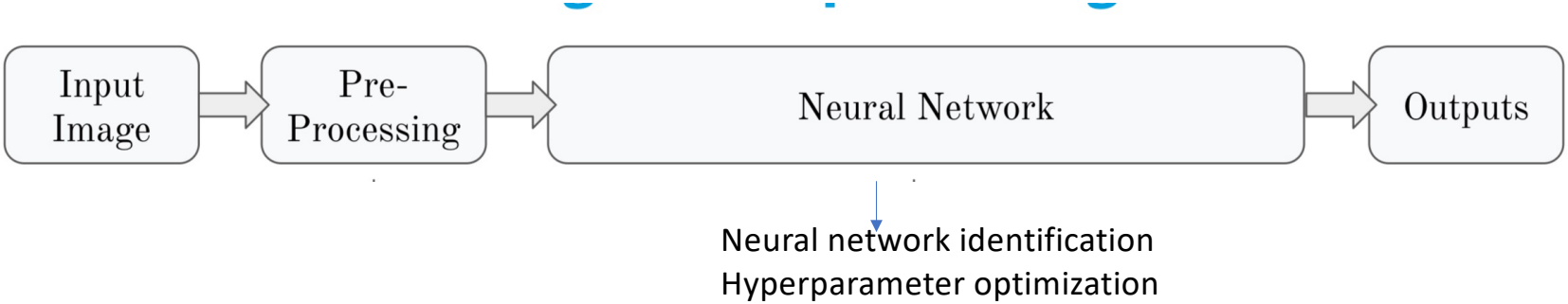
DS-UA 301
Advanced Topics in Data Science
Advanced Techniques in ML and Deep Learning

LECTURE 12

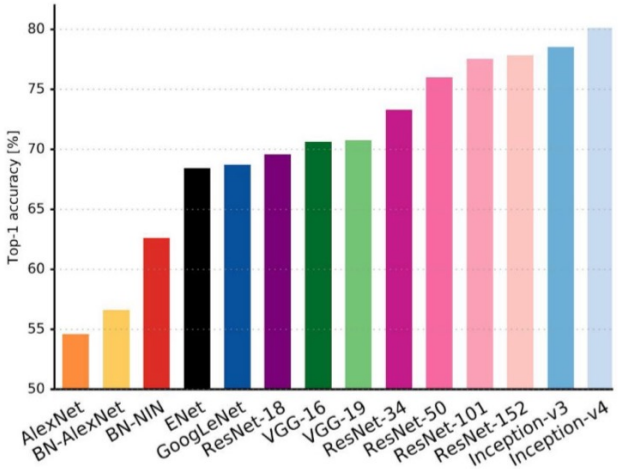
Parijat Dube

Automated Machine Learning

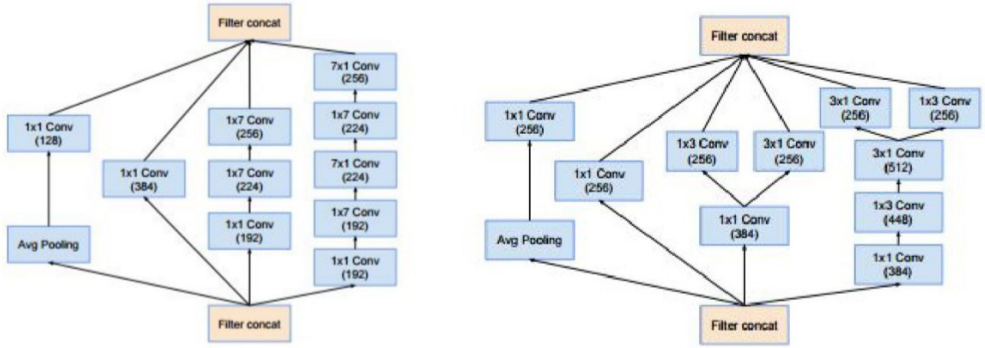


- Data Preprocessing: normalization, data-augmentation
- Neural architecture search (NAS)
  - Standard, off the shelf
  - Synthesize a new
    - Types of layers (conv, maxpool), number of different layers, how to stack the layers
    - Convolution layer parameters (filter dim, stride dim)
- Hyperparameter optimization
  - Batch size, learning rate, momentum
- NAS and hyperparameter optimization can be done jointly or sequentially

Importance of Neural Architectures in Vision



Canziani et al (2017)



Complex hand-engineered layers from Inception-V4 (Szegedy et al., 2017)

Design Innovations (2012 - Present): Deeper networks, stacked modules, skip connections, squeeze-excitation block, ...

