Learning Low-Rank Representations for Model Compression

Zezhou Zhu, Yuan Dong
School of Artificial Intelligence
Beijing University of Posts and Telecommunications
Beijing, China
{zhuzezhou, yuandong}@bupt.edu.cn

Zhong Zhao
Central Media Technology Institute
Huawei
Shenzhen, China
zorro.zhongzhao@huawei.com

Abstract—Vector Quantization (VQ) is an appealing model compression method to obtain a tiny model with less accuracy loss. While methods to obtain better codebooks and codes under fixed clustering dimensionality have been extensively studied, optimizations via the reduction of subvector dimensionality are not carefully considered. This paper reports our recent progress on model compression with the combination of dimensionality reduction and vector quantization, proposing Low-Rank Representation Vector Quantization (LR\(^2\)VQ). LR\(^2\)VQ joins low-rank representation with subvector clustering to construct a new kind of building block that is optimized by end-to-end training. In our method, the compression ratio could be directly controlled by the dimensionality of subvectors, and the final accuracy is solely determined by clustering dimensionality \(d\). We recognize \(d\) as a trade-off between low-rank approximation error and clustering error and carry out both theoretical analysis and experimental observations that empower the estimation of the proper \(d\) before fine-tuning. With a proper \(d\), we evaluate LR\(^2\)VQ with ResNet-18/ResNet-50 on ImageNet classification datasets, achieving 2.8%/1.0% top-1 accuracy improvements over \(\times\) compression factor.

Index Terms—Deep learning, low-rank representation, vector quantization, model compression

I. INTRODUCTION

In recent years, deep neural networks have achieved remarkable performance on different vision tasks like image classification, object detection, and semantic segmentation. However, this progress is fueled by wider and deeper network architectures, which possess a large memory footprint and computation overhead. These networks are hard to be deployed on battery-powered and resource-constrained hardware such as wearable devices and mobile phones. At the same time, deep neural networks are redundant in parameters and layer connections, which implies that it is possible to compress them without much accuracy loss. Therefore, compressing neural networks is essential for real-world applications.

There are several approaches to compressing deep neural networks from different perspectives. Compact network design\([1; 2; 3; 4]\) and neural architecture search (NAS)\([5; 6; 7; 8; 9]\) focus on exploring lightweight architectures; Pruning\([10]\) tends to eliminate unnecessary connections or channels on heavy models directly; Quantization\([11; 12]\) compresses each parameter from FP32 to lower bits. Here we concentrate on Vector Quantization(VQ)\([13]\) to achieve a higher compression-to-accuracy ratio.

VQ exploits filter redundancy to reduce storage cost with codebook and codes. Most of the prior works\([14; 15; 16; 17; 18; 19]\) demonstrate impressive performance on compression-to-accuracy ratio. The key to the success of VQ is clustering and fine-tuning, where clustering decreases the memory demand for storage and computation, and fine-tuning helps recover the model performance by data-driven optimization. The accuracy loss of the quantized network is closely related to the clustering error. Great progress have been achieved on finding more expressive codebooks and codes by reducing the clustering error under a fixed clustering dimensionality. However, the clustering dimensionality is a critical factor in impacting clustering error. Applying a distance-based clustering method in high-dimensional space is more likely to suffer from the curse of dimensionality\([20]\), which introduces significant difficulties for clustering to produce large clustering errors. On the contrary, clustering in low-dimensional space is much easier to produce lower clustering error. Specifically, the dimensionality of subvectors in convolutional layers is always 9 or 18, raising the difficulty in finding expressive codebooks and codes in high dimensional space, and resulting in poor quantization performance. On the question of dimensionality reduction, low-rank representation methods are widely used for all sorts of dimensionality reduction tasks. These remarkable methods invoke the demand to investigate the possibility to design significantly better quantizers by the mixture of VQ and low-rank representations.

To keep up with the above demand, this paper proposes Low-Rank Representation Vector Quantization (LR\(^2\)VQ), a new method that jointly considers the compression ratio and the clustering dimensionality to achieve better quantization. LR\(^2\)VQ utilizes low-rank representations (LRRs) to approximate the original filters and clustering the subvectors in LRRs to achieve quantization. Our method possesses an extraordinary characteristic: the compressed model size is controlled by \(m\) (the subvector size), and the quantization performance is solely determined by \(d\) (the clustering dimensionality). This characteristic decouples \(m\) and \(d\), which are always equal in previous VQ approaches. At this time, \(d\) can vary in a wide range freely. The changeable \(d\) contributes to clustering...
in low-dimensional space, which benefits quantization with lower clustering error. Besides, it provides an opportunity to comprehensively research the dimensionality reduction and vector quantization without impacting the compressed model size. Moreover, it introduces a trade-off between the approximation error in low-rank representation learning and the clustering error in quantization. Based on theoretical analysis and experimental results, we empower the estimation of \( d \) which appropriately balances errors to achieve better quantization performance. We experiment LR\(^2\)VQ on large-scale datasets like ImageNet and COCO. Compared with the results in PQF\([19]\), LR\(^2\)VQ improves 2.8%/1.0% top-1 accuracy for ResNet-18/ResNet-50 on ImageNet with 43×31× compression factor, and 0.91% box AP for Mask R-CNN on COCO with 26× compression factor. In summary, our main contributions are as follows:

