Automated Machine Learning – An Introduction to Network Architecture Search

Lecture 8 for Advanced Deep Learning Systems

Aaron Zhao, Imperial College London, a.zhao@imperial.ac.uk

Table of contents

Introduction

Introduction

During the last lecture, we explored optimizing our network through informed manual adjustments. Today, we'll delve into Automated Machine Learning and the process of Network Architecture Search.

Introduction - Automated Machine Learning

The complete AutoML pipeline consists of:

- Data preparation and automated data cleaning
- Feature Engineering
- Model selection or Network Architecture Search
- Hyperparameter tuning ...

Introduction - Network Architecture Search

Core idea: Can we design the best network architecture purely from observing the data?

We want to pick the optimal architecture $a \in \mathcal{A}$ from a set of architectures $\mathcal{A}$.

At the same time, we want to pick the optimal parameters $w^*(a)$ for the architecture a.

$$\begin{aligned} min_{a \in \mathcal{A}} \mathcal{L}_{val}(w^*(a), a) \\ s.t. w^*(a) = argmin_w(\mathcal{L}_{train}(w, a)) \end{aligned} \tag{1}$$

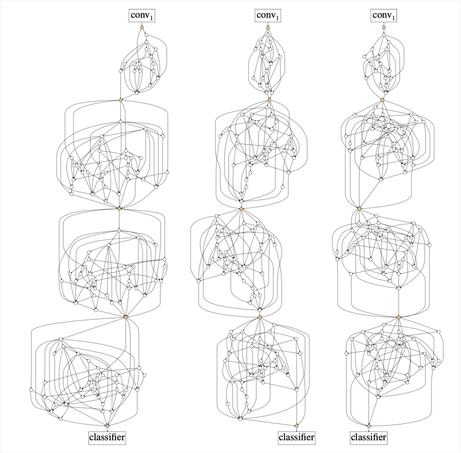
Search Space

What is a search space?

We normally have to search through a given set of architectures $\mathcal{A}$, which is known as the search space.

- Global search space
- Modular search space
- Combined search space

Global search space



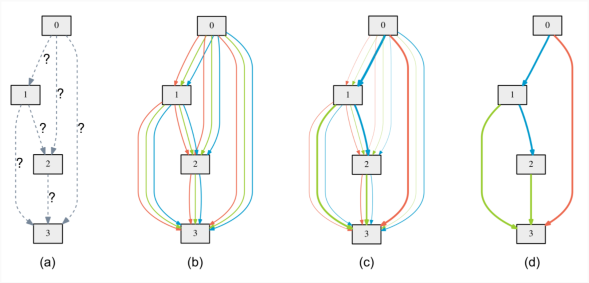
A global search space considers directly all possible elements (search options) in the DAG (Directed Acyclic Graph): this is an extremely large search space, not even tractable.

Modular search space

- Search a critical component in the search space, and then duplicate that component based on heuristics (DARTS).
- Template and backbone template with heuristics, search for the design options in the backbone (MobileNet-V3).

The DARTS search space

- 6-node DAG as a Cell
  - two input nodes
  - one output node
  - four intermediate data nodes
- Nodes are connected by an operation
  - conv3x3
  - conv1x1
  - skip, relu, sigmoid, tanh and so on.
- roughly 10^9 options



The DARTS search space

- Normal Cell (no resolution change)
- Reduction Cell (resolution changes with striding/pooling)

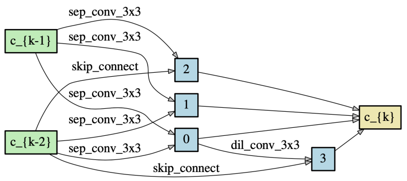


Figure 4: Normal cell learned on CIFAR-10.

