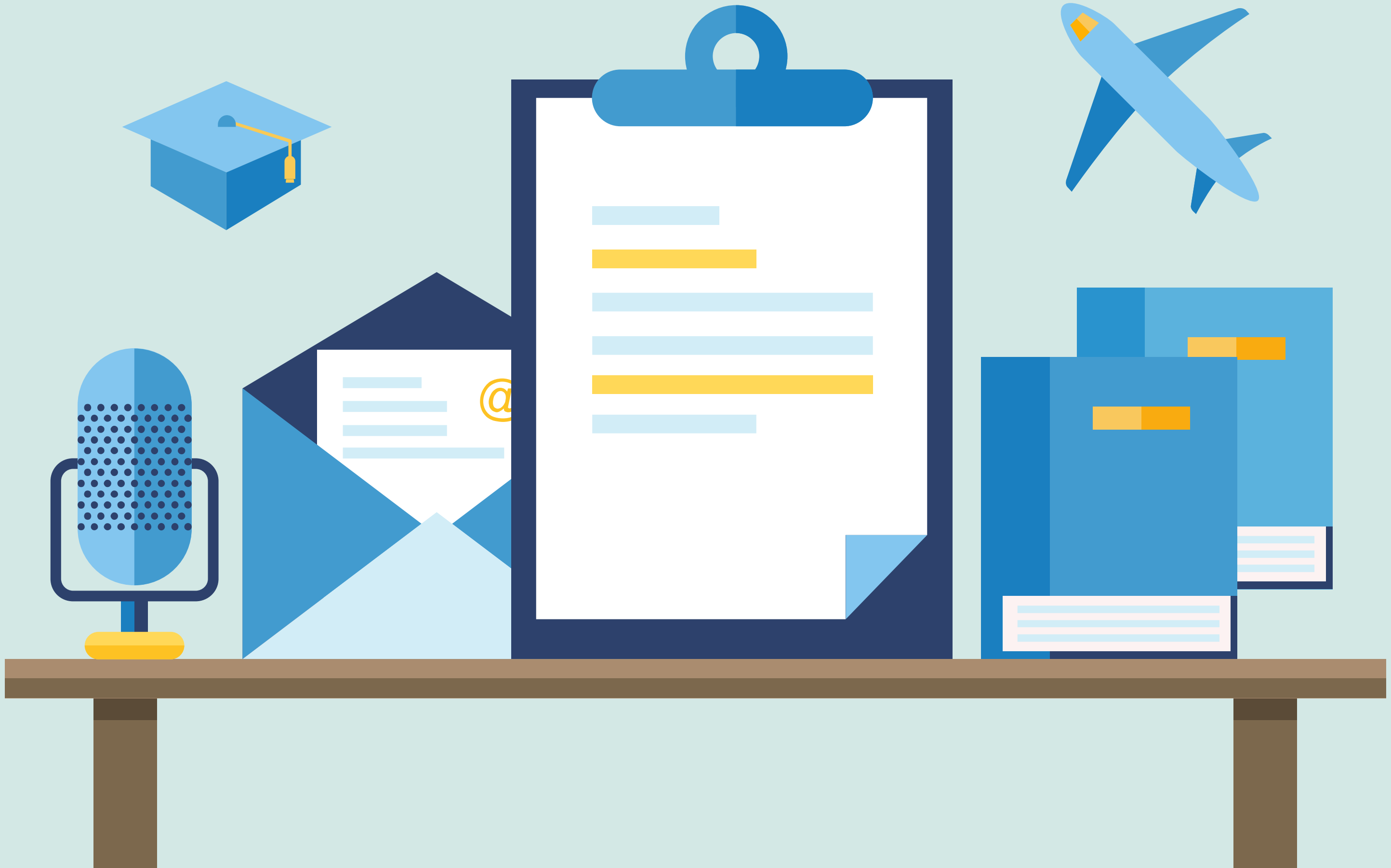
Neural Architecture Optimization
神经网络结构优化

赵鉴



CONTENTS

01

AutoML

Auto Machine Learning

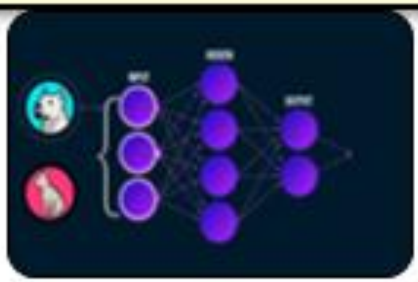
Typical Machine Learning

- Typical Machine **Learning** Process

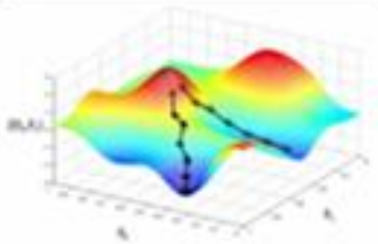
- Fixed data order
- Fixed model space
- Fixed loss function

