Foundations of Deep Learning, Winter Term 2021/22

Week 10: Neural Architecture Search (NAS)

Neural Architecture Search (NAS)

Frank Hutter Abhinav Valada

University of Freiburg

Lecture Overview

1. Overview
2. Search Spaces
3. Blackbox Optimization Methods
4. Speedup Techniques
5. DARTS: Differentiable Architecture Search
6. Practical Recommendations for NAS
7. Learning Goals & Further Reading

Foundations of Deep Learning, Winter Term 2021/22

Week 10: Neural Architecture Search (NAS)

Overview

Frank Hutter Abhinav Valada

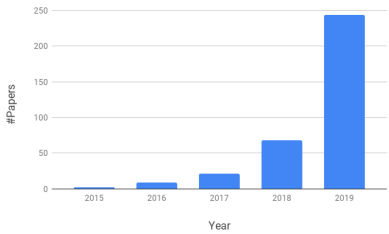
University of Freiburg

Neural Architecture Search (NAS)

- Goal: automatically find neural architectures with strong performance
  - Optionally, subject to a resource constraint

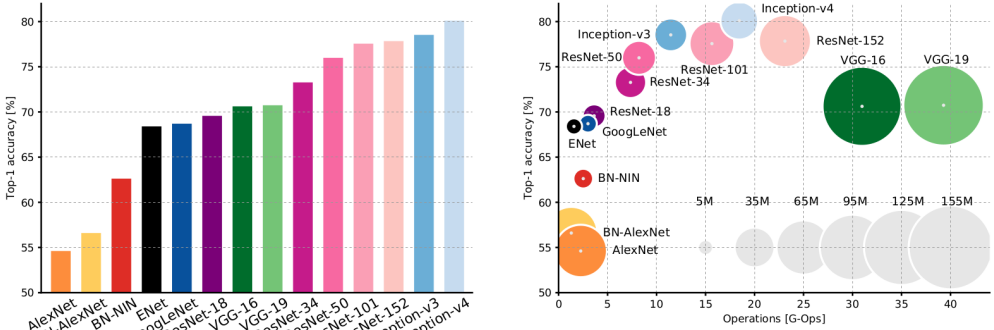
- A decade-old problem, but main stream since 2017 and now intensely researched
- One of the main problems AutoML is known for

- Initially extremely expensive
- By now several methods promise low overhead over a single model training



Motivation for NAS

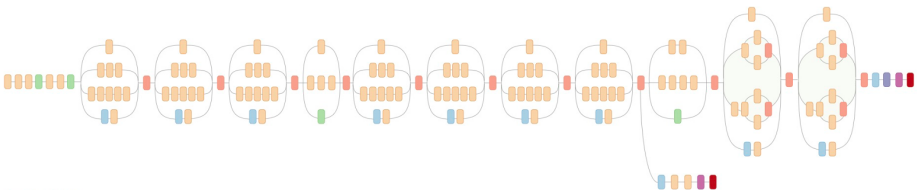
- Performance improvements on various tasks due to novel architectures
- Can we automate this design process, potentially discovering new components/topologies?



[Canziani et al. 2017]

Motivation for NAS

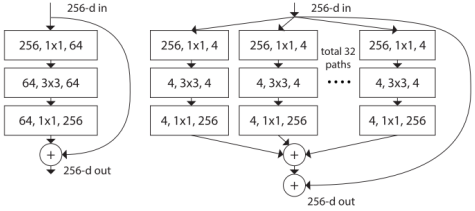
- Manual design of architectures is time consuming
- Complex state-of-the-art architectures are a result of years of trial and errors by experts



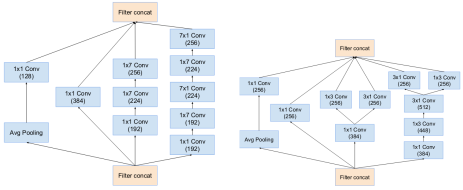
Inception-v3 [Szegedy et al. 2016]

Motivation for NAS

- Manual design of architectures is time consuming
- Complex state-of-the-art architectures are a result of years of trial and errors by experts
  - Main pattern: Repeated blocks with same structure (topology)

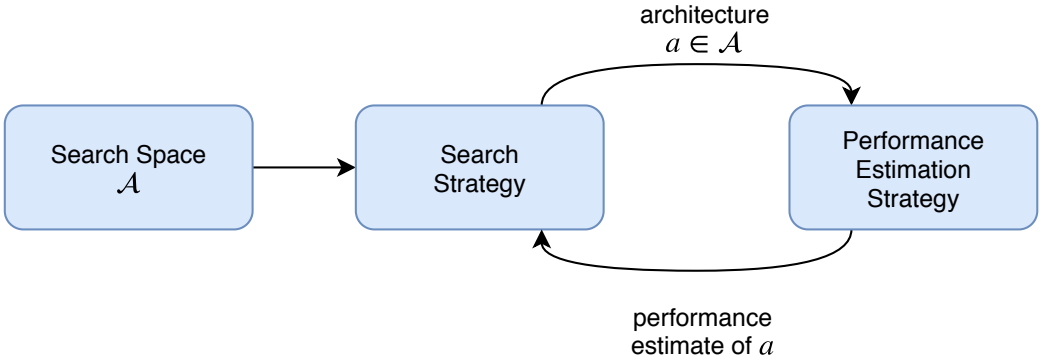


ResNet/ResNeXt blocks
[He et al. 2016; Xie et al. 2016]



Inception-v4 blocks [Szegedy et al. 2017]

NAS components [Elsken et al. 2019]



- Search Space: the types of architectures we consider; micro, macro, hierarchical, etc.
- Search Strategy: Reinforcement learning, evolutionary strategies, Bayesian optimization, gradient-based, etc.
- Performance Estimation Strategy: validation performance, lower fidelity estimates, one-shot model performance, etc.

