状态

- 已开源
- 相关链接

简介

作者提出了一种适用于移动端的NAS方法，首先其评价指标是精度和在手机上的运行速度，而不是FLOPS，其次提出了一种新的`factorized hierarchical search space`，使得搜索更加多样，而不是单纯的堆叠某个搜索出的block。

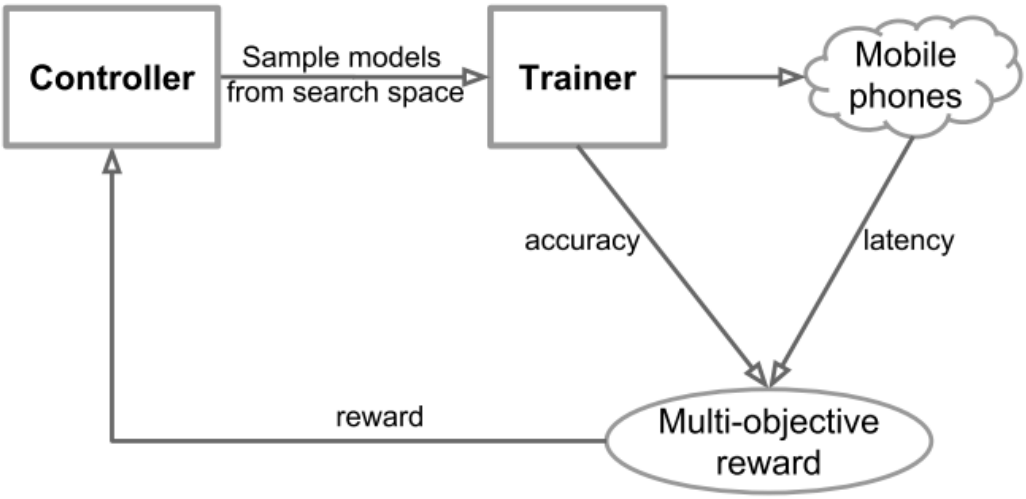


Figure 1: **An Overview of Platform-Aware Neural Architecture Search for Mobile.**

Problem Formulation

在运行速度的限制下获得最高的精度。 那么可以得到如下的限制：

$$\begin{aligned} \underset{m}{\text{maximize}} \quad & ACC(m) \\ \text{subject to} \quad & LAT(m) \leq T \end{aligned} \tag{1}$$

其中T是目标延迟，上面的公式是一种强制的限制，但是这种方式并没有提供Pareto最优解，我们希望优化的过程在精度不降的情况下提高速度，反之亦然。

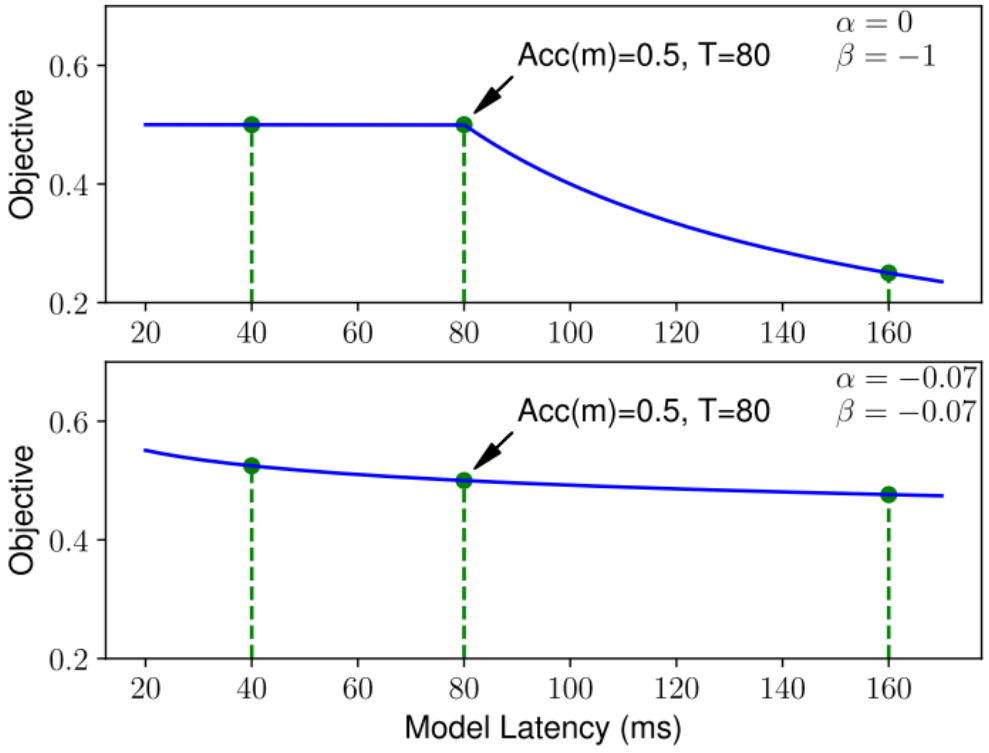
$$\underset{m}{\text{maximize}} \quad ACC(m) \times \left[\frac{LAT(m)}{T}\right]^{w} \tag{2}$$