Can we try and learn good architectures automatically?

https://metalearning-cvpr2019.github.io/assets/CVPR_2019_Metalearning_Tutorial_Nikhil_Naik.pdf

Neural Architecture Search

Defines the neural architectures a NAS approach may discover

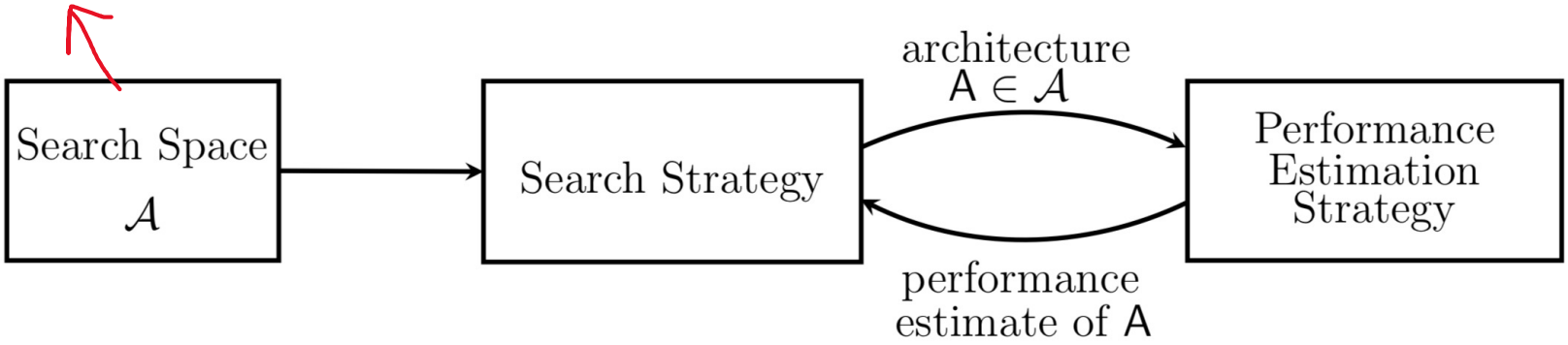
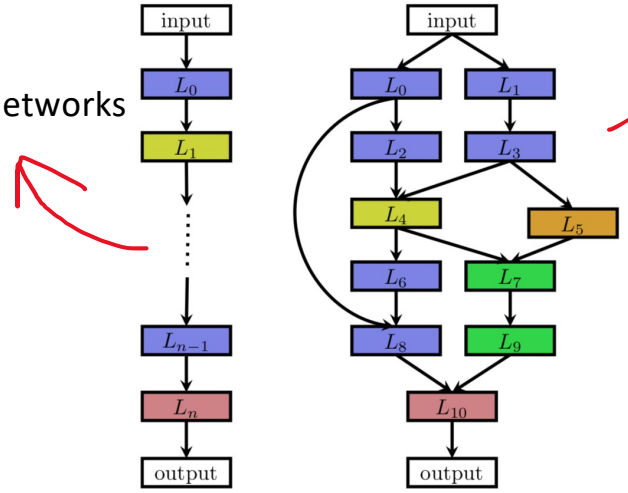


Figure 1: Abstract illustration of Neural Architecture Search methods. A search strategy selects an architecture A from a predefined search space $\mathcal{A}$. The architecture is passed to a performance estimation strategy, which returns the estimated performance of A to the search strategy.

Elsken et al. Neural Architecture Search: A Survey. JMLR 2019

NAS Search Space: Layer Based

Chain structured networks



Mutli branch networks with skip connections

Problem with layer based NAS search space

- Search space is of the order of layer types and associated parameters
  - Conv layer: filter size, number of filters, stride length, padding

Figure 2: An illustration of different architecture spaces. Each node in the graphs corresponds to a layer in a neural network, e.g., a convolutional or pooling layer. Different layer types are visualized by different colors. An edge from layer L_i to layer L_j denotes that L_j receives the output of L_i as input. Left: an element of a chain-structured space. Right: an element of a more complex search space with additional layer types and multiple branches and skip connections.