Training data D

Model Space Ω



Loss function L



Auto Machine Learning

Choose Training Data D

Design Model Space Ω

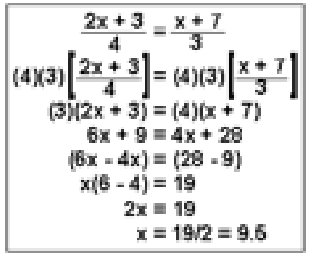
Set Loss Functions L

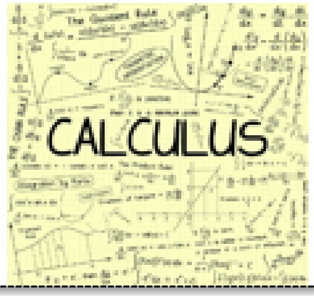
Learning of f_ω

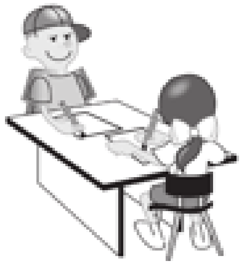
- Auto data selection/process
- Auto model selection and training
- Auto hyper parameter tuning

Text Boo

dents' Learning Process







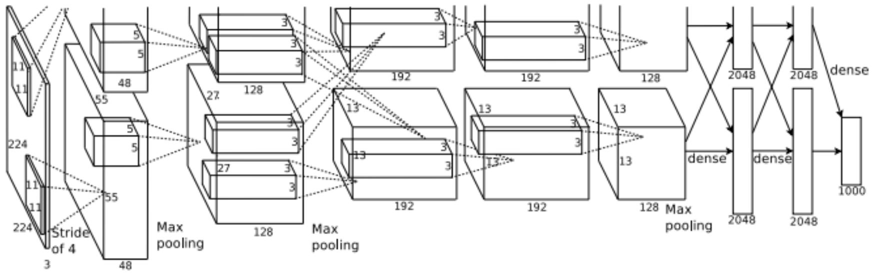
02

NAS

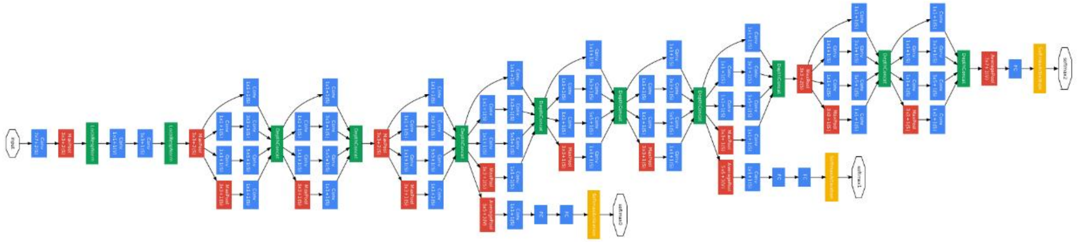
Neural Architecture Search

Architecture of a Neural Network is Crucial to its Performance

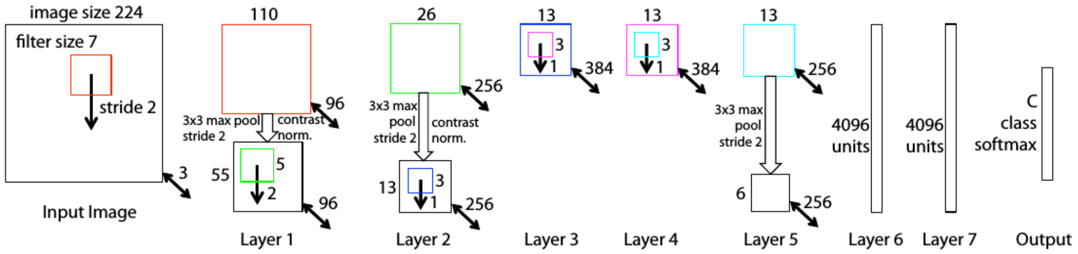
ImageNet Winning Neural Architectures



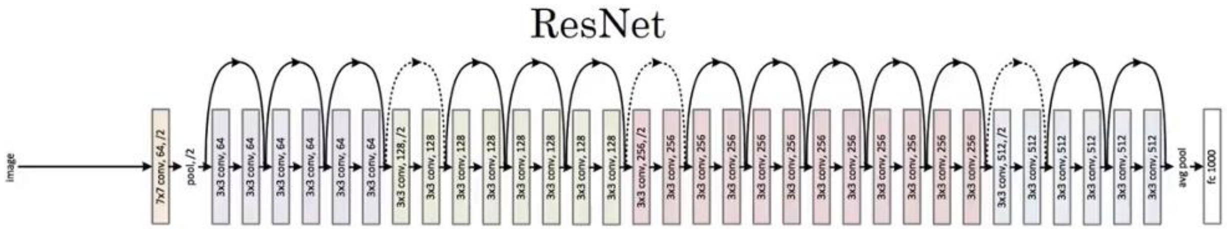
