Supplementary Document

Aditya Singh[1] and Haohan Wang[2]

[1] aditya.91.singh@gmail.com
[2] School of Information Sciences, University of Illinois Urbana-Champaign, Champaign, IL, USA
haohanw@illinois.edu

Algorithm 1 PyTorch based implementation

```
# f_s, f_t: student and teacher networks
# β: scale for the loss
# Ω: nearest neighbour set
# k: neighbours to sample

for x in loader: # load a minibatch x with b samples
    x = [x, Ω^k]
    x = augment(x) # random augmentation

    A_s = f_s(x) # student output bXd
    with torch.nograd():
        A_t = f_t(x) # teacher output bXd

    l_co = dot(A_s, A_t) #feature term
    l_ss = dot(A_s.T, A_t.T) #space term
    loss = β*(l_co + l_ss) #feature and space terms

    loss.backward() # back-propagate
    update(f_s) # SGD

def dot(s, t):
    s = torch.norm(s, dim=-1)
    t = torch.norm(t, dim=-1)
    return - torch.mean((s*t).sum(dim=-1))
```

1 Adapting Supervised Methods

As many distillation approaches operate on the features of the students and teachers, they can be applied to an unsupervised setting. However, to the best of our knowledge, such a study has not yet been performed. In this experiment, we make slight adjustments to the baselines implemented by CRD [10]. We remove the final-classification layer of the teacher and the student to simulate an unsupervised setting. Moreover, we set the contributions due to ground-truth and soft-labels to 0. For a fair comparison, for the baselines, we report the best top-1 achieved from hyper-parameter sweep by scaling loss $\{0.5, 1, 2, 4, 8, 16, 32\}\times$.

In Tab. 1, we report the results of baselines and CoSS. We observe that baseline methods don't readily work in an unsupervised setting. In the future, it will be interesting to delve deeper into understanding why such a wide gap in performance exists for distillation methods when applied to an unsupervised setting.

Table 1: Unsupervised distillation performance of adapted supervised distillation methods on CIFAR-100.

Arch	ATT [14]	SP [11]	VID [1]	RKD [8]	PKT [9]	Factor [7]	NST [5]	CRD [10]	SRRL [13]	DIST [4]	CoSS
Res20/Res56	49.00	63.14	68.61	56.28	61.58	46.63	25.62	65.03	69.31	67.13	**71.11**
Vgg8/Vgg13	56.22	71.62	73.65	44.50	71.81	39.71	42.78	69.73	72.85	73.40	**74.58**
Res8x4/Res32x4	43.11	62.51	69.60	32.33	64.53	36.64	41.27	65.78	69.19	67.67	**73.90**

Table 2: Importance of different loss components.

Models	L_{co}	L_{ss}	L_{coss}
Resnet8x4/32x4	72.05	73.53	**73.90**
ResNet20/56	70.58	70.42	**71.11**

2 Ablation on CIFAR-100

2.1 Importance of loss components

Following Tian et al. [10], we utilise the teacher models which were trained using supervision. We remove the final classification layer of the teacher to simulate an unsupervised setting. In Tab. 2, we report the results of distillation on CIFAR-100 dataset. We can observe that combining feature and space similarity yields the best performing student.

2.2 Contribution of λ

Table 3: Contribution of λ to distillation.

Student/Teacher	$\lambda = 0$	$\lambda = 0.25$	$\lambda = 0.5$	$\lambda = 1$
ResNet8x4/32x4	72.05	73.44	73.90	**73.91**
ResNet20/56	70.58	**71. 34**	71.11	70.68

For our main experiments, we used $\lambda = 1.0$. Here, we evaluate the distillation performance with different weights assigned to the space similarity component. From the reported results in Tab. 3, we note that while combining space similarity and feature similarity yields the best-performing student, the optimal results are achieved with varying degrees of space similarity contribution. Different architectures generally demonstrate varied behaviors in response to distillation. It is not yet known why some architectures perform better than others, even when they have similar capacities. We believe that space similarity similarly impacts different architectures to a varying degree.

3 SEED + Space Similarity

We train the SEED objective with our proposed space similarity objective ($k = 0$). We observe that the SEED + SS student achieves a top-1 of 58.30% compared to the baseline SEED model's 57.60. As we did not employ the nearest neighbour sampling, we expect the performance to further increase upon doing so.

4 Batch Normalisation For Space Similarity

Batch Normalisation (BN) was proposed to reduce the covariate shift which occurs during the mapping of inputs from one layer to another [6]. Since its introduction, BN has found place in numerous deep learning architectures[2, 3, 12]. Here, we show how one can directly aim the distillation process to match student and network's embeddings and subsequently compare its performance with our formulation.

BN operates on a batch of input data, $X \in R^{b \times d}$ where batch size is b and d is the feature dimension. It first performs standardisation $\hat{X}_{:,i} = \frac{X_{:,i} - \mu_i}{\sigma_i}$ where, : denotes all the entries in the batch dimension and μ_i, σ_i are the mean and variances respectively for the i^{th} feature dimension. The normalized values are then scaled by trainable parameters γ_i and β_i as:

$$Z_{:,i} = \gamma \hat{X}_{:,i} + \beta \tag{1}$$

here, γ_i and β_i can be interpreted as affine transformations which operate independently for different spatial dimensions. We can utilise it to map the standardised student embeddings to the teacher's unnormalized embedding space. The corresponding loss can be defined as follows:

$$\mathcal{L}_{coss} = \frac{1}{b} \sum_{i=0}^{i<b} \mathcal{D}(Z_i^s, X_i^t)$$

where, X_i^t is the teacher's embedding for the i^{th} sample and Z_i^s corresponds to the BatchNormalized student's embeddings. In table 4, we report the results using this approach on CIFAR-100 distillation task. We utilised mean-squared-error for the metric D.

Table 4: CIFAR-100 unsupervised distillation with BN.

Methods	VGG13	Resnet32x/8x	WRN-40/16
BN	74.01	72.22	73.42
CoSS($k = 0$)	74.58	73.90	74.65

Bibliography

[1] Sungsoo Ahn, Shell Xu Hu, Andreas Damianou, Neil D Lawrence, and Zhenwen Dai. Variational information distillation for knowledge transfer. In *CVPR*, pages 9163–9171, 2019.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *ECCV*, 2016.

[4] Tao Huang, Shan You, Fei Wang, Chen Qian, and Chang Xu. Knowledge distillation from a stronger teacher. *Advances in Neural Information Processing Systems*, 35:33716–33727, 2022.

[5] Zehao Huang and Naiyan Wang. Like what you like: Knowledge distill via neuron selectivity transfer. *arXiv preprint arXiv:1707.01219*, 2017.

[6] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[7] Jangho Kim, SeongUk Park, and Nojun Kwak. Paraphrasing complex network: Network compression via factor transfer. In *NeurIPS*, pages 2760–2769, 2018.

[8] Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. In *CVPR*, pages 3967–3976, 2019.

[9] Nikolaos Passalis and Anastasios Tefas. Learning deep representations with probabilistic knowledge transfer. In *ECCV*, pages 268–284, 2018.

[10] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. *ICLR*, 2020.

[11] Frederick Tung and Greg Mori. Similarity-preserving knowledge distillation. In *ICCV*, pages 1365–1374, 2019.

[12] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *arXiv preprint arXiv:1611.05431*, 2016.

[13] Jing Yang, Brais Martinez, Adrian Bulat, and Georgios Tzimiropoulos. Knowledge distillation via softmax regression representation learning. In *ICLR2021*, 2021.

[14] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *ICLR*, 2017.