NeurIPS | 2023

Thirty-seventh Conference on Neural Information Processing Systems

ForkMerge: Mitigating Negative Transfer in Auxiliary-Task Learning

Junguang Jiang[*], Baixu Chen[*], Junwei Pan[§], Ximei Wang[§], Dapeng Liu[§], Jie Jiang[§], Mingsheng Long[✉]

Junguang Jiang

Baixu Chen

Junwei Pan

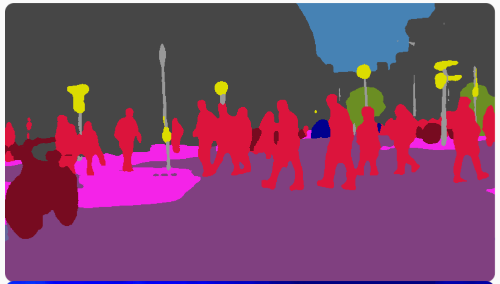
Ximei Wang

Mingsheng Long

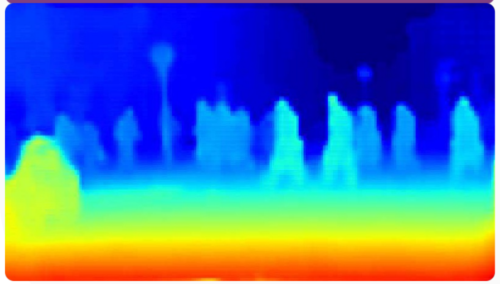
Auxiliary-Task Learning (ATL)

➢ *Aim to improve the performance of target tasks by leveraging the useful signals provided by related auxiliary tasks.*

Scene Understanding

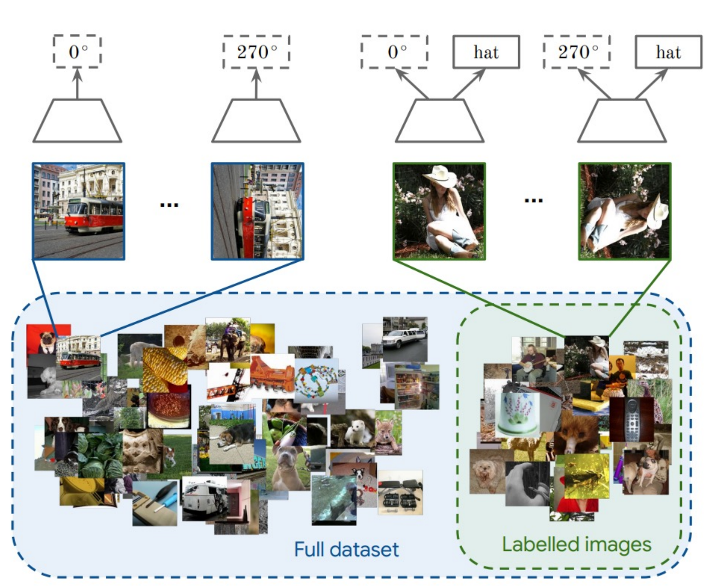


Target Task:
Semantic Segmentation



Auxiliary Task:
Depth Estimation

Semi-supervised Learning



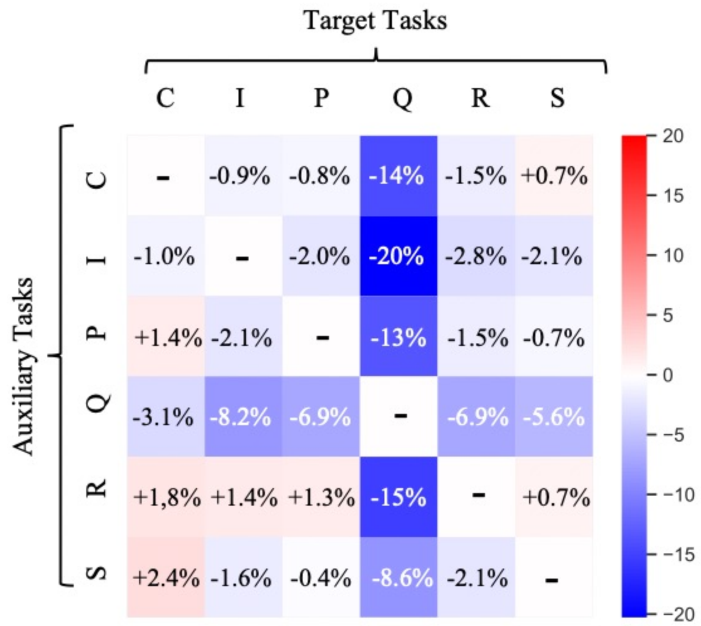
Target Task:
Classification

Auxiliary Task:
Rotation Prediction

[1] Kendall A, Gal Y, Cipolla R. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In CVPR 2018.
[2] Zhai X, Oliver A, Kolesnikov A, et al. S4l: Self-supervised semi-supervised learning. In ICCV 2019.

Negative Transfer in ATL

➢ *The widely existing phenomenon where the introduced auxiliary tasks lead to performance degradation.*



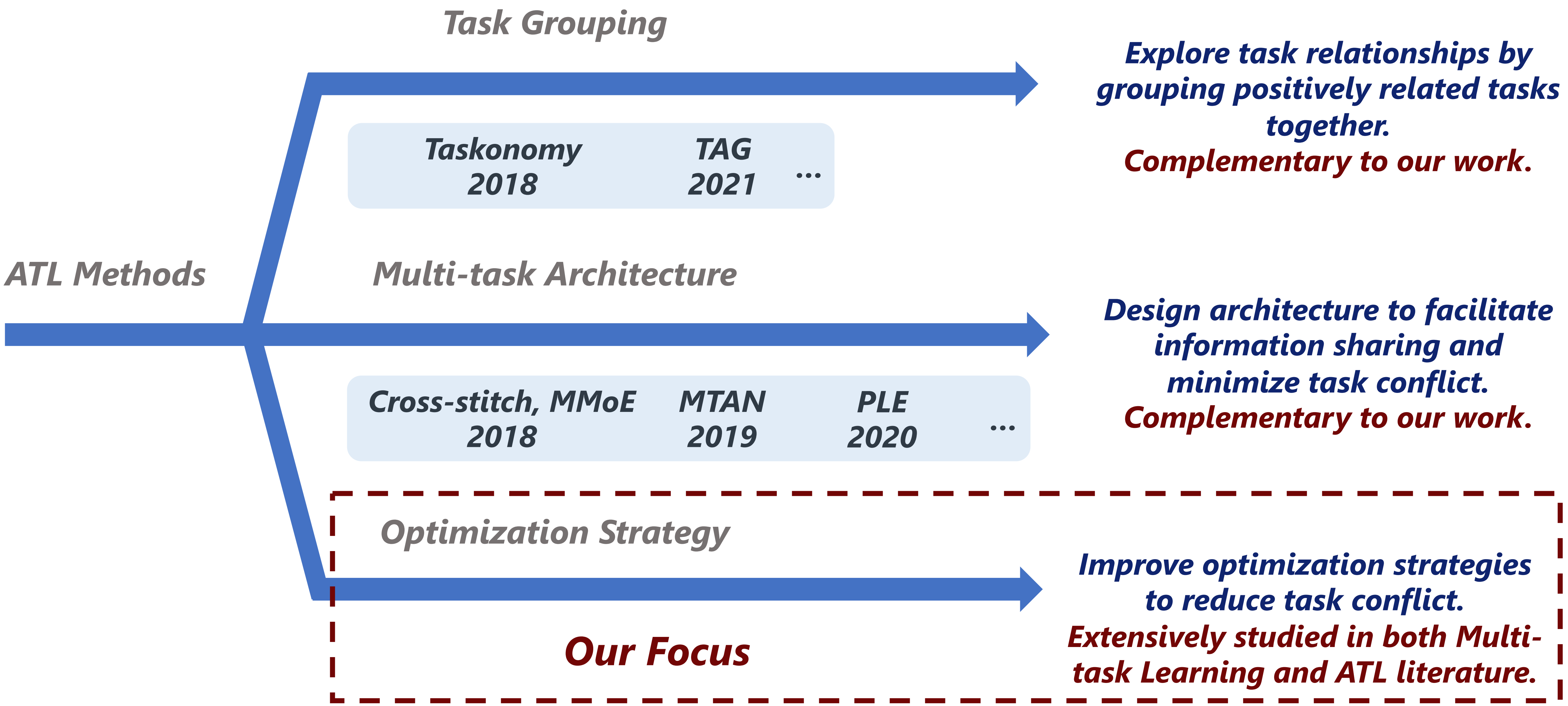
Pairwise transfer learning results on DomainNet. 23 of 30 combinations lead to negative transfer (blue cell).

