AutoML for Object Detection

Xiangyu Zhang
MEGVII Research
AutoML for Object Detection

1. Advances in AutoML
2. Search for Detection Systems
AutoML for Object Detection

1. Advances in AutoML

2. Search for Detection Systems
Introduction

- **AutoML**
  - A meta-approach to generate machine learning systems
  - Automatically search vs. manually design

- **AutoML for Deep Learning**
  - Neural architecture search (NAS)
  - Hyper-parameters turning
  - Loss function
  - Data augmentation
  - Activation function
  - Backpropagation
  - ...
Revolution of AutoML

- ImageNet 2012 -
  - Hand-craft feature
    vs. deep learning

- Era of Deep Learning begins!

<table>
<thead>
<tr>
<th>Network</th>
<th>Classification Top-5 Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OXFORD</td>
<td>27</td>
</tr>
<tr>
<td>ISI</td>
<td>26.2</td>
</tr>
<tr>
<td>AlexNet</td>
<td>16.4</td>
</tr>
<tr>
<td>SPPNet</td>
<td>8.1</td>
</tr>
<tr>
<td>VGG</td>
<td>7.3</td>
</tr>
<tr>
<td>GoogleNet</td>
<td>6.6</td>
</tr>
<tr>
<td>PReLU</td>
<td>4.9</td>
</tr>
<tr>
<td>ResNet 152</td>
<td>3.57</td>
</tr>
</tbody>
</table>
Revolution of AutoML (cont’d)

- ImageNet 2017 -
  - **Manual architecture**
  - vs. **AutoML models**

<table>
<thead>
<tr>
<th>Model</th>
<th>Classification Top-1 Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNeXt-101</td>
<td>19.1</td>
</tr>
<tr>
<td>SENet</td>
<td>17.3</td>
</tr>
<tr>
<td>NASNet-A</td>
<td>17.3</td>
</tr>
<tr>
<td>PNASNet-5</td>
<td>17.1</td>
</tr>
<tr>
<td>AmoebaNet-A</td>
<td>16.1</td>
</tr>
<tr>
<td>EfficientNet</td>
<td>15.6</td>
</tr>
</tbody>
</table>

Era of AutoML?
Revolution of AutoML (cont’d)

- Literature
  - 200+ since 2017

## LITERATURE ON NEURAL ARCHITECTURE SEARCH

The following list considers papers related to neural architecture search. It is by no means a complete list. If you miss a paper on the list, please let us know.

**Update (Dec 2018):** Since the list is already quite long by now, we will highlight papers accepted at conferences and journals in the future. This should hopefully provide some guidance towards high-quality papers.

- Architecture Search (and Hyperparameter Optimization):
  - Surrogate-Assisted Evolutionary Deep Learning Using an End-to-End Random Forest-based Performance Predictor (Sun et al. 2019; accepted by IEEE Transactions on Evolutionary Computation)
    https://ieeexplore.ieee.org/document/8744404
  - Adaptive Genomic Evolution of Neural Network Topologies (AGENT) for State-to-Action Mapping in Autonomous Agents (Brihaye et al. 2019; accepted and presented in ICRA 2019)
    https://arxiv.org/abs/1903.01107
  - Densely Connected Search Space for More Flexible Neural Architecture Search (Fang et al. 2019)
  - SwiftNet: Using Graph Propagation as Meta-knowledge to Search Highly Representative Neural Architectures (Cheng et al. 2019)
    https://arxiv.org/abs/1906.08305
  - Transfer NAS: Knowledge Transfer between Search Spaces with Transformer Agents (Borsos et al. 2019)
  - XNAS: Neural Architecture Search with Expert Advice (Nayman et al. 2019)
  - A Study of the Learning Progress in Neural Architecture Search Techniques (Singh et al. 2019)
Revolution of AutoML (cont’d)

- Literature
  - 200+ since 2017

- Google Trends
Recent Advances in AutoML (1)

- Surpassing handcraft models
  - NASNet

- Keynotes
  - RNN controller + policy gradient
  - Flexible search space
  - Proxy task needed

Zoph et al. Learning Transferable Architectures for Scalable Image Recognition
Zoph et al. Neural Architecture Search with Reinforcement Learning
Recent Advances in AutoML (2)

- Search on the target task
  - MnasNet

- Keynotes
  - Search directly on ImageNet
  - Platform aware search
  - Very costly (thousands of TPU-days)

Tan et al. MnasNet: Platform-Aware Neural Architecture Search for Mobile
Recent Advances in AutoML (3)

- Weight Sharing for Efficient Search & Evaluation
  - ENAS
  - One-shot methods

- Keynotes
  - Super network
  - Finetuning & inference only instead of retraining
  - Inconsistency in super net evaluation

Pham et al. Efficient Neural Architecture Search via Parameter Sharing
Bender et al. Understanding and Simplifying One-Shot Architecture Search
Guo et al. Single Path One-Shot Neural Architecture Search with Uniform Sampling
Recent Advances in AutoML (4)

- Gradient-based methods
  - DARTS
  - SNAS, FBNet, ProxylessNAS, ...

