Appendix

Anonymous Authors

1 Related works

1.1 Network sparsity and sparse training

Overparameterization is a common problem in Deep Neural Networks, meaning many redundant parameters exist in models. In machine learning, overparameterization usually means overfitting and a colossal waste of resources. Therefore, we need model sparsification methods to explore the redundancy of model parameters and compress the model by reducing these redundancy parameters. Network pruning methods, including non-structured and structured pruning, are the most common ways to compress models.

Further, on the theoretical basis of pruning and sparsifying a pre-trained network, Frankle and Carbin [1] propose the Lottery Ticket Hypothesis (LTH), which enables to train a sparse network from scratch. The Lottery Ticket Hypothesis states that any randomly initialized dense network contains sub-networks with the following properties. When independently trained, the initialized subnetwork can achieve similar accuracy to the original network. On the basis of LTH, various sparse training methods have been proposed.

1.2 Representation similarity measurements

The representational similarity is an important concept in machine learning. General methods (including Euclidean distance, Manhattan distance, Cosine distance, etc.) compute statistics of two feature representations or utilize the geometric properties to calculate representation similarity.

However, when we need to measure the feature similarity between layers of neural networks, the above measurements are not robust and are sensitive to disturbance due to the randomness of neural networks. Frontier researchers propose two categories of methods to measure feature similarity between layers, including Canonical Correlation Analysis (CCA) [2, 3, 6] and Centered Kernel Alignment (CKA) [4]. Briefly, CCA method maps the two feature matrices to be measured onto a set of bases and then maximizes the correlation.

1.3 Information bottleneck theory

The information bottleneck (IB) theory proposed by Tishby *et al.* [8] is an extension of the rate distortion theory of source compression. This theory shows a trade-off between preserving relevant label information and obtaining efficient compression. Tishby *et al.* [9] further research the relationship between information bottleneck theory and deep learning. They interpret the goal of deep learning as an information-theoretic trade-off between compression and prediction.

2 Additional analysis

For convenience, we abbreviate $\mathbf{CKA}_{Linear}$ as $\mathbf{CKA}_L$ in the appendix.

2.1 Proof of two lemmas

Lemma 1. *Minimizing the distance between X^TY and zero matrix is equivalent to minimizing the mutual information $I(X;Y)$ between representation X and Y.*

Lemma 2. *Minimizing $\mathbf{CKA}_L(X_i, X_j)$ is equivalent to minimizing $I(X_i; X_j)$.*

We prove the two Lemmas [10] as follows.

Proof. According to the definition of mutual information, $I(X;Y) = H(X) + H(Y) - H(X,Y)$. And according to the definition of multivariate Gaussian distribution, $H(X) = \frac{1}{2}ln((2\pi e)^D|\Sigma_X|)$. We assume that X and Y obey the multivariate Gaussian distribution, then we have $(X,Y) \sim N(0, \Sigma_{(X,Y)})$, in which

$$\Sigma_{(X,Y)} = \begin{pmatrix} \Sigma_X & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_Y \end{pmatrix} \tag{1}$$

The assumption that X and Y obey the multivariate Gaussian distribution with zero mean is reasonable, for there exist batch normalization layers in neural networks. Therefore, the mutual information between random variables X and Y following a Gaussian distribution is formulated as,

$$I(X;Y) = ln|\Sigma_X| + ln|\Sigma_Y| - ln|\Sigma_{(X,Y)}| \tag{2}$$

According to Everitt inequality, $|\Sigma_{(X,Y)}| \leq |\Sigma_X||\Sigma_Y|$, this inequality holds if and only if $\Sigma_{YX} = \Sigma_{XY^T} = X^TY$ is a zero matrix. Besides, the definition of linear HSIC is as follows,

$$\text{nHSIC}_{linear}(X,Y) = \frac{||Y^TX||_F^2}{||X^TX||_F||Y^TY||_F} \tag{3}$$