Exam	Base model	RLHF model
LSAT (MCQ)	67.0 %	72.0 %
SAT EBRW – Reading Portion	92.3 %	90.4 %
SAT EBRW – Writing Portion	90.9 %	84.1 %
SAT Math (MCQ)	91.4 %	86.2 %
Graduate Record Examination (GRE) Quantitative	57.5 %	67.5 %
Graduate Record Examination (GRE) Verbal	87.5 %	90.0 %
USNCO Local Section Exam 2022	51.7 %	63.3 %
AP Art History (MCQ)	72.5 %	66.2 %
AP Biology (MCQ)	98.3 %	96.7 %
AP Calculus BC (MCQ)	66.7 %	57.8 %
AP Chemistry (MCQ)	58.3 %	71.7 %
AP English Language and Composition (MCQ)	55.6 %	51.1 %
AP English Literature and Composition (MCQ)	63.6 %	69.1 %
AP Environmental Science (MCQ)	72.5 %	67.5 %

Negative transfer (red item) when applying RLHF in GPT-4.

[1] OpenAI. Gpt-4 technical report, 2023

Overview of ATL Methods



Analysis on Negative Transfer

Problem Setup

- *Learning Objective in ATL*

$$\min_{\theta} \underbrace{\mathbb{E}_{\mathcal{T}_{\mathrm{tgt}}} \mathcal{L}_{\mathrm{tgt}}(\theta)}_{\textit{Target Task}} + \underbrace{\lambda \mathbb{E}_{\mathcal{T}_{\mathrm{aux}}} \mathcal{L}_{\mathrm{aux}}(\theta)}_{\textit{Auxiliary Task}}$$

- *$\mathcal{L}$ represents the loss function and λ is the relative weighting hyper-parameter.*

Analysis on Negative Transfer

Problem Setup

- *Transfer Gain*

$$TG(\lambda, \mathcal{A}) = \mathcal{P}\left(\underbrace{\theta_{\mathcal{A}}(\mathcal{T}_{tgt}, \mathcal{T}_{aux}, \lambda)}_{\text{Model obtained with ATL method } \mathcal{A}}\right) - \mathcal{P}\left(\underbrace{\theta(\mathcal{T}_{tgt})}_{\text{STL Model}}\right)$$

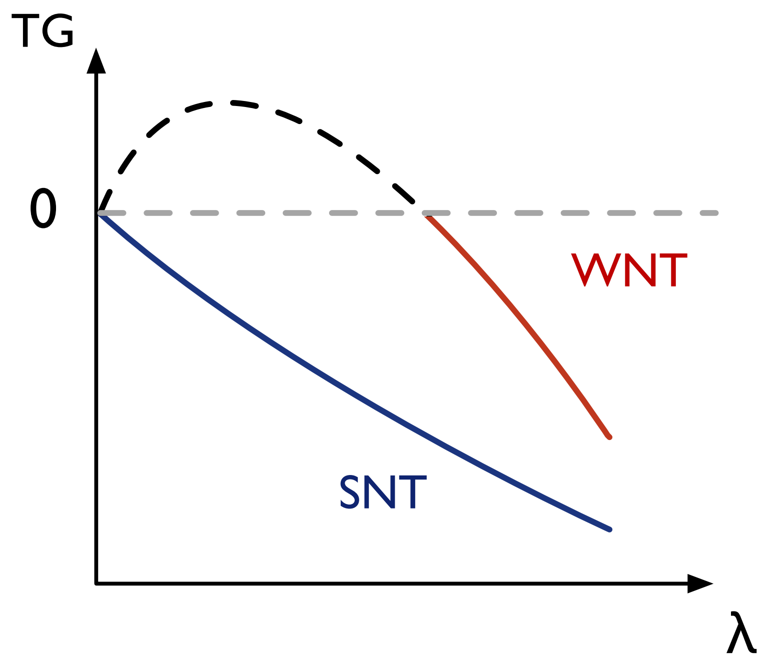
- *$\mathcal{P}$ represents the relative performance measure, where a higher $\mathcal{P}$ indicates better performance.*

Analysis on Negative Transfer

Problem Setup

- *Weak Negative Transfer*
  - *For some ATL algorithm $\mathcal{A}$ with weighting hyper-parameter λ , weak negative transfer occurs if $TG(\lambda, \mathcal{A}) < 0$.*

- *Strong Negative Transfer*
  - *For some ATL algorithm $\mathcal{A}$, strong negative transfer occurs if $\max_{\lambda>0} TG(\lambda, \mathcal{A}) < 0$.*

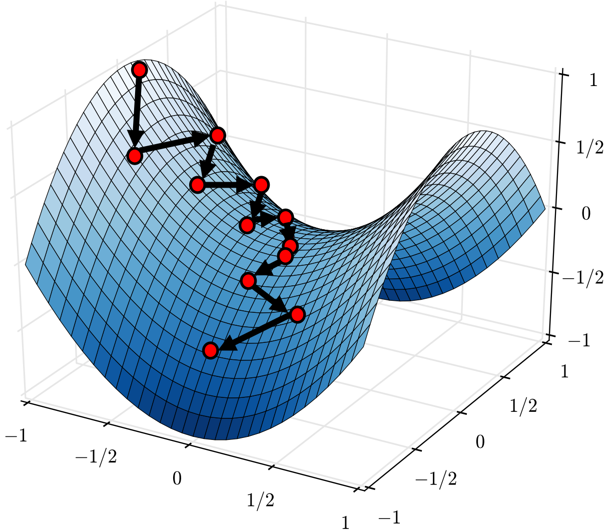


Analysis on Negative Transfer

Effect of Gradient Conflicts

➢ *At each optimization step t, we have*

$$\theta_{t+1}(\lambda) = \theta_t - \eta(\underbrace{g_{tgt}(\theta_t)}_{\textbf{Gradient of Target Task}} + \underbrace{\lambda g_{aux}(\theta_t)}_{\textbf{Gradient of Auxiliary Task}})$$



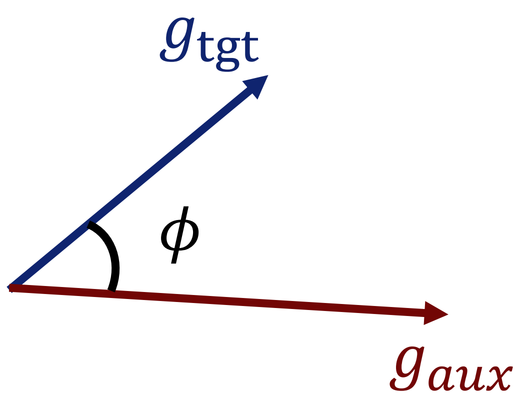
Analysis on Negative Transfer

Effect of Gradient Conflicts

- *It is widely believed the gradient conflict between $\boldsymbol{g}_{tgt}$ and $\boldsymbol{g}_{aux}$ will lead to negative transfer, where the degree of conflict is measured by Gradient Cosine Similarity.*

