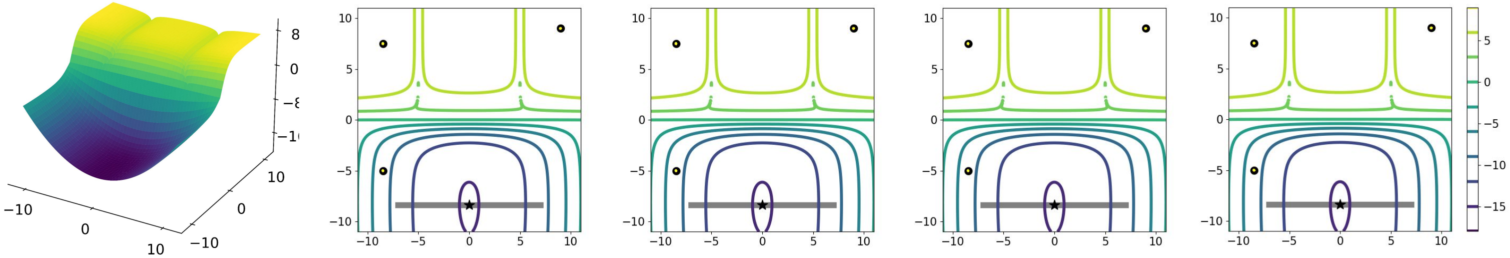
Conflict-Averse Gradient Descent for Multitask Learning



Loss Landscape | GD | MGDA [Sener et al., 2018] | PCGrad [Yu et al., 2020] | CAGrad (ours)

Bo Liu[1], Xingchao Liu[1], Xiaojie Jin[2], Peter Stone[1,3], Qiang Liu[1]

[1]The University of Texas at Austin, [2]Bytedance Research, [3]Sony AI

2021 Conference on Neural Information Processing Systems (NeurIPS)

Multitask Learning (MTL)

Definition of MTL:

Learning a *single* model that can tackle *multiple* different tasks.

Multitask Learning (MTL)

Definition of MTL:

Learning a *single* model that can tackle *multiple* different tasks.

Why MTL?

Multitask Learning (MTL)

Definition of MTL:

Learning a *single* model that can tackle *multiple* different tasks.

Why MTL:

- **Necessity:** An ideal intelligent agent should possess diverse skills.

Multitask Learning (MTL)

Definition of MTL:

Learning a *single* model that can tackle *multiple* different tasks.

Why MTL:

- **Necessity:** An ideal intelligent agent should possess diverse skills.
- **Better Efficiency**: MTL methods learn *more efficiently* with an overall *smaller* model compared to learning separate models.

Multitask Learning (MTL)

Definition of MTL:

Learning a *single* model that can tackle *multiple* different tasks.

Why MTL:

- **Necessity:** An ideal intelligent agent should possess diverse skills.
- **Better Efficiency**: MTL methods learn *more efficiently* with an overall *smaller* model compared to learning separate models.
- **Improved Performance**: It has been shown that MTL can improve the quality of representation learning across different tasks [1].

[1] Swersky, Kevin, Jasper Snoek, and Ryan Prescott Adams. "Multi-task bayesian optimization." (2013).

Formal Definition

Definition of MTL:

Learning a *single* model that can tackle *multiple* different tasks.

Formally, assume we have $K \geq 2$ tasks, each task has its own loss function $L_i(\theta)$ with a shared set of parameters θ. The objective is to optimize:

$$\theta^* = \arg\min_{\theta \in \mathbb{R}^m} \left\{ L_0(\theta) \triangleq \frac{1}{K} \sum_{i=1}^{K} L_i(\theta) \right\}.$$

Formal Definition

Definition of MTL:

Learning a *single* model that can tackle *multiple* different tasks.

Formally, assume we have $K \geq 2$ tasks, each task has its own loss function $L_i(\theta)$ with a shared set of parameters θ. The objective is to optimize:

$$\theta^* = \arg\min_{\theta \in \mathbb{R}^m} \left\{ L_0(\theta) \triangleq \frac{1}{K} \sum_{i=1}^{K} L_i(\theta) \right\}.$$

Remark: we implicitly assume the preference over tasks are expressed in individual losses $L_i(\theta)$ so that the goal is to search for an optimum of the average loss.