Questions to Answer for Yourself / Discuss with Friends

- Repetition:
  List three major components of NAS methods.
- Discussion:
  Is there a problem for which you would like to apply NAS yourself?

Lecture Overview

1. Overview
2. Search Spaces
3. Blackbox Optimization Methods
4. Speedup Techniques
5. DARTS: Differentiable Architecture Search
6. Practical Recommendations for NAS
7. Learning Goals & Further Reading

Foundations of Deep Learning, Winter Term 2021/22

Week 10: Neural Architecture Search (NAS)

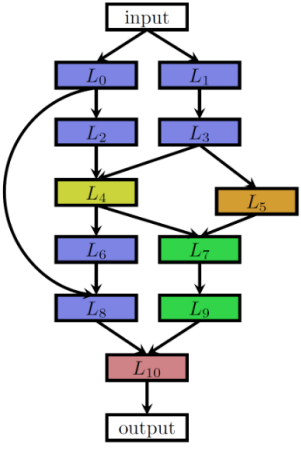
Search Spaces

Frank Hutter Abhinav Valada

University of Freiburg

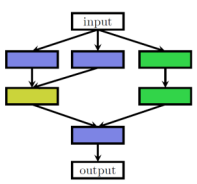
Basic Neural Architecture Search Spaces

Chain-structured space
(different colours:
different layer types)

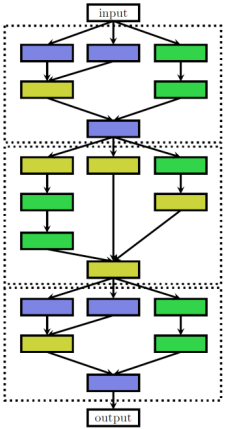


More complex space
with multiple branches
and skip connections

Cell Search Spaces [Zoph et al., 2018]



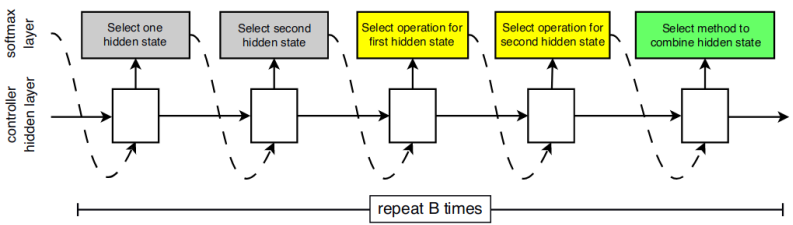
Two possible cells

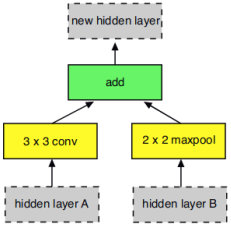


Architecture composed of stacking together individual cells

Details on Cell Search Spaces [Zoph et al., 2018]

- 2 types of cells: normal and reduction cells
- For each type of cell: B blocks, each with 5 choices
  - Choose two previous feature maps (from this cell)
  - For each of these, choose an operation (3×3 conv, max-pool, etc.)
  - Choose a merge operation to combine the two results (concat or add)





Pros and Cons of Cell Search Space

What are some pros and cons of the cell search space compared to the basic one?

Please think about this for a few minutes before continuing.

Pros and Cons of Cell Search Space

Pros:

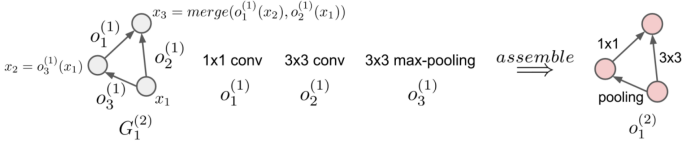
- Reduced search space size; speed-ups in terms of search time.
- Transferability to other datasets (e.g., cells found on CIFAR-10 transfer to ImageNet)
- Stacking repeating patterns is proven to be a useful design principle (ResNet, Inception, etc.)

Cons:

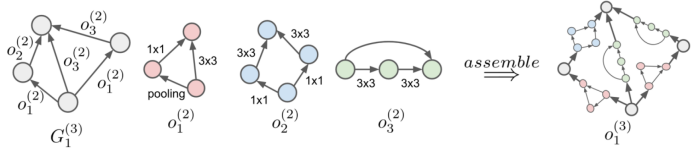
- Still need to (manually) determine the *macro* architecture, i.e., the way in which cells are connected.
- Limiting if different cells work better in different parts of the network
  - E.g., different spatial resolutions may favour different convolutions

Hierarchical representation of search space [Liu et al, 2017]

- Directed Acyclic Graph (DAG) representation of architectures
  - Each node is a latent representation; each edge is an operation/motif
- There are different levels of motivs
  - Level-1 primitives: standard operators; e.g., 3x3 conv, max pooling, ...
  - Level-2 motivs: combinations of level-1 primitives



  - Level-3 motivs: combinations of level-2 motivs



Pros and Cons of Hierarchical Search Space

What are some pros and cons of a hierarchical search space compared to the cell search space?

