BMVC 2022 #705

Multi-task Curriculum Learning Based on Gradient Similarity

DENSO Crafting the Core

Hiroaki Igarashi[1], Kenichi Yoneji[1], Kohta Ishikawa[2], Rei Kawakami[3], Teppei Suzuki[2], Shingo Yashima[2], and Ikuro Sato[23]

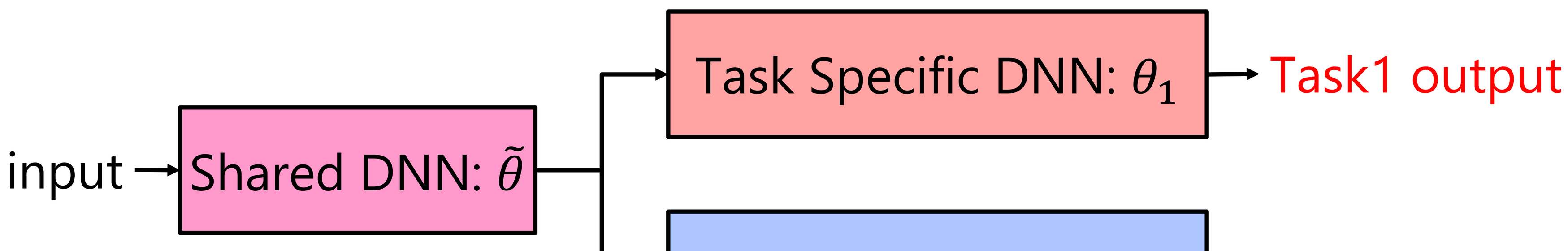
[1]DENSO Corp., [2]DENSO IT Lab., [3]Tokyo Tech

Background

Robotics applications such as autonomous driving require multiple perceptional tasks.
(e.g. Object Detection and Semantic Segmentation)

⇒ Multi-task learning (MTL)

MTL Model (e.g. 2 task)



MTL shares a portion of the network between multiple tasks, and reduce the complexity.

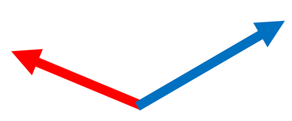
Naïve update rule

$$\theta' \leftarrow \theta - \eta \frac{1}{N}\sum_{i=0}^{N} g_i$$

L_i: Loss of task i, $g_i = \frac{dL_i}{d\theta}$: Gradient of task i,
N: #of task, η: learning rate

A major challenge of MTL: gradient conflict

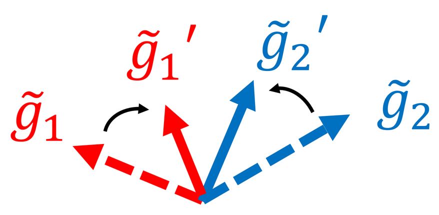
Gradient components can point in opposite directions between tasks.

$\tilde{g}_1 = \frac{dL_1}{d\tilde{\theta}}$  $\tilde{g}_2 = \frac{dL_2}{d\tilde{\theta}}$

In shared DNN ($\tilde{\theta}$), since the parameter updates of each task are oriented different directions, conflicting gradients sometimes lead to insufficient performance for each task.

Related Works

PCGrad [29]



$$\tilde{g}_1' = \tilde{g}_1 - \boxed{\frac{\tilde{g}_1 \cdot \tilde{g}_2}{\|\tilde{g}_2\|^2}\tilde{g}_2} \qquad \tilde{g}_2' = \tilde{g}_2 - \boxed{\frac{\tilde{g}_1 \cdot \tilde{g}_2}{\|\tilde{g}_1\|^2}\tilde{g}_1}$$

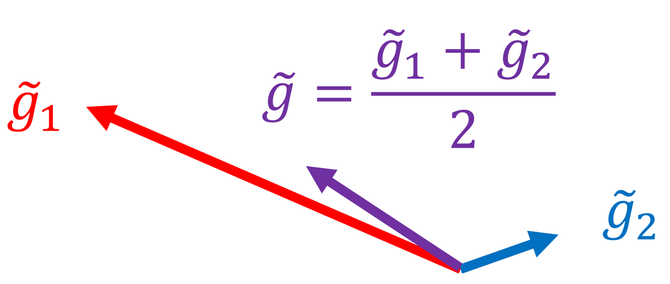
Conflicting components

PCGrad manipulates gradients such that the conflicting components are removed.

