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

Abstract

This paper presents a novel neural architecture search method, called LiDNAS, for generating lightweight monocular depth estimation models. Unlike previous neural architecture search (NAS) approaches, where finding optimized networks is computationally demanding, the introduced novel Assisted Tabu Search leads to efficient architecture exploration. Moreover, we construct the search space on a pre-defined backbone network to balance layer diversity and search space size. The LiDNAS method outperforms the state-of-the-art NAS approach, proposed for disparity and depth estimation, in terms of search efficiency and output model performance. The LiDNAS optimized models achieve result superior to compact depth estimation state-of-the-art on NYU-Depth-v2, KITTI, and ScanNet, while being 7%-500% more compact in size, i.e the number of model parameters.

1. Introduction

Depth information is essential to numerous computer vision applications, including robotics, mixed reality, and scene understanding. Traditionally, accurate depth measurements are acquired using stereo or multi-view setups [23, 52] or active sensors such as ToF cameras, LIDARs. However, deploying such methods for resource-limited devices is costly or may even be infeasible in practice. Considering this, current advances in learning-based monocular depth estimation proffering them as viable alternatives to conventional approaches.

Recent deep neural networks (DNN) show compelling results on single image depth estimation by formulating apparent depth cues [3, 5, 13, 27, 28, 33, 35, 36, 44, 57] or estimating relative depth in unconstrained settings [4, 34, 45]. Moreover, self-supervised methods [15, 19] offered appealing solutions for single image depth estimation. Nevertheless, most studies focus on increasing accuracy at the expense of model complexity, making them infeasible to devices with limited hardware capabilities.

To tackle this problem, lightweight depth estimation methods [55] were proposed by utilizing small and straightforward architecture. Usually such simple designs are unreliable and yield low-quality predictions. Other popular strategies include quantizing the weights of a network into low-precision fixed-point operations [22] or pruning by directly cutting off less important filters [58]. That being said, these methods depend on a baseline model, tend to degrade its performance afterward and incapable of exploring new combinations of DNN operations. Moreover, creating a

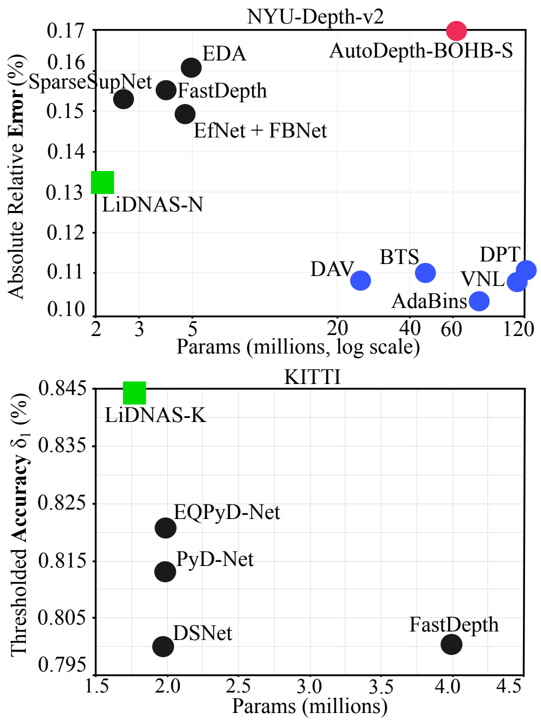

Figure 1. Absolute relative error and thresholded accuracy (δ_1) vs. the number of parameters for recent depth estimation methods on NYU-Depth-v2 (top) and KITTI (bottom) – the LiDNAS models outperforms the lightweight baselines (black), while using substantially less parameters than the current state-of-the-art methods (blue). Compared to the recent NAS-based depth estimation method (red), LiDNAS improves in both performance and compactness.

resource-constrained model is a non-trivial task requiring 1) expert knowledge to carefully balance accuracy and resource and 2) plenty of tedious trial-and-error work.

Neural architecture search (NAS), proposed recently [61, 62], exhibits compelling results, and more importantly, promises to relieve from the manual tweaking of deep neural architectures. Unfortunately, NAS methods mostly obligate thousands of training hours on hundreds of GPUs. To address this, recent NAS studies introduced various efficiency increasing techniques, which include weight sharing [41], and network transformation [11]. These methods show promising results, but they are still expensive and mainly focus on classification and detection.

This paper introduces LiDNAS, an efficient model compactness-aware NAS framework, with the objective of searching for accurate and lightweight monocular depth estimation architectures. The approach is based on two main ideas. First, we observe that previous NAS methods essentially search for a few types of cells and then repeatedly accumulate the same cells to build the whole network. Although doing this simplifies the search process, it also restrains layer diversity that is important for computational efficiency. Instead, we construct a pre-defined backbone network that utilizes different layers striving for the right balance between flexibility and search space size. Secondly, we proposed the Assisted Tabu Search (ATS) for efficient neural architecture search. Inspired by the recent NAS study that suggests estimating network performance without training [40], we integrate this idea into our multi-objective search function to swiftly evaluate our candidate networks. This, in turn, reduces $\sim 90\%$ search time compared to state-of-the-art NAS-based disparity and depth estimation approaches [48].

Figure 1 summarizes a comparison between our LiDNAS models and other state-of-the-art lightweight approaches. Compared to PyD-Net [42], our method improves the REL, RMSE, and thresholded accuracy by 13.6%, 8.3%, and 3% with similar execution time on the Google Pixel 3a phone (see Table 4). Compared to FastDepth [55] and EDA [54], our model achieves higher accuracy with fewer parameters. To summarize, our work makes the following contributions:

- We propose a multi-objective exploration framework, LiDNAS, searching for accurate and lightweight monocular depth estimation architectures.
- We introduce a novel scheme called Assisted Tabu Search, enabling fast neural architecture search.
- We create a well-defined search space that allows computational flexibility and layer diversity.
- We achieve the state-of-the-art results compared to the lightweight baselines on NYU-Depth-v2, KITTI, and ScanNet while using less parameters.

The implementation of LiDNAS will be made publicly available upon publication of the paper.

