Stanford University

CS 329P : Practical Machine Learning (2021 Fall)

9.3 NAS algorithms

Qingqing Huang, Mu Li, Alex Smola

https://c.d2l.ai/stanford-cs329p

Neural Architecture Search (NAS)

- A neural network has different types of hyperparameters:
  - Topological structure: resnet-ish, mobilenet-ish, #layers
  - Individual layers: kernel_size, #channels in convolutional layer, #hidden_outputs in dense/recurrent layers
- NAS automates the design of neural network
  - How to specify the search space of NN
  - How to explore the search space
  - Performance estimation

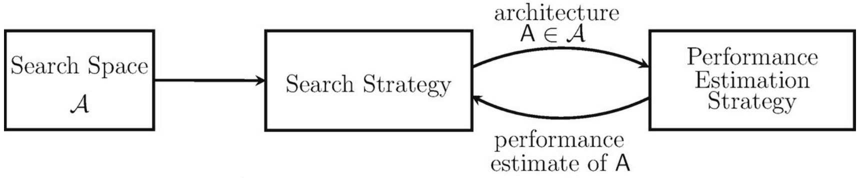


Image source: Elsken, et al. 2019

NAS with Reinforcement Learning

- Zoph & Le 2017
    - A RL-based controller (REINFORCE) for proposing architecture.
    - RNN controller outputs a sequence of tokens to config the model architecture.
    - Reward is the accuracy of a sampled model at convergence
- Naive approach is expensive and sample in-efficient (~2000 GPU days). To speed up NAS:
    - Estimate performance
    - Parameter sharing (e.g. EAS, ENAS)

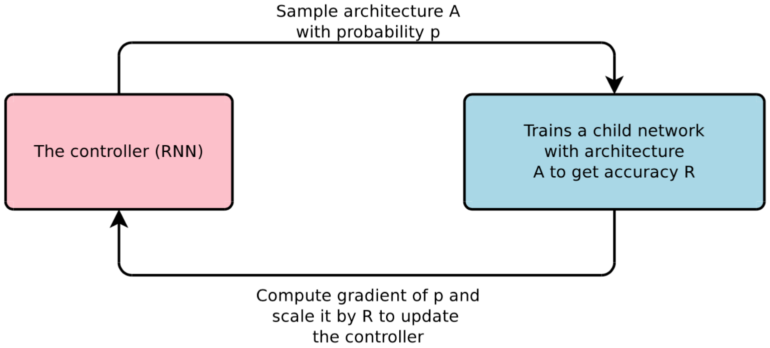


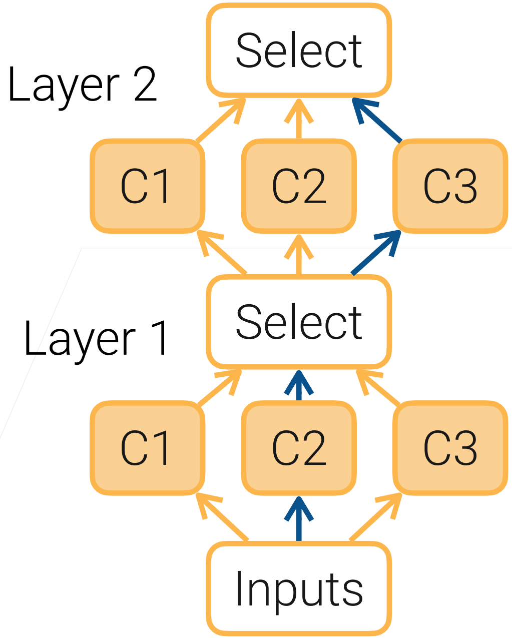
Image source: Zoph & Le 2017

The One-shot Approach

- Combines the learning of architecture and model params
- Construct and train a single model presents a wide variety of architectures
- Evaluate candidate architectures
  - Only care about the candidate ranking
  - Use a proxy metric: the accuracy after a few epochs
- Re-train the most promising candidate from scratch

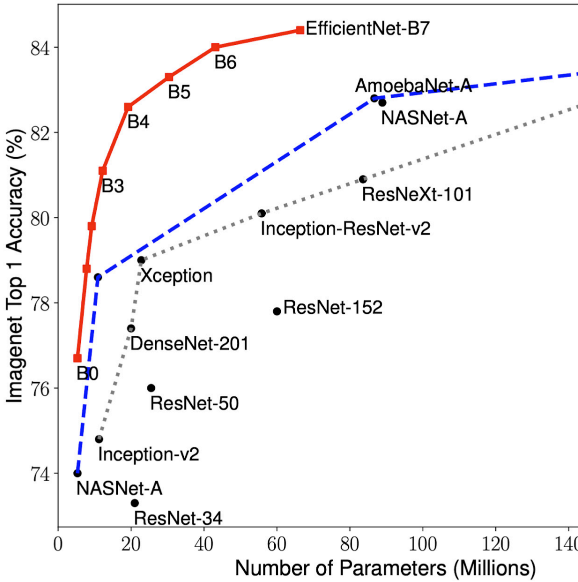
Differentiable Architecture Search

- Relax the categorical choice to a softmax over possible operations:
  - Multiple candidates for each layer
  - Output of i-th candidate at layer l is o_i^l
  - Learn mixing weights $\mathbf{a}^l$. The input for $i+1$-the layer is
    $$\sum_i \alpha_i^l o_i^l \quad \text{with} \quad \boldsymbol{\alpha}^l = \mathrm{softmax}(\mathbf{a}^l)$$
  - Choose candidate $\mathbf{argmax}_i\, \alpha_i$
  - Jointly learn $\mathbf{a}^l$ and network parameters
- A more sophisticated version (DARTS) achieves SOTA and reduces the search time to ~3 GPU days



Scaling CNNs

- A CNN can be scaled by 3 ways:
  - Deeper: more layers
  - Wider: more output channels
  - Larger inputs: increase input image resolutions
- EfficientNet proposes a compound scaling
  - Scale depth by α^{ϕ}, width by β^{ϕ}, resolution by γ^{ϕ}
  - $\alpha\beta^2\gamma^2 \approx 2$ so increase FLOP by 2x if $\phi = 1$
  - Tune $\alpha, \beta, \gamma, \phi$



Research directions

- Explainability of NAS result
- Search architecture to fit into edge devices
  - Edge devices are more and more powerful, data privacy concerns
  - But they are very diverse (CPU/GPU/DSP, 100x performance difference) and have power constraints
  - Minimize both model loss and hardware latency
    - E.g. minimize loss $\times \log(\text{latency})^\beta$
- To what extend can we automates the entire ML workflow?

Summary

- NAS searches a NN architecture for a customizable goal
  - Maximize accuracy or meet latency constraints on particular hardware
- NAS is practical to use now:
  - Compound depth, width, resolution scaling
  - Differentiable one-hot neural network