Supplemental Document

Learning Multiple Pixelwise Tasks Based on Loss Scale Balancing

Jae-Han Lee
Korea University
jaehanlee@mcl.korea.ac.kr

Chul Lee
Dongguk University
chullee@dongguk.edu

Chang-Su Kim
Korea University
changsukim@korea.ac.kr

1. Network Architecture

We test two encoder backbones of MobileNet.v2 [9] and PNASNet [6]. Each network, excluding the last estimation layer, is used as the encoder. Table S-1 summarizes the sizes of input RGB images and encoder output features.

Table S-1. The sizes of input RGB images and encoder output features in experiments.

	NYUv2		Taskonomy
	MobileNet.v2	PNASNet	
Input image	$384 \times 288 \times 3$	$384 \times 288 \times 3$	$256 \times 256 \times 3$
Encoder output	$12 \times 9 \times 4320$	$12 \times 9 \times 4320$	$12 \times 9 \times 4320$

The decoders are used to expand low-resolution features to higher-resolution estimates. Each decoder consists of 12 convolutional layers of 3×3 kernels, ReLU activations, and 5 bilinear up-sample operations, as shown in Figure S-1. The number c_{o} of output channels is determined differently for each task, as listed in Table S-2. For example, for the full-sharing network for NYUv2, c_{o} is set to 17 to perform the three tasks (*i.e.*, depth, normal, and segmentation) simultaneously.

Table S-2. The number of output channels for each task.

	Depth	Normal	Segmentation	Curvature	Reshading	2D edge	3D edge	2D keypoint	3D keypoint
c_{o}	1	3	13	2	1	1	1	1	1

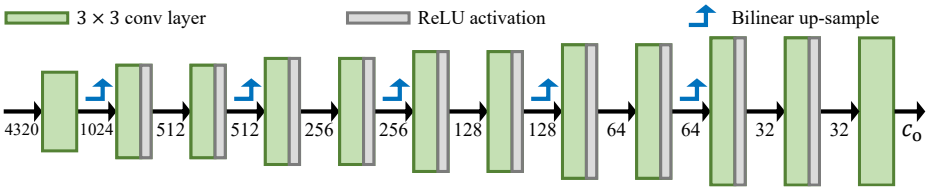


Figure S-1. The decoder network. Each number indicates the number of channels.

In the full-sharing and multi-decoder architectures, encoder representations are transferred directly to decoders. In the multi-column architecture, different representations from different encoders are linearly combined using the cross-stitch unit. Specifically, let x_k^{o} and x_k^{i} denote the output from the kth encoder and the input to the kth decoder, respectively. Then, for triple-task learning, the relation between x_k^{o} and x_k^{i} is given by

$$\begin{pmatrix} x_1^{\mathrm{i}} \\ x_2^{\mathrm{i}} \\ x_3^{\mathrm{i}} \end{pmatrix} = \begin{pmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} \end{pmatrix} \begin{pmatrix} x_1^{\mathrm{o}} \\ x_2^{\mathrm{o}} \\ x_3^{\mathrm{o}} \end{pmatrix} \tag{1}$$

where each α_{ij} is a learnable parameter. In other words, learning n tasks requires an $n \times n$ relation matrix. For the three-task learning of the NYUv2 dataset, α_{ij} is initialized to 0.9 if $i = j$, and to 0.05 otherwise. Also, on the Taskonomy dataset, every α_{ij} is initialized equally; $\alpha_{ij} = \frac{1}{4}$ for the 4-task learning and $\alpha_{ij} = \frac{1}{8}$ for the 8-task learning.

2. Loss Functions

We adopt a loss function for each task. Each loss function contains one or two regularizers to constrain the range of estimation results.

- The depth loss $\ell_{\mathbf{D}}$ is defined as the L_1-norm between the logarithms of a ground-truth depth map $\mathbf{D}$ and its estimate $\widehat{\mathbf{D}}$ with regularizers $r_{\mathbf{D}}^{\min}$ and $r_{\mathbf{D}}^{\max}$,

$$\ell_{\mathbf{D}} = \tfrac{1}{wh} \left\| \log(\widehat{\mathbf{D}}) - \log(\mathbf{D}) \right\|_1 + r_{\mathbf{D}}^{\min} + r_{\mathbf{D}}^{\max}. \tag{2}$$

  For a dataset with depth range $[d_{\min}, d_{\max}]$, the regularizers $r_{\mathbf{D}}^{\min}$ and $r_{\mathbf{D}}^{\max}$ are defined as

$$r_{\mathbf{D}}^{\min} = \tfrac{1}{wh} \left\| \min(\widehat{\mathbf{D}}, d_{\min}\mathbf{1}) - d_{\min}\mathbf{1} \right\|_1 \tag{3}$$

$$r_{\mathbf{D}}^{\max} = \tfrac{1}{wh} \left\| \max(\widehat{\mathbf{D}}, d_{\max}\mathbf{1}) - d_{\max}\mathbf{1} \right\|_1 \tag{4}$$

  where $\mathbf{1}$ is the constant map, every pixel of which has value 1.

- The normal loss $\ell_{\mathbf{N}}$ is obtained by calculating the inner products between normal vectors of a ground-truth normal map $\mathbf{N}$ and its estimate $\widehat{\mathbf{N}}$,

$$\ell_{\mathbf{N}} = 1 - \tfrac{1}{wh} \textstyle\sum_{x=1}^{w} \sum_{y=1}^{h} \mathbf{n}_{xy}^T \widehat{\mathbf{n}}_{xy} + r_{\mathbf{N}} \tag{5}$$

  where $\mathbf{n}_{xy}$ and $\widehat{\mathbf{n}}_{xy}$ are the normal vectors of pixel (x, y) in $\mathbf{N}$ and $\widehat{\mathbf{N}}$, respectively. Note that $\|\mathbf{n}_{xy}\| = 1$ for every (x, y). Thus, the regularizer $r_{\mathbf{N}}$ encourages each $\widehat{\mathbf{n}}_{xy}$ also to be a unit vector, which is given by

$$r_{\mathbf{N}} = \tfrac{1}{wh} \textstyle\sum_{x=1}^{w} \sum_{y=1}^{h} \left| \|\widehat{\mathbf{n}}_{xy}\| - 1 \right|. \tag{6}$$

