- Supplementary Material -
Lightweight Monocular Depth with a Novel Neural Architecture Search Method

Lam Huynh[1] Phong Nguyen[1] Jiri Matas[2]
Esa Rahtu[3] Janne Heikkilä[1]
[1]University of Oulu [2]Czech Technical University in Prague [3]Tampere University

This material provides a comparison between the original tabu search and assisted tabu search in section 1. Section 2 describes a solution to search for good multi-objective balance coefficients α (Eq. 1 and 4 in the main paper), while more information on the generated network architectures is presented in Section 3.

1. Experiment with the original tabu search (TS) and assisted tabu search (ATS)

Tabu search [3] is a robust metaheuristic procedure for solving combinatorial optimization problems consisting of two distinguishing features. First, it prohibits reversal moves to discourage the search from coming back to previously-visited solutions. Second, it accepts degenerating moves if improving movement is unavailable to avoid being trap in local minima. Figure 1 presents a simplified flowchart of the tabu search. By iteratively making *good moves*, it is more likely to reach optimized solutions.

However, directly employing tabu search for architecture exploration is expensive as the evaluation process (Figure 1, orange box) typically requires training and validation on the whole dataset. For comparison, we utilize the original tabu search (TS) and assisted tabu search (ATS) to perform architecture search for 20 iterations from one parent network. The result in Table 1 demonstrates the efficacy of ATS as the search using ATS runs ~ 7 times faster while producing comparable results to the TS instance.

2. Searching for a good balance coefficient (α)

Determining a good balance coefficient value (α) for Eq.1 and 4 in the main paper is crucial as it greatly affects the search performance. To this end, we perform grid search on randomly subsampled sets from the training data seeking for the optimized α value. The pattern in Figure 2 shows that, for NYU-Depth-v2 [4], KITTI [2], and ScanNet [1], approximately good α values range from 0.55 to 0.6. Additionally, the grid search is much faster (took ~ 15 hours on one dataset), enabling finding good α values when deploying to different datasets.

Table 1. Searching for 20 iterations from one parent network using tabu search (TS) and assisted tabu search (ATS) on the NYU-Depth-v2 dataset.

Method	Search Time	REL↓	RMSE↓	δ_1↑
w/ TS	24.4 GPU hours	0.161	0.554	0.753
w/ ATS	3.4 GPU hours	0.162	0.557	0.751

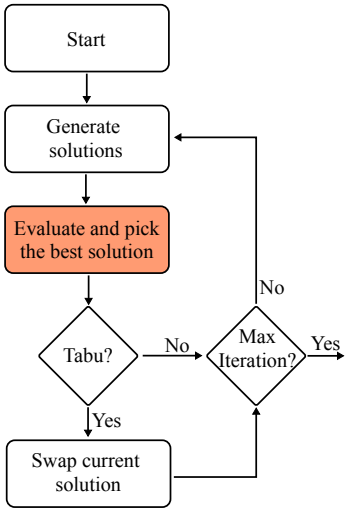


Figure 1. A simple flowchart of tabu search.

3. Generated Architectures

Table 2, 3, 4 illustrate the generated LiDNAS-N, LiDNAS-K, LiDNAS-S architectures for NYU-Depth-v2, KITTI and ScanNet, respectively. As expected, the networks consist of various layers with some typical operations as shown in Figure 3.

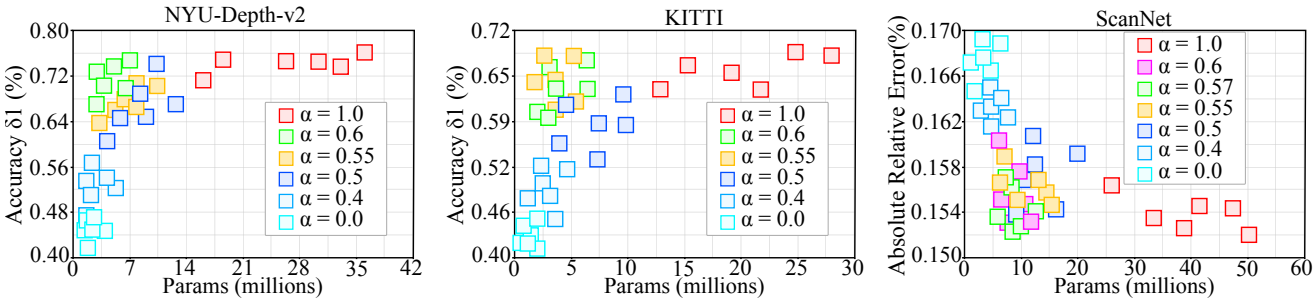
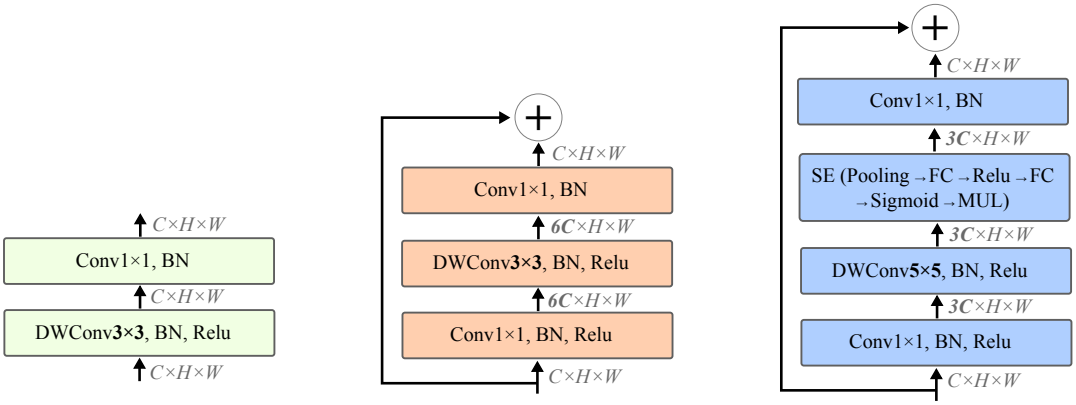