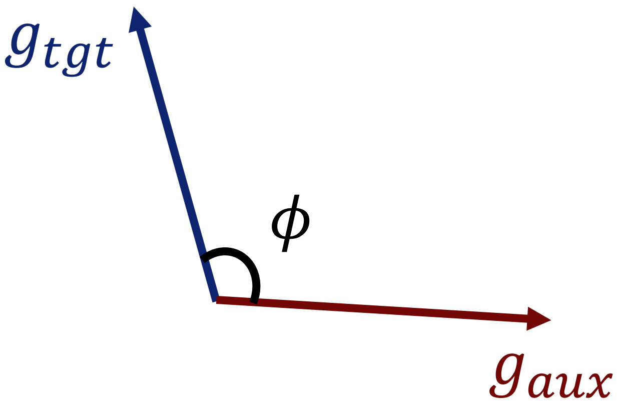
Gradient Cosine Similarity

$$\cos\phi$$



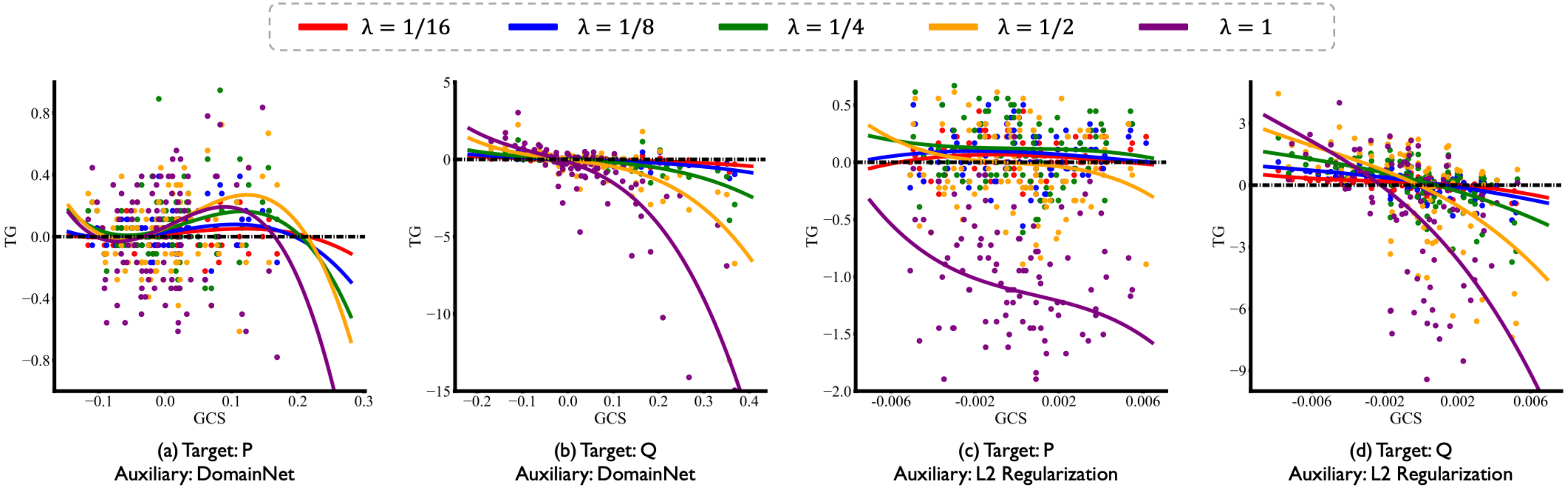
Gradient Conflict Occurs if

$$\cos\phi < 0$$



Analysis on Negative Transfer

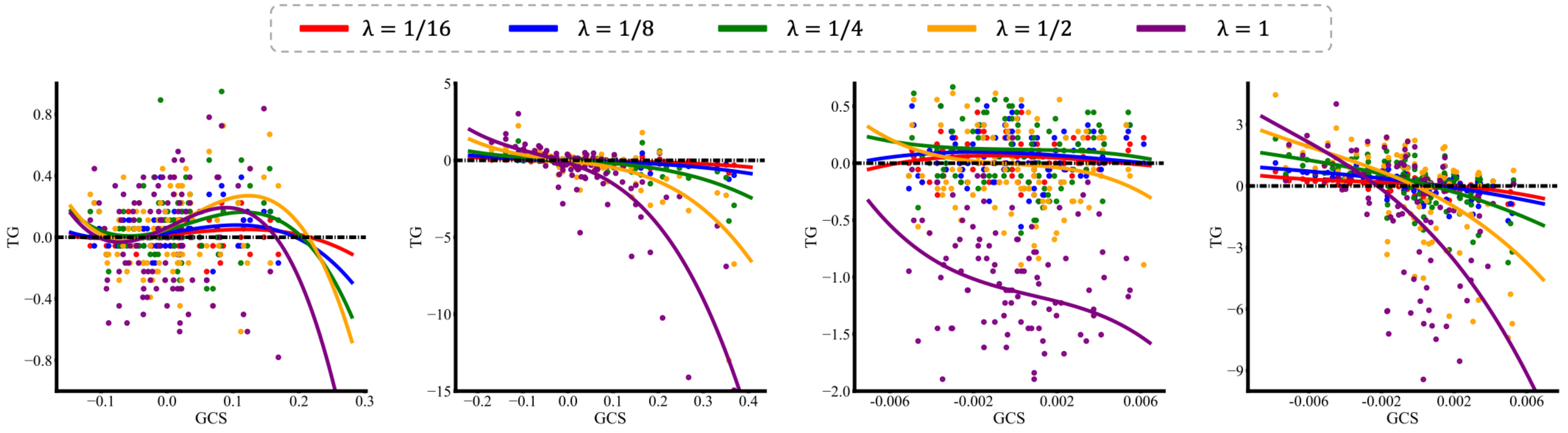
Effect of Gradient Conflicts



The correlation curve between Transfer Gain (TG) and Gradient Cosine Similarity (GCS) under different λ.

Analysis on Negative Transfer

Effect of Gradient Conflicts



(a) Target: P
Auxiliary: DomainNet

(b) Target: Q
Auxiliary: DomainNet

(c) Target: P
Auxiliary: L2 Regularization

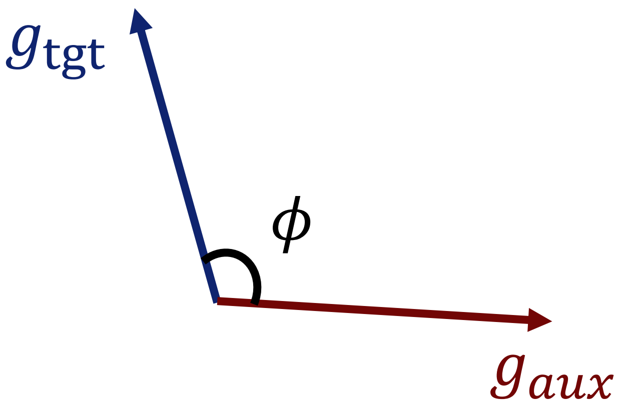
(d) Target: Q
Auxiliary: L2 Regularization

