Supplementary Materials of Multi-Task Parameter Passing Networks

Anonymous Author(s)
Affiliation
Address
email

A Proofs

For better conciseness, we will simply the model $f_{\boldsymbol{w}}(\mathbb{D})$ as $f_{\boldsymbol{w}}$, by only specifying the the parameter vector $\boldsymbol{w}$ without the explicit involvement of input data $\mathbb{D}$.

We first have the following definition of L-smooth function.

Definition 1. *The function f is called L-smooth iff we have the following inequation for any two parameter vectors $\boldsymbol{a}$ and $\boldsymbol{b}$,*

$$f_{\boldsymbol{b}} \leq f_{\boldsymbol{a}} + (\boldsymbol{b} - \boldsymbol{a})^\top \nabla f_{\boldsymbol{a}} + \frac{L}{2}\|\boldsymbol{b} - \boldsymbol{a}\|^2. \tag{5}$$

Theorem 1. *Suppose f is L-smooth, the expected gradient over data satisfies $\mathbb{E}[\|\nabla f_{\boldsymbol{w}}\|^2] \leq \sigma^2$, the initial and the minimum point function values of the i-th task are given by $f_i^{(0)}$ and f_i^*, respectively. Besides, Algorithm 1 applies one-step SGD to update $\boldsymbol{w}_i$ with the learning rate λ, and the update by adjacent-learning is valid: $f_{\boldsymbol{w}'_i} \leq f_{\boldsymbol{w}_i}$. Then, we have the following convergence:*

$$\min_{t=0,\cdots,T-1} \sum_{i=1}^{M} \mathbb{E}[\|\nabla f_{\boldsymbol{w}_i^{(t)}}\|^2] \leq \sum_{i=1}^{M} \frac{f_i^{(0)} - f_i^*}{T\lambda} + \frac{LM\sigma^2\lambda}{2}. \tag{6}$$

Proof. We first illustrate the computation flow for each iteration in Algorithm 1:

$$\boldsymbol{w}_i^{(t)} \xrightarrow{\text{Self-Learn}} \boldsymbol{w}_i^{(t+\frac{1}{2})} \overset{\text{Passing}}{=} \boldsymbol{w}_i'^{(t)} \xrightarrow{\text{Adj-Learn}} \boldsymbol{w}_i'^{(t+\frac{1}{2})} \overset{\text{Set-Back}}{=} \boldsymbol{w}_i^{(t+1)}. \tag{7}$$

We now discuss self-learning and adjacent-learning, respectively.

1. For self-learning, we apply one-step SGD on sampling data x, which implies $\boldsymbol{w}_i^{(t+\frac{1}{2})} = \boldsymbol{w}_i^{(t)} - \lambda \nabla f_{\boldsymbol{w}_i^{(t)}}(x)$. Then, by adopting the L-smooth definition in (5), we attain:

$$
\begin{aligned}
f_{\boldsymbol{w}_i^{(t+\frac{1}{2})}} &\leq f_{\boldsymbol{w}_i^{(t)}} + (\boldsymbol{w}_i^{(t+\frac{1}{2})} - \boldsymbol{w}_i^{(t)})^\top \nabla f_{\boldsymbol{w}_i^{(t)}} + \frac{L}{2}\|\boldsymbol{w}_i^{(t+\frac{1}{2})} - \boldsymbol{w}_i^{(t)}\|^2, \\
\overset{\text{Expectation over } x}{\Longrightarrow} \mathbb{E}[f_{\boldsymbol{w}_i^{(t+\frac{1}{2})}}] &\leq f_{\boldsymbol{w}_i^{(t)}} - \lambda\|\nabla f_{\boldsymbol{w}_i^{(t)}}\|^2 + \frac{\lambda^2 L}{2}\mathbb{E}[\|\nabla f_{\boldsymbol{w}_i^{(t)}}\|^2], \\
&\leq f_{\boldsymbol{w}_i^{(t)}} - \lambda\|\nabla f_{\boldsymbol{w}_i^{(t)}}\|^2 + \frac{\lambda^2\sigma^2 L}{2}, \\
\Rightarrow \|\nabla f_{\boldsymbol{w}_i^{(t)}}\|^2 &\leq \frac{1}{\lambda}(f_{\boldsymbol{w}_i^{(t)}} - \mathbb{E}[f_{\boldsymbol{w}_i^{(t+\frac{1}{2})}}]) + \frac{L\sigma^2\lambda}{2}, \\
\overset{\text{Expectation over } \boldsymbol{w}_i^{(t)}}{\Longrightarrow} \mathbb{E}[\|\nabla f_{\boldsymbol{w}_i^{(t)}}\|^2] &\leq \frac{1}{\lambda}(\mathbb{E}[f_{\boldsymbol{w}_i^{(t)}}] - \mathbb{E}[f_{\boldsymbol{w}_i^{(t+\frac{1}{2})}}]) + \frac{L\sigma^2\lambda}{2}, \\
\overset{\text{Summation over } t}{\Longrightarrow} \sum_{t=0}^{T-1}\mathbb{E}[\|\nabla f_{\boldsymbol{w}_i^{(t)}}\|^2] &\leq \frac{1}{\lambda}\sum_{t=0}^{T-1}(\mathbb{E}[f_{\boldsymbol{w}_i^{(t)}}] - \mathbb{E}[f_{\boldsymbol{w}_i^{(t+\frac{1}{2})}}]) + \frac{L\sigma^2\lambda T}{2}.
\end{aligned}
\tag{8}
$$

2. For adjacent-learning, the flow in Eq. (7) derives:

$$
f_{\boldsymbol{w}_i^{(t+1)}} = f_{\boldsymbol{w}_i'^{(t+\frac{1}{2})}} \leq f_{\boldsymbol{w}_i'^{(t)}} = f_{\boldsymbol{w}_i^{(t+\frac{1}{2})}}, \tag{9}
$$

where we have assumed the valid update in adjacent-learning $f_{\boldsymbol{w}_i'^{(t+\frac{1}{2})}} \leq f_{\boldsymbol{w}_i'^{(t)}}$.