Figure 2. Grid search using randomly subsampled sets from the training data to look for good balance coefficient values on NYU-Depth-v2 (left), KITTI (center), and ScanNet (right).

Table 2. Detailed structure of an LiDNAS-N instance. MBConv denotes mobile inverted residual block, SepConv indicates depthwise separable convolution, and SE is squeeze-and-excitation module (see Figure 3 for more details). n is the number of layers within the block, t is the expansion factor of the MBConv module, and k is the kernel size. **#in** and **#out** are the number of input and output channels.

Input	**Operations**	**n**	**t**	**k**	**#in**	**#out**	**Resolutions**	**Output**
image	Conv2d (k3x3)	2	-	3	3	16	304x228	*en_blk_11*
en_blk_11	MBConv (t6,k3x3)	3	6	3	16	32	304x228	*en_blk_12*
en_blk_12(//)*up_blk_27*	MBConv, SE (t3,k5x5)	3	3	5	64	56	304x228	*de_blk_13*
de_blk_13	MBConv (t7,k3x3)	2	7	3	56	64	304x228	*de_blk_14*
de_blk_14	MBConv (t6,k5x5)	3	6	5	64	1	304x228	*depth_1*
down(*en_blk_12*)	SepConv (k3x3)	2	-	3	32	16	152x114	*down_blk_26*
up(*de_blk_24*)	MBConv (t6,k3x3)	1	6	3	72	32	304x228	*up_blk_27*
down_blk_26	MBConv (t5,k3x3)	2	5	3	16	32	152x114	*en_blk_21*
en_blk_21	MBConv, SE (t3,k5x5)	1	3	5	32	48	152x114	*en_blk_22*
en_blk_12, *up_blk_37*	MBConv (t7,k3x3)	1	7	3	62	64	152x114	*de_blk_23*
de_blk_23	MBConv, SE (t3,k5x5)	2	3	5	64	72	152x114	*de_blk_24*
de_blk_24	MBConv (t6,k3x3)	2	6	3	72	1	152x114	*depth_2*
down(*en_blk_22*)	MBConv (t6,k5x5)	1	6	5	48	32	76x52	*down_blk_36*
up(*de_blk_34*)	MBConv (t6,k3x3)	1	6	3	96	14	152x114	*up_blk_37*
down_blk_36	MBConv (t7,k3x3)	1	7	3	32	48	76x52	*en_blk_31*
en_blk_31	SepConv (k3x3)	2	-	3	48	56	76x52	*en_blk_32*
en_blk_32, *up_blk_47*	MBConv (t5,k3x3)	1	5	3	74	72	76x52	*de_blk_33*
de_blk_33	MBConv (t7,k3x3)	1	7	3	72	96	76x52	*de_blk_34*
de_blk_34	MBConv (t3,k5x5)	2	3	5	96	1	76x52	*depth_3*
down(*en_blk_32*)	MBConv (t6,k3x3)	1	6	3	56	48	37x26	*down_blk_46*
up(*de_blk_44*)	MBConv (t6,k5x5)	1	6	5	112	18	76x52	*up_blk_47*
down_blk_46	SepConv (k3x3)	3	-	3	48	56	37x26	*en_blk_41*
en_blk_41	MBConv, SE (t3,k5x5)	1	3	5	56	64	37x26	*en_blk_42*
en_blk_42, *up_blk_57*	MBConv, SE (t3,k5x5)	2	3	5	80	96	37x26	*de_blk_43*
de_blk_43	MBConv (t6,k3x3)	1	6	3	96	112	37x26	*de_blk_44*
de_blk_44	MBConv (t6,k3x3)	2	6	3	112	1	37x26	*depth_4*
down(*en_blk_42*)	MBConv (t6,k5x5)	1	6	5	64	56	19x15	*down_blk_56*
up(*de_blk_54*)	SepConv (k3x3)	1	-	3	112	16	37x26	*up_blk_57*
down_blk_56	MBConv, SE (t3,k5x5)	1	3	5	56	64	19x15	*en_blk_51*
en_blk_51	SepConv (k3x3)	2	-	3	64	72	19x15	*en_blk_52*
en_blk_52	MBConv (t7,k3x3)	1	7	3	72	96	19x15	*de_blk_53*
de_blk_53	MBConv (t3,k5x5)	1	3	5	96	112	19x15	*de_blk_54*
de_blk_54	MBConv (t6,k3x3)	1	6	3	112	1	19x15	*depth_5*