AlexNet 2012



Inception 2014



ZFNet 2013



ResNet 2015

NAS

Neural Architecture Search

Given Dataset

i.e., CIFAR-10, CIFAR-100

PTB, WikiText-2

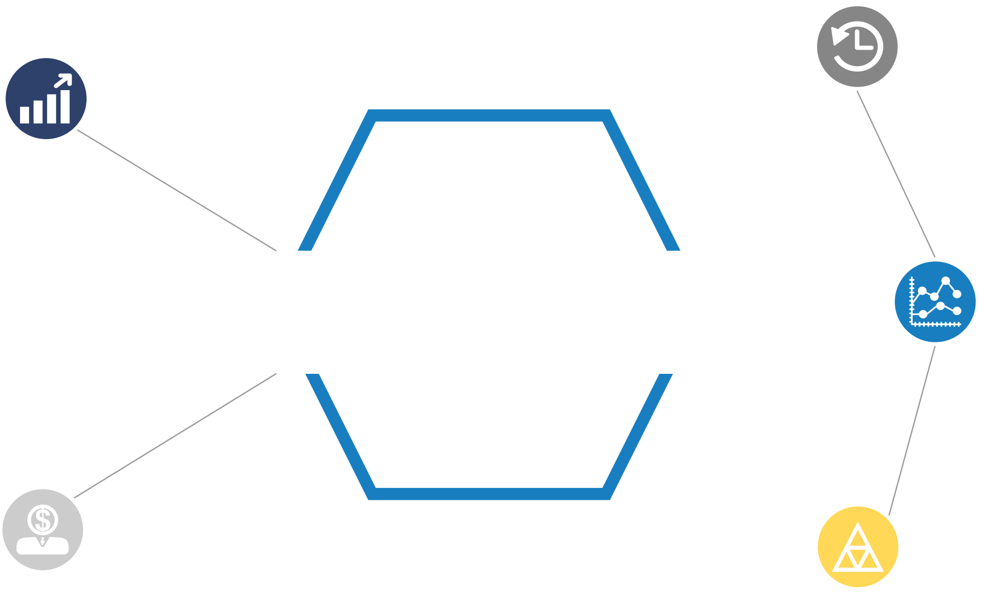
…

Target Task

i.e., image classification,

language modeling,

…



Automatic

Not many human efforts

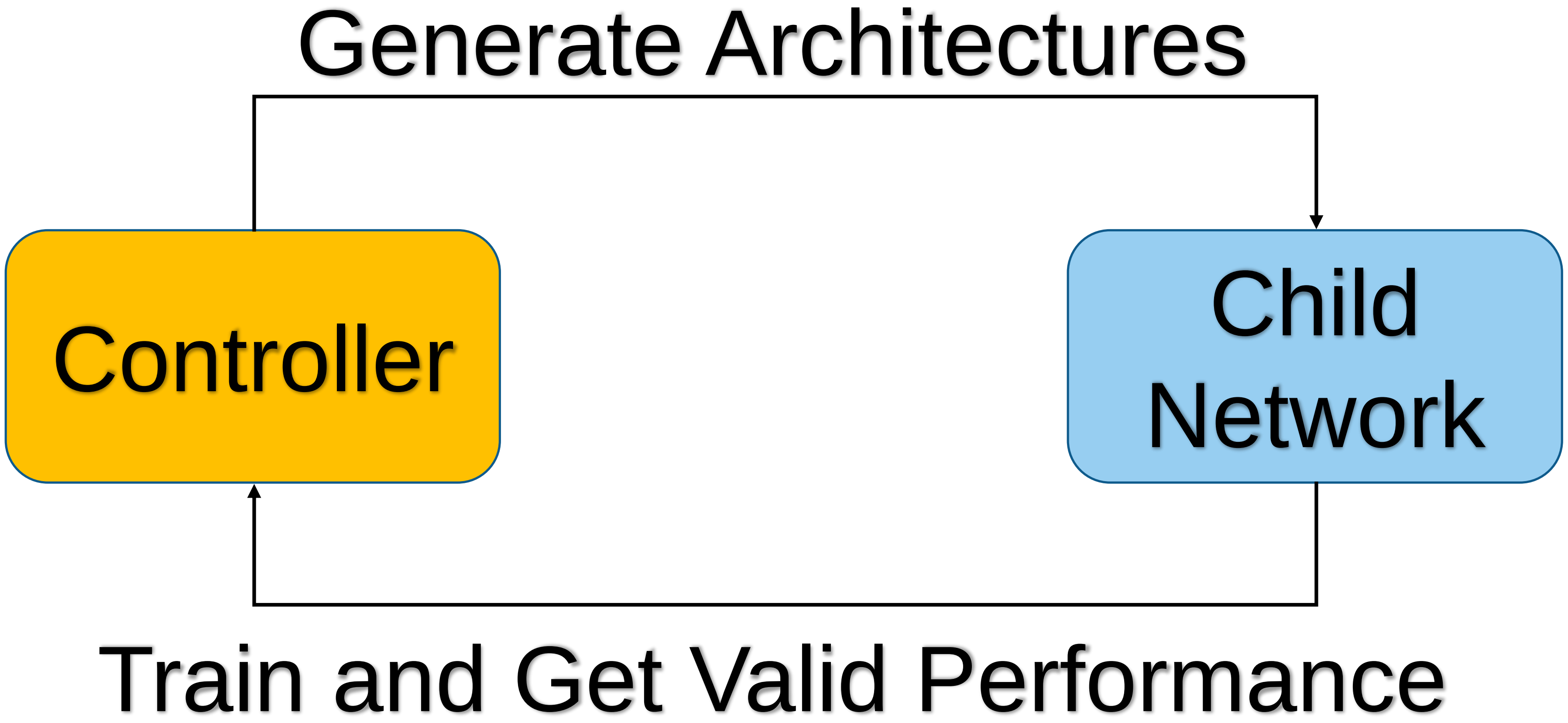
Output

Network architecture that fits given dataset on the target task

Goal

Alleviate the pain of human efforts

General Framework



Typical Search Methods/Algorithms

- Reinforcement Learning
  - Take each architecture choice (i.e., sub-architecture) as **action**
  - Take valid performance as **reward**
  - Use **policy gradient** to search the best action