By leveraging (9) recursively in Eq. (8), we arrive at:

$$
\begin{aligned}
\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla f_{\boldsymbol{w}_i^{(t)}}\|^2] &\leq \frac{1}{\lambda}\sum_{t=0}^{T-1}(\mathbb{E}[f_{\boldsymbol{w}_i^{(t)}}] - \mathbb{E}[f_{\boldsymbol{w}_i^{(t+\frac{1}{2})}}]) + \frac{L\sigma^2\lambda T}{2}, \\
&\leq \frac{1}{\lambda}\sum_{t=0}^{T-1}(\mathbb{E}[f_{\boldsymbol{w}_i^{(t)}}] - \mathbb{E}[f_{\boldsymbol{w}_i^{(t+1)}}]) + \frac{L\sigma^2\lambda T}{2}, \\
&= \frac{1}{\lambda}(\mathbb{E}[f_{\boldsymbol{w}_i^{(0)}}] - \mathbb{E}[f_{\boldsymbol{w}_i^{(T)}}]) + \frac{L\sigma^2\lambda T}{2}, \\
&\leq \frac{1}{\lambda}(f_i^{(0)} - f_i^*) + \frac{L\sigma^2\lambda T}{2}, \\
\Rightarrow \min_{t=0,\cdots,T-1}\sum_{i=1}^{M}\mathbb{E}[\|\nabla f_{\boldsymbol{w}_i^{(t)}}\|^2] &\leq \sum_{i=1}^{M}\frac{f_i^{(0)} - f_i^*}{T\lambda} + \frac{LM\sigma^2\lambda}{2},
\end{aligned}
\tag{10}
$$

which concludes the proof. $\square$

Theorem 2. *We inherit the conditions in Theorem 1, and further suppose that Algorithm 1 applies one-step SGD to update $\mathbf{w}_i'$ with learning rate λ and the norm of $\boldsymbol{W}$ is bounded: $\|\boldsymbol{W}\|^2 \leq \gamma$. Then,*

$$
\min_{t=0,\cdots,T-1}\sum_{i=1}^{M}\mathbb{E}[\|\nabla f_{\boldsymbol{w}_i^{(t)}}\|^2] + E[\|(\boldsymbol{W}'^{(t)})^\top \nabla f_{\boldsymbol{w}_i'^{(t)}}\|^2] \leq \sum_{i=1}^{M}\frac{f_i^{(0)} - f_i^*}{T\lambda} + \frac{LM\sigma^2\lambda(1+\gamma^2)}{2}. \tag{11}
$$

Proof. We only need to discuss adjacent-learning, since self-learning naturally follows Eq. (8).

For adjacent-learning, we have the one-step SGD update as follows:

$$
\begin{aligned}
\boldsymbol{W}'^{(t+\frac{1}{2})} &= \boldsymbol{W}'^{(t)}\boldsymbol{A}^{(t)}, \\
\boldsymbol{W}'^{(t+\frac{1}{2})} &= \boldsymbol{W}'^{(t)}\left(\boldsymbol{I} - \lambda\frac{\partial}{\partial\boldsymbol{A}^{(0)}}f_{\boldsymbol{W}'^{(t)}}(x)\right), \\
&= \boldsymbol{W}'^{(t)}\left(\boldsymbol{I} - \lambda(\boldsymbol{W}'^{(t)})^\top\nabla f_{\boldsymbol{W}'^{(t)}}(x)\right), \\
&= \boldsymbol{W}'^{(t)} - \lambda\boldsymbol{W}'^{(t)}(\boldsymbol{W}'^{(t)})^\top\nabla f_{\boldsymbol{W}'^{(t)}}(x).
\end{aligned}
\tag{12}
$$

Hence, for each task i, the adjacent update is:

$$\boldsymbol{w}_i'^{(t+\frac{1}{2})} \quad = \quad \boldsymbol{w}_i'^{(t)} - \lambda \boldsymbol{W}'^{(t)} (\boldsymbol{W}'^{(t)})^\top \nabla f_{\boldsymbol{w}_i'^{(t)}}(x). \tag{13}$$