And according to the definition of the Frobenius norm, we have,

$$||Y^TX||_F^2 = ||X^TY||_F^2 \tag{4}$$

Combining Eq. (2), Eq. (3), Eq. (4), and Everitt inequality, we have, $I(X;Y) \geq 0$, and this inequality holds if and only if X^TY is a zero matrix. Further, we have: When minimizing Eq. (3), we are also minimizing the distance between $X^TY = X^Tf(X)$ and the zero matrix, *i.e.* **Lemma1**. (As X and Y are feature representations produced by different layers of the same network, we can represent Y as $Y = f(X)$, in which f is the sub-network between X and Y. As a sub-network of the neural network, f is a nonlinear function.) In other words, while minimizing $nHSIC_{linear}$, we also minimize the mutual information between the two Gaussian distributions, *i.e.* **Lemma2**.

2.2 Proof of Theorem 1

Theorem 1. *The ϵ-sparsity and the sparsity of neural networks are approximately equivalent.*

Proof. For convenience, we consider a neural network N consisting of a full-connected layer. We mark the sparsity of N as s. For a given input X, the network N outputs $Y = WX + b$, in which W and b are the weight matrix and bias, respectively.

We mark the minimum non-zero absolute value of weight matrix W as $min(W)$. We take an ϵ which meet the conditions of $\epsilon \to 0$ and $\epsilon << min(W)$. Obviously, as long as ϵ approaches 0, both conditions above are satisfied. Then we replace 0 values of the weight matrix W with values smaller than ϵ, and name this weight matrix as W'. We mark the neural network with weight matrix W' as N'. Because we have $\epsilon << min(W)$, the ϵ-sparsity of network N' is s. The two networks are identical except that N has the sparsity of s while N' has the ϵ-sparsity of s.

For a given input X, we consider the outputs of the two networks N and N'. The output of N is $Y = WX + b$, and the output of N' is $Y' = W'X + b$. We take the difference of Y and Y' as follows:

$$||Y - Y'|| = ||(WX + b) - (W'X + b)|| = ||(W - W')X|| \tag{5}$$

According to the definition of W', each element of the $W - W'$ matrix approaches 0. So we have $||W - W'|| \to 0$, then we have $||Y - Y'|| = ||(W - W')X|| \to 0$, which means the outputs of N and N' are approximately the same. Because the two networks are identical except for their sparsity, we come to the conclusion that ϵ-sparsity and sparsity of neural networks are approximately equivalent in practice.

2.3 Proof of Theorem 2

Theorem 2. *$\mathcal{L}_{\mathcal{C}}$ is continuous and optimizable.*

Proof. First we prove that $\mathcal{L}_{\mathcal{C}}$ is continuous. Because $\mathcal{L}_{\mathcal{C}}$ is the simple summation of $\mathbf{CKA}_L(X_i, X_j)$, we only need to prove $\mathbf{CKA}_L(X_i, X_j)$ is continuous for arbitrary element of matrix X_i and X_j. Selecting an arbitrary element $X_{ik'l'}$ of matrix X_i, we rewrite $\mathbf{CKA}_L(X_i, X_j)$ as

$$\mathbf{CKA}_L(X_i, X_j) = \frac{\sum_{k=1}^{n}\sum_{l=1}^{n}(X_i^T X_j)_{kl}^2}{K\sqrt{\sum_{k=1}^{n}\sum_{l=1}^{n}(X_i^T X_i)_{kl}^2}}, \tag{6}$$

in which $K = \sqrt{\sum_{k=1}^{n}\sum_{l=1}^{n}(X_j^T X_j)_{kl}^2}$ is a constant. Further, we have

$$(X_i^T X_j)_{kl} = \sum_{t=1}^{n}\sum_{u=1}^{n} X_{itk} X_{jul} \tag{7}$$

$$(X_i^T X_i)_{kl} = \sum_{t=1}^{n}\sum_{u=1}^{n} X_{itk} X_{iul}. \tag{8}$$

Substituting Eq. (7) into Eq. (6), we have

$$\mathbf{CKA}_L(X_i, X_j) = \frac{\sum_{k=1}^{n}\sum_{l=1}^{n}(\sum_{t=1}^{n}\sum_{u=1}^{n} X_{itk}X_{jul})^2}{K\sqrt{\sum_{k=1}^{n}\sum_{l=1}^{n}(\sum_{t=1}^{n}\sum_{u=1}^{n} X_{itk}X_{iul})^2}}. \tag{9}$$