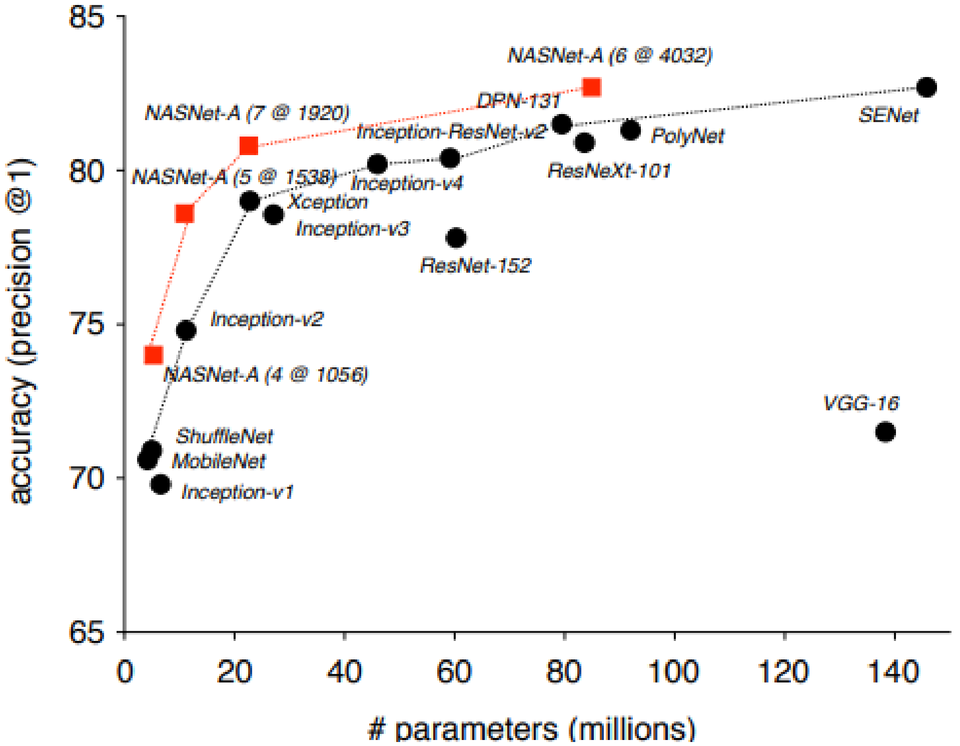
  - NAS-RL (Google, 2017)
  - NASNet (Google, 2017)
  - ENAS (CMU & Google, 2018)
  - …

- Evolutionary Computing
  - Changing the architecture as **mutation** and **selection**
  - Take the valid performance as **fitness**
  - Evolve the architectures

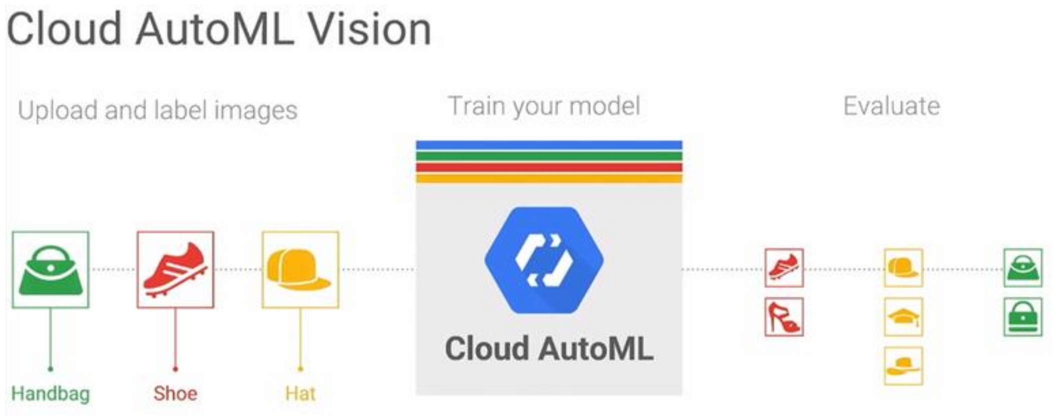
  - AmoebaNet
  - …

Results of Previous NAS Works

- In terms of pushing SOTA results
  - On ImageNet



- In terms of building products with AutoML
  - Microsoft, Google, …
  - Startups focus on AutoML