- The segmentation loss $\ell_{\mathbf{S}}$ is the average cross-entropy between one-hot vectors in a ground-truth map $\mathbf{S}$ and probability vectors in an estimated map $\widehat{\mathbf{S}}$, which are obtained by computing the softmax function on an output feature map $\widehat{\mathbf{F}}$ of a decoder network. It is given by

$$\ell_{\mathbf{S}} = -\tfrac{1}{wh} \textstyle\sum_{x=1}^{w} \sum_{y=1}^{h} \mathbf{s}_{xy}^T \log \widehat{\mathbf{s}}_{xy} + r_{\mathbf{S}} \tag{7}$$

  where $\mathbf{s}_{xy}$ and $\widehat{\mathbf{s}}_{xy}$ are the one-hot and probability vectors of pixel (x, y). To constrain the range of $\widehat{\mathbf{F}}$, the regularizer $r_{\mathbf{S}}$ is defined as

$$r_{\mathbf{S}} = \tfrac{1}{wh} \textstyle\sum_{x=1}^{w} \sum_{y=1}^{h} \sum_{c=1}^{n_c} \left(\max\left(|\widehat{\mathbf{f}}_{xy}(c)|, \mathrm{f}_{\max} \right) - \mathrm{f}_{\max} \right) \tag{8}$$

  where n_c is the number of semantic segmentation classes (*e.g.*, 13 for the NYUv2 dataset) and $\mathrm{f}_{\max}$ is set to 10. Also, $\widehat{\mathbf{f}}_{xy}$ is the feature vector of pixel (x, y) in $\widehat{\mathbf{F}}$.

- The remaining 6 losses for curvature $\ell_{\mathbf{C}}$, reshading $\ell_{\mathbf{R}}$, 2D edge $\ell_{\mathbf{E}2}$, 3D edge $\ell_{\mathbf{E}3}$, 2D keypoint $\ell_{\mathbf{K}2}$, and 3D keypoint $\ell_{\mathbf{E}3}$ are the norms between ground-truth maps and their estimates. For example, the curvature loss $\ell_{\mathbf{C}}$ is defined as

$$\ell_{\mathbf{C}} = \tfrac{1}{wh} \left\| \widehat{\mathbf{C}} - \mathbf{C} \right\| + r_{\mathbf{C}} \tag{9}$$

  where $r_{\mathbf{C}}$ enforces the estimate to have a similar standard deviation to the ground-truth, which is given by

$$r_{\mathbf{C}} = |\sigma(\widehat{\mathbf{C}}) - \sigma(\mathbf{C})|. \tag{10}$$

  The other five losses $\ell_{\mathbf{R}}$, $\ell_{\mathbf{E}2}$, $\ell_{\mathbf{E}3}$, $\ell_{\mathbf{K}2}$, and $\ell_{\mathbf{K}3}$ are defined in the same way as (9) and (10).

Evaluation protocol for Taskonomy [10]: From the tiny split of Taskonomy, we build a mini dataset of 2,762 training images and 548 test images with a sampling ratio of $\frac{1}{100}$. In the evaluation, we compute losses, which are defined above in (2)∼(10), but exclude regularizer values. For example, to compute depth errors in Table 5 in the main paper, we compute $\frac{1}{wh} \left\| \log(\widehat{\mathbf{D}}) - \log(\mathbf{D}) \right\|_1$. For easier comparison of errors in different orders of magnitude in Table 5, values are multiplied by 10 for 2D edge errors, and multiplied by 100 for 3D edge and 2D keypoint errors.

3. Training Details

During training, we perform the horizontal flip data augmentation in an online manner. For the NYUv2 dataset, we fill in missing values in depth maps and normal maps using the colorization algorithm in [5]. We initialize the encoder parameters to those pre-trained with ImageNet [2] and the decoder parameters using the Xavier [3] method. As a solver, we adopt the AdamW algorithm [8] and set its learning rate, weight decay, β_1, β_2, and ϵ to 10^{-4}, 0.01, 0.9, 0.999, and 10^{-8}, respectively. In all experiments, one period is set to one epoch. For the NYUv2 dataset with the MobileNet.v2 backbone, the network for each configuration is trained for 1,000 epochs. In the default mode, the increasing rate of the hyper-parameter β is set to 0.02 per epoch. For the NYUv2 dataset with the PNASNet backbone, the network is trained for 50 epochs, and the increasing rate of β is 0.05 per epoch. For the Taskonomy dataset, the network is trained for 200 epochs, and the increasing rate of β is 0.5 per epoch.

In MTL, the performances of different tasks are in a trade-off. Initial weights, hence, affect the final performances. For a fair comparison, each algorithm is initialized in the same way using equal weighting, *i.e*. the weight vector is set to $(w_{\mathbf{D}}, w_{\mathbf{N}}, w_{\mathbf{S}}) = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$. In the proposed algorithm, this is achieved by setting the priority factors in Eq. (4) in the main paper to $(\pi_{\mathbf{D}}, \pi_{\mathbf{N}}, \pi_{\mathbf{S}}) = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$. For Uncert, it is done by setting its parameter σ. Since DWA and GradNorm initialize the weights of all losses equally, it is the same as the equal weighting scheme. In contrast, GLS formulates the overall loss as the geometric mean of individual losses, instead of the weighted sum in Eq. (2) in the main paper. Thus, we test GLS as it is.

4. Repeated Tests for Reliable Comparison

To compare the algorithms more reliably, we repeat the same experiments 5 times for the multi-column architecture in Table 2 in the main paper and summarize the results below in Table S-3. Due to the randomness in training, each experimental trial yields a different result. However, in general, the proposed algorithm achieves the best MTL results. Note that, for each metric, there are 25 trials. Out of those 25 trials, the top 20% trials are mostly performed by the proposed algorithm. Also, the average rank over the five trials of the proposed algorithm is 6.82, which is significantly better than the second-best one ($= 11.62$) of the Uncert.

The last five rows in Table S-3 summarize the average performances over the five trials of each algorithm. The proposed algorithm achieves best results in all metrics with no exception. Thus, it has the best average rank ($= 1.08$), which is significantly better than the second-best one ($= 2.58$) of Uncert.

