Knowledge Transfer Graph for Deep Collaborative Learning

Tao Shen

Zhejiang University

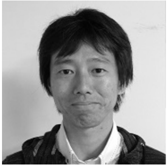
September 21, 2019

Overview

Authors

Department of Computer Science,
Chubu University,
Machine Perception & Robotics Group

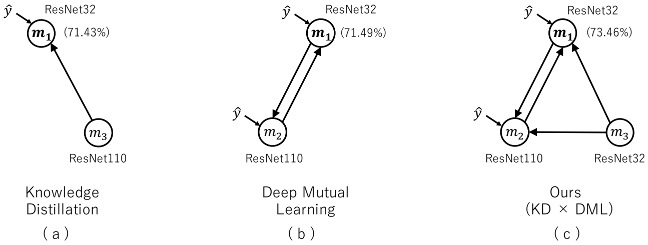





Soma Minami TsubasaHirakawa Takayoshi Yamashita Hironobu Fujiyoshi

KD vs DML vs DCL

Deep Collaborative Learning (DCL) is a method that incorporates Knowledge Distillation and Deep Mutual Learning,



and represents graph using a more generalized knowledge transfer method.

Knowledge Distillation

What is Knowledge Distillation?

Model Compression
Knowledge Transfer

How it works?

Training a student model from the output of teacher model rather than raw dataset.

Why it works?

Soft-target is better
Errors in labels

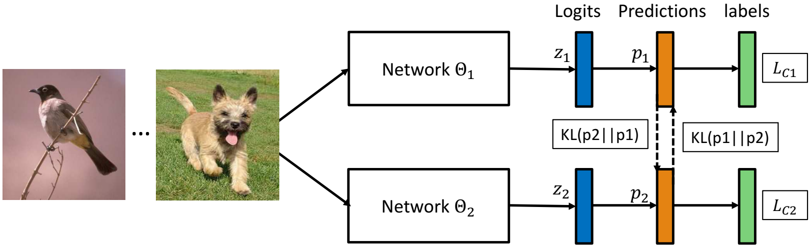
[1] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. NIPS Deep Learning and Representation Learning Workshop, 2015.

Deep Mutual Learning

What is Deep Mutual Learning?

KD needs a pre-trained model & Single-direction

How it works?



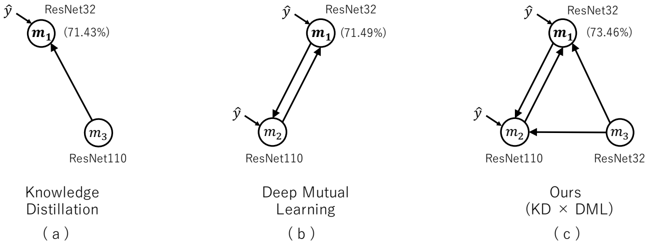
Why it works?

Self-Loss is more important at the beginning.

[2] Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu. Deep mutual learning. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

Deep Collaborative Learning

Deep Collaborative Learning (DCL) is a method that incorporates Knowledge Distillation and Deep Mutual Learning,

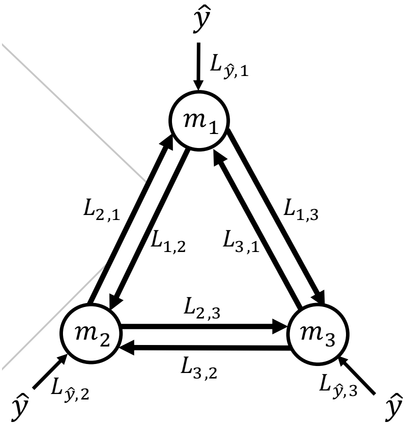


and represents graph using a more generalized knowledge transfer method.

Contributions

1. A DCL method
   - Multiple networks incorporating KD & DM
   - Hyperparameter search for optimal knowledge graph
2. Four types of gate structure
   - Through, cutoff, linear, correct gates to control gradient information
   - Contributes to the improvement of accuracy
3. More accurate than the conventional method.
4. The optimized graph can be transferred to a different dataset

Transfer Graph



Define a directed graph where node mi represents the ith model used in learning, and two edges are defined for each node. These edges represent the directions in which gradient information is transferred.

