iNAS: Integral NAS for Device-Aware Salient Object Detection

Yu-Chao Gu[1*] Shang-Hua Gao[1*] Xu-Sheng Cao[1] Peng Du[2]
Shao-Ping Lu[1] Ming-Ming Cheng[1†]
[1]TKLNDST, CS, Nankai University [2]Huawei Technologies
https://mmcheng.net/inas/

Abstract

Existing salient object detection (SOD) models usually focus on either the backbone feature extractor or saliency head, ignoring the relations between them. A powerful backbone could still achieve sub-optimal performance with a weak saliency head and vice versa. Moreover, the balance between model performance and inference latency poses a great challenge to model design, especially when considering different deployment scenarios. Considering all components in an integral neural architecture search (iNAS) space, we propose a flexible device-aware search scheme that only trains the SOD model once and quickly finds the high-performance but low-latency models on multiple devices. An evolution search with latency group sampling (LGS) is proposed to explore the entire latency area of our enlarged search space. Models searched by iNAS achieve similar performance with SOTA methods but reduce the 3.8×, 3.3×, 2.6×, 1.9× latency on Huawei Nova6 SE, Intel Core CPU, the Jetson Nano and Nvidia Titan Xp. The code is released at https://mmcheng.net/inas/.

1. Introduction

Salient object detection (SOD) aims to segment the most attractive objects in the image [1, 56]. Served as a pre-processing step, SOD is required by many downstream applications, *i.e.,* image editing [8], image retrieval [19], visual tracking [21], and video object segmentation [16]. These applications often require the SOD model to be deployed with low inference latency on multiple devices, *i.e.,* GPUs, CPUs, mobile phones, and embedded devices. Each device has unique properties. For instance, GPUs are good at massively parallel computing [40] while the embedded devices are energy-friendly at the cost of a low computing budget [24]. Thus, different deployment scenarios require quite different designs of SOD models.

State-of-the-art (SOTA) SOD methods mostly design handcraft saliency heads [33, 41, 44, 74, 77] to aggregate multi-level features from the pre-trained backbone, *e.g.,* VGG [48], and ResNet [20]. However, the prohibitive inference latency often prevents them from been applied on other devices except for GPUs. Handcraft low-latency SOD models designed for resource-constrained scenarios [13,43] suffer from large performance drop due to the reduced representation ability. The dilemma between the model performance and the inference latency causes heavy workloads to design SOD models for different devices manually. Therefore, we aim at a device-aware search scheme to quickly find suitable low-latency SOD models on multiple devices.

There are several obstacles to achieve low-latency SOD models on different devices, as shown in Fig. 2. Firstly, the relative latency of operators varies among different devices due to different parallel computation abilities, IO bottlenecks, and implementations. Transfer the SOD model designed for one device to another would result in sub-optimal latency and performance. Secondly, conventional handcraft SOD models either design more powerful saliency heads [33, 41, 44, 77] or more efficient backbones [13, 43], while ignoring the relations between them. Similarly, most neural architecture search (NAS) methods either focus on the backbone for the classification task [32, 50] or incorporate a fixed segmentation head [30, 31], while ignoring the

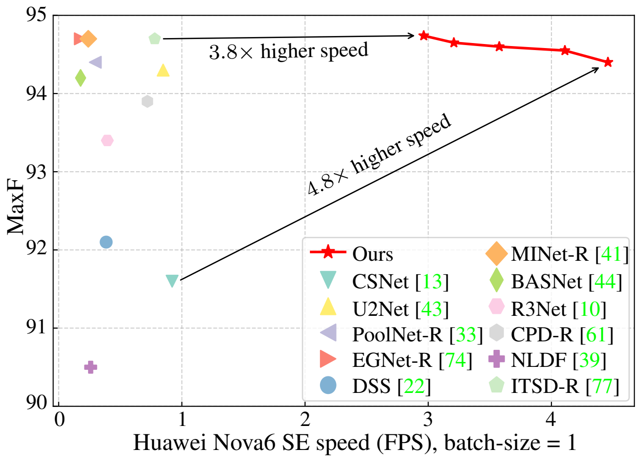


Figure 1. Mobile latency and performance comparison between our iNAS and recent state-of-the-art SOD models.

*Both authors contributed equally to this work.
†M.M. Cheng (cmm@nankai.edu.cn) is the corresponding author.

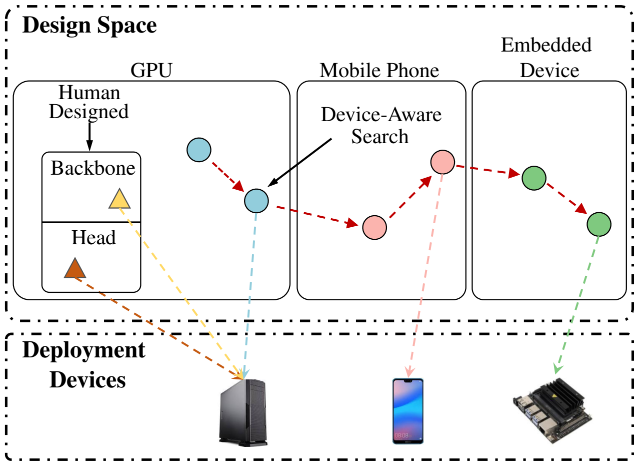


Figure 2. iNAS unifies the backbone and head design into an integral design space and specializes low-latency SOD models to different devices.

relationship between the backbone and head. We observe that a powerful backbone achieves sub-optimal efficiency with a weak saliency head and vice versa. These obstacles prevent the community from designing device-aware low-latency SOD models either with handcraft or NAS scheme.

To deal with these problems, we propose a device-aware search scheme with an integral search space to train the model once and quickly find high-performance but low-latency SOD models on multiple devices. Specifically, we propose an integral search space for SOD models that holistically consider the backbone and saliency head. To meet the multi-scale requirements of SOD models while avoiding the latency increased by the multi-branch structure, we construct a searchable multi-scale unit (SMSU). The SMSU supports searchable parallel convolutions with different kernel sizes and reparameterizes searched multi-branch convolutions to one branch for low inference latency. We also generalize the handcraft saliency heads [22, 33, 38, 41, 71] into the searchable transport and decoder parts, resulting in a rich saliency head search space for cooperating with the backbone space.

The proposed integral SOD search space contains nearly 10^{26} architectures, about 10^7 larger than the NAS space for the classification task [2]. Previous evolution search with layer-wise uniform sampling (LWUS) [2, 18, 68] uniformly sample the components layer-by-layer. Thus the accumulated latency of the whole sampled model obeys a polynomial distribution, *i.e.,* extremely low-latency or extremely high-latency area are under-sampled but the middle latency area is over-sampled. This imbalance sampling problem prevents LWUS from exploring the entire latency area of our enlarged search space. To overcome this imbalance sampling problem, we propose a latency group sampling (LGS) that introduces the device latency to guide sampling. Dividing the layer-wise search space into several latency groups and aggregating layer-wise samples in specific latency groups, LGS preserves the offspring in the under-sampled area but controls the samples of the over-sampled area. Compared with LWUS, the evolution search with LGS can explore the entire integral search space and finds a group of models on a higher and wider Pareto frontier.

The main contributions of this paper are:

- An integral SOD search space that considers the backbone-head relation and covers existing SOTA handcraft SOD designs.
- A device-aware evolution search with latency group sampling for exploring the entire latency area of the proposed search space.
- A thorough evaluation of the iNAS on five popular SOD datasets. Our method can reach a similar performance with handcraft SOTA methods but largely reduces inference latency on different devices, which helps to scale up the application of SOD to different deployment scenarios.

