用于设备感知型显著性物体检测任务的整体性神经架构搜索*

Yu-Chao Gu[1] Shang-Hua Gao[1] Xusheng Cao[1] Shao-Ping Lu[1] Peng Du[2]
[1]TNList, CS, Nankai University [2]Huawei Technologies
https://mmcheng.net/inas/

Abstract

现有的显著性物体检测（SOD）模型通常着重于骨干特征提取器或显著性头，而忽略了它们之间的关系。一个强大的骨干网络仍然可以在显著性头较弱的情况下实现次优性能，反之亦然。此外，模型性能与推断延迟之间的平衡对模型设计提出了巨大的挑战，特别是在考虑到不同的部署方案时。考虑到整体性神经体系结构搜索（iNAS）空间中的所有组件，我们提出了一种灵活的设备感知型搜索方案，该方案可以仅训练一次 SOD 模型并迅速在多个设备上找到高性能低延迟的模型。

我们提出了带有延迟分组采样（LGS）的进化搜索，来探索扩大后的搜索空间的整个延迟时间区域。通过 iNAS 搜索到的模型与 SOTA 方法具有相似的性能，但分别在 Huawei Nova6 SE，Intel Core CPU，the Jeston Nano 和 Nvidia Titan Xp 减少了 3.8×，3.3×，2.6×，1.9× 倍的延迟。代码即将开源。

1. 简介

显著性物体检测（SOD）旨在分割出图像中最吸引人的物体 [3, 62]. 作为预处理步骤，许多下游应用程序都需要 SOD，例如图片编辑 [9]，图像检索 [22, 6]，视觉追踪 [24]，图片质量评估 [65]，和视频对象分割 [19]。这些应用程序通常要求以低推理延迟将 SOD 模型部署在多个设备上，例如 GPU，CPU，移动电话和嵌入式设备。每个设备都有独特的属性。例如，GPU 擅长进行大规模并行计算 [43]，而嵌入式设备则以低计算预算 [27]为代价实现了能源友好型。因此，不同的部署方案需要完全不同的 SOD 模型设计。

最新的（SOTA）SOD 方法大多手工设计显著性头 [85, 44, 82, 36, 47] 来聚合来自预训练骨干网（例如 VGG [51]，ResNet [23]）的多级特征。然而，过高的推理延迟通常会限制将它们应用于 GPU 以外的其他设备。由于表达能力的降低，为资源受限的情况而设计的手工低延迟 SOD 模型 [46, 17]会遭受较大的性能下降。模型性能和推理延迟之间的难题导致繁重的工作量，需要手动为不同设备设计 SOD 模型。因此，我们旨在提供一种设备感知型的搜索方案，以便在多个设备上快速找到合适的低延迟 SOD 模型。

在不同设备上实现低延迟 SOD 模型存在多个挑战。首先，由于不同的并行计算能力，IO 瓶颈和实现方式，各种操作的相对延迟在不同的设备之间有所不同。将为一台设备设计的 SOD 模型转移到另一台设备会导致次佳的延迟和性能。其次，传统的手工 SOD 模型要么设计功能更强大的显著头 [44, 85, 36, 47] ，要么设计更高效的骨干网 [46, 17]，而忽略了它们之间的关系。我们观察到，功能强大的骨干网在弱显著头的情况下实现了次优效率，反之亦然。这些障碍使学界无法通过手工或 NAS 方案设计设备感知的低延迟 SOD 模型。

为了解决这些问题，我们提出了一种具有整体搜索空间的设备感知搜索方案，一次训练模型并快速在

*本文为中文翻译版。

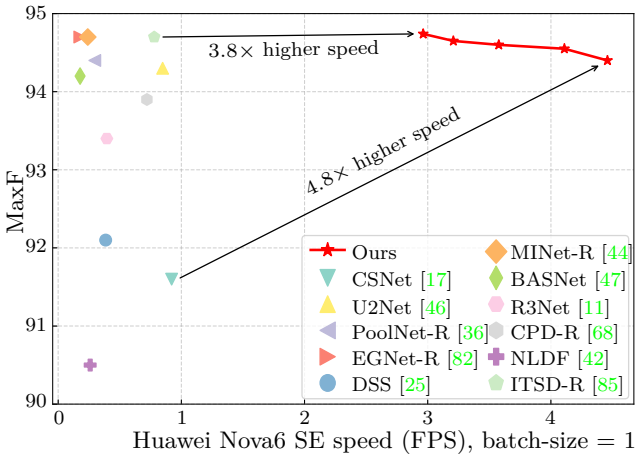


图 1: 我们的 iNAS 与最新的 SOD 模型在移动设备上延迟和性能比较。

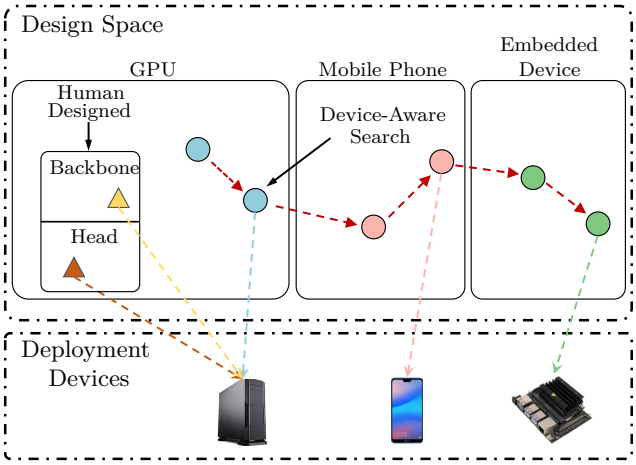


图 2: iNAS 与以前的手工 SOD 方法之间的比较。iNAS 将骨干和头部设计统一到一个完整的设计空间中，并将低延迟的 SOD 模型指定给不同的设备。

多个设备上找到高性能低延迟的 SOD 模型。具体来说，我们为整体考虑骨干网和显著头的 SOD 模型提出了一个完整的搜索空间。为了满足 SOD 模型的多尺度要求，同时又避免了多分支结构增加的延迟，我们构建了一个可搜索的多尺度单位（SMSU）。SMSU 支持具有不同内核大小的可搜索并行卷积，并将搜索到的多分支卷积重新参数化为一个分支，以降低推理延迟。我们还将手工的显著头 [41, 44, 79, 36, 25] 概括为可搜索的传输和解码器部分，从而为与主干空间协作提供了丰富的显著头搜索空间。

我们提出的整体 SOD 搜索空间包含将近 10^{26} 的体系结构，比用于分类任务的网络空间要大 10^7 倍 [4]。先前使用分层均匀采样（LWUS）进行的演化搜索 [21, 4, 76]逐层均匀采样组件。因此，整个采样模型的累积等待时间服从多项式分布，也就是说，极低等待时间或极高等待时间的区域被欠采样，而中间等待时间的区域被过度采样。这个不平衡采样问题阻止了 LWUS 探索扩大后的搜索空间的整个延迟区域。为了克服这种不平衡采样问题，我们提出了一个延迟组采样（LGS），它引入了设备延迟来指导采样。将搜索空间分为几个潜伏时间组，LGS 会将有潜力的后代保留在欠采样区域中，但控制过采样区域的样本。与 LWUS 相比，LGS 的演化搜索可以探索整个积分搜索空间，并在更高和更广的帕累托边界上找到一组模型。