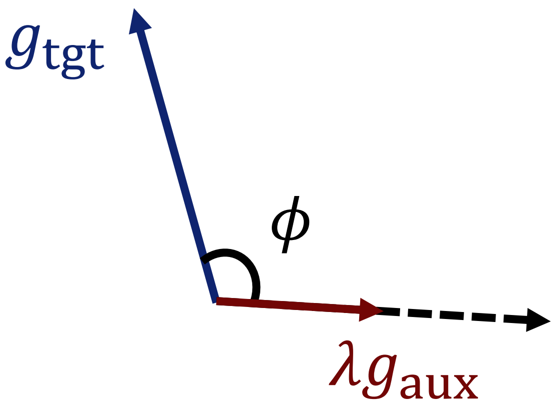
***[Observation 1]** Negative transfer is **not necessarily** caused by gradient conflicts and gradient conflicts do **not necessarily** lead to negative transfer.*

Analysis on Negative Transfer

Effect of Gradient Conflicts

- *Moreover, it can be observed that the weighting hyper-parameter λ in ATL has a large impact on negative transfer.*
  - *Changing λ will not influence the gradient cosine similarity.*





Analysis on Negative Transfer

Effect of Distribution Shift

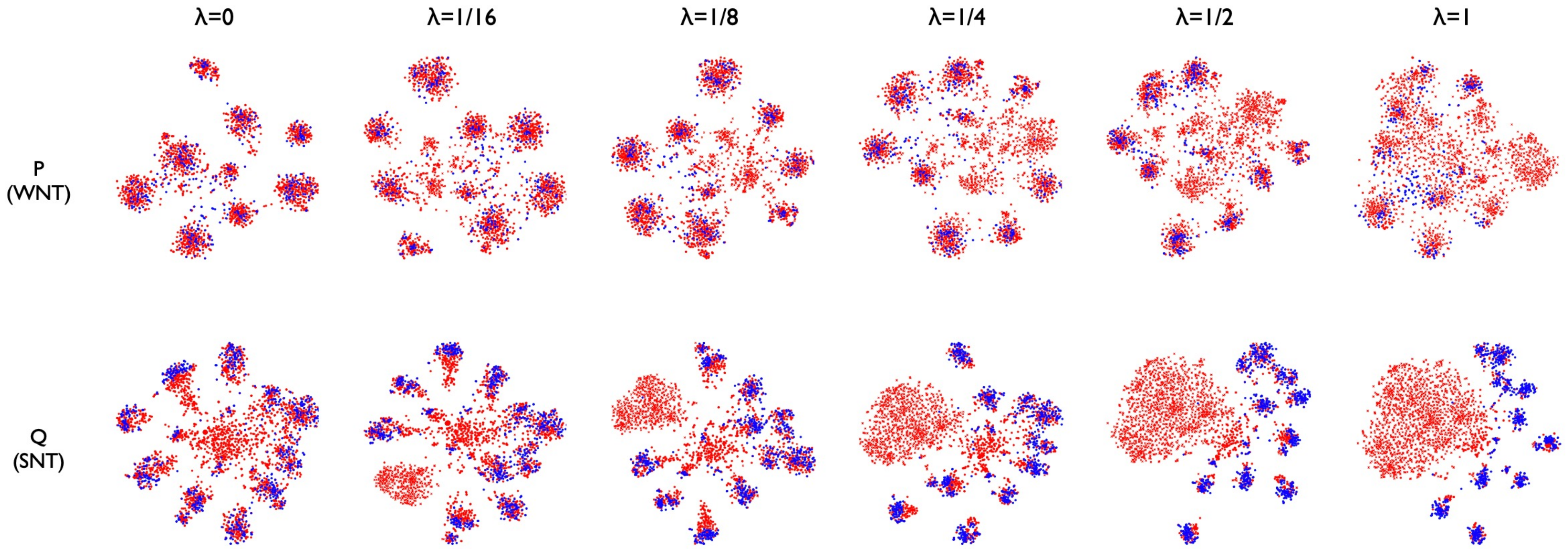
- *Adjusting λ will change the data distribution that the model is fitting.*
  - *Formula for Interpolated Distribution*

$$\mathcal{T}_{inter} \sim \underbrace{(1 - Z)\mathcal{T}_{tgt}}_{\textbf{Target Task}} + \underbrace{Z\mathcal{T}_{aux}}_{\textbf{Auxiliary Task}} \qquad \underbrace{Z \sim Bernoulli\left(\frac{\lambda}{1+\lambda}\right)}_{\textbf{Bernoulli Distribution}}$$

Analysis on Negative Transfer

Effect of Distribution Shift

➢ *Qualitative Measurement - t-SNE*



t-SNE visualization of interpolated training distribution and target task test distribution with different λ.

Analysis on Negative Transfer

Effect of Distribution Shift

- *Quantitative Measurement - Confidence Score Discrepancy (CSD)*

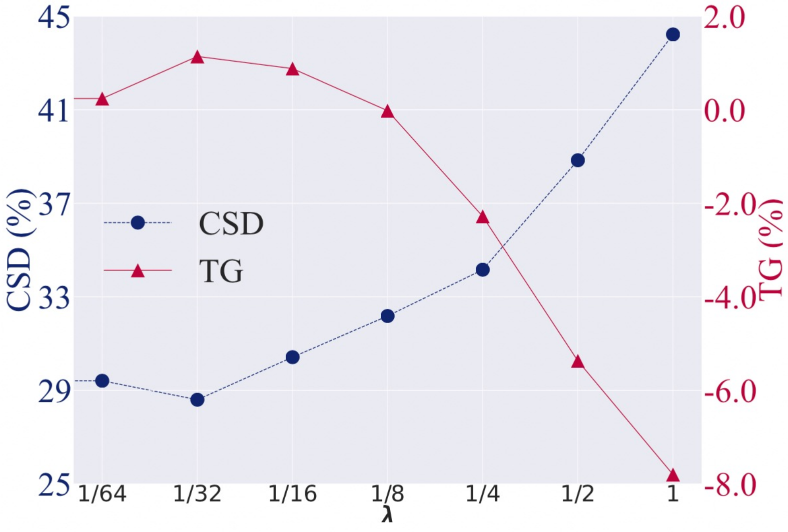
$$d_{\mathcal{F}}(D, D') \triangleq 1 - \underbrace{\mathbb{E}_{x\sim D'}}_{\textbf{Data from } D'} \max_{y\in\mathcal{Y}} \underbrace{f_D^*}_{\textbf{Optimal Scoring Function on } D}(x, y)$$

- *Confidence score discrepancy indicates how unconfident the model is, which is expected to increase when the data shift enlarges.*

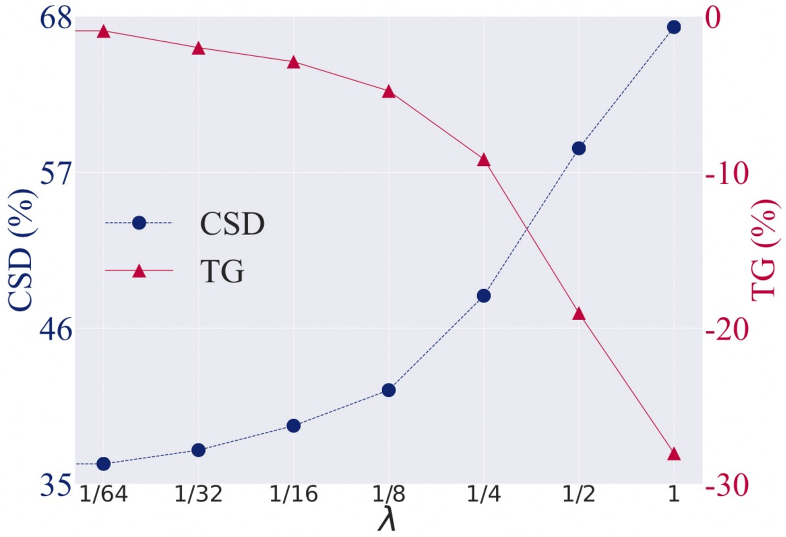
Analysis on Negative Transfer

Effect of Distribution Shift

➢ *Quantitative Measurement - Confidence Score Discrepancy (CSD)*



P (WNT)