Then, similar to Eq. (8), adopting the L-smooth definition in (5) gives:

$$\begin{aligned}
f_{\boldsymbol{w}_i'^{(t+\frac{1}{2})}} &\leq f_{\boldsymbol{w}_i'^{(t)}} + (\boldsymbol{w}_i'^{(t+\frac{1}{2})} - \boldsymbol{w}_i'^{(t)})^\top \nabla f_{\boldsymbol{w}_i'^{(t)}} + \frac{L}{2}\|\boldsymbol{w}_i'^{(t+\frac{1}{2})} - \boldsymbol{w}_i'^{(t)}\|^2, \\
\Rightarrow \mathbb{E}[f_{\boldsymbol{w}_i'^{(t+\frac{1}{2})}}] &\leq f_{\boldsymbol{w}_i'^{(t)}} - \lambda\|(\boldsymbol{W}'^{(t)})^\top \nabla f_{\boldsymbol{w}_i^{(t)}}\|^2 + \frac{\lambda^2 L}{2}\mathbb{E}[\|\boldsymbol{W}'^{(t)}(\boldsymbol{W}'^{(t)})^\top \nabla f_{\boldsymbol{w}_i^{(t)}}\|^2], \\
&\leq f_{\boldsymbol{w}_i^{(t)}} - \lambda\|(\boldsymbol{W}'^{(t)})^\top \nabla f_{\boldsymbol{w}_i^{(t)}}\|^2 + \frac{\lambda^2\sigma^2\gamma^2 L}{2}, \\
\Rightarrow \sum_{t=0}^{T-1}\mathbb{E}[\|(\boldsymbol{W}'^{(t)})^\top \nabla f_{\boldsymbol{w}_i'^{(t)}}\|^2] &\leq \frac{1}{\lambda}\sum_{t=0}^{T-1}(\mathbb{E}[f_{\boldsymbol{w}_i'^{(t)}}] - \mathbb{E}[f_{\boldsymbol{w}_i'^{(t+\frac{1}{2})}}]) + \frac{L\sigma^2\lambda\gamma^2 T}{2}.
\end{aligned} \tag{14}$$

Eq. (14) + Eq. (8) yields:

$$\begin{aligned}
&\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla f_{\boldsymbol{w}_i^{(t)}}\|^2] + \mathbb{E}[\|(\boldsymbol{W}'^{(t)})^\top \nabla f_{\boldsymbol{w}_i'^{(t)}}\|^2] \quad \leq \\
&\frac{1}{\lambda}\sum_{t=0}^{T-1}\left(\mathbb{E}[f_{\boldsymbol{w}_i^{(t)}}] - \mathbb{E}[f_{\boldsymbol{w}_i^{(t+\frac{1}{2})}}] + \mathbb{E}[f_{\boldsymbol{w}_i'^{(t)}}] - \mathbb{E}[f_{\boldsymbol{w}_i'^{(t+\frac{1}{2})}}]\right) \quad + \quad \frac{L\sigma^2\lambda(1+\gamma^2)T}{2}.
\end{aligned}$$

By checking the flow in (7), we further derive:

$$\begin{aligned}
\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla f_{\boldsymbol{w}_i^{(t)}}\|^2] + \mathbb{E}[\|(\boldsymbol{W}'^{(t)})^\top \nabla f_{\boldsymbol{w}_i'^{(t)}}\|^2] &\leq \frac{1}{\lambda}\sum_{t=0}^{T-1}\left(\mathbb{E}[f_{\boldsymbol{w}_i^{(t)}}] - \mathbb{E}[f_{\boldsymbol{w}_i^{(t+1)}}]\right) \\
&\quad + \frac{L\sigma^2\lambda(1+\gamma^2)T}{2}, \\
&\leq \frac{1}{\lambda}(f_i^{(0)} - f_i^*) + \frac{L\sigma^2\lambda(1+\gamma^2)T}{2}, \\
\Rightarrow \min_{t=0,\cdots,T-1}\mathbb{E}[\|\nabla f_{\boldsymbol{w}_i^{(t)}}\|^2] + E[\|(\boldsymbol{W}'^{(t)})^\top \nabla f_{\boldsymbol{w}_i'^{(t)}}\|^2] &\leq \frac{f_i^{(0)} - f_i^*}{T\lambda} + \frac{L\sigma^2\lambda(1+\gamma^2)}{2}.
\end{aligned} \tag{15}$$

Summing the above inequation over all tasks gives the conclusion of this theorem.

□

B More Details of Experiments

B.1 Datasets

Taskonomy. The Taskonomy dataset [5] is a large-scale multi-task dataset that consists of indoor scenes with diverse tasks. The dataset is indexed by different buildings, each of which includes thousands of images. Since the scale of the entire dataset is too hard to tackle (1̃2TB in total), in our experiment we construct two subsets of the entire dataset, namely Taskonomy 1-building and Taskonomy 5-building for evaluations under three of the proposed scenarios. Taskonomy 1-building includes 9464 images from the building named Cauthron. Taskonomy 5-building involves 47320 images in total from the following five buildings: Darden, Hanson, Muleshoe, Newfields, Ranchester. In both two datasets, we randomly split them with the ratio 5:1 representing the training and testing set, respectively. The detailed split is referred to our attached code. In our main experiments, we perform five tasks for analysis, namely Depth Prediction, Surface Normal Prediction, Keypoint Detection, Edge Detection (3D), and Reshading. In Appendix F, we also provide two more task combinations, which involve another three tasks (Semantic Segmentation, Curvature Estimation, and Edge Detection 2D).

CityScapes. The CityScapes dataset [1] consists of street views in high resolution. We investigate the two-task learning scenario including Semantic Segmentation and Depth Estimation on this dataset. We apply the official train/test split, and use the 19-class labels for Semantic Segmentation. Similar to [4], we also perform random cropping and flipping technique during the training process.

