Synergistic Self-supervised and Quantization Learning

Yun-Hao Cao[1], Peiqin Sun[2*], Yechang Huang[2], Jianxin Wu[1], and Shuchang Zhou[2]

[1] State Key Laboratory for Novel Software Technology, Nanjing University, China
[2] MEGVII Technology
{caoyunhao1997, wujx2001}@gmail.com, {sunpeiqin, huangyechang, zsc}@megvii.com

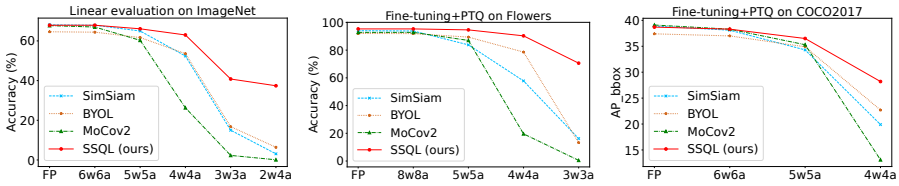
Abstract. With the success of self-supervised learning (SSL), it has become a mainstream paradigm to fine-tune from self-supervised pretrained models to boost the performance on downstream tasks. However, we find that current SSL models suffer severe accuracy drops when performing low-bit quantization, prohibiting their deployment in resource-constrained applications. In this paper, we propose a method called synergistic self-supervised and quantization learning (SSQL) to pretrain quantization-friendly self-supervised models facilitating downstream deployment. SSQL contrasts the features of the quantized and full precision models in a self-supervised fashion, where the bit-width for the quantized model is randomly selected in each step. SSQL not only significantly improves the accuracy when quantized to lower bit-widths, but also boosts the accuracy of full precision models in most cases. By only training once, SSQL can then benefit various downstream tasks at different bit-widths simultaneously. Moreover, the bit-width flexibility is achieved without additional storage overhead, requiring only one copy of weights during training and inference. We theoretically analyze the optimization process of SSQL, and conduct exhaustive experiments on various benchmarks to further demonstrate the effectiveness of our method.

Keywords: Quantization, self-supervised learning, transfer learning

1 Introduction

Deep supervised learning has achieved great success in the last decade. However, traditional supervised learning approaches rely heavily on a large set of annotated training data. Self-supervised learning (SSL) has gained popularity because of its ability to avoid the cost of annotating large-scale datasets as well as the ability to obtain task-agnostic representations [26]. After the emergence of the contrastive learning (CL) paradigm [4,16], SSL has clearly gained momentum and several recent works [5,13,6] have achieved comparable or even better accuracy than the supervised pretraining when transferring to downstream tasks. A

* Corresponding author.



(a) ImageNet (b) Flowers (c) COCO2017 detection

Fig. 1: ImageNet linear evaluation and transfer results using ImageNet pretrained models. Directly applying current self-supervised contrastive methods does not work well for low-bit quantization when transferring, while our method (SSQL) leads to a dramatic performance boost. See Section 4.3 for details. '2w4a' means the weights are quantized to 2 bits and activations to 4 bits, etc.

standard pipeline for SSL is to learn representations (i.e., pretrained backbone networks) on unlabeled datasets and then transfer to various downstream tasks (e.g., image classification [18] and object detection [17]) by fine-tuning.

With the fast development of self-supervised learning, an increasing proportion of the models that need to be deployed in downstream tasks are fine-tuned from SSL pretrained models. When we want to deploy them on some resource-constrained devices, it is essential to reduce the memory consumption and latency of the neural network. To facilitate deployment, several model compression techniques have been proposed, including lightweight architecture design [35,42], knowledge distillation [19], network pruning [14,27], and quantization [43,9]. Among them, quantization is one of the most effective methods and is directly supported by most current hardware. But severe accuracy degradation is often encountered during quantization, especially in the case of low bit-widths. As shown in Fig. 1, although current state-of-the-art self-supervised learning methods achieve impressive performance with full precision (FP) models, they all incur severe drop in accuracy when bit-width goes below 5. Inspired by SSL that can learn a good representation shared by various downstream tasks, we are thus motivated to ask a question: "Can we learn a quantization-friendly representation such that the pretrained model can be quantized more easily to facilitate deployment when transferring to different downstream tasks?".

We propose Synergistic Self-supervised and Quantization Learning (SSQL) by contrasting features of the quantized and full precision models as our solution: *SSL and quantization become synergistic—they help each other*. On one hand, the contrastive loss encourages similarity of the quantized and FP models. On the other hand, quantization improves SSL by encouraging feature consistency under differently augmented weights/activations. Our contributions are:

- To the best of our knowledge, we are the first to propose quantization-friendly training for SSL. We design an effective method called SSQL, which not only greatly improves the performance when quantized to low bit-widths, but also boosts the performance of full precision models in most cases.

- With SSQL, models only need to be trained *once* and can then be customized for a variety of downstream tasks at different bit-widths, allowing flexible speed-accuracy trade-off for real-world deployment. The bit-width flexibility is achieved without additional storage overhead, as only *one copy of weights* needs to be kept, both in the training and inference stage.
- SSQL is versatile. First, it can be combined with existing negative-based/free CL methods. Second, the pretrained models of SSQL are compatible with existing quantization methods to further boost the performance when quantizing.
- We provide theoretical analysis about the synergy between SSL and quantization in SSQL. Exhaustive experimental results further show that our SSQL achieves better performance on various benchmarks at all bit-widths.

2 Related Works

Network Quantization. Quantization is a method that converts the weights and activations in networks from full precision (i.e., 32-bit floating-point) to fixed-point integers. According to whether or not quantization is introduced into the training process, network quantization can be divided into two categories: Quantization-Aware Training (QAT) and Post-Training Quantization (PTQ). QAT methods [43,7,9] introduce a simulated quantization operation in the training stage. While it generally closes the gap to full precision accuracy compared to PTQ for low-bit quantization, it requires more effort in training and potentially hyperparameter tuning. In contrast, PTQ methods [20,29,23] take a trained full precision network and quantize it with little or no data [30], which requires minimal hyperparameter tuning and no end-to-end training. In this work, we introduce quantization into self-supervised learning to get a quantization-friendly pretrained model. Our pretrained model is compatible with existing QAT and PTQ methods when transferred to downstream tasks and hence can be combined to further improve performance.

AdaBits [21] enables adaptive bit-widths of weights and activations, but is a supervised learning method. Our pretrained model can adapt to different bit-widths, thus our work is also a method that only trains once for all bits, but in an unsupervised manner. More importantly, AdaBits focuses on the current task while we investigate the transfer ability of our models and also evaluate the quantization property on downstream tasks. OQAT [36] explores extremely low-bit architecture search by combining network architecture search methods with quantization. There are also works that study quantization-friendly properties. GDRQ [40] reshapes weights or activations into a uniform-like distribution dynamically. [15] proposes a bin regularization algorithm to improve low-bit network quantization. [38] proposes a quantization-friendly separable convolution for MobileNets. In contrast, we consider quantization-friendly properties from the perspective of pretraining under the self-supervised paradigm.

Self-supervised Learning. To avoid time-consuming and expensive data annotations and to explore better representations, many self-supervised methods were proposed to learn visual representations from large-scale unlabeled images

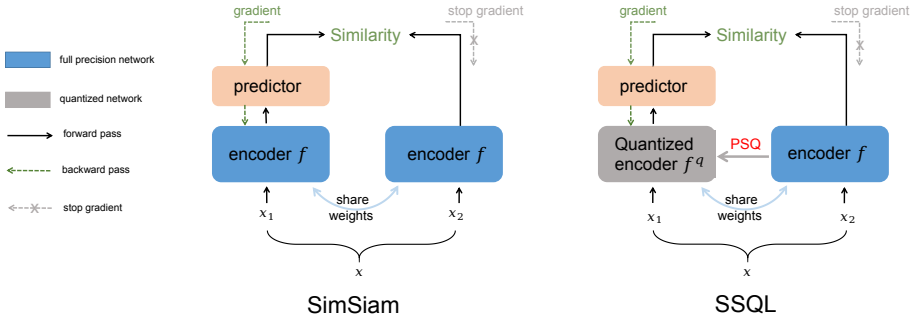


Fig. 2: Illustration of our method. Left: SimSiam [6]. Right: The proposed method SSQL. 'PSQ' denotes post step quantization, see Sec. 3.2 for details.

or videos [41,31,8,2,12]. As the driving force of state-of-the-art SSL methods, contrastive learning methods greatly improve the performance of representation learning in recent years [32,16,4,13,3,6]. Contrastive learning is a discriminative approach that aims at pulling similar samples closer and pushing diverse samples far from each other. SimCLR [4] and MoCo [16] both employ a contrastive loss function InfoNCE [32], which requires negative samples. A more radical step is made by BYOL [13], which discards negative sampling in contrastive learning but achieves even better results in case a momentum encoder is used. [6] proposes a follow-up work SimSiam and demonstrates that simple siamese networks can learn meaningful representations even without the momentum encoder. However, previous works did not consider whether the pretrained model is quantization-friendly when transferring to downstream tasks.