- We propose a simple yet effective quantization method called Low-Rank Representation Vector Quantization (LR\(^2\)VQ) for deep model compression. The subvector size \( m \) and the clustering dimensionality \( d \) are mutually independent, making it possible to achieve dimensionality reduction for better compression performance.
- We identify the trade-off of \( d \), and provide estimations to \( d \) with both theoretical and empirical results to balance the approximation error and the clustering error to lower the overall reconstruction error in LR\(^2\)VQ.
- We evaluate LR\(^2\)VQ with various \( d \), which produces state-of-the-art performance on different datasets and architectures.

II. RELATED WORK

There is a large body of literature on deep neural network compression. We review related literature from four perspectives: compact network design, pruning, tensor decomposition, and quantization.

A. Compact network design

Light weight networks like SqueezeNet\([21]\), NasNet\([5]\), ShuffleNet\([1]\), MobileNets\([2; 3]\) and EfficientNets\([4]\) are proposed to be computation and memory efficient. However, these architectures are either hand-crafted or produced by searching algorithms. It is inefficient to design networks manually, and current searching approaches demand vast computation resources. To avoid these issues, another line of work directly operates on the existed network architectures (like VGG or ResNets) to achieve compression and acceleration.

B. Pruning

The simplest pruning operates on filters by removing the connections according to an importance criteria until achieving desirable compression-to-accuracy trade-off\([10; 22; 23]\). However, most pruning methods discard individual parameters, and the pruned networks are too sparse to achieve acceleration on embedded devices. Hence, some approaches focus on pruning unimportant channels to realize compression and acceleration simultaneously\([24; 25; 26]\).

C. Quantization

Quantization amounts to reducing the bitwidth of each parameter in neural networks. In this context, we focus on Vector Quantization (VQ), which treats each vector individually to decompose a high-dimensional tensor with small-sized codebooks and low-bit codes. BGD\([17]\), P&G\([18]\), PQF\([19]\) and DKM\([27]\) are recently proposed vector quantization approaches. BGD minimizes the reconstruction error of feature maps in the quantized network and optimizes codebooks via layer-wise distillation. P&G directly minimizes the reconstruction error of parameters to achieve improvements. PQF addresses the invariance problem for quantization and formulates a permutation-based method to find a functional equivalent network for easier clustering. Nevertheless, the permutation is also performed on high-dimensional space, which is more likely to result in a large clustering error. DKM is an attention-based method that learns to cluster during network training. However, the differentiable clustering method requires expensive end-to-end training.

Unlike these approaches, our method takes advantage of learning LRRs to achieve changeable clustering dimensionality, which benefits the overall reconstruction error for quantization. Based on this property, we exploit the trade-off in LR\(^2\)VQ to guide the searching for the proper clustering dimensionality. The accuracy recovery process in LR\(^2\)VQ is also high efficiency, which needs a few fine-tuning epochs over the task loss. All these contributions result in an efficient and effective method for deep neural network compression.

III. Method

Our proposed method aims to achieve a target compression ratio with codebooks and codes from different clustering dimensionality. Toward this goal, LR\(^2\)VQ consists of three steps:

1) **Learning low-rank representations:** This step learns low-rank representations (LRRs) for all the convolutional filters using gradient-based learning methods. We use LRRs and linear transformations (LTs) to replace the convolutional filters for computation.

2) **Quantizing low-rank representations:** The dimensionality of LRRs is usually different from the original subvectors. After learning LRRs, we generate codebooks and codes by clustering the subvectors in LRRs. This operation realizes the variation of clustering dimensionality.

3) **Global fine-tuning:** We fine-tune codebooks by minimizing the loss function over training datasets with gradient-based optimization. After fine-tuning, we merge codebooks and linear transformations (LTs) to eliminate the additional computation complexity during inference.

A. Learning Low-Rank Representations

This section presents how to generate convolutional filters by LRRs and initialize LRRs for robust learning.

**Definition:** Let us denote a convolutional filter \( W \in \mathbb{R}^{C_{out} \times C_{in} \times K \times K} \) in uncompressed neural networks, with \( C_{out} \) as the number of output channels, \( C_{in} \) as the number of...
input channels, and $K$ as the spatial size of the filter. In normal VQ, $W$ should be reshaped into a 2D matrix $W_r \in \mathbb{R}^{C_{out}C_{in}KK \times m}$, where $m$ is the size of subvectors, and $C_{out}C_{in}KK/m$ is the number of subvectors. The value of $m$ determines the number of subvectors and compression ratios. In our method, we construct $W$ by approximating $W_r$ with two 2D matrices $A$ and $B$:

$$W_r \approx W' = A \times B,$$  

where $A \in \mathbb{R}^{C_{out}C_{in}KK / m}$ and $B \in \mathbb{R}^{d \times m}$. The dimensionality of $A$ is $d \in [1, m]$. Once $W'$ is computed, we can reshape it into a 4D tensor of shape $C_{out} \times C_{in} \times K \times K$ to replace $W$ for computation.