Please think about this for a few minutes before continuing.

Pros and Cons of Hierarchical Search Space

Pros:

- Flexibility of constructing building blocks and reusing them many times
  - like a cell search space
- Flexibility of using different building blocks in different parts of the network
  - like a basic search space
- Ability to reuse building blocks at various levels of abstraction
  - again, this pattern has been used in manual design, e.g., in Inception nets

Cons:

- Larger than cell search space
- Vastly more expressive than cell search space $\rightarrow$ potentially much harder to search

Questions to Answer for Yourself / Discuss with Friends

- Repetition:
  What are some pros and cons of the cell search space compared to the basic one?
- Repetition:
  Explain the way in which level-3 motivs in the hierarchical search space use level-2 motivs.
- Repetition:
  What are some pros and cons of the hierarchical search space compared to the other ones?

Lecture Overview

1. Overview
2. Search Spaces
3. Blackbox Optimization Methods
4. Speedup Techniques
5. DARTS: Differentiable Architecture Search
6. Practical Recommendations for NAS
7. Learning Goals & Further Reading

Foundations of Deep Learning, Winter Term 2021/22

Week 10: Neural Architecture Search (NAS)

Blackbox Optimization Methods

Frank Hutter Abhinav Valada

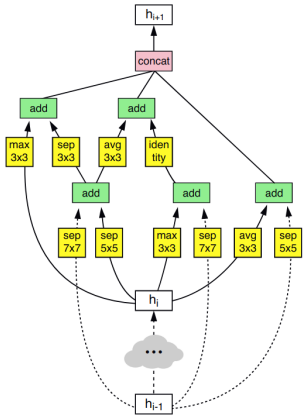
University of Freiburg

NAS as Hyperparameter Optimization

- NAS can be formulated as a HPO problem

- E.g., cell search space by [Zoph et al. 2018]
  has 5 categorical choices per block
  - 2 categorical choices of hidden states
    - For block N, the domain of these categorical variables is $\{h_i, h_{i-1}, \text{output of block } 1, ..., \text{output of block } N-1\}$
  - 2 categorical variables choosing between operations
  - 1 categorical variable choosing the combination method
  - Total number of hyperparameters for the cell:
    5B (with B=5 by default)

- In general: one may require conditional hyperparameters
  - E.g., chain-structured search space
    - Top-level hyperparameter: number of layers L
    - Hyperparameters of layer k conditional on L $\geq$ k



Reduction Cell

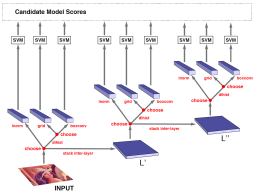
Early Work on Neuroevolution (already since the 1990s)

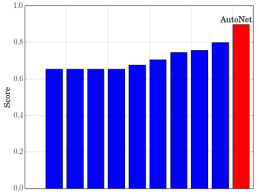
[Kitano 1990; Angeline et al. 1994; Stanley and Miikkulainen, 2002; Bayer et al, 2009; Floreano et al. 2008]

- Evolves architectures & often also their weights
- Typical approach:
  - Initialize a population of N random architectures
  - Sample N individuals from that population (with replacement) according to their fitness
  - Apply mutations to those N individuals to produce the next generation's population
  - Optionally: elitism to keep best individuals in the population
- Mutations include adding, changing or removing a layer

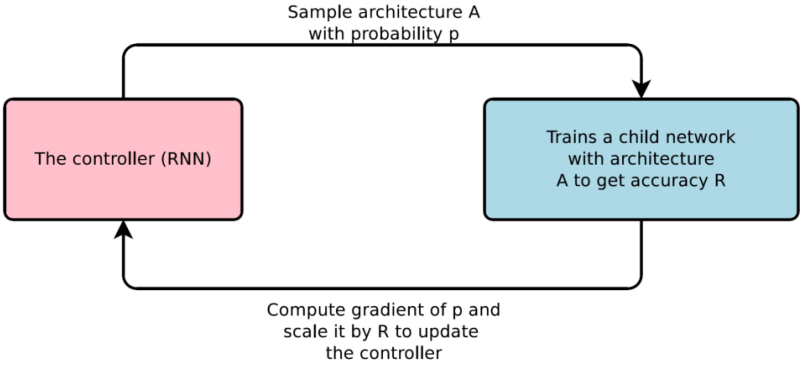
Early Work on Bayesian Optimization (since 2013)

- With TPE [Bergstra et al. 2011]:
  - Joint optimization of a vision architecture with 238 hyperparameters [Bergstra et al, 2013]
  - State-of-the-art performance on 3 disparate problems:
    - Face matching, face identification, and object recognition
- With SMAC [Hutter et al, 2011]:
  - New state-of-the-art performance on CIFAR-10 w/o data augmentation [Domhan et al. 2015]
  - Joint architecture and hyperparameter search, yielding Auto-Net [Mendoza et al. 2016]
  - In 2015, Auto-Net already had several successes in ML competitions
    - E.g., human action recognition:
      54491 data points, 5000 features, 18 classes
    - First automated deep learning (Auto-DL) method to win a machine learning competition dataset against human experts
- With Gaussian processes:
  - Arc kernel [Swersky et al, 2013]