Optimization Challenge: Conflicting Gradients

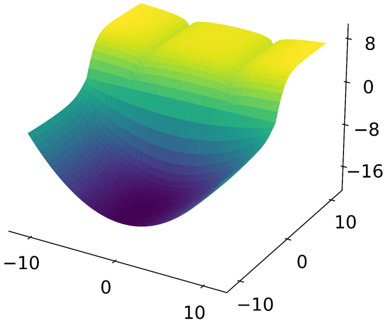
Directly optimizing the average loss $L_0(\theta)$ can be challenging.

Denote $g_i = \nabla_\theta L_i(\theta)$ the task gradient and $g_0 = \nabla_\theta L_0(\theta)$ the average task gradient. Then, conflicting gradients means that $\exists\ i, \quad \langle g_i, g_0 \rangle < 0.$

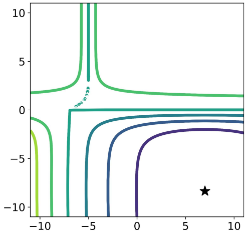
In other words, updating the average loss can **sacrifice** the performance of an individual task. This could lead to failure of optimization!

Optimization Challenge: Conflicting Gradients

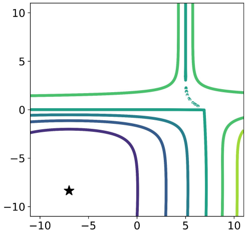
Denote $g_i = \nabla_\theta L_i(\theta)$ the task gradient and $g_0 = \nabla_\theta L_0(\theta)$ the average task gradient. Then, conflicting gradients means that $\exists\ i, \quad \langle g_i, g_0 \rangle < 0.$



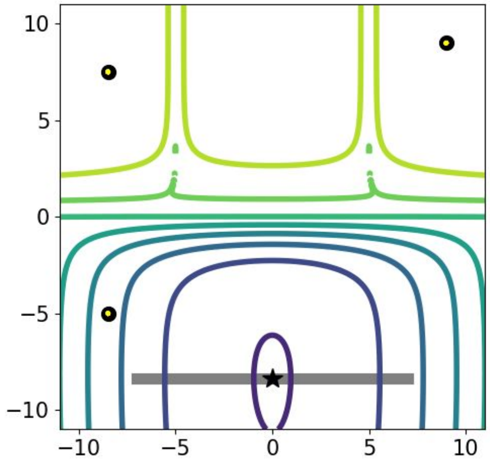
Loss Landscape



Task 1 Loss



Task 2 Loss



Visualization of optimization using Adam starting from 3 initial points.

Gradient Descent (GD) can get stuck at places of "high curvature", due to the conflicting gradients.

Pareto Concepts

Unlike single task learning where any two parameter vectors θ_1 and θ_2 can be ordered in the sense that either $L(\theta_1) \leq L(\theta_2)$ or $L(\theta_2) \leq L(\theta_1)$, MTL can have two parameter vectors where one performs better on task i and the other performs better on task j.

Pareto Concepts

Unlike single task learning where any two parameter vectors θ_1 and θ_2 can be ordered in the sense that either $L(\theta_1) \leq L(\theta_2)$ or $L(\theta_2) \leq L(\theta_1)$. MTL can have two parameter vectors where one performs better on task i and the other performs better on task j.

To this end, we need the concept of pareto optimality:

Pareto Optimality and Pareto Set (Informal)

A parameter is Pareto-optimal if no other parameters perform uniformly better than it. The set of all Pareto-optimal points is the Pareto set.

Prior Attempts and Convergence

Several methods are proposed to mitigate the challenge in MTL optimization. In this work, we mainly focus on gradient manipulation methods that calculate a new update using task gradients (other methods include novel multi-task network design [1]). Representatives are:

1. Multiple-gradient descent algorithm (MGDA) [2]: directly optimize towards the pareto set.
2. Dynamically reweighting each objective [3].
3. Projecting Gradient [4]: project each gradient to the normal plane of others.

[1] Liu, Shikun, Edward Johns, and Andrew J. Davison. "End-to-end multi-task learning with attention." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019.
[2] Sener, Ozan, and Vladlen Koltun. "Multi-task learning as multi-objective optimization." *Conference on Neural Information Processing Systems.* 2018.
[3] Chen, Zhao, et al. "Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks." *International Conference on Machine Learning*. PMLR, 2018.
[4] Yu, Tianhe, et al. "Gradient surgery for multi-task learning." *Conference on Neural Information Processing Systems.* 2020.

