Neural Architecture Search with Network Morphisms and Successive Halving

Friday, December 14, 2018

Machine Learning Lab

Miray Yüce Olesya Tsapenko

University of Freiburg

Overview

Overview

Neural Network Architecture

Manual architecture design

- Suboptimal architectures
- Time consuming
- Expensive

Neural Network Architecture

Manual architecture design

- Suboptimal architectures
- Time consuming
- Expensive

Solution?

Neural Network Architecture

Manual architecture design

- Suboptimal architectures
- Time consuming
- Expensive

Solution?

Neural Architecture Search

Neural Architecture Search

Characteristics of neural architecture search methods: [Elsken et al., 2018b]

- Search Space
- Search Strategy
- Performance Estimation Strategy

Neural Architecture Search

Characteristics of neural architecture search methods: [Elsken et al., 2018b]

- Search Space
- Search Strategy
- Performance Estimation Strategy

Approaches:

- Evolutionary methods

  [Stanley and Miikkulainen, 2002, Real et al., 2017, Real et al., 2018, Elsken et al., 2018a]
- Reinforcement learning algorithms

  [Zoph and Le, 2017, Pham et al., 2018, Zoph et al., 2018, Baker et al., 2017]
- And many others

Our work: an extension of NASH

In this work we extend Neural Architecture Search by Hill Climbing (NASH) [Elsken et al., 2018a].

Our work: an extension of NASH

In this work we extend Neural Architecture Search by Hill Climbing (NASH) [Elsken et al., 2018a].

NASH:

- Evolutionary based method
- Generates new networks and estimates their performances
- Selects the best architecture

Our work: an extension of NASH

- Can we have better test accuracy?
- Can we find smaller models?

Our work: an extension of NASH

- Can we have better test accuracy?
- Can we find smaller models?

GOAL: Achieve better classification results on CIFAR 10 with smaller models

Our work: an extension of NASH

- Can we have better test accuracy?
- Can we find smaller models?

GOAL: Achieve better classification results on CIFAR 10 with smaller models

What we did:

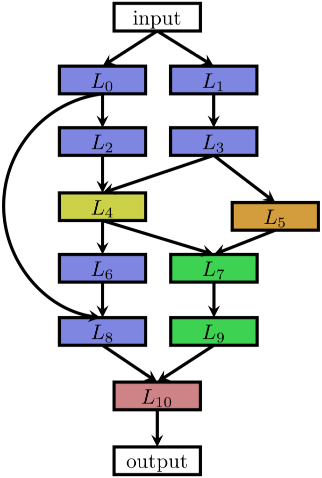
- Reimplement NASH
- Change network selection method
- Enlarge search space with depthwise separable convolutional layers
- Test different learning rate schedulers

Overview

General view

Characteristics of our method:

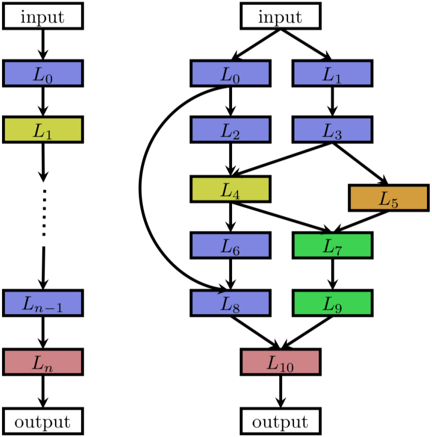
- Search Space
  - Multi branch CNN
- Search Strategy
  - Evolutionary method using Network Morphisms
- Performance Estimation Strategy
  - Successive halving and Network Morphisms



[Elsken et al., 2018b]

Network Morphisms

The concept of **transforming a network to a new one** with complete knowledge of the parent network



[Elsken et al., 2018b]