Elsken et al. Neural Architecture Search: A Survey. JMLR 2019

NAS Search Space: Cell Based

Normal cell

Reduction cell

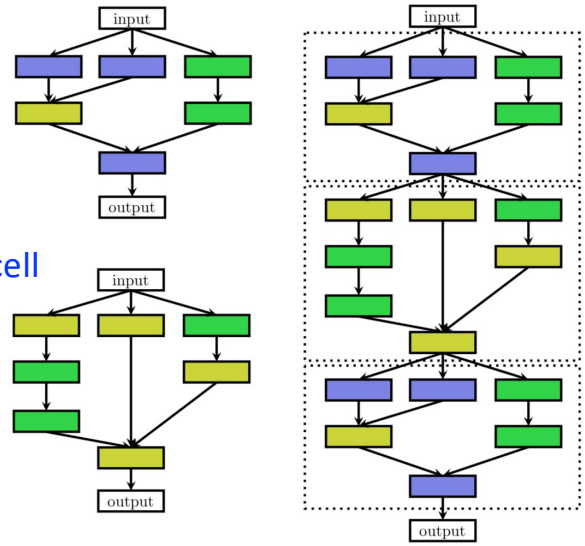


Figure 3: Illustration of the cell search space. Left: Two different cells, e.g., a normal cell (top) and a reduction cell (bottom) (Zoph et al., 2018). Right: an architecture built by stacking the cells sequentially. Note that cells can also be combined in a more complex manner, such as in multi-branch spaces, by simply replacing layers with cells.

Motivated by hand-crafted architectures consisting of repeated motifs (cells/blocks)

- Inception blocks: normal cell, reduction cell
- Residual blocks: normal cell reduction cell

Elsken et al. Neural Architecture Search: A Survey. JMLR 2019

NAS Search Space: Cell Based

Advantages of cell based NAS search space

- Drastically reduced search space compared to layer based
- Easy adaptability of a network for across datasets of varying size and complexity a given domain
- Applicability across different domains, LSTM blocks in RNN, inception and residual blocks in CNNs

- Macro-architecture search problem: how many cells shall be used and how should they be connected to build the actual model?

NAS Search Strategy

- Random search
- Bayesian optimization
- **Evolutionary algorithms (EA)**
- **Reinforcement learning (RL)**
- **Gradient based methods**

EA based NAS search

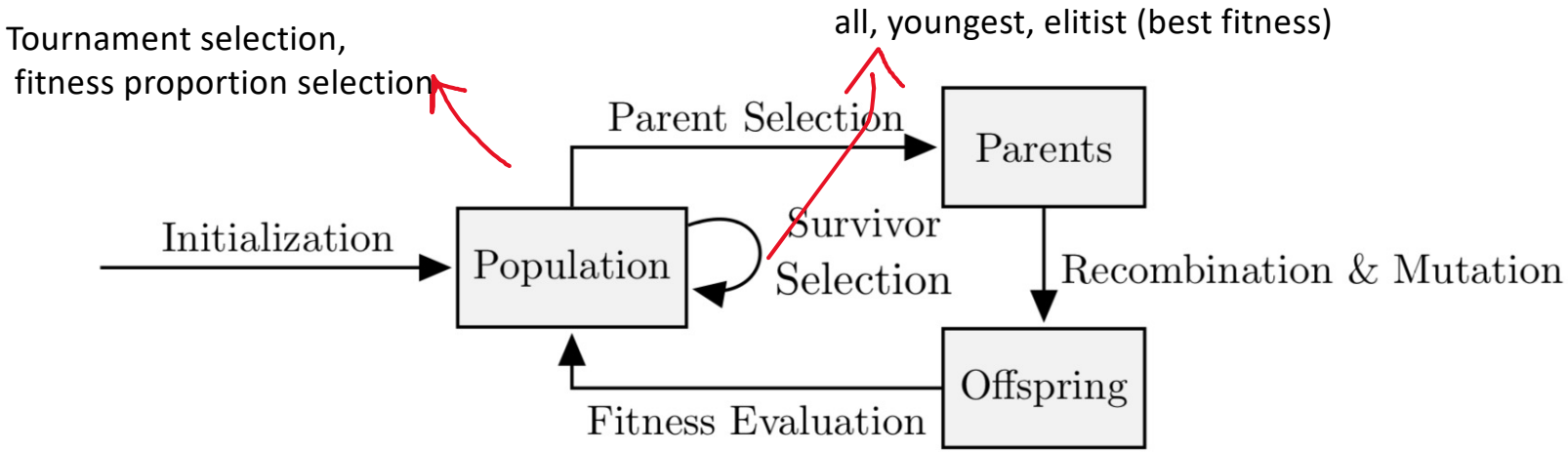


Figure 11: A general framework for evolutionary algorithms.