Equation (1) is similar to low-rank matrix decomposition, so $A$ can be treated as the LRR of $W'$ and $B$ is the corresponding linear transformation from $d$ to $m$. Instead of mathematical decomposing $W$, we directly learn $A$ and $B$ by end-to-end training from scratch, which is more adaptive for obtaining expressive LRR for $W'$.

**Initialization:** Weights initialization strongly influences the neural network’s optimization and final performance. As we replace the original weights by the learned LRRs to obtain codebooks $C$ and codes $I$, then, we discuss the trade-off of $d$ in LR$^2$VQ. Finally, we introduce an analytic method to search for a coarse estimation of $d$ that may result in a lower reconstruction error.

**Codebook and codes:** In LR$^2$VQ, LRRs take a majority of network parameters. So we apply VQ on LRRs to save storage with codebooks $C$ and codes $I$. In our definition, $A$ is matrix with $N/m$ rows and $d$ columns, where $N = C_{out}C_{in}KK/m$. We treat each row in $A$ as an individual subvector to be compressed, so the total number of subvectors for quantization is $N/m$, and the dimensionality of subvectors is $d$. To save the storage of these subvectors, we use $k$ centroids with size $\tilde{d}$ for approximation. We call the set of centroids as codebook $C = \{c_1, ..., c_k\} \subseteq \mathbb{R}^{k \times \tilde{d}}$ where each row in $C$ is a centroid. Codes $I \subseteq \mathbb{R}^d$ is a set of assignments that identify the best approximation mapping between subvectors and centroids

$$I_p = \arg \min_q ||A_p - C_q||^2_2,$$  

where $p$ and $q$ are the row indexes in $A$ and $C$, and $A_p \subseteq \mathbb{R}^{1 \times \tilde{d}}$ is the $p$th subvector, $C_q \subseteq \mathbb{R}^{1 \times \til{d}}$ is the $q$th centroid, $I_p \subseteq \mathbb{R}^1$ is a single code to $A_p$. With $C$ and $I$, we can decode them to construct $\tilde{A}$ by looking up $C$ with all the codes $I$

$$\tilde{A} = C(I) = \{C_{I_1}, ..., C_{I_m}\} \subseteq \mathbb{R}^{k \times \til{d}},$$

then transform $\tilde{A}$ to $m$-dimensional space

$$W' \approx \tilde{W} = \tilde{A} \times B.$$
All the codebooks and codes can be generated by clustering. The subvector size in normal VQ is $m$, while the subvector size in our method is $\tilde{d}$. Note that the clustering dimensionality equals the size of subvectors for all VQ methods. Therefore, $\tilde{d}$ is the clustering dimensionality in LR$^2$VQ, which can vary within $[1, m]$. The number of subvectors in LRR is also directly controlled by $m$, resulting in equivalent number of subvectors to the original filters. As a result, clustering on LRRs does not impact the compression ratio. Figure 1 depicts the comparison between normal VQ and LR$^2$VQ.

LR$^2$VQ is compatible with any clustering methods. Here, we introduce how LR$^2$VQ works with $k$-means clustering. After initializing a codebook with the subvectors in $\mathbf{A}$, $k$-means loops within the following steps:

1) Update assignments via Equation (3) according to the subvectors in LRRs and codebook;
2) Update codebook (centroids) according to assignments.

Once the loop ends, codes are fixed as each subvector should be replaced by a specific centroid. Although the Euclidean errors between subvectors and centroids might be tiny, such errors introduce a large gap between $\tilde{A}$ and $\tilde{\mathbf{A}}$. Therefore, extra network fine-tuning is necessary to compensate for performance loss.

**Trade-off of $\tilde{d}$:** In LR$^2$VQ, $\tilde{d}$ is the dimensionality for low-rank approximation and vector quantization. This value simultaneously influences the approximation error in low-rank representation learning and the clustering error in vector quantization, which affects the overall reconstruction error for model compression. Here, we discuss the trade-off of $\tilde{d}$ in LR$^2$VQ.