- Keynotes
  - Joint optimization of architectures and weights
  - Weight sharing implied
  - Sometimes less flexible

Liu et al. DARTS: Differentiable Architecture Search
Xie et al. SNAS: Stochastic Neural Architecture Search
Cai et al. ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware
Wu et al. FBNet: Hardware-Aware Efficient ConvNet Design via Differentiable Neural Architecture Search
Recent Advances in AutoML (5)

- **Performance Predictor**
  - Neural Architecture Optimization
  - ChamNet

- **Keynotes**
  - Architecture encoding
  - Performance prediction models
  - Cold start problem

---

Luo et al. Neural Architecture Optimization
Dai et al. ChamNet: Towards Efficient Network Design through Platform-Aware Model Adaptation
Recent Advances in AutoML (6)

- Hardware-aware Search
  - Search with complexity budget
  - Quantization friendly
  - Energy-aware search
  
  ...  

- Keynotes
  - Complexity-aware loss & reward
  - Multi-target search
  - Device in the loop

Wu et al. Mixed Precision Quantization of ConvNets via Differentiable Neural Architecture Search
Veniat et al. Learning Time/Memory-Efficient Deep Architectures with Budgeted Super Networks
Wang et al. HAQ: Hardware-Aware Automated Quantization with Mixed Precision
Recent Advances in AutoML (7)

- AutoML in Model Pruning
  - NetAdapt
  - AMC
  - MetaPruning

- Keynotes
  - Search for the pruned architecture
  - Hyper-parameters like channels, spatial size, ...

Yang et al. NetAdapt: Platform-Aware Neural Network Adaptation for Mobile Applications
He et al. AMC: AutoML for Model Compression and Acceleration on Mobile Devices
Liu et al. MetaPruning: Meta Learning for Automatic Neural Network Channel Pruning
Recent Advances in AutoML (8)

- Handcraft + NAS
  - Human-expert guided search (IRLAS)
  - Boosting existing handcraft models (EfficientNet, MobileNet v3)

- Keynotes
  - Very competitive performance
  - Efficient
  - Search space may be restricted

Howard et al. Searching for MobileNetV3
Tan et al. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks
Guo et al. IRLAS: Inverse Reinforcement Learning for Architecture Search
Recent Advances in AutoML (9)

- Various Tasks
  - Object Detection
  - Semantic Segmentation
  - Super-resolution
  - Face Recognition
  ...

- Not only NAS, search for everything!
  - Activation function
  - Loss function
  - Data augmentation
  - Backpropagation
  ...

Chu et al. Fast, Accurate and Lightweight Super-Resolution with Neural Architecture Search
Ramachandra et al. Searching for Activation Functions
Alber et al. Backprop Evolution
Recent Advances in AutoML (10)

- Rethinking the Effectiveness of NAS
  - Random search
  - Random wire network

- Keynotes
  - Reproducibility
  - Search algorithm or search space?
  - Baselines

Li et al. Random Search and Reproducibility for Neural Architecture Search
Xie et al. Exploring Randomly Wired Neural Networks for Image Recognition
Summary: Trends and Challenges

- **Trends**
  - Efficient & high-performance algorithm
  - Flexible search space
  - Device-aware optimization
  - Multi-task / Multi-target search

- **Challenges**
  - Trade-offs between efficiency, performance and flexibility
  - Search space matters!
  - Fair benchmarks
  - Pipeline search
AutoML for Object Detection

1. Advances in AutoML

2. Search for Detection Systems
AutoML for Object Detection

- Components to search
  - Image preprocessing
  - Backbone
  - Feature fusion
  - Detection head & loss function
  ...
AutoML for Object Detection

- Components to search
  - Image preprocessing
  - Backbone
  - Feature fusion
  - Detection head & loss function
  ...
AutoML for Object Detection

- Components to search
  - Image preprocessing
  - **Backbone**
  - Feature fusion
  - Detection head & loss function
  ...
AutoML for Object Detection

- Components to search
  - Image preprocessing
  - Backbone
  - Feature fusion
  - Detection head & loss function
  ...

