Neural Archetecture Search with Reinforcement Learning

Bio-Medical Computing Laboratory

TAEHEUM, CHO

2th May, 2019

Neural Archetecture Search with Reinforcement Learning – ICLR 2017

Under review as a conference paper at ICLR 2017

NEURAL ARCHITECTURE SEARCH WITH REINFORCEMENT LEARNING

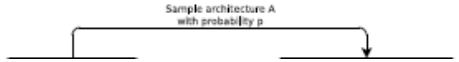
Barret Zoph,* Quoc V. Le
Google Brain
{barretzoph,qvl}@google.com

ABSTRACT

Neural networks are powerful and flexible models that work well for many difficult learning tasks in image, speech and natural language understanding. Despite their success, neural networks are still hard to design. In this paper, we use a recurrent network to generate the model descriptions of neural networks and train this RNN with reinforcement learning to maximize the expected accuracy of the generated architectures on a validation set. On the CIFAR-10 dataset, our method, starting from scratch, can design a novel network architecture that rivals the best human-invented architecture in terms of test set accuracy. Our CIFAR-10 model achieves a test error rate of 3.65, which is 0.09 percent better and 1.05x faster than the previous state-of-the-art model that used a similar architectural scheme. On the Penn Treebank dataset, our model can compose a novel recurrent cell that outperforms the widely-used LSTM cell, and other state-of-the-art baselines. Our cell achieves a test set perplexity of 62.4 on the Penn Treebank, which is 3.6 perplexity better than the previous state-of-the-art model. The cell can also be transferred to the character language modeling task on PTB and achieves a state-of-the-art perplexity of 1.214.

1 INTRODUCTION

The last few years have seen much success of deep neural networks in many challenging applications, such as speech recognition (Hinton et al., 2012), image recognition (LeCun et al., 1998; Krizhevsky et al., 2012) and machine translation (Sutskever et al., 2014; Bahdanau et al., 2015; Wu et al., 2016). Along with this success is a paradigm shift from feature designing to architecture designing, i.e., from SIFT (Lowe, 1999), and HOG (Dalal & Triggs, 2005), to AlexNet (Krizhevsky et al., 2012), VGGNet (Simonyan & Zisserman, 2014), GoogleNet (Szegedy et al., 2015), and ResNet (He et al., 2016a). Although it has become easier, designing architectures still requires a lot of expert knowledge and takes ample time.



Introduction

- AI Researcher can also be replaced AI
- A kind of AutoML algorithm

ARTIFICIAL INTELLIGENCE

Google's New AI Is Better at Creating AI Than the Company's Engineers

This could open AI tech up to non-experts.

Tom Ward | May 19th 2017

Motivation for Architecture Search

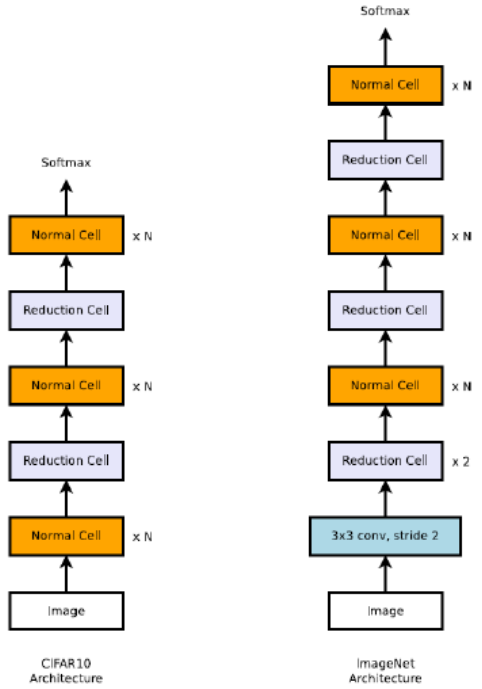
- Designing Neural Network must spend lots of time
- This research is first attempt for AutoML field

Related Work

- Hyperparameter optimization
- Modern neuro-evolution algorithms
- Efficient Neural Architecture Search
- Learning to learn or Meta-Learning
- Bayesian optimization algorithm

Method

- Creating "configuration string" to represent network architecture
  - ["Filter Width: 5", "Filter Height: 3", "Num Filters: 24"] - Caffe
- Overall architectures of the convolutional nets are manually predetermined
  - Normal Cell: convolutional cells that return a feature map of the same dimension
  - Reduction Cell: convolutional cells that return a feature map where the feature map height and width is reduced by a factor of two
- Using softmax classifier for selecting hyper-parameter
- Using RNN architecture for creating this representation string
- Updating parameter of Controller model by reinforcement learning
- Accuracy is used as Reward signal



Training with REINFORCE

Parameter of Controller

Accuracy from Child, Reward signal

$$J(\theta_c) = E_{P(a_{1:T};\theta_c)}[R]$$

number of hyperparameter predicted by Controller

Architecture predicted by the controller RNN viewed as a sequence of actions

$$\nabla_{\theta_c} J(\theta_c) = \sum_{t=1}^{T} E_{P(a_{1:T};\theta_c)}\left[\nabla_{\theta_c} \log P(a_t | a_{(t-1):1}; \theta_c) R\right]$$

- REINFORCE has a few of hyperparameter than compared with Q-learning.
- Accuracy is non-differential value.