where w is the weight factor defined as:

$$w = \begin{cases} \alpha, & \text{if } LAT(m) \leq T \\ \beta, & \text{otherwise} \end{cases} \tag{3}$$

首先阿尔法和贝塔都是负数。这两个参数的选取方式可以采取使Pareto优化两个维度时获得相同的reward。

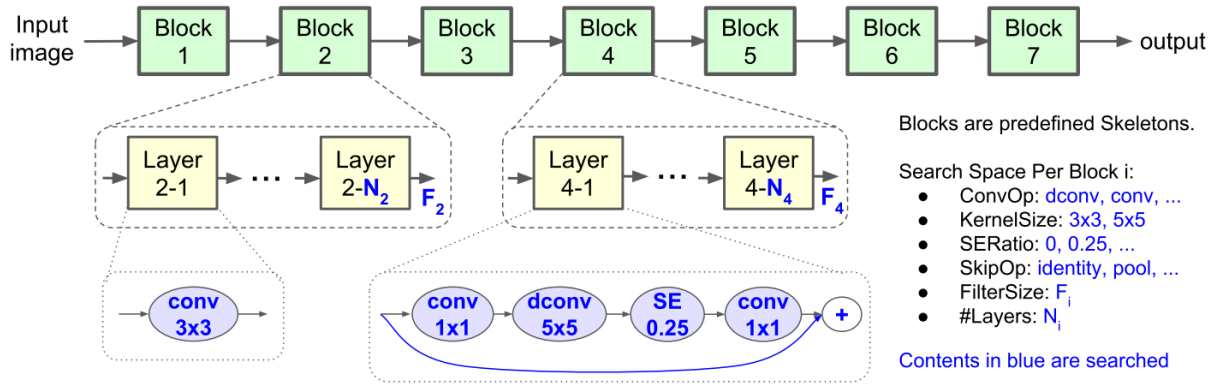
trade-offs. For instance, we empirically observed doubling the latency usually brings about 5% relative accuracy gain. Given two models: (1) M1 has latency l and accuracy a; (2) M2 has latency $2l$ and 5% higher accuracy $a \cdot (1 + 5\%)$, they should have similar reward: $Reward(M2) = a \cdot (1 + 5\%) \cdot (2l/T)^{\beta} \approx Reward(M1) = a \cdot (l/T)^{\beta}$. Solving this gives $\beta \approx -0.07$. Therefore, we use $\alpha = \beta = -0.07$ in our experiments unless explicitly stated.