2. Related work

Monocular depth estimation Learning-based single image depth estimation was first introduced by Saxena et al. [50]. Later studies improved accuracy by using large network architectures [5, 9, 10, 27, 32] or integrating semantic information [30] and surface normals [43]. Fu et al. [14] formulated depth estimation as an ordinal regression problem, while [4, 34] estimated relative instead of metric depth. Facil et al. [13] proposed to learn camera calibration from the images for depth estimation. Recent approaches further improve the performance by exploiting monocular priors such as planarity constraints [35, 36, 59, 28, 33] or occlusion [44]. Gonzalez and Kim [20] estimated depth by synthesizing stereo pairs from a single image, while [57] and [45] applied vision-transformer for depth prediction. However, these studies mostly focus on increasing accuracy at the cost of model complexity that is infeasible in resource-limited settings.

Lightweight depth estimation architectures For resource-limited hardware, it is more desirable to not only have a fast but also accurate model. One simple alternative is employing lightweight architectures such as MobileNet [24, 25, 49, 55], GhostNet [21], and FBNet [54]. One popular approach is utilizing network compression techniques, including quantization [22], network pruning [58], and knowledge distillation [60]. Other methods employ well-known pyramid networks or dynamic optimization schemes [1]. However, these tasks are tedious, require a lot of trial-and-error, and usually lead to architectures with low accuracy.

Neural Architecture Search There has been increasing interest in automating network design using neural architecture search. Most of these methods focus on searching high-performance architecture using reinforcement learning [2, 37, 41, 61, 62], evolutionary search [46], differentiable search [38], or other learning algorithms [39]. However, these methods are usually very slow and require huge resources for training. Other studies [8, 12, 26] also attempt to optimize multiple objectives like model size and accuracy. Nevertheless, their search process optimizes only on small tasks like CIFAR. In contrast, our proposed method targets real-world data such as NYU, KITTI and ScanNet.

3. LiDNAS

We propose the LiDNAS framework to search for accurate and lightweight monocular depth estimation architectures. The overview the our approach is presented in Figure 3. It takes in a dataset as input to search for the best possible model. This model can be deployed for depth estimation on hardware-limited devices. The first subsection defines the

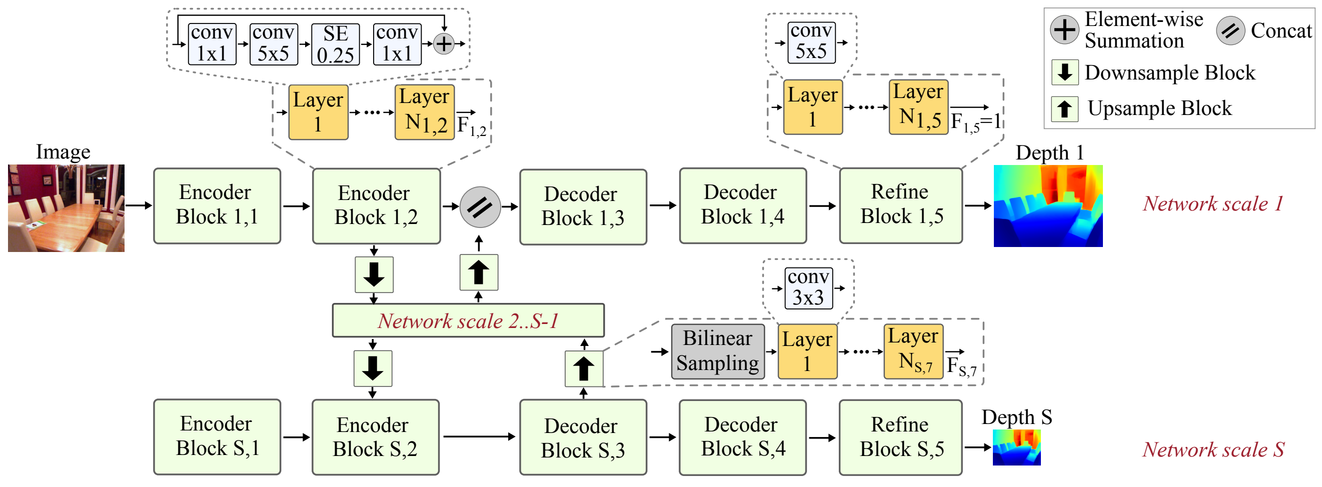

Figure 2. The search space of our LiDNAS framework. Models are constructed from a pre-defined backbone network containing encoder, decoder, refine, downsample and upsample *blocks* (green). A block is formed by several identical layers (orange) that are generated from a pool of operations and connections. Layers within a block are the same while layers of different blocks can be different.

search space while the remaining two describe our multi-objective exploration and search algorithm.

3.1. Search Space

Previous neural architecture search (NAS) studies demonstrated the significance of designing a well-defined search space. A common choice of NAS is searching for a small set of complicated cells from a smaller dataset [62, 37, 46]. These cells are later replicated to construct the entire architecture that hindered layer diversity and suffered from domain differences [53]. On the other hand, unlike classification tasks, dense prediction problems involve mapping a feature representation in the encoder to predictions at larger spatial resolution in the decoder.

To this end, we build our search space upon a pre-defined backbone that is shown as the set of green blocks in Figure 2. The backbone is divided into multi-scale pyramid networks operating at different spatial resolutions. Each network scale consists of two encoder blocks, two decoder blocks, a refine block, a downsample and a upsample block (except for scale 1). Each block is constructed from a set of identical layers (marked as orange in Figure 2). Inspired by [53], we search for the layer from a pool of operations and connections, including:

- The number of resolution scales S.

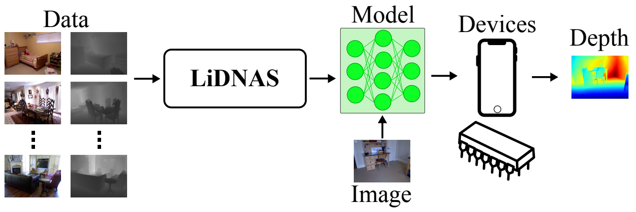

Figure 3. Overview of the proposed approach.

- The number of layer for each block $N_{i,j}$.
- Convolutional operations (ConvOps): vanilla 2D convolution, depthwise convolution, and inverted bottleneck convolution.
- Convolutional kernel size (KSize): 3×3, 5×5.
- Squeeze and excitation ratio (SER): 0, 0.25.
- Skip connections (SOps): residual or no connection.
- The number of output channels: $F_{i,j}$.

where i indicates the resolution scale and j is the block index at the same resolution. Internal operations such as ConvOps, KSize, SER, SOps, $F_{i,j}$ are utilized to construct the layer while $N_{i,j}$ determines the number of layer that will be replicated for block$_{i,j}$. In other words, as shown in Figure 2, layers within a block (e.g. layers 1 to N$_{1,2}$ of Encoder Block 1,2 are the same) are similar while layers of different blocks (e.g. Layer 1 in Refine Block 1,5 versus Layer 1 in Upsample Block S,7) can be different.

