Multi-Loss Rebalancing Algorithm for Monocular Depth Estimation

Jae-Han Lee[0000−0002−3674−4023] and Chang-Su Kim[0000−0002−4276−1831]

School of Electrical Engineering, Korea University, Korea
jaehanlee@mcl.korea.ac.kr, changsukim@korea.ac.kr

Abstract. An algorithm to combine multiple loss terms adaptively for training a monocular depth estimator is proposed in this work. We construct a loss function space containing tens of losses. Using more losses can improve inference capability without any additional complexity in the test phase. However, when many losses are used, some of them may be neglected during training. Also, since each loss decreases at a different speed, adaptive weighting is required to balance the contributions of the losses. To address these issues, we propose the loss rebalancing algorithm that initializes and rebalances the weight for each loss function adaptively in the course of training. Experimental results show that the proposed algorithm provides state-of-the-art depth estimation results on various datasets. Codes are available at https://github.com/jaehanlee-mcl/multi-loss-rebalancing-depth.

Keywords: Monocular depth estimation, multi-loss rebalancing

1 Introduction

Monocular depth estimation is the task to estimate depth information of a scene from a single image. It is applicable to various higher-level vision tasks, since the depth information is essential for understanding 3D scene geometry. It is more challenging than the depth estimation using stereo images [38] or video frames [44] due to the lack of reliable cues. Early algorithms [7, 13, 26, 39] made assumptions, such as 'blocks world,' to make it easier. They yield unreliable results when the assumptions are invalid. After Eigen *et al.* [9] introduced a convolutional neural network (CNN) for monocular depth estimation, various CNN-based algorithms have been developed.

Recent advances in monocular depth estimation are due to better backbone networks [14, 17, 18, 42], huge RGBD datasets [4, 5, 11, 39, 41, 43, 45], richer labels for training [22, 37, 48], or sophisticated loss functions [5, 8–10, 16, 25]. Among them, the loss function design has two advantages. First, using a sophisticated loss can improve inference capability without requiring additional memory complexity for more complicated networks. Second, it may increase the computation time for training but does not affect the time complexity for testing. Existing algorithms adopt various loss functions, including scale-invariant loss [9], gradient loss [8], and normal loss [16]. They also formulate a loss function as a combination of several losses for more effective training.

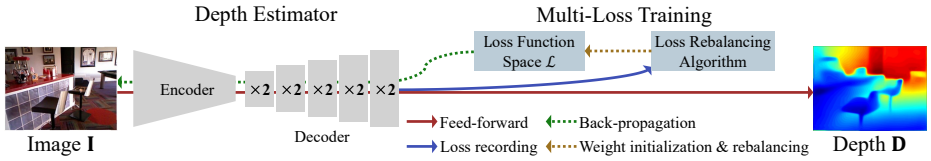

Fig. 1. Overview of the loss rebalancing algorithm: A depth estimator estimates a depth map **D** from an image **I**. Throughout its training, losses are recorded and their weights are initialized and adjusted by the loss rebalancing algorithm.

However, there is no systematic analysis of these loss functions, and there is a lack of understanding how these losses improve the performance. In this paper, we attempt to address this issue. We construct a loss function space, containing many loss terms: some are from existing algorithms, and the others are newly designed. Through extensive experiments and analysis, we find that, as more loss terms are used to train a depth estimator, the performance gets better. However, to exploit a large loss function space in training, two weighting issues should be addressed. First, each loss has a different order of magnitudes. Thus, some losses may be neglected, if the losses are simply summed up. Second, in the course of training, each loss decreases at a different speed. Hence, the contributions of the losses should be balanced periodically. Therefore, we propose the loss rebalancing algorithm that initializes and adjusts the weights for multiple losses adaptively, so that each loss contributes properly to the training of a depth estimator and thus improves the estimation performance. Fig. 1 shows an overview of the proposed loss rebalancing algorithm.

Extensive experimental results and ablation studies show that the proposed loss rebalancing algorithm improves the performances of monocular depth estimators meaningfully.

This paper has the following main contributions:

- For monocular depth estimation, we construct a loss function space of several tens of losses, and propose the loss rebalancing algorithm to utilize the loss function space effectively.
- The proposed algorithm improves depth estimation performance significantly, without requiring additional network parameters or inference time.
- The proposed depth estimators yield excellent results on the NYUv2 [41] and Make3D [39] datasets.

2 Related Work

Loss functions: To train monocular depth estimators based on deep learning, many loss functions have been proposed. *Depth losses* directly measure the differences between ground-truth depths and their estimates. Several depth losses are based on the L2-norm [2, 8, 9, 15, 31, 49], but there are different types of depth losses, such as the L1-norm [35], multinomial logistic loss [10, 28, 30, 36] and berHu loss [25].

Eigen *et al.* [9] pointed out that, in monocular depth estimation, it is ill-posed to estimate a global depth scale, and a significant amount of depth errors are caused by the mismatch between true and predicted scales. They hence proposed a *scale-invariant loss* to eliminate the impacts of global scales, which has been also adopted in [8,36].

Gradient losses and *normal losses* are also often used. A gradient loss penalizes errors, especially near object boundaries in a scene. It is measured from the differences between the derivatives of ground-truth and predicted depth maps. Some algorithms [8,12,16,22] use gradient losses jointly with depth losses. Chakrabarti *et al.* [3] generalized the gradient losses using higher-order derivatives. A normal loss imposes a penalty on mismatches between ground-truth and predicted normal vectors. It can be computed using normal vector labels of training data [37]. When they are unavailable, normal vectors can be approximated from depth maps [16]. In addition, the semantic loss [19], the pairwise loss [50], the SSIM loss [12] have been developed.

Even though various losses have been proposed to yield successful results, to the best of our knowledge, no paper analyzes the effectiveness of different losses systematically. Moreover, in most algorithms, the overall loss is defined as a weighted sum of only a few loss terms, and the weights are determined heuristically. In this work, we define a loss function space consisting of 78 losses. Then, we propose an effective algorithm to use all those losses and to initialize and adjust their weights.

Loss function balancing: Balancing between various loss functions has been considered in the multi-task learning field [6,21,40]. During the training of models, the weights of loss functions for various tasks, such as depth estimation, semantic segmentation, and surface normal estimation, are dynamically adjusted. Kendall *et al.* [21] proposed a weighting scheme based on the uncertainty of each loss function. They assumed that the tasks are independent of one another in their maximum likelihood formulation. Chen *et al.* [6] designed another weighting scheme, which balances the gradient magnitude for each task. Sener and Koltun [40] attempted to achieve a Pareto optimum between loss functions. Note that, in both [6] and [40], it was assumed that the objectives of multiple loss functions conflict with one another. All these algorithms are for multi-task learning, so they are fundamentally different from the proposed rebalancing algorithm of loss functions for a monocular depth estimator. Their assumptions are invalid in the proposed loss function space, because we derive all losses from the same depth map.

