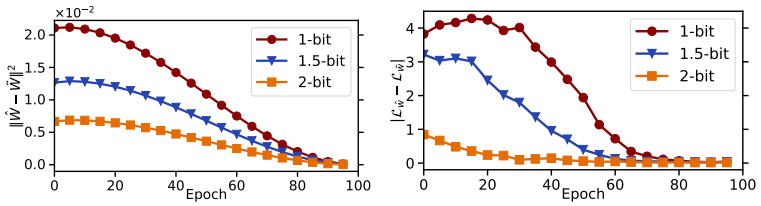
**Effectiveness of each component.** In Table 6, we set $r = 4$ and $p_0 = 0.5$ as default and report compression performance under three bit-widths. "VQ" denotes fine-tuning codebooks only without codeword reassignment, while "LoRA & STE" applies STE-based gradient estimation for LoRA fine-tuning followed by codeword reassignment based on the LoRA-updated weights. As shown in Table 6, introducing LoRA (*i.e.*, "LoRA & STE") consistently improves performance across all three bit-widths, highlighting the importance of codeword reassignments for effective downstream adaptation. Applying our stochastic sub-vector replacement strategy enhances performance, suggesting that directly combining LoRA with STE-based VQ fine-tuning is suboptimal. When combined with progressive alignment, additional gains are observed, confirming that the optimization under stochastic sub-vector replacement is effectively aligned with the intended objective. We also verify the effectiveness of the continuous-anchored and VQ-anchored modes defined in Eq. 7. Notably, VQT yields larger improvements over the baseline in second row of Table 6 at lower bit-widths. This result is expected according to Eq. 5 and Eq. 6: lower bit-widths enlarge the gap between $W$ and $\hat{W}$, resulting in larger CVQN. By mitigating CVQN, VQT achieves greater gains under these conditions, further validating its effectiveness. Full results on all 19 VTAB datasets are provided in the supplementary material.

**Robustness analysis.** Since VQT employs a stochastic strategy, we evaluate its robustness by measuring performance under three different random seeds for each bit-width setting. As shown in Fig. 6a, we report the mean and standard deviation (Std) across seeds for each bit-width. The mean closely match the results in Table 1, and the standard deviation remains below 0.15, confirming that our VQT is robust for different random seeds. Full results on all 19 downstream datasets for each random seed are provided in the supplementary material.

**Hyperparameter $r$ and $p_0$.** We perform ablation studies to evaluate the impact of $r$ and $p_0$ on performance at 1 bit. In Fig. 6b, we fix $p_0 = 0.8$ and vary $r$, using the rsLoRA strategy [21] to reduce the influence of $\alpha$ in Eq. 1. Results show that even moderate increases in $r$ keep the effective bit-width for task-specific parameters well below 1 bit, and performance saturates when $r > 4$. We thus adopt $r = 4$ in our experiments. In Fig. 6c, we fix $r = 4$ and vary $p_0$. Performance drops when $p_0$ is small, indicating that CVQN is insufficiently mitigated,

**Table 7:** Analysis on extending VQT to DoRA [30] on ViT-B. "VQT+DoRA" represents directly combining DoRA with VQ and applies STE-based gradient estimation.

| Method | Bit | Natural | Specialized | Structured | Average All |
|---|---|---|---|---|---|
| VQ+DoRA | 1 | 52.4 | 78.3 | 51.4 | 57.4 |
| VQT *w.* DoRA | 1 | **70.8** | **83.4** | **57.6** | **67.9** |




**(a)** Empirical trajectory of $\|\hat{W} - \tilde{W}\|^2$ **(b)** Empirical trajectory of $|\mathcal{L}_{\hat{w}} - \mathcal{L}_{\tilde{w}}|$

**Fig. 7:** Empirical trajectory of $\|\hat{W} - \tilde{W}\|^2$ (summed over all 12 blocks) and $|\mathcal{L}_{\hat{w}} - \mathcal{L}_{\tilde{w}}|$ on ViT-B during fine-tuning.

leading to suboptimal LoRA updates. We set $p_0 = 0.8$ in our experiments. Full results on all 19 downstream datasets and more results at 2 bits and 1.5 bits are provided in the supplementary material.

**Analysis on other PEFT methods.** We evaluate whether VQT generalizes to weight-merging PEFT methods beyond LoRA. In Table 7, we replace LoRA with DoRA [30]. VQT outperforms the direct combination, VQ+DoRA, suggesting that CVQN negatively affects low-rank weight-merging PEFT methods.