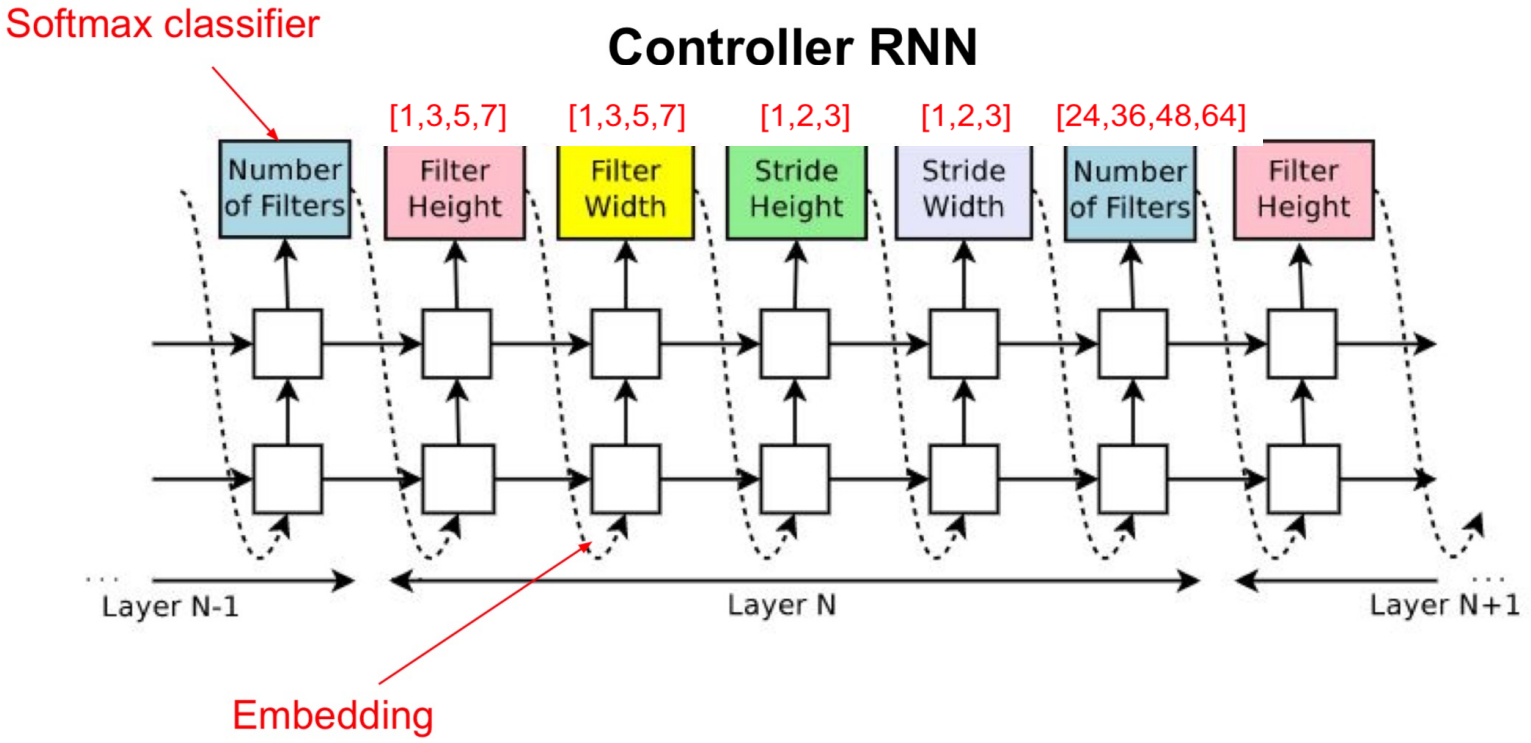
- Earlier work used genetic algorithms to optimize both the neural architecture and its weights
- Recent work use evolutionary algorithms only for optimizing the neural architecture whereas the weights of each candidate architecture are optimized using gradient-based methods like SGD or its variants