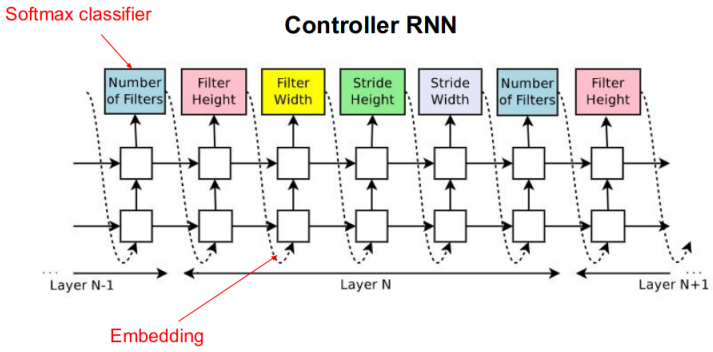


Reinforcement Learning [Zoph & Le, 2017]



- Use RNN ("Controller") to generate a NN architecture piece-by-piece
- Train this NN ("Child Network") and evaluate it on a validation set
- Use Reinforcement Learning (RL) to update the parameters of the Controller RNN to optimize the performance of the child models

Learning CNNs with RL [Zoph & Le, 2017]

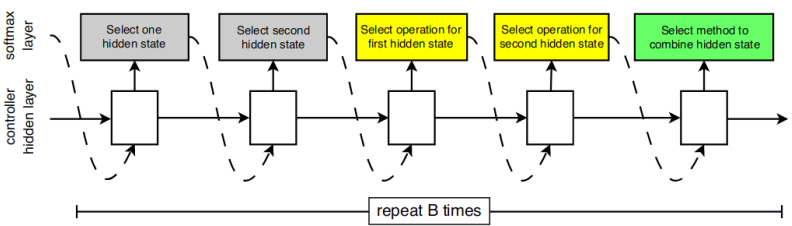


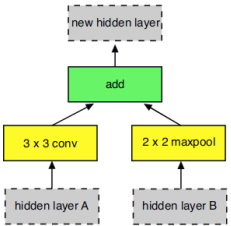
- For a fixed number of layers, select:
  - Filter width/height, stride width/height, number of filters
- Large computational demands (800 GPUs for 2 weeks, 12.800 architectures evaluated)
- State-of-the-art results for CIFAR-10 & Penn Treebank architecture

  → Brought NAS into the limelight

Learning CNN cells with RL [Zoph et al. 2018]

- 2 types of cells: normal and reduction cells
- For each type of cell: B blocks, each with 5 choices
  - Choose two previous feature maps (from this cell)
  - For each of these, choose an operation (3×3 conv, max-pool, etc.)
  - Choose a merge operation to combine the two results (concat or add)

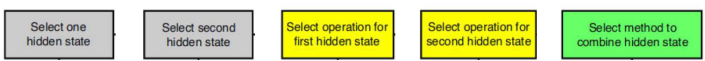




Learning CNN cells with evolution [Real et al, 2019]

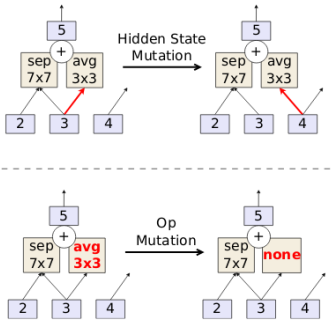
- 2 types of cells: normal and reduction cells
- For each type of cell: B blocks, each with 5 choices
  - Choose two previous feature maps (from this cell)
  - For each of these, choose an operation (3×3 conv, max-pool, etc.)
  - Choose a merge operation to combine the two results (concat or add)

- Evolution simply tackles this as a HPO problem with 2×5×B variables:

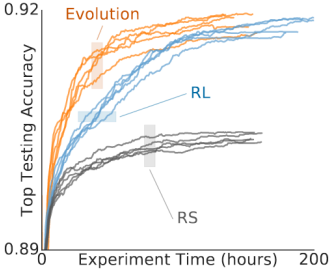


Regularized/Aging Evolution [Real et al, 2019]

- Quite standard evolutionary algorithm
  - But oldest solutions are dropped from population, instead of the worst
- Standard SGD for training weights (optimizing the same blackbox as RL)
- Same fixed-length (HPO) search space as used for RL [Zoph et al. 2018]



Different types of mutations in cell search space



State-of-the-art performance in apples-to-apples comparison
→ AmoebaNet

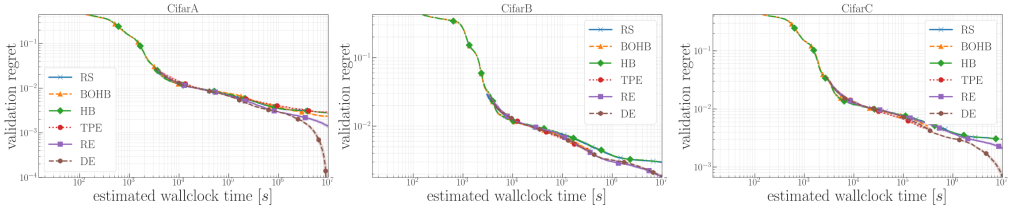
Bayesian Optimization (BO)