![Fig. 2: Illustration of the concreational relationships for $E_r$, $E_a$ and $E_c$.](image)

The following equation may describe the concreational relationship between the overall reconstruction error $E_r$, the approximation error $E_a$ and the clustering error $E_c$:

$$E_r \propto E_a + E_c.$$  \hspace{1cm} (6)

With a fixed $m$ and $k$, both $E_a$ and $E_c$ are only affected by the clustering dimensionality $\tilde{d}$. Therefore, we need a proper $\tilde{d}$ to reduce these two errors. For $E_a$, a larger $\tilde{d}$ means more parameters in the LRR network, indicating a smaller $E_a$ in LR$^2$VQ. Especially when $\tilde{d}$ is close to $m$, the number of parameters in the LRR network is already sufficient to approximate the original network, which can produce a “zero” approximation error. So $E_a$ monotonically decreases with a rising $\tilde{d}$. For $E_c$, high-dimensional subvectors are hard to be clustered, resulting in a large clustering error, while the low-dimensional subvectors are more favourable for clustering. For example, when $\tilde{d} = 1$, $E_c$ may become extremely tiny because each subvector is a single scaler in 1D space, which is the lowest and the simplest space for clustering; When $\tilde{d} = m$, the subvectors in LRR have the same dimensionality as the original subvectors, leading to significant clustering difficulty and the largest clustering error. So $E_c$ monotonically increases with a rising $\tilde{d}$. Based on the above discussion, we can conclude that $E_a$ and $E_c$ are inversely correlated. Adding these two errors together, we expect that the variation of $E_r$ is similar to Figure 2, which first declines then increases with a rising $\tilde{d}$. Such variation provides reliable insurance that there must be a proper $\tilde{d}$ to produce a smaller reconstruction error. In summary, the trade-off of $\tilde{d}$ is required to promise the lower reconstruction error and better compression performance in our LR$^2$VQ. We extensively experiment on various $\tilde{d}$ in later sections.

**Searching for proper $\tilde{d}$:** The most effective way to choose proper $\tilde{d}$ is grid search, which is time and resources consuming. Theoretically, an accurate analytical method is useful to search a proper $\tilde{d}$ after low-rank representation learning. Based on the assumption and analysis in PQF[19], we apply a simple method to coarsely estimate $\tilde{d}$ after LRR learning as a starting point.

As different architectures have different characteristics, it is hard to define $E_a$ precisely in mathematics. Besides, $E_a$ is fixed after LRR learning and can be measured by the model performance. Consequently, we choose the LRR networks perform similarly to the original network for searching $\tilde{d}$ because $E_a$ in these networks is negligible. At this time, $E_r$ is dominated by $E_c$, and we only need to consider $E_c$ in searching proper $\tilde{d}$. We note that $E_c$ indicates the clustering error between $\tilde{W}$ and $\tilde{\mathbf{W}}$ rather than $\mathbf{A}$ and $\tilde{\mathbf{A}}$. The essence of LR$^2$VQ is clustering the subvectors in $\tilde{\mathbf{W}}$, and we achieve this goal with an indirect method, which is clustering on $\mathbf{A}$. Therefore, the clustering error between $\tilde{\mathbf{W}}$ and $\tilde{\mathbf{W}}$ is the actual $E_c$ in LR$^2$VQ. Based on the above analysis, the following computations are all performed on $\tilde{\mathbf{W}}$ rather than $\mathbf{A}$.

According to PQF, we can infer that $\tilde{\mathbf{W}} \sim \mathcal{N}(0, \Sigma)$, where $\Sigma \in \mathbb{R}^{m \times m}$ is the covariance of $\tilde{\mathbf{W}}$, and the lower bound of $E_c$ is

$$E_c \geq k^{-\frac{1}{2}} m |\Sigma|^\frac{1}{2}.$$  \hspace{1cm} (7)

Similar to PQF, we minimize $|\Sigma|$ to reduce $E_c$.

After LRR learning, each LRR network with a candidate $\tilde{d}$ adopts the following computations:

1) Generating $\tilde{\mathbf{W}}$ for all convolutional layers;
2) Calculating $|\Sigma|$ for $\tilde{\mathbf{W}}$;
3) Adding all the $|\Sigma|$ to represent $E_c$. The quantization regimes for all FC layers are the same, and we do not apply LR$^2$VQ to them. So the $|\Sigma|$ in FC layers can be treated as equal, which can be neglected in each network. After the above computation, the network with the lowest $|\Sigma|$ provides a coarse estimation for $\tilde{d}$, and quantizing the corresponding LRR network is more likely to achieve a better compression-to-accuracy ratio in LR$^2$VQ. We demonstrate the results of $|\Sigma|$ with different $\tilde{d}$ in the later section to validate the coarse estimations of our method.

C. Global Fine-tuning

After clustering, global fine-tuning is necessary to compensate for performance loss caused by $E_c$. In this step, we fix the assigned centroids to all subvectors and fine-tune the compressed network with the loss function and training data in low-rank representation learning. During fine-tuning, all the subvectors will be replaced by Equation (4) and 5 for computation. So the centroids are differentiable and can be updated by gradients as follows:

$$c(i)_{t+1} = c(i)_t - \eta \frac{\partial L}{\partial c(i)_t},$$

where $\hat{L}$ is the loss function, $c(i)$ is the $i$th centroid, and the learning rate is modified to $\hat{\eta}$. This procedure is potent in boosting the model performance of the quantized networks.