3 Proposed Algorithm

Fig. 1 is an overview of the proposed algorithm. The proposed depth estimator has the encoder-decoder architecture [1,47] similarly to most deep-learning-based depth estimators. The architecture is described in detail in Section 4.1. During

the training, losses are recorded by the loss rebalancing algorithm, which initializes and adjusts the weights of loss functions.

Let f denote the depth estimator network and $\boldsymbol{\theta}$ be its parameters. Given an image $\mathbf{I}$, f estimates a depth map $\widehat{\mathbf{D}} = f(\mathbf{I};\boldsymbol{\theta})$. The objective is to determine the optimal parameters to minimize the overall loss function ℓ_{all},

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{k:\mathbf{I}_k \in \mathcal{I}} \ell_{\text{all}}\left(f(\mathbf{I}_k;\boldsymbol{\theta}), \mathbf{D}_k\right) \tag{1}$$

where $\mathcal{I}$ is the set of training images, and $\mathbf{D}_k$ is the ground-truth depth map for the kth training image $\mathbf{I}_k$.

3.1 Loss function space

The overall loss function ℓ_{all} is defined as a weighted sum of multiple loss functions in the loss function space $\mathcal{L}$,

$$\ell_{\text{all}} = \sum_{i:\ell_i \in \mathcal{L}} w_i \ell_i \tag{2}$$

where ℓ_i denotes the ith loss function in $\mathcal{L}$ and w_i is the corresponding weight. Table 1 lists all 78 ($= 6 \times 13$) losses in $\mathcal{L}$. Given a depth map $\mathbf{D}^0$, we repeatedly halve its spatial resolution in both horizontal and vertical directions to yield downsampled depth maps $\mathbf{D}^1, \mathbf{D}^2, \mathbf{D}^3, \mathbf{D}^4, \mathbf{D}^5$. Here, superscripts denote spatial scales. Then, we compute 13 kinds of losses at each spatial scale. Let us describe each loss subsequently. For notational convenience, we omit the superscripts representing spatial scales.

Table 1. Loss functions, which compose the loss function space $\mathcal{L}$.

	Spatial scale					
	0	1	2	3	4	5
Depth losses	ℓ^0_{D}	ℓ^1_{D}	ℓ^2_{D}	ℓ^3_{D}	ℓ^4_{D}	ℓ^5_{D}
Mean-removed losses	ℓ^0_{M} ℓ^0_{M5} ℓ^0_{M17} ℓ^0_{M65}	ℓ^1_{M} ℓ^1_{M5} ℓ^1_{M17} ℓ^1_{M65}	ℓ^2_{M} ℓ^2_{M5} ℓ^2_{M17} ℓ^2_{M65}	ℓ^3_{M} ℓ^3_{M5} ℓ^3_{M17} ℓ^3_{M65}	ℓ^4_{M} ℓ^4_{M5} ℓ^4_{M17} ℓ^4_{M65}	ℓ^5_{M} ℓ^5_{M5} ℓ^5_{M17} ℓ^5_{M65}
Gradient losses	ℓ^0_{r} ℓ^0_{c} ℓ^0_{rr} ℓ^0_{rc} ℓ^0_{cc}	ℓ^1_{r} ℓ^1_{c} ℓ^1_{rr} ℓ^1_{rc} ℓ^1_{cc}	ℓ^2_{r} ℓ^2_{c} ℓ^2_{rr} ℓ^2_{rc} ℓ^2_{cc}	ℓ^3_{r} ℓ^3_{c} ℓ^3_{rr} ℓ^3_{rc} ℓ^3_{cc}	ℓ^4_{r} ℓ^4_{c} ℓ^4_{rr} ℓ^4_{rc} ℓ^4_{cc}	ℓ^5_{r} ℓ^5_{c} ℓ^5_{rr} ℓ^5_{rc} ℓ^5_{cc}
Normal losses	ℓ^0_{N} ℓ^0_{Nr} ℓ^0_{Nc}	ℓ^1_{N} ℓ^1_{Nr} ℓ^1_{Nc}	ℓ^2_{N} ℓ^2_{Nr} ℓ^2_{Nc}	ℓ^3_{N} ℓ^3_{Nr} ℓ^3_{Nc}	ℓ^4_{N} ℓ^4_{Nr} ℓ^4_{Nc}	ℓ^5_{N} ℓ^5_{Nr} ℓ^5_{Nc}

Let $\mathbf{D}$ be a ground-truth depth map and $\widehat{\mathbf{D}}$ be its estimate. The common depth loss ℓ_{D} is given by the entrywise L1-norm for a matrix,

$$\ell_{\mathrm{D}} = \frac{1}{HW} \|\widehat{\mathbf{D}} - \mathbf{D}\|_1 \tag{3}$$

where W and H are the width and height of the depth maps. We use the L1-norm for most losses, which is known to facilitate more efficient training than the L2-norm [35].

A mean-removed loss measures a difference between depth maps after removing depth scales. This loss function is similar to the scale-invariant term in [9]. First, the global-mean-removed loss ℓ_{M} is defined as

$$\ell_{\mathrm{M}} = \frac{1}{HW} \|(\widehat{\mathbf{D}} - \hat{\mu}) - (\mathbf{D} - \mu)\|_1 \tag{4}$$

where $\hat{\mu}$ and μ are the average depths in $\widehat{\mathbf{D}}$ and $\mathbf{D}$, respectively. This loss is based on the observation that, although it is ambiguous to estimate the global depth scale (*i.e.* average depth) from an image, the relative depth of each pixel with respect to the average depth can be predicted more reliably. Note that relative estimation is easier than absolute estimation in other applications as well, such as age estimation [32]. Similarly, a depth estimator should be capable of predicting whether a pixel within a region is farther or nearer than the average depth of the region. Thus, we introduce a local-mean-removed loss $\ell_{\mathrm{M}n}$, which penalizes the relative depth errors with respect to local $n \times n$ square regions,

$$\ell_{\mathrm{M}n} = \frac{1}{HW} \left\| \left(\widehat{\mathbf{D}} - \widehat{\mathbf{D}} \circledast \frac{\mathbf{J}_n}{n^2} \right) - \left(\mathbf{D} - \mathbf{D} \circledast \frac{\mathbf{J}_n}{n^2} \right) \right\|_1 \tag{5}$$

where $\circledast$ denotes the convolution, and $\mathbf{J}_n$ is the $n \times n$ matrix composed of all ones. As listed in Table 1, we consider three square sizes ($n = 5, 17, 65$).