B.2 Implementation

Our method implements an alternating training with two processes: self-learning and adjacent-learning. In self-learning, we adopt the SGD optimizer with learning rate 0.05, momentum 0.8, and weight decay 0.0001 on Taskonomy. Note that we fix this setting for all scenarios without any hyper-parameter hacking to avoid over-fitting to the evaluations. On CityScapes, We apply the Adam optimizer with learning rate 0.001, βs = (0.5, 0.999) and weight decay 0.0001, under the same setting as [4]. In adjacent learning, we apply the SGD optimizer (which aligns with Theorem 2) with learning rate 0.1 and no momentum for all datasets. The batch size is set to 16 for all datasets. The self-learning process and adjacent-learning process both alternate with a counter of 10 epochs. Indeed, we also test the counter of 20 epochs but find it results in very similar performance. The total training epoch is set to 300 for all methods on Taskonomy and 200 on CityScapes. We further train our method until 400 epochs on CityScapes and the converged optimization gives more significant improvements, compared with the baselines reported in [4].

Besides, for Adj-only, Adj-fixed, Share-W, and the regularization methods, we adopt exactly the same hyper-parameter setting as our method for a fair comparison. Furthermore, the coefficient of the regularizer is set to 1×10^{-5} in the fixed case and 5×10^{-5} in the dynamic case respectively, to yield the best performance. The fixed and dynamic regularizer [2] take the following mathematical forms:

$$\mathcal{R}_{\text{fixed}}(\boldsymbol{W}, \boldsymbol{A}) = \lambda \cdot \text{tr}(\boldsymbol{W}\boldsymbol{A}\boldsymbol{W}^\top), \text{where } \boldsymbol{A} = (\boldsymbol{I}_{M\times M} - \frac{1}{M}\mathbf{1}\mathbf{1}^\top)^2.$$

$$\mathcal{R}_{\text{dynamic}}(\boldsymbol{W}, \boldsymbol{A}) = \lambda \cdot \text{tr}(\boldsymbol{W}\boldsymbol{A}^{-1}\boldsymbol{W}^\top), \text{where } \boldsymbol{A} = \left(\frac{(\boldsymbol{W}^\top\boldsymbol{W})^{\frac{1}{2}}}{\text{tr}((\boldsymbol{W}^\top\boldsymbol{W})^{\frac{1}{2}})}\right).$$

C Model size comparison

Table 5 depicts the relative size of different models on CityScapes. It is noticed that our PPNet does not increase the model complexity in inference compared with Self-Learn, since the parameters in the passing network are not involved in the final inference or testing. Although there exists multi-task models (such as AdaShare) that yield lower parameter consumption compared with PPNet, our PPNet is effective in reaching the remarkably superior performance while maximally controlling the model size and being implementation-friendly amongst diverse scenarios.

Table 5: Model size comparison on CityScapes.

Model	Self-Learn	Share-W	Cross-Stitch	Sluice	NDDR-CNN	MTAN	DEN	AdaShare	PPNet
# Params	2	1	2	2	2.07	2.41	1.12	1	2

D Error Bar

Table 6 illustrates the detailed testing losses of PPNet with standard deviations. The results are computed with three independent runs. As displayed, the standard deviations are small, compared with the testing loss. This indicates our training strategy and the PPNet model do not increase the instability over the original backbone, and the improvement over other comparing methods is convincingly significant.

E Distributed and General Cases

In Table 3 in the paper, we have provided the total testing loss for all compared methods. Here, Table 7 and Table 8 additionally display the detailed task specific losses in the distributed case on Xception and ResNet34, respectively, while Table 9 and Table 10 show the results in the general case. Clearly, PPNet almost achieves the best performance for all tasks.

Table 6: Detailed results of PPNet (Loss $\times 10^{-2}$). The standard deviations are displayed in the parenthesis, averaged across three independent runs. The standard deviations are very small compared with the mean values.

Case	Backbone	Task1	Task2	Task3	Task4	Task5	Total
Multi-Output	Xception	3.85 (0.05)	4.70 (0.06)	5.01 (0.01)	8.15 (0.10)	8.24 (0.08)	29.95 (0.15)
	ResNet34	4.88 (0.10)	5.26 (0.07)	6.72 (0.07)	11.02 (0.03)	10.29 (0.08)	38.16 (0.18)
Distributed	Xception	3.48 (0.00)	3.78 (0.04)	3.89 (0.17)	3.79 (0.03)	4.02 (0.13)	18.96 (0.14)
	ResNet34	4.98 (0.02)	5.06 (0.11)	5.23 (0.03)	5.12 (0.11)	4.81 (0.08)	25.20 (0.05)
General	Xception	3.45 (0.09)	5.47 (0.02)	5.09 (0.11)	7.75 (0.06)	8.84 (0.09)	30.60 (0.17)
	ResNet34	4.78 (0.01)	6.24 (0.04)	7.02 (0.15)	9.66 (0.10)	11.51 (0.05)	39.21 (0.20)

Table 7: Taskonomy 5-building testing loss ($\times 10^{-2}$) based on Xception in the distributed case.