We also perform layer mutation to further diversifying the network structure during the architecture search process. The mutation operations include:

- Swapping operations of two random layers with compatibility check.
- Modifying a layer with a new valid layer from the predefined operations.

Moreover, we also set computational constraints to balance the kernel size with the number of output channels. Therefore, increasing the kernel size of one layer usually results in decreasing output channels of another layer.

Assuming we have a network of S scales, and each block has a sub-search space of size M then our total search space will be $M^{5+[(S-1)*7]}$. Supposedly, a standard case with $M = 192$, $S = 5$ will result in a search space of size $\sim 2 \times 10^{75}$.

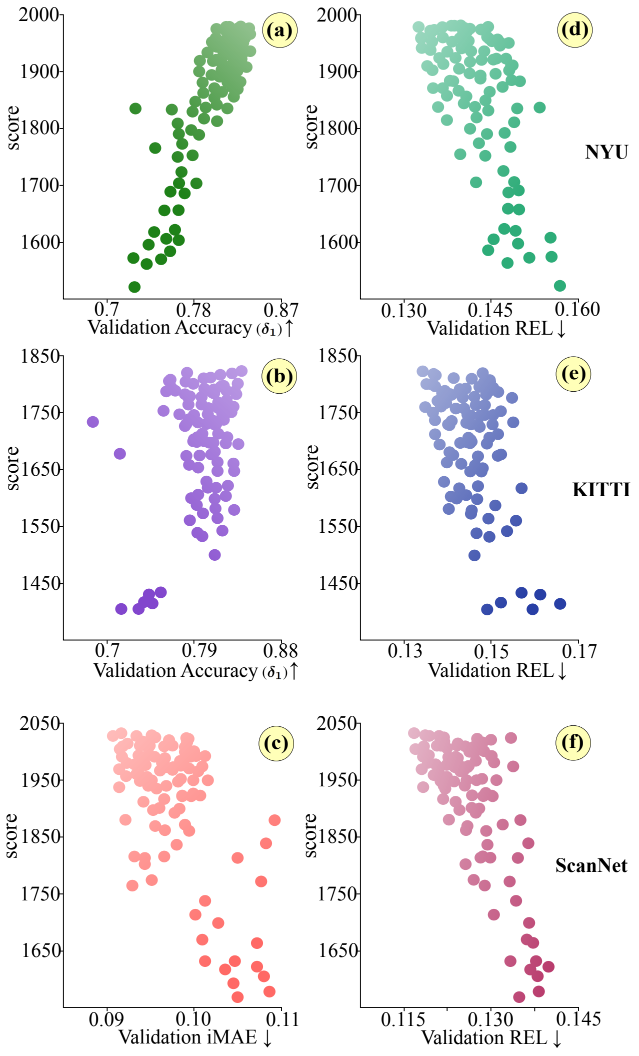

Figure 4. Plots of the *score* at initialisation of untrained architectures against evaluation metrics after training: (a), (b) accuracy (δ_1); (e) mean absolute error of the inverse depth (iMAE); and (d), (e), (f) absolute relative error (REL). Plots from the first, second and third row are obtained from NYU-Depth-v2, KITTI and ScanNet dataset, respectively.

3.2. Multi-Objective Exploration

We introduce a multi-objective search paradigm seeking for both accurate and compact architectures. For this purpose, we monitor the *validation grade* $\mathcal{G}$ that formulates both accuracy and the number of parameter of a trained model. It is defined by

$$\mathcal{G}(m) = \alpha \times A(m) + (1-\alpha) \times \left[\frac{P}{P(m)}\right]^r \quad (1)$$

where $A(m)$ and $P(m)$ are validation accuracy and the number of parameters of model m. P is the target compactness, α is the balance coefficient, and r is an exponent with $r = 0$ when $P(m) \leq P$ and otherwise $r = 1$. The goal is to search for an architecture m^* where $G(m^*)$ is maximum.

However, computing G requires training for every architecture candidate, resulting in considerable search time. To mitigate this problem, Mellor et al. [40] suggested to score an architecture at initialisation to predict its performance before training. For a network f, the *score(f)* is defined as:

$$score(f) = log|K_H| \quad (2)$$

where K_H is the kernel matrix. Assume the mapping of model f from a batch of data $X = \{x_i\}_{i=1}^N$ is $f(x_i)$. By assigning binary indicators to every activation units in f, a linear region x_i of data point i is represented by the binary code c_i. The kernel matrix K_H is defined as:

$$K_H = \begin{pmatrix} N_A - d_H(c_1, c_1) & \dots & N_A - d_H(c_1, c_N) \\ \vdots & \ddots & \vdots \\ N_A - d_H(c_N, c_1) & \dots & N_A - d_H(c_N, c_N) \end{pmatrix} \quad (3)$$

where N_A is the number of activation units, and $d_H(c_i, c_j)$ is the Hamming distance between two binary codes. Inspired by this principle, we generate and train a set of different architectures on NYU, KITTI, and ScanNet. We evaluate the performance of these models and visualize the results against the *score* that in our case is the mapping of depth values within image batches. Plots in Figure 4 show that models with higher *score* tend to yield better results. Leveraging this observation, we 1) utilize the *score* in our initial network ranking, and 2) define the mutation exploration reward $\mathcal{R}$ as:

$$\mathcal{R}(m_i, m_j) = \alpha \times \frac{score(m_j)}{score(m_i)} + (1-\alpha) \times \left[\frac{P}{P(m_j)}\right]^r \quad (4)$$

where m_j is a child network that is mutated from m_i architecture.

3.3. Search Algorithm

The flowchart of our architecture search is presented in Figure 5. We first randomly generate $60K$ unique parent models and create the initial network ranking based on the *score* in Eq. 2. We then select *six* architectures in which *three* are the highest-ranked while the other *three* have the highest score of the networks with the size closest to the target compactness.