可以看到hard constraint和soft constraint的区别。

Mobile Neural Architecture Search

Factorized Hierarchical Search Space



搜索时预定义一些基本的结构，然后根据输入输出的shape，按照模块的方式去搜索，需要注意的是为了缩小搜索空间。每个block里的layer，都是一个基本skeleton的重复。

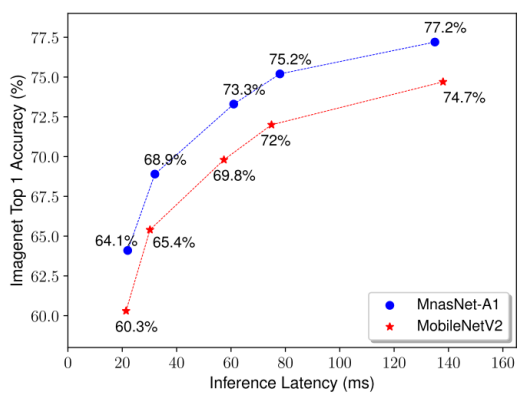
搜索算法

采用强化学习去搜索，目标是最大化期望reward。搜索的框架包含三个部分：

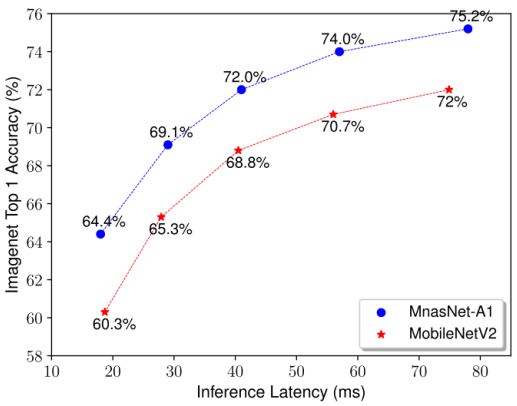
1. RNN based controller
2. trainer to obtain acc
3. mobile device for mearsuring latency

具体的方式是每一步都先sample一个batch的模型，然年通过RNN的softmax logits预测一个序列的token参数。然后得到模型的准确率和推断时间，计算reward，每一次模型参数的更新都是通过最大化期望得到的。

实验和结果分析



(a) Depth multiplier = 0.35, 0.5, 0.75, 1.0, 1.4, corresponding to points from left to right.



(b) Input size = 96, 128, 160, 192, 224, corresponding to points from left to right.

Figure 5: **Performance Comparison with Different Model Scaling Techniques**. MnasNet is our baseline model shown in Table 1. We scale it with the same depth multipliers and input sizes as MobileNetV2.

		Inference Latency	Top-1 Acc.
w/o SE	MobileNetV2	75ms	72.0%
	NASNet	183ms	74.0%
	MnasNet-B1	77ms	74.5%
w/ SE	MnasNet-A1	78ms	75.2%
	MnasNet-A2	84ms	75.6%