Table S-3. Performance comparison of the proposed algorithm with the conventional algorithms on the NYUv2 dataset using the multi-column architecture with the MobileNet.v2 backbone. Each algorithm is tested 5 times. The rank performance is provided within parentheses. The best average rank is **boldfaced**. Also, the results of top 20% trials for each metric are in red. The last five rows summarize the average performances over the five trials of each algorithm.

	Depth		Normal		Segmentation		Rank			
	δ_1	RMSE	δ_{30°	$\angle^{\text{mean}}$	mIoU	Acc.	Min	Max	Avg.	
Equal weighting	59.5% (21.5)	0.801 (24.5)	44.3% (15.5)	40.0 (12.5)	19.9% (20)	55.7% (18)	12.5	24.5	18.67	16.88
	59.7% (18.5)	0.793 (17)	43.6% (24)	40.3 (21)	20.2% (17)	55.5% (23.5)	17	24	20.08	
	60.3% (8)	0.790 (12)	45.1% (4.5)	38.9 (2)	19.7% (22.5)	55.5% (23.5)	2	23.5	12.08	
	59.4% (23.5)	0.799 (22.5)	44.3% (15.5)	40.0 (12.5)	20.0% (19)	56.2% (9.5)	9.5	23.5	17.08	
	59.3% (25)	0.796 (20)	43.9% (20)	40.1 (16)	20.8% (7)	56.1% (11)	7	25	16.50	
Uncert [4]	60.9% (3.5)	0.775 (2)	44.1% (18)	40.1 (16)	20.8% (7)	56.3% (7.5)	2	18	9.00	11.62
	60.1% (10)	0.792 (14)	43.9% (20)	40.5 (23)	20.4% (15)	55.9% (13.5)	10	23	15.92	
	59.4% (23.5)	0.789 (11)	45.3% (2)	**38.8 (1)**	20.8% (7)	55.6% (21)	1	23.5	10.92	
	59.8% (15.5)	0.788 (9.5)	44.3% (15.5)	40.0 (12.5)	20.5% (12.5)	55.9% (13.5)	9.5	15.5	13.17	
	59.8% (15.5)	0.798 (21)	45.2% (3)	39.1 (3)	20.8% (7)	56.5% (5)	3	21	9.08	
DWA [7]	59.9% (13.5)	0.792 (14)	43.7% (23)	40.2 (19)	20.5% (12.5)	56.2% (9.5)	9.5	23	15.25	16.12
	59.6% (20)	0.794 (19)	43.8% (22)	40.5 (23)	20.2% (17)	55.8% (16)	16	23	19.50	
	59.7% (18)	0.801 (24.5)	43.9% (20)	40.2 (19)	19.8% (21)	55.7% (18)	18	24.5	20.08	
	59.9% (13.5)	0.799 (22.5)	43.3% (25)	40.8 (25)	20.8% (7)	56.3% (7.5)	7	25	16.75	
	60.1% (10)	0.783 (8)	44.6% (11)	39.8 (9.5)	20.5% (12.5)	56.6% (3)	3	12.5	9.00	
GLS [1]	60.6% (7)	0.778 (4.5)	44.3% (15.5)	40.5 (23)	19.6% (24.5)	55.9% (13.5)	4.5	24.5	14.67	13.57
	60.7% (6)	0.782 (6.5)	44.6% (11)	40.1 (16)	19.6% (24.5)	55.6% (21)	6	24.5	14.17	
	59.7% (18)	0.793 (17)	44.6% (11)	40.0 (12.5)	19.7% (22.5)	55.6% (21)	11	22.5	17.00	
	60.9% (3.5)	0.782 (6.5)	44.9% (8)	39.8 (9.5)	20.2% (17)	55.7% (18)	3.5	18	10.42	
	60.0% (12)	0.788 (9.5)	45.0% (6.5)	39.3 (4)	20.5% (12.5)	55.1% (25)	4	25	11.58	
Proposed	**61.6% (1)**	**0.772 (1)**	44.8% (9)	39.5 (6)	21.0% (3.5)	56.5% (5)	1	9	4.25	**6.82**
	60.1% (10)	0.792 (14)	44.5% (13)	40.2 (19)	21.0% (3.5)	56.5% (5)	3.5	19	10.75	
	59.5% (21.5)	0.793 (17)	45.1% (4.5)	39.4 (5)	20.6% (10)	55.9% (13.5)	4.5	21.5	11.92	
	61.1% (2)	0.778 (4.5)	45.0% (6.5)	39.7 (8)	**21.4% (1.5)**	56.8% (2)	1.5	8	4.08	
	60.8% (5)	0.776 (3)	**45.4% (1)**	39.6 (7)	**21.4% (1.5)**	**57.0% (1)**	1	7	**3.08**	

	Average performance of five trials for each algorithm								
Equal weighting	59.6% (5)	0.796 (5)	44.2% (4)	39.9 (3)	20.1% (4)	55.8% (4)	3	5	4.17
Uncert [4]	60.0% (3)	0.788 (3)	43.6% (3)	**39.7 (1.5)**	20.7% (2)	56.0% (3)	1.5	3	2.58
DWA [7]	59.8% (4)	0.794 (4)	43.9% (5)	40.3 (5)	20.4% (3)	56.1% (2)	2	5	3.83
GLS [1]	60.4% (2)	0.785 (2)	44.7% (2)	39.9 (4)	19.9% (5)	55.6% (5)	2	5	3.33
Proposed	**60.6% (1)**	**0.782 (1)**	**45.0% (1)**	**39.7 (1.5)**	**21.1% (1)**	**56.5% (1)**	1	1.5	**1.08**

5. Loss Scale Trends

We compare weight and loss scale trends in Figure S-2, Figure S-3, and Figure S-4. Note that a similar experiment is done in Figure 9 in the main paper. Here, the results on different combinations of encoder backbone and dataset are reported. Figure S-2 shows the comparison results using the MobileNet.v2 backbone on the NYUv2 dataset. In equal weighting and DWA, the segmentation loss is dominant throughout the training. In Uncert, each loss scale converges gradually to a similar level as the training progresses. However, it converges only after many training epochs. On the other hand, the proposed algorithm equalizes the loss scales much faster by adjusting the weights effectively.