Prior Attempts and Convergence

Several methods are proposed to mitigate the challenge in MTL optimization. In this work, we mainly focus on gradient manipulation methods that calculate a new update using task gradients (other methods include novel multi-task network design [1]). Representatives are:

1. Multiple-gradient descent algorithm (MGDA) [2]: directly optimize towards the pareto set.
2. Dynamically reweighting each objective [3].
3. Projecting Gradient [4]: project each gradient to the normal plane of others.

Remark: while all these methods mitigate the challenge in MTL optimization, they manipulate the gradient without respecting the original objective. Therefore, they either have *no convergence guarantee* or can converge to *any* point on the Pareto-set in principle.

[1] Liu, Shikun, Edward Johns, and Andrew J. Davison. "End-to-end multi-task learning with attention." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019.
[2] Sener, Ozan, and Vladlen Koltun. "Multi-task learning as multi-objective optimization." *Conference on Neural Information Processing Systems.* 2018.
[3] Chen, Zhao, et al. "Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks." *International Conference on Machine Learning*. PMLR, 2018.
[4] Yu, Tianhe, et al. "Gradient surgery for multi-task learning." *Conference on Neural Information Processing Systems.* 2020.

Conflict-Averse Gradient Descent (CAGrad)

Assume we update θ by $\theta' = \theta - \alpha d$, where α is the step size and d the update vector.

Conflict-Averse Gradient Descent (CAGrad)

Assume we update θ by $\theta' = \theta - \alpha d$, where α is the step size and d the update vector.

In general, we want to not only decrease the average loss, but also every individual loss. Therefore, we consider *the worst relative decrease* over individual losses:

$$R(\theta, d) = \max_i \left\{ \frac{1}{\alpha} \left(L_i(\theta - \alpha d) - L_i(\theta) \right) \right\} \approx - \min_i \langle g_i, d \rangle$$

Conflict-Averse Gradient Descent (CAGrad)

Assume we update θ by $\theta' = \theta - \alpha d$, where α is the step size and d the update vector.

In general, we want to not only decrease the average loss, but also every individual loss. Therefore, we consider *the worst relative decrease* over individual losses:

$$R(\theta, d) = \max_i \left\{ \frac{1}{\alpha} \left(L_i(\theta - \alpha d) - L_i(\theta) \right) \right\} \approx -\min_i \langle g_i, d \rangle$$

The objective of CAGrad is then:

$$\max_{d \in \mathbb{R}^m} \min_i \langle g_i, d \rangle \quad \text{s.t.} \quad \|d - g_0\| \leq c \, \|g_0\|$$

Conflict-Averse Gradient Descent (CAGrad)

Assume we update θ by $\theta' = \theta - \alpha d$, where α is the step size and d the update vector.

In general, we want to not only decrease the average loss, but also every individual loss. Therefore, we consider *the worst relative decrease* over individual losses:

$$R(\theta, d) = \max_i \left\{ \frac{1}{\alpha} \left(L_i(\theta - \alpha d) - L_i(\theta) \right) \right\} \approx -\min_i \langle g_i, d \rangle$$

The objective of CAGrad is then:

The worst improvement over tasks

still close to the average gradient, useful for convergence

$$\max_{d \in \mathbb{R}^m} \min_i \langle g_i, d \rangle \quad \text{s.t.} \quad \|d - g_0\| \leq c \|g_0\|$$

Conflict-Averse Gradient Descent (CAGrad)

In practice, we solve the **dual objective** for efficiency (the dual objective only involves K parameters where K is the number of tasks).

Algorithm 1 Conflict-averse Gradient Descent (CAGrad) for Multi-task Learning

---

Input: Initial model parameter vector θ_0, differentiable loss functions $\{L_i\}_{i=1}^K$, a constant $c \in [0, 1)$ and learning rate $\alpha \in \mathbb{R}^+$.

repeat

At the t-th optimization step, define $g_0 = \frac{1}{K}\sum_{i=1}^K \nabla L_i(\theta_{t-1})$ and $\phi = c^2 \|g_0\|^2$.