Loss Functions

Example (Self Loss)

$$
\begin{aligned}
H\left(\boldsymbol{p}_{\hat{y}_n}, \boldsymbol{p}_t\left(\boldsymbol{x}_n\right)\right) &= KL\left(\boldsymbol{p}_{\hat{y}_n} \| \boldsymbol{p}_t\left(\boldsymbol{x}_n\right)\right) + H\left(\boldsymbol{p}_{\hat{y}_n}, \boldsymbol{p}_{\hat{y}_n}\right) \\
&= KL\left(\boldsymbol{p}_{\hat{y}_n} \| \boldsymbol{p}_t\left(\boldsymbol{x}_n\right)\right)
\end{aligned}
$$

$$
H\left(\boldsymbol{p}_{\hat{y}_n}, \boldsymbol{p}_t\left(\boldsymbol{x}_n\right)\right) = KL\left(\boldsymbol{p}_{\hat{y}_n} \| \boldsymbol{p}_t\left(\boldsymbol{x}_n\right)\right)
$$

Example (Mutual Loss)

$$
L_{s,t} = \sum_{n}^{|\mathcal{B}|} G_{s,t}\left(KL\left(\boldsymbol{p}_s\left(\boldsymbol{x}_n\right) \| \boldsymbol{p}_t\left(\boldsymbol{x}_n\right)\right)\right)
$$

Overall Loss:

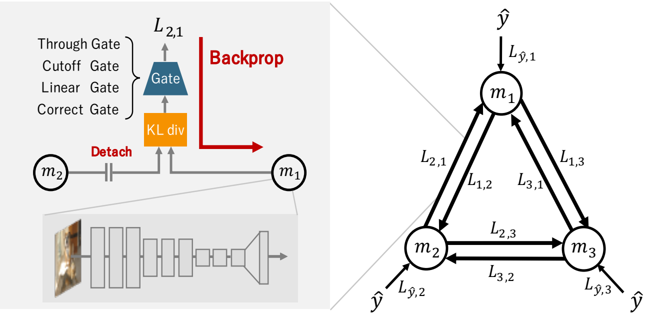
$$
L_t = \sum_{s=0, s\neq t}^{M} L_{s,t}
$$

Gates

Example (Gates)

$$G_{s,t}^{Through}(a) = a \qquad G_{s,t}^{Cutoff}(a) = 0$$

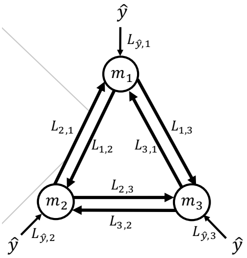
$$G_{s,t}^{Linear}(a) = \frac{k}{k_{end}} a \qquad G_{s,t}^{Correct}(a; \hat{y}, y_s) = \delta_{\hat{y}, y_s} \cdot a$$



Pseudocode

Algorithm 1 Network Optimization

Input: Number of nodes M, number of epochs E
Initialize: Initialize all network weightings, or read in the weightings of a trained network
for _ = 1 to E **do**
  Input the same image to each network m_n, and obtain the response value $\boldsymbol{p}_n$.
  Obtain the loss L_n according to Eq. (3).
  Obtain the update quantity of m_n from the gradient L_n.
  Update the weighting of all the networks.
end for



Graph optimization

Asynchronous Successive Halving Algorithm (ASHA)

The hyperparameters to be optimized are the model type and gate type.

[3] Liam Li, Kevin Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. Massively parallel hyperparameter tuning. arXiv preprint arXiv:1810.05934, 2018.

Experimental setting

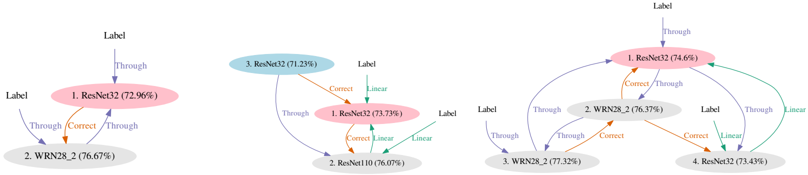
Example (Dataset)

We used the CIFAR-10, CIFAR-100 and Tiny-ImageNet datasets, which are used for general object recognition. CIFAR-10 and CIFAR-100 consist of 50,000 images for training and 10,000 images for verification. Both datasets consist of images with dimensions of 32x32 pixels, and include labels for 10 classes and 100 classes.

Example (Models)

We used three typical network: ResNet32, ResNet110, and Wide ResNet 28-2. The other nodes were selected by ASHA to achieve the best recognition rate at the evaluation target node (ResNet32).