本文的主要贡献是：

- 一个完整的 SOD 搜索空间，该空间考虑了主干-显著头关系，并涵盖了现有的 SOTA 手工 SOD 设计。
- 带有延迟组采样的设备感知进化搜索，用于探索整体搜索空间的整个延迟区域。
- 在五个流行的 SOD 数据集上对 iNAS 进行了全面评估。我们的方法可以达到与手工 SOTA 方法类似的性能，但是可以大大减少不同设备上的推理延迟，这有助于将 SOD 的应用程序扩展到不同的部署方案。

2. 相关工作

2.1. 显著性物体检测

传统的 SOD 方法 [16, 26, 59, 74, 56, 7, 86] 主要依靠手工特征和启发式先验。[83, 29, 30] 最早尝试使用卷积神经网络（CNN）提取块?? 级特征。受 FCN[41]的启发，最近的 SOD 方法 [58, 80, 28, 60, 39, 63] 将 SOD 公式化为像素级的预测任务，与传统方法或基于 CNN 的方法相比，取得了很大的进步。我们向读者介绍综合调查 [3, 62]。大多数 SOD 方法都是通过手工制作显著头来有效融合预先训练的主干（例如 ResNet[23]和 VGG[51]）提取的多级特征的多尺度信息。

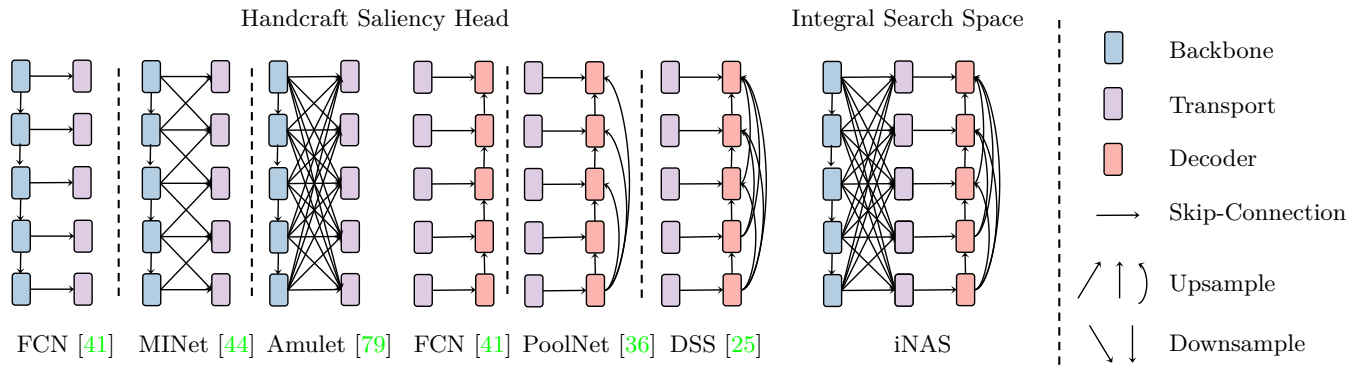


图 3: 最近的手工 SOD 模型的设计和我们提出的整体搜索空间。

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	1-5	1	3,5,7	2-5
1	MBconv1	128x128-192x192	16-24	1-2	3						
2	MBconv6	128x128-192x192	24-32	2-3	3	2	3,5,7	1-5	2	3,5,7	2-4
3	MBconv6	64x64-96x96	32-48	2-3	3,5,7,9	3	3,5,7	1-5	3	3,5,7	2-3
4	MBconv6	32x32-48x48	64-88	2-4	3,5,7,9	4	3,5,7	1-5	4	3,5,7	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	1-5	5	3,5,7	1
7	MBconv6	16x16-24x24	320-352	1-2	3,5,7,9						

表 1: 我们提出的整体搜索空间的详细配置。

这些方法 [38, 25, 5, 61, 36, 72] 继承了编码器-解码器结构，其中解码器负责自下而上的特征融合。转运层 [79, 78, 15, 64, 44, 84] 被包含在显著头内，从而得到自底向上和自顶向下的特征融合。将边缘线索引入显著头以进行精确边界细化的方法 [31, 82, 53, 69, 67]与我们的搜索空间正交。逐渐复杂的 SOD 模型会稳定地提高性能，同时增加巨大的推理延迟。

最近的工作 [68, 85, 17]尝试设计轻量级模型以消除较大的推理延迟。其中，CPD [68] 和 ITSD [85] 设计了轻量级的显著头，分别在 CPU 和 GPU 上实现加速。CSNet [17]设计了轻量型 SOD 骨干以在手机和嵌入式设备上实现低延迟。但是，当硬件特性完全不同时，将设计设备和部署设备分开会导致次优延迟。

在这项工作中，我们引入了一个整体的搜索空间，其中涵盖了大多数手工 SOD 设计。基于我们的整体搜索空间，我们提出了一种设备感知的搜索方案，该方案可以实现与 SOTA 方法相似的性能，但是可以大大减少不同设备上的延迟。

2.2. 神经架构搜索

神经架构搜索（NAS）展示了其为各种任务自动设计高效网络的潜力 [34, 49, 77, 18, 33]。基于早期强化学习 [87, 88] 和基于进化算法 [70, 48]的 NAS 方法训练了数千种候选架构来学习元控制器，花费数百天的 GPU 搜索时间。

后来，可微分的 NAS[35, 20]和 one-shot NAS[21, 4, 76]利用权重共享的思想来降低搜索成本，其中，one-shot NAS 将超网训练与架构搜索分离开来。

大多数 one-shot NAS[21, 4, 76]的目标都是改善超网培训，但只是采用带有分层统一采样（LWUS）的演化搜索。但是，我们发现 LWUS 会导致不平衡采样问题，其中大多数采样位于搜索空间的中间延迟部分。

除了搜索方法外，搜索空间在 NAS 中也起着至关重要的作用。早期方法 [70, 48, 35, 45] 利用基于单元（cell）的搜索空间，其中单元由多个可搜索操作组成。基于单元的搜索空间，Auto-deeplab[34]还支持搜索比例转换的宏结构。为了适应分段任务，Auto-deeplab 合并了固定并行 ASPP 解码器。但是，基于单元的搜索空间的搜索结构具有复杂的分支连接，这在当前的深度学习框架 [52, 1]中很难并行化，从而限制了其在低延迟应用程序中的潜力。

利用人类专业知识，MnasNet[54]和以下工作 [10, 55, 66] 开发了一个基于 MobileNet[50] 的搜索空间，该搜索空间比基于单元的搜索空间支持更多的硬件友好体系结构。