	Building	Darden	Hanson	Muleshoe	Newfields	Ranchester	Total	Δ_T
Baseline	Self-Learn	5.53	5.38	4.94	5.13	4.61	25.58	0.0
	Adj-only	4.36	4.95	4.95	4.69	4.52	23.48	+7.9
	Adj-fixed	4.09	4.68	4.69	5.01	4.07	22.54	+11.6
	Share-W	4.94	4.50	4.98	4.95	4.52	23.88	+6.3
Regularizer	Fixed	4.58	5.10	5.71	5.15	5.10	25.64	-0.8
	Dynamic	4.72	6.78	5.70	5.32	4.99	27.51	-7.7
Grouping	Pessimistic	5.53	5.38	5.51	5.16	4.69	26.27	-2.8
	Optimal	4.38	4.31	4.68	4.48	4.13	21.98	+13.8
	Random	4.66	4.71	4.95	4.75	4.46	23.53	+7.7
	PPNet	**3.48**	**3.78**	**3.89**	**3.79**	**4.02**	**18.96**	**+25.4**

Table 8: Taskonomy 5-building testing loss ($\times 10^{-2}$) based on ResNet34 in the distributed case.

	Building	Darden	Hanson	Muleshoe	Newfields	Ranchester	Total	Δ_T
Baseline	Self-Learn	6.38	6.31	5.89	6.85	5.40	30.82	0.0
	Adj-only	5.71	5.52	6.27	5.69	5.66	28.86	+5.7
	Adj-fixed	5.19	6.17	6.72	5.48	5.42	28.99	+5.3
	Share-W	6.25	5.87	6.39	6.35	5.57	30.43	+0.9
Regularizer	Fixed	6.68	5.66	6.30	7.12	6.10	31.85	-0.8
	Dynamic	6.55	6.55	7.77	6.98	5.95	33.81	-7.7
Grouping	Pessimistic	6.39	6.31	6.62	6.85	5.97	32.14	-4.6
	Optimal	5.48	5.62	5.77	5.56	5.33	27.76	+9.4
	Random	6.03	5.92	6.11	5.88	5.53	29.47	+3.9
	PPNet	**4.98**	**5.06**	**5.23**	**5.12**	**4.81**	**25.20**	**+17.8**

F More Task Combinations

To demonstrate the generalization capability of PPNet, we further compare the performance over two more task combinations in the radar format, as displayed in Fig. 5. Clearly, our PPNet yields the best performance on every single task regardless of the task combinations, even when different types of tasks like Semantic Segmentation, Curvature, and Edge Detection are involved.

G Detailed Learning Curve Comparison

We provided a detailed comparison on learning curves between PPNet, Self-Learn, Adj-only, and the non-linear extension of our PPNet. The non-linearity is introduced by setting $\boldsymbol{W}_l' = \boldsymbol{W}_l + \tanh(\boldsymbol{W}_l \boldsymbol{A}_l)$, where the re-initialization of $\boldsymbol{A}_l$ takes the form $\boldsymbol{A}_l^{(0)} = \boldsymbol{0}$ to ensure the entire initial

Table 9: Taskonomy 5-building testing loss ($\times 10^{-2}$) based on Xception in the general case.

	Task Building	Depth Darden	Normal Hanson	Keypoint Muleshoe	Edge Newfields	Reshading Ranchester	Total	Δ_T
Baseline	Self-Learn	4.66	5.92	6.79	9.30	9.65	36.33	0.0
	Adj-only	5.12	5.80	5.39	**7.41**	9.35	33.08	+7.2
	Adj-fixed	5.03	5.70	5.59	7.91	9.78	34.01	+5.4
	Share-W	4.89	7.06	8.76	8.96	11.25	40.92	-13.23
Regularizer	Fixed	5.80	5.90	6.34	8.77	9.73	36.53	-2.5
	Dynamic	5.18	6.86	5.53	8.81	10.41	36.80	-2.2
Grouping	Pessimistic	5.90	7.62	10.13	10.10	12.38	46.13	-28.3
	Optimal	4.11	5.92	5.68	7.96	9.41	33.08	+9.0
	Random	5.01	6.56	7.75	8.74	10.44	38.50	-6.9
	PPNet	**3.45**	**5.47**	**5.09**	7.75	**8.84**	**30.60**	**+16.7**

Table 10: Taskonomy 5-building testing loss ($\times 10^{-2}$) based on ResNet34 in the general case.

	Task Building	Depth Darden	Normal Hanson	Keypoint Muleshoe	Edge Newfields	Reshading Ranchester	Total	Δ_T
Baseline	Self-Learn	5.91	6.70	8.15	11.33	12.56	44.66	0.0
	Adj-only	5.67	6.09	8.67	10.46	12.35	43.25	+3.2
	Adj-fixed	5.91	6.39	7.73	11.15	12.16	43.34	+2.9
	Share-W	7.63	8.69	10.91	11.32	15.17	53.72	-22.7
Regularizer	Fixed	6.55	6.35	7.63	11.38	12.29	44.20	+0.5
	Dynamic	6.47	6.29	8.71	10.92	13.08	45.47	-2.1
Grouping	Pessimistic	10.85	10.28	13.01	12.87	17.93	64.94	-50.6
	Optimal	5.67	6.70	8.00	9.95	11.86	42.18	+4.7
	Random	7.01	7.88	9.82	11.24	14.09	50.05	-13.6
	PPNet	**4.78**	**6.24**	**7.02**	**9.66**	**11.51**	**39.21**	**+12.6**