Witsuba et al. A Survey on Neural Architecture Search. 2019

RL based NAS search

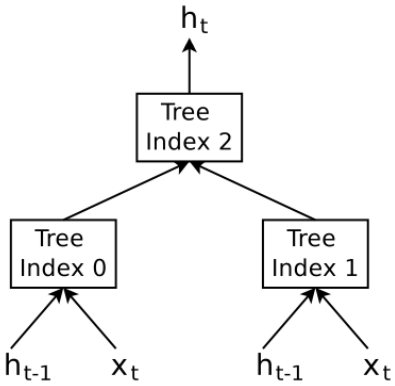
- Specify the structure and connectivity of a neural network by using a configuration string (e.g., *[“Filter Width: 5”, “Filter Height: 3”, “Num Filters: 24”]*)
- Zoph and Le (2017): Use a RNN (“Controller”) to generate this string that specifies a neural network architecture
- Train this architecture (“Child Network”) to see how well it performs on a validation set
- Use reinforcement learning to update the parameters of the Controller model based on the accuracy of the child model

Neural Architecture Search for Convolutional Networks

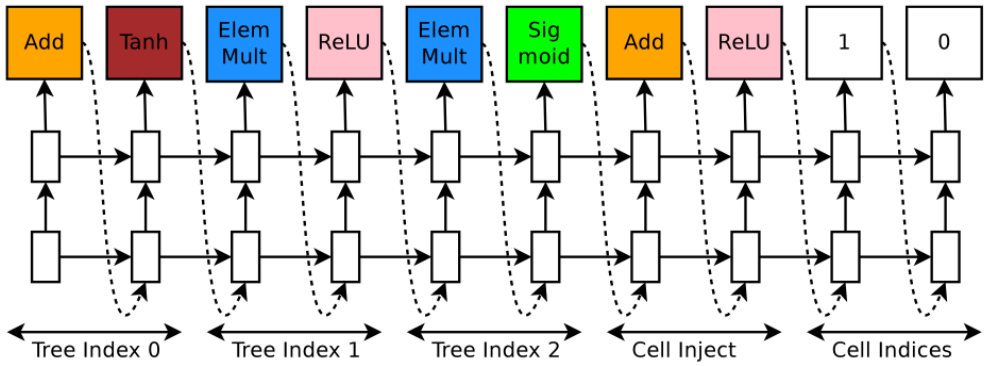


Zoph et al. NEURAL ARCHITECTURE SEARCH WITH REINFORCEMENT LEARNING. ICLR 2017

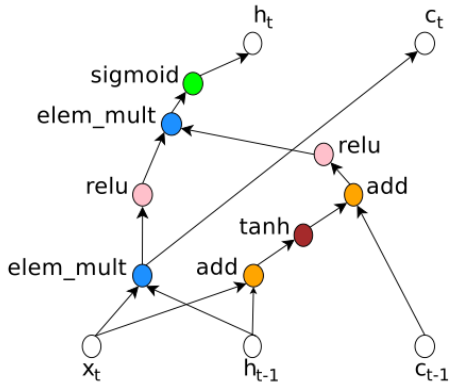
Recurrent Cell Prediction from NAS



tree that defines the computation steps to be predicted by controller



example set of predictions made by the controller for each computation step in the tree