Starting from these initial networks, we strive for the best possible model utilizing the Assisted Tabu Search (ATS). Tabu search (TS) [18] is a high level procedure for solving multicriteria optimization problems. It is an iterative algorithm that starts from some initial feasible solutions and

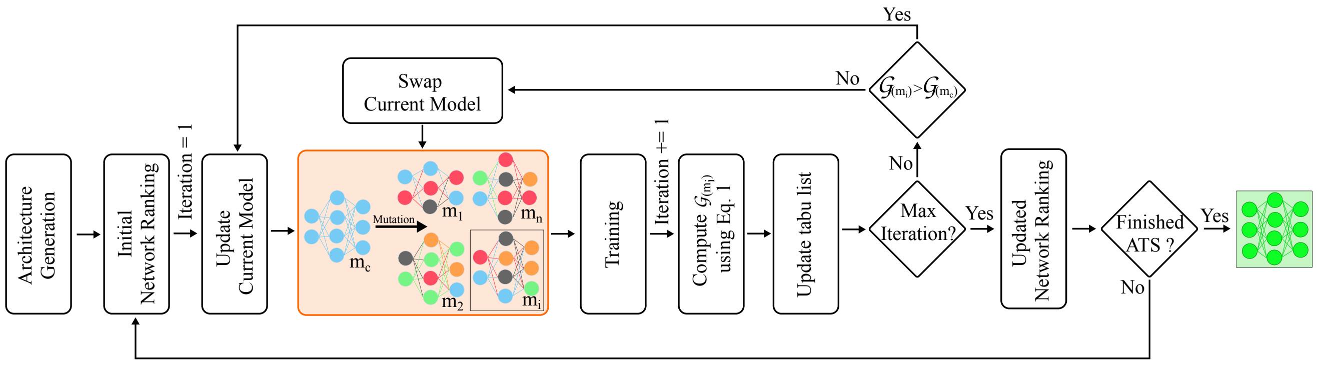

Figure 5. The flowchart of our architecture search that utilizes the Assisted Tabu Search (ATS) with mutation to search for accurate and lightweight monocular depth estimation networks.

aims to determine better solutions while being designed to avoid traps at local minima.

We propose ATS by applying Eq. 1 and 4 to TS to speed up the searching process. Specifically, we mutate numerous children models (m_1, m_2, .., m_n) from the current architecture (m_c). The mutation exploration reward $\mathcal{R}(m_c, m_i)$ is calculated using Eq. 4. ATS then chooses to train the mutation with the highest rewards (e.g. architecture m_i as demonstrated in Figure 5). The validation grade of this model $\mathcal{G}(m_i)$ is calculated after the training. The performance of the chosen model is assessed by comparing $\mathcal{G}(m_i)$ with $\mathcal{G}(m_c)$. If $\mathcal{G}(m_i)$ is larger than $\mathcal{G}(m_c)$, then m_i is a good mutation, and we opt to build the next generation upon its structure. Otherwise, we swap to use the best option in the tabu list for the next mutation. The process stops when reaching a maximum number of iterations or achieving a terminal condition. The network ranking will be updated, and the search will continue for the remaining parent architectures. Further comparison between TS and ATS is provided in the supplementary material.

4. Performance analysis

In this section, we evaluate the performance of the proposed method and compare it with several baselines on the NYU-Depth-v2, KITTI, and ScanNet datasets.

4.1. Datasets

We evaluate the proposed method using NYU-Depth-v2 [51], ScanNet [7] and KITTI [16] datasets. NYU-Depth-v2 contains $\sim$ $120K$ RGB-D images obtained from 464 indoor scenes. From the entire dataset, we use 50K images for training and the official test set of 654 images for evaluation. The ScanNet dataset comprises of 2.5 million RGB-D images acquired from 1517 scenes. For this dataset, we use the training subset of $\sim$ $20K$ images provided by the Robust Vision Challenge 2018 [17] (ROB). In this paper, we report the results on the ScanNet official test set of 5310 images instead. KITTI is an outdoor driving dataset, where we use the standard Eigen split [9, 10] for training (39K images) and testing (697 images).

4.2. Evaluation metrics

The performance is assessed using the standard metrics provided for each dataset. That is, for NYU-Depth-v2 and KITTI we calculate the mean absolute relative error (REL), root mean square error (RMSE), and thresholded accuracy (δ_i). For the ScanNet dataset, we provide the mean absolute relative error (REL), mean square relative error (sqREL), scale-invariant mean squared error (SI), mean absolute error (iMAE), and root mean square error (iRMSE) of the inverse depth values.

4.3. Implementation Details

For searching, we directly perform our architecture exploration on the training samples of the target dataset. We set the target compactness parameter P using the previously published compact models as a guideline. We set the maximum number of exploration iteration to 100 and stop the exploration procedure if a better solution cannot be found after 10 iterations. The total search time required to find optimal architecture is $\sim$ 4.3 GPU days.

For training, we use the Adam optimizer [31] with $(\beta_1, \beta_2, \epsilon) = (0.9, 0.999, 10^{-8})$. The initial learning rate is $7 * 10^{-4}$, but from epoch 10 the learning is reduced by 5% per 5 epochs. We use batch size 256 and augment the input RGB and ground truth depth images using random rotation ([-5.0, +5.0] degrees), horizontal flip, rectangular window droppings, and colorization (RGB only).

4.4. Comparison with state-of-the-art

NYU-Depth-v2: We set the target compactness $P = 2M$ with the balance coefficient $\alpha = 0.6$ to search for the optimized model on NYU-Depth-v2. We then select the best performance model (LiDNAS-N) and compare its results with lightweight state-of-the-art methods [54, 55, 56, 60]

Table 1. Evaluation on the NYU-Depth-v2 dataset. Metrics with ↓ mean lower is better and ↑ mean higher is better. Type column shows the exploration method used to obtain the model. RL, ATS, and manual, refer to reinforcement learning, assisted tabu search, and manual design, respectively.