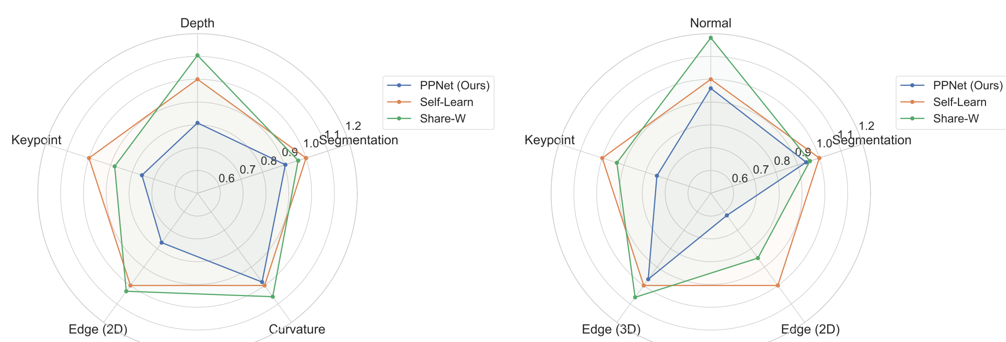

Figure 5: The comparison on two extra task combinations. Left: The combination with Semantic Segmentation, Principal Curvature, Edge Detection (2D), Keypoint Detection, and Depth Estimation. Right: The combination with Semantic Segmentation, Edge Detection (2D), Edge Detection (3D), Keypoint Detection, and Surface Normal Detection. Excitingly, PPNet achieves the best performance on all tasks regardless of which task combination is given.

passing to be the identity mapping, and l represents the index of the current layer. Fig. 6 illustrates the learning curves on Taskonomy 1-building in the multi-output case. It is observed that PPNet significantly outperforms Self-Learn and Adj-only by yielding lower loss and faster convergence, which necessitates our proposal of alternating the self-learning and adjacent-learning optimization.

Yet and still, we do not see a convincing improvement of adding the non-linearity into PPNet. Therefore, we prefer the linear PPNet under this scenario for less computational cost.

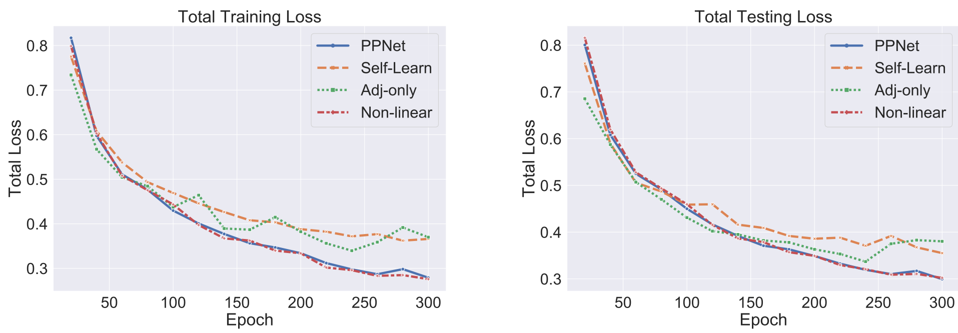

Figure 6: The comparison on learning curves on Taskonomy 1-building in the multi-output case. Left: the total training loss; Right: the total testing loss. PPNet and its non-linear extension result in the lowest training and testing loss.

H Qualitative Visualization

We visualize the results of Self-Learn, Share-W and PPNet on the testing set of CityScapes in Fig. 7. Clearly, PPNet gives out more accurate predictions on semantic segmentation and clearer depth estimations.

I Task Relationship

In this section, we discuss and contrast the task relationship revealed by task grouping [3] and our PPNet on Xception in the multi-output case of Taskonomy.

I.1 Task Grouping

Table 11 and 12 respectively show the optimal and pessimistic grouping discovered after comparing all possible task combinations. We observe that different task combination leads to different performance, and it is indeed important to leverage positive combination while suppressing negative transfer.

I.2 PPNet

Different from task grouping which searches desired task correlation by hand-crafted comparison, PPNet dynamically correlates the parameter space of different tasks by a trainable parameter passing matrix $\boldsymbol{A}_l$ for each layer. To better understand such dynamics, we require to track the values of $\boldsymbol{A}_l$ during training. Recalling that in our channel-wise version, we have reshaped $\boldsymbol{A}_l$ to enhance its expressivity. Here, for visualization, we first transpose and reshape $\boldsymbol{A}_l$ of shape $(c_{\text{out}}M) \times (c_{\text{out}}M)$ back into the tensor of shape $(M \times M) \times (c_{\text{out}} \times c_{\text{out}})$. By this means, the first two dimensions represent task-to-task relation, while the rest two dimensions compute the channel-wise dependency. Afterwards, we sum the channel-wise dependency and divide it by a factor of c_{out}, which reduces $\boldsymbol{A}_l$ into a matrix $\boldsymbol{A}'_l$ of shape $M \times M$. A notable property of our transformation is that $\boldsymbol{A}'_l$ will be an identity matrix since $\boldsymbol{A}_l$ is identically re-initialized at the beginning of each adjacent-learning stage. In $\boldsymbol{A}'_l$, each column represents the task affinity of all tasks towards the task of this column. Furthermore, we prefer to compute the averaged change of $\boldsymbol{A}'_l$ across the time horizon, denoted as $\tilde{\boldsymbol{A}}_l$ satisfying $\tilde{\boldsymbol{A}}_l^{(t)} = \frac{1}{t}\sum_{\tau=1}^{t}(\boldsymbol{A}_l'^{(\tau)} - \boldsymbol{I})$, where $\boldsymbol{A}_l'^{(\tau)} - \boldsymbol{I}$ represents the update of $\boldsymbol{A}_l'^{(\tau)}$ after the τ-th adjacent-learning phase. Employing $\tilde{\boldsymbol{A}}_l^{(t)}$ is more advantageous in reflecting the accumulated effect during the entire training process.