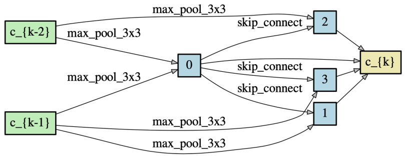
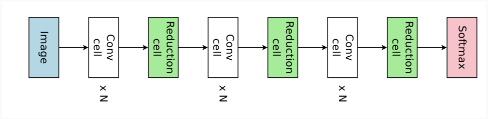


Figure 5: Reduction cell learned on CIFAR-10.

The DARTS search space

- Searched on CIFAR10 with one normal and one reduction cell.
- There exists an approximation/belief that architectures learned on CIFAR10 are generally transferrable to other image tasks!
- Stack the searched cell multiple times to build bigger networks on ImageNet.
- There exists a heuristics-defined stacking pattern! Figure is from ENAS but DARTS does the same thing.

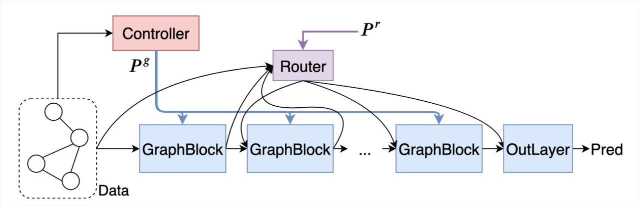


Combined search space

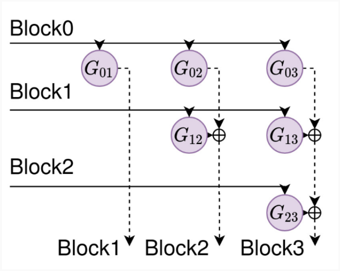
- Micro-architecture design: the design of a single cell, through the modular search space.
- Macro-architecture design: how these cells should be connected, through the global search space.
- The decomposition makes the total search space smaller and more tractable.

Combined search space

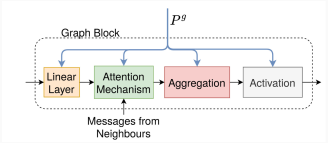
Combined search space



Macro-architecture



Micro-architecture



Search Strategies

What is a search strategy?

The core-algorithm used to conduct the search.

- Reinforcement learning
- Gradient-based optimization
- Evolutionary Algorithm
- Performance estimator

Reinforcement Learning based NAS (NASNet)

A controller, parameterized by θ_c performs a list of actions $(a_0, a_1 ... a_{T-1})$ to design the architecture of a child network.

The child network achieves an accuracy R on a held-out validation dataset.

Concretely, the controller is optimizing a reward:

$$J(\theta_c) = E_{P(\{a_0,\ldots,a_{T-1}\},\theta_c)}[R] \tag{2}$$

Reinforcement Learning based NAS (NASNet)

The reward signal R is non-differentiable, so we use a policy gradient to iteratively update θ_c

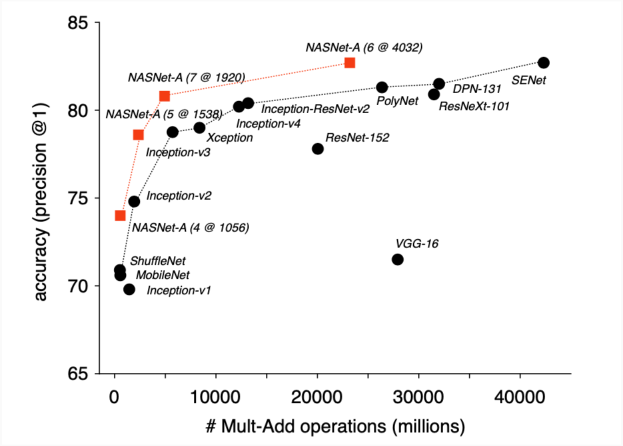
$$J(\theta_c) = \frac{1}{M} \sum_{k=0}^{M-1} \sum_{t=0}^{T-1} logP(a_t | \{a_{t-1} ... a_0\}; \theta_c) R \tag{3}$$

Where M is the number of different architectures that the controller samples in one batch and T is the number of hyperparameters our controller has to predict to design a neural network architecture.