- Encode the architecture space by categorical hyperparameters (like regularized evolution)
- Strong performance with tree-based models
  - TPE [Bergstra et al. 2013]
  - SMAC [Domhan et al. 2015, Mendoza et al. 2016, Zela et al. 2018]
- Kernels for GP-based NAS
  - Arc kernel [Swersky et al, 2013]
  - NASBOT [Kandasamy et al, 2018]
- There are also several recent promising BO approaches based on neural networks
  - BANANAS [White et al. 2020]
- BO is very competitive, has been shown to outperform RL [Ying et al, 2019]

Current State of the Art: Differential Evolution

- Comprehensive experiments on a wide range of 12 different NAS benchmarks
  [Awad, Mallik and Hutter, AutoML 2020]

- Results:
  - Regularized evolution is very robust, typically amongst best of the methods discussed so far
  - Evolution variant of differential evolution is yet better; most efficient and robust method



Questions to Answer for Yourself / Discuss with Friends

- Repetition:
  What are some pros and cons of using black-box optimizers for NAS?
- Repetition:
  How can NAS be modelled as a HPO problem?
- Discussion:
  Given enough resources, will blackbox NAS approaches always improve performance?
- Discussion:
  Why does discarding the oldest individual (rather than the worst) help in regularized/aging evolution?
- Transfer:
  How would you write NAS with the hierarchical search space as a HPO problem?

Lecture Overview

1. Overview
2. Search Spaces
3. Blackbox Optimization Methods
4. Speedup Techniques
5. DARTS: Differentiable Architecture Search
6. Practical Recommendations for NAS
7. Learning Goals & Further Reading

Foundations of Deep Learning, Winter Term 2021/22

Week 10: Neural Architecture Search (NAS)

Speedup Techniques

Frank Hutter Abhinav Valada

University of Freiburg

Overview of NAS Speedup Methods

- Multi-fidelity optimization
- Learning curve prediction
- Meta-learning across datasets
- Network morphisms & weight inheritance
- Weight sharing & the one-shot model

NAS Speedup Technique 1: Multi-fidelity optimization

- Analogous to multi-fidelity optimization in HPO
  - Many evaluations for cheaper fidelities (less epochs, smaller datasets, down-sampled images, shallower networks, etc)
  - Fewer evaluations necessary for more expensive fidelities
- Compatible with any blackbox optimization method
  - Using random search: ASHA [Li & Talwalkar, 2019]
  - Using Bayesian optimization: BOHB [Zela et al, 2018]
  - Using differential evolution: DEHB [Awad et al, under review]
  - Using regularized evolution: progressive dynamic hurdles [So et al, 2019]
- Often used for joint optimization of architecture & hyperparameters
  - Auto-Pytorch [Mendoza et al, 2019] [Zimmer et al, under review]
  - "Auto-RL" [Runge et al, 2019]

NAS Speedup Technique 2: Learning Curve Prediction

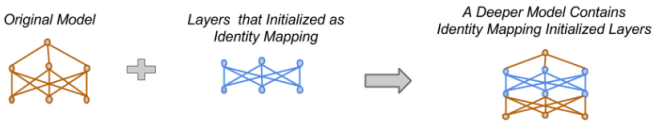
- Analogous to learning curve prediction in HPO
  - Observe initial learning curve and predict performance at the end
  - Can use features of the architecture as input (just like hyperparameters as inputs)
- Often used for joint optimization of architecture & hyperparameters
- Compatible with any blackbox optimization method
  - Using random search and Bayesian optimization: [Domhan et al, 2015]
  - Using reinforcement learning: [Baker et al, 2018]

NAS Speedup Technique 3: Meta-Learning

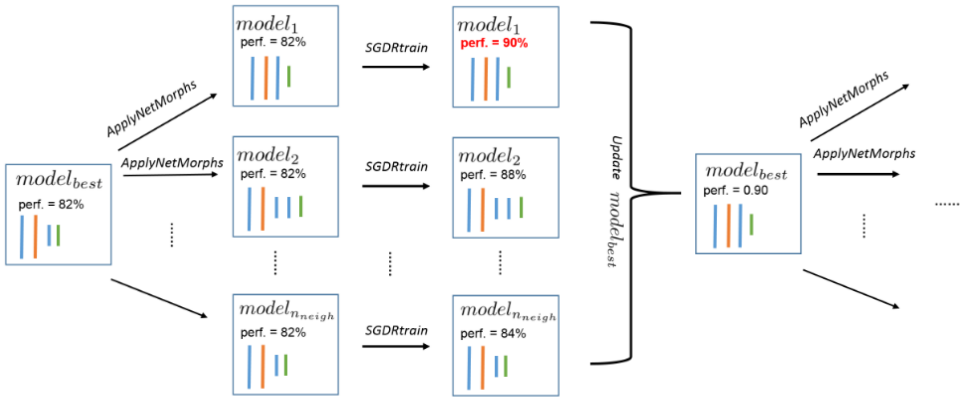
- Lots of work on meta-learning for HPO
- Only little work on meta-learning for NAS
  - Find a set of good architectures to initialize BOHB in Auto-Pytorch [Zimmer et al, under review]
  - Learn RL agent's policy network on previous datasets [Wong et al, 2018]
  - Learn neural architecture that can be quickly adapted [Lian et al, 2019; Elsken et al, 2019]

NAS Speedup Technique 4: Network Morphisms

- Network Morphisms [Chen et al. '16; Wei et al. '16; Cai et al. '17]
  - Change the network structure, but not the modelled function
  - I.e., for every input the network yields the same output as before applying the network morphisms operations
  - Examples: "Net2DeeperNet", "Net2WiderNet", etc.