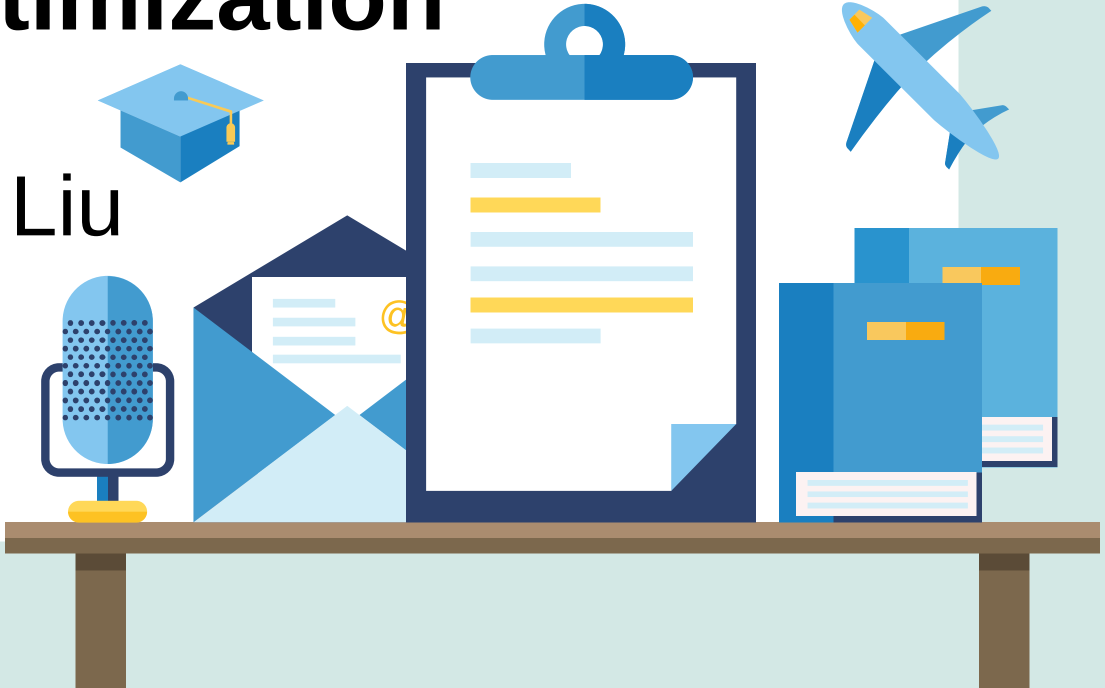


03

Neural Architecture Optimization

Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, Tie-Yan Liu

NIPS 2018



Are Previous NAS Works Perfect Enough?

Why Search in Discrete space?

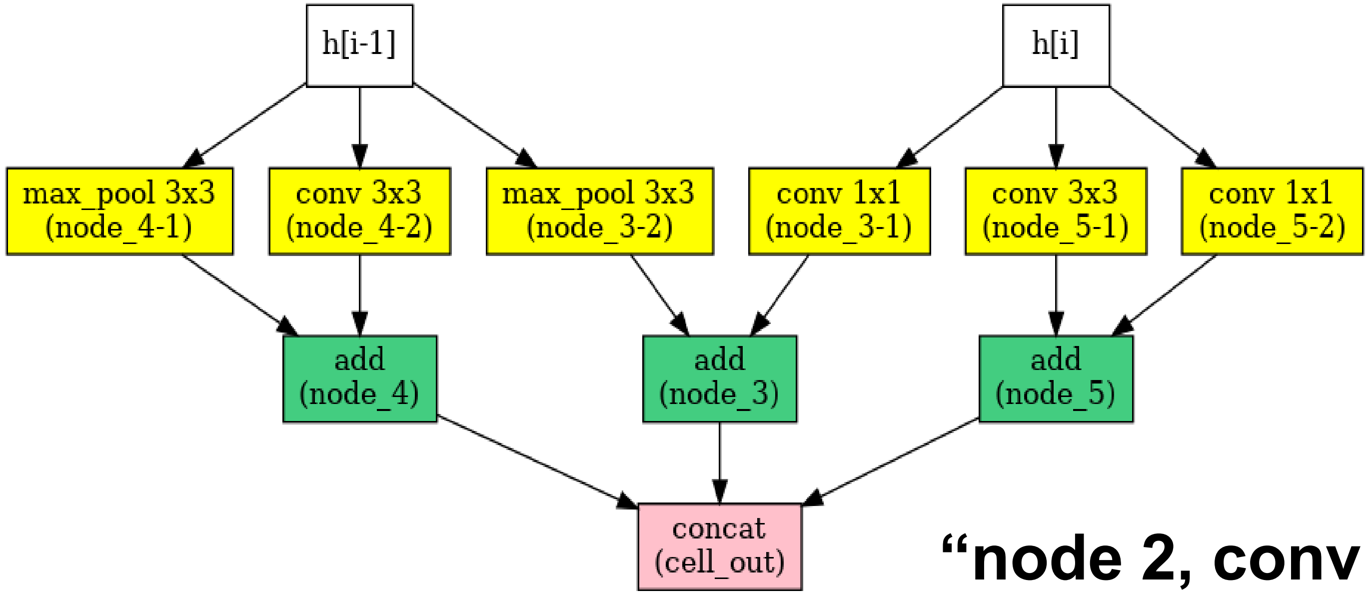
- Exponentially large and thus hard to search

How about Optimize in Continuous Space?