This is slightly different from standard RL setup since there is only basically two states (start and finish) in the trajectory.

It requires a full-training run to obtain R, so this is very expensive!

Reinforcement Learning based NAS (NASNet)



Gradient-based NAS (DARTs)

The problem with RL-based NAS is that it is hard for the controller to receive gradients that relates to the child network accuracy.

But what if we can make the operators differentiable?

Gradient-based NAS (DARTs)

We associate each operator with a trainable parameter:

Let O be a set of candidate operators (eg. convolution, max pool ...), To make a continuous search space, one can relax the discrete categorical distribution:

$$\bar{o}^{(i,j)}(x) = \sum_{o \in O} \frac{exp(\alpha_o^{(i,j)})}{\sum_{o' \in O} exp(\alpha_{o'}^{(i,j)})} o(x) \tag{4}$$

The operation mixing weights for a pair of nodes (i, j) are parameterized by a vector $\alpha(i, j)$ of dimension $|O|$.

Translate this to words: weighting each operator using trainable α values.

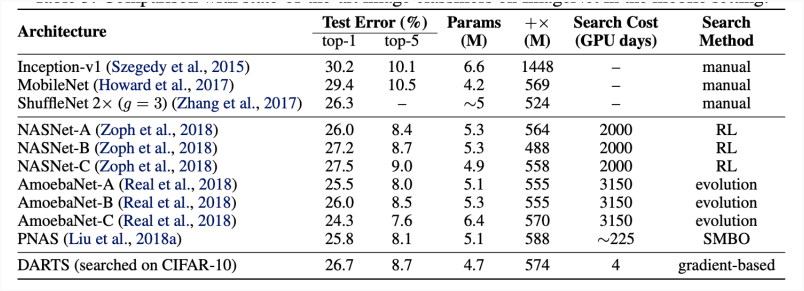
Gradient-based NAS (DARTs)

- Update architecture α by descending $\nabla_\alpha L_{\text{val}}(w - \xi \nabla_w L_{\text{train}}(w, \alpha), \alpha)$, ξ is zero for first order optimization)
- Update weights w by descending $\nabla_w L_{\text{train}}(w, \alpha)$
- Derive the final architecture based on the learned α.

Dual-level optimization using SGD.

Gradient-based NAS (DARTs)

Significant time reduction, since now we run search on the cell-based space only once! There is now no need for evaluating each child network



Architecture	Test Error (%)		Params (M)	$+\times$ (M)	Search Cost (GPU days)	Search Method
	top-1	top-5				
Inception-v1 (Szegedy et al., 2015)	30.2	10.1	6.6	1448	–	manual
MobileNet (Howard et al., 2017)	29.4	10.5	4.2	569	–	manual
ShuffleNet 2× ($g = 3$) (Zhang et al., 2017)	26.3	–	∼5	524	–	manual
NASNet-A (Zoph et al., 2018)	26.0	8.4	5.3	564	2000	RL
NASNet-B (Zoph et al., 2018)	27.2	8.7	5.3	488	2000	RL
NASNet-C (Zoph et al., 2018)	27.5	9.0	4.9	558	2000	RL
AmoebaNet-A (Real et al., 2018)	25.5	8.0	5.1	555	3150	evolution
AmoebaNet-B (Real et al., 2018)	26.0	8.5	5.3	555	3150	evolution
AmoebaNet-C (Real et al., 2018)	24.3	7.6	6.4	570	3150	evolution
PNAS (Liu et al., 2018a)	25.8	8.1	5.1	588	∼225	SMBO
DARTS (searched on CIFAR-10)	26.7	8.7	4.7	574	4	gradient-based