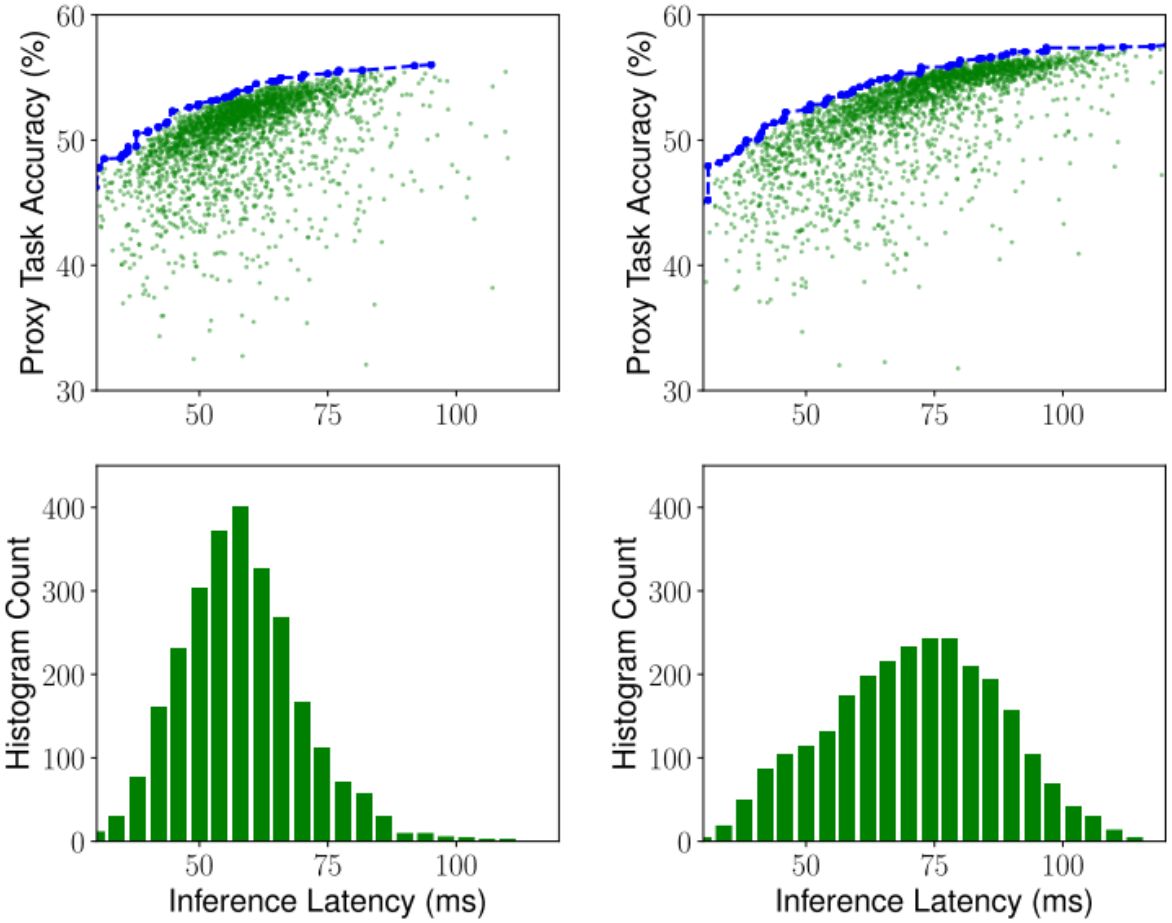
Table 2: **Performance Study for Squeeze-and-Excitation SE [13]** – *MnasNet-A* denote the default MnasNet with SE in search space; *MnasNet-B* denote MnasNet with no SE in search space.

Network	#Params	#Mult-Adds	mAP	mAP_S	mAP_M	mAP_L	Inference Latency
YOLOv2 [27]	50.7M	17.5B	21.6	5.0	22.4	35.5	-
SSD300 [22]	36.1M	35.2B	23.2	5.3	23.2	39.6	-
SSD512 [22]	36.1M	99.5B	26.8	9.0	28.9	41.9	-
MobileNetV1 + SSDLite [11]	5.1M	1.3B	22.2	-	-	-	270ms
MobileNetV2 + SSDLite [29]	4.3M	0.8B	22.1	-	-	-	200ms
MnasNet-A1 + SSDLite	**4.9M**	**0.8B**	**23.0**	**3.8**	**21.7**	**42.0**	**203ms**

Table 3: **Performance Results on COCO Object Detection** – *#Params*: number of trainable parameters; *#Mult-Adds*: number of multiply-additions per image; mAP: standard mean average precision on test-dev2017; mAP_S, mAP_M, mAP_L: mean average precision on small, medium, large objects; *Inference Latency*: the inference latency on Pixel 1 Phone.

	Params	MAdds	Latency	Top1 Acc.
MobileNetV2 (0.35x)	1.66M	59M	21.4ms	60.3%
MnasNet-A1 (0.35x)	1.7M	63M	22.8ms	64.1%
MnasNet-search1	1.9M	65M	22.0ms	64.9%
MnasNet-search2	2.0M	68M	23.2ms	66.0%

Table 4: **Model Scaling vs. Model Search** – *MobileNetV2 (0.35x) and MnasNet-A1 (0.35x)* denote scaling the baseline models with depth multiplier 0.35; *MnasNet-search1/2* denotes models from a new architecture search that targets 22ms latency constraint.



(a) $\alpha = 0, \beta = -1$ (b) $\alpha = \beta = -0.07$

Figure 6: **Multi-Objective Search Results** based on equation 2 with (a) α=0, β=-1; and (b) α=β=-0.07. Target latency is T=$75ms$. Top figure shows the Pareto curve (blue line) for the 3000 sampled models (green dots); bottom figure shows the histogram of model latency.

