Knowledge Transfer via Dense Cross-Layer Mutual-Distillation

Ksenofontov Gregory

MIPT

May 16, 2023

KD and DML

Given the training data $X = \{x_n\}_{n=1}^N$ of M classes, with labels $Y = \{y_n\}_{n=1}^N$. W_t – a teacher network, and W_s – a student network.

Knowledge Distillation

The teacher network W_t is pretrained and fixed. The student network is trained by minimizing

$$L_s = L_c(W_s, X, Y) + \lambda L_{kd}(\hat{P}_t, \hat{P}_s),$$

where L_{kd} – cross entropy

$$L_{kd}(\hat{P}_t, \hat{P}_s) = \frac{1}{N} \sum_{n=1}^{N} \sum_{m=1}^{M} \hat{P}_t^m(x_n) \log \hat{P}_s^m(x_n)$$

and λ – a balancing coefficient

KD and DML

Given the training data $X = \{x_n\}_{n=1}^N$ of M classes, with labels $Y = \{y_n\}_{n=1}^N$. W_t – a teacher network, and W_s – a student network.

Deep Mutual Learning

The teacher network W_t and the student network are trained by jointly minimizing

$$L_s = L_c(W_s, X, Y) + \lambda L_{dml}(\hat{P}_t, \hat{P}_s)$$

$$L_t = L_c(W_t, X, Y) + \lambda L_{dml}(\hat{P}_s, \hat{P}_t),$$

where L_{dml} – KL divergence (relative entropy)

$$L_{dml}(\hat{P}_t, \hat{P}_s) = \frac{1}{N} \sum_{n=1}^{N} \sum_{m=1}^{M} \hat{P}_t^m(x_n) \log \frac{\hat{P}_t^m(x_n)}{\hat{P}_s^m(x_n)}$$

and λ – a balancing coefficient

KD and DML

Authors of the article proposes a new method **Dense Cross-layer Mutual-distillation (DCM)**, that enhances DML via jointly addressing two issues listed below.

1. The information contained in the hidden layers of networks has not been explored
2. The problem of connecting more effective knowledge representation learning

DCM

Given the training data $X = \{x_n\}_{n=1}^N$ of M classes, with labels $Y = \{y_n\}_{n=1}^N$. W_t – a teacher network, and W_s – a student network.

Dense Cross-layer Mutual-distillation

Let Q $= \{(t_k, s_k)\}_{k=1}^K$ – set of K pairs of the same-staged layer indices of the teacher network W_t and the student network W_s, indicating the locations of auxiliary classifiers ((t_{K+1}, s_{K+1}) – default classifier). DCM simultaneously minimizes the following two objectives

$$L_s = L_c(W_s, X, Y) + \alpha L_{ds}(W_s, X, Y) + L_{dcm1}(\hat{P}_t, \hat{P}_s) + \gamma L_{dcm2}(\hat{P}_t, \hat{P}_s)$$

$$L_t = L_c(W_t, X, Y) + \alpha L_{ds}(W_t, X, Y) + L_{dcm1}(\hat{P}_s, \hat{P}_t) + \gamma L_{dcm2}(\hat{P}_s, \hat{P}_t),$$

DCM

$L_{ds}(W_s, X, Y)$

This loss denotes sum of cross-entropy losses for all auxiliary classifier

$$L_{ds}(W_s, X, Y) = \sum_{k=1}^{K} L_c(W_s, X, Y)$$

$L_{dcm}(\hat{P}_t, \hat{P}_s)$

This two losses in total denote sum of all KD losses of all pair combinations of classifiers

$$L_{dcm1}(\hat{P}_t, \hat{P}_s) = \sum_{k=1}^{K+1} L_{kd}(\hat{P}_{t_k}, \hat{P}_{s_k})$$

$$L_{dcm2}(\hat{P}_t, \hat{P}_s) = \sum_{i,j=1;i\neq j}^{K+1} L_{kd}(\hat{P}_{t_i}, \hat{P}_{s_j})$$

DCM algorithm

Algorithm 1: The DCM algorithm

Input : Training data $\{X, Y\}$, two CNN models W_t and W_s, classifier locations $\{(t_k, s_k)\}_{k=1}^{K+1}$, learning rate γ_i

Initialise W_t and W_s, $i = 0$;

repeat

$i \leftarrow i + 1$, update γ_i;