但是，由于这些方法是为分类任务而设计的，它具有较少的多尺度表示能力，不能直接应用于 SOD。

两种设计原则使我们提出的 iNAS 与 Auto-deeplab 和 MnasNet 有所不同：1) 对所有组件的整体搜索减少了总体推理延迟；2) 可搜索的多尺度单元支持搜索多分支结构，而无需额外的推理等待时间。为了全面探索提出的整体搜索空间，我们提出了等待时间组采样以解决以前的 one-shot NAS[21, 4, 76]的不平衡采样问题。

3. 方法

3.1. 整体 SOD 设计空间

先前的手工 SOD 模型 [28, 39, 3, 62]主要基于固定的预训练主干（例如 VGG [51] 和 ResNet [23]）和设计显著头，以融合来自主干的多级特征。最近的一些工作 [17]已经注意到，经过预训练的骨干网占据了大部分延迟成本。他们没有采用笨重的主干，而是为 SOD 设计了轻型主干。

但是，两种设计策略都将骨干网和解码器设计分开，这阻碍了在整体设计空间中找到低延迟高性能 SOD 模型。

本节介绍了整体的 SOD 设计空间，由第3.1.1节中的基本搜索单元（即可搜索的多尺度单位）和第3.1.2节中的可搜索的显著头组成。

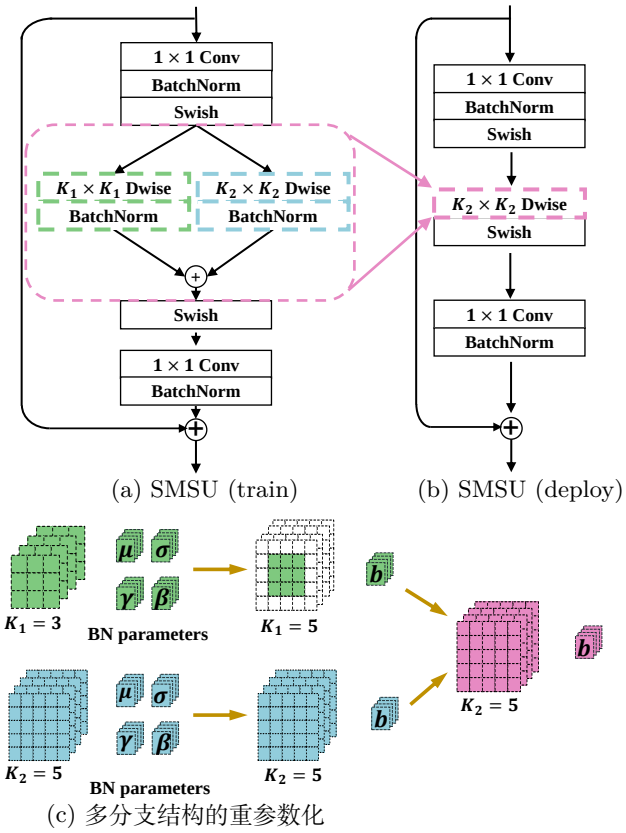


图 4: 可搜索的多尺度单位（SMSU）

3.1.1 可搜索的多尺度单位

由于以前的通用骨干网占据了大部分延迟成本，因此，最新设计的 SOD 骨干干网 [17, 46] 通过组卷积 [71] 或可分卷积 [50] 代替了普通卷积以减少延迟。为了捕获图像中的多尺度表示，他们设计了多个分支以对具有不同接收场的特征进行编码，并融合多尺度特征。但是，多分支结构对硬件不友好 [66, 50, 81]，这会降低推理速度。例如，CSNet [17]比 ITSD-R [85]减少了 13.4 倍计算量，但仅在 GPU 上实现了类似的推理延迟。因此，我们提出了可搜索的多尺度单位（SMSU），可以自动查找合适的多尺度融合。

SMSU 使多分支结构能够在训练中捕获多尺度特征表示，并采用重新参数化策略 [12, 13] 将多个分支融合到单个分支中以进行快速推理。我们在图4(a,b) 中显示了 SMSU 的两个分支设定。We show a two-branches setting of SMSU in 图4(a,b). SMSU

可以提取具有不同内核大小的多尺度特征表示。具体来说，假设有 3×3 卷积和 5×5 卷积，我们使用 $W_1 \in \mathcal{R}^{C\times1\times3\times3}$ 和 $W_2 \in \mathcal{R}^{C\times1\times5\times5}$ 指代深度卷积参数。分别使用 $\mu_1, \sigma_1, \gamma_1, \beta_1$ 和 $\mu_2, \sigma_2, \gamma_2, \beta_2$ 指代紧接着 3×3 卷积和 5×5 卷积的 batch norm 参数。给定输入特征 $F_{in} \in \mathcal{R}^{C\times H\times W}$，我们使用 $M = F_{in} * W$ 指代卷积输出特征，其中 $*$ 代表卷积。两个分支的融合可以表示为：

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

其中 i 代表第 i 个通道。Eqn. (1) 描述了 SMSU 中的训练时的多尺度融合。在部署中，我们将转化权重及其以下 BN 参数合并为转化权重和偏差，定义为：

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

其中 V 是合并的转换权重，b 是偏差。然后，我们将小内核零填充（zero-pad）到给定的分支中，以匹配最大内核。最后，我们对这两个分支求平均值，以获得一个单一的转化权重和偏差。

引入的两分支融合可以轻松扩展到任何分支。因此，我们可以在 SMSU 中搜索融合内核组合。我们用 SMSU 代替了 MobileNet 搜索空间的倒置瓶颈，并在表1总结了搜索空间。

3.1.2 可搜索显著头

先前的手工显著头结合了传输器或解码器，以融合骨干网中的多级特征。高层特征提供了突出对象的粗略位置，低层特征提供了恢复边缘和边界所需的详细特征。如图3所示，典型的转运设计 [79, 44]可实现自下而上和自上而下的多层特征融合。

我们可搜索的转运（transport）连接到骨干网的相应分辨率级别。我们最大转运级别的每个级别都可以聚合来自所有五个分辨率级别的特征，例如 Amulet[79]，而我们最小传输级别的每个级别仅保留 identity 分支，例如 FCN[41]。

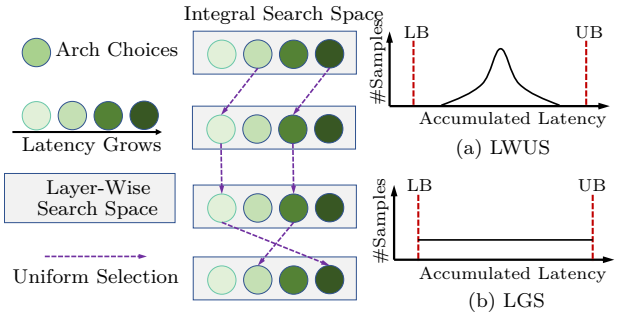


图 5: 分层均匀采样（LWUS）和我们提出的延迟组采样的图示 (LGS)。LB: lower bound. UB: upper bound.