Figure S-3 and Figure S-4 compare weight and loss scale trends of the 4-task learning and 8-task learning on the Taskonomy dataset, respectively. Since equal weighting and GLS [1] are trained with fixed weights, the corresponding weight graphs are omitted. Similar trends to Figure S-2 are observed. In particular, in the 8-task learning in Figure S-4, the 2D keypoint loss has quite a small magnitude compared to the other losses. The conventional algorithms keep this loss on a small scale throughout the training. On the other hand, the proposed algorithm trains the 2D keypoint loss in a balanced manner with the other losses. This results in the best 2D keypoint performance in Table 5 in the main paper.

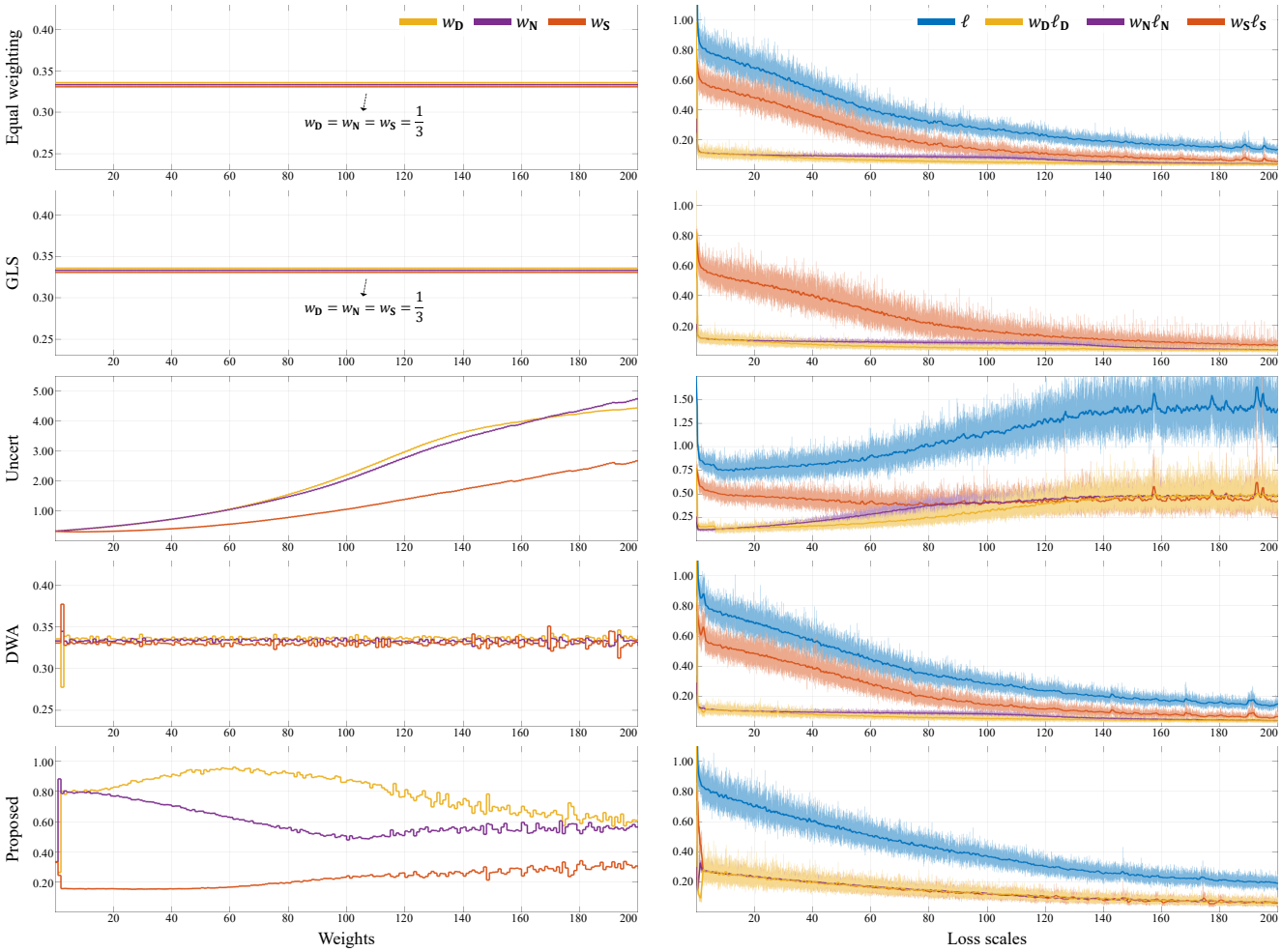


Figure S-2. Illustration of weights and loss scales of the three tasks on the NYUv2 dataset.