**Analysis on $\tilde{W}$ and $\mathcal{L}_{\tilde{w}}$.** We further analyze the trajectory of $\|\hat{W} - \tilde{W}\|^2$ and $|\mathcal{L}_{\hat{w}} - \mathcal{L}_{\tilde{w}}|$ during fine-tuning. As shown in Fig. 7, using progressive alignment, $\tilde{W}$ and $\mathcal{L}_{\tilde{w}}$ are eventually aligned with $\hat{W}$ and $\mathcal{L}_{\hat{w}}$, respectively.

## 7 Conclusion

Our proposed VQT provides a novel framework for efficient downstream fine-tuning and compression of pre-trained ViTs based on vector quantization. VQT mitigates coupled vector quantization noise using stochastic sub-vector replacement to stabilize LoRA updates and prevent incorrect codeword assignments, and uses progressive alignment to prevent optimization mismatch between fine-tuning and deployment. Extensive experiments show that VQT achieves strong downstream performance under high compression, while maintaining high fine-tuning efficiency and enabling efficient storage on edge devices for multiple tasks.

## Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant 62131003, and the Fundamental Research Funds for the Central Universities under Grant WK2100000059.

## References


1. Bengio, Y., Léonard, N., Courville, A.: Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv preprint arXiv:1308.3432 (2013)
2. Chen, M., Shao, W., Xu, P., Wang, J., Gao, P., Zhang, K., Luo, P.: Efficientqat: Efficient quantization-aware training for large language models. In: ACL (2025)
3. Chen, S., Ge, C., Tong, Z., Wang, J., Song, Y., Wang, J., Luo, P.: Adaptformer: Adapting vision transformers for scalable visual recognition. In: NeurIPS (2022)
4. Chen, T., Zhu, L., Deng, C., Cao, R., Wang, Y., Zhang, S., Li, Z., Sun, L., Zang, Y., Mao, P.: Sam-adapter: Adapting segment anything in underperformed scenes. In: ICCV Workshop (2023)
5. Cho, M., Alizadeh-Vahid, K., Adya, S., Rastegari, M.: DKM: differentiable k-means clustering layer for neural network compression. In: ICLR (2022)
6. Deng, J., Li, S., Wang, Z., Gu, H., Xu, K., Huang, K.: Vq4dit: Efficient post-training vector quantization for diffusion transformers. In: AAAI (2025)
7. Deng, J., Li, S., Wang, Z., Xu, K., Gu, H., Huang, K.: Vim-vq: Efficient post-training vector quantization for visual mamba. In: ICCV (2025)
8. Dettmers, T., Pagnoni, A., Holtzman, A., Zettlemoyer, L.: Qlora: Efficient finetuning of quantized llms. In: NeurIPS (2023)
9. Ding, X., Liu, X., Tu, Z., Zhang, Y., Li, W., Hu, J., Chen, H., Tang, Y., Xiong, Z., Yin, B., et al.: Cbq: Cross-block quantization for large language models. In: ICLR (2025)
10. Dong, X., Bao, J., Zhang, T., Chen, D., Zhang, W., Yuan, L., Chen, D., Wen, F., Yu, N., Guo, B.: Peco: Perceptual codebook for BERT pre-training of vision transformers. In: AAAI (2023)
11. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: ICLR (2021)
12. Esser, S.K., McKinstry, J.L., Bablani, D., Appuswamy, R., Modha, D.S.: Learned step size quantization. In: ICLR (2020)
13. Fan, D., Ji, G., Sun, G., Cheng, M., Shen, J., Shao, L.: Camouflaged object detection. In: CVPR (2020)
14. Frantar, E., Ashkboos, S., Hoefler, T., Alistarh, D.: Gptq: Accurate post-training quantization for generative pre-trained transformers. arXiv preprint arXiv:2210.17323 (2022)
15. Hartigan, J.A., Wong, M.A.: Algorithm as 136: A k-means clustering algorithm. Journal of the royal statistical society. series c (applied statistics) (1979)
16. Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., Gelly, S.: Parameter-efficient transfer learning for NLP. In: ICML (2019)
17. Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al.: LoRA: Low-rank adaptation of large language models. In: ICLR (2022)
18. Huang, M., Mao, Z., Chen, Z., Zhang, Y.: Towards accurate image coding: Improved autoregressive image generation with dynamic vector quantization. In: CVPR (2023)
19. Jha, D., Smedsrud, P.H., Riegler, M.A., Halvorsen, P., de Lange, T., Johansen, D., Johansen, H.D.: Kvasir-seg: A segmented polyp dataset. In: MMM (2020)