Network Operators

Fact: Every combination of network morphisms results to a network morphism again

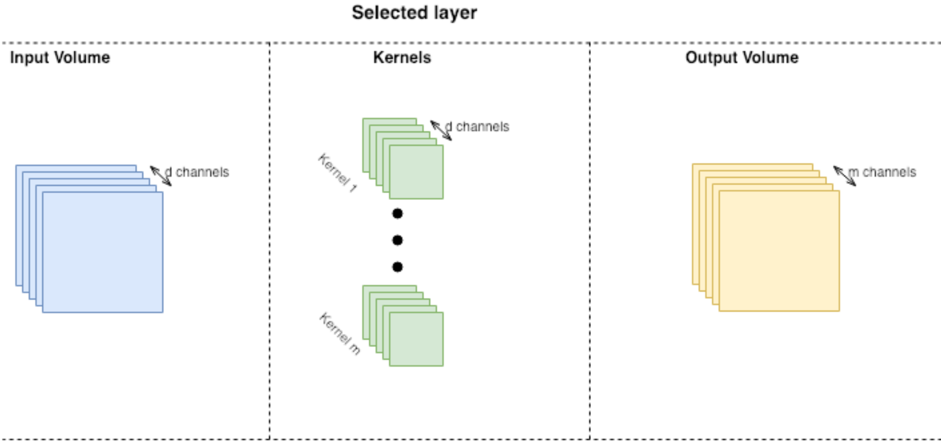
Network morphism operators we implemented:

- Alter Channels
- Insert Convolutional Layer
- Merge by Convex Combination
- Merge by Concatenation
- Split Up Convolutional Layer

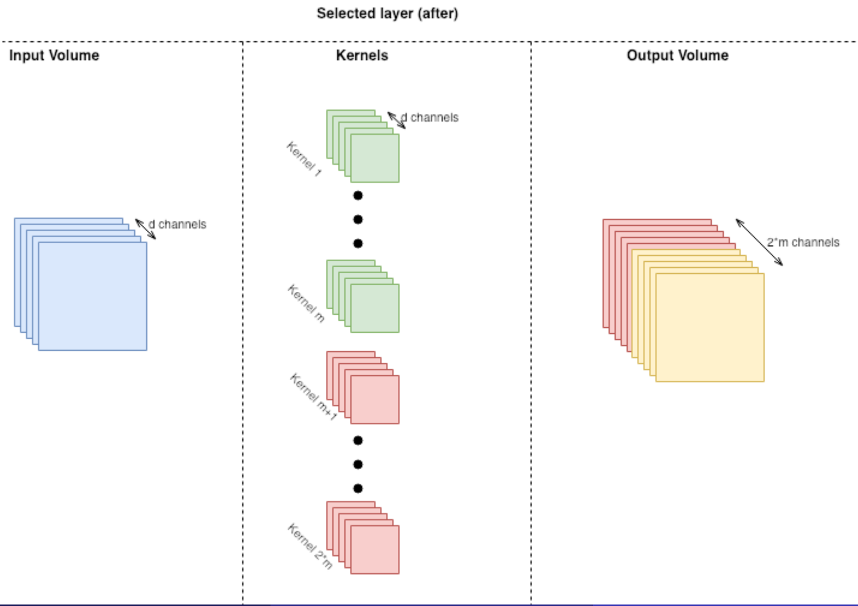
Network morphism operator we constructed:

