Neural Architecture Search: A Survey

2. Search Space

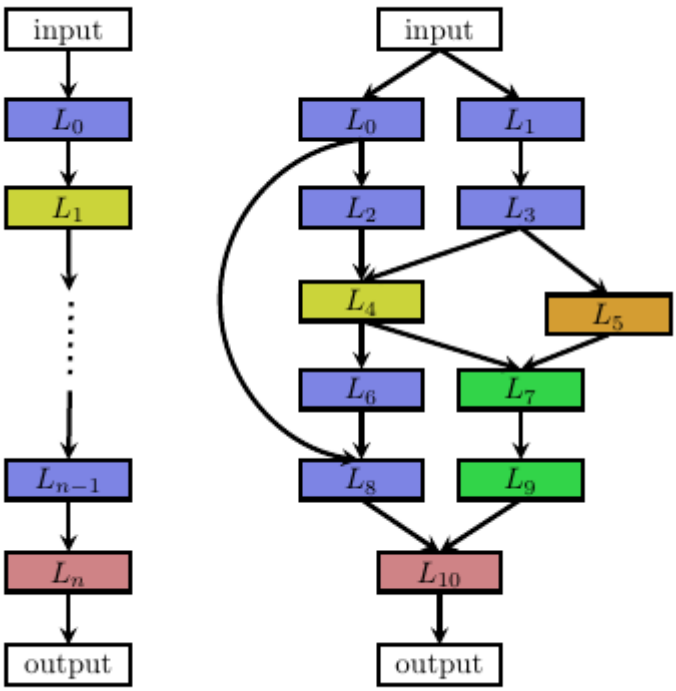
2.1 Chain-structured neural networks

1) Definition

A chain-structured neural network architecture A can be written as a sequence of n layers, as showed in below:

left : simple chain-structures nn.

right : Multi-branch architectures which will be introduced in 3)



2) Three parameters

- The (maximum) number of layers n

- The type of operation every layer can execute , e.g., pooling, convolution, or more advanced layer types like depthwise separable convolutions.
- Hyperparameters associated with the operation, e.g., number of filters, kernel size and strides for a convolutional layer.

2.2 Multi-branch architectures

Take more complex structures into consideration.

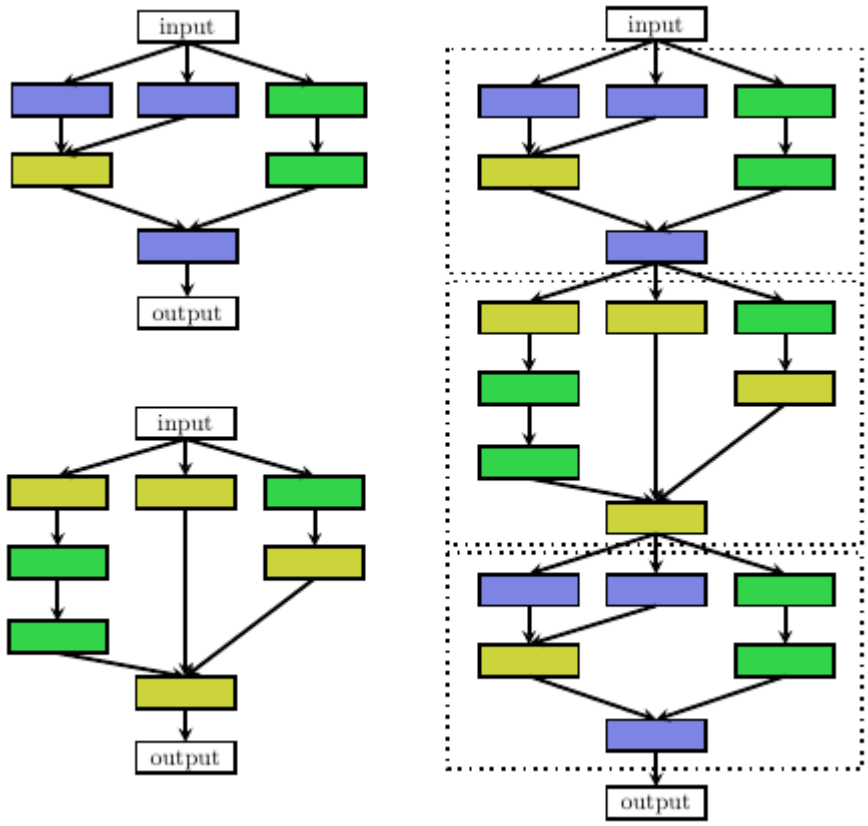
- The chain-structured networks
- Residual Networks
- DenseNets

2.3 Cell-based search space

1. Definition

- Two kinds of basic cells or blocks
    - Normal cell : preservers the dimensionality of the input
    - Reduction cell : reduces the spatial dimension.
- The final architecture is then built by **stacking these cells in a predefined manner**

2. Show in graph



3. Two adventages

- The size of the search space is drastically reduced since cells can be comparably small.
- Cells can more easily be transferred to other datasets by adapting the number of cells used within a model. ?