Next, we use gradient losses [3, 8, 12, 22]. The gradient loss ℓ_{r} in the row direction is defined as

$$\ell_{\mathrm{r}} = \frac{1}{HW} \|\nabla_{\mathrm{r}}\widehat{\mathbf{D}} - \nabla_{\mathrm{r}}\mathbf{D}\|_1 \tag{6}$$

where the partial derivative ∇_{r} is implemented as the difference between horizontally adjacent pixels. Similarly, the gradient loss ℓ_{c} in the column direction and the 2nd-order derivatives ℓ_{rr}, ℓ_{rc}, and ℓ_{cc} are also defined.

We also use the normal loss ℓ_{N}. This loss may be computed using ground-truth normal vectors [8, 37]. However, we train a depth estimator using only depth labels. Thus, as in [16], we approximate the normal vector $\mathbf{n}_{ij}$ at (i, j) from the depth gradient by

$$\mathbf{n}_{ij} = [-\nabla_{\mathrm{r}}\mathbf{D}(i,j), -\nabla_{\mathrm{c}}\mathbf{D}(i,j), 1]^T. \tag{7}$$

Then, ℓ_{N} is defined using the cosine similarity by

$$\ell_{\mathrm{N}} = \frac{1}{HW} \sum_{i=1}^{H} \sum_{j=1}^{W} \left(1 - \frac{\widehat{\mathbf{n}}_{ij}^T \mathbf{n}_{ij}}{\|\widehat{\mathbf{n}}_{ij}\|_2 \|\mathbf{n}_{ij}\|_2} \right) \tag{8}$$

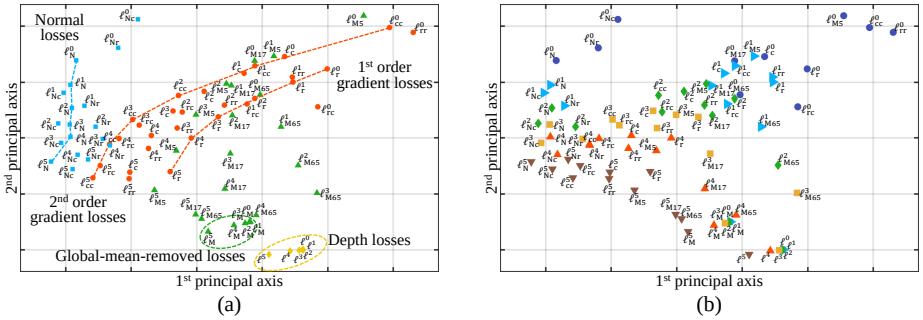

Fig. 2. Projection of the losses in $\mathcal{L}$ onto the plane, determined by the two principal axes. The losses are represented by different colors and marks according to (a) loss types and (b) spatial scales. Also, in (a), to facilitate observation, some losses in the same types are connected by dashed lines or enclosed by dashed ellipses.

We further introduce the 2nd-order normal losses ℓ_{Nr} and ℓ_{Nc}, by measuring the normal vectors of the partial derivative maps $\nabla_{\mathrm{r}}\mathbf{D}$ and $\nabla_{\mathrm{c}}\mathbf{D}$. Specifically, to compute ℓ_{Nr}, the second-order normal vector $[-\nabla_{\mathrm{r}}\nabla_{\mathrm{r}}\mathbf{D}, -\nabla_{\mathrm{c}}\nabla_{\mathrm{r}}\mathbf{D}, 1]^T$ is used instead of (7). Similarly, ℓ_{Nc} is computed.

Some loss functions in $\mathcal{L}$ exhibit similar tendencies during the training. To analyze the similarity or dissimilarity among the loss functions, we train a preliminary depth estimator and record the output values of each loss function ℓ_i in a vector $\mathbf{t}_i$. Then, we project these vectors $\mathbf{t}_i$ for $\ell_i \in \mathcal{L}$ onto the 2D plane, by performing the principal component analysis (PCA). Fig. 2 shows the projection results. In Fig. 2 (a), losses are represented in different colors and marks according to their types. The depth losses ℓ_{D} and the global-mean-removed losses ℓ_{M}, respectively, are tightly located regardless of the spatial scales. In contrast, the gradient losses, especially the 2nd-order ones ℓ_{rr} and ℓ_{cc}, exhibit different characteristics according to the spatial scales and are located far from one another in the function space. In Fig. 2 (b), we also classify the losses by their spatial scales. Except for the depth losses ℓ_{D} and the global-mean-removed losses ℓ_{M}, each type of losses are widely distributed in the space according to the scales.

3.2 Loss rebalancing algorithm

We use the loss functions in $\mathcal{L}$ to train a monocular depth estimator. Those functions address different aspects of depth structures. Combining these diverse loss functions improves the inference capability of the trained estimator. As compared to using a single loss function, the proposed approach increases the training time but does not require any additional complexity in the test phase; after training, it demands the same time or memory complexity to estimate the depth map of a test image.

There are two weighting issues to be addressed.

- How to initialize the weight w_i for each $\ell_i \in \mathcal{L}$ in (2)?

- In the course of training, how to adjust w_i for ℓ_i, whose output values decrease at a different speed from the other loss functions? For example, if ℓ_i is a relatively easy optimization function and decreases much faster than the others, should we increase or decrease w_i?

To solve these issues, we propose weight initialization and rebalancing schemes.

Weight initialization: Each loss function ℓ_i yields output values, which may have different orders of magnitudes from those of the other loss functions. This is because ℓ_i is computed at a different spatial scale, or it deals with a different feature. Thus, to balance the contribution of each ℓ_i to the overall loss function ℓ_{all}, we train the network preliminarily after setting each weight equally to $\frac{1}{|\mathcal{L}|}$ and record the average output $\bar{\ell}_i$ of ℓ_i. Then, we initialize the weight w_i so that the contribution $w_i\bar{\ell}_i$ to the overall loss $\bar{\ell}_{\text{all}}$ is identical for every i. In other words, we initially set

$$w_i^{(0)} = \frac{\bar{\ell}_{\text{all}}}{\bar{\ell}_i} \qquad \text{for each } \ell_i \in \mathcal{L}, \tag{9}$$

where the superscripts (0) mean that it is the initial weight.