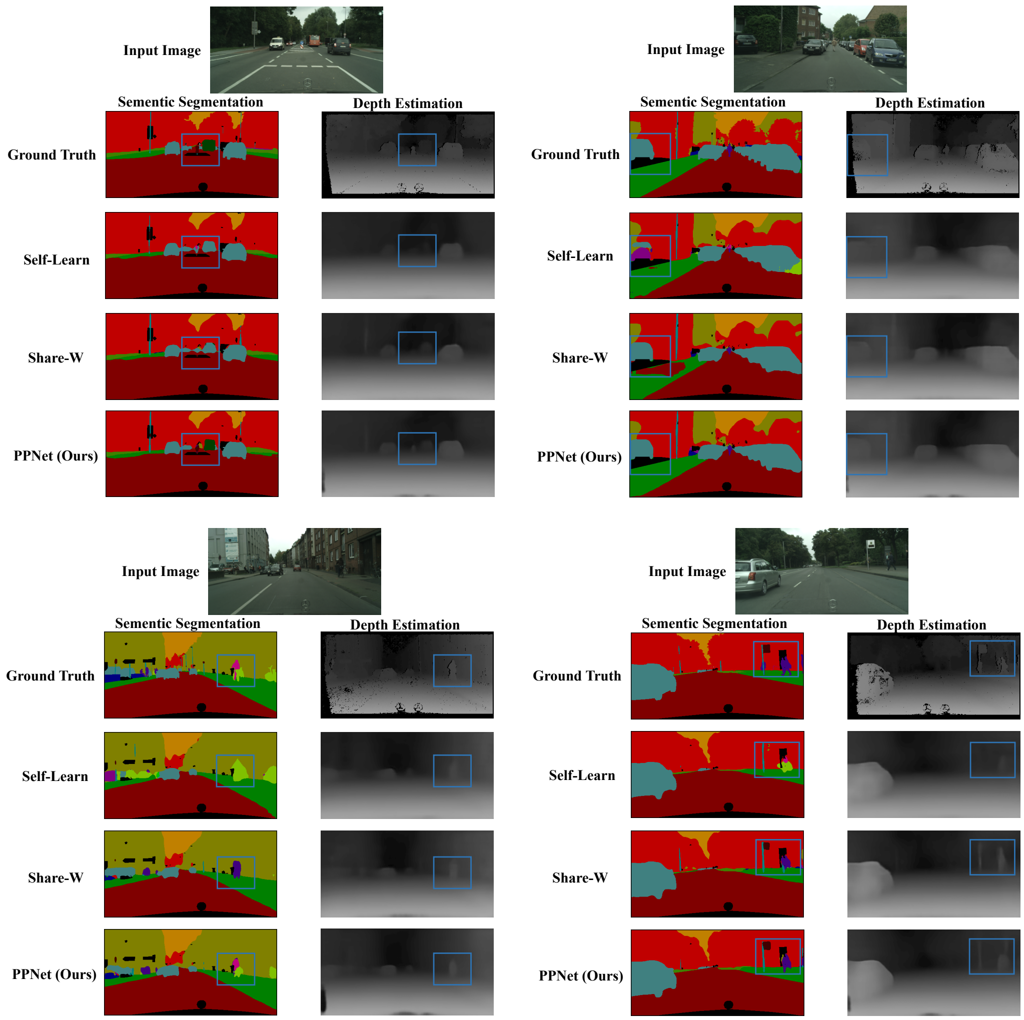

Figure 7: Qualitative Visualization on CityScapes. The blue box indicates the region of interest. Better viewed by zooming in.

Dynamics of $\tilde{\boldsymbol{A}}_l^{(t)}$. We choose the last convolutional layer in Xception and plot the values of $\tilde{\boldsymbol{A}}_l^{(t)}$ in each t-th adjacent-learning phase (every 20 epochs during training) in Fig. 8. The first five figures exhibit the external interaction (the relation of all tasks $j \neq i$ towards task i), while the last one refers to the self interaction. As observed, the task relation could change dynamically. For example, for the Normal task, Depth is initially negatively valued, but after a certain period of training, it is positively correlated to Normal, which could possibly explain the success of PPNet in utilizing the information of different tasks. In terms of self interaction, the value floats above the zero, which is reasonable since each task usually leverages more information from itself.