下采样和上采样分支由 1×1 Conv-BN 和 max-pool / bilinear 上采样组成。

考虑到最佳的接收场在不同的分辨率级别上会有所不同，因此我们也支持在转运中搜索融合核。我们可搜索的转运涵盖了许多 SOTA SOD 转运设计 [15, 64, 44, 84, 37]。与转运不同，解码器 [36, 25]仅支持自下而上的预测细化，并逐渐融合低级特征以恢复边界。

因此，我们不支持解码器中的自上而下的融合分支。相邻分辨率级别的标识（identity）分支和上采样分支是固定的，而其他分支则是可搜索的。最大的子解码器具有与 DSS 相似的结构 [25]，而最小的子解码器像 FCN [41]。我们还支持解码器中的可搜索融合内核。可搜索的解码器还涵盖了许多手工的 SOD 解码器设计 [38, 68, 5, 61, 72]。

尽管已证明多尺度融合在 SOD 中是有效的，但是如何修剪冗余融合分支并选择具有延迟约束的合适融合内核对于人类来说是一项劳动密集型的工作。我们提出的显著头使这些关键组件可搜索，并可根据延迟限制自动设计出合适的结构。

3.2. 延迟组采样

先前的 one-shot 方法采用具有逐层均匀采样（LWUS）的进化搜索，这会导致不平衡采样问题。

如图5所示，整个搜索空间由分层搜索空间组成。逐层搜索空间中的组件的延迟有所不同。假设我们逐层均匀地采样组件，则整个采样模型的累积等待时间将服从多项式分布，即极低等待时间或极高等

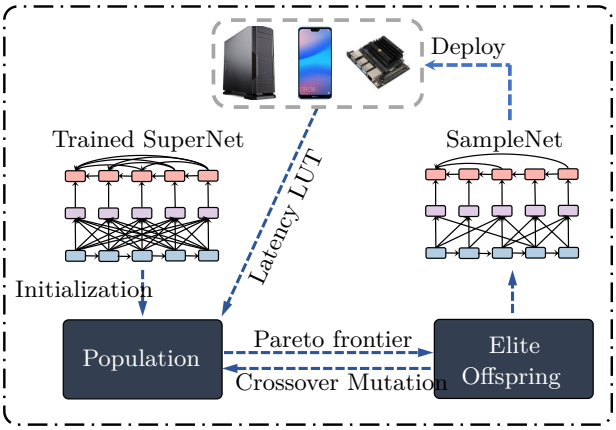


图 6: iNAS 的搜索和部署

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.

Sample the smallest and largest child model (i.e. arch_{min} and arch_{max});
Compute the lower bound and upper bound latency (i.e. LAT_{min} and LAT_{max}) based on LUT;
Divide the $(\text{LAT}_{min}, \text{LAT}_{max})$ into G groups;
Sample $\frac{N}{G}$ child models for each latency group $\{P_i | i = 1 \dots G\}$;
Population $P = P_1 \cup \dots \cup P_G$;
Evaluate performance for models in P;
for $j = 1 \dots iter$ do
    for each P_i do
        $S_i \leftarrow$ Select $\frac{k}{G}$ models from the Pareto frontier of each latency group P_i;
    $S = S_1 \cup \dots \cup S_G$;
    for each model in S do
        Crossover and mutate under probability p_c, p_m.
    Evaluate performance for models in S;
    $P = P \cup S$
$P \leftarrow$ Select Pareto frontier of P;
Return P

待时间的区域采样不足，而中等延迟区域过抽样。为了探索整体搜索空间的整个延迟区域，我们提出了延迟组采样 (LGS)。将搜索空间分为几个延迟组，我们将精英后代保留在欠采样区域中，但控制过采样区域的样本。图6描绘了设备感知演变的一般流程。我们首先建立目标设备的延迟查找表 (LUT)。然后我们基于 LGS 进行进化搜索。搜索后，搜索到的模型将继承超网权重，并且可以直接部署而无需重新训练。LGS 的演化搜索包含四个阶段：

- S1: **初始化**。我们从搜索空间中采样最小和最大子模型，并计算下限和上限延迟。搜索空间分为 G 延迟组。我们在初始种群 P 中对 N 个候选样本进行抽样，其中每个等待时间组都有 $\frac{n}{G}$ 个样本。
- S2: **选拔**。我们从 P 的 Pareto 边界中选择 k 个模型到候选集 S 中，其中每个等待时间组选择 $\frac{k}{G}$ 个样本。
- S3: **交叉**。对于 S 中的每个模型，都有与 p_c 与 S 中的另一个模型交叉的可能性。我们允许交换骨干网中的阶段配置和头部中的交换级别配置。
- S4: **突变**。对于 S 中的每个模型，每个配置都有 p_m 变异的可能性。然后，我们将 S 合并到总体 P 中，并继续进行 S2，直到目标迭代次数 $iter$。

LGS 和 LWUS 之间的主要区别在于初始化和选拔。在初始化步骤中，LGS 在不同等待时间区域中平衡样本，而 LWUS 对中间等待时间区域进行过采样。在选择步骤中，LGS 在欠采样区域中保留了一定数量的精英后代，但是 LWUS 在欠采样区域中放弃了精英后代。我们在第4.2节中比较 LGS 和 LWUS，发现 LWUS 仅探索整个搜索空间的有限延迟区域。

4. 实验

4.1. 实现细节

超网训练细节。我们使用 Pytorch 库实现 iNAS [52]。我们将搜索空间组织为一个嵌套的 one-shot 超网，如 [4, 76]。较小的内核大小是最大内核的中心部分，

Method	FLOPs	Latency (ms)		ECSSD(1000)			DUT-O(5168)			DUTS-TE(5019)			HKU-IS(4447)			PASCAL-S(850)		
	(G)	GPU	Embedded	maxF	MAE	S_m	maxF	MAE	S_m	maxF	MAE	S_m	maxF	MAE	S_m	maxF	MAE	S_m
VGG-16/VGG-19																		
NLDF$_{CVPR17}$ [42]	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$_{CVPR17}$ [25]	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$_{CVPR18}$ [39]	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$_{CVPR19}$ [68]	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$_{CVPR20}$ [85]	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$_{CVPR19}$ [36]	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$_{ICCV19}$ [82]	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$_{CVPR20}$ [44]	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$_{IJCAI18}$ [11]	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$_{CVPR19}$ [68]	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$_{CVPR19}$ [47]	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$_{CVPR19}$ [36]	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$_{ICCV19}$ [82]	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$_{CVPR20}$ [44]	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$_{CVPR20}$ [85]	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$_{ECCV20}$ [17]	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$_{PR20}$ [46]	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