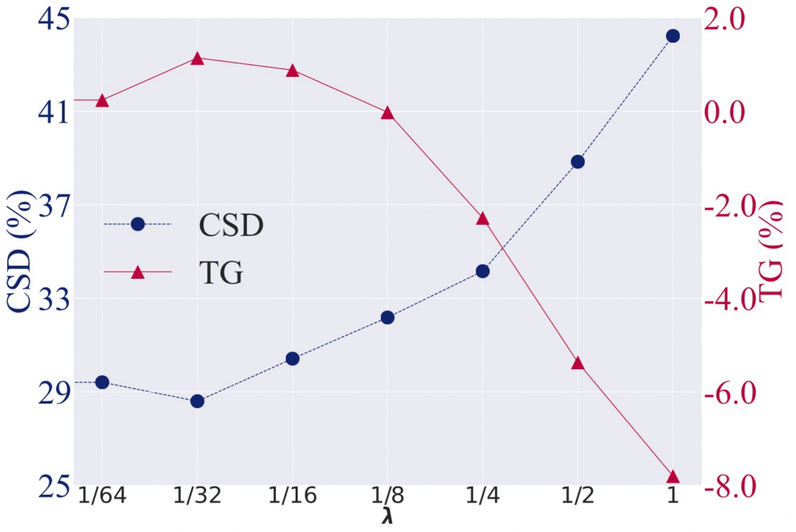


Q (SNT)

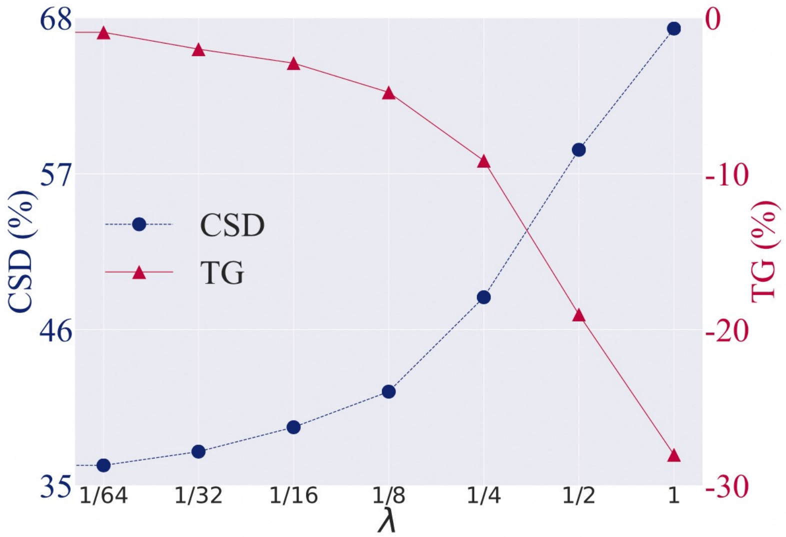
The correlation curve between Transfer Gain (TG) and Confidence Score Discrepancy (CSD) with different λ.

Analysis on Negative Transfer

Effect of Distribution Shift



P (WNT)



Q (SNT)

[Observation 2] Negative transfer is likely to occur if the introduced auxiliary task enlarges the distribution shift between training and test data for the target task.

ForkMerge Algorithm

Motivation

$$\theta_{t+1}(\lambda) = \theta_t - \eta(\underbrace{g_{tgt}(\theta_t)}_{\text{Gradient of Target Task}} + \underbrace{\lambda g_{aux}(\theta_t)}_{\text{Gradient of Auxiliary Task}})$$

- *[Optimization View] The gradient conflict between g_{tgt} and g_{aux} does not necessarily lead to negative transfer.*
- *[Algorithm Design] Unlike prior works, our algorithm does not aim to directly resolve gradient conflicts.*

ForkMerge Algorithm

Motivation

- *[Generalization View] Different λ will lead to diverse distribution shift, resulting in different generalization performance.*
- *[Algorithm Design] In ForkMerge, we will dynamically adjust λ based on the generalization performance on the validation set.*

ForkMerge Algorithm

Algorithm

- *Denote $\hat{\mathcal{P}}$ the performance measure on the validation set, the learning process can be formulated as a bi-level optimization problem.*

$$\lambda^* = \underset{\lambda}{\operatorname{argmax}} \hat{\mathcal{P}}(\theta_{t+1}) = \hat{\mathcal{P}}(\theta_t - \eta(g_{\mathrm{tgt}}(\theta_t) + \lambda g_{\mathrm{aux}}(\theta_t)))$$

ForkMerge Algorithm

Algorithm

$$\lambda^* = \arg\max_{\lambda} \hat{\mathcal{P}}(\theta_{t+1}) = \hat{\mathcal{P}}(\theta_t - \eta(g_{\text{tgt}}(\theta_t) + \lambda g_{\text{aux}}(\theta_t)))$$

- *Existing methods usually approximate $\hat{\mathcal{P}}$ with the loss of a batch of data, and then use first-order approximation to update λ (e.g. use Meta Learning).*

ForkMerge Algorithm

Algorithm

$$\lambda^* = \underset{\lambda}{\mathrm{argmax}} \hat{\mathcal{P}}(\theta_{t+1}) = \hat{\mathcal{P}}(\theta_t - \eta(g_{\mathrm{tgt}}(\theta_t) + \lambda g_{\mathrm{aux}}(\theta_t)))$$

- *Existing methods usually approximate $\hat{\mathcal{P}}$ with the loss of a batch of data, and then use first-order approximation to update λ (e.g. use Meta Learning).*

- *However, these approximations within a single step of gradient descent (1) introduce large noise to the estimation of λ and also (2) increase the risk of over-fitting the validation set.*

ForkMerge Algorithm

Algorithm

$$\lambda^* = \underset{\lambda}{\operatorname{argmax}} \hat{\mathcal{P}}(\theta_{t+1}) = \hat{\mathcal{P}}(\theta_t - \eta(g_{\mathrm{tgt}}(\theta_t) + \lambda g_{\mathrm{aux}}(\theta_t)))$$

$$= \underset{\lambda}{\operatorname{argmax}} \hat{\mathcal{P}}((\theta_t - \eta g_{\mathrm{tgt}}(\theta_t)) + \lambda(-\eta g_{\mathrm{aux}}(\theta_t)))$$

$$= \underset{\lambda}{\operatorname{argmax}} \hat{\mathcal{P}}((1-\lambda)(\theta_t - \eta g_{\mathrm{tgt}}(\theta_t)) + \lambda(\theta_t - \eta(g_{\mathrm{tgt}}(\theta_t) + g_{\mathrm{aux}}(\theta_t))))$$

$$\boxed{= \underset{\lambda}{\operatorname{argmax}} \hat{\mathcal{P}}((1-\lambda)\theta_{t+1}(0) + \lambda(\theta_{t+1}(1))} \quad \textit{// gradient descent}$$

➢ *By derivation, we obtain the equivalent optimization objective based on the interpolation of model parameters.*