SEED [11] uses self-supervised knowledge distillation for SSL with small models. S^2-BNN [37] investigates training self-supervised binary neural networks (BNN) by distilling knowledge from real networks. However, they all require a pretrained model as the teacher for distillation while ours does not. Moreover, [37] is tailored for BNN while our method is adaptive to different bit-widths. More importantly, our method can improve the performance of the full precision (FP) model over the baseline counterpart by encouraging feature consistency under differently augmented weights/activations via quantization.

3 Method

In this section, we introduce our approach, which we called synergistic self-supervised quantization learning (SSQL). We begin with the basic notation and a brief review of previous works, followed by our algorithm and analysis.

3.1 Background and notation

Let $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ denote two randomly augmented views from an input image $\boldsymbol{x}$. Let f denote an encoder network consisting of a backbone (e.g., ResNet [18])

and a projection MLP head [4]. By default we use SimSiam [6] as the baseline counterpart to develop our algorithm, as shown in Fig. 2.

SimSiam maximizes the similarity between two augmentations of one image. A prediction MLP head [13], denoted as h, transforms the output of one view and matches it to the other view. The output vectors for $\boldsymbol{x}_1$ is denoted as $\boldsymbol{z}_1 \triangleq f(\boldsymbol{x}_1)$ and $\boldsymbol{p}_1 \triangleq h(f(\boldsymbol{x}_1))$, and $\boldsymbol{z}_2$ and $\boldsymbol{p}_2$ are defined similarly.

The negative cosine similarity is defined as $D(\boldsymbol{p}, \boldsymbol{z}) \triangleq -\frac{\boldsymbol{p}}{\|\boldsymbol{p}\|_2} \cdot \frac{\boldsymbol{z}}{\|\boldsymbol{z}\|_2}$ and we assume both $\boldsymbol{z}$ and $\boldsymbol{p}$ have been l_2-normalized for simplicity in the following. Let $SG(\cdot)$ denote the stop-gradient operation. Then, the objective to be minimized in SimSiam is then:

$$L_{\text{SimSiam}} = D(\boldsymbol{p}_1, SG(\boldsymbol{z}_2)) + D(\boldsymbol{p}_2, SG(\boldsymbol{z}_1)) \,. \tag{1}$$

3.2 Our method

Our motivation is to train a quantization-friendly pretrained model, hence we proposed to introduce quantization into contrastive learning. We denote f^q as the quantized version of f, where q is the assigned quantization bit-width. Correspondingly, the resulting outputs become $\boldsymbol{z}^q$ and $\boldsymbol{p}^q$. We simply adopt the commonly used uniform quantizer for both weights and activations:

$$X_{int} = clip\left(\lfloor \frac{X}{S} + Z \rceil, 0, 2^q - 1\right) , \tag{2}$$

$$X_q = (X_{int} - Z)S \,, \tag{3}$$

where S (scale) and Z (zero-point) are quantization parameters determined by the lower bound l and the upper bound u of X, while X can be either the model weights or activations. We use minimum and maximum values for l and u:

$$l = \min(X), u = \max(X) \,, \tag{4}$$

$$S = \frac{u - l}{2^q - 1} \,. \tag{5}$$

Our solution SSQL is to let the quantized encoder f^q predict the output of the full precision (FP) encoder f (i.e., use FP outputs as the target):

$$L_{SSQL} = D(\boldsymbol{p}_1^q, SG(\boldsymbol{z}_2)) + D(\boldsymbol{p}_2^q, SG(\boldsymbol{z}_1)) \,. \tag{6}$$

It is worth noting that we need only one copy of the model weights, which is f. f^q can be obtained directly from f using (2) and (3). Further, we can add the auxiliary SimSiam loss to improve performance by combining (1) and (6):

$$L_{SSQL\text{-}aux} = L_{SimSiam} + L_{SSQL} \,. \tag{7}$$

In order to make the model quantization-friendly to different bit-widths, we *randomly select* values from a set of candidate bit-widths *in each step* for the

assignment of q. In addition, we also find that this random selection operation, as a kind of augmentation, brings a performance boost. We use $2 \sim 8$ and $4 \sim 8$ bits for weight and activation, respectively, in all our experiments. Also, we quantize f to get f^q after each step to ensure consistency, which we name as *post step quantization* (PSQ). Notice that we calculate S and Z during the forward pass of f and hence PSQ brings negligible overhead. During the backward pass, we adopt the straight-through estimator (STE) [1] for the quantization step. Notice that the quantized network and the full precision network *share weights*, hence when we backprop on the quantized network f_q using STE, the gradients will directly operate on the full precision network f. We will discuss the impact of the choice of loss functions and the candidate bit-widths set in Sec. 4.4.

3.3 The synergy between SSL and quantization

Following the notations and analyses in [6], the optimization process can be viewed as an implementation of an Expectation-Maximization (EM) like algorithm. The loss function of SSQL can be organized in the following form:

$$\mathcal{L}(\theta, \eta) = \mathbb{E}_{x,\mathcal{T},q}[\|\mathcal{F}_\theta^q(\mathcal{T}(x)) - \eta_x\|_2^2]\,, \tag{8}$$

where $\mathcal{F}_\theta$ is a network parameterized by θ, $\mathcal{F}_\theta^q$ is obtained by quantizing $\mathcal{F}_\theta$, $\mathcal{T}$ is the augmentation and x is an image. The expectation $\mathbb{E}[\cdot]$ is over the distribution of images, augmentations and bit-widths. η_x is the representation of image x.

With the formulation of Eq. (8), we consider solving

$$\min_{\theta,\eta} \mathcal{L}(\theta, \eta)\,. \tag{9}$$

The problem in (9) can be solved by alternating between two subproblems:

$$\theta^t \leftarrow \arg\min_\theta \mathcal{L}(\theta, \eta^{t-1})\,; \quad \eta^t \leftarrow \arg\min_\eta \mathcal{L}(\theta^t, \eta)\,. \tag{10}$$

Here t is the index of alternation and "$\leftarrow$" means assigning. The optimization step for η^t is the same as [6] and we analyze the optimization step for θ^t:

$$\theta^{t+1} \leftarrow \arg\min_\theta \mathbb{E}_{x,\mathcal{T},q}\left[\|\mathcal{F}_\theta^q(\mathcal{T}(x)) - \mathcal{F}_{\theta^t}(\mathcal{T}'(x))\|_2^2\right]. \tag{11}$$

Here $\mathcal{T}'$ implies another view and detailed derivation of (11) is included in the appendix. Moreover, we have

$$\mathbb{E}_{x,\mathcal{T},q}\left[\|\mathcal{F}_\theta^q(\mathcal{T}(x)) - \mathcal{F}_{\theta^t}(\mathcal{T}'(x))\|_2^2\right] \tag{12}$$

$$= \mathbb{E}_{x,\mathcal{T},q}\left[\|\mathcal{F}_\theta^q(\mathcal{T}(x)) - \mathcal{F}_\theta(\mathcal{T}(x)) + \mathcal{F}_\theta(\mathcal{T}(x)) - \mathcal{F}_{\theta^t}(\mathcal{T}'(x))\|_2^2\right] \tag{13}$$

$$= \underbrace{\mathbb{E}_{x,\mathcal{T},q}\left[\|\mathcal{F}_\theta^q(\mathcal{T}(x)) - \mathcal{F}_\theta(\mathcal{T}(x))\|_2^2\right]}_{\text{Q term (quantization term)}} + \underbrace{\mathbb{E}_{x,\mathcal{T},q}\left[\|\mathcal{F}_\theta(\mathcal{T}(x)) - \mathcal{F}_{\theta^t}(\mathcal{T}'(x))\|_2^2\right]}_{\text{CL term (contrastive learning term)}} \tag{14}$$

$$+ \underbrace{2\mathbb{E}_{x,\mathcal{T},q}\left[\left(\mathcal{F}_\theta^q(\mathcal{T}(x)) - \mathcal{F}_\theta(\mathcal{T}(x))\right)^T\left(\mathcal{F}_\theta(\mathcal{T}(x)) - \mathcal{F}_{\theta^t}(\mathcal{T}'(x))\right)\right]}_{\text{cross term}}. \tag{15}$$

It is reasonable to assume that the quantization error and the contrastive learning error are at most weakly correlated (see appendix for empirical verification), hence we can remove the cross term and are left with two objectives in the optimization step for θ. The Q term minimizes the distance between the quantized network $\mathcal{F}_\theta^q$ and the FP network $\mathcal{F}_\theta$, which naturally leads to the desired quantization-friendly property. The CL term is the original optimization term in SimSiam to learn image representations. Also notice that we take expectations over 3 terms, where the extra q term can be seen as one kind of augmentation on weights/activations. It is well-known that strong image augmentations are essential in SSL [4]. Hence, the quantization can potentially assist the learning of SSL, by encouraging feature consistency under differently augmented weights/activations via quantization. In conclusion, the design of our loss function makes quantization and SSL work in a synergistic fashion.