Obviously, $\sum_{k=1}^{n}\sum_{l=1}^{n}(\sum_{t=1}^{n}\sum_{u=1}^{n} X_{itk}X_{jul})^2$ and $\sum_{k=1}^{n}\sum_{l=1}^{n}(\sum_{t=1}^{n}\sum_{u=1}^{n} X_{itk}X_{iul})^2$ are polynomials of $X_{ik'l'}$. We write them as $p_1(X_{ik'l'})$ and $p_2(X_{ik'l'})$, respectively. Then we rewrite $\mathbf{CKA}_L(X_i, X_j)$ as

$$\mathbf{CKA}_L(X_i, X_j) = \frac{p_1(X_{ik'l'})}{K\sqrt{p_2(X_{ik'l'})}}. \tag{10}$$

From Eq. (10), it's obvious that $\mathbf{CKA}_L(X_i, X_j)$ is derivable.(It's derivative is $\frac{1}{K}p_1'(X_{ik'l'})p_2(X_{ik'l'})^{-\frac{1}{2}} + \frac{-1}{2K}p_1(X_{ik'l'})p_2(X_{ik'l'})^{-\frac{3}{2}}p_2'(X_{ik'l'})$) Because derivable functions must be continuous, we prove that $\mathbf{CKA}_L(X_i, X_j)$ is continuous. Therefore, $\mathcal{L}_\mathcal{C}$ is continuous. Further, because $\mathcal{L}_\mathcal{C}$ has derivative at domain of function, we could carry out local optimization (by gradient descent) of $\mathcal{L}_\mathcal{C}$ to approach extremum values. That is to say, $\mathcal{L}_\mathcal{C}$ is optimizable in machine learning.

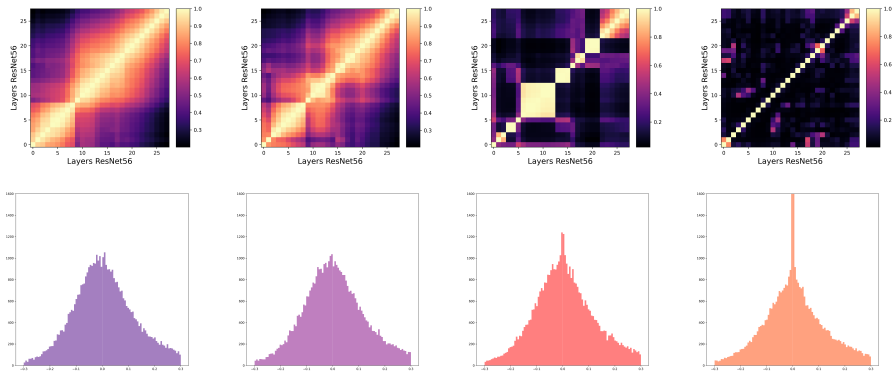
2.4 Proof of Theorem 3

Theorem 3. *Minimizing $\mathcal{L}_\mathcal{C}$ minimizes the mutual information $R = I(X;\hat{X})$ between intermediate representation $\hat{X}$ and input representation X.*

Proof. According to Theorem 2, CKA-SR is optimizable. In consideration of clear expression, we take a one-stage network or a single stage of multi-stage networks for example.

