Distilling Knowledge from Self-Supervised Teacher by Embedding Graph Alignment

Yuchen Ma*[1]
y.yuchenma@gmail.com
Yanbei Chen*[1]
yanbeic@gmail.com
Zeynep Akata[1,2,3]
zeynep.akata@uni-tuebingen.de

[1] University of Tübingen, Germany
[2] MPI for Informatics, Germany
[3] MPI for Intelligent Systems, Germany

In this document, we first provide an algorithmic overview in Section 1. We provide ablation study of our model in Section 2. We give more quantitative and qualitative results in Section 3 and Section 4 respectively. Finally, we give more implementation details in Section 5.

1 Algorithmic Overview

As mentioned in the paper in Section 3.4, we adopt two different training strategies for optimizing the teacher and the student networks: (1) mutual learning [10], which optimizes the teacher with the target objective $\mathcal{L}_{ce}$ and the student with the full objective *simultaneously*; (2) sequential learning [3], which optimizes the teacher and student networks *sequentially* by first training the teacher with the task objective $\mathcal{L}_{ce}$ and then training the student with the full objective. We present the algorithmic overviews for our Embedding Graph Alignment (EGA) model using these two training strategies in Algorithm 1 and Algorithm 2 respectively.

2 Ablation Study

Model ablation. We analyze the different loss terms in our EGA model. As Table 1 shows, we evaluate our model in comparison to three baseline models: (1) baseline (which is the student network trained without distillation); (2) EGA w/o $\mathcal{L}_{node}$, which removes the loss term $\mathcal{L}_{node}$ for aligning node matrix to an identity matrix; (3) EGA w/o $\mathcal{L}_{edge}$, which removes the loss term $\mathcal{L}_{edge}$ for aligning the edge matrices between the teacher graph and the student graph. From Table 1, we have the following observations. First, our full model formulation EGA performs much better than the other EGA variants with a single loss term, offering the best accuracy of 76.65%, 84.15%, 60.61% on CIFAR100, STL-10 and TinyImageNet. Second, "EGA w/o $\mathcal{L}_{node}$" and "EGA w/o $\mathcal{L}_{edge}$" both outperform the baseline, which suggests

*Equal contribution. The work was done when Yuchen Ma was enrolled in the MSc program at the University of Heidelberg.

Algorithm 1 Distilling self-supervised knowledge by Embedding Graph Alignment using a *simultaneous* training strategy

Input: Training image data $\mathcal{I}$ with labels Y, Training iterations τ; Self-supervised teacher networks $\mathcal{N}_t$.

Output: Distilled student networks $\mathcal{N}_s$.

Initialisation: Randomly initialise $\mathcal{N}_s$, teacher's new added layers $\mathcal{M}_t$, two optimizers.

for iteration t in $[1:\tau]$:

Feed a batch of B images $\{I_1, I_2, ..., I_B\}$ to $\mathcal{N}_t$, $\mathcal{N}_s$.

Get teacher feature embeddings X_t.

Get student feature embeddings X_s.

Derive teacher edge matrix $E_t = E(X_t, X_t)$.

Derive student edge matrix $E_s = E(X_s, X_s)$.

Derive node matrix $N_{st} = E(X_t, X_s)$.

Compute edge matching loss $\mathcal{L}_{edge} = \|E_t - E_s\|_2$.

Compute node matching loss $\mathcal{L}_{node} = \|N_{st} - \mathcal{I}\|_2$.

Compute the EGA loss $\mathcal{L}_{EGA} = \mathcal{L}_{node} + \lambda \mathcal{L}_{edge}$.

Compute teacher final loss $\mathcal{L}_t = \mathcal{L}_{ce_t}$.

Compute student final loss $\mathcal{L}_s = \mathcal{L}_{ce_s} + \lambda_{EGA}\mathcal{L}_{EGA}$.

Backpropagation on the teacher's new layers $\mathcal{M}_t$.