4 Experiments

We introduce the implementation details in Sec. 4.1. We experiment on CIFAR-10 and CIFAR-100 [22] in Sec. 4.2 and ImageNet [34] (IN) in Sec. 4.3. Then, we evaluate the transfer performance of ImageNet pretrained models on downstream classification and object detection benchmarks in Sec. 4.3. Finally, we study the effects of different components and hyper-parameters in our algorithm in Sec. 4.4.

4.1 Implementation details

Datasets. The main experiments are conducted on three benchmark datasets, i.e., CIFAR-10, CIFAR-100 [22] and ImageNet [34]. We also conduct transfer experiments on 7 recognition benchmarks (see appendix for details) as well as 2 detection benchmarks Pascal VOC 07&12 [10] and COCO2017 [25].
Backbones. Apart from the commonly used ResNet-50 [18] in recent SSL papers, we also adopt 2 smaller networks, i.e., ResNet-18 [18] and ResNet-34 [18] for our experiments. We use the same settings as [6] for prediction and projection MLP. Sometimes we abbreviate ResNet-18/50 to R-18/50.
Training details. We follow the training setup in SimSiam [6] for our method. More specifically, we use SGD for pretraining, with batch size of 256 and a base lr=0.05. The learning rate has a cosine decay schedule. The weight decay is 0.0001 and the SGD momentum is 0.9. We pretrain for 400 epochs on CIFAR-10 and CIFAR-100 and 100 epochs on ImageNet unless otherwise specified. Please see appendix for more training details for linear evaluation and fine-tuning.
Evaluation protocols. Following previous works [16], we adopt linear evaluation and fine-tuning to evaluate the pretrained representations. Moreover, we want to evaluate the performance of the representations after quantization. Hence, we make corresponding adjustments and propose a new evaluation protocol when combining quantization and SSL, as shown in Fig. 3. More specifically, we freeze and quantize the backbone and only update the classification head for linear evaluation (i.e., backbone weights frozen). For fine-tuning, we first train

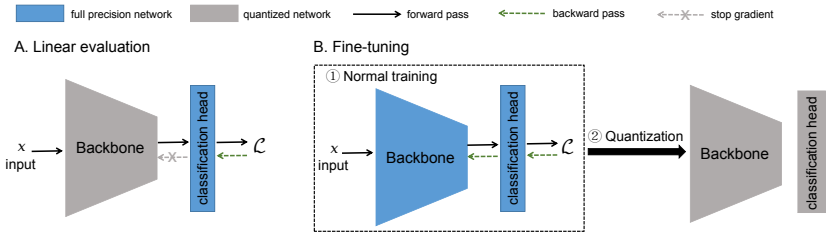


Fig. 3: Illustration of the evaluation protocols adopted in our paper.

Table 1: Linear evaluation results on CIFAR-10. All pretrained for 400 epochs. SimSiam-PACT trains 7 models separately and we color it grey.

Backbone	Method	Linear evaluation accuracy (%)							
		FP	8w8a	6w6a	5w5a	4w4a	3w3a	2w8a	2w4a
ResNet-18	SimSiam [6]	**90.7**	90.7	**90.6**	90.3	88.9	66.0	70.1	63.8
	BYOL [13]	89.3	89.3	89.4	89.3	88.0	75.1	71.9	63.3
	SimSiam-PACT [7]	-	89.2	89.2	89.3	89.2	88.2	89.3	88.3
	SSQL (ours)	**90.7**	**90.8**	**90.6**	**90.6**	**90.1**	**85.6**	**88.0**	**86.5**
	SimCLR [4]	**89.4**	**89.3**	**89.2**	**88.8**	87.1	73.9	65.6	55.6
	MoCov2 [5]	88.9	88.8	88.4	88.2	86.8	72.2	66.4	50.7
	SSQL-NCE (ours)	89.0	89.0	89.0	**88.8**	**87.9**	**82.9**	**87.1**	**84.9**
ResNet-50	SimSiam [6]	90.9	90.9	91.0	90.6	89.5	74.1	55.1	57.1
	BYOL [13]	90.3	90.3	90.0	89.7	87.5	58.5	82.4	67.8
	SSQL (ours)	**91.1**	**91.1**	**91.1**	**91.1**	**90.0**	**77.4**	**89.5**	**87.2**
	SimCLR [4]	91.5	91.4	91.3	90.5	88.1	59.6	63.5	42.4
	MoCov2 [5]	90.2	90.2	90.2	89.4	87.9	72.1	68.8	49.5
	SSQL-NCE (ours)	**92.1**	**92.1**	**92.0**	**91.9**	**89.8**	**74.0**	**88.6**	**84.9**

the backbone as well as the classification head as normal (i.e., backbone weights updated). Then, based on the fine-tuned FP model, we conduct either PTQ or QAT to evaluate the performance after quantization. We adopt PTQ after fine-tuning in our experiments by default. We use 'nwma' to denote that we quantize weight to n-bit and quantize activation to m-bit in this paper (e.g., 4w4a).

4.2 CIFAR results

We compare our method with popular SSL methods BYOL [13], SimSiam [6], SimCLR [4] and MoCov2 [5]. We evaluate the linear evaluation accuracy under different bit-widths after quantization, as mentioned in Sec 4.1. Notice that we only pretrain one full precision (FP) model and then use it for evaluation on different bit-widths. To better illustrate the effectiveness of our method, we also create one strong baseline SimSiam-PACT, by combining PACT [7] and SimSiam during pretraining. Notice that it is not a fair comparison with other methods because it needs to pretrain different models for different bit-widths (i.e., need 7 pretrained models for 7 bit-widths). In other words, it is not flexible and the training overhead is unbearable for large data volumes. Experimental results on CIFAR-10 and CIFAR-100 are shown in Table 1 and Table 2, respectively.

Table 2: Linear evaluation results on CIFAR-100. All pretrained for 400 epochs.

Backbone	Method	Linear evaluation accuracy (%)							
		FP	8w8a	6w6a	5w5a	4w4a	3w3a	2w8a	2w4a
ResNet-18	SimSiam [6]	65.5	65.5	65.4	64.6	62.6	41.6	40.1	36.9
	BYOL [13]	62.6	62.6	62.5	62.0	60.6	47.9	44.1	38.8
	SimCLR [4]	59.2	59.2	59.0	57.9	54.4	34.1	38.4	28.8
	MoCov2 [5]	62.5	62.5	62.1	61.5	59.5	43.5	40.1	30.8
	SSQL (ours)	**66.9**	**66.8**	**66.9**	**65.8**	**65.0**	**57.4**	**53.9**	**50.6**
ResNet-50	SimSiam [6]	64.3	64.2	64.1	62.9	61.3	44.9	32.9	32.6
	BYOL [13]	66.7	66.5	65.0	59.6	47.2	14.5	55.3	27.2
	SimCLR [4]	66.2	66.1	65.9	64.8	60.1	40.2	43.8	24.7
	MoCov2 [5]	66.5	66.5	66.3	65.4	61.9	44.2	41.1	28.5
	SSQL (ours)	**68.0**	**67.9**	**67.8**	**67.8**	**67.8**	**59.9**	**62.9**	**61.5**

As shown in Table 1, take ResNet-18 as an example, our SSQL achieves comparable performance with the baseline counterpart SimSiam under linear evaluation in full precision on CIFAR-10. However, when we lower the bit-width (from 8w8a to 2w4a), our advantages over the baseline SimSiam will become more and more obvious. For instance, our SSQL achieves **19.6%** and **22.7%** higher accuracy than SimSiam at 3w3a and 2w4a, respectively. When comparing with SimSima-PACT, we can find that our SSQL achieves higher accuracy at 4w4a and above. However, SimSiam-PACT achieves slightly higher accuracy than our method at 3w3a and below but the gap is within 3%. Moreover, we achieve higher accuracy than SimSiam under ResNet-50 at FP, and the advantages when reducing bit-widths are consistent. Finally, our SSQL can also be combined with InfoNCE [32] based methods, e.g., SimCLR and we name it SSQL-NCE. We can observe similar trends as above and it demonstrates that our SSQL is compatible with both negative-based and negative-free CL methods.

As shown in Table 2, our SSQL achieves the highest accuracy on CIFAR-100 in all cases. For instance, when comparing the first column (FP), our SSQL is significantly better than baseline counterpart SimSiam: up to **+1.4%** and **+3.7%** accuracy for ResNet-18 and ResNet-50, respectively. Our advantages become bigger when we further lower the bit-widths: up to **+6.5%**, **+15%** and **+28.9%** accuracy at 4w4a, 3w3a and 2w4a, respectively, for ResNet-50.

To demonstrate the effectiveness of the proposed method in a more intuitive way, we visualize the feature spaces learned by different methods in Fig. 4. First, three models are trained on the CIFAR-10 dataset by using SimCLR, SimSiam and SSQL, respectively. After that, 5,000 samples in CIFAR-10 are represented accordingly and then are reduced to a two-dimensional space by t-SNE [28]. As seen, the samples are more separable in the feature space learned by SSQL than both SimCLR and SimSiam (especially at 2w8a and 2w4a), showing that SSQL can learn better feature representations after quantization.

