On-Device Image Classification with Proxyless Neural Architecture Search and Quantization-Aware Fine-Tuning

Han Cai, Tianzhe Wang, Zhanghao Wu, Kuan Wang, Ji Lin, Song Han

Massachusetts Institute of Technology
From Manual Design to Automatic Design

Manual Architecture Design
- VGGNets
- Inception Models
- ResNets
- DenseNets
- ...

Use Human Expertise

Automatic Architecture Search
- Reinforcement Learning
- Neuro-evolution
- Bayesian Optimization
- Monte Carlo Tree Search
- ...

Use Machine Learning

Computational Resources

ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware, ICLR’19
On-Device Image Classification with Proxyless Neural Architecture Search and Quantization-Aware Fine-tuning, ICCV Workshop’2019
From Manual Design to Automatic Design

• Previously, people tend to design a single efficient CNN for all platforms and all datasets.

• But, different platform in fact has different properties, e.g. degree of parallelism, cache size, #PE, memory BW.

• Machine learning wants *generalization*
  Hardware efficiency needs *specialization*
  Build a *generalized* model to handle *specialized* hardware?
From General Design to Specialized CNN

Previous Paradigm:
One CNN for all platforms.

Proxyless NAS:
Customize CNN for each platform.

ResNet
Inception
DenseNet
MobileNet
ShuffleNet

ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware, ICLR’19
On-Device Image Classification with Proxyless Neural Architecture Search and Quantization-Aware Fine-tuning, ICCV Workshop’2019
Conventional NAS: Computation Expensive, thus Proxy-Based

Current neural architecture search (NAS) is **VERY EXPENSIVE**.
- NASNet: 48,000 GPU hours $\approx$ 5 years on single GPU
- DARTS: 100Gb GPU memory* $\approx$ 9 times of modern GPU

*if directly search on ImageNet, like us

Therefore, previous work have to utilize **proxy tasks**:
- CIFAR-10 -> ImageNet
- Small architecture space (e.g. low depth) -> large architecture space
- Fewer epochs training -> full training

ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware, ICLR’19
On-Device Image Classification with Proxyless Neural Architecture Search and Quantization-Aware Fine-tuning, ICCV Workshop’2019
Conventional NAS: Proxy-Based

Limitations of Proxy

- **Suboptimal** for the target task
- Blocks are forced to *share the same structure*.
- Cannot optimize for *specific hardware*.

ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware, ICLR’19
On-Device Image Classification with Proxyless Neural Architecture Search and Quantization-Aware Fine-tuning, ICCV Workshop’2019
**Proxyless, Save GPU Hours by 200x**

**Goal:** Directly learn architectures on the target task and hardware, while allowing all blocks to have different structures. We achieved by

1. Reducing the cost of NAS (GPU hours and memory) to the same level of regular training.
2. Cooperating hardware feedback (e.g. latency) into the search process.

ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware, ICLR’19
On-Device Image Classification with Proxyless Neural Architecture Search and Quantization-Aware Fine-tuning, ICCV Workshop’2019
Model Compression

Pruning

Neural Architecture Search

Binarization

Save GPU hours

Save GPU Memory

ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware, ICLR’19
On-Device Image Classification with Proxyless Neural Architecture Search and Quantization-Aware Fine-tuning, ICCV Workshop’2019
Direct Search on Target Hardware: Making Latency Differentiable

- Mobile farm infrastructure is expensive and slow.
- Use the latency estimation model as an economical alternative
- Optimize during search stage use Gradient.

\[
\mathbb{E}[\text{Latency}] = \alpha \times F(\text{conv} \_3x3) + \\
\beta \times F(\text{conv} \_5x5) + \\
\sigma \times F(\text{identity}) + \\
\cdots \\
\zeta \times F(\text{pool} \_3x3)
\]

\[
\mathbb{E}[\text{latency}] = \sum_i \mathbb{E}[\text{latency}_i]
\]

\[
\text{Loss} = \text{Loss}_{CE} + \lambda_1 ||w||_2^2 + \lambda_2 \mathbb{E}[\text{latency}]
\]
Save GPU Hours

**Pruning** redundant paths based on architecture parameters

Simplify NAS to be a **single training process** of a over-parameterized network.

No meta controller. Stand on the shoulder of giants.

Build the cumbersome network **with all candidate paths**

**ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware, ICLR’19**

**On-Device Image Classification with Proxyless Neural Architecture Search and Quantization-Aware Fine-tuning, ICCV Workshop’2019**
Binarize the architecture parameters and allow only one path of activation to be active in memory at run-time.

We propose gradient-based and RL methods to update the binarized parameters. Thereby, the memory footprint reduces from $O(N)$ to $O(1)$.
ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware, ICLR’19
On-Device Image Classification with Proxyless Neural Architecture Search and Quantization-Aware Fine-tuning, ICCV Workshop’2019
The History of Architectures

(1) The history of finding efficient Mobile model

(2) The history of finding efficient CPU model

(3) The history of finding efficient GPU model

ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware, ICLR'19

On-Device Image Classification with Proxyless Neural Architecture Search and Quantization-Aware Fine-tuning, ICCV Workshop'2019
## Results for LPIRC

<table>
<thead>
<tr>
<th>Model</th>
<th>Setting</th>
<th>Accuracy</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>MoblieNetV2</td>
<td>224-0.5</td>
<td>63.7%(65.4%)</td>
<td>28ms</td>
</tr>
<tr>
<td>MobileNetV2</td>
<td>192-0.75</td>
<td>67.4%(68.7%)</td>
<td>36ms</td>
</tr>
<tr>
<td>MobileNetV2</td>
<td>160-1.0</td>
<td>67.4%(68.8%)</td>
<td>31ms</td>
</tr>
<tr>
<td>ProxylessNAS</td>
<td>224-0.5</td>
<td>65.7%(67.0%)</td>
<td>31ms</td>
</tr>
<tr>
<td>ProxylessNAS</td>
<td>160-1.0</td>
<td><strong>69.2%(70.3%)</strong></td>
<td>35ms</td>
</tr>
</tbody>
</table>

Table 1. Results of 8-bit model using different preprocessing, the number in the bracket denotes the full-precision model’s top-1 accuracy on ImageNet. The latency is directly measured on Google Pixel 2. It takes only 200 GPU hours to find the specialized model with ProxylessNAS in the table.
Open-source

• Both search code and models are released on Github:

```python
# https://github.com/MIT-HAN-LAB/ProxylessNAS
from proxyless_nas import *
net = proxyless_cpu(pretrained=True)
net = proxyless_gpu(pretrained=True)
net = proxyless_mobile(pretrained=True)
```
Open-source

• ProxylessNAS is available on Pytorch Hub:

```python
# https://pytorch.org/hub/pytorch_vision_proxylessnas
import torch
target_platform = 'proxyless_mobile'
net = torch.hub.load('mit-han-lab/ProxylessNAS',
                     target_platform, pretrained=True)
```
Thank you!

songhan@mit.edu