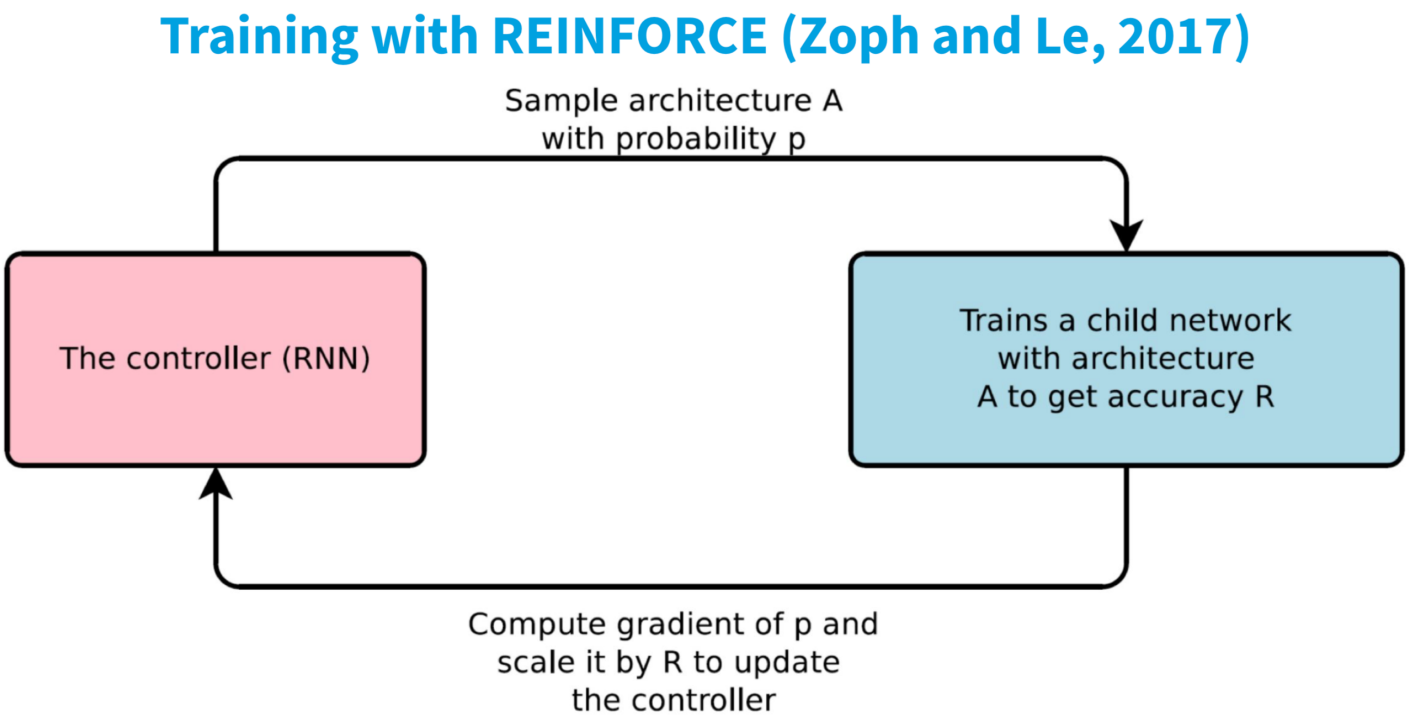


computation graph of the recurrent cell constructed from example predictions of the controller

Computation Steps in Predicted RNN Cell

- The controller predicts Add and $Tanh$ for tree index 0, this means we need to compute $a_0 = \tanh(W_1 * x_t + W_2 * h_{t-1})$.
- The controller predicts $ElemMult$ and $ReLU$ for tree index 1, this means we need to compute $a_1 = \mathrm{ReLU}\big((W_3 * x_t) \odot (W_4 * h_{t-1})\big)$.
- The controller predicts 0 for the second element of the "Cell Index", Add and $ReLU$ for elements in "Cell Inject", which means we need to compute $a_0^{new} = \mathrm{ReLU}(a_0 + c_{t-1})$. Notice that we don't have any learnable parameters for the internal nodes of the tree.
- The controller predicts $ElemMult$ and $Sigmoid$ for tree index 2, this means we need to compute $a_2 = \mathrm{sigmoid}(a_0^{new} \odot a_1)$. Since the maximum index in the tree is 2, h_t is set to a_2.
- The controller RNN predicts 1 for the first element of the "Cell Index", this means that we should set c_t to the output of the tree at index 1 before the activation, i.e., $c_t = (W_3 * x_t) \odot (W_4 * h_{t-1})$.

RL based NAS Search



Zoph et al. NEURAL ARCHITECTURE SEARCH WITH REINFORCEMENT LEARNING. ICLR 2017

Training with REINFORCE

Parameters of Controller RNN

Accuracy of architecture on held-out dataset

$$J(\theta_c) = E_{P(a_{1:T};\theta_c)}[R]$$

Architecture predicted by the controller RNN viewed as a sequence of actions

RENFORCE rule to iteratively update θ_c

$$\nabla_{\theta_c} J(\theta_c) = \sum_{t=1}^{T} E_{P(a_{1:T};\theta_c)}\big[\nabla_{\theta_c} \log P(a_t|a_{(t-1):1};\theta_c)R\big]$$