![Diagram of AutoML for Object Detection](image)
AutoML for Object Detection

- Components to search
  - Image preprocessing
  - Backbone
  - Feature fusion
  - Detection head & loss function
  ...
Search for Detection Systems

Backbone

Feature Fusion

Augmentation

DetNAS

Chen et al. DetNAS: Backbone Search for Object Detection
Challenges of Backbone Search

- Similar to general NAS, but …
  - Controller & evaluator loop
  - Performance evaluation is very slow

- Detection backbone evaluation involves a costly pipeline
  - ImageNet pretraining
  - Finetuning on the detection dataset (e.g. COCO)
  - Evaluation on the validation set
Decoupled weight training and architecture optimization

\[ w_a = \arg\min_{w} \mathcal{L}_{\text{train}} (\mathcal{N}(a, w)) , \]

\[ a^* = \arg\max_{a \in \mathcal{A}} \text{ACC}_{\text{val}} (\mathcal{N}(a, w_a)) , \]

\[ W_A = \arg\min_{W} \mathcal{L}_{\text{train}} (\mathcal{N}(\mathcal{A}, W)) , \]

\[ a^* = \arg\max_{a \in \mathcal{A}} \text{ACC}_{\text{val}} (\mathcal{N}(a, W_A(a))) . \]

Super net training

\[ W_A = \arg\min_{W} \mathbb{E}_{a \sim \Gamma(\mathcal{A})} [\mathcal{L}_{\text{train}}(\mathcal{N}(a, W(a)))] , \]

Guo et al. Single Path One-Shot Neural Architecture Search with Uniform Sampling
Pipeline

- Single-pass approach
  - Pretrain and finetune super net only once

Step 1: Supernet pre-training

Step 2: Supernet fine-tuning

Step 3: Evolutionary search on the trained supernet
Search Space

- Single path super net
  - 20 (small) or 40 (large) choice blocks
  - 4 candidates for each choice block
  - Search space size: $4^{20}$ or $4^{40}$
Evolutionary search

- Sample & reuse the weights from super net
- Very efficient

Algorithm 1: Evolutionary Architecture Search

**Input:** supernet weights $W_A$, population size $P$, architecture constraints $C$, max iteration $T$, validation dataset $D_{val}$

**Output:** the architecture with highest validation accuracy under architecture constraints

1. $P_0 := \text{Initialize\_population}(P, C)$;
2. $n := P/2$; \hspace{1cm} # Crossover number
3. $m := P/2$; \hspace{1cm} # Mutation number
4. $\text{prob} := 0.1$; \hspace{1cm} # Probability to mutate
5. Topk := $\emptyset$;
6. for $i = 1 : T$ do
7. $\text{ACC}_{i-1} := \text{Inference}(W_A, D_{val}, P_{i-1})$;
8. Topk := $\text{Update\_Topk}$(Topk, $P_{i-1}$, $\text{ACC}_{i-1}$);
9. $P_{\text{crossover}} := \text{Crossover}(\text{Topk}, n, C)$;
10. $P_{\text{mutation}} := \text{Mutation}(\text{Topk}, m, \text{prob}, C)$;
11. $P_i := P_{\text{crossover}} \cup P_{\text{mutation}}$;
12. end for
13. return the entry with highest accuracy in Topk;
Results

- High performance
  - Significant improvements over commonly used backbones (e.g. ResNet 50) with fewer FLOPs
  - Best classification backbones may be suboptimal for object detection

<table>
<thead>
<tr>
<th>Backbone</th>
<th>ImageNet Classification</th>
<th>Object Detection with FPN on COCO</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FLOPs</td>
<td>Accuracy</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>3.8G</td>
<td>76.15</td>
</tr>
<tr>
<td>ShuffleNetv2-40</td>
<td>1.3G</td>
<td>77.18</td>
</tr>
<tr>
<td>ResNet-101</td>
<td>7.6G</td>
<td>77.37</td>
</tr>
<tr>
<td>DetNASNet</td>
<td>1.3G</td>
<td>77.20</td>
</tr>
<tr>
<td>DetNASNet (3.8)</td>
<td>3.8G</td>
<td>78.44</td>
</tr>
</tbody>
</table>

Table 2: Main result comparisons.

<table>
<thead>
<tr>
<th>Backbone</th>
<th>ImageNet Classification</th>
<th>COCO (mAP, %)</th>
<th>VOC (mAP, %)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FPN</td>
<td>RetinaNet</td>
<td>FPN</td>
</tr>
<tr>
<td>ShuffleNetv2-20</td>
<td>73.1</td>
<td>34.8</td>
<td>80.6</td>
</tr>
<tr>
<td>ClsNASNet</td>
<td>74.3</td>
<td>35.1</td>
<td>78.5</td>
</tr>
<tr>
<td>DetNAS-scratch</td>
<td>73.8 - 74.3</td>
<td>35.9</td>
<td>81.1</td>
</tr>
<tr>
<td>DetNAS</td>
<td>73.9 - 74.1</td>
<td>36.4</td>
<td><strong>81.5</strong></td>
</tr>
</tbody>
</table>

Table 3: Ablation studies.


Results

- Search cost
  - Super nets greatly speed up search progress!

Table 5: Computation cost for each step on COCO.