Reward	Search Space	Latency	Top-1 Acc.
Single-obj [36]	Cell-based [36]	183ms	74.0%
Multi-obj	Cell-based [36]	100ms	72.0%
Multi-obj	**MnasNet**	**78ms**	**75.2%**

Table 5: **Comparison of Decoupled Search Space and Reward Design** – **Multi-obj** denotes our multi-objective reward; **Single-obj** denotes only optimizing accuracy.

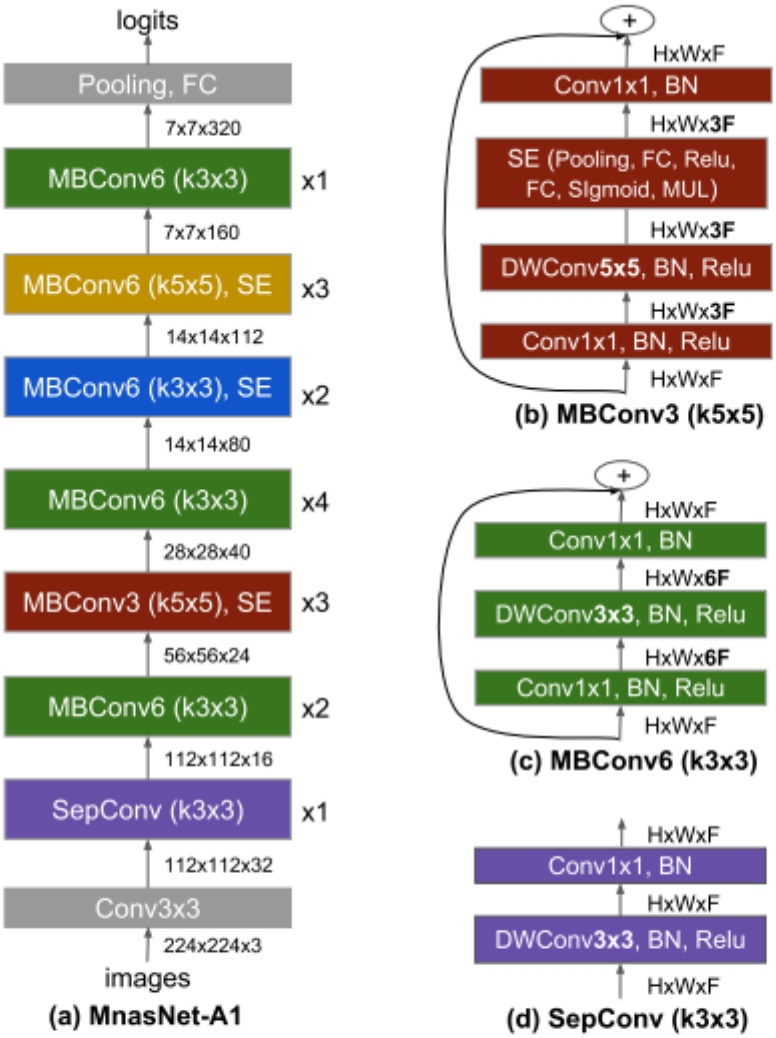


Figure 7: **MnasNet-A1 Architecture** – (a) is a representative model selected from Table 1; (b) - (d) are a few corresponding layer structures. *MBConv* denotes mobile inverted bottleneck conv, *DWConv* denotes depthwise conv, k3x3/k5x5 denotes kernel size, *BN* is batch norm, HxWxF denotes tensor shape (height, width, depth), and $\times 1/2/3/4$ denotes the number of repeated layers within the block.

	Top-1 Acc.	Inference Latency
MnasNet-A1	**75.2%**	**78ms**
MBConv3 (k3x3) only	71.8%	63ms
MBConv3 (k5x5) only	72.5%	79ms
MBConv6 (k3x3) only	74.9%	116ms
MBConv6 (k5x5) only	75.6%	146ms

Table 6: **Performance Comparison of MnasNet and Its Variants** – *MnasNet-A1* denotes the model shown in Figure 7(a); others are variants that repeat a single type of layer throughout the network. All models have the same number of layers and same filter size at each layer.