表 2: 与现有 SOD 方法的比较。FLOP 和延迟是用 224×224 输入图像来衡量的。N/A 表示由于内存不足错误，无法将其部署在嵌入式设备上。

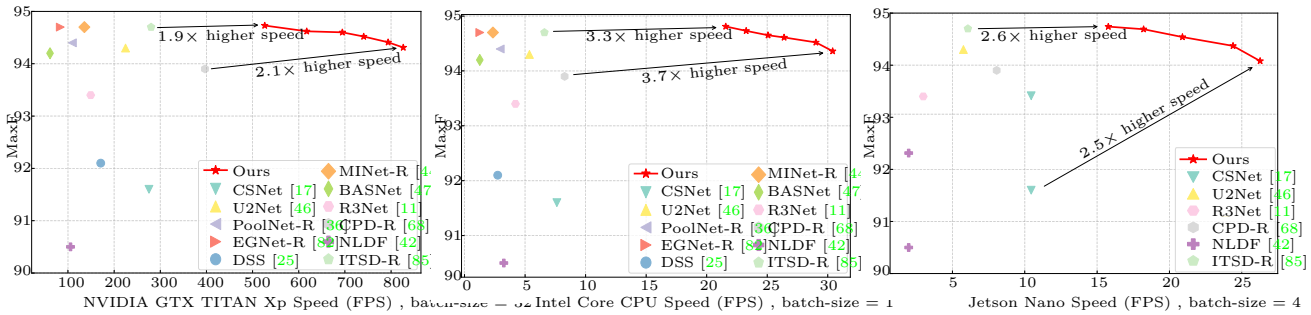
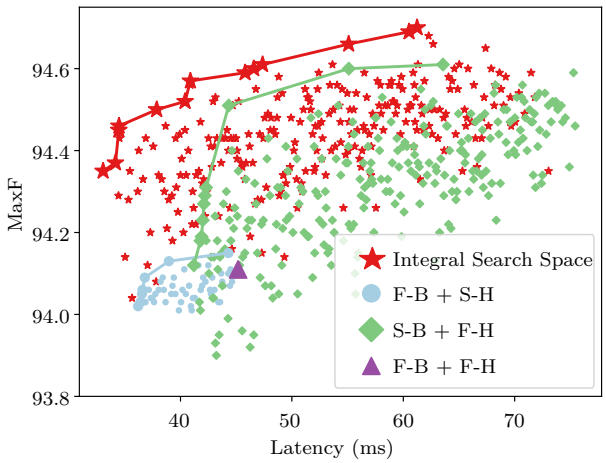


图 7: 与不同设备上的现有 SOD 方法进行速度比较。iNAS 实现了 SOTA 性能和一致的加速。

同时索引较低的通道和层是共享的。使用 ImageNet 预训练，我们在 DUTS-TR 上对 one-shot 超级网络进行了 100 轮的训练。训练 batch size 设为 40。我们使用学习率为 1e-4 和 poly 学习率计划 [40]的 Adam 优化器。我们为每次迭代采样最大，最小和两个中间模型，并融合其梯度以更新超网。最大的子模型使与真值的二元交叉熵最小化，其他模型通过最大的子模型预测使均方误差最小化。遵循 [25]的做法，我们在每个解码器级别的预测上添加了更深的监督。在四台 Tesla V100 上，超网训练耗时 17 个小时。

搜索和部署细节。我们将初始人口规模 (population) N 设置为 1000，并将延迟组 G 设置为 10。进化迭代 $iter$ 设置为 20。每次选拔都保留 $k = 100$ 的后代。交叉和变异概率 (p_c 和 p_m) 设置为 0.2。我们在



(a) 探索搜索空间。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) 整体和部分搜索的定量分析。

图 8: 整数搜索和部分搜索之间的比较。

训练集上微调每个样本模型的 BN 200 次迭代 [75]. 我们使用 Pytorch-Mobile 库 [52] 来构建手机上的延迟表。每个设备在一个 Tesla V100 GPU 上的搜索阶段成本为 0.8 GPU 天。

数据集。超网在 DUTS-TR [57]数据集上训练。The supernet is trained with the DUTS-TR dataset [57]. 我们在五个流行的 SOD 数据集进行了评估，ECSSD [73], DUT-O [74], DUTS-TE [57], HKU-IS [29], PASCAL-S [32]，分别包含 1000，5168，5019，4447，和 850 张图片和对应的显著图。

评估指标。遵循通用设置 x [2, 39, 47]，我们使用 MAE [8]，Max F-measure (F_β) [2] 和 S-measure (S_m) [14] 作为评估我们的结果的评估指标。由于我们旨在设计低延迟 SOD 模型，因此推断延迟也被用作评估指标。

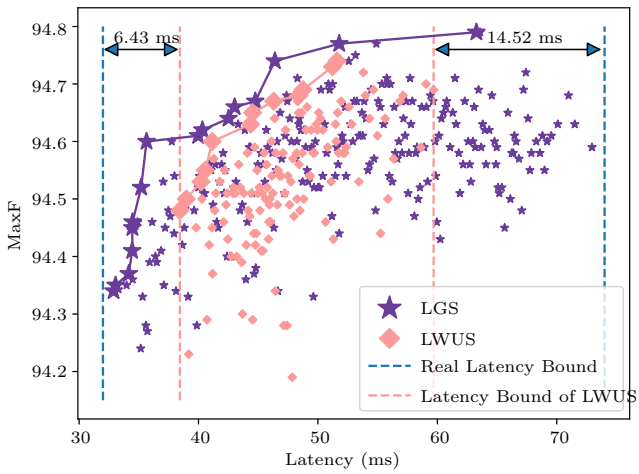


图 9: 将进化搜索与分层均匀采样（LWUS）和我们的 LGS 进行比较。

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%

表 3: 在 GPU 和专用设备上进行搜索的比较。

4.2. 性能评估

与最新方法对比。表2 显示了我们搜索的模型与以前的手工 SOTA SOD 方法之间的比较。其中 iNAS(GPU)-L，GPU 上搜索到的大模型，需要与 CSNet 相似的 FLOP，但是减少了 47%的推理延迟，并在 ECSSD 数据集上提高了的 3.1%的 F_β，这表明 FLOP 与推理延迟没有高度相关关系。我们还在 图1 and 图7展示了在不同设备上我们搜索的模型的延迟比较。我们的方法实现了与 SOTA 相似的性能，但在 GPU，CPU，嵌入式设备和手机上分别减少了 1.9 倍，3.3 倍，2.6 倍，3.8 倍的延迟。与之前最快的方法相比，iNAS 搜索的最快模型在这些设备上的速度提高了 2.1 倍，3.7 倍，2.5 倍和 4.8 倍。

当前的 SOD 模型主要是为 GPU 设计的，而忽略了其他设备。由于内存不足错误，某些基于 ResNet 和基于 VGG 的方法甚至无法应用于嵌入式设备。相比之下，我们的设备感知搜索模型可在所有设备上实现一致的延迟减少。