ForkMerge Algorithm

Algorithm

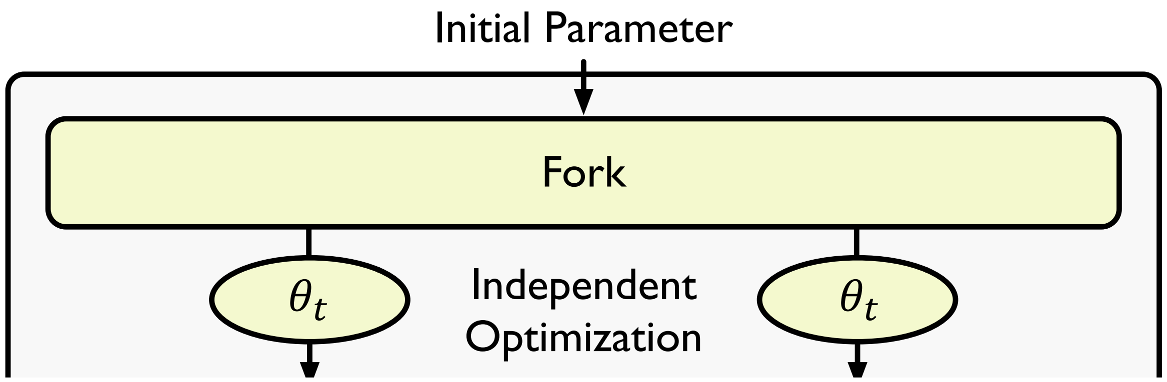
$$\lambda^* = \underset{\lambda}{\operatorname{argmax}} \hat{\mathcal{P}}((1-\lambda)\theta_{t+1}(0) + \lambda(\theta_{t+1}(1))$$

- *An accurate estimation in the above equation is computationally expensive and prone to over-fit. Thus, we extend the one gradient step to Δt steps.*

$$\boxed{\lambda^* = \underset{\lambda}{\operatorname{argmax}} \hat{\mathcal{P}}((1-\lambda)\theta_{t+\Delta t}(0) + \lambda(\theta_{t+\Delta t}(1))}$$

ForkMerge Algorithm

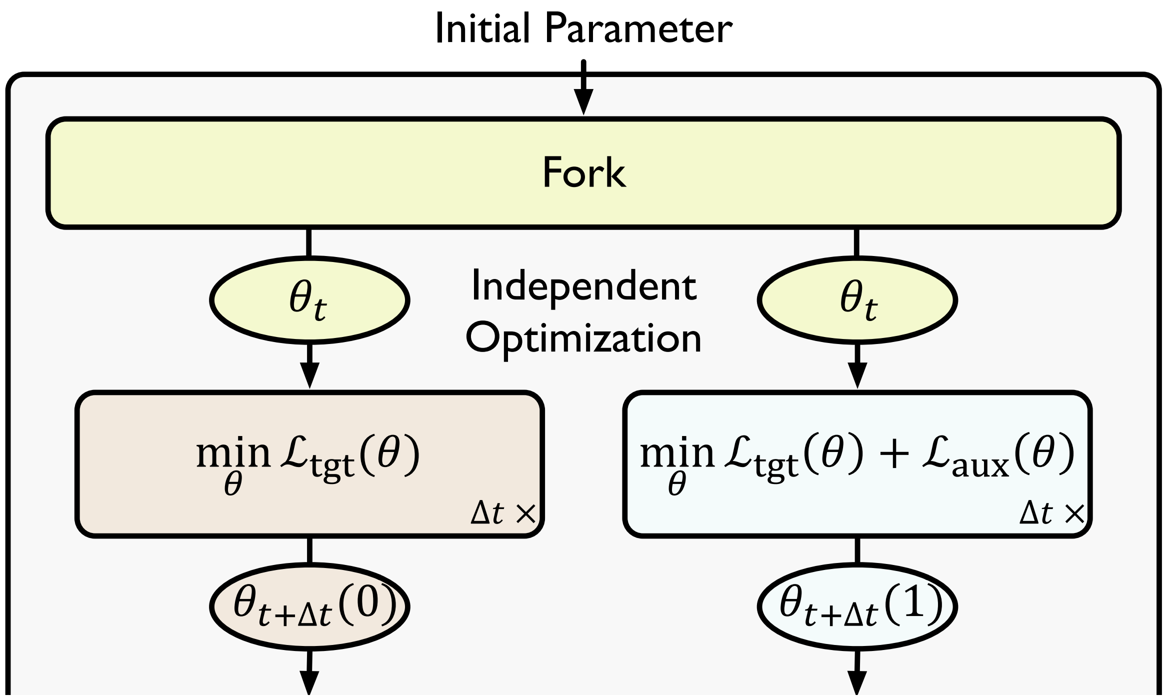
Algorithm



(1) Fork. *The initial model will be copied into two independent branches with the same parameters.*

ForkMerge Algorithm

Algorithm



(2) Optimize. *The first branch is only optimized with the target task loss. While the second branch is jointly optimized. Train for Δt steps.*