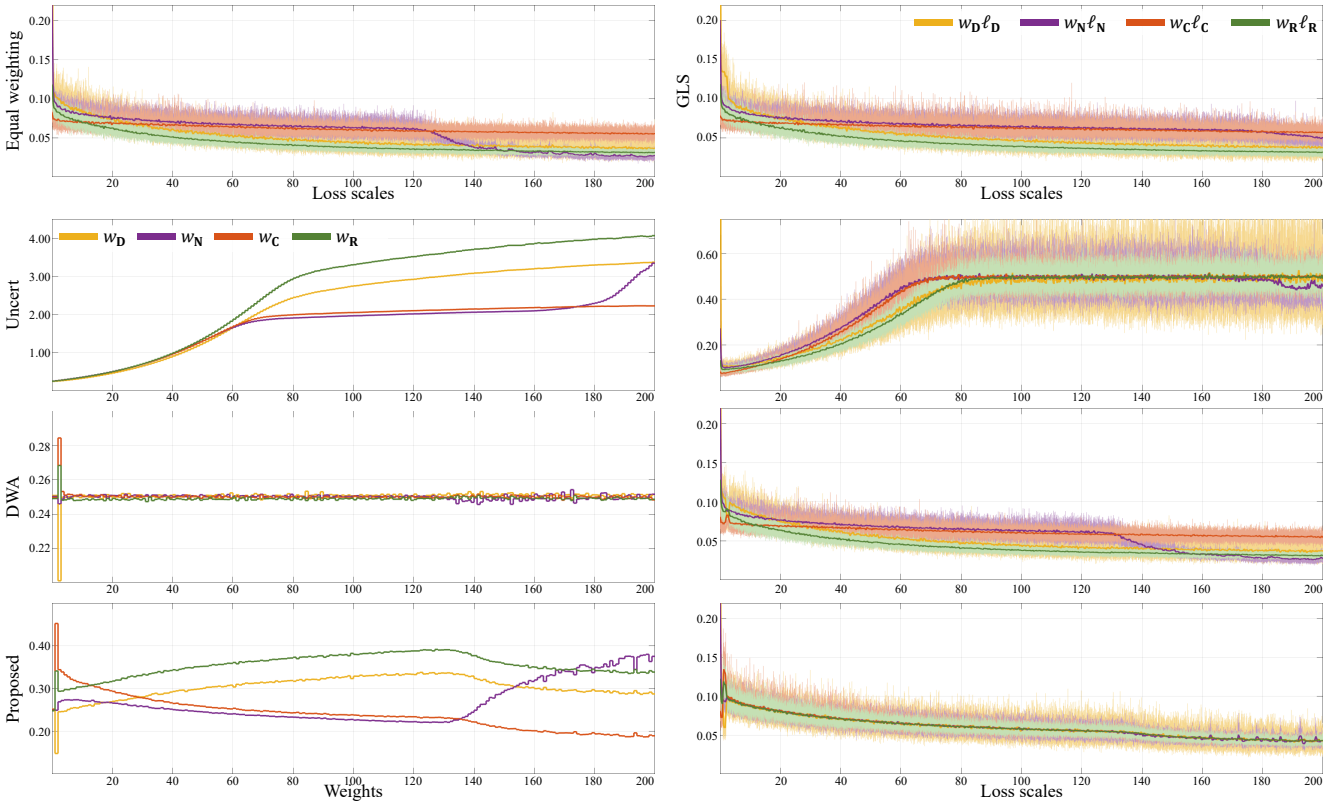


Figure S-3. Illustration of weights and loss scales of the four tasks on the Taskonomy dataset.

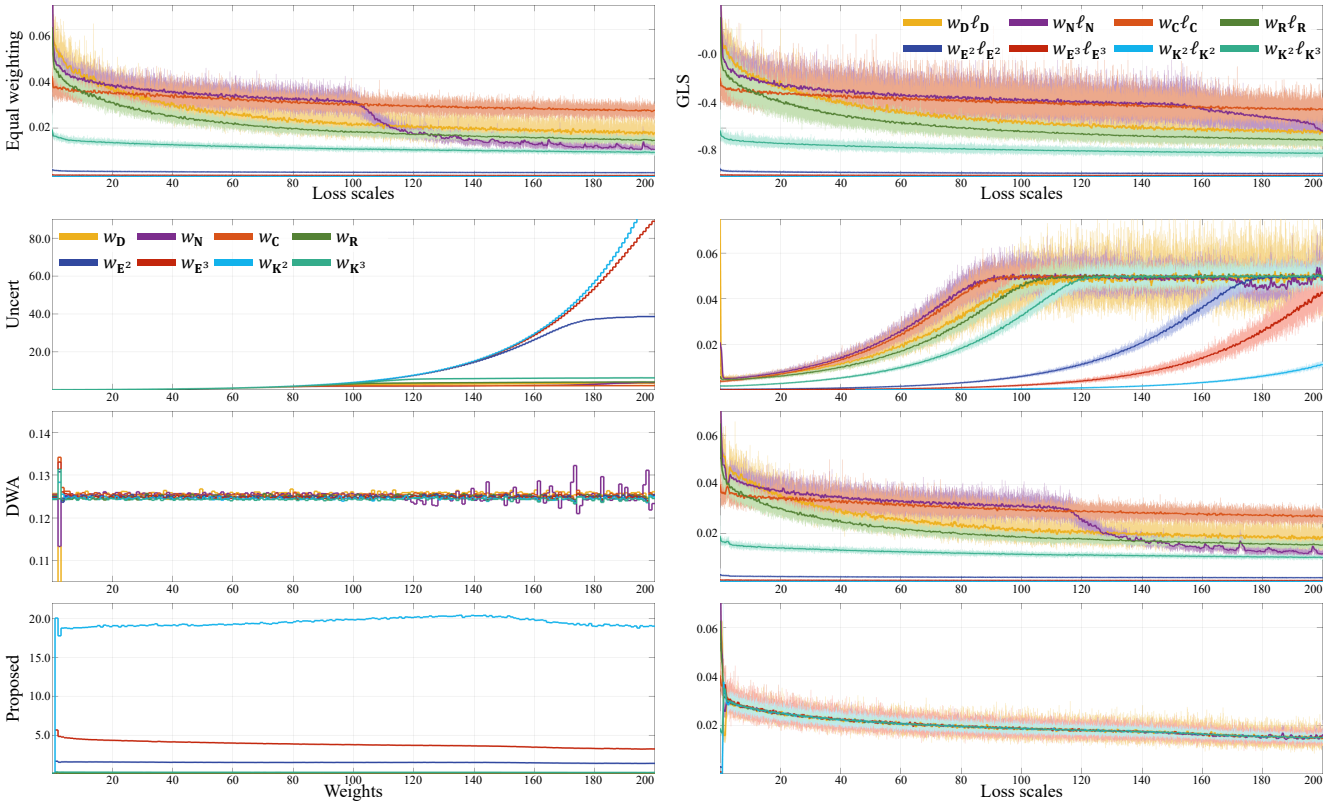


Figure S-4. Illustration of weights and loss scales of the eight tasks on the Taskonomy dataset.

6. Training Visualization

Figure S-5 visualizes the regions where losses occur in each task of the 8-task learning on the Taskonomy dataset. We compare the results of the proposed algorithm with those of equal weighting after 2, 10, and 200 training epochs. For easier comparison, we also show the overall error map, which is the sum of the error maps of the eight tasks. Throughout the training, edge and keypoint losses have relatively small magnitudes. For this reason, in the overall error maps of equal weighting, edges and keypoints are less activated. In contrast, in the proposed algorithm, all scene components, including edges and keypoints, are learned in a balanced manner, enabling the proposed algorithm to provide better MTL results.