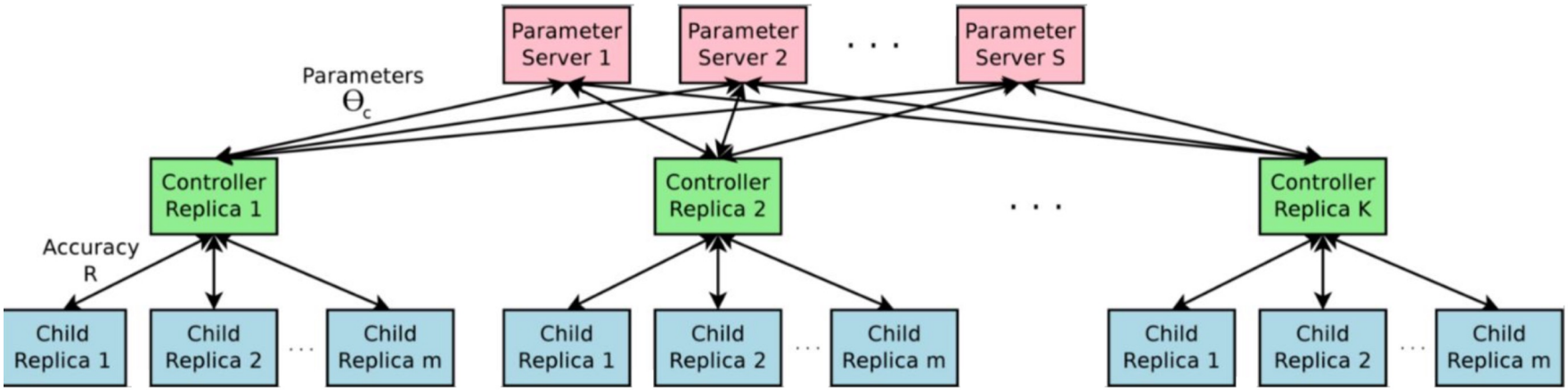
An empirical approximation of the above quantity is:

$$\frac{1}{m}\sum_{k=1}^{m}\sum_{t=1}^{T} \nabla_{\theta_c} \log P(a_t|a_{(t-1):1};\theta_c)R_k$$

Scaling NAS

Distributed Training

- Parameter server of S shards
- Each controller samples m child networks and train them in parallel
- The controller then collects gradients according to the results of that minibatch of m architectures at convergence and sends them to the parameter server in order to update the weights across all controller replicas



Zoph et al. NEURAL ARCHITECTURE SEARCH WITH REINFORCEMENT LEARNING. ICLR 2017

Computational Cost of NAS with RL and EA

CIFAR10 dataset

	Reference	Error (%)	Params (Millions)	GPU Days
RL	Baker et al. (2017)	6.92	11.18	100
	Zoph and Le (2017)	3.65	37.4	22,400
	Cai et al. (2018a)	4.23	23.4	10
	Zoph et al. (2018)	3.41	3.3	2,000
	Zoph et al. (2018) + Cutout	2.65	3.3	2,000
	Zhong et al. (2018)	3.54	39.8	96
	Cai et al. (2018b)	2.99	5.7	200
	Cai et al. (2018b) + Cutout	2.49	5.7	200
EA	Real et al. (2017)	5.40	5.4	2,600
	Xie and Yuille (2017)	5.39	N/A	17
	Suganuma et al. (2017)	5.98	1.7	14.9
	Liu et al. (2018b)	3.75	15.7	300
	Real et al. (2019)	3.34	3.2	3,150
	Elsken et al. (2018)	5.2	19.7	1
	Wistuba (2018a) + Cutout	3.57	5.8	0.5

Thousands of GPU hours

Witsuba et al. A Survey on Neural Architecture Search. 2019

Differentiable Architecture Search

Continuous relaxation of the architecture representation, allowing efficient search of the architecture using gradient descent