2. Related Work

2.1. Salient Object Detection.

Traditional SOD methods [1, 6, 52, 79] mainly rely on handcraft features and heuristic priors. [25, 26, 75] make an early attempt to use convolution neural networks (CNNs) to extract patch-level features. Inspired by FCN [38], the recent SOD methods [36, 54, 57] formulate SOD as a pixel-wise prediction task, which achieves large improvement over traditional or CNN-based methods. We refer readers to comprehensive surveys [1, 56, 78].

Most of the SOD methods handcraft the saliency head to effectively fuse the multi-scale information of the multi-level feature extracted by the pre-trained backbone, *e.g.,* ResNet [20] and VGG [48]. These methods [4, 14, 22, 33, 35, 55, 64] inherit an encoder-decoder structure, in which the decoder is responsible for the bottom-up feature fusion. Transport layers [12, 41, 70, 71, 76] are included inside the saliency head, enabling both the bottom-up and top-down feature fusion. Methods that introduce edge cues into the saliency head for precise boundary refinement [27, 59, 74] are orthogonal to our search space.

The gradually complicated SOD models bring improvements in the performance steadily while increasing prohibitive inference latency. Recent works [13, 16, 60, 61, 77] try to design light-weight models to eliminate the large inference latency. Among them, CPD [61] and ITSD [77] design light-weight saliency heads, achieving the fast speed on the CPU and GPU, respectively. CSNet [13] designs a light SOD backbone to achieve the low-latency on the mobile phone and embedded device. However, separating the design and the deployment devices causes sub-optimal latency when the hardware characteristics are quite different.

In this work, we introduce an integral search space that

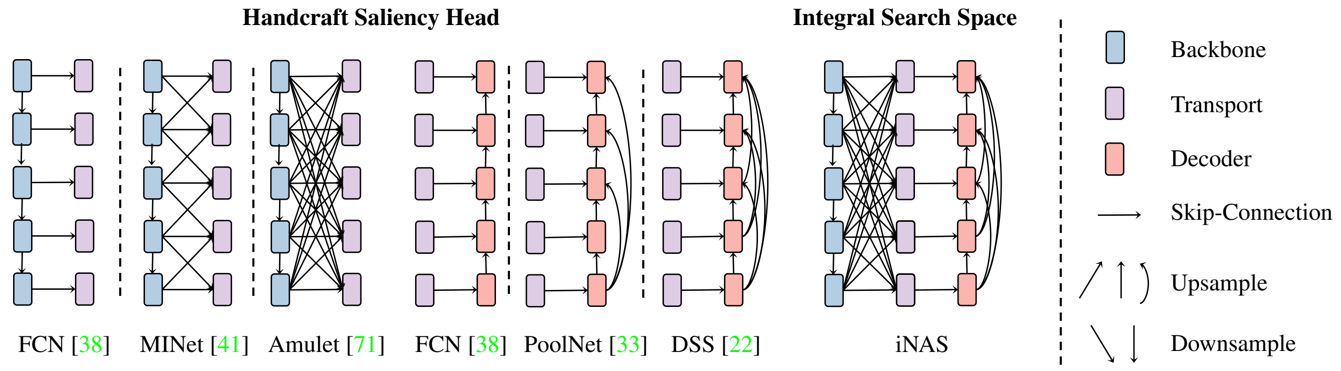


Figure 3. The designs of recent handcraft SOD models and the proposed integral search space.

Backbone						Transport			Decoder		
Stage	Operator	Resolutions	Channels	Layers	Kernel	Level	Kernel	Fusions	Level	Kernel	Fusions
stem	Conv	256x256-384x384	32-40	1	3	1	3,5,7,9	1-5	1	3,5,7,9	2-5
1	MBconv1	128x128-192x192	16-24	1-2	3						
2	MBconv6	128x128-192x192	24-32	2-3	3	2	3,5,7,9	1-5	2	3,5,7,9	2-4
3	MBconv6	64x64-96x96	32-48	2-3	3,5,7,9	3	3,5,7,9	1-5	3	3,5,7,9	2-3
4	MBconv6	32x32-48x48	64-88	2-4	3,5,7,9	4	3,5,7,9	1-5	4	3,5,7,9	2
5	MBconv6	32x32-48x48	96-128	2-6	3,5,7,9						
6	MBconv6	16x16-24x24	160-216	2-6	3,5,7,9	5	3,5,7,9	1-5	5	3,5,7,9	1
7	MBconv6	16x16-24x24	320-352	1-2	3,5,7,9						

Table 1. Detailed configurations of the proposed integral search space.

covers most of the handcraft SOD designs. Based on our integral search space, we propose a device-aware search scheme, which achieves similar performance to SOTA methods but largely reduces latency on different devices.

2.2. Neural Architecture Search.

Neural architecture search (NAS) demonstrates its potential to design efficient networks for various tasks automatically [15, 29, 31, 46, 69, 72]. Early reinforcement learning [80, 81] and evolutional algorithm [45, 62] based NAS methods train thousands of candidate architectures to learn a meta-controller, cost hundreds of GPU days to search. Later, differentiable NAS [17, 32] and one-shot NAS [2, 18, 68] exploit the idea of weight-sharing [42] to reduce the search cost, where the one-shot NAS decouples the supernet training and architecture search. Most one-shot NAS [2, 18, 68] targets improving the supernet training but simply adopt evolution search with layer-wise uniform sampling (LWUS). However, we find LWUS causes an imbalance sampling problem where most of the samples are located in the middle latency part of the search space.

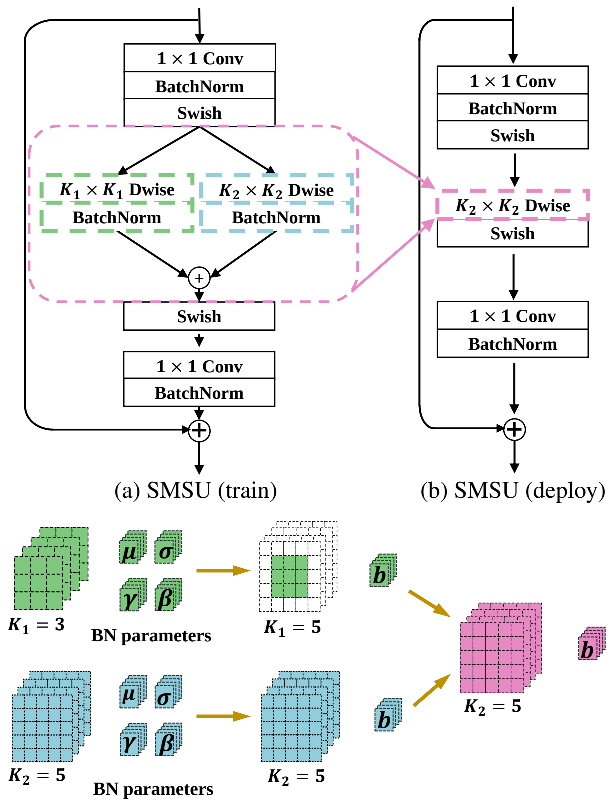
Apart from the search method, the search space plays a vital role in NAS. Early methods [32,42,45,62] utilize cell-based search space, where the cell is composed of multiple searchable operations. Based on cell-based search space, Auto-deeplab [31] additionally supports searching for the macro-structure of scale transformation. In order to adapt the segmentation task, Auto-deeplab incorporates fixed parallel ASPP [3] decoders. However, the searched structures of cell-based search space have complicated branch connections, which is hard to be parallelized in current deep learning frameworks [49], limiting its potential to low-latency applications. Exploiting the human expert knowledge, MnasNet [50] and the following works [9, 51, 58] develop a MobileNet [47] based search space, which supports more hardware-friendly architectures than cell-based search space. However, since these methods are designed for the classification task, it has less multi-scale representation capability and can not be directly applied to SOD.