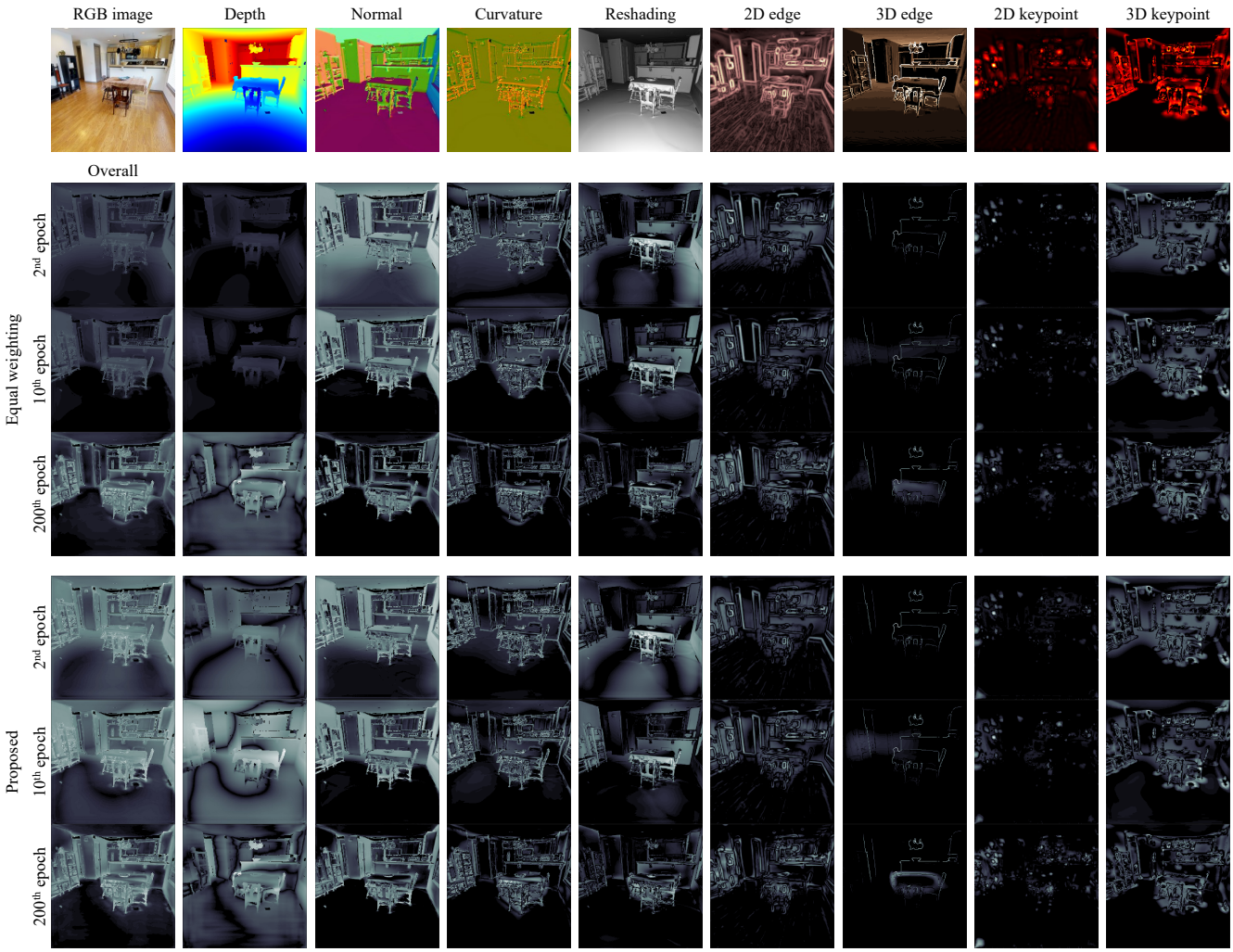
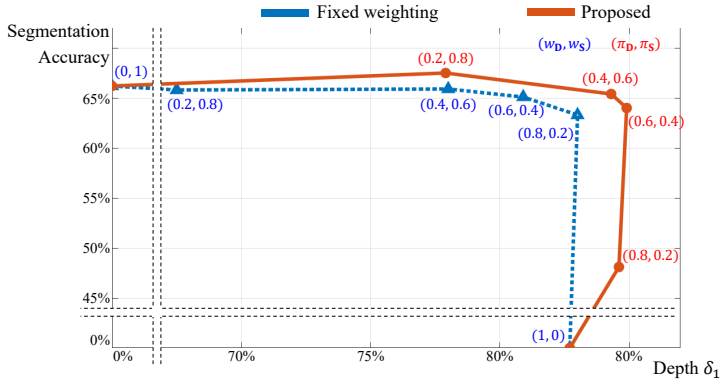


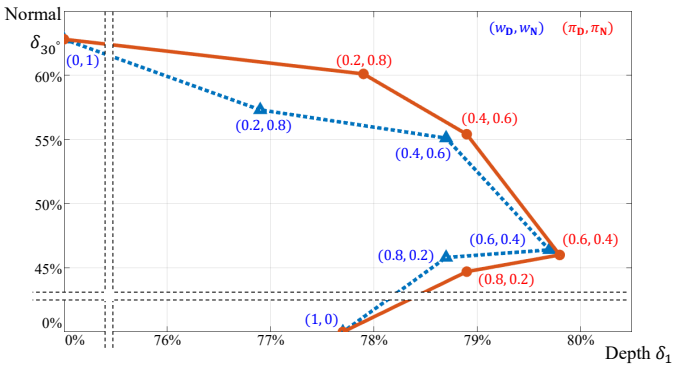
Figure S-5. Visualization of losses of the 8-task learning on the Taskonomy dataset.