Weight rebalancing: During the training, we adjust the weights periodically. Suppose that a period consists of N training images. Let $L_i^{(t)}$ denote the sum of the N losses, generated by ℓ_i in period t. Also, let $w_i^{(t-1)}$ be the weight for $L_i^{(t)}$, which is determined after the previous period $t-1$ and used for the current period t. Then, the overall loss $L_{\text{all}}^{(t)}$ in period t is given by

$$L_{\text{all}}^{(t)} = \sum_i w_i^{(t-1)} L_i^{(t)}. \tag{10}$$

After period t, we update weight $w_i^{(t)}$ by comparing $L_i^{(t)}$ with $L_i^{(t-1)}$. Let $P_i^{(t)} = L_i^{(t)}/L_{\text{all}}^{(t)}$ be the ratio of the ith loss $L_i^{(t)}$ to the overall loss $L_{\text{all}}^{(t)}$. We compute the change in the ratio by $\Delta P_i^{(t)} = P_i^{(t)} - P_i^{(t-1)}$. Then, we adjust the weight $w_i^{(t)}$ via

$$w_i^{(t)} = w_i^{(t-1)} \left(1 - \lambda \times \frac{\Delta P_i^{(t)}}{P_i^{(t)}}\right) \tag{11}$$

where λ is called the rebalancing parameter. This rebalancing equation has the following properties:

- If $\lambda = 0$, then $w_i^{(t)} = w_i^{(t-1)}$. Thus, the initial weights remain unchanged without rebalancing.
- If $\lambda = 1$, (11) can be rewritten as

$$w_i^{(t)} P_i^{(t)} = w_i^{(t-1)} P_i^{(t-1)}, \tag{12}$$

which means that the weight is updated to maintain the contribution of ith loss to the overall loss at the same level between the two periods $t-1$ and t.

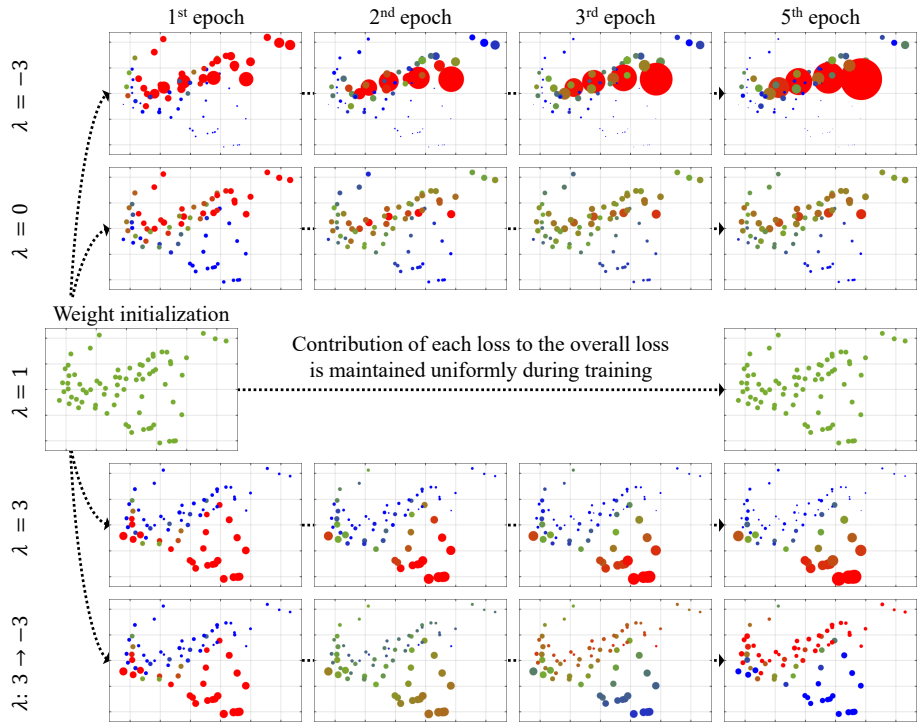

Fig. 3. Visualizing the contributions of losses according to the scheduling of λ: The radius of each circle represents the magnitude of $w_i^{(t)} P_i^{(t)}$ for the corresponding loss. If the magnitude decreases as compared to the previous epoch, the circle is in blue. If it increases, the circle is in red. Green means no change. Please refer to Fig. 2 to see which loss is indicated by each circle.

- In general, if $\lambda > 0$, the weight for an 'easy' loss function increases. Suppose that a certain loss function ℓ_i is easier to train than the others. Then, its current percentage $P_i^{(t)}$ is lower than the previous $P_i^{(t-1)}$, resulting in a negative $\Delta P_i^{(t)}$. Then, $w_i^{(t)} > w_i^{(t-1)}$ from (11). Therefore, ℓ_i is multiplied with a bigger weight in the next period. Hence, a positive λ encourages the training algorithm to focus on easy loss terms.
- On the contrary, if $\lambda < 0$, the training focuses on 'difficult' loss terms.
- It can be shown that $\sum_i w_i^{(t-1)} L_i^{(t)} = \sum_i w_i^{(t)} L_i^{(t)}$ for all t. Thus, rebalancing in (11) is done so that the overall loss $L_{\text{all}}^{(t)}$ in (10) is maintained when $w_i^{(t-1)}$ is replaced with $w_i^{(t)}$. This makes the monitoring of overall losses over periods easier, since the scale of an overall loss is not affected by the rebalancing.

We can start with a positive λ to reduce easy losses more quickly and teach the network to learn uncomplicated depth structures in scenes first. Then, we

can gradually reduce λ to teach the network hard-to-learn structures. In the default mode, we start with $\lambda = 3$ and monotonically decrease it to -3.

Fig. 3 visualizes the contributions of losses according to the scheduling of λ. The radius of each circle indicates the magnitude of $w_i^{(t)} P_i^{(t)}$ for the corresponding loss, and its color represents the increase or decrease of the magnitude. Losses with increased contributions (*i.e.* magnitudes), as compared to the previous epoch, are in red, while those with decreased contributions are in blue. The following observations can be made from Fig. 3.

- ($\lambda = 0$) The loss rebalancing is not applied. Difficult loss terms, such as ℓ_{rr}^0 and ℓ_{cc}^0, occupy more portions of contributions as the training goes on.
- ($\lambda = 1$) The contribution of each loss is maintained at the same level.
- ($\lambda = 3$) Easy losses are emphasized as the training goes on. We see that ℓ_{D} and ℓ_{M} at all scales and all types of coarse scale losses are on the easier side.
- ($\lambda = -3$) Difficult losses are emphasized. Most fine-scale losses are difficult.
- ($\lambda : 3 \rightarrow -3$) By decreasing λ from 3 to -3, the training focuses on easy loss terms first and on difficult ones later.

In Section 4, it is shown experimentally that the default mode ($\lambda : 3 \rightarrow -3$) provides the best depth estimation results.

4 Experimental Results

4.1 Implementation details

Encoder-decoder architecture: We adopt the encoder-decoder architecture [1, 47] for the monocular depth estimator. We design the network structure of the depth estimator in the simplest way, so that the proposed multi-loss rebalancing algorithm can be applied to other existing or future backbone networks with minimal modifications.