D. Inference without $B$

The linear transformation (LT) $B$ is necessary for low-rank representation learning. Nevertheless, once all the parameters are fixed after global fine-tuning, we can eliminate $B$ and the computation in Equation (1). Such elimination is accomplished by the commutative property between LT and Look-Up Table (LUT) operation. After the codebook $C$ is fixed, we can obtain a new codebook $C'$ by

$$C' = C \times B \in \mathbb{R}^{k \times m},$$

and the computation of LR$^2$VQ is modified to

$$W \approx \widetilde{W} = C'(I) = (C \times B)(I) = C(I) \times B$$

Equation (10) implies that only codes $I$ and the new codebook $C'$ are expected to be stored, and the computation in Equation (1) is completely removed by a simple LUT operation $C'(I)$ during inference.

IV. EXPERIMENTS

A. Experiments on ImageNet

In this section, we evaluate our LR$^2$VQ with vanilla ResNet-18, ResNet-50[29] on ImageNet[30] classification datasets.

**Baselines:** We compare our method with PQF and vanilla DKM as they are the most competitive VQ methods up to the writing of this paper. As LR$^2$VQ requires low-rank pre-training before quantization, we do not use the pre-trained models from the Pytorch model zoo. Instead, we implement our training procedures and re-produce PQF and vanilla DKM under the same code base for a fair comparison. As a representative VQ method, PQF has conducted sufficient experiments and set solid baselines, so we follow its compression settings to compare different methods under identical model sizes.

**Compression setups:** Let's denote $c_v$ for $3 \times 3$ convolution, $p_w$ for $1 \times 1$ convolution and $f_c$ for fully-connected layer. To identify the hyperparameters in different convolutional layers in LR$^2$VQ, we use $m_{cv}, m_{pw}, k_{cv}, k_{pw}, \hat{d}_{cv}, \hat{d}_{pw}$ to represent $m, k$ and $\hat{d}$ in $3 \times 3$ and pointwise convolution. We set two compression regimes to achieve different compression ratios. The detailed configurations are shown in Table I. The large blocks regime means fewer codes and a smaller model size for quantized networks. Specifically, the clustering dimensionality for $3 \times 3$ and pointwise convolution in PQF and vanilla DKM are equal to $m_{cv}, m_{pw}$, and $k_{cv}, k_{pw}$ are also the same as in LR$^2$VQ. For FC layers, the dimensionality of subvectors is 4, and $k = 2048$ for ResNet-18 and $k = 1024$ for ResNet-50. For other settings, we follow [19] that we clamp the number of centroids to $\min(k, N/4)$ for stability, and we do not compress the first $7 \times 7$ convolution in ResNets since they take less than 0.1% model size.

| Architecture | Regime | $m_{cv}$ | $m_{pw}$ | $\hat{d}_{cv}$ | $\hat{d}_{pw}$ |
|--------------|--------|----------|----------|----------------|----------------|
| ResNet-18    | Small blocks | 9        | 4        | [1, $m_{cv}$] | [1, $m_{pw}$] |
|              | Large blocks | 18       | 4        |                |                |
| ResNet-50    | Small blocks | 9        | 4        | [1, $m_{cv}$] | [1, $m_{pw}$] |
|              | Large blocks | 18       | 8        |                |                |

**Memory footprint:** Following PQF, we only compress the weights in convolutional and FC layers and ignore the bias in FC and batchnorm layers. We train networks with 32-bit floats but store the codebooks with 16-bit floats. For $k = 256$, all the codes can be stored as 8-bit integers. These settings effectively reduce model sizes with negligible accuracy loss.

**Training details:** The low-rank representation learning uses a total batch size of 1024 on 16 NVIDIA V100 GPUs to train 100 epochs with SGD optimizer plus Nesterov and momentum 0.9. Weight decay is 0.0001, and the learning rate is 0.4 with a cosine annealing scheduler. Label smooth is set to 0.1. We train LRR networks with $\hat{d} \in [3, 7]$ because other values produces either larger $E_a$ or larger $E_c$ to harm quantization. After LRR learning, we run our searching method to estimate a coarse $\hat{d}$, and start quantizing the corresponding LRR network. We run $k$-means for 100 iterations to obtain codebooks and codes, then fine-tune the compressed network with an Adam optimizer. The initial learning rate of fine-tuning is 0.001 and annealed by a cosine scheduler. This procedure runs on 16 GPUs with batch size 2048 for 9 epochs, which takes half an hour for ResNets.

**Results:** Table II shows the comparison of our LR$^2$VQ with $k$-means clustering against PQF and vanilla DKM on standard ResNet-18 and ResNet-50 with the configurations
TABLE II: ImageNet results for PQF, vanilla DKM and LR\textsuperscript{2}VQ. The value of \(\hat{d}_{cv}\) and \(\hat{d}_{pw}\) are shown after LR\textsuperscript{2}VQ’s results.