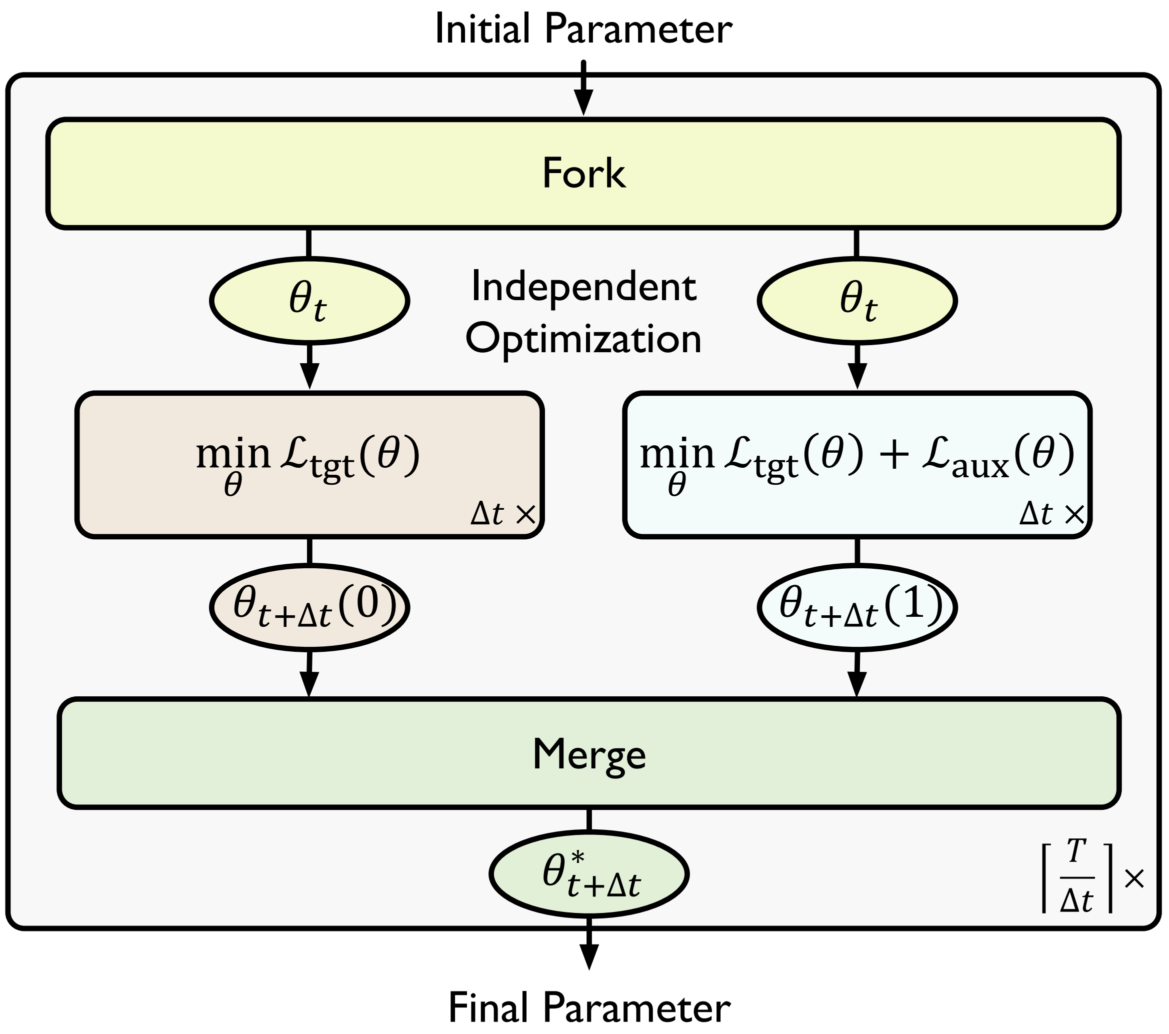
ForkMerge Algorithm

Algorithm

(3) Merge. *Search for the optimal λ^* that linearly combines two sets of parameters to maximize the validation performance.*

$$\lambda^* = \underset{\lambda}{\operatorname{argmax}} \widehat{\mathcal{P}}((1-\lambda)\theta_{t+\Delta t}(0) + \lambda(\theta_{t+\Delta t}(1))$$

Overall, the [Fork -> Optimize -> Merge] loop is iterated for $\left\lceil \frac{T}{\Delta t} \right\rceil$ times.



ForkMerge Algorithm

Extension to Multiple Auxiliary Tasks

➢ *Learning Objective*

$$\min_{\theta} \underbrace{\lambda_0 \mathbb{E}_{\mathcal{T}_0} \mathcal{L}_0(\theta)}_{\textit{Target Task}} + \sum_{k=1}^{K} \underbrace{\lambda_k \mathbb{E}_{\mathcal{T}_k} \mathcal{L}_k(\theta)}_{\textit{Auxiliary Tasks}}, \sum_{k=1}^{K} \lambda_k \leq 1$$

➢ *Similar to the case where there is only one auxiliary task, we outline the following equivalent objective.*

ForkMerge Algorithm

Extension to Multiple Auxiliary Tasks

➢ *Equivalent Objective*

$$\omega_i^k = \mathbb{I}[i = k \text{ or } i = 0]$$

$$\Lambda_k = \begin{cases} 1 - \sum_{i \neq 0} \lambda_i \,, & k = 0 \\ \lambda_k \,, & k \neq 0 \end{cases}$$

$$\Lambda^* = \underset{\Lambda}{\operatorname{argmax}} \hat{\mathcal{P}} \left(\sum_{k=0}^{K} \Lambda_k \, \theta_{t+1}(\omega^k) \right)$$

➢ *The first branch is only optimized with the target task loss.*

➢ *For other branches, each is jointly optimized with the target task and the k-th auxiliary task.*

ForkMerge Algorithm

Extension to Multiple Auxiliary Tasks

- *Equivalent Objective*

$$\omega_i^k = \mathbb{I}[i = k \text{ or } i = 0]$$

$$\Lambda_k = \begin{cases} 1 - \sum_{i \neq 0} \lambda_i \,, & k = 0 \\ \lambda_k \,, & k \neq 0 \end{cases}$$

$$\Lambda^* = \underset{\Lambda}{\operatorname{argmax}} \hat{\mathcal{P}} \left(\sum_{k=0}^{K} \Lambda_k \, \theta_{t+1}(\omega^k) \right)$$

- *Search for the optimal Λ^* that linearly combines the $K + 1$ sets of parameters to maximize the validation performance.*

ForkMerge Algorithm

General Form

$$\bar{\Lambda}^* = \underset{\bar{\Lambda}}{\operatorname{argmax}} \hat{\mathcal{P}} \left(\sum_{b=1}^{B} \bar{\Lambda}_b \, \theta_{t+\Delta t}(\nu^b) \right)$$

- *The general form has no constraints on the number of branches B and the task weighting vector ν^b.*

ForkMerge Algorithm

General Form

$$\bar{\Lambda}^* = \underset{\bar{\Lambda}}{\operatorname{argmax}} \hat{\mathcal{P}} \left(\sum_{b=1}^{B} \bar{\Lambda}_b \, \theta_{t+\Delta t}(\nu^b) \right)$$

- *The general form has no constraints on the number of branches B and the task weighting vector ν^b.*
  - *Allow us to introduce human prior into ForkMerge by constructing more efficient branches.*

ForkMerge Algorithm

General Form

$$\overline{\Lambda}^* = \underset{\overline{\Lambda}}{\operatorname{argmax}} \hat{\mathcal{P}} \left(\sum_{b=1}^{B} \overline{\Lambda}_b \, \theta_{t+\Delta t}(\nu^b) \right)$$

- *The general form has no constraints on the number of branches B and the task weighting vector ν^b.*
  - *Allow us to introduce human prior into ForkMerge by constructing more efficient branches.*
  - *Provide possibilities for combining ForkMerge with previous task grouping methods.*

ForkMerge Algorithm

Discussion on Computation Cost

$$\overline{\Lambda}^* = \underset{\overline{\Lambda}}{\operatorname{argmax}} \hat{\mathcal{P}} \left(\sum_{b=1}^{B} \overline{\Lambda}_b \, \theta_{t+\Delta t}(\nu^b) \right)$$

- *The choice of the B implies a trade-off between performance and efficiency. In practice, users may tailor B to align with their computational resources.*

- *We have also developed several techniques to reduce computation cost such as the pruning strategy and the greedy merging strategy. Please refer to our paper for details.*

Experiments

Main Results

- *ForkMerge consistently achieves state-of-the-art performance across 4 benchmarks, including:*

- *Auxiliary-Task Scene Understanding (+4.03% v.s. previous SOTA +2.10%).*

- *Auxiliary-Domain Image Recognition (+2.00% over STL, while most existing methods fail to improve performance).*