For the encoder, we test two backbones, DenseNet161 [18] and PNASNet [33], after removing their classification layers. The spatial resolution of an input image to the encoder is 288×384, while that of the encoder output is 9×12. Thus, the encoder reduces the spatial resolution by a factor of $\frac{1}{2^5}$ both horizontally and vertically. The decoder includes 5 up-sampling blocks, which expand the encoder output to the resolution of $\mathbf{D}^0$. We describe the decoder structure in detail in the supplemental document.

Network training: For the performance comparison, we train the proposed networks using the Adam optimizer [23] for 20 epochs with a learning rate of 10^{-4}. The batch size is set to 12 and 8 for the DenseNet-based and PNAS-based networks, respectively. We perform the weight initialization in (9) after the preliminary training of 1/4 epoch. Next, the weight rebalancing in (11) is performed periodically every 1/4 epoch. In the default mode, the rebalancing parameter λ is initialized to 3 at the first epoch and gradually decreased to -3 until the fifth epoch. Also, we adopt the augmentation policy of [16].

Table 2. Evaluation metrics for estimated depth maps: $\hat{d}_i$ and d_i denote estimated and ground-truth depths of pixel i, respectively, and N is the number of pixels. Also, l_{ij} has a value of 1 or -1 depending on the relative relation between d_i and d_j.

Metrics for ordinary depth estimation	
δ_n	% of d_i such that $\max\{\frac{\hat{d}_i}{d_i}, \frac{d_i}{\hat{d}_i}\} < 1.25^n$
RMSE_{lin}	$(\frac{1}{N}\Sigma_i(\hat{d}_i - d_i)^2)^{0.5}$
ARD	$\frac{1}{N}\Sigma_i\lvert\hat{d}_i - d_i\rvert/d_i$
log10	$\frac{1}{N}\Sigma_i\lvert\log_{10}\hat{d}_i - \log_{10} d_i\rvert$
RMSE_{log}	$(\frac{1}{N}\Sigma_i(\log\hat{d}_i - \log d_i)^2)^{0.5}$
RMSE_{si}	RMSE (log) with global scale removed
SRD	$\frac{1}{N}\Sigma_i\lvert\hat{d}_i - d_i\rvert^2/d_i$

Relative depth perception: We apply the proposed algorithm to relative depth perception [5, 45, 52] as well. By fusing relative depth maps with ordinary ones, we can improve the depth estimation performance further. We include the training details and experiment results for the relative depth estimator in the supplemental document.

4.2 Datasets and evaluation metrics

We use two depth datasets: NYUv2 [41] for indoor scene and Make3D [39] for outdoor scene. Ground-truth depth maps have blank areas with missing depth. For training, we fill in the incomplete depth maps using the colorization scheme [29], as done in [41].

For experiments on NYUv2, we develop two depth estimators: 'Proposed (Dense)' and 'Proposed (PNAS)'. Both estimators are trained using the sequences of the training split in [41]. The two estimators differ only in the encoder backbones, and their training details are the same. For experiments on Make3D, we train 'Proposed (PNAS)' using the Make3D training data.

Table 2 lists the evaluation metrics. We follow the evaluation protocol of [10].

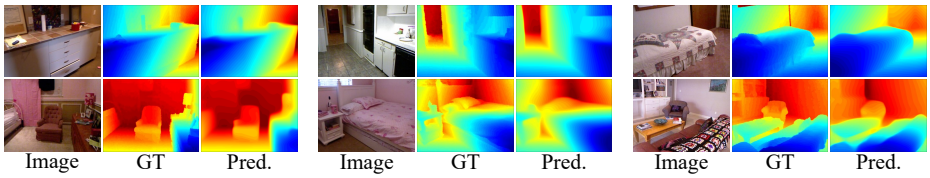

Fig. 4. Qualitative results. Farther depths are in red, while closer ones in blue.

Table 3. Performance comparison on NYUv2 [41]. The best results are boldfaced.

	The higher, the better			The lower, the better					
	δ_1	δ_2	δ_3	RMSE_{lin}	ARD	log10	RMSE_{log}	RMSE_{si}	SRD
Eigen *et al.* [9]	61.1%	88.7%	97.1%	0.907	0.215	-	0.285	0.219	0.212
Li *et al.* [31]	62.1%	88.6%	96.8%	0.821	0.232	0.094	-	-	-
Eigen and Fergus [8]	76.9%	95.0%	98.8%	0.641	0.158	-	0.214	0.171	0.121
Chakrabarti *et al.* [3]	80.6%	95.8%	98.7%	0.620	0.149	-	0.205	-	0.118
Laina *et al.* [25]	81.1%	95.3%	98.8%	0.573	0.127	0.055	0.195	-	-
Xu *et al.* [46]	81.1%	95.4%	98.7%	0.586	0.121	0.052	-	-	-
Lee *et al.* [27]	81.5%	96.3%	99.1%	0.572	0.139	-	0.193	-	0.096
Fu *et al.* [10]	82.8%	96.5%	99.2%	0.509	0.115	0.051	-	-	-
Kundu *et al.* [24]	85.6%	96.6%	99.1%	0.506	0.114	0.046	-	-	-
Zhang *et al.* [51]	81.5%	96.2%	99.2%	0.501	0.144	-	0.181	-	-
Lee and Kim [28]	83.7%	97.1%	**99.4%**	0.538	0.131	-	0.180	0.148	0.087
Zhang *et al.* [50]	84.6%	96.8%	**99.4%**	0.497	0.121	-	0.175	-	-
Hu *et al.* [16]	86.6%	**97.5%**	99.3%	0.530	**0.115**	**0.050**	-	-	-
Proposed (Dense)	85.0%	96.9%	99.2%	0.457	0.127	0.053	0.160	0.128	0.088
Proposed (PNAS)	**87.0%**	97.4%	99.3%	**0.430**	0.119	**0.050**	**0.151**	**0.123**	**0.078**

4.3 Comparison with conventional algorithms

Table 3 compares the performances on NYUv2. Even 'Proposed (Dense)' outperforms all conventional algorithms by a large margin in terms of RMSE_{lin} and RMSE_{log}, and provides comparable or better results in terms of other metrics. In particular, [28] adopts the same DenseNet as the backbone, but its decoder structure and post-processing scheme are much more complicated that those of 'Proposed (Dense)'. Nevertheless, 'Proposed (Dense)' outperforms [28] in most metrics, which means that the proposed loss rebalancing algorithm improves the depth estimation performance effectively. Moreover, 'Proposed (PNAS)' outperforms all algorithms in most metrics. Fig. 4 shows examples of depths maps, predicted by 'Proposed (PNAS)'.

Table 4. Comparison on Make3D [39].