- Insert Separable Convolutional Layer

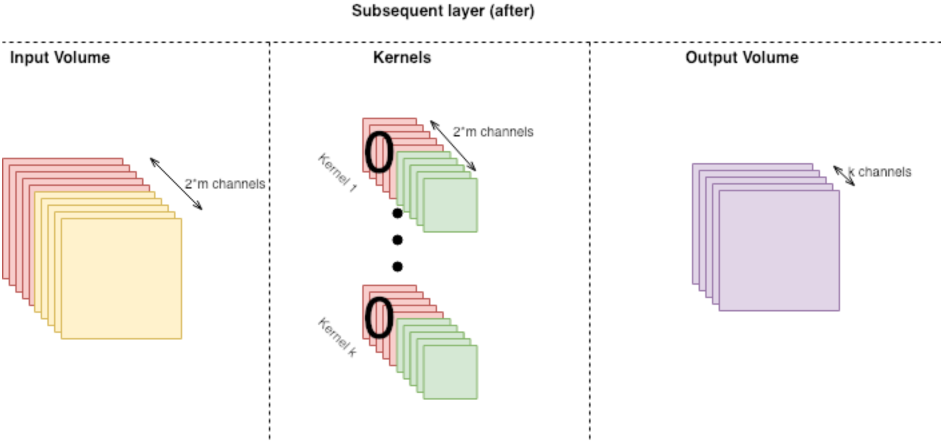
Network Operators: Alter Channels (1)



Network Operators: Alter Channels (2)



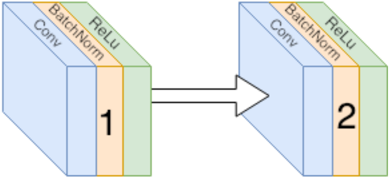
Network Operators: Alter Channels (3)



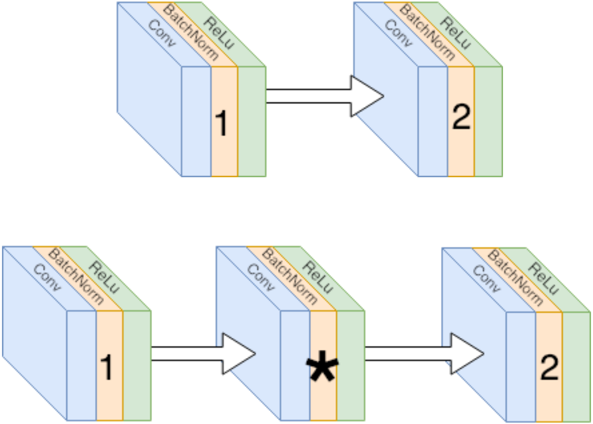
Network Operators: Insert Convolutional Layer (1)

- Insert a block of $Conv - BatchNorm - ReLu$ layers

- Insert a block of $Conv - BatchNorm - ReLu$ layers



- Insert a block of $Conv - BatchNorm - ReLu$ layers



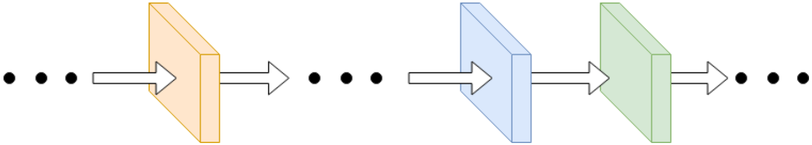
Network Operators: Insert Convolutional Layer (2)

- Initializing convolutional layer weights with identity mapping
- Initializing *Batch Normalization* layer's parameters to keep $y = x$:

$$y = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}} * \gamma + \beta \tag{1}$$

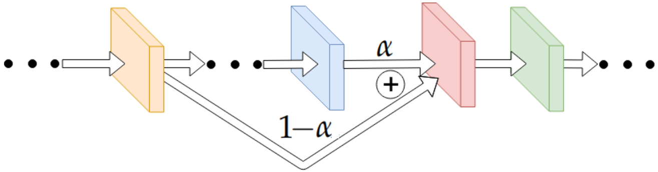
Network Operators: Merge by Convex Combination (1)

- Only for $ReLu$ or $MaxPool2d$ layers
- Dimensions (C, H, W) of layers must be the same



Network Operators: Merge by Convex Combination (2)

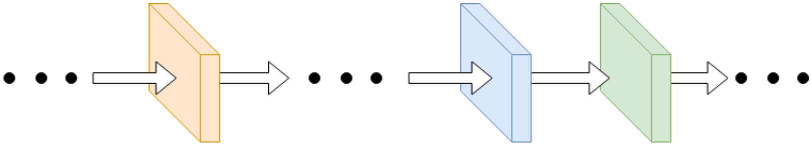
- α is a trainable parameter. Initialized as 1