Two design principles make the proposed iNAS different from the Auto-deeplab and MnasNet: 1) The integral search for all components reduces the overall inference latency; 2) The searchable multi-scale unit supports searching for the multi-branch structure without additional inference latency cost. To fully explore the proposed integral search space, we propose latency group sampling to address the imbalance sampling problem of the previous one-shot NAS [2, 18, 68]. Different from FairNAS [9], which aims to improve the fairness of optimizing different components in the supernet training stage, our proposed latency group sampling hopes to sample models of overall latency range in a balance way in the search stage.

3. Methodology

3.1. Integral SOD Design Space.

The previous handcraft SOD models [1, 36, 56] are mainly based on the fixed pre-trained backbone (*e.g.,* VGG [48] and ResNet [20]) and design saliency head to fuse the multi-level feature from the backbone. Some



(c) Reparameterization of multi-branch structure

Figure 4. Illustration of the searchable multi-scale unit (SMSU).

recent works have noticed that the pre-trained backbone accounts for most of the latency cost [13]. Instead of adopting a heavy backbone, they design light-weight backbones for SOD. However, both design strategies separate the backbone and decoder design, which hinder finding the low-latency high-performance SOD model in the integral design space. This section introduces an integral SOD design space, composed of the basic search unit (*i.e.*, searchable multi-scale unit) in Sec. 3.1.1 and the searchable saliency head in Sec. 3.1.2.

3.1.1 Searchable Multi-Scale Unit.

Since the previous general backbone accounts for most of the latency cost, the recent designs [13, 43] of the SOD backbone replace the vanilla convolution by group convolution [63] or separable convolution [47] for reducing latency. To capture the multi-scale representation in the image, they design several branches to encode features with different receptive fields and fuse the multi-scale features. However, the multi-branch structures are not hardware-friendly [47, 58, 73], which will slow down the inference speed. For example, the CSNet [13] has reduced 13.4× flops of the ITSD-R [77] but only achieves similar inference latency on the GPU. We thus propose the searchable multi-scale unit (SMSU), which supports finding the suitable multi-scale fusion automatically. The SMSU enables a multi-branch structure to capture multi-scale feature representation in training and adopt the reparameterization strategy [11] to fuse multiple branches into a single branch for fast inference.

We show a two-branches setting of SMSU in Fig. 4(a,b). The SMSU can extract multi-scale feature representation with different kernel sizes. Specifically, assume there are 3×3 conv and 5×5 conv, we denote the depthwise convolution parameters $W_1 \in \mathcal{R}^{C\times1\times3\times3}$ and $W_2 \in \mathcal{R}^{C\times1\times5\times5}$. The batch norm (BN) parameters following 3×3 conv and 5×5 conv are denoted as $\mu_1, \sigma_1, \gamma_1, \beta_1$ and $\mu_2, \sigma_2, \gamma_2, \beta_2$, respectively. Given the input feature $F_{in} \in \mathcal{R}^{C\times H\times W}$, we denote the conv output feature as $M = F_{in} * W$ where $*$ is the convolution. The fusion of two branches can be denoted as:

$$\begin{aligned} F_{out}^{(i)} =& (M_1^{(i)} - \mu_1^{(i)})\frac{\sigma_1^{(i)}}{\gamma_1^{(i)}} - \beta_1^{(i)} \\ &+ (M_2^{(i)} - \mu_2^{(i)})\frac{\sigma_2^{(i)}}{\gamma_2^{(i)}} - \beta_2^{(i)}, \end{aligned} \tag{1}$$

where i represent i-th channel. Eqn. (1) describes the training time multi-scale fusions in SMSU. In deployment, we merge the conv weight and its following BN parameters into a conv weight and bias, which is defined as:

$$V^{(i)} = \frac{\gamma^{(i)}}{\sigma^{(i)}} W^{(i)}, \qquad b^{(i)} = -\frac{\mu^{(i)}\gamma^{(i)}}{\sigma^{(i)}} + \beta^{(i)}, \tag{2}$$

where V is the merged conv weight and b is the bias. Then we zero-pad the small kernel in given branches to match the largest kernel. Finally, we average these two branches to get one single conv weight and bias.

The introduced two-branches fusion can be easily extended to any branches. Thus We enable searching for the fusion kernel combinations in the SMSU. We replace the inverted bottleneck of MobileNet search space with SMSU and summarize the search space in Tab. 1.

3.1.2 Searchable Saliency Head.

Previous handcraft saliency head incorporates the transport or decoder to fuse the multi-level features from the backbone. The high-level feature provides a rough location of the salient object, and the low-level feature provides the detailed feature for recovering the edge and boundary. As shown in Fig. 3, the typical transport design [41, 71] enables both the bottom-up and top-down fusion of the multi-level features. Our searchable transport connects to the all resolution levels of the backbone. Each level of our largest child transport can aggregate the feature from all five resolution levels like Amulet [71], while each level of our smallest

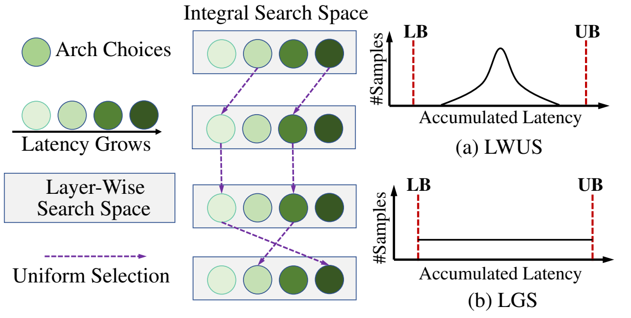


Figure 5. Illustration of the layer-wise uniform sampling (LWUS) and proposed latency-group sampling (LGS). LB: lower bound. UB: upper bound.

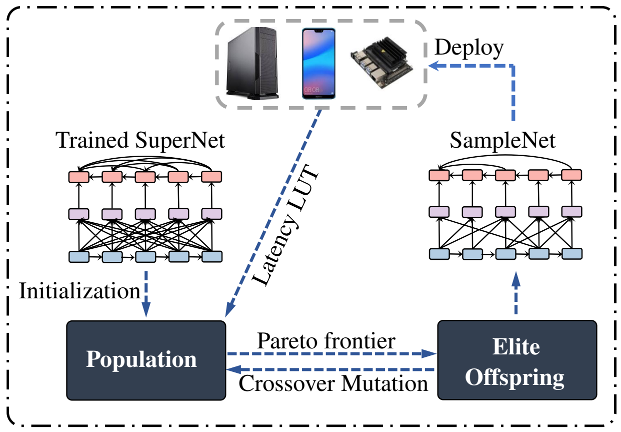


Figure 6. Illustration of iNAS search and deployment.

child transport only keeps the identity branches like FCN [38]. The downsample and the upsample branches are composed of 1×1 Conv-BN and maxpool operation/bilinear interpolation. Considering the best receptive field varies from different resolution levels, we also support searching for the fusion kernel in the transport. Our searchable transport covers many SOTA SOD transport designs [12, 34, 41, 76].

Unlike the transport, the decoder [22, 33] only supports a bottom-up prediction refinement and gradually fuses the low-level feature to recover the boundary. Thus we do not support top-down fusion branches in the decoder. The identity branch and the upsample branch from the adjacent resolution level are fixed, while other branches are searchable. The largest child decoder has a similar structure to DSS [22], while the smallest child decoder is like FCN [38]. We also support the searchable fusion kernel in the decoder. The searchable decoder also covers many handcraft SOD decoder designs [4, 35, 55, 61, 64].