4.3 ImageNet and transfer learning results

In this section, we do unsupervised pretraining on the large-scale ImageNet training set [34] without using labels. The linear evaluation results on ImageNet are

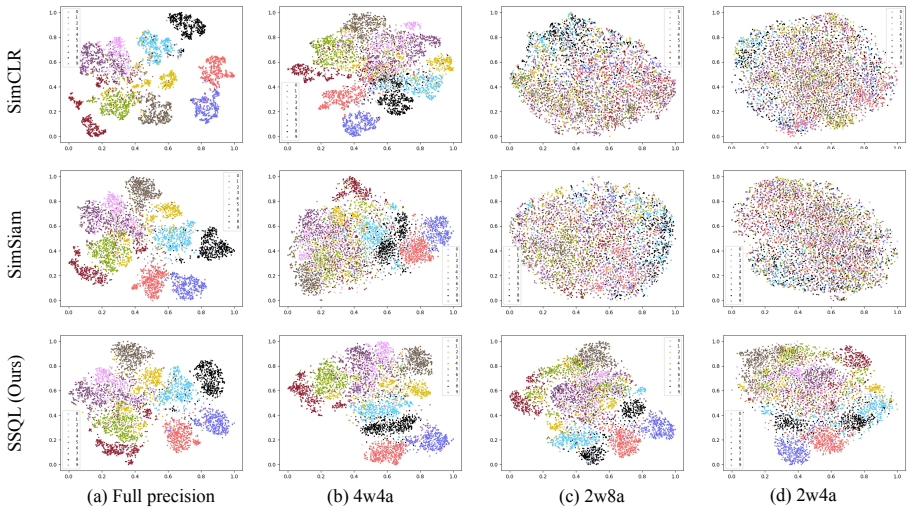


Fig. 4: t-SNE [28] visualization of CIFAR-10 using ResNet-18. The column (a) shows the results using FP backbone. The column (b), (c) and (d) shows the results at 4w4a, 2w8a and 2w4a, respectively. This figure is best viewed in color.

Table 3: Linear evaluation results on ImageNet. All pretrained for 100 epochs, except for MoCov2. † denotes that we use the official MoCov2 200ep checkpoint. SimSiam-PACT trains 5 models separately and we color it grey.

Backbone	Method	Linear evaluation accuracy (%)					
		FP	8w8a	5w5a	4w4a	3w3a	2w4a
ResNet-18	SimSiam [6]	55.0	54.7	53.9	36.7	6.3	1.5
	BYOL [13]	54.1	54.0	51.9	42.4	13.6	3.6
	SimSiam-PACT [7]	-	52.8	52.8	52.3	51.0	51.6
	SSQL (ours)	**57.6**	**57.6**	**56.7**	**52.8**	**41.0**	**43.1**
ResNet-50	SimSiam [6]	**68.1**	**67.9**	65.0	52.4	15.0	3.1
	BYOL [13]	64.6	64.4	61.7	53.6	16.8	6.4
	MoCov2† [5]	67.7	67.0	60.3	26.3	2.3	0.1
	SSQL (ours)	67.9	**67.9**	**66.1**	**63.0**	**40.8**	**37.4**

shown in Table 3. Also, we evaluate the transfer ability of the learned representations on ImageNet later. We train SSQL, SimSiam and BYOL for 100 epochs on ImageNet and directly use the official checkpoint for MoCov2.

As shown in Table 3, when comparing the first column (FP), our SSQL achieves higher accuracy than the baseline counterpart SimSiam (57.6 v.s. 55.0) under ResNet-18. When comparing the fourth column (4w4a), our SSQL achieves 16.1% and 10.6% gains for ResNet-18 and ResNet-50, respectively. In short, our SSQL achieves comparable or better accuracy at full precision and is more quantization-friendly at lower bit-widths. When compared with SimSiam-PACT, our SSQL achieves better results at 4 bits or higher, *with only one copy of weights*.

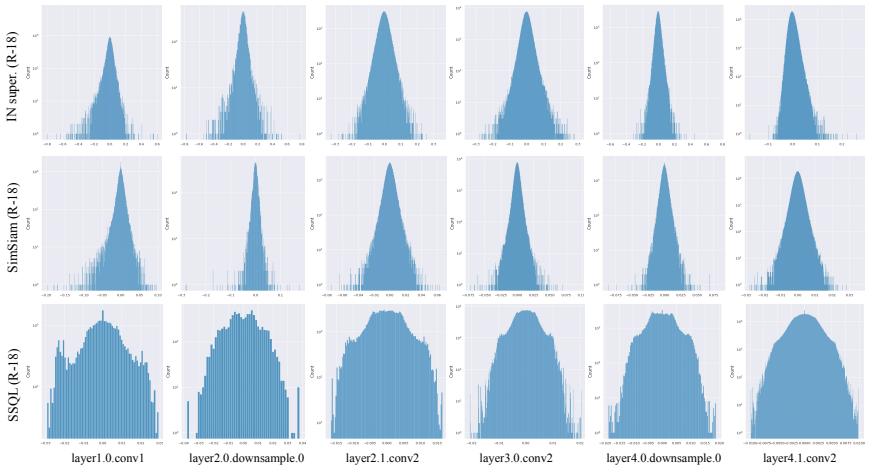


Fig. 5: Visualization of weight distribution for ResNet-18. The first, second and third row are the results of ImageNet supervised, SimSiam and ours, respectively.

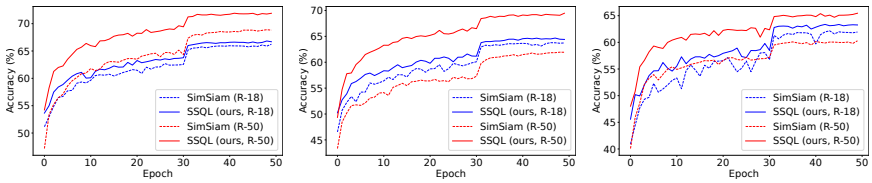
Table 4: Fine-tuning+PTQ results on ImageNet subsets. Here we adopt the fine-tuning settings on 1%/10% labeled data and report Top-5 accuracy (%).

Backbone	Method	1% labels				10% labels			
		FP	6w6a	5w5a	4w4a	FP	6w6a	5w5a	4w4a
ResNet-18	SimSiam [6]	43.7	43.4	42.4	37.5	**76.1**	75.8	74.3	64.5
	BYOL [13]	36.7	36.5	35.5	31.2	75.5	75.1	73.9	65.2
	SSQL (ours)	**47.7**	**47.6**	**47.1**	**45.0**	**76.1**	**75.9**	**75.0**	**70.7**
ResNet-50	SimSiam [6]	53.2	52.8	51.5	36.4	82.5	81.7	79.0	67.9
	BYOL [13]	47.3	47.2	46.4	40.4	81.1	80.7	79.6	69.9
	SSQL (ours)	**55.2**	**55.0**	**54.4**	**51.8**	**83.0**	**82.7**	**81.0**	**76.7**

As shown in Fig. 1, the ImageNet linear evaluation performance can somehow indicate the performance at downstream tasks at different bit-widths (i.e., the trend is consistent). We plot the weight distribution of different pretrained models in Fig. 5. As seen, the weights of our model (third row) are more quantization-friendly when compared with the two baseline counterparts in terms of 3 aspects: more uniform distribution, smaller ranges, and much fewer outliers. (There is a similar phenomenon after fine-tuning on downstream tasks, too, see appendix).

Fine-tuning with partial labels. Following common practices, we also fine-tune the pretrained models on ImageNet with 1% and 10% labeled data in Tab. 4. As seen, SSQL achieves the best performance in all cases. We also report the PTQ performance and our advantages become greater as the bit-width decreases. For instance, when fine-tuned using 10% labels under R-50, SSQL achieves 0.5% and 8.8% higher accuracy than SimSiam at FP and 4w4a, respectively.

Combining with QAT method. To further illustrate the effectiveness of SSQL, we combine different pretrained models with the state-of-the-art QAT method LSQ [9]. We initialize LSQ with ImageNet linear evaluated FP models



(a) 4w4a (b) 3w3a (c) 2w4a

Fig. 6: ImageNet results using LSQ [9]. See appendix for training details of LSQ.

Table 5: ImageNet transfer results on recognition benchmarks under R-50.