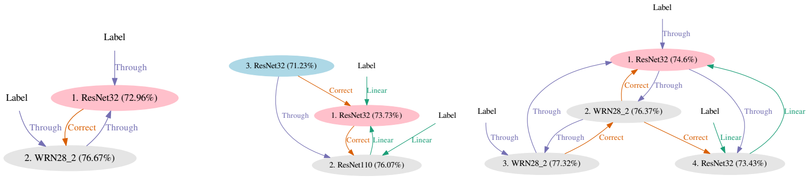
Optimized Graph



(a) 2 nodes **(b) 3 nodes** **(c) 4 nodes**

	ResNet32	ResNet110	Wide ResNet 28-2
Accuracy [%]	70.71 ± 0.39	72.59 ± 0.54	74.60 ± 0.38

Comparison



(a) 2 nodes **(b) 3 nodes** **(c) 4 nodes**

Method	Accuracy (Node 1)	Node 1	Node 2	Node 3	Node 4
KD ($T = 2$) [6]	71.43 ± 0.43	ResNet32	ResNet110*	-	-
DML [21]	71.49 ± 0.24	ResNet32	ResNet110	-	-
DML [21]	72.09 ± 0.43	ResNet32	ResNet32	ResNet32	-
DCL (Ours)	$\mathbf{73.46} \pm 0.42$	ResNet32	ResNet110	ResNet32*	-
DML [21]	72.76 ± 0.35	ResNet32	ResNet32	ResNet32	ResNet32
[17]	$73.36^{**} \pm 0.26$	(Multiple ResNet32 with shared intermediate layers)			
ONE [9]	$73.48^{**} \pm$ N/A	(Multiple ResNet32 with shared intermediate layers)			
DCL (Ours)	$\mathbf{74.34} \pm 0.32$	ResNet32	WRN28-2	WRN28-2	ResNet32

Validity of gates on various datasets

# of nodes	Gates	CIFAR10	CIFAR-100	Tiny-ImageNet
1	-	93.12 ± 0.27	70.71 ± 0.39	53.18 ± 0.08
2	Fixed (Through Gate)	93.25 ± 0.50	72.47 ± 0.78	$\mathbf{54.93} \pm 0.29$
	Optimized	$\mathbf{93.65} \pm 0.14$	$\mathbf{72.88} \pm 0.41$	54.69 ± 0.16
3	Fixed (Through Gate)	93.53 ± 0.24	71.88 ± 0.43	53.78 ± 0.78
	Optimized	$\mathbf{93.92} \pm 0.20$	$\mathbf{73.46} \pm 0.42$	$\mathbf{55.02} \pm 0.31$
4	Fixed (Through Gate)	93.01 ± 0.79	73.40 ± 0.39	53.92 ± 0.21
	Optimized	$\mathbf{93.99} \pm 0.27$	$\mathbf{74.34} \pm 0.32$	$\mathbf{55.80} \pm 0.26$
5	Fixed (Through Gate)	93.61 ± 0.23	73.40 ± 0.28	52.12 ± 0.30
	Optimized	$\mathbf{94.14} \pm 0.16$	$\mathbf{74.54} \pm 0.59$	$\mathbf{55.30} \pm 0.16$
6	Fixed (Through Gate)	93.84 ± 0.39	73.85 ± 0.45	49.37 ± 1.70
	Optimized	$\mathbf{94.17} \pm 0.21$	$\mathbf{74.22} \pm 0.22$	$\mathbf{55.16} \pm 0.19$
7	Fixed (Through Gate)	93.75 ± 0.27	73.53 ± 0.27	53.10 ± 0.44
	Optimized	$\mathbf{94.07} \pm 0.14$	$\mathbf{74.71} \pm 0.23$	$\mathbf{54.78} \pm 0.36$

Graph transfer to another dataset

# of nodes	CIFAR-100	CIFAR-10 to CIFAR-100
2	**72.88** ± 0.41	72.47 ± 0.37
3	73.46 ± 0.42	**73.63** ± 0.18
4	**74.34** ± 0.32	73.76 ± 0.25
5	74.54 ± 0.59	**74.62** ± 0.24

Conclusion & future work

1. A DCL method
   - Multiple networks incorporating KD & DM
   - Hyperparameter search for optimal knowledge graph
2. Four types of gate structure
   - Through, cutoff, linear, correct gates to control gradient information
   - Contributes to the improvement of accuracy
3. More accurate than the conventional method.
4. The optimized graph can be transferred to a different dataset

Knowledge transfer from an intermediate layer?
Knowledge transfer using the ensemble of multiple networks?

The End