Through the multi-scale fusion is proven to be effective in the SOD, how to prune the redundant fusion branch and choose appropriate fusion kernel with the latency constraint is a labor-intensive work for the human. Our proposed saliency head makes these key components searchable, automatically designed with latency constraints.

Algorithm 1: Evolution Search with LGS

```
Input: Trained supernet, initial population size $N$,
       latency lookup table (LUT), latency groups $G$,
       offspring size $k$, crossover probability $p_c$,
       mutation probability $p_m$, iteration $iter$.
Output: Pareto frontier of population $P$.
1  Sample the smallest and largest child model
     (i.e., $\textbf{arch}_{min}$ and $\textbf{arch}_{max}$);
2  Compute the lower bound and upper bound latency
     (i.e., $\textbf{LAT}_{min}$ and $\textbf{LAT}_{max}$) based on LUT;
3  Divide the $(\textbf{LAT}_{min}, \textbf{LAT}_{max})$ into $G$ groups;
4  Sample $\frac{N}{G}$ child models for each latency group
     $\{P_i | i = 1 \dots G\}$;
5  Population $P = P_1 \cup \dots \cup P_G$;
6  Evaluate performance for models in $P$;
7  for $j = 1 \dots iter$ do
8      for each $P_i$ do
9          $S_i \leftarrow$ Select $\frac{k}{G}$ models from the Pareto frontier
             of each latency group $P_i$;
10     $S = S_1 \cup \dots \cup S_G$;
11     for each model in $S$ do
12         Crossover and mutate under probability $p_c, p_m$.
13     Evaluate performance for models in $S$;
14     $P = P \cup S$
15 $P \leftarrow$ Select Pareto frontier of $P$;
16 Return $P$
```

3.2. Latency Group Sampling.

Previous one-shot methods adopt evolution search with layer-wise uniform sampling (LWUS), which causes an imbalance sampling problem. As illustrated in Fig. 5, the whole search space is composed of the layer-wise search space. The components in the layer-wise search space vary in latency. Suppose we uniformly sample the components layer-by-layer, the accumulated latency of the whole sampled model will obey a Polynomial distribution, *i.e.,* the extremely low-latency or extremely high-latency area are under-sampled but the middle latency area are over-sampled. To explore the entire latency area of the integral search space, we propose latency-group sampling (LGS). Given a latency lookup table (LUT), we divide the layer-wise search space into several latency groups. To sample a model in specific latency group, we sample layer-wise configuration in this latency group. And also, we preserve the elite offspring in the under-sampled area but controls the samples of the over-sampled area.

The general pipeline of the device-aware evolution is depicted in Fig. 6. We first build a latency lookup table (LUT) of the target device. Then we perform the evolution search based on LGS. After searching, the searched model inherits the supernet weight and can be directly deployed without retraining. As shown in Algorithm. 1, the evolution search

Method	FLOPs (G)	Latency (ms) GPU	Latency (ms) Embedded	ECSSD(1000) maxF	MAE	S_m	DUT-O(5168) maxF	MAE	S_m	DUTS-TE(5019) maxF	MAE	S_m	HKU-IS(4447) maxF	MAE	S_m	PASCAL-S(850) maxF	MAE	S_m
VGG-16/VGG-19																		
NLDF$_{\text{CVPR17}}$ [39]	66.68	9.48	505.59	0.905	0.063	0.875	0.753	0.080	0.770	0.813	0.065	0.805	0.902	0.048	0.879	0.822	0.098	0.805
DSS$_{\text{CVPR17}}$ [22]	48.75	5.85	N/A	0.921	0.052	0.882	0.781	0.063	0.790	0.825	0.056	0.812	0.916	0.040	0.878	0.831	0.093	0.798
PiCANet$_{\text{CVPR18}}$ [36]	59.82	34.21	N/A	0.931	0.046	0.914	0.794	0.068	0.826	0.851	0.054	0.861	0.921	0.042	0.906	0.856	0.078	0.848
CPD-V$_{\text{CVPR19}}$ [61]	24.08	3.78	266.40	0.936	0.040	0.910	0.793	0.057	0.818	0.864	0.043	0.866	0.924	0.033	0.904	0.861	0.072	0.845
ITSD-V$_{\text{CVPR20}}$ [77]	17.08	9.97	494.93	0.939	0.040	0.914	0.807	0.063	0.829	0.876	0.042	0.877	0.927	0035	0.906	0.869	0.068	0.856
PoolNet-V$_{\text{CVPR19}}$ [33]	48.80	8.81	N/A	0.941	0.042	0.917	0.806	0.056	0.833	0.876	0.042	0.878	-	-	-	0.865	0.072	0.852
EGNet-V$_{\text{ICCV19}}$ [74]	120.15	11.58	N/A	0.943	0.041	0.919	0.809	0057	0.836	0.877	0.044	0.878	0.930	0.034	0.912	0.858	0.077	0.848
MINet-V$_{\text{CVPR20}}$ [41]	71.76	14.78	N/A	0.943	0.036	0.919	0.794	0.057	0.822	0.877	0.039	0.875	0.930	0.031	0.912	0.865	0.064	0.854
ResNet-34/ResNet-101/ResNetXt-101																		
R3Net$_{\text{IJCAI18}}$ [10]	26.19	6.70	335.14	0.934	0.040	0.910	0.795	0.063	0.817	0.831	0.057	0.835	0.916	0.036	0.895	0.835	0.092	0.807
CPD-R$_{\text{CVPR19}}$ [61]	7.19	2.52	124.09	0.939	0.037	0.918	0.797	0.056	0.825	0.865	0.043	0.869	0.925	0.034	0.906	0.859	0.071	0.848
BASNet$_{\text{CVPR19}}$ [44]	97.51	16.37	N/A	0.942	0.037	0.916	0.805	0.056	0.836	0.859	0.048	0.865	0.928	0.032	0.909	0.854	0.076	0.838
PoolNet-R$_{\text{CVPR19}}$ [33]	38.17	9.13	N/A	0.944	0.039	0.921	0.808	0.056	0.836	0.880	0.040	0.883	0.932	0.033	0.916	0.863	0.075	0.849
EGNet-R$_{\text{ICCV19}}$ [74]	120.85	12.01	N/A	0.947	0.037	0.925	0.815	0.053	0.841	0.888	0.039	0.887	0.935	0.031	0.917	0.865	0.074	0.852
MINet-R$_{\text{CVPR20}}$ [41]	42.68	7.38	N/A	0.947	0.033	0.925	0.810	0.056	0.833	0.884	0.037	0.884	0.935	0.029	0.919	0.867	0.064	0.856
ITSD-R$_{\text{CVPR20}}$ [77]	9.65	3.57	164.76	0.947	0.034	0.925	0.820	0.061	0.840	0.882	0.041	0.884	0.934	0.031	0.917	0.870	0.066	0.859
Handcraft SOD Backbone																		
CSNet$_{\text{ECCV20}}$ [13]	0.72	3.63	95.75	0.916	0.065	0.893	0.775	0.081	0.805	0.813	0.075	0.822	0.898	0.059	0.881	0.828	0.103	0.813
U^2-Net$_{\text{PR20}}$ [43]	9.77	4.45	173.61	0.943	0.041	0.918	0.813	0.060	0.837	0.852	0.054	0.858	0.928	0.037	0.908	0.847	0.086	0.831
Searched Models on Different Devices																		
iNAS(GPU)-S	0.43	**1.32**	48.56	0.944	0.037	0.921	0.819	0.055	0.842	0.872	0.043	0.875	0.930	0.033	0.914	0.864	0.071	0.852
iNAS(Embedded)-S	**0.41**	1.53	**40.99**	0.944	0.038	0.920	0.816	0.056	0.840	0.871	0.043	0.875	0.931	0.033	0.915	0.865	0.070	0.852
iNAS(GPU)-L	0.70	1.94	71.70	0.947	0.036	0.924	0.824	0.052	0.846	0.879	0.040	0.881	0.935	0.031	0.918	0.867	0.071	0.852
iNAS(Embedded)-L	0.63	2.30	63.39	0.947	0.036	0.924	0.820	0.055	0.842	0.875	0.041	0.879	0.935	0.031	0.919	0.865	0.070	0.852

Table 2. Comparison with existing SOD methods. The FLOPs and latency is measured with 224 $\times$ 224 input image. N/A means it could not be deployed on the embedded device because of the out-of-memory error.