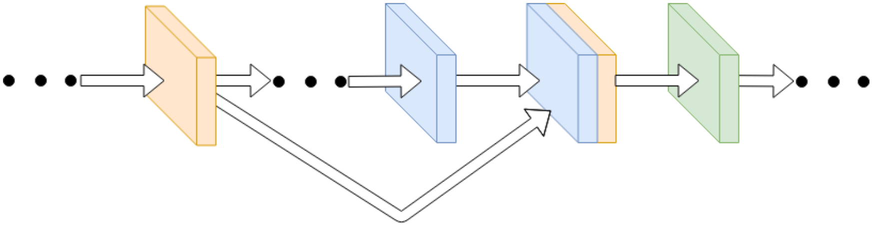
$$red = \alpha \cdot blue + (1 - \alpha) \cdot orange \tag{2}$$

Network Operators: Merge by Concatenation (1)

- Only for *ReLu* or *MaxPool2d* layers
- Numbers of channels can be different for layers to be concatenated



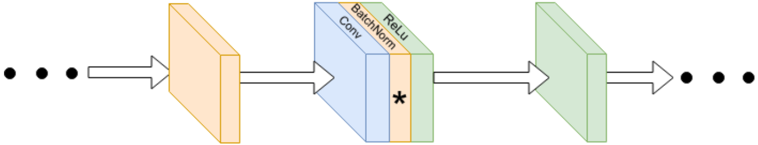
Network Operators: Merge by Concatenation (2)



- Green layer has more input channels now!

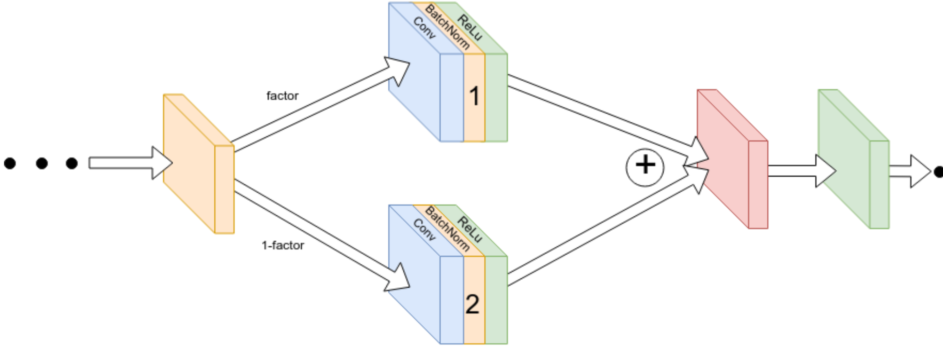
Network Operators: Split Up Convolutional Layer (1)

- $Conv - BatchNorm - ReLu$ blocks are splitted into two



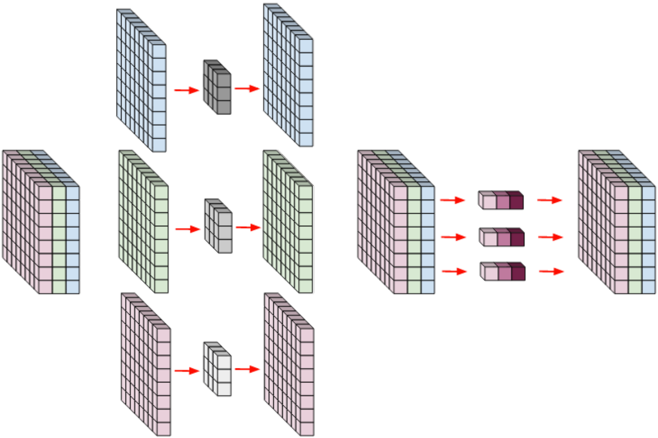
Network Operators: Split Up Convolutional Layer (2)