Comparison with task grouping. If we compare the eventual values of $\tilde{\boldsymbol{A}}_l^{(t)}$ with those selected tasks in Table 11, we find that PPNet exhibits some similar but not the entirely same behavior compared to task grouping. For instance, regarding Depth prediction, both task grouping and our PPNet return the positive relation with respect to Normal and Edge. Yet and additionally, our PPNet reveals that Keypoint is also positively interacted, which potentially thanks to the advantage of our adaptive parameter passing over simple task grouping.

Updated direction of $\boldsymbol{A}_l'^{(\tau)}$. We further visualize the updated direction of $\boldsymbol{A}_l'^{(\tau)} - \boldsymbol{I}$ at the end of adjacent-learning from the identity matrix via the heatmap. We display the results of Epoch 100, 200,

Table 11: Optimal grouping based on Xception in the multi-output case. The symbol ✓ denotes the dataset of the task in column is selected for the training of the task in row.

	Depth	Normal	Keypoint	Edge	Reshading	Loss
Depth	✓	✓		✓		4.50
Normal		✓				5.20
Keypoint	✓		✓	✓	✓	5.52
Edge				✓		8.50
Reshading		✓			✓	8.40

Table 12: Pessimistic grouping based on Xception in the multi-output case.

	Depth	Normal	Keypoint	Edge	Reshading	Loss
Depth	✓			✓		5.77
Normal		✓	✓			6.16
Keypoint	✓	✓	✓		✓	7.23
Edge	✓			✓		10.28
Reshading		✓	✓		✓	10.07

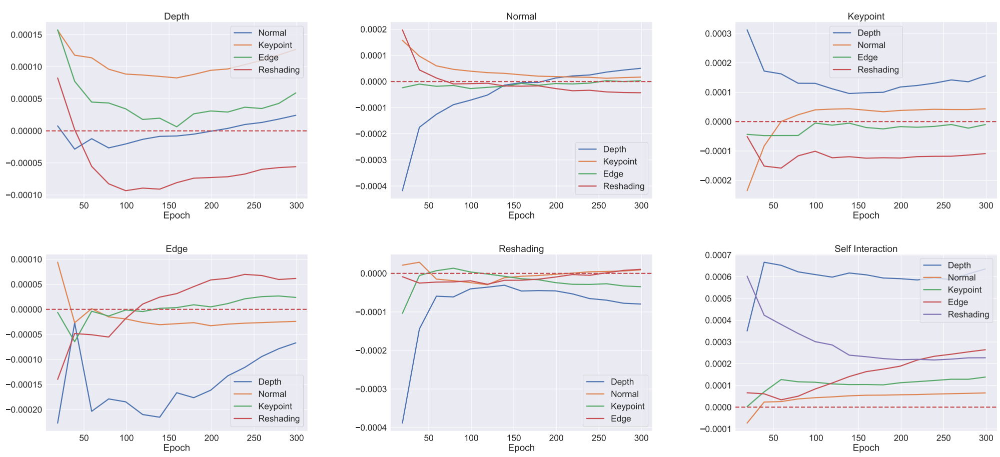

Figure 8: The values of $\tilde{\boldsymbol{A}}_l^{(t)} = \frac{1}{t}\sum_{\tau=1}^{t}(\boldsymbol{A}_l'^{(\tau)} - \boldsymbol{I})$ of the last convolutional layer in Xception encoder *w.r.t.* epoch. Here, $\boldsymbol{A}_l'^{(\tau)}$ is a reshaped and processed version of $\boldsymbol{A}_l^{(\tau)}$ from size $(c_{\text{out}}M) \times (c_{\text{out}}M)$ to $M \times M$ at phase τ.

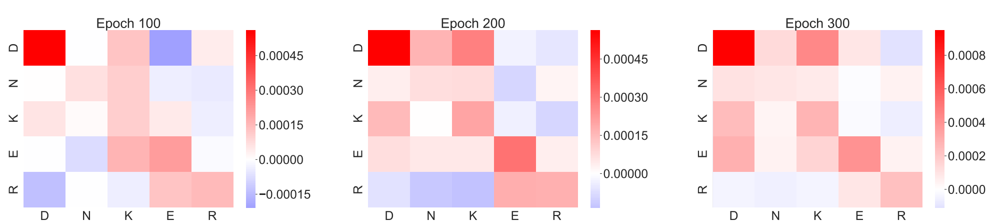

Figure 9: The heatmaps of $\boldsymbol{A}_l'^{(\tau)} - \boldsymbol{I}$ of the last convolutional layer in Xception encoder.

and 300 in Fig. 9. Similar to the discussions above, the contribution of different tasks dynamically varies and finally converges.

J Broader Impact

In this paper, we propose the parameter passing networks that leverage parameter passing for dynamic task relationship modeling. As a multi-task learning framework, our method does not raise ethical concerns. However, one should be aware that our PPNet is based on parameter passing, which requires an exchange of model parameters and does not maintain model privacy in some real-world scenarios.

References

[1] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[2] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet Talwalkar. Federated multi-task learning. *arXiv preprint arXiv:1705.10467*, 2017.

[3] Trevor Standley, Amir Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Which tasks should be learned together in multi-task learning? In *International Conference on Machine Learning*, pages 9120–9132. PMLR, 2020.

[4] Ximeng Sun, Rameswar Panda, Rogerio Feris, and Kate Saenko. Adashare: Learning what to share for efficient deep multi-task learning. *Advances in Neural Information Processing Systems*, 33, 2020.

[5] Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3712–3722, 2018.