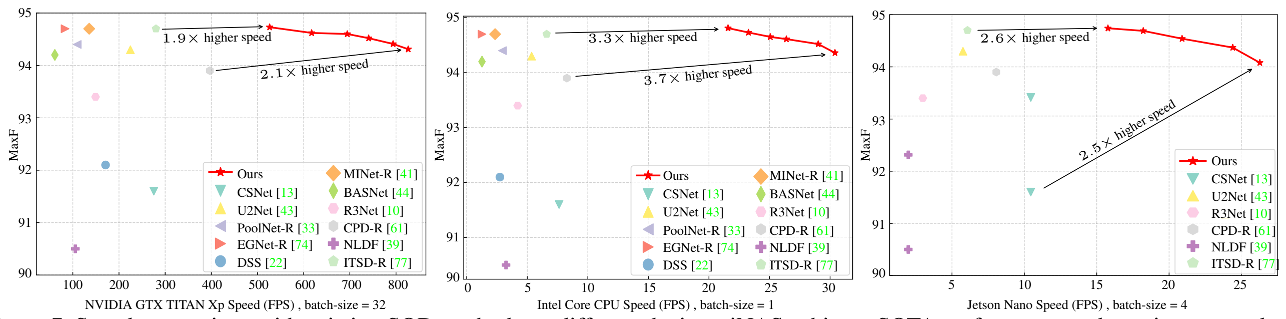


Figure 7. Speed comparison with existing SOD methods on different devices. iNAS achieves SOTA performance and consistent speedup.

with LGS contains four stages:

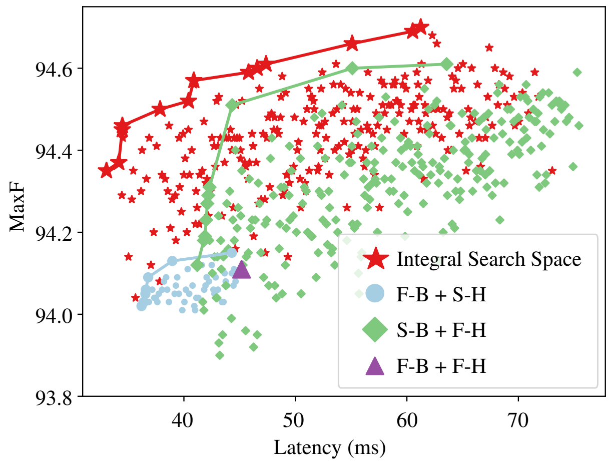
- **S1: Initialization.** We sample the smallest and largest child model from the search space and compute the lower bound and upper bound latency. The search space is divided into G latency groups. We sample N candidates in the initial population $\mathbf{P}$, where each latency group has $\frac{n}{G}$ samples.
- **S2: Selection.** We select k models from the Pareto frontier of P into candidate set S, where each latency group selects $\frac{k}{G}$ samples.
- **S3: Crossover.** For each model in S, it has a probability of p_c to crossover with another model in S. We allow swap the stage-wise configuration in backbone and swap level-wise configuration in the head.
- **S4: Mutation.** For each model in S, each configuration has a probability of p_m to mutate. Then we merge the S into the population P and continue to S2 until target iterations $iter$.

The main difference between LGS and LWUS is in the initialization and selection. In the initialization step, LGS balances the samples in the different latency area While LWUS over-samples the middle latency area. And in the selection step, LGS preserves a certain number of elite offspring in the under-sampled area, but LWUS drops the elite offspring in the under-sampled area. We compare the LGS and LWUS in Sec. 4.2 and find the LWUS only explore a limited latency area of the whole search space.

4. Experiments

4.1. Implementation Details.

Details of supernet training. We implement iNAS using the Pytorch [49] and Jittor [23]. We organize the search space as a nested one-shot supernet as [2, 68]. The small



(a) Search space exploration. F: fixed, S: searchable, B: backbone, H: head.

Searchable		Low Latency Arch		High Performance Arch	
Backbone	Head	Latency (ms)	maxF	Latency (ms)	maxF
✗	✗	45.17	0.941	45.17	0.941
✓	✗	41.20	0.941	63.56	0.946
✗	✓	36.20	0.940	44.30	0.942
✓	✓	**33.06**	**0.944**	**61.24**	**0.947**

(b) Quantitative analysis of the integral and partial search.

Figure 8. Comparison between the integral and partial search.

kernel size is the center part of the largest kernel and the lower-index channels and layers are sharing. The one-shot supernet is trained on DUTS-TR for 100 epochs with ImageNet pre-training. The training batch size is set to 40. We use the Adam optimizer with a learning rate of 1e-4 and the poly learning rate schedule [37]. We sample the largest, the smallest, and two middle models for each iteration and fuse their gradient to update the supernet. The largest child model minimizes the binary cross-entropy with the ground-truth but the other models minimize the mean squared error with the largest child model prediction. Following [22], we add deep supervision on the prediction of each decoder level. The supernet training costs 17 hours on four Tesla V100.

Details of search and deployment. We set the initial population size N to 1000, and the latency group G to 10. The evolution iteration $iter$ is set to 20. Each selection retains $k = 100$ offspring. The crossover and mutation probability (p_c and p_m) are set to 0.2. We finetune the BN of each sample model on the training set for 200 iterations [67]. We use the Pytorch-Mobile [49] library to build the latency LUT on the mobile phone. The search phase costs 0.8 GPU-Days on one Tesla V100 GPU for each device.

Dataset. The supernet is trained with the DUTS-TR dataset [53]. We conduct evaluations on five popular SOD datasets, *i.e.,* ECSSD [65], DUT-O [66], DUTS-TE [53], HKU-IS [25], PASCAL-S [28], containing 1000, 5168, 5019, 4447, and 850 pairs of images and saliency maps, respectively.

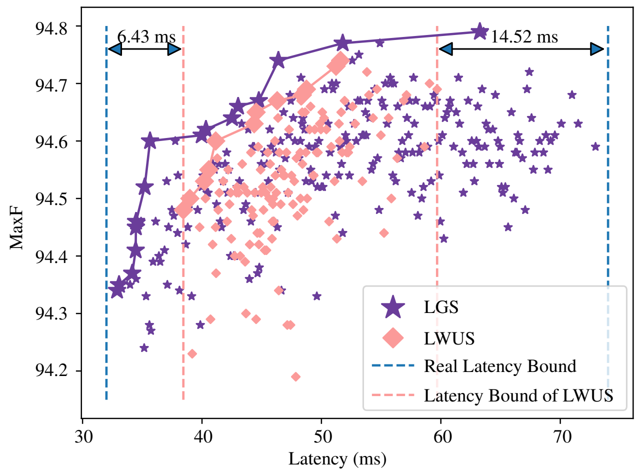


Figure 9. Comparison of the evolution search with layer-wise uniform sampling (LWUS) and proposed LGS.