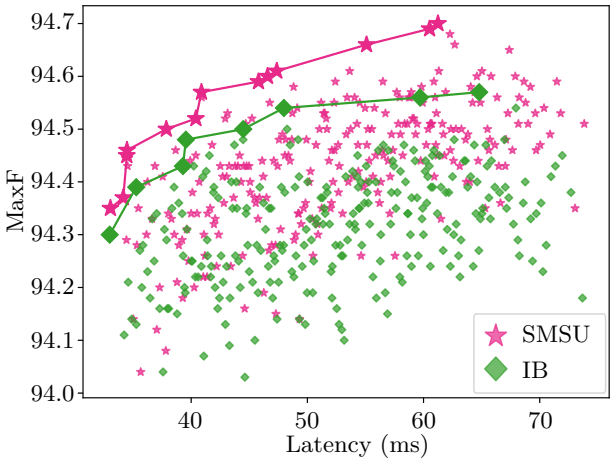


图 10: 比较由反向瓶颈（IB）[50]和我们提出的可搜索多尺度单位（SMSU）构成的搜索空间。

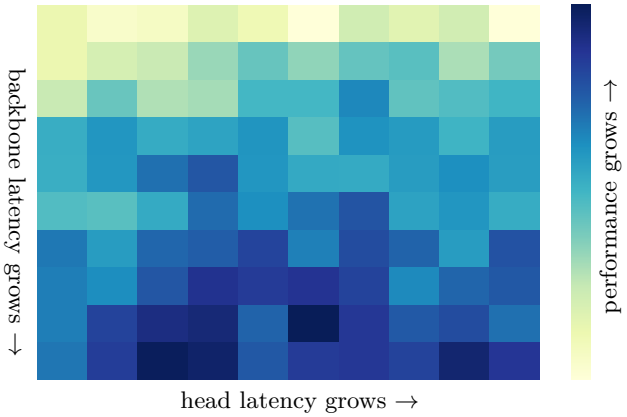


图 11: 骨干/头延迟和性能之间的对应关系的可视化。

设备感知搜索。为了验证设备感知搜索的有效性，我们在表3比较了在 GPU 和其他设备上搜索的模型。我们首先对其他设备上的 iNAS（GPU）-L 的延迟进行基准测试。与 iNAS(GPU)-L 相比，凭借一致的性能，在专用设备上搜索的模型在 CPU，手机和嵌入式设备上的延迟减少了 12.1%，14.5%，10.9%。该观察结果验证了设备感知搜索可以找到目标设备的合适模型以减少等待时间。

整体搜索空间。iNAS 支持用于 SOD 的整体搜索空间。图8 验证了整体搜索空间的重要性。对于基准网络，我们使用 MobileNetV2 结构 [79]作为固定骨干网，并将 Amulet 传输 [79]和 DSS 解码器 [25] 结合起来，形成固定的显著头。如 图8(b) 所示，固定基准网络在 CPU 上需要 45.17 ms 的推理延迟，而在 ECSSD 上则需要 94.1。仅启用可搜索的骨干网或可搜索的显著头，才能在相似性能下将等待时间下限降低到 41.20 ms (-8.7) 或 36.20 ms (-19.8)。使用整体搜索空间时，可将延迟下限降低到 33.06 ms (-26.8)，但将最快架构的性能提高到 94.4。同样，性能上限提高到 94.7。图8(a) 说明整体搜索空间的局部帕累托边界始终优于部分可搜索空间，并且在等待时间和性能方面都显著改善了手工结构。

延迟组采样。图9 比较了基于分层均匀采样（LWUS）和提出的延迟组采样（LGS）的进化搜索。搜索空间的下限和上限延迟分别为 32.1 ms 和 74.1 ms。如 图9所示，LWUS 的下限和上限延迟分别为 38.5 ms 和 59.6 ms，仅占整个搜索空间的 50.2%。而我们建议的 LGS 确保每个潜伏组都有足够的样本和后代，从而可以探索 99%的搜索空间。

最终，我们提出的 LGS 在 LWUS 上获得了更广泛的帕累托边界。

可搜索的多尺度单位。图10 验证了所提出的可搜索多尺度单位（SMSU）的有效性。我们将 SMSU 构造的搜索空间与反向瓶颈（IB）进行了比较。SMSU 构造的搜索空间增强了 IB 的多尺度能力，与 IB 构造的搜索空间相比，它显示出更好的延迟性能帕累托边界。我们观察到较高延迟模型的改进要大得多，我们假设松弛延迟约束可以启用大型内核，而大型内核具有更强大的多尺度内核组合。

4.3. 观察

为了探索性能与骨干延迟和头部延迟之间的关系，我们将骨干和头部延迟分为 10 组，并在每个网格中采样 20 个模型，得出 2000 个样本。观察 图11，我们发现（1）更复杂的主干能够不断提高性能；（2）复杂的显著头并非总是最佳选择。这些观察结果表明，在我们的搜索空间中进行搜索，仍有减少模型延迟的空间。同样，选择合适的显著头以获得更好的延迟性能平衡也没有明显的模式，这表明搜索可能是设计更好的 SOD 模型的有效解决方案。

5. 总结

在这项工作中，我们提出了 SOD 的整体搜索（iNAS）空间，该空间概括了手工 SOD 模型的设计。整体搜索可以自动找到骨干和头部之间的对应关系，并获得最佳的性能-延迟平衡。

然后，我们提出了一个潜伏时间组采样来探索我们的全部整体搜索空间。实验表明，iNAS 具有与手工 SOTA SOD 方法相似的性能，但在很大程度上减少了它们在各种设备中的延迟。我们的工作为低功耗设备上的 SOD 应用铺平了道路。

参考文献

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 4

[2] R. Achanta, S. Hemami, F. Estrada, and S. Susstrunk. Frequency-tuned salient region detection. In CVPR, pages 1597–1604. IEEE, 2009. 8

[3] A. Borji, M.-M. Cheng, Q. Hou, H. Jiang, and J. Li. Salient object detection: A survey. Computational visual media, 5(2):117–150, 2019. 1, 2, 4

[4] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han. Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791, 2019. 2, 3, 4, 6

[5] S. Chen, X. Tan, B. Wang, and X. Hu. Reverse attention for salient object detection. In ECCV, pages 234–250, 2018. 3, 5

[6] M.-M. Cheng, Q.-B. Hou, S.-H. Zhang, and P. L. Rosin. Intelligent visual media processing: When graphics meets vision. Journal of Computer Science and Technology, 32(1):110–121, 2017. 1

[7] M.-M. Cheng, N. J. Mitra, X. Huang, P. H. Torr, and S.-M. Hu. Global contrast based salient region detection. IEEE TPAMI, 37(3):569–582, 2015. 2

[8] M.-M. Cheng, J. Warrell, W.-Y. Lin, S. Zheng, V. Vineet, and N. Crook. Efficient salient region detection with soft image abstraction. In ICCV, pages 1529–1536, 2013. 8

[9] M.-M. Cheng, F.-L. Zhang, N. J. Mitra, X. Huang, and S.-M. Hu. Repfinder: finding approximately repeated scene elements for image editing. ACM Transactions on Graphics (TOG), 29(4):1–8, 2010. 1

[10] X. Chu, B. Zhang, R. Xu, and J. Li. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. arXiv preprint arXiv:1907.01845, 2019. 4

[11] Z. Deng, X. Hu, L. Zhu, X. Xu, J. Qin, G. Han, and P.-A. Heng. R3net: Recurrent residual refinement network for saliency detection. In IJCAI, pages 684–690. AAAI Press, 2018. 2, 7

