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.

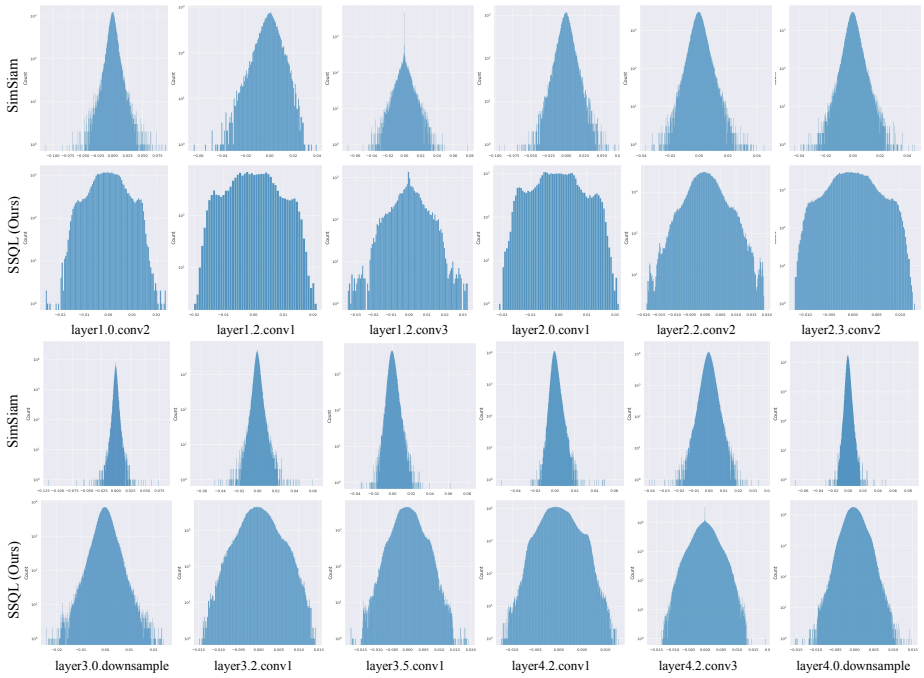


Fig. 7: Visualization of weights distribution for ResNet-50 fine-tuned on Dtd.

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**

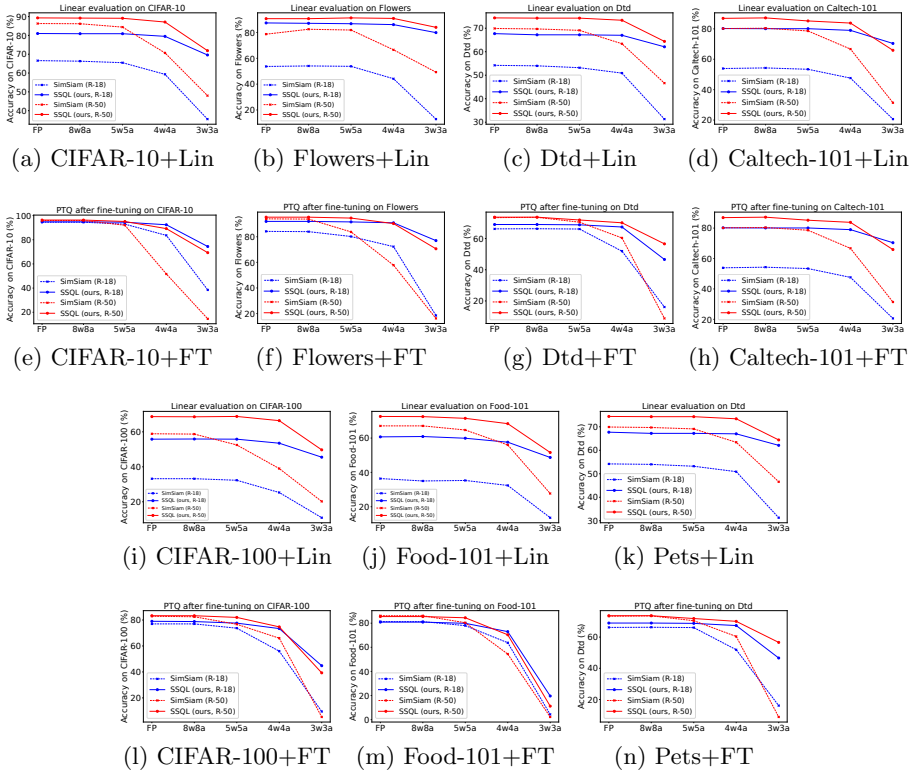
Combining with LSQ on COCO. We initialize LSQ with COCO fine-tuned models. Notice that we only quantize the backbone here (without quantizing ROI heads). As shown in Fig. 9, we can observe that our SSQL provides a better starting point for low bit QAT training on COCO. Take 2w4a (AP_{bb}) as an example, 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

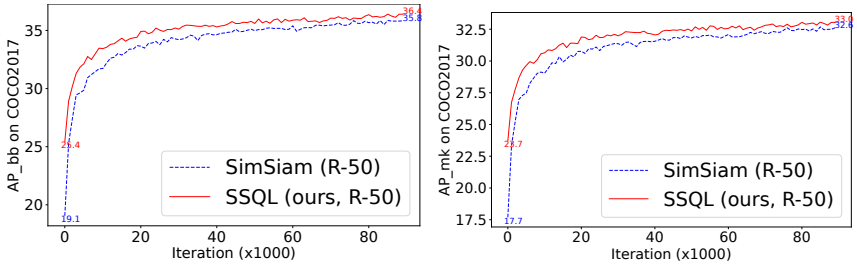
Fig. 8: Transfer recognition results. The first/third and second/fourth row shows the results under linear evaluation and fine-tuning ('FT'), respectively. Best viewed in color.

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.

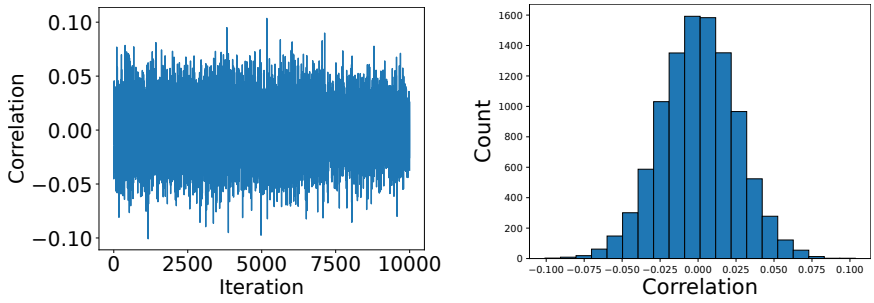


Fig. 10: The correlation between the quantization and contrastive errors during training on CIFAR-10 using ResNet-18. Left: Curve of the correlation. Right: Histogram distribution of the correlation.

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.