Search Dev.	Latency (ms)			
	GPU	CPU	Mobile	Embedded
GPU	1.94	48.90	397.17	71.70
Device-Aware	1.94	42.99	339.61	63.39
Latency Reduction	0%	12.1%	14.5%	10.9%

Table 3. Comparison of searching on GPU and specialized device.

Evaluation metrics. Following common settings [36, 44], we use MAE [7], Max F-measure (F_β) [6] and S-measure (S_m) [5] as the evaluation metrics to evaluate our results. Since we aim to design low-latency SOD models, the inference latency is also used as the evaluation metric.

4.2. Performance Evaluation.

Comparison to the state-of-the-art. Tab. 2 shows the comparison between our searched models and previous handcraft SOTA SOD methods. The iNAS(GPU)-L, the large model searched on GPU, requires similar FLOPs to the CSNet, but reduces 47% inference latency and improves 3.1% F_β on ECSSD, which suggests the FLOPs are not highly related to the inference latency. We also show the latency comparison of our searched models on different devices in Fig. 1 and Fig. 7. Our method achieves similar performance to the SOTA but reduces 1.9×, 3.3×, 2.6×, 3.8× latency on GPU, CPU, embedded device, and mobile phone, respectively. Compared to the previous fastest methods, the fastest models searched by iNAS speed up 2.1×, 3.7×, 2.5×, and 4.8× on these devices. Current SOD models are mostly designed for GPU while ignoring other devices. Some ResNet-based and VGG-based methods can not even be applied to the embedded device due to the out-of-memory error. In comparison, our device-aware searched models achieve consistent latency reduction on all

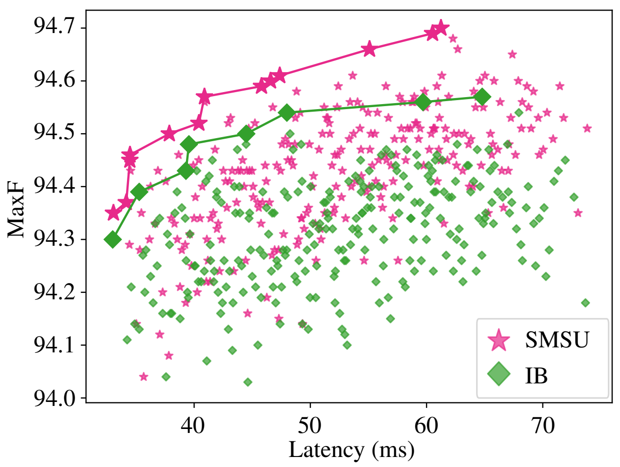


Figure 10. Comparison of the search space constructed by the inverted bottleneck (IB) [47] and our proposed searchable multi-scale unit (SMSU).

devices.

Device-aware search. To verify the effectiveness of device-aware search, we compare the models searched on GPU and other devices in Tab. 3. We firstly benchmark the latency of the iNAS(GPU)-L on other devices. With aligned performance, models searched on specialized devices achieve 12.1%, 14.5%, 10.9% latency reductions on CPU, mobile phone, and embedded device compared with iNAS(GPU)-L. This observation verifies the device-aware search can find the suitable model for the target device to reduces the latency.

Integral search space. iNAS supports an integral search space for SOD. Fig. 8 verifies the importance of the integral search space. For the baseline network, we use the MobileNetV2 structure [47] as the fixed backbone and combine the Amulet transport [71] and DSS decoder [22] to form the fixed saliency head. As shown in Fig. 8(b), the fixed baseline network requires 45.17 ms inference latency on CPU and gets 94.1% on ECSSD. Only enabling the searchable backbone or searchable saliency head reduces the latency lower bound to 41.20 ms (-8.7%) or 36.20 ms (-19.8%) with similar performance. While using the integral search space greatly reduces the latency lower bound to 33.06 ms (-26.8%) but improves the performance of the fastest architecture to 94.4%. Similarly, the performance upper bound is promoted to 94.7%. Fig. 8(a) shows the integral search space has a consistently better Pareto frontier over partial searchable space and significantly improves the handcraft structure on both the latency and performance.

Latency group sampling. Fig. 9 compares the evolution search based on the layer-wise uniform sampling (LWUS) and proposed latency group sampling (LGS). The lower bound and upper bound latency of the search space is 32.1 ms and 74.1 ms, respectively. As shown in Fig. 9, the lower bound and upper bound latency of LWUS are 38.5 ms and 59.6 ms, which only account for 50.2% of the whole search space. While our proposed LGS ensures each latency group with sufficient samples and offspring, thus can explore 99% of the search space. As a result, our proposed LGS obtains broader Pareto frontiers over LWUS.

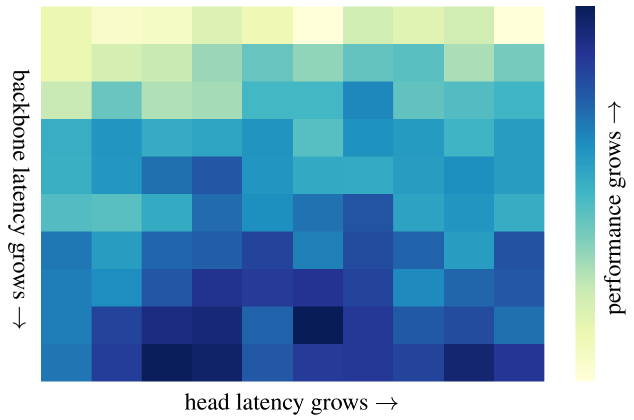


Figure 11. Visualization of the correspondence between the backbone/head latency and the performance.

Searchable multi-scale unit. Fig. 10 verifies the effectiveness of the proposed searchable multi-scale unit (SMSU). We compare the search space constructed by the SMSU with the inverted bottleneck (IB). Enhancing the IB with multi-scale ability, search space constructed by SMSU shows a better latency-performance Pareto frontier over that constructed by IB. We observe that the improvement of higher latency models is much larger, which we assume that relaxed latency constrain enables large kernel, which has more powerful multi-scale kernel combinations.

4.3. Observation

In order to explore the relation of performance with the backbone latency and the head latency, we divide the backbone and the head latency into 10 groups and sample 20 models in each grid, resulting in 2000 samples. Observing Fig. 11, we find (1) a more complicated backbone consistently improves the performance; (2) while the complicated saliency head is not always the best choice. These observations show there is still room for reducing the model latency by searching in our search space. Also, choosing an appropriate saliency head for better latency-performance balance has no apparent pattern, suggesting the searching may be the efficient solution to design better SOD models.

5. Conclusion

In this work, we propose an integral search (iNAS) space for SOD, which generalizes the designs of handcraft SOD models. The integral search can automatically find correspondence between backbone and head and get the best performance-latency balance. Then we propose a latency-group sampling to explore our entire integral search space. The experiment demonstrates that the iNAS has similar performance to the handcraft SOTA SOD methods but largely reduces their latency in various devices. Our work paves the way for SOD applications on low-power devices.

Acknowledgement. This research was supported by NSFC (61922046), S&T innovation project from Chinese Ministry of Education, and the Fundamental Research Funds for the Central Universities (Nankai University, NO. 63213090).