[12] X. Ding, Y. Guo, G. Ding, and J. Han. Acnet: Strengthening the kernel skeletons for powerful cnn via asymmetric convolution blocks. In ICCV, pages 1911–1920, 2019. 4

[13] X. Ding, X. Zhang, N. Ma, J. Han, G. Ding, and J. Sun. Repvgg: Making vgg-style convnets great again. arXiv preprint arXiv:2101.03697, 2021. 4

[14] D.-P. Fan, M.-M. Cheng, Y. Liu, T. Li, and A. Borji. Structure-measure: A new way to evaluate foreground maps. In ICCV, pages 4548–4557, 2017. 8

[15] M. Feng, H. Lu, and E. Ding. Attentive feedback network for boundary-aware salient object detection. In CVPR, pages 1623–1632, 2019. 3, 5

[16] D. Gao, V. Mahadevan, and N. Vasconcelos. The discriminant center-surround hypothesis for bottom-up saliency. NeurIPS, 20:497–504, 2007. 2

[17] S.-H. Gao, Y.-Q. Tan, M.-M. Cheng, C. Lu, Y. Chen, and S. Yan. Highly efficient salient object detection with 100k parameters. In ECCV, 2020. 1, 2, 3, 4, 7

[18] G. Ghiasi, T.-Y. Lin, and Q. V. Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In CVPR, pages 7036–7045, 2019. 3

[19] Y. Gu, L. Wang, Z. Wang, Y. Liu, M.-M. Cheng, and S.-P. Lu. Pyramid constrained self-attention network for fast video salient object detection. In AAAI, pages 10869–10876, 2020. 1

[20] Y.-C. Gu, Y. Liu, Y. Yang, Y.-H. Wu, S.-P. Lu, and M.-M. Cheng. Dots: Decoupling operation and topology in differentiable architecture search. arXiv preprint arXiv:2010.00969, 2020. 3

[21] Z. Guo, X. Zhang, H. Mu, W. Heng, Z. Liu, Y. Wei, and J. Sun. Single path one-shot neural architecture search with uniform sampling. In European Conference on Computer Vision, pages 544–560. Springer, 2020. 2, 3, 4

[22] J. He, J. Feng, X. Liu, C. Tao, and S. F. Chang. Mobile product search with bag of hash bits and boundary reranking. In CVPR, 2012. 1

[23] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In CVPR, pages 770–778, 2016. 1, 2, 4

[24] S. Hong, T. You, S. Kwak, and B. Han. Online tracking by learning discriminative saliency map with convolutional neural network. In ICML, pages 597–606. PMLR, 2015. 1

[25] Q. Hou, M.-M. Cheng, X. Hu, A. Borji, Z. Tu, and P. Torr. Deeply supervised salient object detection with short connections. IEEE TPAMI, 41(4):815–828, 2019. 2, 3, 5, 7, 9

[26] D. A. Klein and S. Frintrop. Center-surround divergence of feature statistics for salient object detection. In ICCV, pages 2214–2219. IEEE, 2011. 2

[27] A. Kurniawan. Introduction to nvidia jetson nano, 2021. 1

[28] G. Li, Y. Xie, L. Lin, and Y. Yu. Instance-level salient object segmentation. In CVPR, pages 2386–2395, 2017. 2, 4

[29] G. Li and Y. Yu. Visual saliency based on multiscale deep features. In CVPR, pages 5455–5463, 2015. 2, 8

[30] G. Li and Y. Yu. Deep contrast learning for salient object detection. In CVPR, pages 478–487, 2016. 2

[31] X. Li, F. Yang, H. Cheng, W. Liu, and D. Shen. Contour knowledge transfer for salient object detection. In ECCV, pages 355–370, 2018. 3

[32] Y. Li, X. Hou, C. Koch, J. M. Rehg, and A. L. Yuille. The secrets of salient object segmentation. In CVPR, pages 280–287, 2014. 8

[33] Y. Li, X. Jin, J. Mei, X. Lian, L. Yang, C. Xie, Q. Yu, Y. Zhou, S. Bai, and A. L. Yuille. Neural architecture search for lightweight non-local networks. In CVPR, pages 10297–10306, 2020. 3

[34] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. L. Yuille, and L. Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In CVPR, pages 82–92, 2019. 3, 4

[35] H. Liu, K. Simonyan, and Y. Yang. Darts: Differentiable architecture search. arXiv preprint arXiv:1806.09055, 2018. 3, 4

[36] J.-J. Liu, Q. Hou, M.-M. Cheng, J. Feng, and J. Jiang. A simple pooling-based design for real-time salient object detection. In CVPR, pages 3917–3926, 2019. 1, 2, 3, 5, 7

[37] J.-J. Liu, Z.-A. Liu, and M.-M. Cheng. Centralized information interaction for salient object detection. arXiv preprint arXiv:2012.11294, 2020. 5

[38] N. Liu and J. Han. DHSNet: Deep hierarchical saliency network for salient object detection. In CVPR, pages 678–686, 2016. 3, 5

[39] N. Liu, J. Han, and M.-H. Yang. PiCANet: Learning pixel-wise contextual attention for saliency detection. In CVPR, pages 3089–3098, 2018. 2, 4, 7, 8

[40] Y. Liu, Y.-C. Gu, X.-Y. Zhang, W. Wang, and M.-M. Cheng. Lightweight salient object detection via hierarchical visual perception learning. IEEE TCYB, 2020. 7

[41] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, pages 3431–3440, 2015. 2, 3, 5

[42] Z. Luo, A. K. Mishra, A. Achkar, J. A. Eichel, S. Li, and P.-M. Jodoin. Non-local deep features for salient object detection. In CVPR, pages 6609–6617, 2017. 2, 7

[43] NVIDIA, P. Vingelmann, and F. H. Fitzek. Cuda, release: 10.2.89, 2020. 1

[44] Y. Pang, X. Zhao, L. Zhang, and H. Lu. Multi-scale interactive network for salient object detection. In CVPR, pages 9413–9422, 2020. 1, 2, 3, 5, 7

[45] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean. Efficient neural architecture search via parameters sharing. In ICML, pages 4095–4104. PMLR, 2018. 4

[46] X. Qin, Z. Zhang, C. Huang, M. Dehghan, O. R. Zaiane, and M. Jagersand. U2-net: Going deeper with nested u-structure for salient object detection. Pattern Recognition, 106:107404, 2020. 1, 2, 4, 7

[47] X. Qin, Z. Zhang, C. Huang, C. Gao, M. Dehghan, and M. Jagersand. BASNet: Boundary-aware salient

object detection. In CVPR, pages 7479–7489, 2019. 1, 2, 7, 8

[48] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le. Regularized evolution for image classifier architecture search. In AAAI, volume 33, pages 4780–4789, 2019. 3, 4

[49] T. Saikia, Y. Marrakchi, A. Zela, F. Hutter, and T. Brox. Autodispnet: Improving disparity estimation with automl. In ICCV, October 2019. 3