| Model         | Original top-1 | Original size | Comp. size | Comp. ratio | PQF | DKM | LR\textsuperscript{2}VQ (\(\hat{d}_{cv} / \hat{d}_{pw}\)) |
|---------------|----------------|---------------|------------|-------------|-----|-----|-------------------------------------------------|
| ResNet-18     | 71.30          | 44.59M        | 1.54M      | 29x         | 67.98 | 68.09 | 69.56 (4/4)                                    |
|               |                |               | 1.03M      | 43x         | 64.39 | 64.33 | 67.21 (4/4)                                    |
| ResNet-50     | 77.75          | 97.49M        | 5.09M      | 19x         | 74.93 | oom  | 76.17 (5/4)                                    |
|               |                |               | 3.19M      | 31x         | 73.40 | 73.88 | 74.48 (5/4)                                    |

TABLE III: Compression results for Mask R-CNN on COCO.

| Method                  | Size   | Ratio | AP\textsuperscript{bb} 50 | AP\textsuperscript{bb} 75 | AP\textsuperscript{mk} 50 | AP\textsuperscript{mk} 75 |
|-------------------------|--------|-------|---------------------------|---------------------------|---------------------------|---------------------------|
| Uncompressed in [19]    | 169.40M| -     | 37.9                      | 59.2                      | 41.1                      | 34.6                      |
| PQF in [19]             | 6.65M  | 25.5× | 36.3                      | 57.9                      | 39.4                      | 33.5                      |
| Uncompressed            | 174.37M| -     | 40.34                     | 60.17                     | 43.88                     | 36.54                     |
| PQF                     | 6.65M  | 26.2× | 37.29                     | 57.21                     | 40.65                     | 34.24                     |
| LR\textsuperscript{2}VQ | 6.65M  | 26.2× | 38.20                     | 57.90                     | 41.41                     | 34.93                     |

Fig. 3: Results of LR\textsuperscript{2}VQ with varying model size and more comparisons with different compression methods. SB and LB are small block and large block configurations. Triangle markers indicate the results reported in papers of reference methods.

in Table I. The table shows that LR\textsuperscript{2}VQ outperforms PQF and vanilla DKM across all configurations. For ResNet-18 with large block compression, LR\textsuperscript{2}VQ presents a definite 2.8% improvement over PQF under 43× compression factor. For ResNet-50, LR\textsuperscript{2}VQ consistently outperforms baselines for more than 1% top-1 accuracy under 31× compression factor. Specifically, we mark the proper \(d_{cv}\) and \(d_{pw}\) after the results of LR\textsuperscript{2}VQ. As can be seen, \(d_{cv}\) is 4 or 5 in both compression regimes, which is lower than \(d_{cv} = 9\) or 18 in PQF and vanilla DKM. This result confirms that low-dimensional clustering effectively reduces the clustering error and the reconstruction error. Besides, it proves the advances in jointly considering the compression ratio and the clustering dimensionality, which possess great potential in benefiting vector quantization. Another result we want to discuss here is the performance of vanilla DKM. Based on the computation of DKM, we expect it to be a better clustering method compared to \(k\)-means as the clusters are optimized end-to-end. Contrary to expectations, vanilla DKM has a similar performance to PQF. This phenomenon may be explained by the optimization of DKM also suffers from high-dimensional clustering, which suggests varying the dimensionality in different clustering methods.

Figure 3 provides more results of LR\textsuperscript{2}VQ and other competitive methods. We set \(k_{cv} = 256, 512, 1024, 2048\) to evaluate PQF and LR\textsuperscript{2}VQ with different compression ratio. Results show that our LR\textsuperscript{2}VQ outperforms PQF in all situations. Besides, we provide the results of PQF reported in [19]. As our
uncompressed pretrained models perform better than Pytorch model zoo, we believe our reproduced PQF provides a solid result to perform a fair comparison with LR\(^2\)VQ.

**B. Experiments on COCO**

To generalize LR\(^2\)VQ to different datasets and architectures, we compress Mask R-CNN with LR\(^2\)VQ and experiment on COCO datasets. We first propose low-rank representation learning from scratch to obtain LRRs, then clustering the subvectors in LRRs with \(k\)-means and fine-tuning the overall network on COCO datasets. The pre-trained network size is different from [19] because we utilize the architecture in Detectron2 instead of Pytorch. For a fair comparison, the compression configurations for PQF and LR\(^2\)VQ are the same as in [19]. The results are demonstrated in Table III. Our method obtains 38.20 box AP, which remarkably surpasses PQF with 0.91 AP under 26× compression factor. These results illustrate the generalizability of LR\(^2\)VQ to different vision tasks and architectures.

![Fig. 4: The top-1 accuracy of pre-trained and compressed models with different clustering dimensionality \(\tilde{d}\).](image)

**C. Ablation on the trade-off of \(\tilde{d}\)**

We have discussed the trade-off of \(\tilde{d}\) and proposed assumptions on the variation of errors in Figure 2. Here, we extensively experiment on \(\tilde{d}\) to investigate this trade-off and prove our assumption. We experiment on ImageNet to compare the model performance with different \(\tilde{d}\). We iterate \(\tilde{d}_{cv}\) among [1, 9] (or [1, 18]) with small (or large) block regime in ResNet-18, \(\tilde{d}_{prv}\) among [1, 8] with large block regime in ResNet-50, and \(\tilde{d}_{cv}\) among [1, 9] with small block regime in Mask R-CNN. These settings ensure that the dimensionality of LRRs varies in a wide range. Other configurations are the same as in Table I. The baseline we compared here is PQF. We plot the accuracy of low-rank pre-trained networks and their corresponding quantized networks. Surprisingly, these curves demonstrate a similar trend across different configurations and architectures.