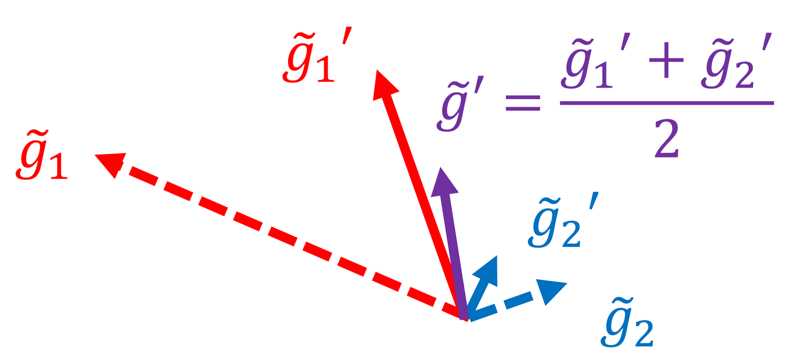
A Problem of PCGrad

e.g. when each task has a big gap in magnitude of gradients

w/o PCGrad



w/ PCGrad



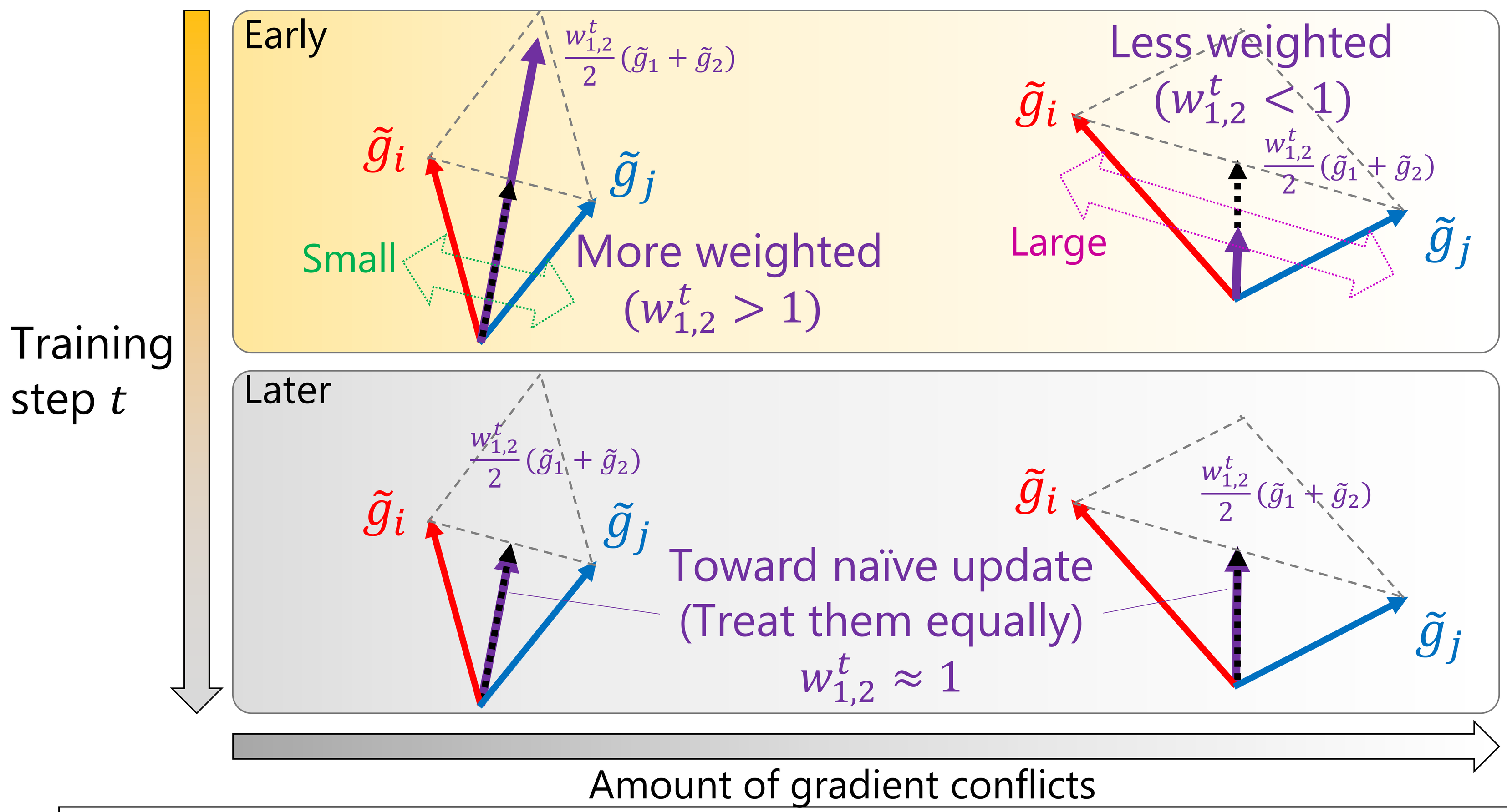
Acquired gradient by PCGrad is much different from the original one.

A converged solution is no longer optimal for original objective due to gradient manipulation.

Proposed method (MCLGS)

=Multi-task Learning × Curriculum Learning*

*It removes hard samples in the early stage of training and makes the solution better.