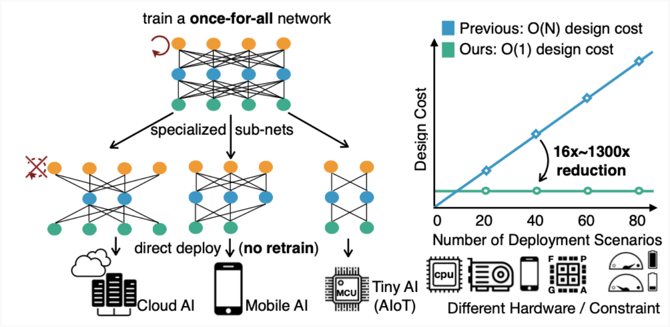
Evolutionary Algorithm (OFA)

Gradient-based methods suffer from the following problems

- Large VRAM usage (Single-path method)
- Is the ranking on a proxy dataset reliable?
- Gradient interference (PC-DARTs)
- No awareness about the deployment device and scenarios

Evolutionary Algorithm (OFA)

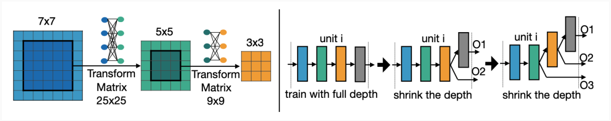
What if we train a supernet, and subsample it?

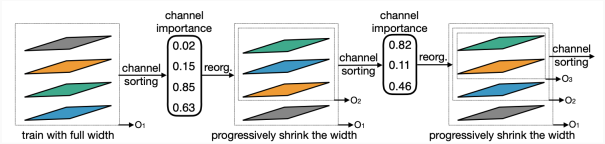

train a **once-for-all** network
specialized sub-nets
direct deploy (**no retrain**)
Cloud AI
Mobile AI
MCU
Tiny AI (AIoT)
Previous: O(N) design cost
Ours: O(1) design cost
Design Cost
16x~1300x reduction
0
20
40
60
80
Number of Deployment Scenarios
cpu
Different Hardware / Constraint

Evolutionary Algorithm (OFA)

Design a large supernet that supports sub-sampling in various dimensions

- Kernels
- Channels
- Layers





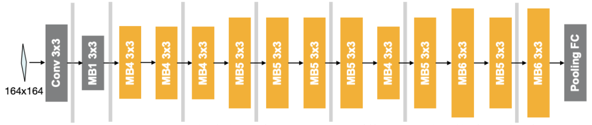
Evolutionary Algorithm (OFA)

Then use an evolutionary algorithm to traverse the whole search space.

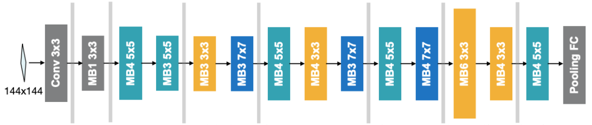
- **Initialization and Evaluation:** At the start, generate a random population of individuals, and evaluate their performance
- **Selection:** Individuals are selected based on their fitness scores.
- **Crossover:** Selected individuals pair and exchange parts of their structure, creating a new individual called offspring. This process is also known as recombination or mating.
- **Mutation:** Offspring are altered randomly to introduce variability in the population. This prevents the genetic algorithm from stagnating at local optima.
- **Replacement:** The population is updated with the new offspring, usually replacing the least fit individuals. The algorithm then goes back to the evaluation step, creating a loop.

Evolutionary Algorithm (OFA)

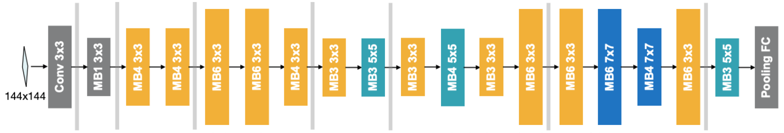
Do the search with different hardware targets, use the latency and accuracy to build a fitness score



(a) 4.1ms latency on Xilinx ZU3EG (batch size = 1).