$$\begin{aligned}
&\min \mathcal{L}_\mathcal{C} \Leftrightarrow \min \beta \cdot \sum_{s=1}^{1}\sum_{i=0}^{N_s}\sum_{j=0}^{N_s} \mathbf{CKA}_L(X_i, X_j) \\
&\Leftrightarrow \min 2 \times \sum_{i=0}^{N_s}\sum_{j=i+1}^{N_s} \mathbf{CKA}_L(X_i, X_j) + \sum_{i=0}^{N_s} \mathbf{CKA}_L(X_i, X_i) \\
&\Leftrightarrow \min 2 \times \sum_{i=0}^{N_s}\sum_{j=i+1}^{N_s} \mathbf{CKA}_L(X_i, X_j) + (N_s + 1) \\
&\Leftrightarrow \min \sum_{i=0}^{N_s}\sum_{j=i+1}^{N_s} \mathbf{CKA}_L(X_i, X_j)
\end{aligned} \tag{11}$$

We define several successive layers (from the i^{th} to the j^{th} layers) of the original network as a layer-sub-network. Then the $R = I(X;\hat{X})$ of the layer-sub-network can be expressed as $R_{ij} = I(X_i; X_j)$, in which $j > i$, X_i is the input of the layer-sub-network, and X_j is the map of X_i through the layer-sub-network. According to Lemma 2, minimizing $\mathbf{CKA}_L(X_i, X_j)$ is equivalent to minimizing $I(X_i; X_j)$, *i.e.*, R_{ij}. Substituting R_{ij} into the minimization objective, we have:



(a) Baseline (b) $\beta = 0.0005$ (c) $\beta = 0.002$ (d) $\beta = 0.005$

Fig. 1. Visualizations of CKA similarity and corresponding parameter distribution (ResNet56)

$$\min \sum_{i=0}^{N_s} \sum_{j=i+1}^{N_s} \mathbf{CKA}_L(X_i, X_j) \Leftrightarrow \min \sum_{i=0}^{N_s} \sum_{j=i+1}^{N_s} R_{ij} \tag{12}$$

That is, minimizing the CKA-SR is equivalent to minimizing the sum of mutual information R of all layer-sub-networks of the network, thus minimizing the mutual information between intermediate representation and input representation.

2.5 Potential negative societal impacts and limitations

To the best of our knowledge, there exists almost no potential negative societal impact in our method, which is a plug-and-play regularization in sparse training and pruning methods. However, considering the expenses to select the hyperparameter and calculate loss function, it's necessary to think about reducing the energy consumption and protecting the environment while conducting experiments.

3 Additional experimental results

3.1 Computation settings

We conduct our experiments on a server with 8 Tesla V100 GPUs and 40 CPUs. Specifically, we use 1 Tesla V100 GPU for CIFAR experiments and we use 8 Tesla V100 GPUs for ImageNet experiments.

Algorithm 1 Dynamic token sparsification with CKA-SR

Initialize: ViT model pretrained with CKA-SR f_W, dataset $\mathcal{D}$, pruning ratio r

1: Calculate the feature map of each patch, and name it as a token.
2: Update each token through multiple transformer layers.
3: **while** Numbers of remaining tokens N_r/Numbers of all tokens $N_t > r$ **do**
4: Use the prediction module to calculate a mask for pruning, in which the keeping probability of the i^{th} token equals $\pi'_{i,1}$ in Eq. (13).
5: Prune the tokens with the mask.
6: Update the remaining tokens through transformer layers.
7: **end while**
8: Conduct classification with the remaining few tokens.

3.2 Dynamic token sparsification with CKA-SR

To validate our CKA-SR method on vision transformers, we conduct dynamic token sparsification [7] with our CKA-SR. Specifically, we calculate the CKA similarity between tokens produced by our CKA-SR model and introduce this similarity into the decision process of dynamic token sparsification. We rewrite the keeping probability of the i^{th} token $\pi_{i,1}$ as

$$\pi'_{i,1} = \pi_{i,1} - \sum_{j \neq i} \mathbf{CKA_L}(X_i, X_j) \tag{13}$$

in which j is all the other tokens except i. The whole algorithm is shown in Algorithm 1. And the results are shown in Table 1.