Architecture		**#params**	**Type**	**Search Time**	**REL↓**	**RMSE↓**	**δ_1↑**	**δ_2↑**	**δ_3↑**
AutoDepth-BOHB-S	Saikia et al.'19 [48]	63.0M	RL	42 GPU days	0.170	0.599	-	-	-
EDA	Tu et al.'21 [54]	5.0M	Manual	-	0.161	0.557	0.782	0.945	0.984
FastDepth	Wofk et al.'19 [55]	3.9M	Manual	-	0.155	0.599	0.778	0.944	0.981
SparseSupNet	Yucel et al.'21 [60]	2.6M	Manual	-	0.153	0.561	0.792	0.949	0.985
Ef+FBNet	Tu & Wu et al. [54, 56]	4.7M	Manual	-	0.149	0.531	0.803	0.952	0.987
LiDNAS-N	Ours	**2.1M**	ATS	4.3 GPU days	**0.132**	**0.487**	**0.845**	**0.965**	**0.993**

along with their numbers of parameters. As shown in Table 1, LiDNAS-N outperforms the baseline while containing the least amount of parameters. Comparing with the best-performing approach, the proposed model improves the REL, RMSE, and θ_1 by 11.4%, 8.2%, and 6.8% while compressing the model size by 55%. Our method produces high-quality depth maps with sharper details as presented in Figure 9. However, we observe that all methods still struggle in challenging cases, such as the scene containing Lambertian surfaces as illustrated by the example in the third column of Figure 9. Moreover, the proposed method improves REL and RMSE by 22.3% and 18.7% while using only 3% of the model parameters comparing to the state-of-the-art NAS-based disparity and depth estimation approaches [48]. In addition, our method requires 90% less search time than [48].

KITTI: In the case of KITTI, we aim at the target compactness of $P = 1.5M$ with $\alpha = 0.55$. We then train our candidate architectures with the same self-supervised procedure proposed by [19] and adopted by the state-of-the-art approaches [1, 6, 42, 55]. After the search, we pick the best architecture (LiDNAS-K) to compare with the baselines and report the performance figures in Table 2. The LiDNAS-K model yields competitive results with the baselines while also being the smallest model. We observe that our proposed method provides noticeable improvement from PyD-Net and EQPyD-Net. Examples from Figure 7 show that the predicted depth maps from LiDNAS-K are more accurate and contain fewer artifacts.

ScanNet: For ScanNet, we set the target compactness to $4.5M$ with $\alpha = 0.57$ for searching. Despite of being compact, our best performance model (LiDNAS-S) produces competitive results compared with state-of-the-art methods, as shown in Table 3. More specifically, it requires only 20% of the number of parameters in comparison with the best performance baseline. We also observe that although SARPN [5] and Hu et al. [27] models are multiple times larger than DS-SIDENet [47] or DAV [28], the latter still yield better results, emphasizing the importance of optimal network structure. Furthermore, our model produces comparable depth maps as shown in Figure 10. Details of the generated architectures are provided in the supplementary material.

Table 2. Evaluation on the KITTI dataset. Metrics with ↓ mean lower is better and ↑ mean higher is better.

Method	**#params**	**REL↓**	**RMSE↓**	**δ_1↑**	**δ_2↑**	**δ_3↑**
FastDepth [55]	3.93M	0.156	5.628	0.801	0.930	0.971
PyD-Net [42]	1.97M	0.154	5.556	0.812	0.932	0.970
EQPyD-Net [6]	1.97M	0.135	5.505	0.821	0.933	0.970
DSNet [1]	1.91M	0.159	5.593	0.800	0.932	0.971
LiDNAS-K	**1.78M**	**0.133**	**5.157**	**0.842**	**0.948**	**0.980**

Runtime Measurement: We also compare the runtime of our models with state-of-the-art lightweight methods on an Android device using the app from the Mobile AI benchmark developed by Ignatov et al. [29]. To this end, we utilize the pre-trained models provided by the authors (Tensorflow [42], PyTorch [55]) and convert them to *tflite*. The results in Table 4 suggest that the proposed approaches produce competing performance, with the potential of running real-time on mobile devices with further optimization.

4.5. Ablation studies

Table 3. Evaluation results on ScanNet [7] dataset.

Architecture	**#params**	**REL**	**sqREL**	**SI**	**iMAE**	**iRMSE**
SARPN [5]	210.3M	0.134	0.077	**0.015**	0.093	0.100
Hu et al. [27]	157.0M	0.139	0.081	0.016	0.100	0.105
DS-SIDENet [47]	49.8M	0.133	**0.057**	-	-	-
DAV [28]	25.1M	0.118	**0.057**	**0.015**	**0.089**	**0.097**
LiDNAS-S	**5.2M**	**0.117**	0.059	**0.015**	0.090	**0.097**

Table 4. Average runtime comparison of the proposed method and other lightweight models. Runtime values are measured using a Pixel 3a phone with input image resolution (640 × 480).

Architecture	**CPU(ms)**
FastDepth [55]	458
Ef+FBNet [54, 56]	852
PyD-Net [42]	226
LiDNAS-K	**205**
LiDNAS-N	262
LiDNAS-S	380

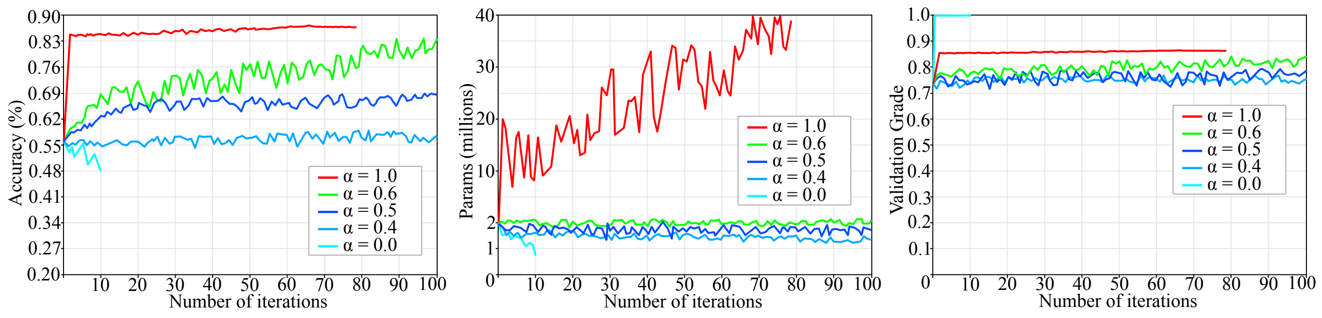

Figure 6. The progress of different searching scenarios on the NYU-Depth-v2 dataset. From left to right, charts show the accuracy, the number of parameters, and validation grade vs. the number of searching iterations.