	Evaluated in 0-70m depth range		
	RMSE_{lin}	ARD	log10
Karsch *et al.* [20]	15.10	0.361	0.148
Liu *et al.* [34]	12.89	0.307	0.125
Kundu *et al.* [24]	9.56	0.452	-
Xu *et al.* [46]	8.56	0.198	-
Fu *et al.* [10]	7.32	**0.162**	**0.067**
Proposed	**5.87**	0.231	0.082

Table 4 shows that the proposed algorithm also provides excellent performance on the outdoor dataset Make3D. We see that the proposed algorithm outperforms the existing algorithms in terms of RMSE_{lin}. The supplemental document shows that the proposed algorithm yields competitive results on the KITTI dataset [11] as well.

4.4 Ablation studies

We analyze the depth estimation performance of the proposed algorithm on the NYUv2 test split in Table 5. Here, $\mathcal{L}_{\mathrm{D}}$. $\mathcal{L}_{\mathrm{M}}$, $\mathcal{L}_{\mathrm{G}}$, and $\mathcal{L}_{\mathrm{N}}$ denote the sets of depth, mean-removed, gradient, and normal losses in Table 1, respectively. Also, $\mathcal{L}^k$ denotes the set of losses at spatial scale k. Each model is trained for 5 epochs using three training datasets [4,41,43]. Since the network training has stochastic properties, we show the average performance of the three models for each setting for more reliable comparison.

Loss function space: From Part (a), the following observations are made.

- $\{\ell_{\mathrm{D}}^0, \ell_{\mathrm{M}}^0, \ell_{\mathrm{r}}^0, \ell_{\mathrm{c}}^0\}$ and $\{\ell_{\mathrm{D}}^0, \ell_{\mathrm{r}}^0, \ell_{\mathrm{c}}^0, \ell_{\mathrm{N}}^0\}$ correspond to the loss sets of [8] and [16], respectively. They make small improvements as compared with the baseline using only ℓ_{D}^0.
- By employing multi-scale losses, greater performance gains are achieved. It is most effective to combine gradient losses with depth losses ($\mathcal{L}_{\mathrm{D}} \cup \mathcal{L}_{\mathrm{G}}$). This is presumably because gradient losses are more widely distributed in the loss function space in Fig. 2 than mean-removed or normal losses are.
- Coarse-scale losses ($\mathcal{L}^3 \cup \mathcal{L}^4 \cup \mathcal{L}^5$) are more effective than fine-scale ones ($\mathcal{L}^0 \cup \mathcal{L}^1 \cup \mathcal{L}^2$). Using the finest-scale set $\mathcal{L}^0$ only is the least reliable.

Loss rebalancing: Part (b) demonstrates the effectiveness of the proposed loss rebalancing algorithm. Compared with the corresponding settings in Part (a) using the same loss functions, the improved scores are in red, while the worsened ones in blue. As mentioned above, coarse-scale losses are more important for reliable depth estimation. Also, as shown in the bottom row in Fig. 3, the loss rebalancing algorithm emphasizes these important losses at the beginning of training. Therefore, in all cases including coarse-scale losses, the loss rebalancing algorithm improves the depth estimation performances. On the other hand, if only fine-scale losses are used ($\mathcal{L}^0$ or $\mathcal{L}^0 \cup \mathcal{L}^1 \cup \mathcal{L}^2$), the loss rebalancing algorithm becomes ineffective. However, by comparing the results of the entire loss function space $\mathcal{L}$ to those of $\mathcal{L}^3 \cup \mathcal{L}^4 \cup \mathcal{L}^5$, we see that these fine-scale losses also contribute to the performance improvement.

Scheduling of λ: Part (c) compares the performance according to the scheduling of λ in (11). As discussed in Section 3.2, a bigger λ focuses more on easy losses, such as depth, global-mean-removed, and other coarse-scale ones. Note that $\lambda = 3$ provides better results than $\lambda = -3$. However, the settings using decreasing λ provide better results. This implies that focusing on easy losses first and on difficult ones later is an effective strategy. Among all settings, the default setting $\lambda : 3 \rightarrow -3$ provides the best results.

Weight initialization: The first two rows in Part (d) show that the weight initialization in itself does not contribute to the training. However, prior to the weight rebalancing, it is essential to equalize the contribution of losses. If each loss has a different magnitudes, the ratio $P_i^{(t)}$ and its change $\Delta P_i^{(t)}$ in (11) are not meaningful, invalidating the rebalancing algorithm. Thus, the rebalancing without the initialization yields unreliable results in the third row of Part (d).

Table 5. Ablation studies using various settings: 'I' means the weight initialization and 'R' means the weight rebalancing. The best results are boldfaced. In Part (b), the improved results (compared to Part (a)) are in red, while the degraded ones in blue.

	# Losses	I	R	λ	δ_1	RMSE_{lin}	RMSE_{log}	ARD
(a) Combination of loss functions								
$\{\ell_{\text{D}}^0\}$	1	-	-	0	84.8%	0.474	0.164	0.128
$\{\ell_{\text{D}}^0, \ell_{\text{M}}^0, \ell_{\text{r}}^0, \ell_{\text{c}}^0\}$	4	-	-	0	85.6%	0.456	0.159	0.124
$\{\ell_{\text{D}}^0, \ell_{\text{r}}^0, \ell_{\text{c}}^0, \ell_{\text{N}}^0\}$	4	-	-	0	84.8%	0.472	0.164	0.127
$\mathcal{L}_{\text{D}} \cup \mathcal{L}_{\text{N}}$	24	-	-	0	86.3%	0.448	0.155	0.121
$\mathcal{L}_{\text{D}} \cup \mathcal{L}_{\text{M}}$	30	-	-	0	85.9%	0.455	0.157	0.124
$\mathcal{L}_{\text{D}} \cup \mathcal{L}_{\text{G}}$	36	-	-	0	86.5%	0.445	0.153	**0.117**
$\mathcal{L}^0$	13	-	-	0	85.3%	0.462	0.160	0.132
$\mathcal{L}^0 \cup \mathcal{L}^1 \cup \mathcal{L}^2$	39	-	-	0	85.7%	0.454	0.157	0.122
$\mathcal{L}^3 \cup \mathcal{L}^4 \cup \mathcal{L}^5$	39	-	-	0	**86.6%**	0.444	**0.153**	0.119
$\mathcal{L}$	78	-	-	0	86.4%	**0.440**	**0.153**	0.122
(b) Effectiveness of loss rebalancing algorithm								
$\mathcal{L}_D \cup \mathcal{L}_{\text{N}}$	24	✓	✓	$3 \rightarrow -3$	86.3%	0.440	0.153	0.119
$\mathcal{L}_D \cup \mathcal{L}_{\text{M}}$	30	✓	✓	$3 \rightarrow -3$	86.3%	0.446	0.154	0.119
$\mathcal{L}_D \cup \mathcal{L}_{\text{G}}$	36	✓	✓	$3 \rightarrow -3$	86.9%	0.437	0.151	0.118
$\mathcal{L}^0$	13	✓	✓	$3 \rightarrow -3$	84.8%	0.466	0.161	0.126
$\mathcal{L}^0 \cup \mathcal{L}^1 \cup \mathcal{L}^2$	39	✓	✓	$3 \rightarrow -3$	85.5%	0.456	0.158	0.123
$\mathcal{L}^3 \cup \mathcal{L}^4 \cup \mathcal{L}^5$	39	✓	✓	$3 \rightarrow -3$	86.7%	0.443	0.152	**0.116**
$\mathcal{L}$	78	✓	✓	$3 \rightarrow -3$	**87.0%**	**0.434**	**0.150**	0.117
(c) Scheduling of λ								
$\mathcal{L}$	78	✓	✓	-3	84.7%	0.459	0.161	0.130
		✓	-	0	86.5%	0.443	0.154	0.118
		✓	✓	1	86.5%	0.442	0.154	0.123
		✓	✓	3	86.6%	0.445	0.153	0.117
		✓	✓	$2 \rightarrow -2$	86.8%	0.439	0.151	**0.116**
		✓	✓	$3 \rightarrow -3$	**87.0%**	**0.434**	**0.150**	0.117
		✓	✓	$5 \rightarrow -5$	86.4%	0.445	0.153	0.121
(d) Necessity of weight initialization								
$\mathcal{L}$	78	-	-	0	86.4%	0.440	0.153	0.122
		✓	-	0	86.5%	0.443	0.154	0.118
		-	✓	$3 \rightarrow -3$	85.6%	0.459	0.157	0.127
		✓	✓	$3 \rightarrow -3$	**87.0%**	**0.434**	**0.150**	**0.117**
(e) Another weighting algorithm [21]								
$\mathcal{L}$	78		-		85.6%	0.455	0.160	0.125