As \(\tilde{d}\) rises, the accuracy of LRR pre-trained models rapidly increases, then oscillates around the original uncompressed network. It is reasonable because many researches have proved that a part of the parameters in neural networks are redundant in computation, and LRR pre-training efficiently removes these redundancies with lower \(\tilde{d}\). The quantized networks’ curves also perform similarly across different settings, which first increase to the top and then decline with a rising \(\tilde{d}\). These trends broadly support the assumptions on \(E_a\) and \(E_r\) in Figure 2 and further guarantees the proper \(\tilde{d}\) can appropriately balance \(E_a\) and \(E_r\) to achieve lower \(E_r\). All the peaks of red curves are the empirical results to the proper \(\tilde{d}\) in LR\(^2\)VQ. Specifically, the accuracy of compressed and uncompressed ResNet-18 with \(\tilde{d} = 1\) is extremely tiny, indicating that the clustering error in 1D space is negligible, so \(E_a\) dominates \(E_r\) to produce a poor performance for the compressed and uncompressed networks. Almost all the LRR pre-trained networks become comparable to the original networks before reaching \(\tilde{d} = m\), which implies enormous parameter redundancy in convolutions. We note that \(d_{cv}\) is fixed to 5 in ResNet-50 with large blocks, which limits the power of \(3 \times 3\) convolutions. So there is always a gap between LRR pre-trained networks and the original uncompressed network. Taking all these results together, the trade-off of \(\tilde{d}\) provides reliable insurance that there must be a proper \(\tilde{d}\) in LR\(^2\)VQ to achieve better quantization performance.

![Fig. 5: Intrinsic dimensionality of subvectors before clustering. Standard and LRR means pre-training with vanilla filter and low-rank representation filter.](image)

### D. Lower intrinsic dimensionality

One unanticipated finding in Figure 4 is that LR\(^2\)VQ outperforms PQF even at \(d_{cv} = m_{cv}\). Intuitively, the clustering difficulty for \(\tilde{d} = m\) is similar to PQF, so the performance of LR\(^2\)VQ and PQF should be close. However, in such circumstances, LR\(^2\)VQ outperforms PQF in all architectures. This discrepancy may be attributed to our low-rank representation learning, which implicitly learns lower intrinsic dimensionality to make the subvectors more favourable for clustering. To validate this speculation, we plot the intrinsic dimensionality of LRRs by principal components analysis (PCA) in each layer with \(d_{cv} = m_{cv}\) in Figure 5. The x-axis is the layer index, and the y-axis is the intrinsic dimensionality with a variance ratio of more than 99.99% after PCA. As the figure shows, the intrinsic dimensionality of LRRs tends to be much lower than the original filters. For example, 11 layers learn lower intrinsic dimensionality among 16 layers in large block compression.
These results suggest that LRRs can automatically learn lower intrinsic dimensionality to benefit clustering.

E. Results of searching $\tilde{d}$

To reduce the fine-tune overhead, a theoretical analysis to refine candidate $\tilde{d}$s is very valuable. Here, we demonstrate the estimation of our coarse method for searching $\tilde{d}$. As described in previous sections, we apply our searching method with $\tilde{d} \in [3, 7]$. Figure 6 presents the summation of $|\Sigma|$ for different $\tilde{d}$. As a starting point, our coarse estimation of the proper $\tilde{d}$ using Equation (7) shows agreements with experimental results. All these results are reasonable because the distribution of subvectors also affects clustering. With a fixed number of centroids, reducing $\tilde{d}$ can decrease the clustering difficulty, but hard-for-clustering subvectors also produce significant clustering errors. Therefore, a lower clustering dimensionality does not equal to a lower clustering error. In summary, our coarse estimation provides the right direction towards the proper $\tilde{d}$.

Fig. 6: The estimation of our coarse analytical method for small blocks compression. Empirical means $\tilde{d}$ is given by the experimental results in Figure 4. $\tilde{d}$ with a lower $|\Sigma|$ is the better estimation.

V. CONCLUSION

We proposed LR$^2$VQ a new vector quantizer for deep neural network compression. It is demonstrated that a well-designed combination of dimensionality reduction and vector quantization could reach state-of-the-art final network accuracy under typical compression ratios. The key to promising an excellent model compression performance using LR$^2$VQ lies in the choice of a hyper-parameter, i.e. the clustering dimensionality $\tilde{d}$, which is modeled as the determining factor of both low-rank approximation error and clustering error. A theoretical evaluation of quantization error is thus provided to help anticipate an optimal $\tilde{d}$, dramatically reducing training overhead, and is validated by our ablation experiments. This work should help future developments of comprehensive neural network compression schemes.