7. Priority Factors π

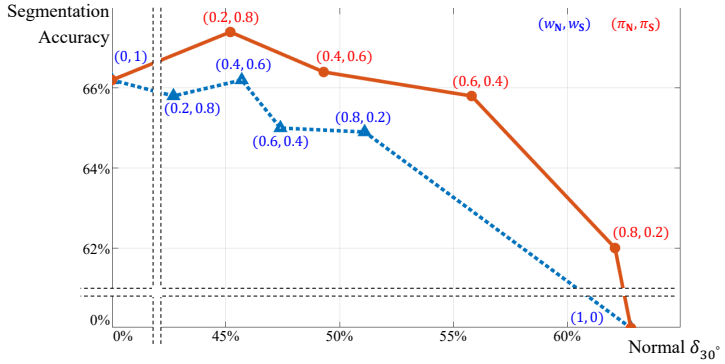
Using the PNAS-Net encoder and multi-column architecture on the NYUv2 dataset, we train the full-sharing network for double tasks in three combinations: depth estimation and segmentation, depth and normal estimation, and normal estimation and segmentation. Figure S-6 shows the performance of the proposed algorithm with different priority factors, compared to fixed weighting. For the double tasks of depth and segmentation in Figure S-6(a), as well as for normal and segmentation in Figure S-6(c), operating points of the proposed algorithm are located to the upper right side of those of fixed weighting, *i.e.* the proposed algorithm performs better than fixed weighting. Also, for the double tasks of depth and normal in Figure S-6(b), the proposed algorithm still outperforms fixed weighting meaningfully.



(a) Depth and segmentation



(b) Depth and normal



(c) Normal and segmentation

Figure S-6. Performance comparison of the proposed algorithm with fixed weighting for double-task learning. The coordinates represent priority factors (π_D, π_S), (π_D, π_N), and (π_N, π_S) or fixed weights (w_D, w_S), (w_D, w_N), and (w_N, w_S).

8. Qualitative Experimental Results

Figure S-7, Figure S-8, and Figure S-9 qualitatively compare the depth, normal, and segmentation results of the proposed algorithm and the conventional algorithms: equal weighting, Uncert [4], DWA [7], and GLS [1] for the four types of networks in Table 2 in the main paper. For easier comparison, estimates (odd rows) and their errors (even rows) are visualized. The proposed algorithm provides more accurate prediction results for all four networks.

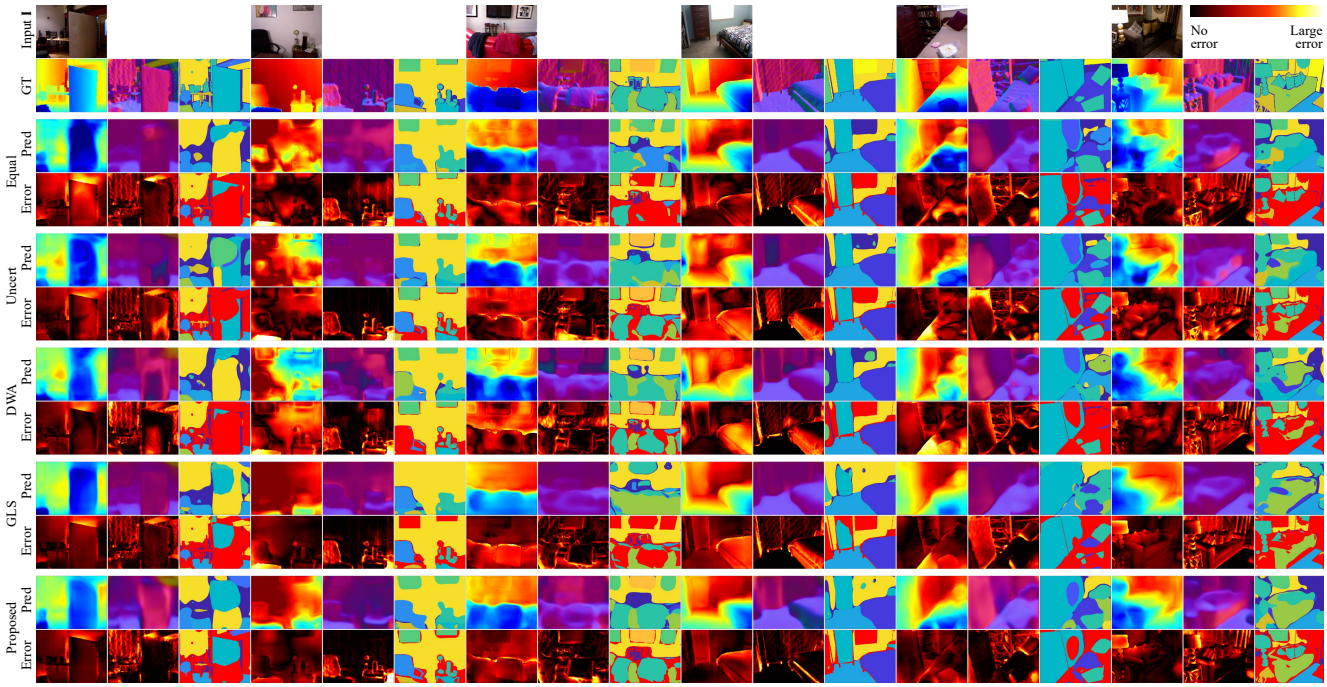


Figure S-7. Qualitative comparison of the proposed algorithm with conventional algorithms for the full-sharing network.