- Constant splitting factor: $factor$ and $1 - factor$



Network Operators: Insert Separable Convolutional Layer

- Insert a block of $SepConv - BatchNorm - ReLu$ layers

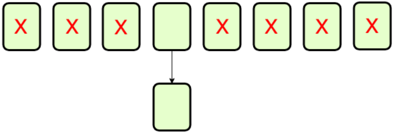


https://bit.ly/2RY9Zpg

Overview

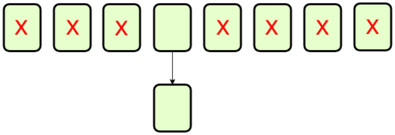
Child Network Selection

Hill Climbing:

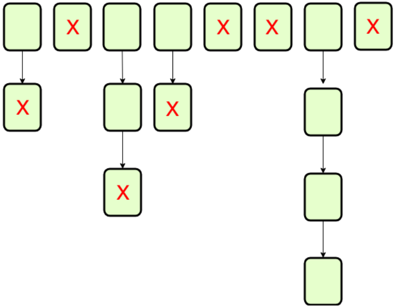


Child Network Selection

Hill Climbing:



Successive Halving:



Child Network Selection

Successive halving on top of hill climbing

- Hill climbing for each evolutionary step
- Successive halving for selecting the most promising half of the population within evalutionary steps

Child Network Selection

Successive halving on top of hill climbing

- Hill climbing for each evolutionary step
- Successive halving for selecting the most promising half of the population within evalutionary steps

Network Morphisms

- Child network has exactly the same knowledge as its parent
- Train it for a short time to evaluate its performance

Overview

Experiments

Our goal:

- Better test accuracy
- Smaller models

Experiments

Our goal:

- Better test accuracy
- Smaller models

What we have so far?

Experiments

Our goal:

- Better test accuracy
- Smaller models

What we have so far?

- 6 network operators
- 2 network selection methods
- Learning rate schedulers

Experiments setting

- 8 independent runs for each model

Experiments setting

- 8 independent runs for each model
- Hill climbing as the global search procedure

Experiments setting

- 8 independent runs for each model
- Hill climbing as the global search procedure
- SGDR and cosine annealing

Experiments setting

- 8 independent runs for each model
- Hill climbing as the global search procedure
- SGDR and cosine annealing
- 160 training epochs in total per evolutionary step

Experiments setting

- 8 independent runs for each model
- Hill climbing as the global search procedure
- SGDR and cosine annealing
- 160 training epochs in total per evolutionary step
- 8 children per evolutionary step

Experiments setting

- 8 independent runs for each model
- Hill climbing as the global search procedure
- SGDR and cosine annealing
- 160 training epochs in total per evolutionary step
- 8 children per evolutionary step
- Uniform distribution of network operators

Experiments setting

- 8 independent runs for each model
- Hill climbing as the global search procedure
- SGDR and cosine annealing
- 160 training epochs in total per evolutionary step
- 8 children per evolutionary step
- Uniform distribution of network operators
- 5 mutations per child network

Experiments setting

- 8 independent runs for each model
- Hill climbing as the global search procedure
- SGDR and cosine annealing
- 160 training epochs in total per evolutionary step
- 8 children per evolutionary step
- Uniform distribution of network operators
- 5 mutations per child network
- Same hardware

Vanilla Model

A base model to start architecture search

- 3 $Conv - BatchNorm - ReLu$ blocks
- $MaxPool2d$ layers between the blocks
- Final $Dense$ layer