Network Morphisms Allow Efficient Moves in Architecture Space

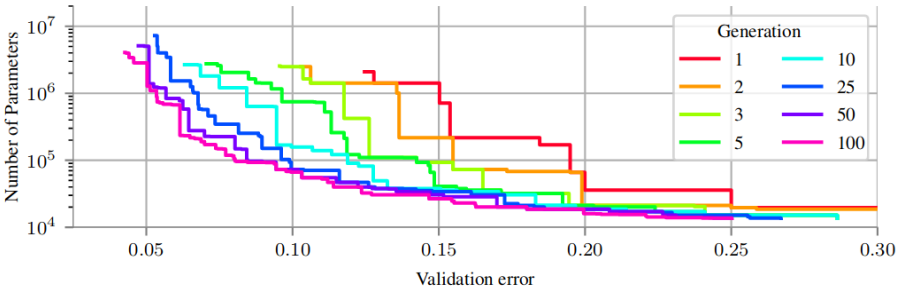


Weight inheritance avoids expensive retraining from scratch

[Real et al, 2017, Cai et al, 2018, Elsken et al, 2017, Cortes et al, 2017, Cai et al, 2018, Elsken et al. '19]

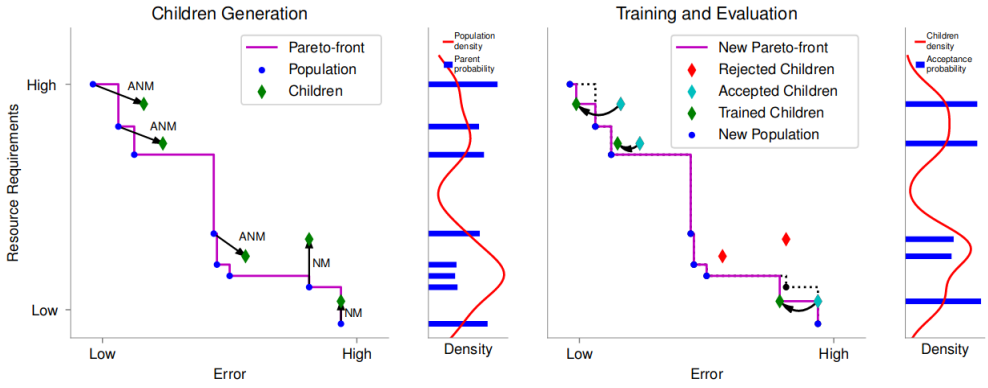
Network Morphisms for Multi-objective NAS [Elsken et al. '19]

- To trade off error vs. resource consumption (e.g, #parameters):
  - Maintain a Pareto front of the two objectives
  - Evolve a population of Pareto-optimal architectures over time



Network Morphisms for Multi-objective NAS [Elsken et al. '19]

- LEMONADE: Lamarckian Evolution for Multi-Objective Neural Architecture Design
- Weight inheritance through approximate morphisms (ANMs)
  - Dropping layers, dropping units within a layer, etc (function not preserved perfectly)

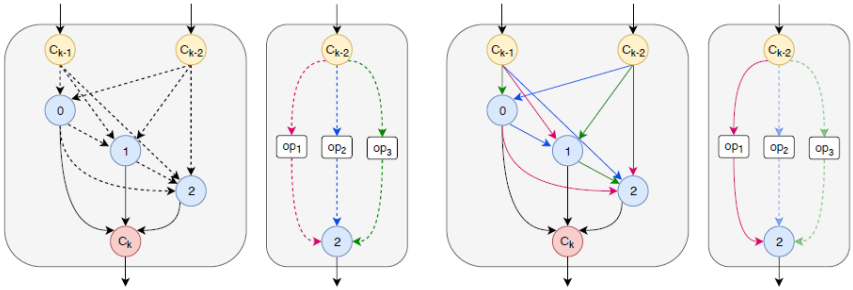


NAS Speedup Technique 5: Weight Sharing and One-shot Models [Pham et al, 2018; Bender et al, 2018]

- All possible architectures are subgraphs of a large supergraph: the one-shot model
- Weights are shared between different architectures with common edges in the supergraph
- Search costs are reduced drastically since one only has to train a single model (the one-shot model).

NAS Speedup Technique 5: Weight Sharing and One-shot Models [Pham et al, 2018; Bender et al, 2018]

- The one-shot model can be seen as a directed acyclic multigraph
  - ⇒ Nodes - latent representations.
  - ⇒ Edges (dashed) - operations.



(a) One-shot search **(b)** Final evaluation

- Architecture optimization problem: Find optimal path from the input to the output

Questions to Answer for Yourself / Discuss with Friends

- Repetition:
  List five methods to speed up NAS over blackbox approaches
- Repetition:
  Which speedup techniques directly carry over from HPO to NAS?
- Discussion:
  Why do network morphisms and the one-shot model only apply to NAS, and not to HPO?