References

[1] Ali Borji, Ming-Ming Cheng, Qibin Hou, Huaizu Jiang, and Jia Li. Salient object detection: A survey. *Computational Visual Media*, 5(2):117–150, 2019. 1, 2, 3

[2] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. In *Int. Conf. Learn. Represent.*, 2020. 2, 3, 6

[3] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(4):834–848, 2017. 3

[4] Shuhan Chen, Xiuli Tan, Ben Wang, and Xuelong Hu. Reverse attention for salient object detection. In *Eur. Conf. Comput. Vis.*, pages 234–250, 2018. 2, 5

[5] Ming-Ming Cheng and Deng-Ping Fan. Structure-measure: A new way to evaluate foreground maps. *Int. J. Comput. Vis.*, 129(9):2622–2638, 2021. 7

[6] Ming-Ming Cheng, Niloy J Mitra, Xiaolei Huang, Philip HS Torr, and Shi-Min Hu. Global contrast based salient region detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(3):569–582, 2015. 2, 7

[7] Ming-Ming Cheng, Jonathan Warrell, Wen-Yan Lin, Shuai Zheng, Vibhav Vineet, and Nigel Crook. Efficient salient region detection with soft image abstraction. In *Int. Conf. Comput. Vis.*, pages 1529–1536, 2013. 7

[8] Ming-Ming Cheng, Fang-Lue Zhang, Niloy J Mitra, Xiaolei Huang, and Shi-Min Hu. Repfinder: finding approximately repeated scene elements for image editing. *ACM Trans. Graph.*, 29(4):1–8, 2010. 1

[9] Xiangxiang Chu, Bo Zhang, and Ruijun Xu. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. In *Int. Conf. Learn. Represent.*, 2021. 3

[10] Zijun Deng, Xiaowei Hu, Lei Zhu, Xuemiao Xu, Jing Qin, Guoqiang Han, and Pheng-Ann Heng. R3net: Recurrent residual refinement network for saliency detection. In *IJCAI*, pages 684–690, 2018. 1, 6

[11] Xiaohan Ding, Yuchen Guo, Guiguang Ding, and Jungong Han. Acnet: Strengthening the kernel skeletons for powerful cnn via asymmetric convolution blocks. In *Int. Conf. Comput. Vis.*, pages 1911–1920, 2019. 4

[12] Mengyang Feng, Huchuan Lu, and Errui Ding. Attentive feedback network for boundary-aware salient object detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1623–1632, 2019. 2, 5

[13] Shang-Hua Gao, Yong-Qiang Tan, Ming-Ming Cheng, Chengze Lu, Yunpeng Chen, and Shuicheng Yan. Highly efficient salient object detection with 100k parameters. In *Eur. Conf. Comput. Vis.*, 2020. 1, 2, 4, 6

[14] Yanliang Ge, Cong Zhang, Kang Wang, Ziqi Liu, and Hongbo Bi. Wgi-net: A weighted group integration network for rgb-d salient object detection. *Computational Visual Media*, 7(1):115–125, 2021. 2

[15] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 7036–7045, 2019. 3

[16] Yuchao Gu, Lijuan Wang, Ziqin Wang, Yun Liu, Ming-Ming Cheng, and Shao-Ping Lu. Pyramid constrained self-attention network for fast video salient object detection. In *AAAI*, pages 10869–10876, 2020. 1, 2

[17] Yu-Chao Gu, Li-Juan Wang, Yun Liu, Yi Yang, Yu-Huan Wu, Shao-Ping Lu, and Ming-Ming Cheng. Dots: Decoupling operation and topology in differentiable architecture search. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 12311–12320, 2021. 3

[18] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *Eur. Conf. Comput. Vis.*, pages 544–560, 2020. 2, 3

[19] Junfeng He, Jinyuan Feng, Xianglong Liu, Cheng Tao, and S F Chang. Mobile product search with bag of hash bits and boundary reranking. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2012. 1

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 770–778, 2016. 1, 2, 3

[21] Seunghoon Hong, Tackgeun You, Suha Kwak, and Bohyung Han. Online tracking by learning discriminative saliency map with convolutional neural network. In *ICML*, pages 597–606, 2015. 1

[22] Qibin Hou, Ming-Ming Cheng, Xiaowei Hu, Ali Borji, Zhuowen Tu, and Philip Torr. Deeply supervised salient object detection with short connections. *IEEE Trans. Pattern Anal. Mach. Intell.*, 41(4):815–828, 2019. 1, 2, 3, 5, 6, 7, 8

[23] Shi-Min Hu, Dun Liang, Guo-Ye Yang, Guo-Wei Yang, and Wen-Yang Zhou. Jittor: a novel deep learning framework with meta-operators and unified graph execution. *Science China Information Sciences*, 63(222103):1–21, 2020. 6

[24] Agus Kurniawan. Introduction to nvidia jetson nano, 2021. 1

[25] Guanbin Li and Yizhou Yu. Visual saliency based on multiscale deep features. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5455–5463, 2015. 2, 7

[26] Guanbin Li and Yizhou Yu. Deep contrast learning for salient object detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 478–487, 2016. 2

[27] Xin Li, Fan Yang, Hong Cheng, Wei Liu, and Dinggang Shen. Contour knowledge transfer for salient object detection. In *Eur. Conf. Comput. Vis.*, pages 355–370, 2018. 2

[28] Yin Li, Xiaodi Hou, Christof Koch, James M Rehg, and Alan L Yuille. The secrets of salient object segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 280–287, 2014. 7

[29] Yingwei Li, Xiaojie Jin, Jieru Mei, Xiaochen Lian, Linjie Yang, Cihang Xie, Qihang Yu, Yuyin Zhou, Song Bai, and Alan L Yuille. Neural architecture search for lightweight non-local networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 10297–10306, 2020. 3

[30] Yanwei Li, Lin Song, Yukang Chen, Zeming Li, Xiangyu Zhang, Xingang Wang, and Jian Sun. Learning dynamic routing for semantic segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 8553–8562, 2020. 1

[31] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 82–92, 2019. 1, 3

[32] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *Int. Conf. Learn. Represent.*, 2019. 1, 3

[33] Jiang-Jiang Liu, Qibin Hou, Ming-Ming Cheng, Jiashi Feng, and Jianmin Jiang. A simple pooling-based design for real-time salient object detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3917–3926, 2019. 1, 2, 3, 5, 6

[34] Jiang-Jiang Liu, Zhi-Ang Liu, and Ming-Ming Cheng. Centralized information interaction for salient object detection. *arXiv preprint arXiv:2012.11294*, 2020. 5

[35] Nian Liu and Junwei Han. DHSNet: Deep hierarchical saliency network for salient object detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 678–686, 2016. 2, 5

[36] Nian Liu, Junwei Han, and Ming-Hsuan Yang. PiCANet: Learning pixel-wise contextual attention for saliency detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3089–3098, 2018. 2, 3, 6, 7

[37] Yun Liu, Yu-Chao Gu, Xin-Yu Zhang, Weiwei Wang, and Ming-Ming Cheng. Lightweight salient object detection via hierarchical visual perception learning. *IEEE TCYB*, 2020. 7

[38] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3431–3440, 2015. 2, 3, 5

[39] Zhiming Luo, Akshaya Kumar Mishra, Andrew Achkar, Justin A Eichel, Shaozi Li, and Pierre-Marc Jodoin. Non-local deep features for salient object detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6609–6617, 2017. 1, 6

[40] NVIDIA, Péter Vingelmann, and Frank H.P. Fitzek. Cuda, release: 10.2.89, 2020. 1

[41] Youwei Pang, Xiaoqi Zhao, Lihe Zhang, and Huchuan Lu. Multi-scale interactive network for salient object detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 9413–9422, 2020. 1, 2, 3, 4, 5, 6

[42] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *ICML*, pages 4095–4104, 2018. 3