Table 1. The accuracy (%) of DeiT-Tiny using CKA-SR for dynamic token sparsification on ImageNet

Model	Hyperparameter ρ	Params	FLOPs	Top-1 Accuracy
DeiT-Tiny	–	5.7M	1.3G	72.20
Dynamic-DeiT-Tiny	0.7	5.9M	0.9G	71.11
+CKA-SR	0.7	5.9M	0.9G	**71.35**
Dynamic-DeiT-Tiny	0.9	5.9M	1.2G	72.31
+CKA-SR	0.9	5.9M	1.2G	**72.52**

3.3 Further ablation studies

Ablation study of samples and batches As our CKA-SR is proposed to explicitly reduce the interlayer similarity of network parameters instead of feature maps themselves, we could utilize several samples of each batch (*generally 8 samples when the batch size is 128 or 256*) to compute CKA-SR. Besides, to reduce the expenses, we could calculate CKA-SR once out of several (*generally 5 or 10*) batches. We conduct the ablation study of samples and batches with

Random Sparse Training [5] method on CIFAR-10 dataset, and the sparsity of the ResNet20 models is 0.95. The results are shown in Table 2.

Table 2. Ablation study of samples and batches (*Batch-m* means we calculate CKA-SR once out of m batches, and *Sample-n* means we utilize n samples of each batch to calculate CKA-SR)

β	Baseline	1e-05	2e-05	5e-05	8e-05	1e-04	2e-04	5e-04	8e-04	1e-03	2e-03	5e-03	1e-02
Original CKA-SR	84.16	84.69	84.38	84.42	84.45	84.11	84.40	84.39	**85.03**	84.82	84.08	83.86	84.03
Sample-16 and Batch-5	84.16	84.06	84.52	84.44	**84.90**	84.59	84.59	84.42	84.61	84.40	84.61	84.71	84.17
Sample-8 and Batch-5	84.16	84.19	84.51	84.44	84.45	**84.97**	83.88	84.68	84.41	84.56	84.52	84.00	84.41
Sample-4 and Batch-5	84.16	84.71	84.35	84.62	84.20	83.85	84.30	84.87	**84.92**	84.62	84.67	84.27	76.08
Sample-8 and Batch-10	84.16	84.28	84.27	**84.62**	84.14	84.27	84.06	84.25	84.17	84.19	84.59	84.43	84.08

To conclude, we can just utilize several samples of each batch and calculate CKA-SR only once out of several batches. By using these implementations above, the performance is slightly worse than full CKA-SR (our original method which calculates CKA-SR for each sample in each batch), but it's still better than the baseline method. At the same time, by using these implementations, the expenses of calculating CKA-SR are greatly reduced.

3.4 Visualizations of CKA similarity and parameter distribution

As an extension of Figure 1 of our paper, we show the visualizations of CKA similarity and corresponding parameter distributions of our CKA-SR sparsity models, and compare them with baseline models. We conduct the visualization with Random Sparse Training [5] method on CIFAR-10 dataset, and the sparsity of ResNet56 and ResNet20 models is 0.95. The results are shown in Figure 1 and Figure 2 (Note that in Figure 2(c), the Y-axis is up to 1750, and in Figure 2(d), the Y-axis is up to 2800, which means the parameters are extremely concentrated around 0).

We can conclude that with the increase of hyperparameter β, the interlayer feature similarity decreases and the network sparsity increases.

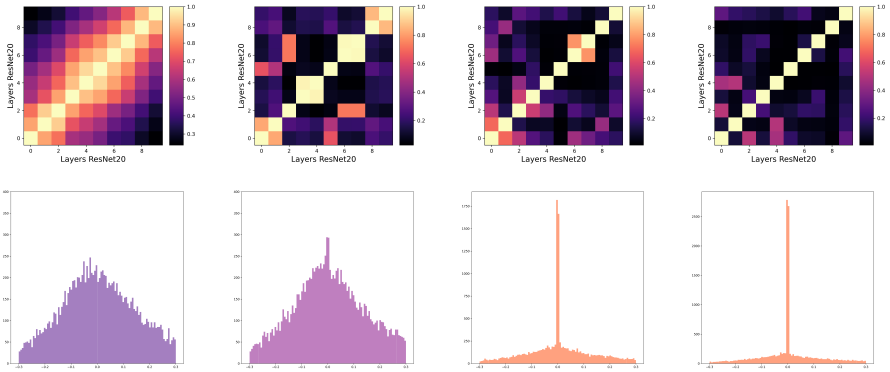
3.5 Comparison with other similarity measurements

To compare the performance of CKA and CCA in sparse training, we introduce CCA into sparse training. We find that CCA also increases the performance of sparse training, but the increment is smaller than CKA-SR. Besides, CCA is less robust than CKA, which means sparse training with CCA is harder to converge than CKA-SR.