- Compact and easy to optimize
- Bring gradient (based optimization) back!

Basic Methods

- Use a string to indicate the architectures
- Search based on the data (x, y), where x is arch string, y is its valid performance



**“node 2, conv 1x1, node 1,
max pooling, node 1, max pooling, node 1,
conv 3x3, node 2, conv 3x3, node 2, conv 1x1”**

Neural Architecture Optimization (NAO)

01

Encoder - LSTM

- Encodes the discrete string tokens $\boldsymbol{x}$ to an embedding vector $\boldsymbol{e_x}$ in continuous space

02

Performance Predictor - FCN

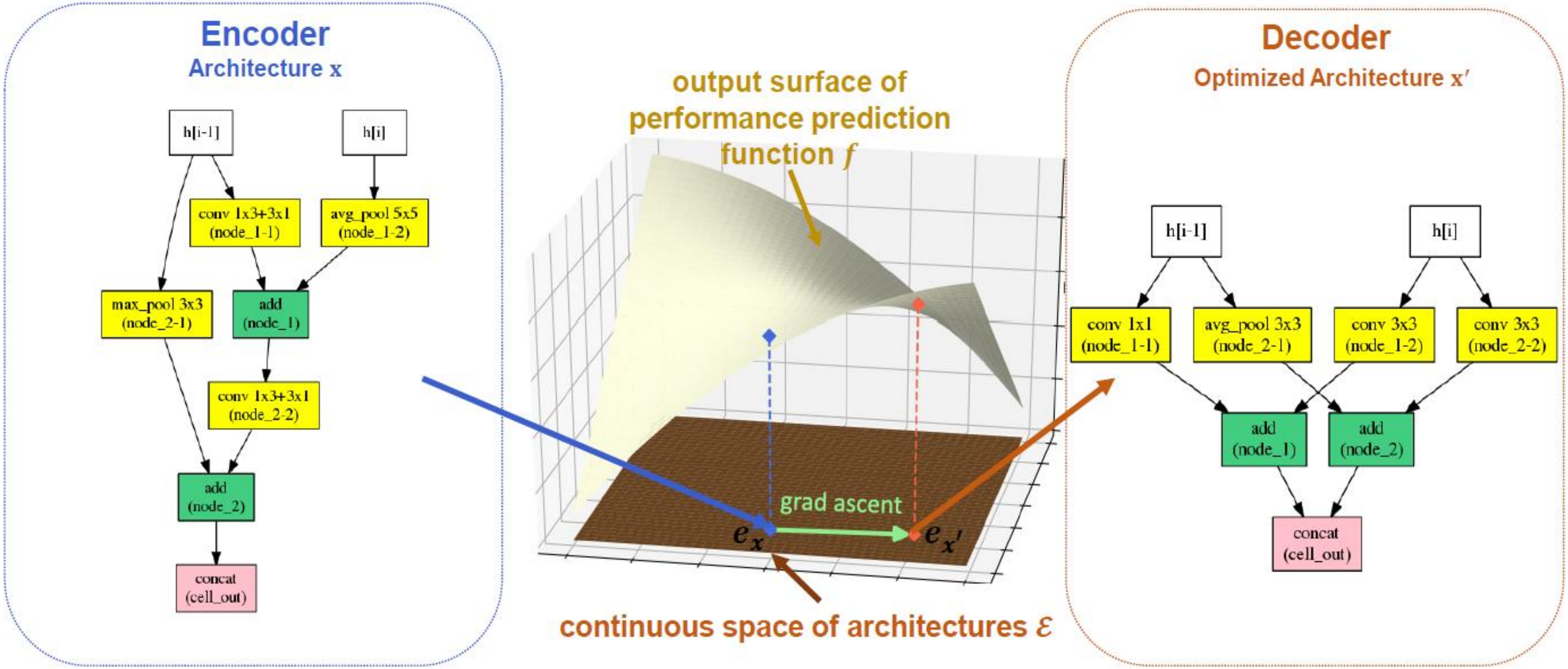
- Maps $\boldsymbol{e_x}$ to its valid performance
- Move towards the direction of gradients

03

Decoder - LSTM

- Decoders the embedding vector $\boldsymbol{e_{x'}}$ back to the discrete tokens $\boldsymbol{x}'$