<table>
<thead>
<tr>
<th>DetNAS</th>
<th>Supernet pre-training</th>
<th>Supernet fine-tuning</th>
<th>Search on the supernet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$3 \times 10^5$ iterations</td>
<td>$9 \times 10^4$ iterations</td>
<td>$20 \times 50$ models</td>
</tr>
<tr>
<td>8 GPUs on 1.5 days</td>
<td>8 GPUs on 1.5 days</td>
<td>20 GPUs on 1 day</td>
<td></td>
</tr>
</tbody>
</table>

* For the small search space, GPUs are GTX 1080Ti. For the large search space, GPUs are Tesla V100.
Search for Detection Systems

Backbone

Feature Fusion
NAS-FPN

Augmentation

Ghaisi et al. NAS-FPN: Learning Scalable Feature Pyramid Architecture for Object Detection
Feature Fusion Modules

- Multi-scale feature fusion
  - Used in state-of-the-art detectors (e.g. SSD, FPN, SNIP, FCOS, …)

- Automatic search vs. manual design
First Glance

- Searched architecture
  - Very different from handcraft structures
Search Space

- Stacking repeated FPN blocks
- For each FPN block, N different merging cells
- For each merging cell, 4-step generations
Search Algorithm

- Controller
  - RNN-based controller
  - Search with Proximal Policy Optimization (PPO)

- Candidate evaluation
  - Training a light-weight proxy task
Architectures During Search

- Many downsamples and upsamples
Results

- State-of-the-art speed/AP trade-off
Search for Detection Systems

Backbone

Feature Fusion

Augmentation

Auto-Augment for Detection

Zoph et al. Learning Data Augmentation Strategies for Object Detection
Data Augmentation for Object Detection

- Augmentation pool
  - Color distortions
  - Geometric transforms
  - Random noise (e.g. cutout, drop block, …)
  - Mix-up
  ...

- Search for the best augmentation configurations
Search Space Design

- Mainly follows AutoAugment
- Randomly sampling from K sub-policies
- For each sub-policy, N image transforms
- Each image transform selected from 22 operations:
  - Color operations
  - Geometric operations
  - Bounding box operations

Cubuk et al. AutoAugment: Learning Augmentation Strategies from Data
Search Space Design (cont’d)

Sub-policy 1. (Color, 0.2, 8), (Rotate, 0.8, 10)
Sub-policy 2. (BBox_Only_ShearY, 0.8, 5)
Sub-policy 3. (SolarizeAdd, 0.6, 8), (Brightness, 0.8, 10)
Sub-policy 4. (ShearY, 0.6, 10), (BBox_Only_Equalize, 0.6, 8)
Sub-policy 5. (Equalize, 0.6, 10), (TranslateX, 0.2, 2)
Search Algorithm

- Very similar to NAS-FPN

- Controller
  - RNN-based controller
  - Search with Proximal Policy Optimization (PPO)

- Evaluation
  - A small proxy dataset
  - Short-time training
Results

- Significantly outperforms previous state-of-the-arts

<table>
<thead>
<tr>
<th>Backbone</th>
<th>Baseline</th>
<th>Our result</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-50</td>
<td>36.7</td>
<td>39.0</td>
<td>+2.3</td>
</tr>
<tr>
<td>ResNet-101</td>
<td>38.8</td>
<td>40.4</td>
<td>+1.6</td>
</tr>
<tr>
<td>ResNet-200</td>
<td>39.9</td>
<td>42.1</td>
<td>+2.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>36.7</td>
</tr>
<tr>
<td>baseline + DropBlock [13]</td>
<td>38.4</td>
</tr>
<tr>
<td>Augmentation policy with color operations</td>
<td>37.5</td>
</tr>
<tr>
<td>+ geometric operations</td>
<td>38.6</td>
</tr>
<tr>
<td>+ bbox-only operations</td>
<td><strong>39.0</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Change</th>
<th># Scales</th>
<th>mAP</th>
<th>mAP$_S$</th>
<th>mAP$_M$</th>
<th>mAP$_L$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MegDet [32]</td>
<td>baseline [14]</td>
<td>multiple</td>
<td>50.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>AmoebaNet + NAS-FPN</td>
<td>+ learned augmentation</td>
<td>1</td>
<td>47.0</td>
<td>30.6</td>
<td>50.9</td>
<td>61.3</td>
</tr>
<tr>
<td></td>
<td>+ ↑ anchors, ↑ image size</td>
<td>1</td>
<td><strong>50.7</strong></td>
<td><strong>34.2</strong></td>
<td><strong>55.5</strong></td>
<td><strong>64.5</strong></td>
</tr>
</tbody>
</table>
Better regularization
Future Work

- More search dimensions
  - E.g. loss, anchor boxes, assign rules, post-processing, ...

- Reducing search cost

- Joint optimization