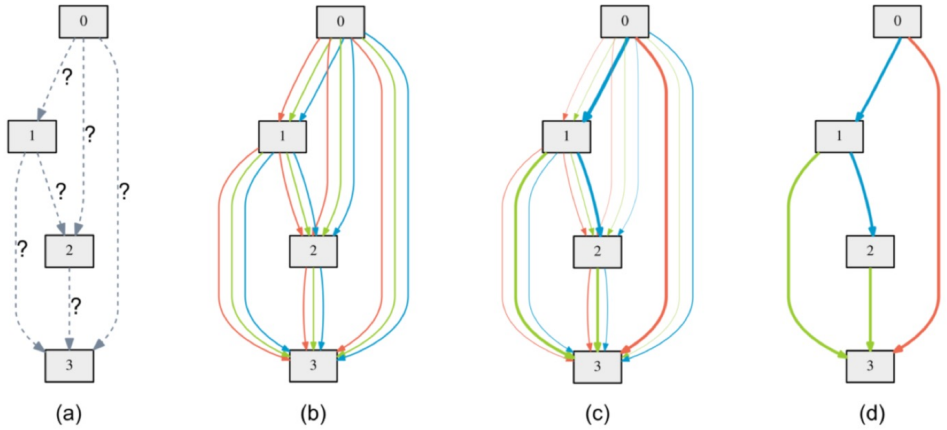


Figure 1: An overview of DARTS: (a) Operations on the edges are initially unknown. (b) Continuous relaxation of the search space by placing a mixture of candidate operations on each edge. (c) Joint optimization of the mixing probabilities and the network weights by solving a bilevel optimization problem. (d) Inducing the final architecture from the learned mixing probabilities.

Liu et al. DARTS: DIFFERENTIABLE ARCHITECTURE SEARCH. ICLR 2019

Details of DARTS

- Computation procedure for an architecture (or a cell in it) is represented as a directed acyclic graph
- Cell-based search
- https://arxiv.org/pdf/1806.09055.pdf

DARTS

- Goal is to jointly learn the architecture α and the weights w within all the mixed operations (e.g. weights of the convolution filters)
- The continuous variable α determines the operation mixing weights for different pair of nodes in the network

$\mathcal{L}_{train}$ and $\mathcal{L}_{val}$ the training and the validation loss,

- Both losses are determined not only by the architecture α, but also the weights w in the network.

$$\begin{aligned} \min_{\alpha} \quad & \mathcal{L}_{val}(w^*(\alpha), \alpha) \\ \text{s.t.} \quad & w^*(\alpha) = \operatorname{argmin}_w \ \mathcal{L}_{train}(w, \alpha) \end{aligned}$$

Bi-level optimization problem

α as the upper-level variable

w as the lower-level variable

NAS Performance Estimation

Speed-up method	How are speed-ups achieved?	References
Lower fidelity estimates	Training time reduced by training for fewer epochs, on subset of data, downscaled models, downscaled data, ...	Li et al. (2017), Zoph et al. (2018), Zela et al. (2018), Falkner et al. (2018), Real et al. (2019), Runge et al. (2019)
Learning Curve Extrapolation	Training time reduced as performance can be extrapolated after just a few epochs of training.	Swersky et al. (2014), Domhan et al. (2015), Klein et al. (2017a), Baker et al. (2017b)
Weight Inheritance/ Network Morphisms	Instead of training models from scratch, they are warm-started by inheriting weights of, e.g., a parent model.	Real et al. (2017), Elsken et al. (2017), Elsken et al. (2019), Cai et al. (2018a,b)
One-Shot Models/ Weight Sharing	Only the one-shot model needs to be trained; its weights are then shared across different architectures that are just subgraphs of the one-shot model.	Saxena and Verbeek (2016), Pham et al. (2018), Bender et al. (2018), Liu et al. (2019b), Cai et al. (2019), Xie et al. (2019)

Elsken et al. Neural Architecture Search: A Survey. JMLR 2019

Reference Papers

- Zoph et al. NEURAL ARCHITECTURE SEARCH WITH REINFORCEMENT LEARNING. ICLR 2017
- Elsken et al. Neural Architecture Search: A Survey. JMLR 2019
- Witsuba et al. A Survey on Neural Architecture Search. 2019
- Liu et al. DARTS: DIFFERENTIABLE ARCHITECTURE SEARCH. ICLR 2019
- Istrate et al. TAPAS: Train-less Accuracy Predictor for Architecture Search. 2018

Reference Materials

- AutoML.org. Literature on Neural Architecture Search. Compilation of papers in NAS. Regularly updated.
- Zoph et al. Neural Architecture Search with Reinforcement Learning. Lecture slides.
- Nikhil Naik. Neural Architecture Search Tutorial. CVPR 2019