We conduct the experiments with Random Sparse Training [5] on the CIFAR-10 dataset, and the sparsity is 0.95. The results are shown in Table 3.

Table 3. Comparison of CKA-SR and CCA

Settings	Baseline	CCA	CKA-SR
Top1-Acc	84.16	84.58	85.03



(a) Baseline (b) $\beta = 0.01$ (c) $\beta = 0.05$ (d) $\beta = 0.08$

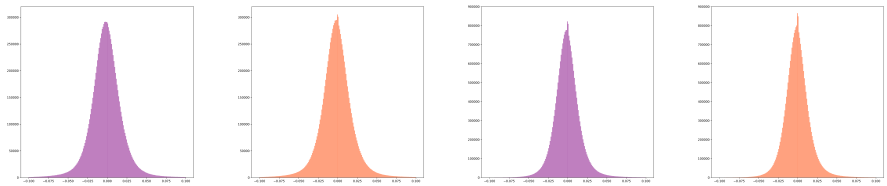
Fig. 2. Visualizations of CKA similarity and corresponding parameter distribution (ResNet20)

3.6 Models pre-trained with CKA-SR

As an additional experimental result, we show the performance increment of models pre-trained with CKA-SR in Table 4. Besides, we show the parameter distribution of these models in Figure 3 to prove that our CKA-SR increases the sparsity of the pre-trained models.

Table 4. The accuracy (%) when plugging CKA-SR to pre-training of ResNet18 and ResNet50 models on CIFAR100 and ImageNet

Backbone	Methods	Dataset	
		CIFAR100	ImageNet
ResNet18	Baseline	75.61	69.62
	+CKA-SR	**76.32**	**69.90**
ResNet50	Baseline	77.39	76.15
	+CKA-SR	**78.94**	**76.39**



(a) Baseline(b) CKA-SR(c) Baseline(d) CKA-SR
ResNet18 ResNet18 ResNet50 ResNet50

Fig. 3. Parameter distributions of baseline and CKA-SR models (ResNet18 and ResNet50 models pre-trained on ImageNet)

It is concluded from Figure 3 and Table 4 that our CKA-SR increases both the sparsity (by making parameters more concentrated around 0) and the classification accuracy of ResNet18 and ResNet50 models trained on ImageNet.

4 AugCKA-SR

4.1 Definition

AugCKA-SR is an improved version of CKA-SR. We introduce the equation of **Aug**CKA-SR as follows, in which we use features with different shapes to calculate CKA-SR.

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_{\mathcal{E}} + \mathcal{L}_{\mathcal{C}} \\ &= \mathcal{L}_{\mathcal{E}} + \beta \cdot \frac{2}{N(N-1)} \cdot \sum_{i=0}^{N-1} \sum_{j=i+1}^{N} \mathbf{CKA}_{Linear}(X_i, X_j) \end{aligned} \tag{14}$$

In Eq. (14), N is the total number of layers. In this Augmented CKA-SR, we calculate the similarity between each pair of different layers and utilize the average similarity in the loss function.