Exploration Convergence: We experiment with various settings for the multi-objective balance coefficient (α) to assess its effect on the performance. For this purpose, we perform the architecture search with α set to 0.0, 0.4, 0.5, 0.6, and 1.0 while the target compactness $P = 2.0M$. Figure 6 presents the searching progress for accuracy (left), the number of parameters (center), and validation grade (right) from one parent architecture on NYU-Depth-v2. We observe that, scenario with $\alpha = 0.0$ quickly becomes saturated as it only gives reward to the smallest model. Searching with $\alpha = 0.4$ favors models with compact size but also with limited accuracy. The case with $\alpha = 0.5$ provides a more balanced option, but accuracy is hindered due to fluctuation during searching. The exploration with $\alpha = 1.0$ seeks for the network with the best accuracy yet producing significantly larger architecture while the case where $\alpha = 0.6$ achieves promising accuracy although with slightly bigger model than the target compactness.

Searching Scenarios: To further analyze the outcome of different searching scenarios, we perform architecture searches for *six* parent networks in five settings with $\alpha = 0.0, 0.4, 0.5, 0.6, 1.0$ and $P = 2.0M$ on NYU-Depth-v2. Results in Figure 8 show that best performance models in case $\alpha = 0.5$ are more spread out, while training instances with $\alpha = 0.6$ tend to produce both accurate and lightweight architectures. This, in turn, emphasizes the trade-off between validation accuracy and network size.

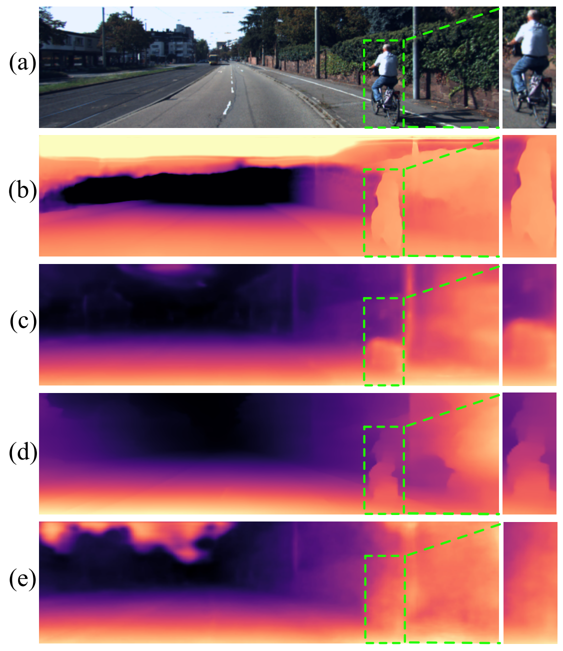

Figure 7. Comparison on the Eigen split of KITTI. (a) input image, (b) LiDNAS-K, (c) DSNet [1], (d) PyD-Net [42], and (e) FastDepth [55]. Images in the right column presented zoom-in view for better visualization.

5. Conclusion

This paper proposed a novel NAS framework to construct lightweight monocular depth estimation architectures using Assisted Tabu Search and employing a well-defined search space for balancing layer diversity and search volume. The proposed method achieves competitive accuracy on three benchmark datasets while running faster on mobile devices and being more compact than state-of-the-art handcrafted and automatically generated models. Our work provides a potential approach towards optimizing the accuracy and the network size for dense depth estimation without the need for manual tweaking of deep neural architectures.

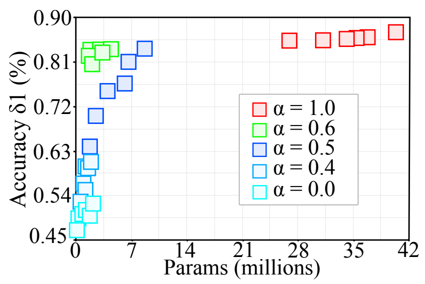

Figure 8. Trade-off between accuracy vs. the number of parameters of best models trained with different searching scenarios on NYU-Depth-v2 dataset.

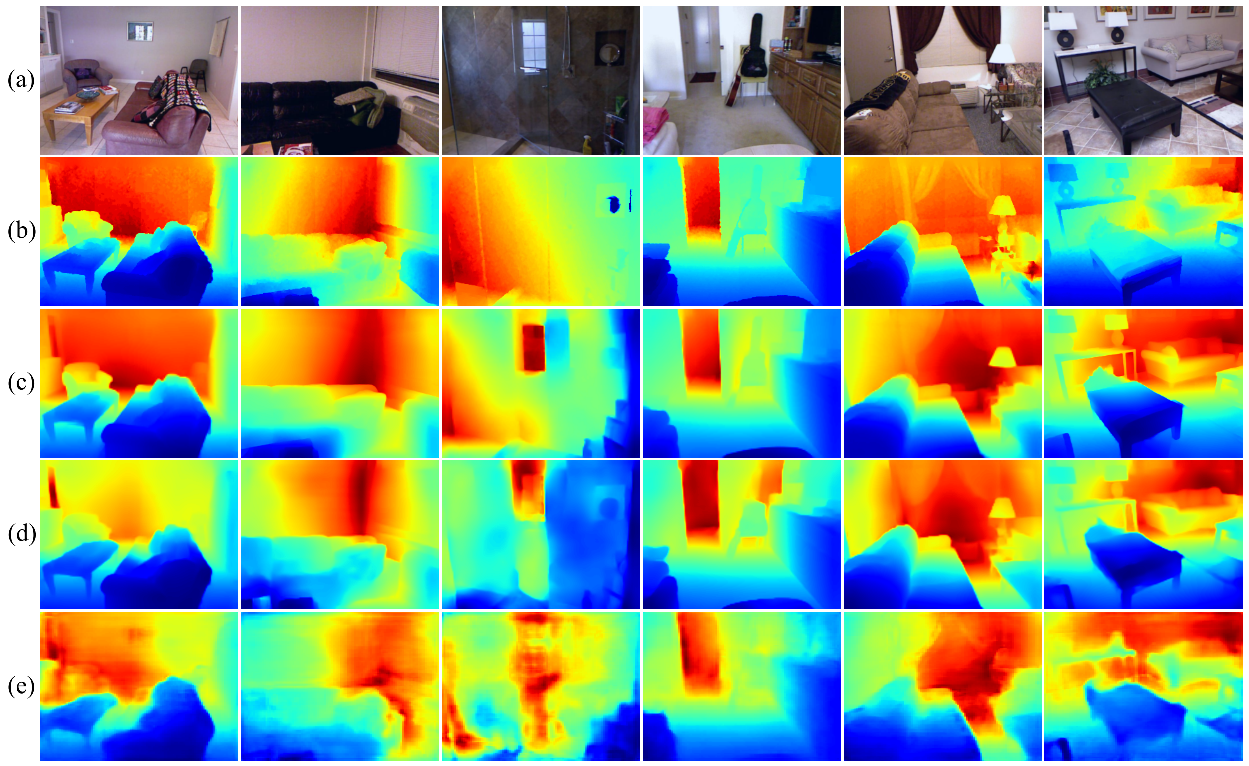

Figure 9. Comparison on the NYU test set. (a) input image, (b) ground truth, (c) LiDNAS-N, (d) Ef+FBNet [54, 56], and (e) FastDepth [55].

