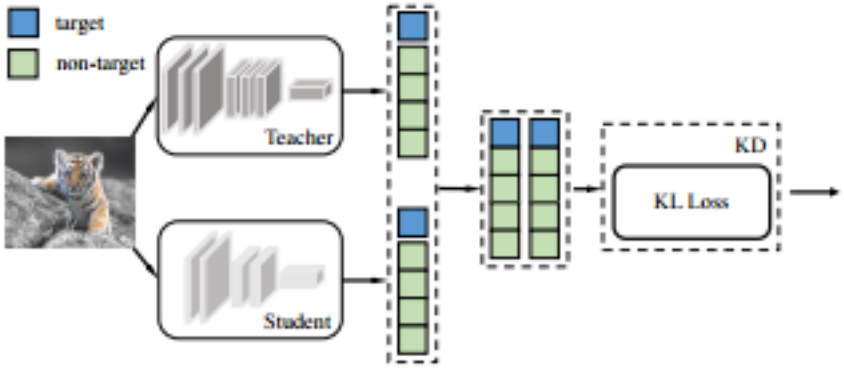
Note:Decoupled Knowledge Distillation

Sinkoo

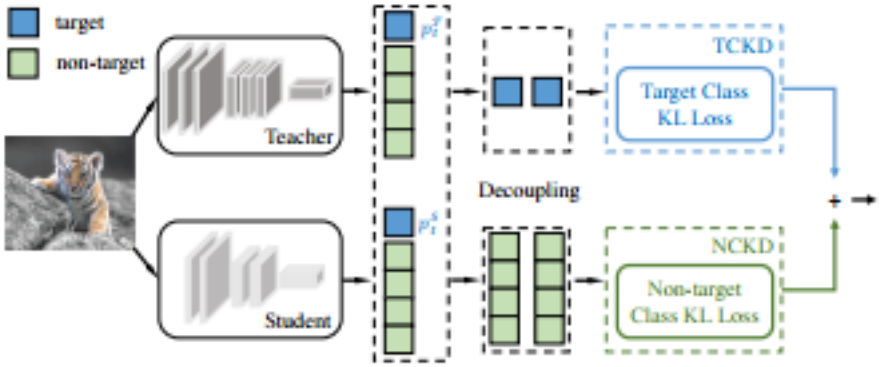
April 10, 2022

1 Abstract

Author gives that DK KL can be divide into 2 parts : NCKD and TCKD and this paper thinks the conventional DK method suppresses the effectiveness of NCKD. So in this paper, authors proposed Decoupled Knowledge Distillation(DKD) to enable NCKD and TCKD to play their role more efficiently and flexibly with parameter α and β (actually only one is enough).



(a) Classical Knowledge Distillation (KD).



(b) Decoupled Knowledge Distillation (DKD).

Figure 1. Illustration of the classical KD [12] and **our DKD**. We reformulate KD into a weighted sum of two parts, *i.e.*, TCKD and NCKD. The first equation shows that KD (1) couples NCKD with $p_t^{\mathcal{T}}$ (the teacher's confidence on the target class), and (2) couples the importance of two parts. Furthermore, we demonstrate that the first coupling suppresses the effectiveness, and the second limits the flexibility for knowledge transfer. We propose DKD to address these issues, which employs hyper-parameters α for TCKD and β for NCKD, killing the two birds with one stone.

2 Rethinking KD

Reformulate KD loss into a weighted sum of two parts, one is relevant to the target class(TCKD), and the other is not(NCKD). ($p_t$is the target class, C is number of classes)

$$KD = p_t^\tau log(\frac{p_t^\tau}{p_t^S}) + p_{\backslash}^\tau log(\frac{p_{\backslash t}^\tau}{p_{\backslash t}^S}) + p_{\backslash t}^\tau \sum_{i=1, i\neq t}^{C} \hat{p}_i^\tau log(\frac{\hat{p}_i^\tau}{\hat{p}_i^S})$$

$$KD = KL(b^\tau || b^S) + (1 - p_t^\tau) KL(\hat{p}^\tau || \hat{p}^S)$$

Name $KL(b^\tau || b^S)$ as TCKD, $KL(\hat{p}^\tau || \hat{p}^S)$ as NCKD. Details are in the Paper.

3 Model

Decouple NCKD with $(1-p_t^\tau)$ by give weights(coefficients) to this 2 part of DK.

$$KD = \alpha TCKD + \beta NCKD$$

Algorithm 1 Pseudo code of DKD in a PyTorch-like style.

```
# logit_stu: student output logits
# logit_tea: teacher output logits
# T: temperature for KD & DKD
# t: labels, (N, C), bool type
# alpha, beta: hyper-parameters for DKD

p_stu = F.softmax(logit_stu / T)
p_tea = F.softmax(logit_tea / T)
# pt & pnt: (N, 1), Eqn.(2)
pt_stu, pnt_stu = p_stu[t], p_stu[1-t].sum(1)
pt_tea, pnt_tea = p_tea[t], p_tea[1-t].sum(1)
# pnct: (N, C-1), Eqn.(3)
pnct_stu = F.softmax(l_stu[1-t]/T)
pnct_tea = F.softmax(l_tea[1-t]/T)

# TCKD
tckd = kl_div(log(pt_stu), pt_tea) +
       kl_div(log(pnt_stu), pnt_tea)
# NCKD
nckd = F.kl_div(log(pnct_stu), pnct_tea)

# ori KD
# kd_loss = (tckd + pnt_tea*nckd) * T**2
# DKD
dkd_loss = (alpha*tckd + beta*nckd) * T**2
```

4 Summary

This paper delved into KD and the KL loss, seperated KL loss into 2 parts: TCKD and NCKD, TCKD transfers "difficulty"(as data get harder to learn, TCKD helps more.) and NCKD is the prominent reason why logit distillation works but is greatly suppressed. This paper found that conventional KD depressed NCKD, so in this paper author gives coefficient to each part and got

good performance.