Gradient-Based Search in Continuous Space



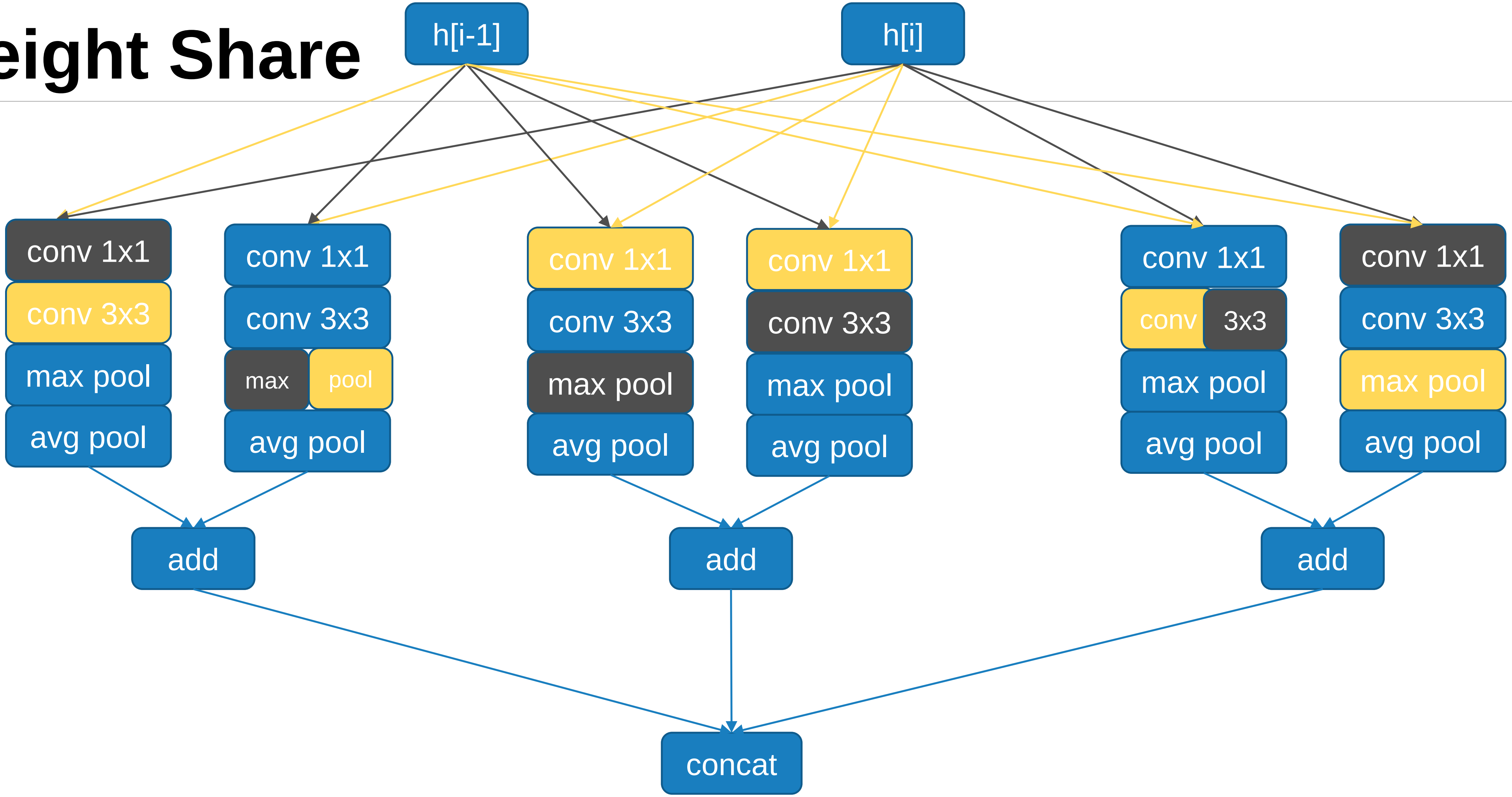
Training & Inferencing

- Train Encoder-Predictor-Decoder
  - Architecture pool of hundreds of (x, y) pairs
  - Data augmentation:
    - symmetry architectures, swap two branches
    - i.e. “node1 conv 1x1 node2 conv 3x3” -> “node2 conv 3x3 node1 conv 1x1”
  - Encoder maps architecture x into $\boldsymbol{e_x}$
  - Performance-Predictor loss: squared error
    - $\boldsymbol{L_{pp}} = \sum_{x \in X} (\boldsymbol{s_x} - \boldsymbol{f}(\boldsymbol{e_x}))^2$
  - Decoder loss: reconstruction loss, nll loss
    - $\boldsymbol{L_{rec}} = \sum_{x \in X} (-\log_e \boldsymbol{P_D}\,(\boldsymbol{x}|\boldsymbol{e_x}))$
  - Jointly train three components together
    - $\boldsymbol{L} = \boldsymbol{\lambda L_{pp}} + (\mathbf{1} - \boldsymbol{\lambda})\boldsymbol{L_{rec}}$

- Generate new architectures:
  - Generate new architecture embedding with step size $\boldsymbol{\eta}$: $\boldsymbol{e_{x'}} = \boldsymbol{e_x} + \boldsymbol{\eta}\nabla\boldsymbol{e_x}$
  - Decoder maps $\boldsymbol{e_{x'}}$ back into $\boldsymbol{x'}$

- Iterate: Train and evaluate new generated architectures and iterate over above steps

Weight Share



Architecture 1: "node 2, conv 1x1, node 1, max pooling, node 1, max pooling, node 1, conv 3x3, node 2, conv 3x3, node 2, conv 1x1"

Architecture 2: "node 1, conv 3x3, node 2, max pooling, node 2, conv 1x1, node 2, conv 1x1, node 1, conv 3x3, node 1, max pooling"

04
Experiments and Results

Task

Image Classification

Classify the images

CIFAR-10

10 classes

50000 images for training

10000 images for testing

CIFAR-100

100 classes

50000 images for training

10000 images for testing

Language Modeling

Modeling the probability distribution over sequences of words in natural language