Trained for 20 epochs before the search starts

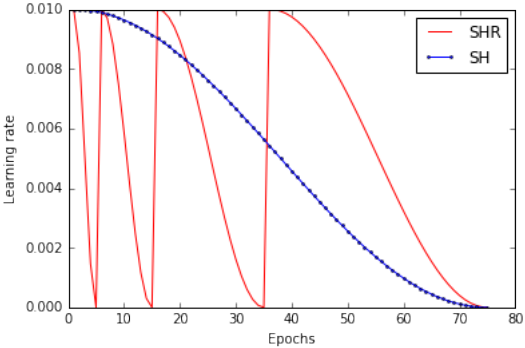
Experimented Models

- Hill climbing
  - NASH (**N**eural **A**rchitecture **S**earch by **H**ill Climbing)
- Successive halving
  - SH (**S**uccessive **H**alving)
  - SHR (**SH** with **R**estarts between successive halving steps)
  - SHRSep (**SHR** with **Sep**arable Convolutions)

Insert Separable Convolutional Layer operator is used only for SHRSep

Scheduler difference

Different learning rate schedules for successive halving models in one evolutionary step for SH and SHR.



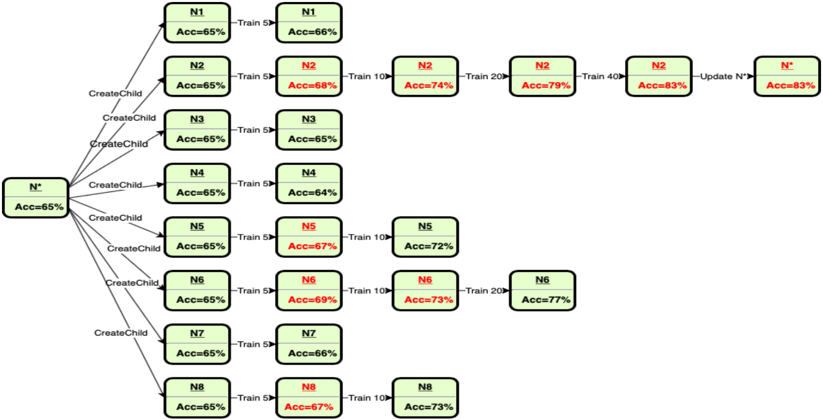
NASH

- NASH


N*
Acc=65%
CreateChild
CreateChild
CreateChild
CreateChild
CreateChild
CreateChild
CreateChild
CreateChild
N1
Acc=65%
N2
Acc=65%
N3
Acc=65%
N4
Acc=65%
N5
Acc=65%
N6
Acc=65%
N7
Acc=65%
N8
Acc=65%
Train 20
N1
Acc=72%
N2
Acc=75%
N3
Acc=74%
N4
Acc=78%
N5
Acc=79%
N6
Acc=74%
N7
Acc=74%
N8
Acc=77%
Update N*
N*
Acc=79%

Successive halving based methods

- SH
- SHR
- SHRSep



Overview

Averaged results

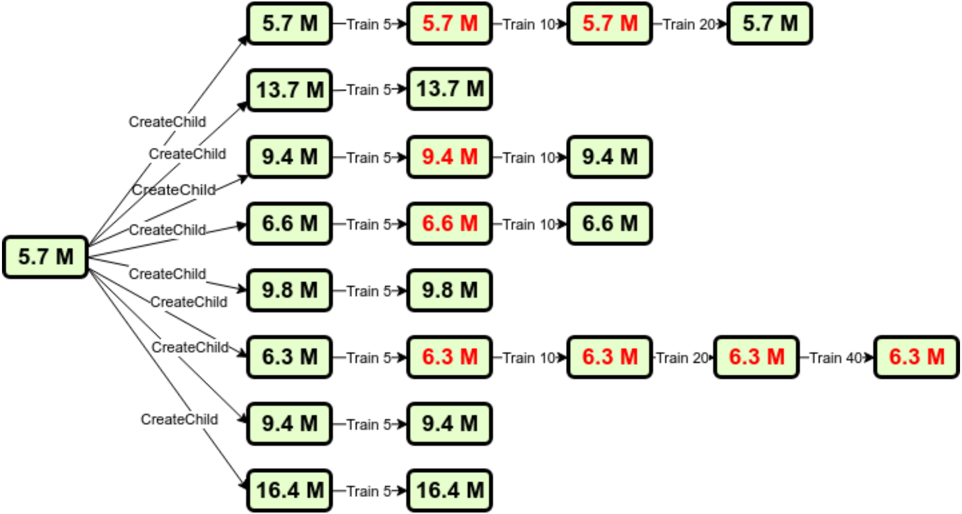
Experiment results for all methods from 8 runs