Loss weight: $w_{i,j}^t = \tanh(s(\tilde{g}_1, \tilde{g}_2)p(t)) + 1$

$$s(\tilde{g}_1, \tilde{g}_2) = \frac{\tilde{g}_1 \cdot \tilde{g}_2}{\|\tilde{g}_1\|\|\tilde{g}_2\|}$$

$$p(t) = \max(a_0 - t\Delta a, 0)$$

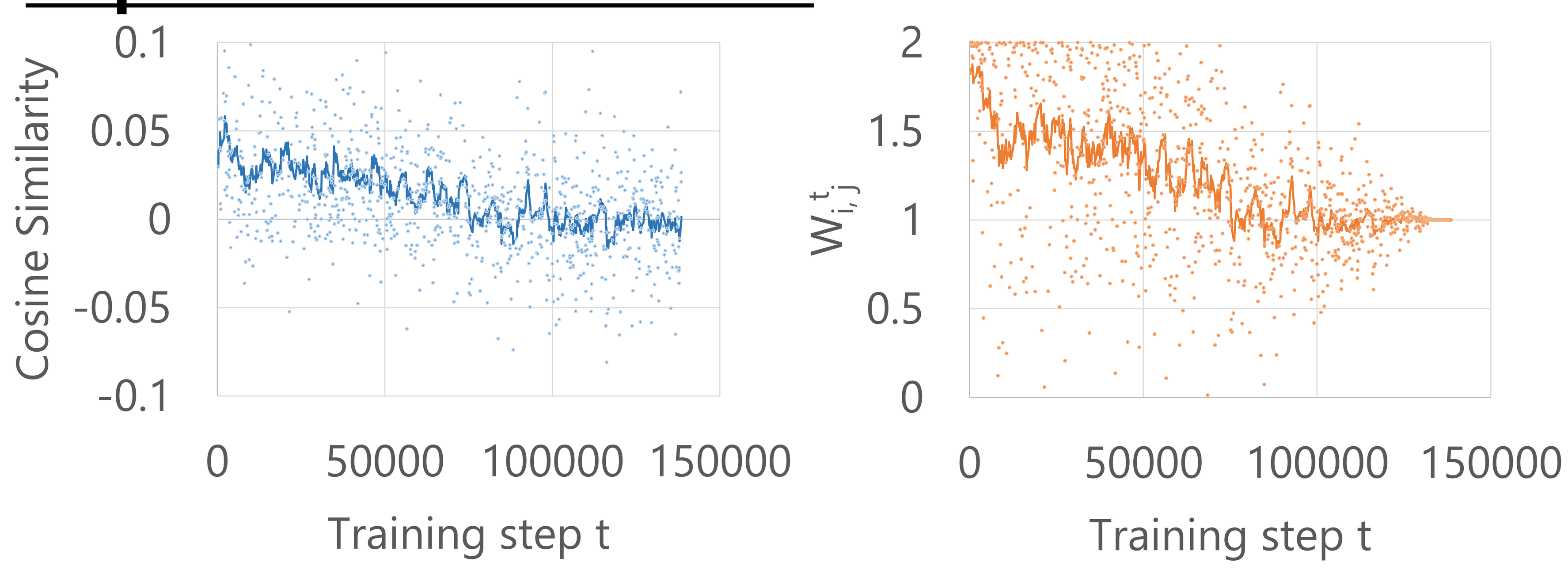
* a_0,Δa: hyper parameters

MCLGS doesn't manipulate gradients but just downweights samples that generate gradient conflicts in the early stage of training.

MCLGS's update rule

$$\theta' \leftarrow \theta - \eta \frac{1}{N(N-1)} \sum_{i=0}^{N-2} \sum_{j=i+1}^{N-1} w_{i,j}^t (g_i + g_j)$$

Experimental Results



The weight depends on the similarity in the early stage of training, but converges to a fixed value of 1 at the end of training.

The NYUv2 Dataset [22]

Methods	Improvement from STL Baseline % ↑			
	Δseg	Δdepth	Δnormal	Δm (mean±stderr*)
STL Baseline	0.00	0.00	0.00	0.00±0.00
MTL Baseline	-0.32	6.39	-4.59	0.49±1.03
MGDA [21]	-4.48	0.03	**2.33**	-0.71±0.84
PCGrad [29]	0.07	**7.37**	-3.25	1.40±0.56
GradDrop [4]	0.06	6.30	-3.92	0.81±0.42
CAGrad [16]	0.63	3.24	-0.28	1.20±0.82
MCLGS (ours)	1.94	6.59	-2.18	2.12±0.48
CAGrad [16] + MCLGS (ours)	**4.08**	4.68	1.34	**3.37±0.72**

The BDD100K Dataset [28]

methods	improvement from STL baseline % ↑		
	Δod	Δseg	Δm (mean±stderr*)
STL baseline	0.00	0.00	0.00±0.00
MTL baseline	3.10	3.56	3.33±0.17
MGDA [21]	-39.57	-6.14	-22.85±0.40
PCGrad [29]	3.06	3.44	3.25±0.23
GradDrop [4]	2.85	3.55	3.20±0.21
CAGrad [16]	0.00	2.10	1.05±0.23
MCLGS (ours)	**3.71**	**4.34**	**4.03±0.27**

*The model is trained over 3 random seeds, and the average and the stderr are reported.