PTB

Penn Tree Bank

WT2

WikiText-2

CIFAR-10

Model	B	N	F	#op	Error(%)	#params	M	GPU Days
DenseNet-BC [19]		100	40	/	3.46	25.6M	/	/
ResNeXt-29 [43]				/	3.58	68.1M	/	/
NASNet-A [47]	5	6	32	13	3.41	3.3M	20000	2000
NASNet-B [47]	5	4	N/A	13	3.73	2.6M	20000	2000
NASNet-C [47]	5	4	N/A	13	3.59	3.1M	20000	2000
Hier-EA [27]	5	2	64	6	3.75	15.7M	7000	300
AmoebaNet-A [38]	5	6	36	10	3.34	3.2M	20000	3150
AmoebaNet-B [38]	5	6	36	19	3.37	2.8M	27000	3150
AmoebaNet-B [38]	5	6	80	19	3.04	13.7M	27000	3150
AmoebaNet-B [38]	5	6	128	19	2.98	34.9M	27000	3150
AmoebaNet-B + Cutout [38]	5	6	128	19	2.13	34.9M	27000	3150
PNAS [26]	5	3	48	8	3.41	3.2M	1280	225
ENAS [36]	5	5	36	5	3.54	4.6M	/	0.45
Random-WS	5	5	36	5	3.92	3.9M	/	0.25
DARTS + Cutout [28]	5	6	36	7	2.83	4.6M	/	4
NAONet	5	6	36	11	3.18	10.6M	1000	200
NAONet	5	6	64	11	2.98	28.6M	1000	200
NAONet + Cutout	5	6	128	11	2.11	128M	1000	200
NAONet-WS	5	5	36	5	3.53	2.5M	/	0.3

Transfer to CIFAR-100

Model	B	N	F	#op	Error (%)	#params
DenseNet-BC [19]	/	100	40	/	17.18	25.6M
Shake-shake [15]	/	/	/	/	15.85	34.4M
Shake-shake + Cutout [11]	/	/	/	/	15.20	34.4M
NASNet-A [47]	5	6	32	13	19.70	3.3M
NASNet-A [47] + Cutout	5	6	32	13	16.58	3.3M
NASNet-A [47] + Cutout	5	6	128	13	16.03	50.9M
PNAS [26]	5	3	48	8	19.53	3.2M
PNAS [26] + Cutout	5	3	48	8	17.63	3.2M
PNAS [26] + Cutout	5	6	128	8	16.70	53.0M
ENAS [36]	5	5	36	5	19.43	4.6M
ENAS [36] + Cutout	5	5	36	5	17.27	4.6M
ENAS [36] + Cutout	5	5	36	5	16.44	52.7M
AmoebaNet-B [38]	5	6	128	19	17.66	34.9M
AmoebaNet-B [38] + Cutout	5	6	128	19	15.80	34.9M
NAONet + Cutout	5	6	36	11	15.67	10.8M
NAONet + Cutout	5	6	128	11	14.75	128M

PTB

Models and Techniques	#params	Test Perplexity	GPU Days
Vanilla LSTM [45]	66M	78.4	/
LSTM + Zoneout [23]	66M	77.4	/
Variational LSTM [14]	19M	73.4	
Pointer Sentinel-LSTM [33]	51M	70.9	/
Variational LSTM + weight tying [20]	51M	68.5	/
Variational Recurrent Highway Network + weight tying [46]	23M	65.4	/
4-layer LSTM + skip connection + averaged weight drop + weight penalty + weight tying [31]	24M	58.3	/
LSTM + averaged weight drop + Mixture of Softmax + weight penalty + weight tying [44]	22M	56.0	/
NAS + weight tying [47]	54M	62.4	1e4 CPU days
ENAS + weight tying + weight penalty [36]	24M	58.6[5]	0.5
Random-WS + weight tying + weight penalty	27M	58.81	0.4
DARTS+ weight tying + weight penalty [28]	23M	56.1	1
NAONet + weight tying + weight penalty	27M	56.0	300
NAONet-WS + weight tying + weight penalty	27M	56.6	0.4

Transfer to WikiText-2

Models and Techniques	#params	Test Perplexity
Variational LSTM + weight tying [20]	28M	87.0
LSTM + continuos cache pointer [16]	-	68.9
LSTM [32]	33	66.0
4-layer LSTM + skip connection + averaged weight drop + weight penalty + weight tying [31]	24M	65.9
LSTM + averaged weight drop + Mixture of Softmax + weight penalty + weight tying [44]	33M	63.3
ENAS + weight tying + weight penalty [36] (searched on PTB)	33M	70.4
DARTS + weight tying + weight penalty (searched on PTB)	33M	66.9
NAONet + weight tying + weight penalty (searched on PTB)	36M	67.0

05
Conclusion

Conclusion

New automatic architecture design algorithm

- Encodes discrete description into continuous embedding
- Performs the optimization within continuous space
- Uses gradient based method rather than search discrete decisions

Project Link

- Paper Link: https://arxiv.org/abs/1808.07233
- Code Link: https://github.com/renqianluo/NAO

Thanks.

QA