Then we provide the implementation of **Aug**CKA-SR below.

```
def centering(K):
    n = K.size(0)
    unit = torch.ones([n, n]).cuda()
    I = torch.eye(n).cuda()
    H = I - unit / n
    return torch.mm(torch.mm(H, K), H)


def linear_HSIC(X, Y):
    L_X = torch.mm(X, X.T)
    L_Y = torch.mm(Y, Y.T)
```

```
    return torch.sum(centering(L_X) * centering(L_Y))


def linear_CKA(X, Y):
    hsic = linear_HSIC(X, Y)
    var1 = torch.sqrt(linear_HSIC(X, X))
    var2 = torch.sqrt(linear_HSIC(Y, Y))
    return hsic / (var1 * var2)


class CKALoss(nn.CrossEntropyLoss):
    def __init__(self, ):
        super(CKALoss, self).__init__()

    def forward(self, blocks):
        blocklist = []
        losssum = torch.zeros(1).cuda()
        for i in range(len(blocks)):
            for j in range(len(blocks[i])):
                blocklist.append(blocks[i][j])
        # Calculate CKA similarity between each two layers
        for i in range(len(blocklist)-1):
            for j in range(i+1, len(blocklist)):
                X = blocklist[i].flatten(1).float()
                Y = blocklist[j].flatten(1).float()
                losssum += linear_CKA(X, Y)
        loss = 2*losssum/len(blocklist)/(len(blocklist)-1)
        return loss
```

4.2 Experimental results

We conduct experiments on our **Aug**CKA-SR. We plug our **Aug**CKA-SR into the training process of Random Sparse Training method. We adopt ResNet20 as the backbone, and apply sparsity ratios from 0.70 to 0.998 for fair comparisons. The results are in Table 5.

Table 5. The accuracy (%) when plugging CKA-SR and **Aug**CKA-SR to Random Sparse Training(RST) method on CIFAR-100 from scratch.

Backbone	Method	Sparsity					
		0.70	*0.85*	*0.90*	*0.95*	*0.98*	*0.998*
ResNet20	Random[5]	65.42	60.37	56.96	47.27	33.74	2.95
	+CKA-SR	65.60	60.86	**57.25**	48.26	34.44	**3.32**
	+**Aug**CKA-SR	**65.85**	**60.89**	57.02	**48.39**	**34.60**	3.04

Table 5 shows that **Aug**CKA-SR is effective in Random Sparse Training and outperforms original CKA-SR in most sparsity levels, which fully demonstrates the effectiveness of the improvement proposed by us.

References

1. Frankle, J., Carbin, M.: The lottery ticket hypothesis: Finding sparse, trainable neural networks. arXiv preprint arXiv:1803.03635 (2018)
2. Hardoon, D.R., Szedmak, S., Shawe-Taylor, J.: Canonical correlation analysis: An overview with application to learning methods. Neural computation **16**(12), 2639–2664 (2004)
3. Hotelling, H.: Relations between two sets of variates. In: Breakthroughs in statistics, pp. 162–190. Springer (1992)
4. Kornblith, S., Norouzi, M., Lee, H., Hinton, G.: Similarity of neural network representations revisited. In: International Conference on Machine Learning. pp. 3519–3529. PMLR (2019)
5. Liu, S., Chen, T., Chen, X., Shen, L., Mocanu, D.C., Wang, Z., Pechenizkiy, M.: The unreasonable effectiveness of random pruning: Return of the most naive baseline for sparse training. In: International Conference on Learning Representations (2021)
6. Ramsay, J., ten Berge, J., Styan, G.: Matrix correlation. Psychometrika **49**(3), 403–423 (1984)
7. Rao, Y., Zhao, W., Liu, B., Lu, J., Zhou, J., Hsieh, C.J.: Dynamicvit: Efficient vision transformers with dynamic token sparsification. Advances in neural information processing systems **34**, 13937–13949 (2021)
8. Tishby, N., Pereira, F.C., Bialek, W.: The information bottleneck method. arXiv preprint physics/0004057 (2000)
9. Tishby, N., Zaslavsky, N.: Deep learning and the information bottleneck principle. In: 2015 ieee information theory workshop (itw). pp. 1–5. IEEE (2015)
10. Zheng, X., Ma, Y., Xi, T., Zhang, G., Ding, E., Li, Y., Chen, J., Tian, Y., Ji, R.: An information theory-inspired strategy for automatic network pruning. arXiv preprint arXiv:2108.08532 (2021)