Backpropagation on the student network $\mathcal{N}_s$.

end for

return $\mathcal{N}_t$

that both loss terms are effective for distilling the knowledge from the teacher to the student network. This ablation shows that our EGA works effectively for distilling self-supervised knowledge to improve recognition.

Method	CIFAR 100	STL-10	TinyImageNet
baseline	72.38	80.63	57.54
EGA w/o $\mathcal{L}_{node}$	75.18	83.78	59.52
EGA w/o $\mathcal{L}_{edge}$	75.21	82.35	59.82
EGA	**76.65**	**84.15**	**60.61**

Table 1: Ablation study on different loss components. Teacher: self-supervised ViT-B/32 from [7]. Student: resnet8x4.

Hyperparameter analysis. We analyze the hyperparameters of the edge matching loss $\mathcal{L}_{edge}$ and node matching loss $\mathcal{L}_{node}$ in Table 2. We explore the effect of these two loss terms by varying their weights λ_{edge} and λ_{node} wrt to the supervised loss term (i.e. $\mathcal{L} = \mathcal{L}_{ce} + \lambda_{edge}\mathcal{L}_{edge}$ and $\mathcal{L} = \mathcal{L}_{ce} + \lambda_{node}\mathcal{L}_{node}$). We test this by setting the weight of one loss term to be zero and changing the weight of another loss term (either λ_{edge} or λ_{node}). In the first two rows of Table 2, we can see that the model performance first increases and then decreases when increasing λ_{edge}. In the last two rows of the table, We can also see that similar trends of model performance as the left figure when increasing λ_{node}. The best performing values for λ_{edge}, λ_{node} are 0.5, 1.5, which we use to trade off $\mathcal{L}_{node}$, $\mathcal{L}_{edge}$ and guide our hyperpameter

Algorithm 2 Distilling self-supervised knowledge by Embedding Graph Alignment using a *sequential* training strategy

Input: Training image data $\mathcal{I}$ with labels Y, Training iterations τ; Self-supervised teacher networks $\mathcal{N}_t$. Pre-trained teacher's new added layers $\mathcal{M}_t$.

Output: Distilled student networks $\mathcal{N}_s$.

Initialisation: Randomly initialise $\mathcal{N}_s$, student optimizer.

for iteration t in $[1:\tau]$:

Feed a batch of B images $\{I_1, I_2, ..., I_B\}$ to $\mathcal{N}_t$, $\mathcal{N}_s$.

Get teacher feature embeddings X_t.

Get student feature embeddings X_s.

Derive teacher edge matrix $E_t = E(X_t, X_t)$.

Derive student edge matrix $E_s = E(X_s, X_s)$.

Derive node matrix $N_{st} = E(X_t, X_s)$.

Compute edge matching loss $\mathcal{L}_{edge} = \|E_t - E_s\|_2$.

Compute node matching loss $\mathcal{L}_{node} = \|N_{st} - \mathcal{I}\|_2$.

Compute the EGA loss $\mathcal{L}_{EGA} = \mathcal{L}_{node} + \lambda \mathcal{L}_{edge}$.

Compute student final loss $\mathcal{L}_s = \mathcal{L}_{ce_s} + \lambda_{EGA} \mathcal{L}_{EGA}$.

Backpropagation on the student network $\mathcal{N}_s$.

end for

return $\mathcal{N}_t$

setting $\lambda = \frac{\lambda_{edge}}{\lambda_{node}} \approx 0.3$.

Node weight	1.2	1.4	1.5	1.6	1.8	2.0
Accuracy	75.11	75.56	**75.85**	75.17	75.51	74.91
Edge weight	0.2	0.4	0.5	0.6	0.8	1.0
Accuracy	74.47	75.32	**75.56**	74.99	74.72	74.27

Table 2: Effect of hyperparameters (node weight, edge weight) on CIFAR10.