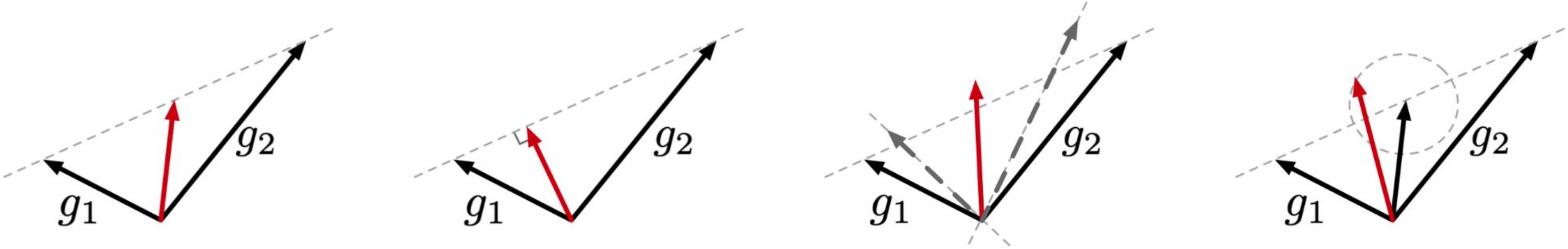
Solve

$$\min_{w \in \mathcal{W}} F(w) := g_w^\top g_0 + \sqrt{\phi}\,\|g_w\|\,, \text{ where } g_w = \frac{1}{K}\sum_{i=1}^K w_i \nabla L_i(\theta_{t-1}).$$

Update $\theta_t = \theta_{t-1} - \alpha\left(g_0 + \frac{\phi^{1/2}}{\|g_w\|} g_w\right)$.

until convergence

Conflict-Averse Gradient Descent (CAGrad)



GD

$$d = (g_1 + g_2)/2$$

MGDA

$$\max_d \min_i g_i^\top d \quad \text{s.t.} \quad \|d\| \leq 1$$

PCGrad

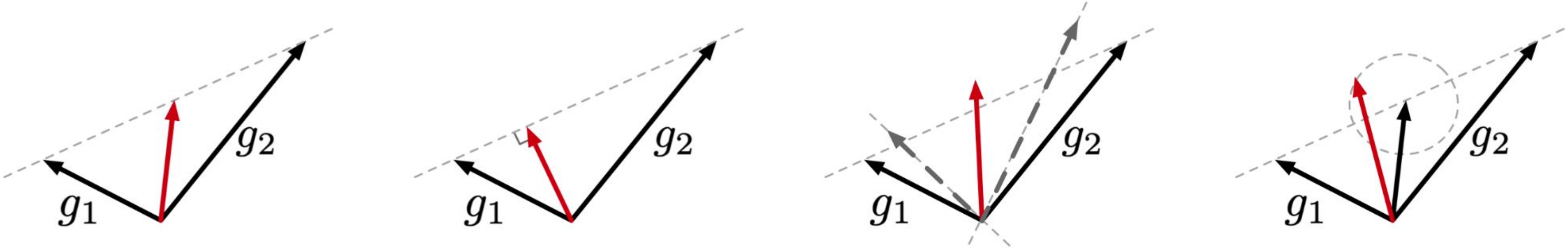
$$d = (g_{1\perp 2} + g_{2\perp 1})/2$$

$$\text{where } g_{i\perp j} = g_i - \frac{g_i^\top g_j}{\|g_j\|} g_j$$

CAGrad (ours)

$$\max_d \min_i g_i^\top d \quad \text{s.t.} \ \|d - g_0\| \leq c \, \|g_0\|$$

Conflict-Averse Gradient Descent (CAGrad)



GD | MGDA | PCGrad | CAGrad (ours)

GD:

$$d = (g_1 + g_2)/2$$

MGDA:

$$\max_d \min_i g_i^\top d \quad \text{s.t.} \quad \|d\| \le 1$$

PCGrad:

$$d = (g_{1\perp 2} + g_{2\perp 1})/2$$

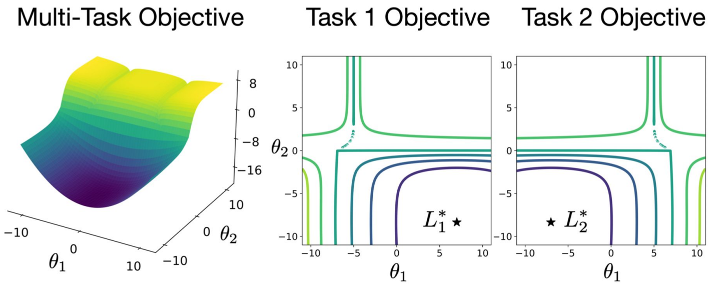
$$\text{where } g_{i\perp j} = g_i - \frac{g_i^\top g_j}{\|g_j\|} g_j$$