Lecture Overview

1. Overview
2. Search Spaces
3. Blackbox Optimization Methods
4. Speedup Techniques
5. DARTS: Differentiable Architecture Search
6. Practical Recommendations for NAS
7. Learning Goals & Further Reading

Foundations of Deep Learning, Winter Term 2021/22

Week 10: Neural Architecture Search (NAS)

DARTS: Differentiable Architecture Search

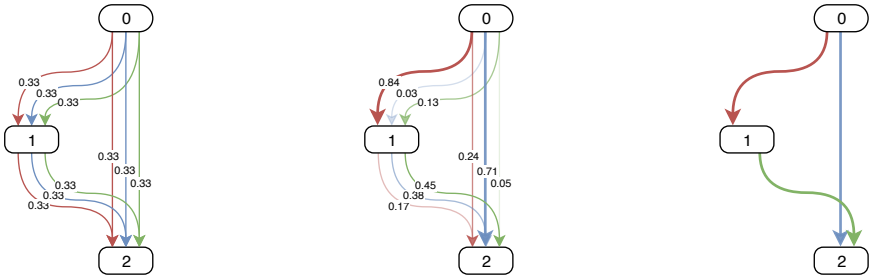
Frank Hutter Abhinav Valada

University of Freiburg

DARTS: Differentiable Architecture Search [Liu et al, 2018]

- Use one-shot model with continuous architecture weight α for each operator

$$x^{(j)} = \sum_{i<j} \tilde{o}^{(i,j)}(x^{(i)}) = \sum_{i<j} \sum_{o \in \mathcal{O}} \frac{exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} exp(\alpha_{o'}^{(i,j)})} o(x^{(i)})$$



(a) Initialization (b) Search end (c) Final cell

- By optimizing the architecture weights α, DARTS assigns importance to each operation
  - Since the α are continuous, we can optimize them with gradient descent
- In the end, DARTS discretizes to obtain a single architecture (c)

DARTS: Architecture Optimization

- The optimization problem (a $\rightarrow$ b) is a bi-level optimization problem:

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

- This is solved using alternating SGD steps on architectural parameters α and weights w

---

Algorithm: DARTS 1st order

---

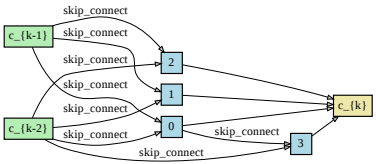
while *not converged* **do**
  Update one-shot weights $\boldsymbol{w}$ by $\nabla_{\boldsymbol{w}} \mathcal{L}_{train}(\boldsymbol{w}, \boldsymbol{\alpha})$
  Update architectural parameters α by $\nabla_{\alpha} \mathcal{L}_{valid}(\boldsymbol{w}, \boldsymbol{\alpha})$

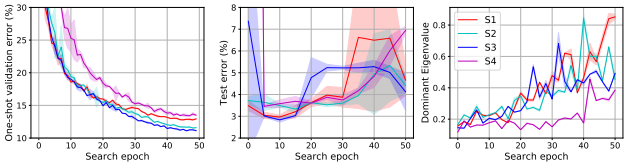
return $\arg\max_{o \in \mathcal{O}} \alpha_o^{(i,j)}$ for each edge (i, j)

- Note: there is no theory showing that this process converges

Issues – Non-robust behaviour

- DARTS is very sensitive w.r.t. its own hyperparameters (e.g. one-shot learning rate, L_2 regularization, etc.)
  - Tuning these hyperparameters for every new task/search space is computationally expensive
  - DARTS may return degenerate architectures, e.g., cells composed with only skip connections





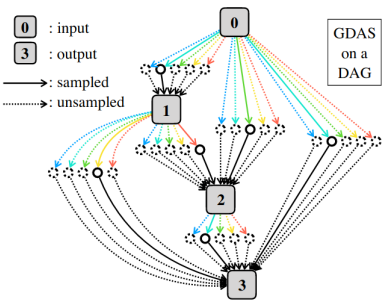
- RobustDARTS [Zela et al, 2020] – tracks the curvature of the validation loss and stops the search early based on that
- SmoothDARTS [Chen and Hsieh, 2020] – applies random perturbation and adversarial training to avoid bad regions

Issues – Memory constraints

- DARTS keeps the entire one-shot model in memory, together with its computed tensors
  - This constrains the search space size and the fidelity used to train the one-shot model
  - Impossible to run on large datasets as ImageNet

A lot of research aims to fix this issue:

- GDAS [Dong et al, 2019] – samples from a Gumbel Softmax distribution to keep only a single architecture in memory
- ProxylessNAS [Cai et al, 2019] – computes approximate gradients on α keeping only 2 edges between two nodes in memory at a time
- PC-DARTS [Xu et al, 2020] – performs the search on a subset of the channels in the one-shot model



Questions to Answer for Yourself / Discuss with Friends

- Repetition:
  What is the main difference between DARTS and the other one-shot NAS methods we saw before?

- Repetition:
  How does DARTS optimize the architectural weights and one-shot weights?

- Repetition:
  What are DARTS' main issues and how can they be fixed?