3 More Quantitative Results

Analysis of the graph size. We analyze the different graph size of the EGA loss in Figure 1. As the graph size (i.e. the number of node) is equal to the batch size, we change the batch size from 16, 32, 64, 128, 256 with a double increase to obtain different graph size. Figure 1 shows the trends of model performance as the graph size change. As can be seen, EGA obtains the best model performance with a graph size of 64, and the worse performance with a graph size of 256. When using a graph size of 16, 32, 64, and 128, the model performs similarly well, which suggests that EGA could work robustly under various choices of batch size.

Numerical results. Due to space limit, we show only the top performing methods in Table 2 and Table 5 in the paper. Here, we present the numerical results of all the comparative

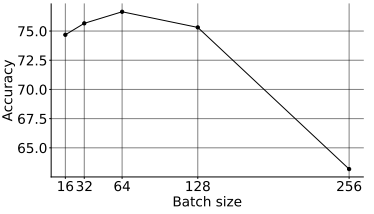

Fig. 1: Model performance (top 1 accuracy, %) under varying graph size (i.e. which is quantified by the batch size).

Method	Same student different teacher			Same teacher different student		
	ViT-B/32	ViT-B/16	RN101	Resnet8x4	ShuffleNetV1	VGG13
KD[3]	71.82	9.6	65.16	71.82	73.52	75.40
FitNet[3]	74.83	67.85	73.69	74.83	nan	76.28
PKT [6]	73.48	72.71	72.66	73.48	74.92	75.35
RKD[5]	73.36	72.43	73.92	73.36	72.62	73.26
NCE [1]	74.42	73.89	74.04	74.42	74.36	76.61
CRD[2]	75.51	73.38	74.85	75.51	74.87	77.41
CCL[1]	75.98	39.56	74.22	75.98	76.05	77.54
EGA	**76.11**	**74.02**	**75.22**	**76.11**	**76.74**	**77.76**

Table 3: Top 1 accuracy (%) of student networks on CIFAR100. In column 1-3, we use the same student (resnet8x4) and vary the self-supervised teacher using ViT-B/32, ViT-B/16, RN101 from [7]. In column 4-6, we use the same teacher (ViT-B/32) and vary the student using resnet8x4, shuffleNetV1, VGG13. The teacher and student are trained ***sequentially*** [10]. Best results are **bold**.

Method	CIFAR 100	STL-10	TinyImageNet
KD [3]	71.82	82.51	53.88
FitNet	74.83	nan	nan
PKT [6]	73.48	nan	59.72
RKD [5]	73.36	82.67	58.32
NCE [1]	74.42	80.07	60.00
CRD [2]	75.51	78.76	60.82
CCL[1]	75.98	80.41	61.24
EGA	**76.11**	**83.01**	**61.85**

Table 4: Top 1 accuracy (%) of student networks on CIFAR100, STL10, TinyImageNet. Teacher: self-supervised ViT-32 [7]. Student: resnet8x4. The teacher and student are trained ***sequentially*** [3]. Best results are highlighted in **bold**.

methods using the same evaluation setup in Table 2 and Table 5, as detailed in Table 3 and Table 4 respectively. As can be seen, our model obtains the best overall performance compared to the other compared methods. These evidences are consistent with our analysis in the paper.

4 More Qualitative Results

Visualizing embeddings with t-SNE. In order to examine the distilled structural semantic relationships learned from the teacher network by different loss terms, we visualize the feature embeddings of the EGA and RDK models from different classes. We compare RKD and our EGA, as both of these models focus on distilling structural relationship information from the teacher networks. Figure 2 and Figure 3 show the feature distributions from RKD

and EGA models, which depicts the seven fine-grained classes from two hyperclasses – *orchids*, *poppies*, *roses* belong to the hyperclass *flowers*, while *bed*, *chair*, *table*, and *wardrobe* belong to the hyperclass *furniture*. As can be seen in two figures, our model is capable of capturing the inter-class and hyper-class structural relations much better than RKD. From Figure 2, We can see that the cluster of *bed* is mixed with other clusters, especially the cluster of *table*. While in Figure 3, the cluster of *bed* has higher compactness, and *table* has clearer decision boundary wrt the clusters of other classes than RKD's. Moreover, the hyperclasses *furniture* and *flowers* are more separated in our model EGA than RKD. Overall, these visualizations suggest that the EGA model learns the structural semantics and captures the instance-instance relations better than RKD.