(a) SepConv (k3x3) (b) MBConv (t6, k3x3) (c) MBConv, SE (t3, k5x5)

Figure 3. Detailed component of some typical operations in Table 2, 3, 4. (a) depthwise separable convolution with kernel size 3×3 , (b) mobile invertable residual convolution block with expansion factor $t = 6$ and kernel size 3×3, and (c) mobile invertable residual convolution block integrated squeeze-and-excitation module with expansion factor $t = 3$ and kernel size 5×3. DWConv indicates depthwise convolution, BN denotes batch normalization and MUL is multiplication.

Table 3. Detailed structure of an LiDNAS-K instance. MBConv denotes mobile inverted residual block, SepConv indicates depthwise separable convolution, and SE is squeeze-and-excitation module (see Figure 3 for more details). n is the number of layers within the block, t is the expansion factor of the MBConv module, and k is the kernel size. **#in** and **#out** are the number of input and output channels.

Input	**Operations**	**n**	**t**	**k**	**#in**	**#out**	**Resolutions**	**Output**
$image$	Conv2d (k3x3)	2	-	3	3	18	640x192	en_blk_11
en_blk_11	MBConv (t5,k3x3)	3	5	3	18	28	640x192	en_blk_12
en_blk_12, up_blk_27	SepConv (k3x3)	3	-	3	52	64	640x192	de_blk_13
de_blk_13	MBConv (t3,k5x5)	2	3	5	64	72	640x192	de_blk_14
de_blk_14	MBConv, SE (t7,k5x5)	2	7	5	72	1	640x192	$depth\_1$
$down(en\_blk\_12)$	MBConv (t3,k5x5)	1	3	5	28	18	320x96	$down\_blk\_26$
$up(de\_blk\_24)$	MBConv (t6,k3x3)	1	6	3	72	24	640x192	up_blk_27
$down\_blk\_26$	MBConv (t5,k3x3)	2	5	3	18	24	320x96	en_blk_21
en_blk_21	SepConv (k5x5)	2	-	5	24	32	320x96	en_blk_22
en_blk_12, up_blk_37	MBConv (t7,k3x3)	1	7	3	48	56	320x96	de_blk_23
de_blk_23	MBConv, SE (t3,k5x5)	2	3	5	56	72	320x96	de_blk_24
de_blk_24	MBConv (t6,k3x3)	2	6	3	72	1	320x96	$depth\_2$
$down(en\_blk\_22)$	MBConv (t6,k5x5)	1	6	5	32	32	160x48	$down\_blk\_36$
$up(de\_blk\_34)$	SepConv (k3x3)	2	-	3	92	16	320x96	up_blk_37
$down\_blk\_36$	MBConv (t7,k3x3)	1	7	3	32	44	160x48	en_blk_31
en_blk_31	SepConv (k3x3)	2	-	3	44	56	160x48	en_blk_32
en_blk_32, up_blk_47	MBConv (t5,k3x3)	1	5	3	68	72	160x48	de_blk_33
de_blk_33	MBConv (t6,k3x3)	1	6	3	72	92	160x48	de_blk_34
de_blk_34	MBConv, SE (t3,k5x5)	2	3	5	92	1	160x48	$depth\_3$
$down(en\_blk\_32)$	SepConv (k3x3)	3	-	3	56	48	80x24	$down\_blk\_46$
$up(de\_blk\_44)$	MBConv (t6,k3x3)	1	6	3	108	12	160x48	up_blk_47
$down\_blk\_46$	SepConv (k3x3)	3	-	3	48	56	80x24	en_blk_41
en_blk_41	MBConv, SE (t3,k5x5)	1	3	5	56	64	80x24	en_blk_42
en_blk_42, up_blk_57	MBConv, SE (t3,k5x5)	2	3	5	78	88	80x24	de_blk_43
de_blk_43	SepConv (k5x5)	2	-	5	88	108	80x24	de_blk_44
de_blk_44	MBConv (t7,k5x5)	1	7	5	108	1	80x24	$depth\_4$
$down(en\_blk\_42)$	MBConv (t6,k5x5)	1	6	5	64	56	40x12	$down\_blk\_56$
$up(de\_blk\_54)$	SepConv (k3x3)	1	-	3	110	14	80x24	up_blk_57
$down\_blk\_56$	MBConv, SE (t3,k5x5)	1	3	5	56	64	40x12	en_blk_51
en_blk_51	SepConv (k3x3)	2	-	3	64	72	40x12	en_blk_52
en_blk_52	MBConv (t3,k5x5)	1	3	5	72	96	40x12	de_blk_53
de_blk_53	SepConv (k5x5)	1	-	5	96	110	40x12	de_blk_54
de_blk_54	MBConv (t3,k5x5)	2	3	5	110	1	40x12	$depth\_5$