Table 6. Performance comparison of the proposed depth estimators using different backbones on NYUv2 [41].

	The higher, the better			The lower, the better					
	δ_1	δ_2	δ_3	RMSE_{lin}	ARD	log10	RMSE_{log}	RMSE_{si}	SRD
VGG16	77.2%	95.0%	99.0%	0.544	0.160	0.067	0.196	0.160	0.117
ResNet50	82.4%	96.3%	99.1%	0.482	0.138	0.058	0.174	0.142	0.094
SENet154	87.1%	97.5%	99.4%	0.426	0.116	0.049	0.149	0.123	0.074

Weighting algorithm [21]: Part (e) replaces the proposed loss rebalancing algorithm with the weighting scheme in [21]. Their maximum likelihood formulation assumes the tasks are independent of one another. This assumption is invalid in $\mathcal{L}$ because all losses are derived from the same depth map. Hence, their scheme performs worse than the proposed algorithm. Similarly to the first row of Fig. 3, [21] emphasizes difficult losses.

4.5 Different backbone networks

We verify that the proposed algorithm is effective regardless of a backbone network. In Table 6, We replace the encoder backbone with widely-used networks: VGG16 [42], ResNet50 [14], and SENet154 [17]. By comparing Table 6 to Table 3, we see that, for each backbone, the proposed algorithm outperforms the conventional algorithms using the same backbone. For instance, [8], [25], and [16] use VGG16, ResNet50, and SENet154 as their backbones, respectively.

Table 7. Training and testing times of the proposed algorithm on NYUv2 [41].

Training	$\{\ell_{\mathrm{D}}^0\}$	$\mathcal{L}^3 \cup \mathcal{L}^4 \cup \mathcal{L}^5$	$\mathcal{L}$	Testing	
s/iter	1.63	1.70	1.97	s/scene	0.047

4.6 Time complexity

Table 7 analyzes the complexity of the proposed algorithm in training and testing. The experiments are done with a TITAN X GPU. For training, using more losses increases the training time. However, compared to the use of ℓ_{D}^0 alone, the training time increases only 4% and 21% for $\mathcal{L}^3 \cup \mathcal{L}^4 \cup \mathcal{L}^5$ and $\mathcal{L}$, respectively, even though the number of losses increases from 1 to 39 and 78. Regardless of the loss setting, proposed algorithm requires the same inference time.

5 Conclusions

For monocular depth estimation, we constructed a loss function space of diverse loss terms and showed that the proposed space improves the depth estimation accuracy without increasing the network complexity or inference time. Also, we proposed the loss rebalancing algorithm to make each loss term contribute to the training in a balanced manner. Experimental results showed that the proposed depth estimators achieve excellent performances on various datasets.

Acknowledgment

This work was conducted by Center for Applied Research in Artificial Intelligence (CARAI) grant funded by Defense Acquisition Program Administration (DAPA) and Agency for Defense Development (ADD) (UD190031RD).

References

1. Badrinarayanan, V., Kendall, A., Cipolla, R.: SegNet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **39**(12), 2481–2495 (Dec 2017) 3, 9
2. Can, Y., Xu, X., Sun, W., Lin, L.: Monocular depth estimation with ainity, vertical pooling, and label enhancement. In: ECCV (2018) 2
3. Chakrabarti, A., Shao, J., Shakhnarovich, G.: Depth from a single image by harmonizing overcomplete local network predictions. In: NIPS (2016) 3, 5, 11
4. Chang, A., Dai, A., Funkhouser, T.A., Halber, M., Niebner, M., Savva, M., Song, S., Zeng, A., Zhang, Y.: Matterport3D: Learning from RGB-D data in indoor environments. In: 3DV (2018) 1, 12
5. Chen, W., Fu, Z., Yang, D., Deng, J.: Single-image depth perception in the wild. In: NIPS (2016) 1, 10
6. Chen, Z., Badrinarayanan, V., Lee, C.Y., Rabinovich, A.: GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In: ICML (2018) 3
7. Delage, E., Lee, H., Ng, A.Y.: A dynamic Bayesian network model for autonomous 3D reconstruction from a single indoor image. In: CVPR (2006) 1
8. Eigen, D., Fergus, R.: Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In: ICCV (2015) 1, 2, 3, 5, 11, 12, 14
9. Eigen, D., Puhrsch, C., Fergus, R.: Depth map prediction from a single image using a multi-scale deep network. In: NIPS (2014) 1, 2, 3, 5, 11
10. Fu, H., Gong, M., Wang, C., Batmanghelich, K., Tao, D.: Deep ordinal regression network for monocular depth estimation. In: CVPR (2018) 1, 2, 10, 11
11. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The KITTI dataset. Int. J. Robot. Res. **32**(11), 1231–1237 (Sep 2013) 1, 11
12. Godard, C., Aodha, O.M., Brostow, G.J.: Unsupervised monocular depth estimation with left-right consistency. In: CVPR (2017) 3, 5
13. Gupta, A., Efros, A., Hebert, M.: Blocks world revisited: Image understanding using qualitative geometry and mechanics. In: ECCV (2010) 1
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016) 1, 14
15. Heo, M., Lee, J., Kim, K.R., Kim, C.S.: Monocular depth estimation using whole strip masking and reliability-based refinement. In: ECCV (2018) 2
16. Hu, J., Ozay, M., Zhang, Y., Okatani, T.: Revisiting single image depth estimation: Toward higher resolution maps with accurate object boundaries. In: WACV (2019) 1, 3, 5, 9, 11, 12, 14
17. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: CVPR (2018) 1, 14
18. Huang, G., Liu, Z., van der Maaten, L.: Densely connected convolutional networks. In: CVPR (2017) 1, 9
19. Jiao, J., Cao, Y., Song, Y., Lau, R.: Look deeper into depth: Monocular depth estimation with semantic booster and attention-driven loss. In: ECCV (2018) 3
20. Karsch, K., Liu, C., Kang, S.B.: Depth transfer: Depth extraction from video using non-parametric sampling. IEEE Trans. Pattern Anal. Mach. Intell. **36**(11), 2144–2158 (Oct 2014) 11
21. Kendall, A., Gal, Y., Cipolla, R.: Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In: CVPR (2018) 3, 13, 14