[50] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In CVPR, June 2018. 4, 9

[51] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014. 1, 2, 4

[52] B. Steiner, Z. DeVito, S. Chintala, S. Gross, A. Paszke, F. Massa, A. Lerer, G. Chanan, Z. Lin, E. Yang, et al. Pytorch: An imperative style, high-performance deep learning library. NeurIPS, 32:8026–8037, 2019. 4, 6, 8

[53] J. Su, J. Li, Y. Zhang, C. Xia, and Y. Tian. Selectivity or invariance: Boundary-aware salient object detection. In ICCV, pages 3799–3808, 2019. 3

[54] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le. Mnasnet: Platform-aware neural architecture search for mobile. In CVPR, pages 2820–2828, 2019. 4

[55] A. Wan, X. Dai, P. Zhang, Z. He, Y. Tian, S. Xie, B. Wu, M. Yu, T. Xu, K. Chen, et al. Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions. In CVPR, pages 12965–12974, 2020. 4

[56] J. Wang, H. Jiang, Z. Yuan, M.-M. Cheng, X. Hu, and N. Zheng. Salient object detection: A discriminative regional feature integration approach. IJCV, 123(2):251–268, 2017. 2

[57] L. Wang, H. Lu, Y. Wang, M. Feng, D. Wang, B. Yin, and X. Ruan. Learning to detect salient objects with image-level supervision. In CVPR, pages 136–145, 2017. 8

[58] L. Wang, L. Wang, H. Lu, P. Zhang, and X. Ruan. Saliency detection with recurrent fully convolutional networks. In ECCV, pages 825–841. Springer, 2016. 2

[59] Q. Wang, Y. Yuan, and P. Yan. Visual saliency by selective contrast. IEEE TCSVT, 23(7):1150–1155, 2012. 2

[60] T. Wang, A. Borji, L. Zhang, P. Zhang, and H. Lu. A stagewise refinement model for detecting salient objects in images. In ICCV, pages 4019–4028, 2017. 2

[61] T. Wang, L. Zhang, S. Wang, H. Lu, G. Yang, X. Ruan, and A. Borji. Detect globally, refine locally: A novel approach to saliency detection. In CVPR, pages 3127–3135, 2018. 3, 5

[62] W. Wang, Q. Lai, H. Fu, J. Shen, H. Ling, and R. Yang. Salient object detection in the deep learning era: An in-depth survey. IEEE TPAMI, 2021. 1, 2, 4

[63] W. Wang, J. Shen, M.-M. Cheng, and L. Shao. An iterative and cooperative top-down and bottom-up inference network for salient object detection. In CVPR, pages 5968–5977, 2019. 2

[64] W. Wang, S. Zhao, J. Shen, S. C. Hoi, and A. Borji. Salient object detection with pyramid attention and salient edges. In CVPR, pages 1448–1457, 2019. 3, 5

[65] X. Wang, X. Liang, B. Yang, and F. W. Li. No-reference synthetic image quality assessment with convolutional neural network and local image saliency. Computational Visual Media, 5(2):193–208, 2019. 1

[66] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In CVPR, pages 10734–10742, 2019. 4

[67] R. Wu, M. Feng, W. Guan, D. Wang, H. Lu, and E. Ding. A mutual learning method for salient object detection with intertwined multi-supervision. In CVPR, pages 8150–8159, 2019. 3

[68] Z. Wu, L. Su, and Q. Huang. Cascaded partial decoder for fast and accurate salient object detection. In CVPR, pages 3907–3916, 2019. 2, 3, 5, 7

[69] Z. Wu, L. Su, and Q. Huang. Stacked cross refinement network for edge-aware salient object detection. In ICCV, pages 7264–7273, 2019. 3

[70] L. Xie and A. Yuille. Genetic cnn. In CVPR, pages 1379–1388, 2017. 3, 4

[71] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In CVPR, pages 1492–1500, 2017. 4

[72] Y. Xu, D. Xu, X. Hong, W. Ouyang, R. Ji, M. Xu, and G. Zhao. Structured modeling of joint deep feature and prediction refinement for salient object detection. In ICCV, pages 3789–3798, 2019. 3, 5

[73] Q. Yan, L. Xu, J. Shi, and J. Jia. Hierarchical saliency detection. In CVPR, pages 1155–1162, 2013. 8

[74] C. Yang, L. Zhang, H. Lu, X. Ruan, and M.-H. Yang. Saliency detection via graph-based manifold ranking. In CVPR, pages 3166–3173, 2013. 2, 8

[75] J. Yu and T. S. Huang. Universally slimmable networks and improved training techniques. In ICCV, pages 1803–1811, 2019. 8

[76] J. Yu, P. Jin, H. Liu, G. Bender, P.-J. Kindermans, M. Tan, T. Huang, X. Song, R. Pang, and Q. Le. Bignas: Scaling up neural architecture search with big single-stage models. In ECCV, pages 702–717. Springer, 2020. 2, 3, 4, 6

[77] Q. Yu, D. Yang, H. Roth, Y. Bai, Y. Zhang, A. L. Yuille, and D. Xu. C2fnas: Coarse-to-fine neural architecture search for 3d medical image segmentation. In CVPR, pages 4126–4135, 2020. 3

[78] L. Zhang, J. Dai, H. Lu, Y. He, and G. Wang. A bi-directional message passing model for salient object detection. In CVPR, pages 1741–1750, 2018. 3

[79] P. Zhang, D. Wang, H. Lu, H. Wang, and X. Ruan. Amulet: Aggregating multi-level convolutional features for salient object detection. In ICCV, pages 202–211, 2017. 2, 3, 5, 9

[80] P. Zhang, D. Wang, H. Lu, H. Wang, and B. Yin. Learning uncertain convolutional features for accurate saliency detection. In ICCV, pages 212–221, 2017. 2

[81] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In CVPR, pages 6848–6856, 2018. 4

[82] J.-X. Zhao, J. Liu, D.-P. Fan, Y. Cao, J. Yang, and M.-M. Cheng. EGNet: Edge guidance network for salient object detection. In ICCV, pages 8779–8788, 2019. 1, 2, 3, 7

[83] R. Zhao, W. Ouyang, H. Li, and X. Wang. Saliency detection by multi-context deep learning. In CVPR, pages 1265–1274, 2015. 2

[84] X. Zhao, Y. Pang, L. Zhang, H. Lu, and L. Zhang. Suppress and balance: A simple gated network for salient object detection. In ECCV, pages 35–51. Springer, 2020. 3, 5

[85] H. Zhou, X. Xie, J.-H. Lai, Z. Chen, and L. Yang. Interactive two-stream decoder for accurate and fast saliency detection. In CVPR, June 2020. 1, 2, 3, 4, 7

[86] W. Zhu, S. Liang, Y. Wei, and J. Sun. Saliency optimization from robust background detection. In CVPR, pages 2814–2821, 2014. 2

[87] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. In ICLR, 2017. 3

[88] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. In CVPR, pages 8697–8710, 2018. 3