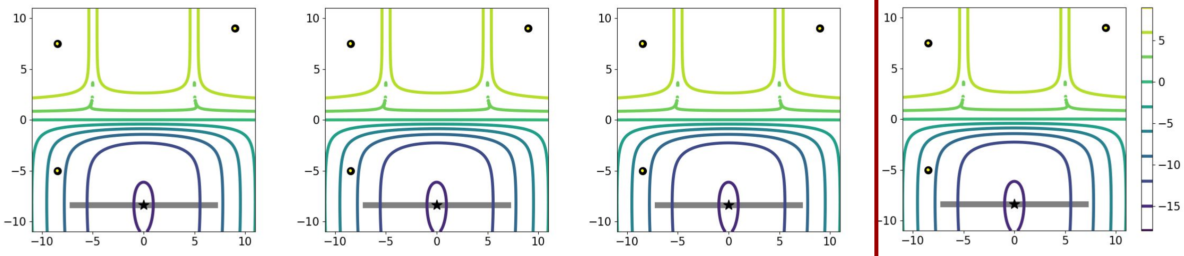
CAGrad (ours):

$$\max_d \min_i g_i^\top d \quad \text{s.t.} \ \|d - g_0\| \le \boxed{c} \|g_0\|$$

controls the radius
of the ball

Visualization of Optimization





GD

MGDA
[Sener et al., 2018]

PCGrad
[Yu et al., 2020]

CAGrad
(ours)

Convergence of CAGrad

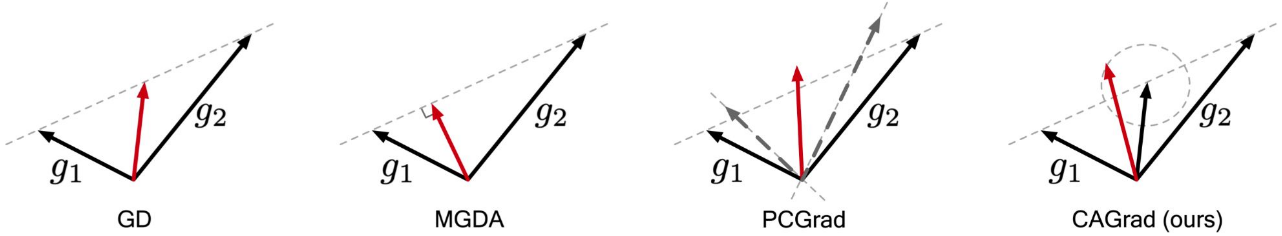
Convergence of CAGrad (Informal): With common differentiable and Lipschitz assumptions, we have:

1. If $0 \leq c < 1$, then CAGrad converges to an optimum of the average loss $L_0(\theta)$.
2. If $c \geq 1$, then CAGrad converges to a Pareto-optimal point.

Connection to GD and MGDA

In fact, CAGrad is *closely* connected to Gradient Descent (GD) and Multiple-Gradient Descent Algorithm (MGDA). Specifically:

1. When $c = 0$, CAGrad recovers GD.
2. When $c \to \infty$, CAGrad recovers MGDA.



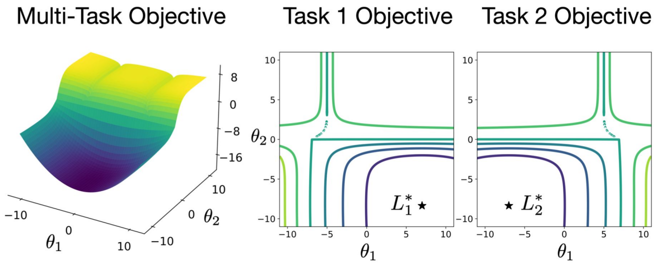
GD: $d = (g_1 + g_2)/2$

MGDA: $\max_d \min_i g_i^\top d$ s.t. $\|d\| \leq 1$

PCGrad: $d = (g_{1\perp 2} + g_{2\perp 1})/2$ where $g_{i\perp j} = g_i - \frac{g_i^\top g_j}{\|g_j\|} g_j$

CAGrad (ours): $\max_d \min_i g_i^\top d$ s.t. $\|d - g_0\| \leq c\,\|g_0\|$

Experiment (Toy Example)