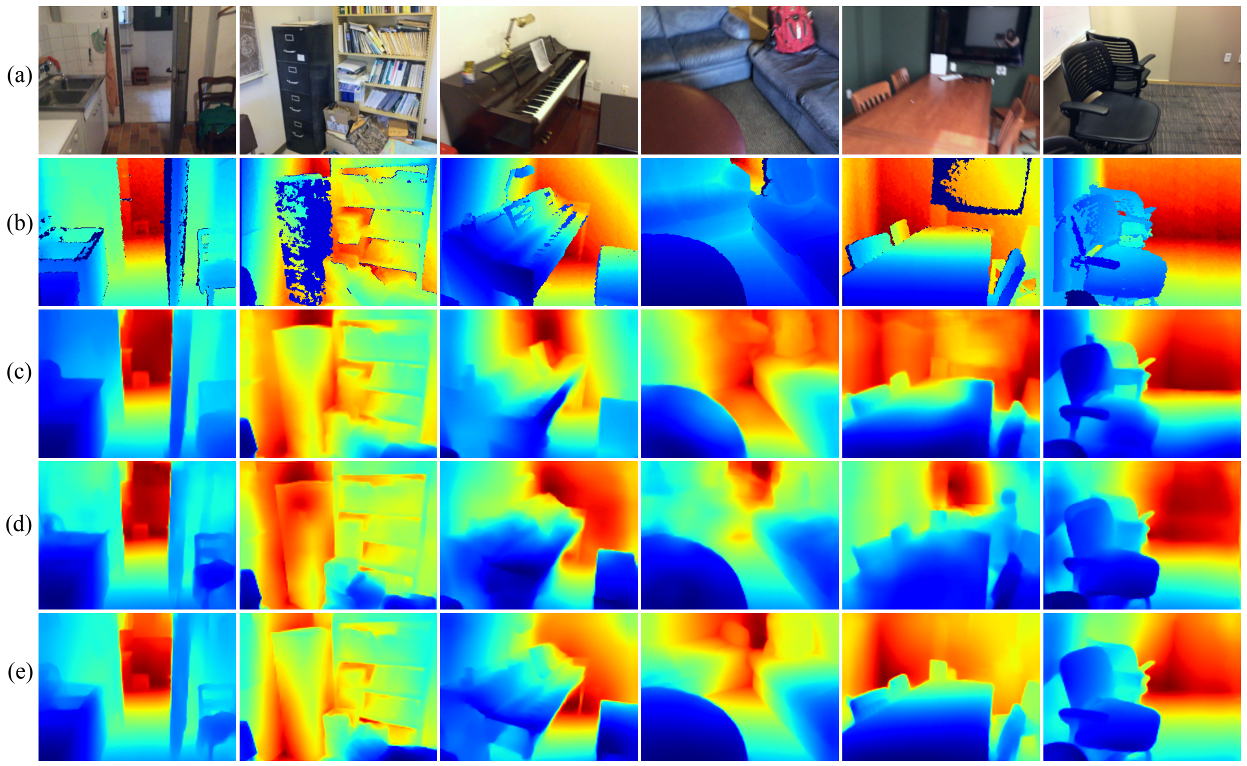

Figure 10. Comparison on the ScanNet test set. (a) input image, (b) ground truth, (c) LiDNAS-S, (d) DAV [28], and (e) SARPN [5].

References

[1] Filippo Aleotti, Giulio Zaccaroni, Luca Bartolomei, Matteo Poggi, Fabio Tosi, and Stefano Mattoccia. Real-time single image depth perception in the wild with handheld devices. *Sensors*, 21(1):15, 2021.

[2] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167*, 2016.

[3] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. Adabins: Depth estimation using adaptive bins. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4009–4018, 2021.

[4] Weifeng Chen, Zhao Fu, Dawei Yang, and Jia Deng. Single-image depth perception in the wild. *Advances in neural information processing systems*, 29:730–738, 2016.

[5] Xiaotian Chen, Xuejin Chen, and Zheng-Jun Zha. Structure-aware residual pyramid network for monocular depth estimation. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 694–700. AAAI Press, 2019.

[6] Antonio Cipolletta, Valentino Peluso, Andrea Calimera, Matteo Poggi, Fabio Tosi, Filippo Aleotti, and Stefano Mattoccia. Energy-quality scalable monocular depth estimation on low-power cpus. *IEEE Internet of Things Journal*, 2021.

[7] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5828–5839, 2017.

[8] Jin-Dong Dong, An-Chieh Cheng, Da-Cheng Juan, Wei Wei, and Min Sun. Dpp-net: Device-aware progressive search for pareto-optimal neural architectures. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 517–531, 2018.

[9] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658, 2015.

[10] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014.

[11] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Efficient multi-objective neural architecture search via lamarckian evolution. *arXiv preprint arXiv:1804.09081*, 2018.

[12] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Multi-objective architecture search for cnns. *arXiv preprint arXiv:1804.09081*, 2, 2018.

[13] Jose M Facil, Benjamin Ummenhofer, Huizhong Zhou, Luis Montesano, Thomas Brox, and Javier Civera. Cam-convs: camera-aware multi-scale convolutions for single-view depth. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11826–11835, 2019.

[14] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2002–2011, 2018.

[15] Ravi Garg, Vijay Kumar Bg, Gustavo Carneiro, and Ian Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *European conference on computer vision*, pages 740–756. Springer, 2016.