Datasets	Method	Linear evaluation					Fine-tuning				
		FP	8w8a	5w5a	4w4a	3w3a	FP	8w8a	5w5a	4w4a	3w3a
CIFAR-10	SimSiam	86.3	86.2	84.4	70.7	48.0	95.9	95.9	92.0	51.7	14.6
	SSQL (ours)	**89.3**	**89.2**	**89.1**	**87.1**	**71.9**	**96.3**	**96.3**	**95.1**	**89.2**	**69.3**
CIFAR-100	SimSiam	58.9	58.7	52.5	39.0	20.2	82.9	82.5	76.7	66.0	5.3
	SSQL (ours)	**68.7**	**68.6**	**68.8**	**66.4**	**49.7**	**83.3**	**83.3**	**82.0**	**74.7**	**39.3**
Flowers	SimSiam	78.7	82.5	81.9	66.6	49.3	94.0	93.8	83.8	57.8	16.2
	SSQL (ours)	**90.7**	**90.7**	**91.3**	**90.9**	**84.0**	**95.3**	**95.3**	**94.6**	**90.3**	**70.6**
Food-101	SimSiam	67.1	67.1	64.7	56.0	27.7	**86.2**	**86.2**	80.4	54.4	2.2
	SSQL (ours)	**72.6**	**72.5**	**71.5**	**68.4**	**51.6**	85.5	85.5	**84.5**	**70.4**	**11.2**
Pets	SimSiam	79.7	79.6	74.3	70.9	32.2	**87.5**	**87.4**	81.3	59.3	10.9
	SSQL (ours)	**83.6**	**83.9**	**83.3**	**82.3**	**73.8**	86.9	86.8	**85.9**	**84.6**	**73.6**
Dtd	SimSiam	69.9	69.7	69.1	63.4	46.6	73.4	73.6	70.5	60.4	8.8
	SSQL (ours)	**74.4**	**74.3**	**74.3**	**73.4**	**64.4**	**73.7**	**73.7**	**71.9**	**70.1**	**56.6**
Caltech-101	SimSiam	80.2	80.4	78.6	66.7	31.4	**86.9**	**86.6**	85.0	76.8	7.9
	SSQL (ours)	**86.9**	**87.2**	**85.2**	**83.8**	**65.9**	86.4	86.3	**85.5**	**82.9**	**59.7**

(i.e., FP column in Table 3). As seen from the learning curves in Fig. 6, our SSQL provides a better starting point. Take R-50 4w4a as an example, SSQL achieves 7% higher accuracy than SimSiam after the first epoch, while the initial accuracy of the FP model is about the same. Consequently, our SSQL achieves higher final accuracy and it shows that our pretrained model can serve as a better initialization when combined with QAT methods to boost performance.

Transferring to recognition benchmarks. We transfer the ImageNet learned representations of R-50 to downstream recognition tasks in Table 5. The results of R-18 and more training details are included in the appendix. As shown in Table 5, our method improves a lot on all recognition benchmarks, especially under linear evaluation. When comparing the fine-tuning results at FP, we can see that our SSQL achieves comparable results with SimSiam. When we further conduct PTQ, we can observe larger improvements as the bit-width decreases, which is consistent with the properties observed in upstream pretraining. Take R-50 on CIFAR-10 as an example, SSQL is slightly better than SimSiam at FP but the improvement expands to 37.5% at 4w4a and 54.7% at 3w3a. In conclusion, the quantization-friendly properties are also well-preserved by SSQL when we fine-tune the weights during transferring. This again confirms our motivation that quantization-friendly pretraining is both important and feasible.

Table 6: Object detection results on VOC2007 under R18-C4. The best results are in **boldface** and the second best results are underlined.

Method	FP			8w8a			6w6a			5w5a			4w4a		
	AP_{50}	AP	AP_{75}	AP_{50}	AP	AP_{75}	AP_{50}	AP	AP_{75}	AP_{50}	AP	AP_{75}	AP_{50}	AP	AP_{75}
random init.	58.9	32.1	30.5	58.7	31.9	30.2	58.4	31.6	30.2	57.0	30.4	28.9	42.4	20.8	17.0
IN supervised	**73.9**	44.6	46.5	**74.1**	44.2	46.2	73.0	43.4	44.5	68.9	39.4	39.3	33.1	16.7	14.0
BYOL	72.8	44.7	46.3	72.4	44.4	46.0	72.2	44.2	45.6	62.7	38.0	39.4	52.7	28.9	27.8
SimSiam	72.8	44.4	46.6	72.9	44.4	46.3	72.4	44.0	46.0	69.7	42.0	43.1	50.4	26.4	23.8
SimSiam-200ep	72.5	44.3	46.5	72.5	44.3	46.5	72.0	43.9	46.3	69.2	41.4	42.7	53.7	29.8	29.0
SSQL (ours)	73.4	44.7	46.8	73.5	**45.0**	46.8	**73.1**	44.5	46.4	**71.6**	42.8	44.4	**61.2**	34.1	33.4
SSQL-200ep (ours)	73.2	**45.0**	**47.3**	73.2	**45.0**	**47.0**	72.9	**44.8**	**46.8**	71.3	**43.3**	**45.0**	**61.2**	**35.1**	**35.0**

Table 7: Object detection/segmentation results on COCO2017 under R50-FPN.

Method	FP						6w6a					
	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}
IN supervised	38.2	56.0	42.0	34.8	56.0	37.2	37.6	58.3	41.4	34.3	55.2	36.8
SimSiam	**38.9**	**59.8**	**42.3**	**35.2**	**56.7**	**37.7**	38.1	58.7	41.5	34.5	55.7	36.8
BYOL	37.4	57.9	40.6	34.1	54.9	36.4	37.0	57.4	40.2	33.7	54.3	36.0
SSQL (ours)	38.7	59.2	**42.3**	**35.2**	56.2	**37.7**	**38.3**	**58.8**	**41.7**	**34.8**	**55.8**	**37.3**
	5w5a						4w4a					
IN supervised	35.2	55.5	38.4	31.9	52.3	34.0	23.4	38.6	24.6	21.4	36.3	22.1
SimSiam	34.3	54.0	36.7	30.9	50.6	32.6	19.9	33.6	20.6	18.1	31.3	18.3
BYOL	34.9	54.4	37.7	31.8	51.4	33.8	22.7	37.4	24.0	20.9	35.2	21.7
SSQL (ours)	**36.5**	**56.9**	**39.4**	**33.3**	**53.6**	**35.5**	**28.2**	**43.1**	**27.5**	**26.0**	**43.1**	**27.5**

Transferring to object detection. We investigate the downstream object detection performance on Pascal VOC07&12 [10] in Table 6 and COCO2017 [25] in Table 7. The detector is Faster R-CNN [33] with a backbone of R18-C4 [17] for VOC and Mask R-CNN [17] with R50-FPN [24] backbone for COCO, implemented in [39]. We follow the same settings in [5] and we evaluate the performance of post-training quantization models (i.e., the fine-tuning+PTQ pipeline).

As shown in Table 6, our SSQL performs better than SimSiam and BYOL on Pascal VOC at FP. Also, as we lower the bit-width, our SSQL is more significantly better than baseline counterparts: up to **+1.9** and **+7.5** AP_{50} over *the best results among other methods* at 5w5a and 4w4a, respectively. We can reach similar conclusions on COCO2017 from Table 7. Although our SSQL achieves slightly lower accuracy than SimSiam at FP on COCO, we achieve **+2.2** and **+8.3** AP^{bb} points higher at 5w5a and 4w4a, respectively. In conclusion, the results show that the quantization-friendly property of our pretrained model can be well-preserved even after fine-tuning on downstream detection tasks.

4.4 Ablation studies

We conduct ablation studies on CIFAR-10 in Table 8 and we keep the training settings the same as in Sec. 4.2. 'Q Pred' denotes whether to quantize the prediction branch and the same for 'Q Target'. 'Aux' denotes whether to add the auxiliary SimSiam loss. 'W/A Bit' represents the candidate bit-widths set for weight/activation. We can have the following conclusions from Table 8:

Table 8: Ablation studies on CIFAR-10 using ResNet-34.

ID	Q Pred	Q Target	Aux	W Bit	A Bit	Linear evaluation accuracy (%)					
						FP	6w6a	4w4a	3w3a	2w4a	Avg.
(a)	×	×	×	-	-	89.0	89.0	87.2	75.6	55.3	79.2
(b)	×	✓	×	4∼16	4∼16	87.6	87.5	85.8	70.4	58.5	78.0
(c)	✓	✓	×	4∼16	4∼16	90.5	90.4	88.9	79.2	73.7	84.5
(d)	✓	×	×	4∼16	4∼16	**91.0**	**91.0**	89.5	83.0	65.2	83.9
(e)	✓	×	×	6	6	90.0	89.9	87.9	69.1	62.1	79.8
(f)	✓	×	×	4	4	36.0	35.9	36.4	29.2	29.7	33.4
(g)	✓	✓	×	2∼8	4∼8	88.3	88.2	86.9	80.3	85.4	85.8
(h)	✓	×	×	2∼8	4∼8	89.6	89.5	88.2	82.9	81.5	86.3
(i)	✓	×	✓	2∼8	4∼8	90.9	90.8	**89.6**	**83.2**	**86.8**	**88.3**

- Quantizing the target branch only degenerates the performance. The row (b) is the worst among the first four rows, which indicates that only using the quantized output as the target makes training more difficult (learning noisy targets). In other words, it is essential to update the quantized branch with the gradients (both row (c) and (d) perform better than the baseline row (a)).
- Random selection of bit-widths for training is better than training with a single bit-width. We can observe that the row (d) surpasses the row (e) and (f) at all bit-widths, where the latter two are trained using a single bit-width. It shows that the random selection operation in our method is beneficial to improve performance, by providing stronger randomness and augmentations.
- Using a reasonable bit perturbation range further improves the performance at lower bit-widths. When comparing the row (d) and (h), we can observe a big boost at 2w4a (81.5 v.s. 65.2) at the expense of FP accuracy. When comparing the row (c) and (g), we can find that quantizing both branches at the same time results in a larger drop in FP accuracy.
- The row (i) achieves the best trade-off among all settings, which is also the default setting for all our experiments. When comparing the row (i) and (h), we can see that the addition of the auxiliary loss makes the full precision model produce better targets, thus improving the accuracies at all bit-widths.