- Discussion:
  RobustDARTS stops the architecture optimization early, before the curvature of the validation loss becomes high. Why do you think this works?
  [Hint: think about the discretization step after the DARTS search.]

Lecture Overview

1. Overview
2. Search Spaces
3. Blackbox Optimization Methods
4. Speedup Techniques
5. DARTS: Differentiable Architecture Search
6. Practical Recommendations for NAS
7. Learning Goals & Further Reading

Foundations of Deep Learning, Winter Term 2021/22

Week 10: Neural Architecture Search (NAS)

Practical Recommendations for NAS

Frank Hutter Abhinav Valada

University of Freiburg

Maturity of the Fields of NAS and HPO

- Hyperparameter optimization is a mature field
  - Blackbox HPO has been researched for decades; there are many software packages
  - Multi-fidelity HPO has also become quite mature
- Neural architecture search is still a very young field
  - Blackbox is quite mature, but slow
  - Multi-fidelity NAS is also quite mature and faster
  - Meta-learning + multi-fidelity NAS is fast, but is still a very young field
  - Gradient-based NAS is fast, but can have failure modes with terrible performance
  - Gradient-based NAS hasn't reached the hands-off AutoML stage yet
- NAS is mainly used to create new architectures that many others can reuse
- Given a new dataset, HPO is crucial for good performance; NAS may not be necessary
  - The biggest gains typically come from tuning key hyperparameters (learning rate, etc)
  - Reusing a previous achitecture often yields competitive results

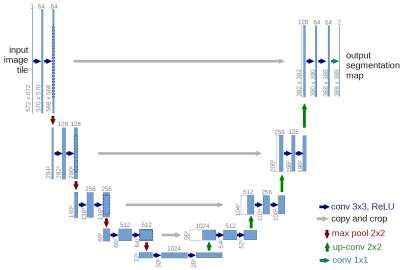
Practical Recommendations for NAS and HPO

- Recommendations for a new dataset
  - → Always run HPO
  - → Try NAS if you can

- How to combine NAS & HPO
  - If the compute budget suffices, optimize them jointly, e.g., using BOHB
    - + Auto-PyTorch Tabular [Zimmer, Lindauer & Hutter, 2020]
    - + Auto-RL [Runge et al, 2019]
  - Else
    - + If you have decent hyperparameters:
      run NAS, followed by HPO for fine-tuning [Saikat et al, 2019]
    - + If you don't have decent hyperparameters: first run HPO to get competitive

Case Study: NAS & HPO in Auto-DispNet

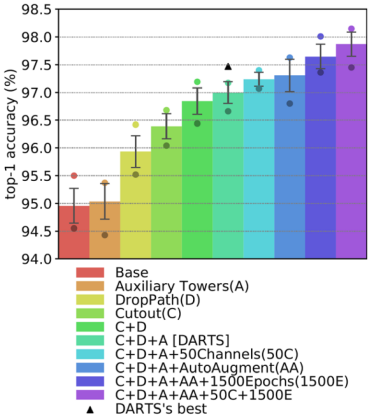
- Problem: disparity estimation
  - Estimate depth from stereo images
- Background: U-Net
  - Skip connections from similar spatial resolution to avoid loosing information
- Search space for DARTS
  - 3 cells: keeping spatial resolution, downsampling, and new upsampling cell that supports U-Net like skip connections
- Both NAS and HPO improved the state of the art [Saikat et al, 2019]:
  - End-point-error (EPE) on Sintel dataset: 2.36 $\rightarrow$ 2.14 (by DARTS)
  - Subsequent HPO: 2.14 $\rightarrow$ 1.94 (by BOHB)



Issues in NAS Research & Evaluations

- Most NAS methods are extremely difficult to reproduce and compare [Li & Talwalkar, 2019]
- For benchmarks used in almost all NAS papers:
  - Training pipeline matters much more than neural architecture
- The final benchmark results reported in different papers are typically incomparable
  - Different training code (often unavailable)
  - Different search spaces
  - Different evaluation schemes

→ We emphasize concepts, not published performance numbers



[Yang et al., ICLR 2020]

Questions to Answer for Yourself / Discuss with Friends

- Repetition:
  If you want to use both HPO and NAS for your problem, how could you proceed?
- Discussion:
  Think of a problem of your particular interest. For that problem, which approach would you use to combine HPO and NAS, and why?

Lecture Overview

1. Overview
2. Search Spaces
3. Blackbox Optimization Methods
4. Speedup Techniques
5. DARTS: Differentiable Architecture Search
6. Practical Recommendations for NAS
7. Learning Goals & Further Reading

Foundations of Deep Learning, Winter Term 2021/22

Week 10: Neural Architecture Search (NAS)

Learning Goals & Further Reading

Frank Hutter Abhinav Valada

University of Freiburg

Summary by Learning Goals

Having heard this lecture, you can now ...

- Define and motivate the neural architecture search problem
- Describe different NAS search spaces and discuss their pros and cons
- Describe blackbox optimization methods for NAS
- Discuss various speedup techniques for NAS
- Explain the DARTS algorithm and discuss its failure modes
- Discuss practical recommendations for NAS and HPO

Further reading

Read the NAS survey [Elsken, Metzen and Hutter, 2019], which is the main source for this week's material.