Exp	Acc ± STD	Params(M)± STD
NASH	94.64 ± 0.38	18.2 ± 6.2
SH	94.57 ± 0.11	12.9 ± 5.9
SHR	94.63 ± 0.29	14.8 ± 4.8
SHRSep	94.39 ± 0.33	14.5 ± 4.5

- All SH* models have less parameters than NASH
- SH and SHR models are as good as NASH
- SHRSep has the lowest test accuracy

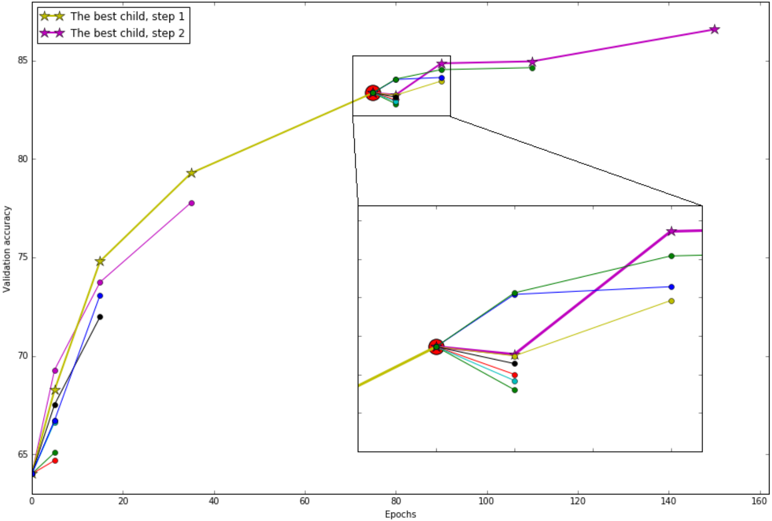
Successive halving budget problems (1)

- Less parameters



Successive halving budget problems (2)

- Drop of validation accuracy after learning rate restart



Layer statistics

Experiment	Conv Blocks	Merging Layers	Sep Conv Blocks
NASH	$74 \pm 4\%$	$19 \pm 3\%$	0%
SH	$67 \pm 2\%$	$25 \pm 2\%$	0%
SHR	$70 \pm 2\%$	$22 \pm 2\%$	0%
SHRSep	$52 \pm 11\%$	$18 \pm 1\%$	$23 \pm 11\%$

No correlation between the model's success and proportion of different layer types it has.

MaxPool2d and *Dense* layer's contributions are excluded.

Overview

Discussion

Results

- In average successive halving models are smaller than models that are found by NASH

Discussion

Results

- In average successive halving models are smaller than models that are found by NASH
- Successive halving models are as good as NASH models

Discussion

Results

- In average successive halving models are smaller than models that are found by NASH
- Successive halving models are as good as NASH models
- Separable convolutions did not improve the performance

Discussion

Results

- In average successive halving models are smaller than models that are found by NASH
- Successive halving models are as good as NASH models
- Separable convolutions did not improve the performance
- Problem of discarding potentially good models

Discussion

Results

- In average successive halving models are smaller than models that are found by NASH
- Successive halving models are as good as NASH models
- Separable convolutions did not improve the performance
- Problem of discarding potentially good models

Future work

Discussion

Results

- In average successive halving models are smaller than models that are found by NASH
- Successive halving models are as good as NASH models
- Separable convolutions did not improve the performance
- Problem of discarding potentially good models