22. Kim, S., Park, K., Sohn, K., Lin, S.: Unified depth prediction and intrinsic image decomposition from a single image via joint convolutional neural fields. In: ECCV (2016) 1, 3, 5
23. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR (2015) 9
24. Kundu, J.N., Uppala, P.K., Pahuja, A., Babu, R.V.: AdaDepth: Unsupervised content congruent adaptation for depth estimation. In: CVPR (2018) 11
25. Laina, I., Rupprecht, C., Belagiannis, V., Tombari, F., Navab, N.: Deeper depth prediction with fully convolutional residual networks. In: 3DV (2016) 1, 2, 11, 14
26. Lee, D.C., Gupta, A., Hebert, M., Kanade, T.: Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. In: NIPS (2010) 1
27. Lee, J.H., Heo, M., Kim, K.R., Kim, C.S.: Single-image depth estimation based on Fourier domain analysis. In: CVPR (2018) 11
28. Lee, J.H., Kim, C.S.: Monocular depth estimation using relative depth maps. In: CVPR (2019) 2, 11
29. Levin, A., Lischinski, D., Weiss, Y.: Colorization using optimization. ACM Trans. Graph. **23**(3), 689–694 (Aug 2004) 10
30. li, B., Dai, Y., He, M.: Monocular depth estimation with hierarchical fusion of dilated CNNs and soft-weighted-sum inference. Pattern Recognit. **83**, 328–339 (Nov 2018) 2
31. Li, B., Shen, C., Dai, Y., van den Hengel, A., He, M.: Depth and surface normal estimation from monocular images using regression on deep features and hierarchical CRFs. In: CVPR (2015) 2, 11
32. Lim, K., Shin, N.H., Lee, Y.Y., Kim, C.S.: Order learning and its application to age estimation. In: ICLR (2020) 5
33. Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.J., Fei-Fei, L., Yuille, A., Huang, J., Murphy, K.: Progressive neural architecture search. In: ECCV (2018) 9
34. Liu, F., Shen, C., Lin, G., Reid, I.: Learning depth from single monocular images using deep convolutional neural fields. IEEE Trans. Pattern Anal. Mach. Intell. **38**(10), 2024–2039 (Oct 2016) 11
35. Ma, F., Karaman, S.: Sparse-to-Dense: Depth prediction from sparse depth samples and a single image. In: ICRA (2018) 2, 5
36. Mousavian, A., Pirsiavash, H.: Joint semantic segmentation and depth estimation with deep convolutional networks. In: 3DV (2016) 2, 3
37. Qi, X., Liao, R., Liu, Z., Urtasun, R., Jia, J.: GeoNet: Geometric neural network for joint depth and surface normal estimation. In: CVPR (2018) 1, 3, 5
38. Rajagopalan, A., Chaudhuri, S., Mudenagudi, U.: Depth estimation and image restoration using defocused stereo pairs. IEEE Trans. Pattern Anal. Mach. Intell. **26**(11), 1521–1525 (Nov 2004) 1
39. Saxena, A., Sun, M., Ng, A.Y.: Make3D: Learning 3-D scene structure from a single still image. IEEE Trans. Pattern Anal. Mach. Intell. **31**(5), 824–840 (Oct 2009) 1, 2, 10, 11
40. Sener, O., Koltun, V.: Multi-task learning as multi-objective optimization. In: NIPS (2018) 3
41. Silberman, N., Hoiem, D., Kohli, P., Fergus, R.: Indoor segmentation and support inference from RGBD images. In: ECCV (2012) 1, 2, 10, 11, 12, 13, 14
42. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: ICLR (2015) 1, 14
43. Song, S., Lichtenberg, S.P., Xiao, J.: SUN RGB-D: A RGB-D scene understanding benchmark suite. In: CVPR (2015) 1, 12

44. Wedel, A., Franke, U., Klappstein, J., Brox, T., Cremers, D.: Realtime depth estimation and obstacle detection from monocular video. In: Joint Pattern Recognition Symposium (2006) 1
45. Xian, K., Shen, C., Cao, Z., Lu, H., Xiao, Y.: Monocular relative depth perception with web stereo data supervision. In: CVPR (2018) 1, 10
46. Xu, D., Ricci, E., Ouyang, W., Wang, X., Sebe, N.: Multi-scale continuous CRFs as sequential deep networks for monocular depth estimation. In: CVPR (2017) 11
47. Yang, J., Price, B., Cohen, S.: Object contour detection with a fully convolutional encoder-decoder network. In: CVPR (2016) 3, 9
48. Yin, Z., Shi, J.: GeoNet: Unsupervised learning of dense depth, optical flow and camera pose. In: CVPR (2018) 1
49. Zhang, Y., Funkhouser, T.: Deep depth completion of a single RGB-D image. In: CVPR (2018) 2
50. Zhang, Z., Cui, Z., Xu, C.: Pattern-affinitive propagation across depth, surface normal and semantic segmentation. In: CVPR (2019) 3, 11
51. Zhang, Z., Cui, Z., Xu, C., Jie, Z., Li, X., Yand, J.: Joint task-recursive learning for semantic segmentation and depth estimation. In: ECCV (2018) 11
52. Zoran, D., Isola, P., Krishnan, D., Freeman, W.T.: Learning ordinal relationships for mid-level vision. In: ICCV (2015) 10