5 Conclusion

In this paper, we proposed a method called SSQL for pretraining quantization-friendly models to facility flexible deployment in resource constrained applications. We provide theoretical analysis for the proposed approach, and experimental results on various benchmarks show that our method not only greatly improves the performance when quantized to lower bits, but also boosts the performance of full precision models. It has also been verified that our method is compatible with PTQ or QAT methods, and the quantization-friendly property can be well-preserved when transferring to downstream tasks. In the future, we will explore applications of SSQL to other architectures, notably Transformers. Also, we will explore fine-tuning methods that can better preserve the quantization-friendly property of our models.

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. Caron, M., Bojanowski, P., Joulin, A., Douze, M.: Deep clustering for unsupervised learning of visual features. In: The European Conference on Computer Vision, LNCS, vol. 11218, pp. 132–149. Springer (2018)
3. Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., Joulin, A.: Unsupervised learning of visual features by contrasting cluster assignments. In: Advances in neural information processing systems. pp. 9912–9924 (2020)
4. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: The International Conference on Machine Learning. pp. 1597–1607 (2020)
5. Chen, X., Fan, H., Girshick, R., He, K.: Improved baselines with momentum contrastive learning. arXiv preprint arXiv:2003.04297 (2020)
6. Chen, X., He, K.: Exploring simple siamese representation learning. In: The IEEE Conference on Computer Vision and Pattern Recognition. pp. 15750–15758 (2021)
7. Choi, J., Wang, Z., Venkataramani, S., Chuang, P.I.J., Srinivasan, V., Gopalakrishnan, K.: Pact: Parameterized clipping activation for quantized neural networks. In: The International Conference on Learning Representations. pp. 1–12 (2018)
8. Doersch, C., Gupta, A., Efros, A.A.: Unsupervised visual representations learning by context prediction. In: The IEEE International Conference on Computer Vision. pp. 1422–1430 (2015)
9. Esser, S.K., McKinstry, J.L., Bablani, D., Appuswamy, R., Modha, D.S.: Learned step size quantization. In: The International Conference on Learning Representations. pp. 1–12 (2020)
10. Everingham, M., Gool, L.V., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (VOC) challenge. International Journal of Computer Vision **88**(2), 303–338 (2010)
11. Fang, Z., Wang, J., Wang, L., Zhang, L., Yang, Y., Liu, Z.: SEED: Self-supervised distillation for visual representation. In: The International Conference on Learning Representations. pp. 1–12 (2021)
12. Gidaris, S., Singh, P., Komodakis, N.: Unsupervised representation learning by predicting image rotations. In: The International Conference on Learning Representations. pp. 1–14 (2015)
13. Grill, J.B., Strub, F., Altche, F., Tallec, C., H.Richemond, P., Buchatskaya, E., Doersch, C., Pires, B.A., Guo, Z.D., Azar, M.G., Piot, B., Kavukcuoglu, K., Munos, R., Valko, M.: Boostrap your own latent: A new approach to self-supervised learning. In: Advances in neural information processing systems. pp. 21271–21284 (2020)
14. Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In: The International Conference on Learning Representations. pp. 1–14 (2016)
15. Han, T., Li, D., Liu, J., Tian, L., Shan, Y.: Improving low-precision network quantization via bin regularization. In: The IEEE International Conference on Computer Vision. pp. 5261–5270 (2021)
16. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. In: The IEEE Conference on Computer Vision and Pattern Recognition. pp. 9729–9738 (2020)

17. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: The IEEE International Conference on Computer Vision. pp. 2961–2969 (2017)
18. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778 (2016)
19. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
20. Hubara, I., Nahshan, Y., Hanani, Y., Banner, R., Soudry, D.: Accurate post training quantization with small calibration sets. In: The International Conference on Machine Learning. pp. 4466–4475 (2021)
21. Jin, Q., Yang, L., Liao, Z.: AdaBits: Neural network quantization with adaptive bitwidths. In: The IEEE Conference on Computer Vision and Pattern Recognition. pp. 2146–2156 (2020)
22. Krizhevsky, A., Hinton, G.E.: Learning multiple layers of features from tiny images. Tech. rep., University of Toronto (2009)
23. Li, Y., Gong, R., Tan, X., Yang, Y., Hu, P., Zhang, Q., Yu, F., Wang, W., Gu, S.: Brecq: Pushing the limit of post-training quantization by block reconstruction. In: The International Conference on Learning Representations. pp. 1–16 (2021)
24. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: The IEEE Conference on Computer Vision and Pattern Recognition. pp. 2177–2125 (2017)
25. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common objects in context. In: The European Conference on Computer Vision, LNCS, vol. 8693, pp. 740–755. Springer (2014)
26. Liu, X., Zhang, F., Hou, Z., Wang, Z., Mian, L., Zhang, J., Tang, J.: Self-supervised learning: Generative or contrastive. arXiv preprint arXiv:2006.08218 (2020)
27. Luo, J.H., Wu, J., Lin, W.: Thinet: A filter level pruning method for deep neural network compression. In: The IEEE International Conference on Computer Vision. pp. 5058–5066 (2017)
28. van der Maaten, L., Hinton, G.: Visualizing data using t-sne. Journal of Machine Learning Research **9**(86), 2579–2605 (2008)
29. Nagel, M., Amjad, R.A., van Baalen, M., Louizos, C., Blankevoort, T.: Up or down? adaptive rounding for post-training quantization. In: The International Conference on Machine Learning. pp. 7197–7206 (2020)
30. Nagel, M., van Baalen, M., Blankevoort, T., Welling, M.: Data-free quantization through weight equalization and bias correction. In: The IEEE International Conference on Computer Vision. pp. 1325–1334 (2019)
31. Noroozi, M., Favaro, P.: Unsupervised learning of visual representations by solving jigsaw puzzles. In: The European Conference on Computer Vision, LNCS, vol. 9910, pp. 69–84. Springer (2016)
32. van den Oord, A., Li, Y., Vinyals, O.: Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748 (2018)
33. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. pp. 91–99 (2015)
34. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet large scale visual recognition challenge. International Journal of Computer Vision **115**(3), 211–252 (2015)

35. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: MobileNetV2: Inverted residuals and linear bottlenecks. In: The IEEE Conference on Computer Vision and Pattern Recognition. pp. 4510–4520 (2018)
36. Shen, M., Liang, F., Gong, R., Li, Y., Li, C., Lin, C., Yu, F., Yan, J., Ouyang, W.: Once quantization-aware training: High performance extremely low-bit architecture search. In: The IEEE International Conference on Computer Vision. pp. 5340–5349 (2021)
37. Shen, Z., Liu, Z., Qin, J., Huang, L., Cheng, K.T., Savvides, M.: S2-bnn: Bridging the gap between self-supervised real and 1-bit neural networks via guided distribution calibration. In: The IEEE Conference on Computer Vision and Pattern Recognition. pp. 2165–2173 (2021)
38. Sheng, T., Feng, C., Zhuo, S., Zhang, X., Shen, L., Aleksic, M.: A quantization-friendly separable convolution for mobilenets. arXiv preprint arXiv:1803.08607 (2018)
39. Wu, Y., Kirillov, A., Massa, F., Lo, W.Y., Girshick, R.: Detectron2. `https://github.com/facebookresearch/detectron2` (2019)
40. Yu, H., Wen, T., Cheng, G., Sun, J., Han, Q., Shi, J.: Low-bit quantization needs good distribution. In: The IEEE Conference on Computer Vision and Pattern Recognition Workshops (2020)
41. Zhang, R., Isola, P., Efros, A.A.: Colorful image colorization. In: The European Conference on Computer Vision, LNCS, vol. 9907, pp. 649–666. Springer (2016)
42. Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: An extremely efficient convolutional neural network for mobile. In: The IEEE Conference on Computer Vision and Pattern Recognition. pp. 6848–6856 (2018)
43. Zhou, S., Wu, Y., Ni, Z., Zhou, X., Wen, H., Zou, Y.: Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. arXiv preprint arXiv:1606.06160 (2016)

A Implementation details

Datasets. The statistics of the classification benchmarks used in our paper are shown in Table 9.

Table 9: Statistics of the classification benchmarks used in the paper.