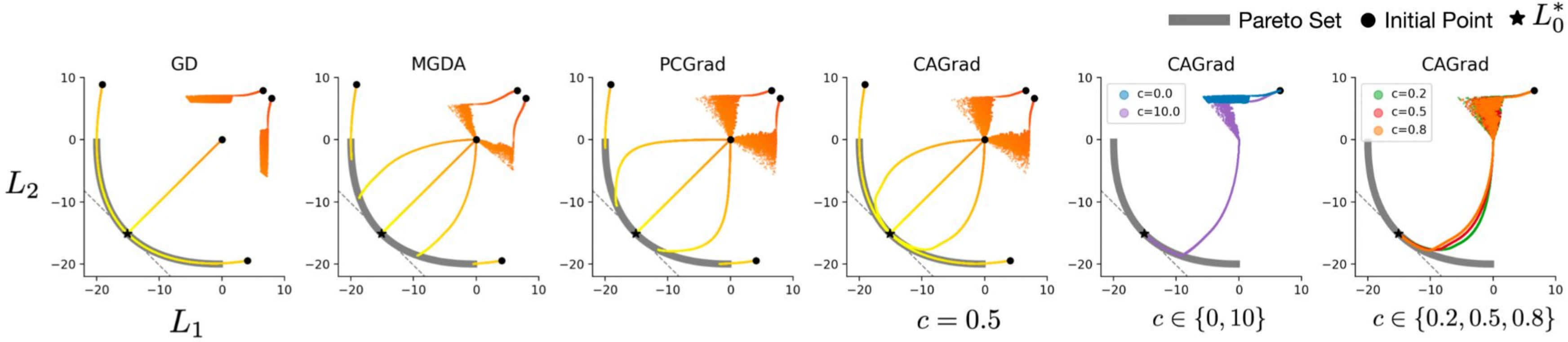


Figure 3: The left four plots are 5 runs of each algorithms from 5 different initial parameter vectors, where trajectories are colored from red to yellow. The right two plots are CAGrad's results with a varying $c \in \{0, 0.2, 0.5, 0.8, 10\}$.

Experiment (MultiMNIST)

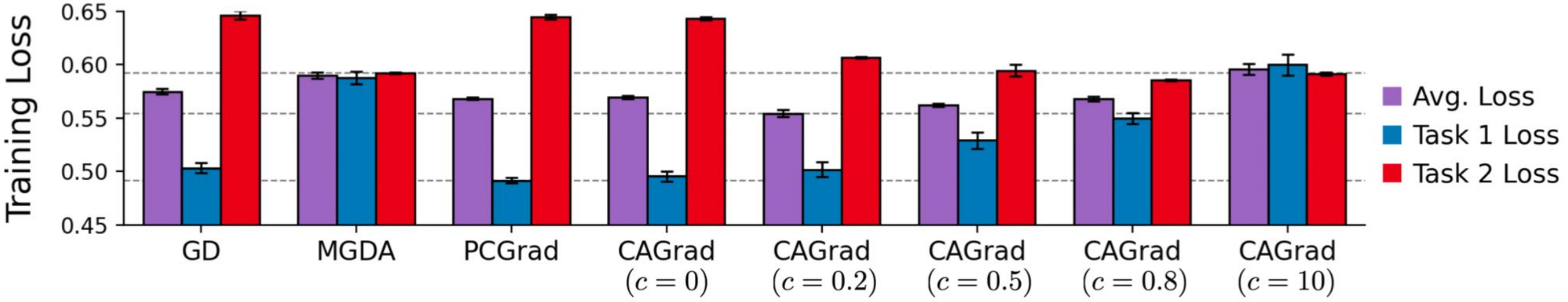


Figure 4: The average and individual training losses on the Fashion-and-MNIST benchmark by running GD, MGDA, PCGrad and CAGrad with different c values. GD gets stuck at the steep valley (the area with a cloud of dots), which other methods can pass. MGDA and PCGrad converge randomly on the Pareto set.