Table 4. Detailed structure of an LiDNAS-S instance. MBConv denotes mobile inverted residual block, SepConv indicates depthwise separable convolution, and SE is squeeze-and-excitation module (see Figure 3 for more details). n is the number of layers within the block, t is the expansion factor of the MBConv module, and k is the kernel size. **#in** and **#out** are the number of input and output channels.

Input	**Operations**	**n**	**t**	**k**	**#in**	**#out**	**Resolutions**	**Output**
$image$	Conv2d (k5x5)	3	-	5	3	32	304x228	en_blk_11
en_blk_11	MBConv (t5,k3x3)	3	5	3	32	48	304x228	en_blk_12
en_blk_12, up_blk_27	MBConv (t5,k3x3)	3	5	3	80	96	304x228	de_blk_13
de_blk_13	MBConv (t3,k5x5)	3	3	5	96	110	304x228	de_blk_14
de_blk_14	MBConv, SE (t7,k5x5)	3	7	5	110	1	304x228	$depth\_1$
$down(en\_blk\_12)$	MBConv (t3,k5x5)	1	3	5	48	24	152x114	$down\_blk\_26$
$up(de\_blk\_24)$	MBConv (t6,k3x3)	1	6	3	92	32	304x228	up_blk_27
$down\_blk\_26$	MBConv (t5,k3x3)	2	5	3	24	32	152x114	en_blk_21
en_blk_21	MBConv, SE (t5,k3x3)	2	5	3	32	48	152x114	en_blk_22
en_blk_12, up_blk_37	MBConv (t5,k5x5)	1	5	5	72	72	152x114	de_blk_23
de_blk_23	MBConv, SE (t5,k3x3)	2	5	3	72	92	152x114	de_blk_24
de_blk_24	MBConv, SE (t7,k5x5)	2	7	5	92	1	152x114	$depth\_2$
$down(en\_blk\_22)$	MBConv (t6,k5x5)	2	6	5	48	24	76x52	$down\_blk\_36$
$up(de\_blk\_34)$	MBConv (t5,k3x3)	1	5	3	108	24	152x114	up_blk_37
$down\_blk\_36$	SepConv (k3x3)	3	-	3	24	36	76x52	en_blk_31
en_blk_31	MBConv, SE (t7,k3x3)	2	7	3	36	48	76x52	en_blk_32
en_blk_32, up_blk_47	MBConv (t3,k5x5)	2	3	5	82	96	76x52	de_blk_33
de_blk_33	MBConv (t6,k3x3)	1	6	3	96	108	76x52	de_blk_34
de_blk_34	MBConv, SE (t6,k5x5)	2	6	5	108	1	76x52	$depth\_3$
$down(en\_blk\_32)$	MBConv (t3,k5x5)	2	3	5	48	32	37x26	$down\_blk\_46$
$up(de\_blk\_44)$	MBConv (t6,k3x3)	2	6	3	114	34	76x52	up_blk_47
$down\_blk\_46$	SepConv (k3x3)	3	-	3	32	52	37x26	en_blk_41
en_blk_41	MBConv, SE (t3,k5x5)	1	3	5	52	60	37x26	en_blk_42
en_blk_42, up_blk_57	MBConv, SE (t3,k5x5)	2	3	5	92	92	37x26	de_blk_43
de_blk_43	MBConv, SE (t3,k5x5)	2	3	5	92	114	37x26	de_blk_44
de_blk_44	MBConv, SE (t5,k3x3)	2	5	3	114	1	37x26	$depth\_4$

References

[1] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, pages 5828–5839, 2017.

[2] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *IJRR*, 32(11):1231–1237, 2013.

[3] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, 13(5):533–549, 1986.

[4] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, pages 746–760. Springer, 2012.