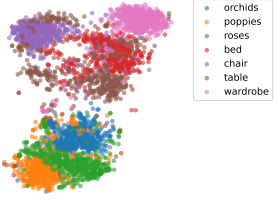

Fig. 2: t-SNE visualization [9] of the RKD feature embeddings from different classes on CIFAR100.

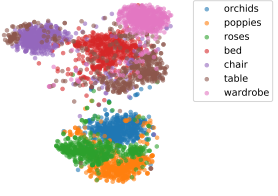

Fig. 3: t-SNE visualization [9] of the EGA feature embeddings from different classes on CIFAR100.

5 Implementation Details

Data augmentation. Each input image is transformed into two ways for teacher and student model. For student networks, The image augmentation pipeline consists of the following transformations: random cropping, random horizontal flipping, and normalization. As for teacher, the pipeline consists of random cropping, resizing to 224 × 224, random horizontal flipping and normalization. If the teacher model is a self-supervised pre-trained model which has its own preprocessing parameter, e.g. CLIP [7], both our teacher and our student will follow the same CLIP image preprocessing normalization parameters. The normalization parameters for evaluation is always consistent with training.

Training Details. The student networks are trained by the combination of a cross-entropy classification objective and a knowledge distillation objective. For the weight balance factor of these two loss terms, we directly use the optimal value from the original paper or adopted from CRD [9]. For the *simultaneous* training mode, the teacher's new added layers are trained by SGD with an initial learning rate of 0.01. For student network, we follow the setting from CRD [9]. That is, if the student is ShuffleNetV1, we use an initial learning rate of 0.01, while 0.05 for other models.

Supervised teacher vs self-supervised teacher networks. As mentioned in our paper, we evaluate multiple self-supervised teacher networks including ViT-B/32, ViT-B/16, RN101 from CLIP [7], which are downloaded from the CLIP GitHub repository. We only fine-tune the last two layers of the teacher networks on the target datasets. Thus, a self-supervised teacher network would capture both self-supervised knowledge and supervised knowledge from a target dataset. As for the supervised teacher networks, we use the pre-trained teacher networks which are pre-trained on ImageNet [2] and fine-tuned on the CIFAR100. Hence, a supervised teacher network would contain the supervised knowledge from the ImageNet and CIFAR100, which requires more annotations efforts compared to a self-supervised teacher network. For instance, in the paper, the self-supervised RN101 teacher network in Table 2 obtains an accuracy of 67.76% while the supervised RN101 teacher network in Table 3 obtains an accuracy of 73.58%. As for the downstream task on CIFAR100, the student with a self-supervised RN101 teacher obtains an accuracy of 75.22% (in Table 2) while the student with a supervised RN101 teacher obtains an accuracy of 75.77% (in Table 3). Overall, these results suggest that a self-supervised teacher network may not start with a better model initialization compared to a supervised teacher; however, with our model formulation, the self-supervised knowledge can be well distilled to improve the generalization of a student network on the downstream tasks.

References

[1] Yanbei Chen, Yongqin Xian, A Koepke, Ying Shan, and Zeynep Akata. Distilling audio-visual knowledge by compositional contrastive learning. In *CVPR*, 2021.

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[3] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[4] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *JMLR*, 2008.

[5] Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. In *CVPR*, 2019.

[6] Nikolaos Passalis and Anastasios Tefas. Learning deep representations with probabilistic knowledge transfer. In *ECCV*, 2018.

[7] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

[8] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *ICLR*, 2015.

[9] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. In *ICLR*, 2019.

[10] Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu. Deep mutual learning. In *CVPR*, 2018.