(b) 10.9ms latency on Intel Xeon CPU (batch size = 1).

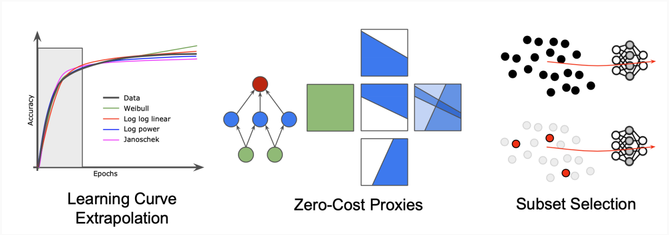


(c) 14.9ms latency on NVIDIA 1080Ti (batch size = 64).

Performance Predictor (Speedy Performance Estimator)

All previous methods require training of one or more networks. Can we predict the performance without or with minimal training?

- Learning curve extrapolation: extrapolating the validation accuracy learning curve via a parameteric model.
- Zero-cost proxies: assessing the generalizability of an architecture with a single forward pass of a single minibatch of data
- Subset selection: Training the architecture on a subset of the data



Performance Predictor (Speedy Performance Estimator)

An example performance predictor can be the following: Let L denote a loss function, $f_\theta(x)$ the output of a neural network f with input x and parameters θ, and let $\theta_{t,i}$ denote the parameters of the network after t epochs and i minibatches of SGD.

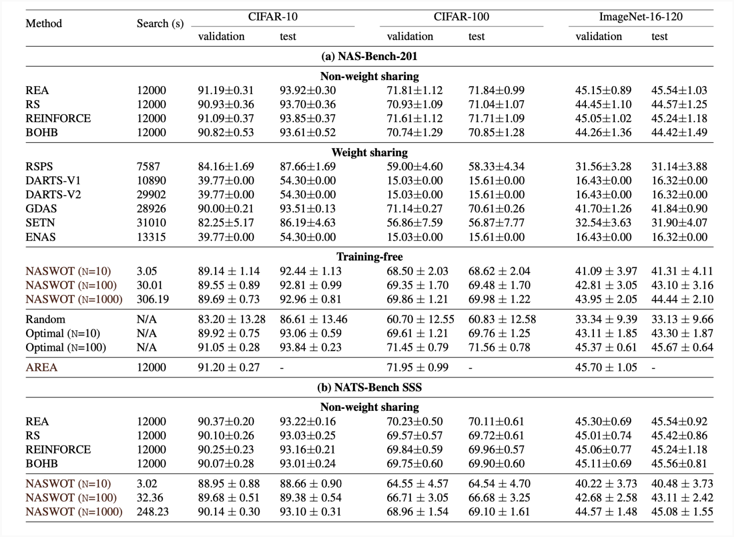
After training the network for T epochs, we sum the training losses collected so far to get the following Training Speed Estimate (TSE)

$$TSE = \sum_{t=1}^{T} [\frac{1}{B} \sum_{1}^{B} L(f_{\theta_{t,i}(x_i),y_i})] \tag{5}$$

The idea is then use this for evaluating the sub-networks, since it gets a better rank correlation performance usually.

Performance Predictor (NASWOT)

Or you can simply take a number of networks and evaluate all of them using these low-fidelity proxies!