Experiment (NYU-v2)

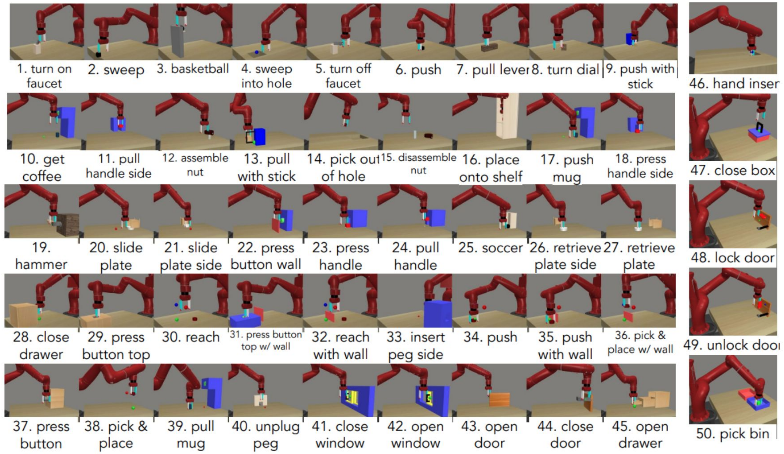
NYU-v2 consists of 3 vision tasks: **a)** 13-class semantic segmentation, **b)** depth prediction, and **c)** surface normal prediction.

#P.	Method	Segmentation (Higher Better)		Depth (Lower Better)		Surface Normal: Angle Distance (Lower Better)		Surface Normal: Within t° (Higher Better)			$\Delta m\% \downarrow$
		mIoU	Pix Acc	Abs Err	Rel Err	Mean	Median	11.25	22.5	30	
3	Independent	38.30	63.76	0.6754	0.2780	25.01	19.21	30.14	57.20	69.15	
≈3	Cross-Stitch [21]	37.42	63.51	0.5487	**0.2188**	*28.85	*24.52	*22.75	*46.58	*59.56	6.96
1.77	MTAN [3]	39.29	65.33	*0.5493	0.2263	*28.15	*23.96	*22.09	*47.50	*61.08	5.59
1.77	MGDA [26]	*30.47	*59.90	*0.6070	0.2555	**24.88**	**19.45**	**29.18**	**56.88**	**69.36**	1.38
1.77	PCGrad [37] (lr=1e-4)	38.06	*64.64	0.5550	0.2325	*27.41	*22.80	23.86	*49.83	*63.14	3.97
1.77	PCGrad [37] (lr=2e-4)	37.70	63.40	*0.5871	*0.2482	*28.18	*24.09	*21.94	*47.20	*60.87	8.12
1.77	GradDrop	39.39	65.12	*0.5455	0.2279	*27.48	*22.96	23.38	*49.44	*62.87	3.58
1.77	CAGrad (c=0.6)	**39.54**	**65.60**	**0.5340**	0.2199	25.87	20.94	25.88	53.78	67.00	**-1.37**

Table 1: Multi-task learning results on NYU-v2 dataset. $\#P$ denotes the relative model size compared to the vanilla SegNet. Each experiment is repeated over 3 random seeds and the mean is reported. The best average result among all multi-task methods is marked in bold. MGDA, PCGrad, GradDrop and CAGrad are applied on the MTAN backbone. CAGrad has statistically significant improvement over baselines methods with an $*$, tested with a p-value of 0.05.

Experiment (Multitask RL)

Test on the metaworld MTRL benchmark: metaworld-MT10 and metaworld-MT50, with 10 and 50 manipulation tasks.



Metaworld [Yu et al., 2020]

Method	Metaworld MT10 success (mean ± stderr)	Metaworld MT50 success (mean ± stderr)
Multi-task SAC [38]	0.49 ±0.073	0.36 ±0.013
Multi-task SAC + Task Encoder [38]	0.54 ±0.047	0.40 ±0.024
Multi-headed SAC [38]	0.61 ±0.036	0.45 ±0.064
PCGrad [37]	0.72 ±0.022	0.50 ±0.017
Soft Modularization [36]	0.73 ±0.043	0.50 ±0.035
CAGrad (ours)	**0.83** ±0.045	**0.52** ±0.023
CAGrad-Fast (ours)	0.82 ±0.039	0.50 ±0.016
CARE [29]	0.84 ±0.051	0.54 ±0.031
One SAC agent per task (upper bound)	0.90 ±0.032	0.74 ±0.041

Conflict-Averse Gradient Descent for Multitask Learning

Bo Liu | Xingchao Liu | Xiaojie Jin | Peter Stone | Qiang Liu

bliu@cs.utexas.edu

2021 Conference on Neural Information Processing Systems (NeurIPS)