[16] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[17] Geiger, A. and Nießner, M. and Dai, A. Robust Vision Challenge CVPR Workshop, 2018.

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

[19] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3828–3838, 2019.

[20] Juan Luis GonzalezBello and Munchurl Kim. Forget about the lidar: Self-supervised depth estimators with med probability volumes. *Advances in Neural Information Processing Systems*, 33, 2020.

[21] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. Ghostnet: More features from cheap operations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1580–1589, 2020.

[22] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[23] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[24] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324, 2019.

[25] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[26] Chi-Hung Hsu, Shu-Huan Chang, Jhao-Hong Liang, Hsin-Ping Chou, Chun-Hao Liu, Shih-Chieh Chang, Jia-Yu Pan, Yu-Ting Chen, Wei Wei, and Da-Cheng Juan. Monas: Multi-objective neural architecture search using reinforcement learning. *arXiv preprint arXiv:1806.10332*, 2018.

[27] Junjie Hu, Mete Ozay, Yan Zhang, and Takayuki Okatani. Revisiting single image depth estimation: Toward higher resolution maps with accurate object boundaries. In *IEEE Winter Conf. on Applications of Computer Vision (WACV)*, 2019.

[28] Lam Huynh, Phong Nguyen-Ha, Jiri Matas, Esa Rahtu, and Janne Heikkilä. Guiding monocular depth estimation using

depth-attention volume. In *European Conference on Computer Vision*, pages 581–597. Springer, 2020.

[29] Andrey Ignatov, Grigory Malivenko, David Plowman, Samarth Shukla, Radu Timofte, Ziyu Zhang, Yicheng Wang, Zilong Huang, Guozhong Luo, Gang Yu, et al. Fast and accurate single-image depth estimation on mobile devices, mobile ai 2021 challenge: Report. *arXiv preprint arXiv:2105.08630*, 2021.

[30] Jianbo Jiao, Ying Cao, Yibing Song, and Rynson Lau. Look deeper into depth: Monocular depth estimation with semantic booster and attention-driven loss. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 53–69, 2018.

[31] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[32] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 239–248. IEEE, 2016.

[33] Jin Han Lee, Myung-Kyu Han, Dong Wook Ko, and Il Hong Suh. From big to small: Multi-scale local planar guidance for monocular depth estimation. *arXiv preprint arXiv:1907.10326*, 2019.

[34] Jae-Han Lee and Chang-Su Kim. Monocular depth estimation using relative depth maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2019.

[35] Chen Liu, Kihwan Kim, Jinwei Gu, Yasutaka Furukawa, and Jan Kautz. Planercnn: 3d plane detection and reconstruction from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4450–4459, 2019.

[36] Chen Liu, Jimei Yang, Duygu Ceylan, Ersin Yumer, and Yasutaka Furukawa. Planenet: Piece-wise planar reconstruction from a single rgb image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2579–2588, 2018.

[37] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *Proceedings of the European conference on computer vision (ECCV)*, pages 19–34, 2018.

[38] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[39] Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. Neural architecture optimization. *arXiv preprint arXiv:1808.07233*, 2018.

[40] Joe Mellor, Jack Turner, Amos Storkey, and Elliot J Crowley. Neural architecture search without training. In *International Conference on Machine Learning*, pages 7588–7598. PMLR, 2021.

[41] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning*, pages 4095–4104. PMLR, 2018.

[42] Matteo Poggi, Filippo Aleotti, Fabio Tosi, and Stefano Mattoccia. Towards real-time unsupervised monocular depth estimation on cpu. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5848–5854. IEEE, 2018.

[43] Xiaojuan Qi, Renjie Liao, Zhengzhe Liu, Raquel Urtasun, and Jiaya Jia. Geonet: Geometric neural network for joint depth and surface normal estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 283–291, 2018.

[44] Michael Ramamonjisoa and Vincent Lepetit. Sharpnet: Fast and accurate recovery of occluding contours in monocular depth estimation. *The IEEE International Conference on Computer Vision (ICCV) Workshops*, 2019.

[45] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. *ArXiv preprint*, 2021.

[46] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789, 2019.

[47] Haoyu Ren, Mostafa El-khamy, and Jungwon Lee. Deep robust single image depth estimation neural network using scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 37–45, 2019.

[48] Tonmoy Saikia, Yassine Marrakchi, Arber Zela, Frank Hutter, and Thomas Brox. Autodispnet: Improving disparity estimation with automl. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1812–1823, 2019.

[49] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[50] Ashutosh Saxena, Sung H Chung, and Andrew Y Ng. Learning depth from single monocular images. In *Advances in neural information processing systems*, pages 1161–1168, 2006.

[51] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, pages 746–760. Springer, 2012.

[52] Richard Szeliski. Structure from motion. In *Computer Vision*, pages 303–334. Springer, 2011.

[53] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019.

[54] Xiaohan Tu, Cheng Xu, Siping Liu, Renfa Li, Guoqi Xie, Jing Huang, and Laurence Tianruo Yang. Efficient monocular depth estimation for edge devices in internet of things. *IEEE Transactions on Industrial Informatics*, 17(4):2821–2832, 2020.

[55] Diana Wofk, Fangchang Ma, Tien-Ju Yang, Sertac Karaman, and Vivienne Sze. Fastdepth: Fast monocular depth esti-

mation on embedded systems. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6101–6108. IEEE, 2019.

[56] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10734–10742, 2019.

[57] Guanglei Yang, Hao Tang, Mingli Ding, Nicu Sebe, and Elisa Ricci. Transformers solve the limited receptive field for monocular depth prediction. *arXiv preprint arXiv:2103.12091*, 2021.

[58] Tien-Ju Yang, Andrew Howard, Bo Chen, Xiao Zhang, Alec Go, Mark Sandler, Vivienne Sze, and Hartwig Adam. Netadapt: Platform-aware neural network adaptation for mobile applications. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 285–300, 2018.

[59] Wei Yin, Yifan Liu, Chunhua Shen, and Youliang Yan. Enforcing geometric constraints of virtual normal for depth prediction. In *The IEEE International Conference on Computer Vision (ICCV)*, 2019.

[60] Mehmet Kerim Yucel, Valia Dimaridou, Anastasios Drosou, and Albert Saa-Garriga. Real-time monocular depth estimation with sparse supervision on mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2428–2437, 2021.

[61] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[62] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.