1. Randomly sample a batch of data from $\{X, Y\}$;

2. Compute knowledge set $\{(\hat{P}_{t_k}, \hat{P}_{s_k})\}_{k=1}^{K+1}$ at all supervised layers of two models by Eq. 3;

3. Compute loss L_t and L_s by Eq. 6, Eq. 7, Eq. 8, and Eq. 9 ;

4. Calculate gradients and update parameters:

$W_t \leftarrow W_t - \gamma_i \frac{\partial L_t}{\partial W_t}$, $W_s \leftarrow W_s - \gamma_i \frac{\partial L_s}{\partial W_s}$

until *Converge*;

Setting of Q

1. *Where to place auxiliary classifiers?* Use a **practical** principle, adding auxiliary classifiers merely to down-sampling layers of a backbone network
2. *How to design structures of auxiliary classifiers?* Use a **heuristic** principle, making the paths from the input to all auxiliary classifiers have the same number of downsampling layers as the backbone network and using backbone's building blocks

Main experiments

Networks		Ind(baseline)		DML		DCM	
Net1	Net2	Net1	Net2	Net1	Net2	Net1|**DCM-DML**	Net2|**DCM-DML**
ResNet-164	ResNet-164	22.56(0.20)	22.56(0.20)	20.69(0.25)	20.72(0.14)	19.57(0.20)|**1.12**	19.59(0.15)|**1.13**
WRN-28-10	WRN-28-10	18.72(0.24)	18.72(0.24)	17.89(0.26)	17.95(0.07)	16.61(0.24)|**1.28**	16.65(0.22)|**1.30**
DenseNet-40-12	DenseNet-40-12	24.91(0.18)	24.91(0.18)	23.18(0.18)	23.15(0.20)	22.35(0.12)|**0.83**	22.41(0.17)|**0.74**
WRN-28-10	ResNet-110	18.72(0.24)	26.55(0.26)	17.99(0.24)	24.42(0.19)	17.82(0.14)|**0.17**	22.99(0.30)|**1.43**
WRN-28-10	WRN-28-4	18.72(0.24)	21.39(0.30)	17.80(0.11)	20.21(0.16)	16.84(0.08)|**0.96**	18.76(0.14)|**1.45**
WRN-28-10	MobileNet	18.72(0.24)	26.30(0.35)	17.24(0.13)	23.91(0.22)	16.83(0.07)|**0.41**	21.43(0.20)|**2.48**
WRN-28-10(+)	WRN-28-10(+)	18.64(0.19)	18.64(0.19)	17.62(0.12)	17.61(0.13)	16.57(0.12)|**1.05**	16.59(0.15)|**1.02**

Setting of Q experiments

1. *Where to place auxiliary classifiers?*

Classifier locations	WRN-28-10	
	Net1	Net2
baseline	18.72(0.24)	18.72(0.24)
C1+C4	17.16(0.14)	17.25(0.15)
C1+C3	16.89(0.21)	17.04(0.06)
C1+C2	17.40(0.20)	17.38(0.17)
C1+C2C3(default)	16.61(0.24)	16.65(0.22)
C1+C2C3C4	**16.59**(0.12)	16.73(0.17)

2. *How to design structures of auxiliary classifiers?*

Net1/Net2	Classifier type	DCM
DenseNet-40-12	APFC	25.10(0.25)
	Narrow	22.45(0.25)
	default	**22.35**(0.12)
WRN-28-10	APFC	18.23(0.10)
	Narrow	16.88(0.17)
	default	**16.61**(0.24)

Loss experiments

Method	Error (%)	
	Net1	Net2
baseline	24.91(0.18)	24.91(0.18)
DML	23.18(0.18)	23.15(0.20)
DML + DS	23.18(0.33)	23.08(0.28)
DCM-1	22.86(0.16)	22.89(0.14)
DCM-2	22.43(0.25)	22.51(0.18)
DCM	**22.35(0.12)**	**22.41(0.17)**

Noise experiments

Corruption ratio	Method	Error (%)
0.2	baseline	9.85(0.24)
	DML	8.13(0.14)
	DCM	**7.11(0.11)**
0.5	baseline	17.93(0.39)
	DML	14.31(0.30)
	DCM	**12.08(0.34)**
0.8	baseline	35.32(0.42)
	DML	32.65(0.96)
	DCM	**31.26(0.94)**