REFERENCES

[1] X. Zhang, X. Zhou, M. Lin, and J. Sun, “Shufflenet: An extremely efficient convolutional neural network for mobile devices,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 6848–6856.

[2] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “Mobilenetv2: Inverted residuals and linear bottlenecks,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 4510–4520.

[3] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan et al., “Searching for mobilenetv3,” in Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 1314–1324.

[4] M. Tan and Q. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks,” in International conference on machine learning. PMLR, 2019, pp. 6105–6114.

[5] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, “Learning transferable architectures for scalable image recognition,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 8697–8710.

[6] Z. Zhong, Z. Yang, B. Deng, J. Yan, W. Wu, J. Shao, and C.-L. Liu, “Blockqnn: Efficient block-wise neural network architecture generation,” IEEE transactions on pattern analysis and machine intelligence, vol. 43, no. 7, pp. 2314–2328, 2020.

[7] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, “Efficient neural architecture search via parameters sharing,” in International conference on machine learning. PMLR, 2018, pp. 4095–4104.

[8] Y. Zhang, J. Zhang, and Z. Zhong, “Autobss: An efficient algorithm for block stacking style search,” Advances in Neural Information Processing Systems, vol. 33, pp. 10 259–10 270, 2020.

[9] M. Fang, Q. Wang, and Z. Zhong, “Betanas: Balanced training and selective drop for neural architecture search,” arXiv preprint arXiv:1912.11191, 2019.

[10] Y. LeCun, J. Denker, and S. Solla, “Optimal brain damage,” in Advances in Neural Information Processing Systems, D. Touretzky, Ed., vol. 2. Morgan-Kaufmann, 1990.

[11] M. Courbariaux, Y. Bengio, and J.-P. David, “Binaryconnect: Training deep neural networks with binary weights during propagations,” Advances in neural information processing systems, vol. 28, 2015.

[12] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, “Quantized neural networks: Training neural networks with low precision weights and activations,” The Journal of Machine Learning Research, vol. 18, no. 1, pp. 6869–6898, 2017.

[13] H. Jegou, M. Douze, and C. Schmid, “Product quantization for nearest neighbor search,” IEEE transactions on pattern analysis and machine intelligence, vol. 33, no. 1, pp. 117–128, 2010.

[14] Y. Gong, L. Liu, M. Yang, and L. Bourdev, “Compressing deep convolutional networks using vector quantization,” arXiv preprint arXiv:1412.6115, 2014.

[15] S. Son, S. Nah, and K. M. Lee, “Clustering convolutional
kernels to compress deep neural networks,” in ECCV, September 2018.

[16] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, “Quantized convolutional neural networks for mobile devices,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 4820–4828.

[17] P. Stock, A. Joulin, R. Gribonval, B. Graham, and H. Jégou, “And the bit goes down: Revisiting the quantization of neural networks,” in International Conference on Learning Representations (ICLR), 2020.

[18] W. Chen, P. Wang, and J. Cheng, “Towards convolutional neural networks compression via global & progressive product quantization,” in BMVC, 2020.

[19] J. Martinez, J. Shewakramani, T. W. Liu, I. A. Bârsan, W. Zeng, and R. Urtasun, “Permute, quantize, and fine-tune: Efficient compression of neural networks,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 15 699–15 708.

[20] P. Indyk and R. Motwani, “Approximate nearest neighbors: Towards removing the curse of dimensionality,” Conference Proceedings of the Annual ACM Symposium on Theory of Computing, vol. 604–613, 10 2000.

[21] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, “Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡0.5mb model size,” 2016.

[22] Y. Guo, A. Yao, and Y. Chen, “Dynamic network surgery for efficient dnnns,” Advances in neural information processing systems, vol. 29, 2016.

[23] S. Han, J. Pool, J. Tran, and W. Dally, “Learning both weights and connections for efficient neural network,” Advances in neural information processing systems, vol. 28, 2015.

[24] Y. He, X. Zhang, and J. Sun, “Channel pruning for accelerating very deep neural networks,” in Proceedings of the IEEE international conference on computer vision, 2017, pp. 1389–1397.

[25] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, “Pruning filters for efficient convnets,” in International Conference on Learning Representations, 2017. [Online]. Available: https://openreview.net/forum?id=rJqFGTs1g

[26] J.-H. Luo, J. Wu, and W. Lin, “Thinet: A filter level pruning method for deep neural network compression,” in Proceedings of the IEEE international conference on computer vision, 2017, pp. 5058–5066.

[27] M. Cho, K. Alizadeh-Vahid, S. Adya, and M. Rastegari, “DKM: Differentiable k-means clustering layer for neural network compression,” in International Conference on Learning Representations, 2022.

[28] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in Proceedings of the IEEE international conference on computer vision, 2015, pp. 1026–1034.

[29] ———, “Deep residual learning for image recognition,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[30] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein et al., “Imagenet large scale visual recognition challenge,” International journal of computer vision, vol. 115, no. 3, pp. 211–252, 2015.