4. My Problem

- How to define a cell?

  Maybe as a part of prior-knowledge.

2.4 Meta-architecture

1) Problems

- How many cells shall be used and how should they be connected to build the actual model?
- Take the cell as layer and do something like we showed in 2.2. Cells can be combined arbitrarily.

2) Methods

- **Hierarchical search space - three levels**
  - First level : consists of the set of primitive operations.

    > **Primitive operations** are basic computations performed by an algorithm. Examples are evaluating an expression, assigning a value to a variable, indexing into an array, calling a method, returning from a method, etc. They are easily identifiable in pseudocode and largely independent from the programming language

  - Second level : **Different motifs(cells)** that connect primitive operations via a direct acyclic graphs
  - Third level : the third level of motifs that encode how to connect second-level motifs.

> Can be anological to cell-based search space which:
>
> - THe second layer corresponds to cells.
> - The third level is the hard-coded meta-architecture.

2.5 Problem in space searching

Even for the simplest problem in settings

- Non-continuous (The difference of models is not differentiable)
- Relatively high-dimensional (exponentional)

3. Search Strategy

3.1 Introduction

- **Serval methods:**

- Random search(RS)
- Bayesian optimization
- Evolutionary methods (used in decades ago)
- RL
- Gradient-based methods

3.2 History

- **Evolutionary methods** (used in decades ago)
- **Bayesian optimization** celebrated several early successes in NAS since 2013,
- NAS **became a mainstream research topic** in the machine learning community after Zoph and Le (2017)

  > Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In International Conference on Learning Representations, 2017.
  >
  > Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. In Conference on Computer Vision and Pattern Recognition, 2018.

  But it's extremely heavy : Zoph and Le (2017) use vast computational resources to achieve this result (800 GPUs for three to four weeks)
- After their work, a wide variety of methods have been published in quick succession to **reduce the computational costs** and achieve further improvements in performance.

3.3 RL methods

1) Definition

- **Generation of a neural architecture** $\rightarrow$ **agent's action**
- **Action space** = **Search space**

- Different RL approaches **differ in** how they represent **the agent's policy** and how they optimize it

2) Encoder of netork structure

- **Encode the neural architecture** with **string**

> Because the encoding of network could change, they use the sequential network to re-encode the network.

3) optimize

- **RNN**

  Then use a recurrent neural network (RNN) to optimize with **REINFORCE policy gradient algorithm**
- **Proximal Policy Optimization (PPO)**
- **Q-learning**

  Baker use Q-learning to train a policy which sequentially chooses a layer's type and corresponding hyperparameters.
- **Sequential decision processes - 1**.
  - **the actions** is generate the architecture sequentially
  - **the policy** definiate how to samples actions.
  - **the environment's "state"** contains a summary of the actions sampled so far
  - **the reward** is obtained only after the final action

> Have problem : There are no intermediate rewards

- **Sequential decision processes - 2**.
  - **the action** corresponds to an application of function-preserving mutations(突然変異), dubbed network morphisms(ネットワーク変形)
  - **the policy** definiate how to samples actions.
  - **the state** is the current (partially trained) architecture
  - **the reward** is an estimate of the architecture's performance

> Advantage compared to prior:

> Can interpret the architecture sampling process as the sequential generation of a single action.

- How to learn it?
  - We have a string to represent the neural network, but it's changable. So they use a **BiLSTM** to turn it into a fixed string.
  - And we have actions set to change the network encoding.
  - The **combination of these two components** constitute the **policy**, which is **trained end-to-end** with the **REINFORCE policy gradient algorithm**.

3.4 Neuro-evolutionary approaches

1) Definition of evolutionary methods

- Evolutionary algorithms evolve a set of models (here it's be the networks) .
- In every evolution step, **at least one model** from the set is **sampled** and **serves as a parent** to **generate offsprings** by applying mutations to it.

> Here mutations are local operations.
>
> such as **adding or removing a layer**, **altering the hyperparameters of a layer**, adding **skip connections**, as well as **altering training hyperparameters**.

- Neuro-evolutionary methods differ in
  - how they sample
  - update populations and generate offsprings.