Future work

- A larger initial budget for successive halving

Discussion

Results

- In average successive halving models are smaller than models that are found by NASH
- Successive halving models are as good as NASH models
- Separable convolutions did not improve the performance
- Problem of discarding potentially good models

Future work

- A larger initial budget for successive halving
- Network operators' distribution

Discussion

Results

- In average successive halving models are smaller than models that are found by NASH
- Successive halving models are as good as NASH models
- Separable convolutions did not improve the performance
- Problem of discarding potentially good models

Future work

- A larger initial budget for successive halving
- Network operators' distribution
- Approximate network morphisms when the model is large enough

Discussion

Results

- In average successive halving models are smaller than models that are found by NASH
- Successive halving models are as good as NASH models
- Separable convolutions did not improve the performance
- Problem of discarding potentially good models

Future work

- A larger initial budget for successive halving
- Network operators' distribution
- Approximate network morphisms when the model is large enough
- Hyperparameter optimization parallel to neural architecture search

Thank you!

Overview

References

Baker, B., Gupta, O., Naik, N., and Raskar, R. (2017).
Designing neural network architectures using reinforcement learning.
https://arxiv.org/abs/1611.02167v3.

Elsken, T., Metzen, J. H., and Hutter, F. (2018a).
Multi-objective architecture search for cnns.
https://arxiv.org/abs/1804.09081v1.

Elsken, T., Metzen, J. H., and Hutter, F. (2018b).
Neural architecture search: A survey.
https://arxiv.org/abs/1808.05377v1.

Pham, H., Guan, M. Y., Zoph, B., Le, Q. V., and Dean, J. (2018).
Efficient neural architecture search via parameter sharing.
https://arxiv.org/abs/1802.03268v2.

Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. (2018).
Regularized evolution for image classifier architecture search.
https://arxiv.org/abs/1802.01548v4.

Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y. L., Tan, J., Le, Q. V., and Kurakin, A. (2017).
Large scale evolution of image classifiers.
https://arxiv.org/abs/1703.01041v2.

Stanley, K. O. and Miikkulainen, R. (2002).
Evolving neural networks through augmenting topologies.
MIT Press Journals.

Zoph, B. and Le, Q. V. (2017).
Neural architecture search with reinforcement learning.
https://arxiv.org/abs/1611.01578v2.

Network Morphisms

The concept of **transforming a network to a new one** with complete knowledge of the parent network

Network Morphism equation:

$$N^{w}(x) = (TN)^{\tilde{w}}(x) \; , \forall x \in \mathcal{X} \tag{3}$$

- $\mathcal{N}(\mathcal{X})$ set of network, $N \in \mathcal{N}$
- $\mathcal{X} \subset \mathbb{R}^{n}$
- T is a **network morphism operator**
- $T : \mathcal{N}(\mathcal{X}) \times \mathbb{R}^{k} \to \mathcal{N}(\mathcal{X}) \times \mathbb{R}^{j}$
- $w \in \mathbb{R}^{k}$ *and* $\tilde{w} \in \mathbb{R}^{j}$

Network Operators: Insert Convolutional Layer

- Identity mapping

$$\begin{bmatrix} \begin{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{bmatrix} \\ \begin{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{bmatrix} \\ \begin{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Network Operators: Insert Separable Conv. Layer

- Identity mapping for *Depthwise Convolutional* layer

$$\begin{bmatrix}\begin{bmatrix}\begin{bmatrix}0 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 0\end{bmatrix}\end{bmatrix}\\ \begin{bmatrix}\begin{bmatrix}0 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 0\end{bmatrix}\end{bmatrix}\\ \begin{bmatrix}\begin{bmatrix}0 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 0\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

- Identity mapping for *Pointwise Convolutional* layer

$$\begin{bmatrix}[[1] & [0] & [0]]\\ [[0] & [1] & [0]]\\ [[0] & [0] & [1]]\end{bmatrix}$$