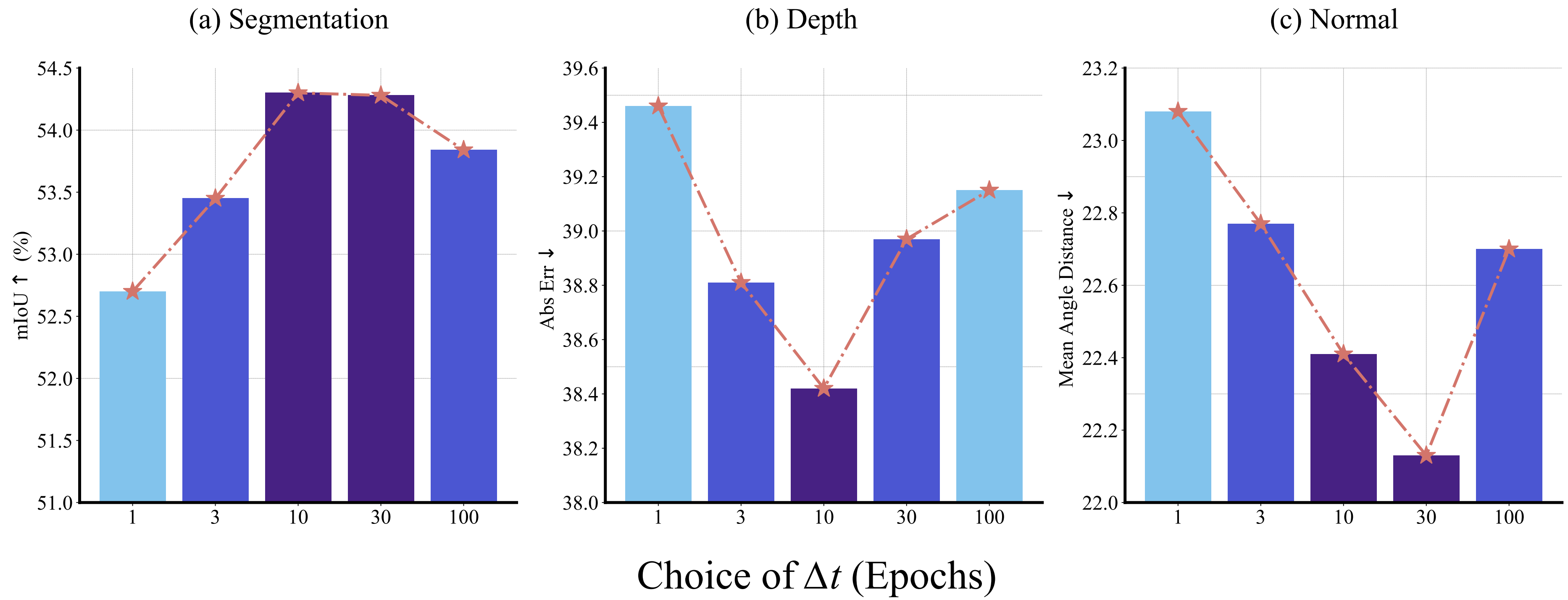
- *CTR and CTCVR Prediction (+1.30% v.s. previous SOTA +0.55%).*

- *Semi-Supervised Learning (SSL) (+46.3% v.s. previous SOTA + 43.2%).*

Experiments

Analysis Experiments

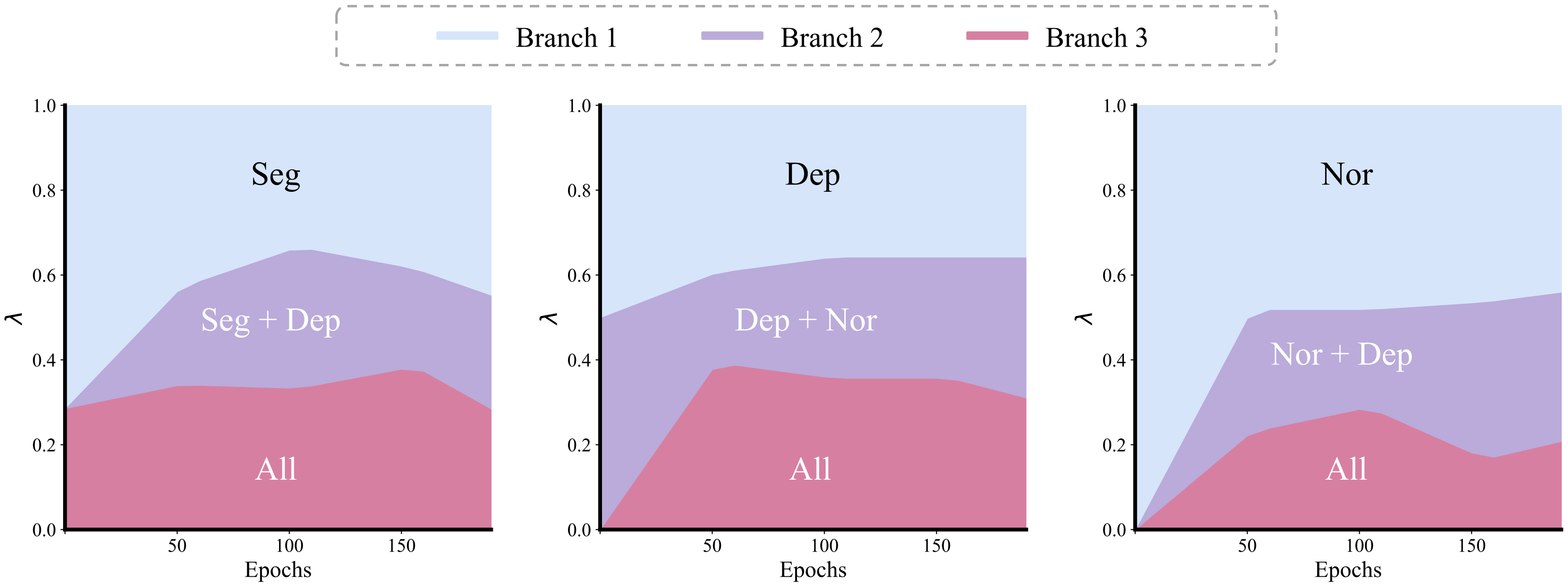
➢ *Effect of the merging step Δt.*



Experiments

Analysis Experiments

➢ *Importance of different forking branches during training.*



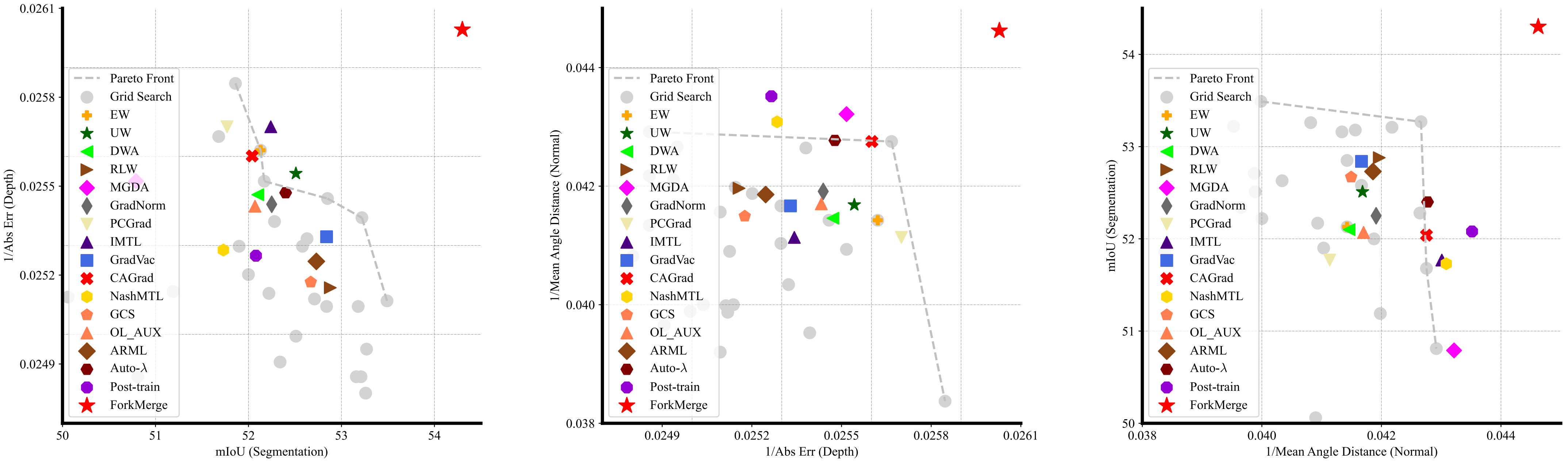
(a) Segmentation (b) Depth (c) Normal

Experiments

➢ *Please refer to our paper for more analysis.*

Analysis Experiments

➢ *Comparison with grid searching (top-right is better).*



Thank You!

JiangJunguang1123@outlook.com
cbx_99_hasta@outlook.com

长按关注，获取最新资讯