Datasets	# Category	# Training	# Testing
CIFAR-10	10	50,000	10,000
CIFAR-100	100	50,000	10,000
Flowers	102	2,040	6,149
Food-101	101	75,750	25,250
Pets	37	3,680	3,669
DTD	47	3,760	1,880
Caltech-101	101	2020	1010

Training details for SSL methods. Training details for MoCov2, SimCLR, BYOL, SimSiam and our SSQL on CIFAR-10/CIFAR-100 are shown in Table 10.

Table 10: Training details for MoCov2, SimCLR, BYOL, SimSiam and our SSQL on CIFAR datasets in Table 1 and Table 2. τ denotes the temperature parameter, k denotes the size of memory bank in MoCov2, and m denotes the momentum in MoCov2 and BYOL.

Method	Settings									
	bs	lr	wd	epochs	optimizer	lr schedule	τ	k	m	dim
SimSiam	512	0.05	5e-4	400	SGD	cosine	-	-	-	2048
MoCov2	256	0.03	1e-4	400	SGD	cosine	0.2	4096	0.999	2048
SimCLR	512	0.5	1e-4	400	SGD	cosine	0.5	-	-	2048
BYOL	512	0.5	5e-4	400	SGD	cosine	-	-	0.99	2048
SSQL (ours)	256	0.05	1e-4	400	SGD	cosine	-	-	-	2048

Training details for linear evaluation and fine-tuning. For ImageNet linear evaluation, we follow the same settings in [6]. For linear evaluation on other datasets, we train for 100 epochs with lr initialized to 30.0, which is divided by 10 at the 60th and 80th epoch. For fine-tuning, we train for 50 epochs with lr initialized to 0.001, which is divided by 10 at the 30th and 40th epoch. The weight decay is 0 for linear evaluation and 1e-4 for fine-tuning.

Training details for LSQ. We initialize LSQ with linear evaluated full precision models on ImageNet. Then, we train 50 epochs using SGD. We set the batch size to 256, weight decay to 1e-5, and base lr to 0.001. We divide the learning rate by 10 at the 30th epoch.

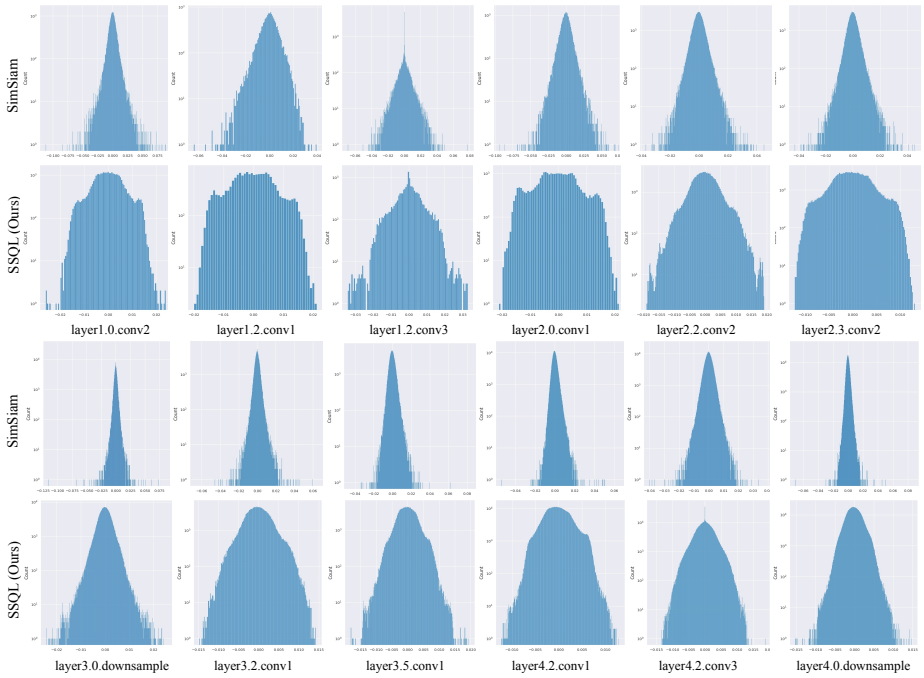


Fig. 7: Visualization of weights distribution for ResNet-50 fine-tuned on Dtd.

B Experimental results

We present more experimental results here in this section. We present more visualizations of weight distribution in Sec. B.1, more transfer results in Sec. B.2. Moreover, we investigate whether our method can be applied in more SSL frameworks in Sec. B.3 and emerging new Transformer-like architectures in Sec. B.4.

B.1 Weight distribution

In Fig. 7, we visualize the weights distribution of different models (fine-tuned from ImageNet pretrained models on Dtd). In this case, the backbone weights have been updated and we can still observe the quantization-friendly property of our model. As seen, our model has more uniform distribution, smaller ranges and much fewer outliers.

B.2 Transfer results

Classification benchmarks. We present the ImageNet transfer results on classification benchmarks under ResNet-18 in Table 11 and the corresponding plots are shown in Fig. 8. We can see that our SSQL not only greatly improves the performance when quantized to low bit-widths, but also improves the performance of full precision models in some cases.

Table 11: ImageNet transfer results on classification benchmarks under R-18.

Datasets	Method	Linear evaluation					Fine-tuning				
		FP	8w8a	5w5a	4w4a	3w3a	FP	8w8a	5w5a	4w4a	3w3a
CIFAR-10	SimSiam	66.6	66.3	65.5	59.3	35.5	94.5	94.5	92.9	83.6	38.5
	SSQL (ours)	**81.0**	**80.9**	**80.9**	**79.5**	**69.6**	**94.8**	**94.8**	**94.5**	**92.4**	**74.4**
CIFAR-100	SimSiam	33.2	33.2	32.3	25.3	10.9	77.0	77.0	73.7	56.0	9.5
	SSQL (ours)	**55.8**	**55.9**	**55.8**	**53.5**	**45.5**	**79.0**	**78.8**	**77.6**	**73.4**	**44.8**
Flowers	SimSiam	53.7	54.1	53.8	44.2	12.9	84.2	84.0	80.2	72.3	18.5
	SSQL (ours)	**87.4**	**87.1**	**86.8**	**86.1**	**79.9**	**92.0**	**92.0**	**91.6**	**90.9**	**77.0**
Food-101	SimSiam	36.4	35.0	35.3	32.4	13.6	**81.3**	**81.3**	78.0	63.8	4.3
	SSQL (ours)	**60.7**	**60.9**	**59.9**	**57.6**	**48.7**	80.9	80.9	**79.9**	**73.0**	**19.6**
Pets	SimSiam	48.3	48.7	47.7	42.4	6.2	80.2	80.2	77.9	54.1	8.8
	SSQL (ours)	**77.3**	**77.3**	**77.0**	**75.1**	**66.3**	**81.9**	**81.8**	**81.4**	**79.9**	**57.4**
Dtd	SimSiam	54.2	54.0	53.2	50.9	31.3	66.2	66.3	66.1	51.9	16.1
	SSQL (ours)	**67.7**	**67.2**	**67.2**	**67.0**	**62.1**	**69.0**	**69.0**	**68.8**	**67.4**	**46.6**
Caltech-101	SimSiam	53.9	54.3	53.4	47.6	20.6	75.6	75.6	73.5	63.1	6.4
	SSQL (ours)	**80.2**	**80.1**	**80.1**	**79.0**	**70.4**	**81.6**	**81.4**	**82.0**	**80.9**	**62.3**

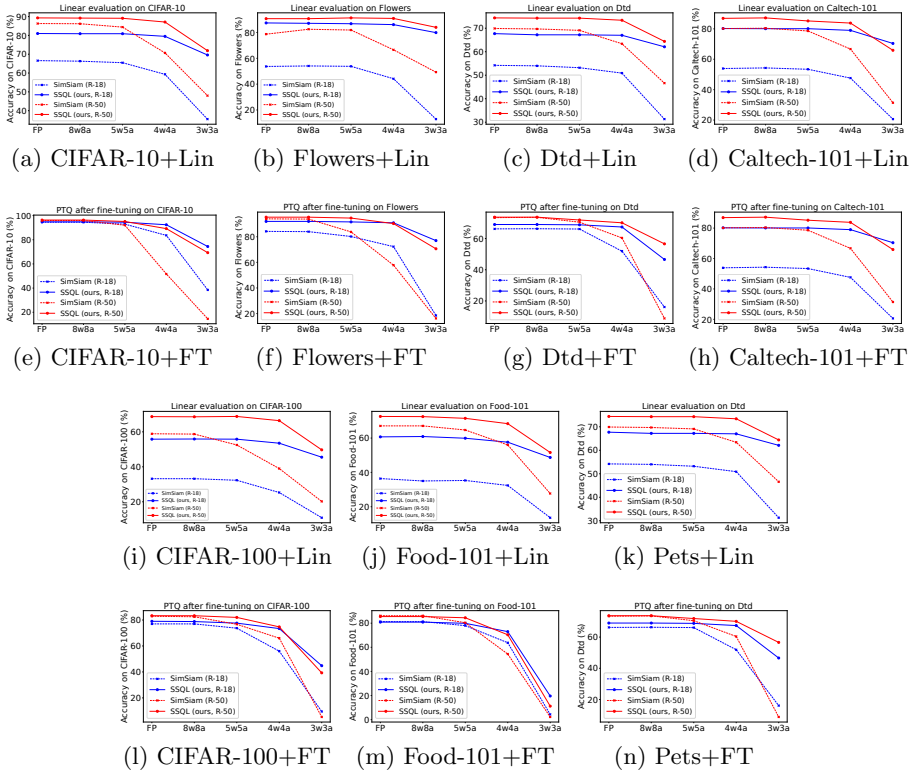
Combining with LSQ on COCO. We initialize LSQ with COCO fine-tuned models. Notice that we only quantize the backbone here (without quantizing ROI heads). As shown in Fig. 9, we can observe that our SSQL provides a better starting point for low bit QAT training on COCO. Take 2w4a (AP_{bb}) as an example, SSQL achieves 6.3 points higher than SimSiam (25.4 v.s. 19.1) after the first 1k iteration, while the initial accuracy of the FP model is about the same (38.2 v.s. 38.4). Consequently, our SSQL achieves higher final accuracy (36.4 v.s. 35.8) and it shows that our pretrained model can serve as a better initialization when combined with QAT methods to boost performance.