2) Methods - sample parents

- Real(2017) : Use **tournament selection** to sample parents
- Elsken(2018) : sample parents **from a multi-objective Pareto front** using an inverse density.
- Real(2017) : remove the worst individual from a population

- Real(2018) : found it beneficial to remove the oldest individual (which decreases greediness(貪欲))
- Liu(2018b) : do not remove individuals at all

3) Methods - generate offspring

- Most methods initialize child networks randomly.
- Elsken(2018) : employ Lamarckian inheritance.

  > **Knowledge** (in the form of **learned weights**) is **passed on from a parent network** to its **children** by using **network morphisms**.

- Real(2017) : let an offspring inherit all parameters of its parent that are not affected by the applied mutation.

  > Can speed up learning compared to a random initialization

4) Comparision with RL and RS(random search)

- Perform equally well in terms of final test accuracy
- Having better anytime performance and finding smaller models
- RS has just a little margin with these two.

3.5 Bayesian Optimization

1). History and definition

- It's a popilar method for hyperparameter optimization.
- But it's not useful in NAS **before** because :

  typical BO are based on Gaussian processes and **focus on low-dimensional continuous optimization problems**

2) Methods

Based on Gaussian processes

- Derive **kernel functions** for architecture search spaces in order to use classic GP-based
  BO methods.
- But so far without achieving new state-of-the-art performance.

> Raiders of the lost architecture: Kernels for bayesian optimization in conditional parameter spaces. In NIPS 2013
>
> Neural Architecture Search with Bayesian Optimisation and Optimal Transport .arXiv:1802.07191, February 2018.

Treed-based methods

- Treed Parzen estimators or random forests
- Can effectively search very high-dimensional conditional spaces
- Can achieve state-of-the-art performance on a wide range of problems

3) Comparision with evolutionary algorithm

- Not a full comparison
- Can also outperform evolutionary algorithm

3.6 Other methods

1) Hierarchical manner

- **Sequential model-based optimization**

> Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical Representations for Efficient Architecture Search. In Interna- tional Conference on Learning Representations , 2018b

- **Monte Carlo Tree Search**

> Negrinho and G. Gordon. DeepArchitect: Automatically Designing and Training Deep Architectures. arXiv:1704.08792 , 2017.

- **Hill climbing**

  Discovers high-quality architectures by greedily moving in the direction of better performing architectures

2) Gradient-based optimization

> Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In arXiv:1806.09055 , 2018c

4. Performance Estimation Strategy

4.1 Definition

Methods to estimate the performance of a given architecture.

> Architecture : A

4.2 Simplest methods

Train A on training data and evaluate its performance on validation data.

Weakness : **Computational demands** in the order of thousands of GPU days for NAS

4.3 Low-fidelity approximations

1) For example :

- Train on a subset of the data
- lower-resolution images
- less filters per layer

2) Weakness:

- **Introduce bias** in the estimate as performance will typically be underestimated

- Recent results indicate that **this relative ranking** can **change dramatically** when the difference between the **cheap approximations**

4.4 Curve extrapolation(曲線外挿)

Predicion-based

1) Definition

- **Definition - What kind of curve?**
    - Propose to extrapolate initial **learning curves**

2) Methods - 1

Terminate those predicted to **perform poorly**

> to speed up the architecture search process.
>
> Domhan(2015)

3) Methods - 2

architectural hyperparameters for predicting which partial learning curves are most promising.

4.5 Reuse of parameter

1) One example

- **Initialize the weights** of novel architectures based on weights of other architectures that **have been trained** before
- dubbed network morphisms can do this.

2) Problems:

Strict network morphisms can only **make architectures larger** and may thus lead to overly complex architectures.

4.6 One-Shot Architecture Search

1) Insight

- Treats all architectures as **different subgraphs** of a **supergraph**
- **Shares weights** between architectures that have edges of this supergraph in common

2) Methods

- Only the weights of **a single one-shot model** need to be trained.

Algorithm 1 SMASH

input Space of all candidate architectures, $\mathbb{R}_c$
  Initialize HyperNet weights H
  loop
    Sample input minibatch x_i, random architecture c and architecture weights $W = H(c)$
    Get training error $E_t = f_c(W, x_i) = f_c(H(c), x_i)$, backprop and update H
  end loop
  loop
    Sample random c and evaluate error on validation set $E_v = f_c(H(c), x_v)$
  end loop
  Fix architecture and train normally with freely-varying weights W

- Then candidate model can be evaluated **without any separate training**.

> greatly speeds up performance estimation, since no training is required

3) Weakness and Strength

- Weakness :
  - This approach typically **incurs a large bias** as it **underestimates the actual performance** of architectures severely.
  - The supergraph defined **a-priori restricts** the **search space** to its subgraphs.
  -
- Strenght :

It allows **ranking architectures reliably**, since the **estimated performance correlates strongly** with the actual performance

4) Sub-methods

- Differ in how the one-shot model is trained