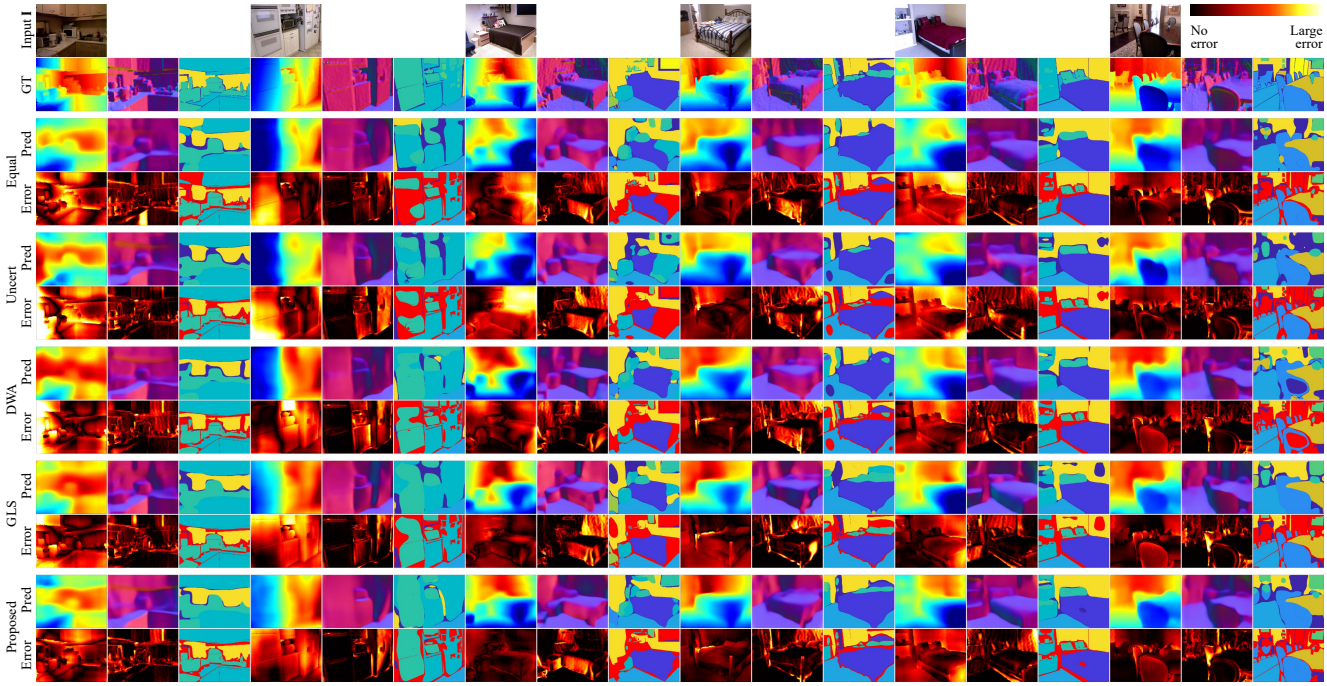


Figure S-8. Qualitative comparison of the proposed algorithm with conventional algorithms for the multi-decoder network.

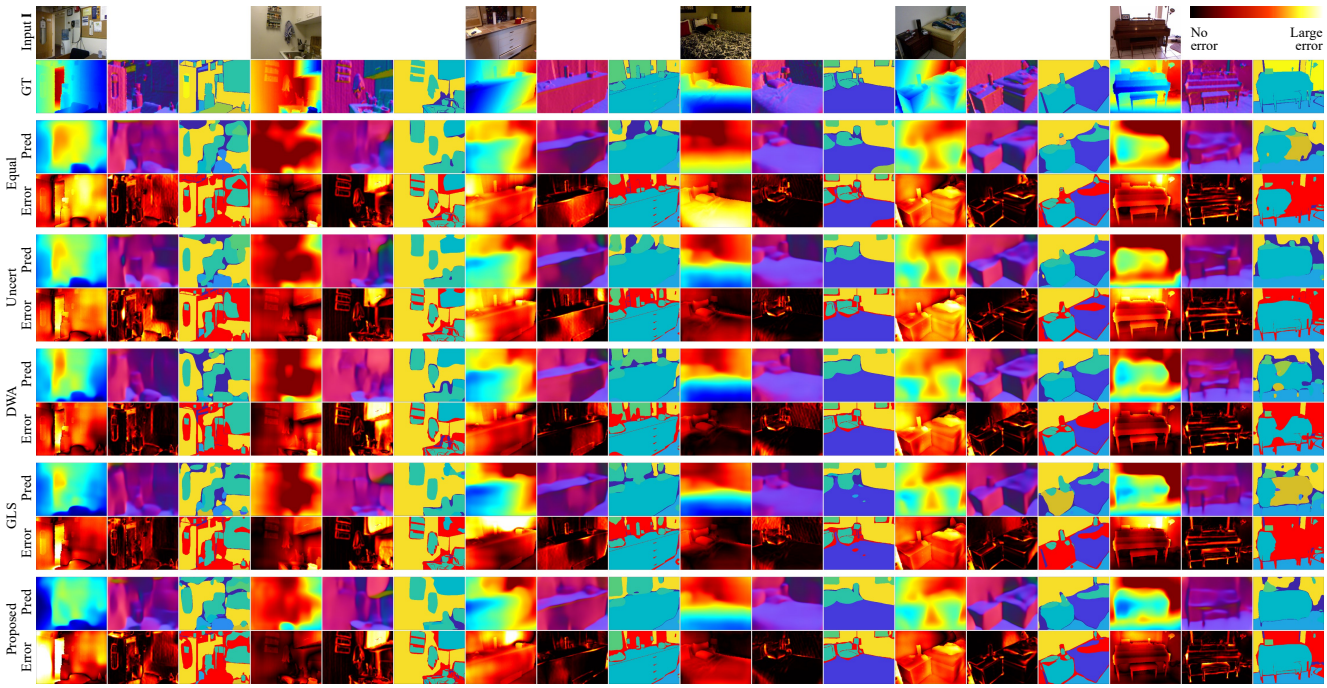


Figure S-9. Qualitative comparison of the proposed algorithm with conventional algorithms for the multi-column network.

References

[1] S. Chennupati, G. Sistu, S. Yogamani, and S. A Rawashdeh. MultiNet++: Multi-stream feature aggregation and geometric loss strategy for multi-task learning. In *CVPR Workshops*, 2019. 4, 5, 9

[2] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009. 3

[3] X Glorot and Y Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010. 3

[4] A. Kendall, Y. Gal, and R. Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*, 2018. 4, 9

[5] A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. *ACM Trans. Graph.*, 23(3):689–694, Aug. 2004. 3

[6] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L. J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy. Progressive neural architecture search. In *ECCV*, 2018. 1

[7] S. Liu, E. Johns, and A. J. Davison. End-to-end multi-task learning with attention. In *CVPR*, 2019. 4, 9

[8] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *ICLR*, 2018. 3

[9] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018. 1

[10] A. R. Zamir, A. Sax, W. Shen, L. J. Guibas, J. Malik, and S. Savarese. Taskonomy: Disentangling task transfer learning. In *CVPR*, 2018. 2