Method	Search (s)	CIFAR-10		CIFAR-100		ImageNet-16-120	
		validation	test	validation	test	validation	test
(a) NAS-Bench-201							
Non-weight sharing							
REA	12000	91.19±0.31	93.92±0.30	71.81±1.12	71.84±0.99	45.15±0.89	45.54±1.03
RS	12000	90.93±0.36	93.70±0.36	70.93±1.09	71.04±1.07	44.45±1.10	44.57±1.25
REINFORCE	12000	91.09±0.37	93.85±0.37	71.61±1.12	71.71±1.09	45.05±1.02	45.24±1.18
BOHB	12000	90.82±0.53	93.61±0.52	70.74±1.29	70.85±1.28	44.26±1.36	44.42±1.49
Weight sharing							
RSPS	7587	84.16±1.69	87.66±1.69	59.00±4.60	58.33±4.34	31.56±3.28	31.14±3.88
DARTS-V1	10890	39.77±0.00	54.30±0.00	15.03±0.00	15.61±0.00	16.43±0.00	16.32±0.00
DARTS-V2	29902	39.77±0.00	54.30±0.00	15.03±0.00	15.61±0.00	16.43±0.00	16.32±0.00
GDAS	28926	90.00±0.21	93.51±0.13	71.14±0.27	70.61±0.26	41.70±1.26	41.84±0.90
SETN	31010	82.25±5.17	86.19±4.63	56.86±7.59	56.87±7.77	32.54±3.63	31.90±4.07
ENAS	13315	39.77±0.00	54.30±0.00	15.03±0.00	15.61±0.00	16.43±0.00	16.32±0.00
Training-free							
NASWOT (N=10)	3.05	89.14 ± 1.14	92.44 ± 1.13	68.50 ± 2.03	68.62 ± 2.04	41.09 ± 3.97	41.31 ± 4.11
NASWOT (N=100)	30.01	89.55 ± 0.89	92.81 ± 0.99	69.35 ± 1.70	69.48 ± 1.70	42.81 ± 3.05	43.10 ± 3.16
NASWOT (N=1000)	306.19	89.69 ± 0.73	92.96 ± 0.81	69.86 ± 1.21	69.98 ± 1.22	43.95 ± 2.05	44.44 ± 2.10
Random	N/A	83.20 ± 13.28	86.61 ± 13.46	60.70 ± 12.55	60.83 ± 12.58	33.34 ± 9.39	33.13 ± 9.66
Optimal (N=10)	N/A	89.92 ± 0.75	93.06 ± 0.59	69.61 ± 1.21	69.76 ± 1.25	43.11 ± 1.85	43.30 ± 1.87
Optimal (N=100)	N/A	91.05 ± 0.28	93.84 ± 0.23	71.45 ± 0.79	71.56 ± 0.78	45.37 ± 0.61	45.67 ± 0.64
AREA	12000	91.20 ± 0.27	-	71.95 ± 0.99	-	45.70 ± 1.05	-
(b) NATS-Bench SSS							
Non-weight sharing							
REA	12000	90.37±0.20	93.22±0.16	70.23±0.50	70.11±0.61	45.30±0.69	45.54±0.92
RS	12000	90.10±0.26	93.03±0.25	69.57±0.57	69.72±0.61	45.01±0.74	45.42±0.86
REINFORCE	12000	90.25±0.23	93.16±0.21	69.84±0.59	69.96±0.57	45.06±0.77	45.24±1.18
BOHB	12000	90.07±0.28	93.01±0.24	69.75±0.60	69.90±0.60	45.11±0.69	45.56±0.81
NASWOT (N=10)	3.02	88.95 ± 0.88	88.66 ± 0.90	64.55 ± 4.57	64.54 ± 4.70	40.22 ± 3.73	40.48 ± 3.73
NASWOT (N=100)	32.36	89.68 ± 0.51	89.38 ± 0.54	66.71 ± 3.05	66.68 ± 3.25	42.68 ± 2.58	43.11 ± 2.42
NASWOT (N=1000)	248.23	90.14 ± 0.30	93.10 ± 0.31	68.96 ± 1.54	69.10 ± 1.61	44.57 ± 1.48	45.08 ± 1.55

Is This A Solved Problem?

Well, Yes and No

- Search spaces are normally manually defined!
- Low-fidelity methods normally work at small search spaces.
- Many of the search methods actually overfit to the search space!

Summary

Search Spaces: Global, Modular and Combined

Search Strategies

- Reinforcement learning
- Gradient-based optimization
- Evolutionary algorithm
- Performance estimators