20. Jia, M., Tang, L., Chen, B., Cardie, C., Belongie, S.J., Hariharan, B., Lim, S.: Visual prompt tuning. In: ECCV (2022)
21. Kalajdzievski, D.: A rank stabilization scaling factor for fine-tuning with lora. arXiv preprint arXiv:2312.03732 (2023)
22. Kim, J., Lee, J.H., Kim, S., Park, J., Yoo, K.M., Kwon, S.J., Lee, D.: Memory-efficient fine-tuning of compressed large language models via sub-4-bit integer quantization. In: NeurIPS (2023)
23. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR (2015)
24. Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W.Y., Dollar, P., Girshick, R.: Segment anything. In: ICCV (2023)
25. Le, T., Nguyen, T.V., Nie, Z., Tran, M., Sugimoto, A.: Anabranch network for camouflaged object segmentation. CVIU (2019)
26. Li, S., Deng, J., Wang, C., Xu, K., Deng, R., Gu, H., Shen, H., Huang, K.: Ssvq: Unleashing the potential of vector quantization with sign-splitting. In: ICCV (2025)
27. Lian, D., Zhou, D., Feng, J., Wang, X.: Scaling & shifting your features: A new baseline for efficient model tuning. In: NeurIPS (2022)
28. Lin, J., Tang, J., Tang, H., Yang, S., Chen, W., Wang, W., Xiao, G., Dang, X., Gan, C., Han, S.: AWQ: activation-aware weight quantization for on-device LLM compression and acceleration. In: MLSys (2024)
29. Liu, H., Tam, D., Muqeeth, M., Mohta, J., Huang, T., Bansal, M., Raffel, C.: Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In: NeurIPS (2022)
30. Liu, S., Wang, C., Yin, H., Molchanov, P., Wang, Y.F., Cheng, K., Chen, M.: Dora: Weight-decomposed low-rank adaptation. In: ICML (2024)
31. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)
32. Loshchilov, I., Hutter, F.: SGDR: stochastic gradient descent with warm restarts. In: ICLR (2017)
33. Martinez, J., Shewakramani, J., Liu, T.W., Barsan, I.A., Zeng, W., Urtasun, R.: Permute, quantize, and fine-tune: Efficient compression of neural networks. In: CVPR (2021)
34. Nesterov, Y.E.: Introductory Lectures on Convex Optimization - A Basic Course, Applied Optimization, vol. 87. Springer (2004)
35. Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., et al.: Dinov2: Learning robust visual features without supervision. arXiv preprint arXiv:2304.07193 (2023)
36. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., Sutskever, I.: Learning transferable visual models from natural language supervision. In: ICML (2021)
37. Ryu, H., Lim, S., Shim, H.: Memory-efficient fine-tuning for quantized diffusion model. In: ECCV (2024)
38. Silva, J., Histace, A., Romain, O., Dray, X., Granado, B.: Toward embedded detection of polyps in WCE images for early diagnosis of colorectal cancer. Int. J. Comput. Assist. Radiol. Surg. (2014)
39. Vicente, T.F.Y., Hou, L., Yu, C., Hoai, M., Samaras, D.: Large-scale training of shadow detectors with noisily-annotated shadow examples. In: ECCV (2016)
40. Xu, Y., Xie, L., Gu, X., Chen, X., Chang, H., Zhang, H., Chen, Z., Zhang, X., Tian, Q.: Qa-lora: Quantization-aware low-rank adaptation of large language models. In: ICLR (2024)

41. Zhai, X., Puigcerver, J., Kolesnikov, A., Ruyssen, P., Riquelme, C., Lucic, M., Djolonga, J., Pinto, A.S., Neumann, M., Dosovitskiy, A., et al.: A large-scale study of representation learning with the visual task adaptation benchmark. arXiv preprint arXiv:1910.04867 (2019)
42. Zhang, Q., Chen, M., Bukharin, A., He, P., Cheng, Y., Chen, W., Zhao, T.: Adaptive budget allocation for parameter-efficient fine-tuning. In: ICLR (2023)
43. Zhong, Z., Tang, Z., He, T., Fang, H., Yuan, C.: Convolution meets lora: Parameter efficient finetuning for segment anything model. In: ICLR (2024)
44. Zhou, K., Yang, J., Loy, C.C., Liu, Z.: Conditional prompt learning for vision-language models. In: CVPR (2022)