[43] Xuebin Qin, Zichen Zhang, Chenyang Huang, Masood Dehghan, Osmar R Zaiane, and Martin Jagersand. U2-net: Going deeper with nested u-structure for salient object detection. *Pattern Recognition*, 106:107404, 2020. 1, 4, 6

[44] Xuebin Qin, Zichen Zhang, Chenyang Huang, Chao Gao, Masood Dehghan, and Martin Jagersand. BASNet: Boundary-aware salient object detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 7479–7489, 2019. 1, 6, 7

[45] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *AAAI*, volume 33, pages 4780–4789, 2019. 3

[46] Tonmoy Saikia, Yassine Marrakchi, Arber Zela, Frank Hutter, and Thomas Brox. Autodispnet: Improving disparity estimation with automl. In *Int. Conf. Comput. Vis.*, October 2019. 3

[47] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, June 2018. 3, 4, 8

[48] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Int. Conf. Learn. Represent.*, 2015. 1, 2, 3

[49] Benoit Steiner, Zachary DeVito, Soumith Chintala, Sam Gross, Adam Paszke, Francisco Massa, Adam Lerer, Gregory Chanan, Zeming Lin, Edward Yang, et al. Pytorch: An imperative style, high-performance deep learning library. *Adv. Neural Inform. Process. Syst.*, 32:8026–8037, 2019. 3, 6, 7

[50] 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 *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2820–2828, 2019. 1, 3

[51] Alvin Wan, Xiaoliang Dai, Peizhao Zhang, Zijian He, Yuandong Tian, Saining Xie, Bichen Wu, Matthew Yu, Tao Xu, Kan Chen, et al. Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 12965–12974, 2020. 3

[52] Jingdong Wang, Huaizu Jiang, Zejian Yuan, Ming-Ming Cheng, Xiaowei Hu, and Nanning Zheng. Salient object detection: A discriminative regional feature integration approach. *Int. J. Comput. Vis.*, 123(2):251–268, 2017. 2

[53] Lijun Wang, Huchuan Lu, Yifan Wang, Mengyang Feng, Dong Wang, Baocai Yin, and Xiang Ruan. Learning to detect salient objects with image-level supervision. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 136–145, 2017. 7

[54] Linzhao Wang, Lijun Wang, Huchuan Lu, Pingping Zhang, and Xiang Ruan. Saliency detection with recurrent fully convolutional networks. In *Eur. Conf. Comput. Vis.*, pages 825–841, 2016. 2

[55] Tiantian Wang, Lihe Zhang, Shuo Wang, Huchuan Lu, Gang Yang, Xiang Ruan, and Ali Borji. Detect globally, refine locally: A novel approach to saliency detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3127–3135, 2018. 2, 5

[56] Wenguan Wang, Qiuxia Lai, Huazhu Fu, Jianbing Shen, Haibin Ling, and Ruigang Yang. Salient object detection

in the deep learning era: An in-depth survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2021. 1, 2, 3

[57] Wenguan Wang, Jianbing Shen, Ming-Ming Cheng, and Ling Shao. An iterative and cooperative top-down and bottom-up inference network for salient object detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5968–5977, 2019. 2

[58] 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 *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 10734–10742, 2019. 3, 4

[59] Runmin Wu, Mengyang Feng, Wenlong Guan, Dong Wang, Huchuan Lu, and Errui Ding. A mutual learning method for salient object detection with intertwined multi-supervision. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 8150–8159, 2019. 2

[60] Yu-Huan Wu, Yun Liu, Jun Xu, Jia-Wang Bian, Yuchao Gu, and Ming-Ming Cheng. Mobilesal: Extremely efficient rgb-d salient object detection. *arXiv preprint arXiv:2012.13095*, 2020. 2

[61] Zhe Wu, Li Su, and Qingming Huang. Cascaded partial decoder for fast and accurate salient object detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3907–3916, 2019. 1, 2, 5, 6

[62] Lingxi Xie and Alan Yuille. Genetic cnn. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1379–1388, 2017. 3

[63] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1492–1500, 2017. 4

[64] Yingyue Xu, Dan Xu, Xiaopeng Hong, Wanli Ouyang, Rongrong Ji, Min Xu, and Guoying Zhao. Structured modeling of joint deep feature and prediction refinement for salient object detection. In *Int. Conf. Comput. Vis.*, pages 3789–3798, 2019. 2, 5

[65] Qiong Yan, Li Xu, Jianping Shi, and Jiaya Jia. Hierarchical saliency detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1155–1162, 2013. 7

[66] Chuan Yang, Lihe Zhang, Huchuan Lu, Xiang Ruan, and Ming-Hsuan Yang. Saliency detection via graph-based manifold ranking. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3166–3173, 2013. 7

[67] Jiahui Yu and Thomas S Huang. Universally slimmable networks and improved training techniques. In *Int. Conf. Comput. Vis.*, pages 1803–1811, 2019. 7

[68] Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Thomas Huang, Xiaodan Song, Ruoming Pang, and Quoc Le. Bignas: Scaling up neural architecture search with big single-stage models. In *Eur. Conf. Comput. Vis.*, pages 702–717, 2020. 2, 3, 6

[69] Qihang Yu, Dong Yang, Holger Roth, Yutong Bai, Yixiao Zhang, Alan L Yuille, and Daguang Xu. C2fnas: Coarse-to-fine neural architecture search for 3d medical image segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4126–4135, 2020. 3

[70] Lu Zhang, Ju Dai, Huchuan Lu, You He, and Gang Wang. A bi-directional message passing model for salient object detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1741–1750, 2018. 2

[71] Pingping Zhang, Dong Wang, Huchuan Lu, Hongyu Wang, and Xiang Ruan. Amulet: Aggregating multi-level convolutional features for salient object detection. In *Int. Conf. Comput. Vis.*, pages 202–211, 2017. 2, 3, 4, 8

[72] Wenqiang Zhang, Jiemin Fang, Xinggang Wang, and Wenyu Liu. Efficientpose: Efficient human pose estimation with neural architecture search. *Computational Visual Media*, pages 1–13, 2021. 3

[73] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6848–6856, 2018. 4

[74] Jia-Xing Zhao, Jiangjiang Liu, Den-Ping Fan, Yang Cao, Jufeng Yang, and Ming-Ming Cheng. EGNet: Edge guidance network for salient object detection. In *Int. Conf. Comput. Vis.*, pages 8779–8788, 2019. 1, 2, 6

[75] Rui Zhao, Wanli Ouyang, Hongsheng Li, and Xiaogang Wang. Saliency detection by multi-context deep learning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1265–1274, 2015. 2

[76] Xiaoqi Zhao, Youwei Pang, Lihe Zhang, Huchuan Lu, and Lei Zhang. Suppress and balance: A simple gated network for salient object detection. In *Eur. Conf. Comput. Vis.*, pages 35–51, 2020. 2, 5

[77] Huajun Zhou, Xiaohua Xie, Jian-Huang Lai, Zixuan Chen, and Lingxiao Yang. Interactive two-stream decoder for accurate and fast saliency detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, June 2020. 1, 2, 4, 6

[78] Tao Zhou, Deng-Ping Fan, Ming-Ming Cheng, Jianbing Shen, and Ling Shao. Rgb-d salient object detection: A survey. *Computational Visual Media*, 7(1):37–69, 2021. 2

[79] Wangjiang Zhu, Shuang Liang, Yichen Wei, and Jian Sun. Saliency optimization from robust background detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2814–2821, 2014. 2

[80] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *Int. Conf. Learn. Represent.*, 2017. 3

[81] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 8697–8710, 2018. 3