B.3 Applications in other SSL methods

In this subsection, we demonstrate that our method SSQL can also work on other SSL frameworks. We experiment on MoCov2 and BYOL on CIFAR-10 under R-18 in Table 12. Our SSQL has consistent improvements, too.

Table 12: Linear evaluation results on CIFAR-10.

Backbone	Method	FP	6w6a	5w5a	4w4a	3w3a	2w4a
ResNet-18	BYOL	89.3	89.4	89.3	88.0	75.1	63.3
	BYOL+SSQL	**90.8**	**90.7**	**90.6**	**89.8**	**85.0**	**85.7**
	MoCov2	88.9	88.4	88.2	86.8	72.2	50.7
	MoCov2+SSQL	**89.6**	**89.6**	**89.5**	**88.5**	**83.4**	**85.2**



(a) CIFAR-10+Lin (b) Flowers+Lin (c) Dtd+Lin (d) Caltech-101+Lin

(e) CIFAR-10+FT (f) Flowers+FT (g) Dtd+FT (h) Caltech-101+FT

(i) CIFAR-100+Lin (j) Food-101+Lin (k) Pets+Lin

(l) CIFAR-100+FT (m) Food-101+FT (n) Pets+FT

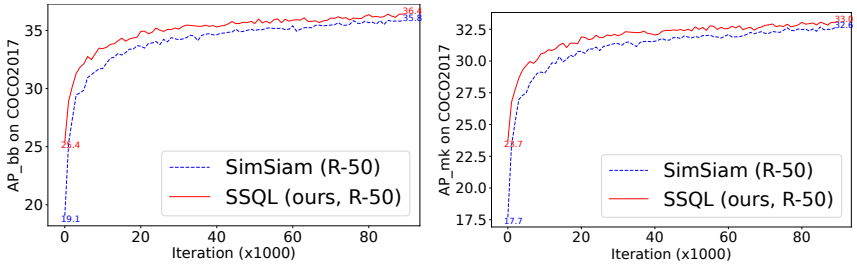
Fig. 8: Transfer recognition results. The first/third and second/fourth row shows the results under linear evaluation and fine-tuning ('FT'), respectively. Best viewed in color.

B.4 Applications in vision transformers

In this subsection, we investigate whether the SSQL can be applied on Transformer-like backbones to achieve effectiveness. We supplement the results on CIFAR-10 using ViT-Small by adapting SSQL to MoCov3 (we use official code and conduct linear evaluation) in Table 13. Our SSQL does have potentials on Transformer-like backbones.

C More analysis of the synergy

In this section, we give more analysis as a supplement to Sec. 3.3 in the paper. We give empirical support for the weakly correlated assumption in Sec. C.1 and analyze the synergy from another perspective in Sec. C.2.



(a) AP_{bb} (b) AP_{mask}

Fig. 9: COCO fine-tuning results using LSQ (2w4a), initialized with the fine-tuned FP models in Table 7.

Table 13: Linear evaluation results on CIFAR-10 under ViT-Small.

Backbone	Method	FP	8w8a	6w6a	5w5a	4w4a
ViT-Small	MoCov3	88.0	87.6	87.2	82.2	82.0
	MoCov3+SSQL	**88.6**	**88.6**	**88.3**	**88.2**	**86.9**

C.1 Support for the weakly correlated assumption

We plot the curve and histogram of correlation between the quantization and contrastive errors during training (10k iterations≈100 epochs) in Fig. 10. Notice that the value range is $[-1, 1]$ and in most cases the correlation is around 0 (i.e., uncorrelated), and it does not exceed ± 0.1. Experimental results verify that our assumption is a reasonable one.

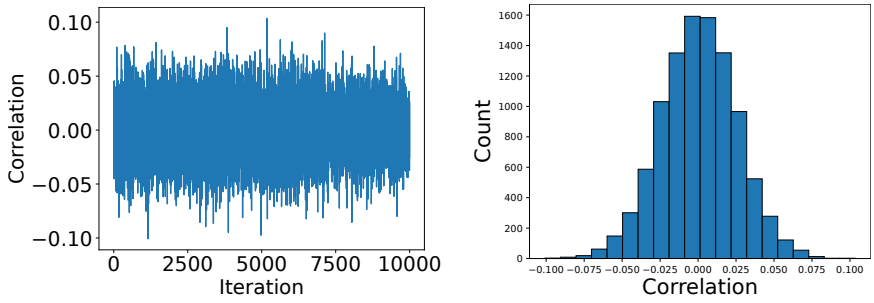


Fig. 10: The correlation between the quantization and contrastive errors during training on CIFAR-10 using ResNet-18. Left: Curve of the correlation. Right: Histogram distribution of the correlation.

C.2 Analysis of the synergy from another perspective

For simplicity, we assume f is a two-layer perceptron:

$$\boldsymbol{z}_1 = \boldsymbol{w}_2\sigma(\boldsymbol{w}_1 \cdot \boldsymbol{x}_1)\,, \tag{16}$$

where $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ are the corresponding weights and $\sigma(\cdot)$ is the activation function.

We consider only the first term in each loss function (i.e., the similarity between $\boldsymbol{p}_1$ and $\boldsymbol{z}_2$) without loss of generality. Suppose we quantize $\boldsymbol{w}_2$ and $\boldsymbol{w}_2^q = \boldsymbol{w}_2 + \Delta w$ and the analysis is the same for other weights or activations.

$$\begin{aligned} L_{SSQL} =& - p_1^q \cdot z_2 = -h(z_1^q) \cdot z_2 \\ =& - h(z_1 + \Delta z) \cdot z_2 \\ \approx& - \Big(h(z_1) + h'(z_1)\Delta z\Big) \cdot z_2 \\ =& - p_1 \cdot z_2 - h'(z_1)\Delta z \cdot z_2\,, \end{aligned}$$

where $\Delta z = \Delta w\sigma(w_1 \cdot x_1)$. By introducing quantization noise, we can see that its effect can be thought of as adding a random perturbation to the points before evaluating their similarity as usual. This provides an explanation on why our method could lead to better results.

We now investigate the backward pass for L_{SSQL}:

$$\begin{aligned} \frac{\partial L_{SSQL}}{\partial z_1} =& \frac{\partial L}{\partial p_1^q} \cdot \frac{\partial p_1^q}{\partial z_1} \\ =& - z_2 \cdot \frac{\partial h(z_1 + \Delta z)}{\partial z_1} \\ \approx& - z_2(h'(z_1) + h''(z_1)\Delta z) \end{aligned} \tag{17}$$

$$\frac{\partial L_{SSQL}}{\partial w_2} = \frac{\partial L}{\partial z_1} \cdot \frac{\partial z_1}{\partial z_1^q} \cdot \frac{\partial z_1^q}{\partial w_2} = \frac{\partial L}{\partial z_1} \cdot \sigma(w_1 \cdot x_1) \tag{18}$$

$$\frac{\partial L_{SSQL}}{\partial w_1} = \frac{\partial L}{\partial z_1} \cdot \frac{\partial z_1}{\partial z_1^q} \cdot \frac{\partial z_1^q}{\partial w_1} = \frac{\partial L}{\partial z_1} \cdot (w_2 + \Delta w)\sigma'(w_1 \cdot x_1) \cdot x_1 \tag{19}$$

The backward pass for $L_{SimSiam}$ is obvious from the above derivations:

$$\frac{\partial L_{SimSiam}}{\partial z_1} = -z_2 h'(z_1) \tag{20}$$

$$\frac{\partial L_{SimSiam}}{\partial w_1} = -\frac{\partial L_{SimSiam}}{\partial z_1} \cdot w_2\sigma'(w_1 \cdot x_1) \cdot x_1 \tag{21}$$

When comparing Equation (17) with Equation (20), we can find an extra term $-z_2 h''(z_1)\Delta z$ in the gradients and we think this second-order term can help model learn better.