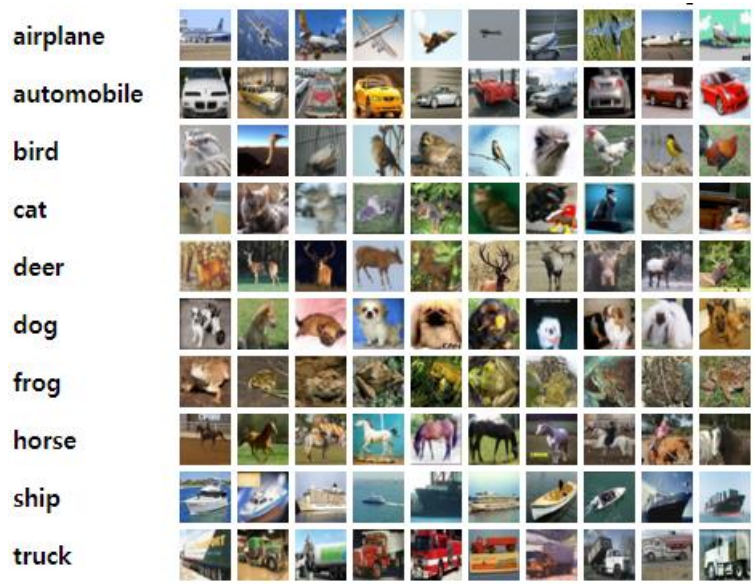
NAS for CNN

number of samples

$$\frac{1}{m}\sum_{k=1}^{m}\sum_{t=1}^{T}\nabla_{\theta_c}\log P(a_t|a_{(t-1):1};\theta_c)R_k$$

baseline for decreasing value

$$\frac{1}{m}\sum_{k=1}^{m}\sum_{t=1}^{T}\nabla_{\theta_c}\log P(a_t|a_{(t-1):1};\theta_c)(R_k-b)$$

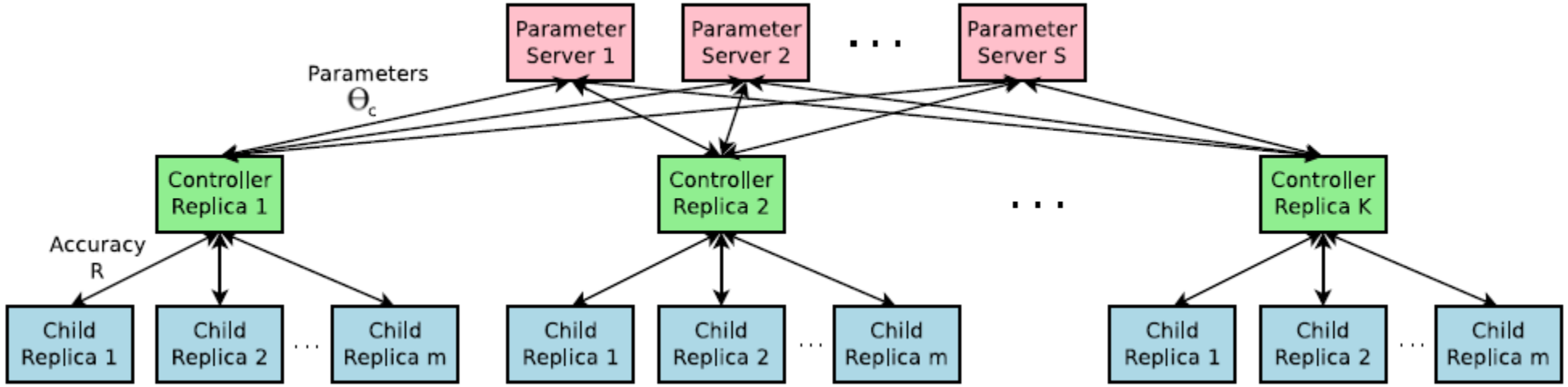


Number	Tag	Description
1.	CC	Coordinating conjunction
2.	CD	Cardinal number
3.	DT	Determiner
4.	EX	Existential *there*
5.	FW	Foreign word
6.	IN	Preposition or subordinating conjunction
7.	JJ	Adjective
8.	JJR	Adjective, comparative
9.	JJS	Adjective, superlative
10.	LS	List item marker
11.	MD	Modal
12.	NN	Noun, singular or mass
13.	NNS	Noun, plural
14.	NNP	Proper noun, singular
15.	NNPS	Proper noun, plural
16.	PDT	Predeterminer
17.	POS	Possessive ending
18.	PRP	Personal pronoun
19.	PRP$	Possessive pronoun
20.	RB	Adverb
21.	RBR	Adverb, comparative
22.	RBS	Adverb, superlative
23.	RP	Particle
24.	SYM	Symbol

15.	NNPS	Proper noun, plural
16.	PDT	Predeterminer
17.	POS	Possessive ending
18.	PRP	Personal pronoun
19.	PRP$	Possessive pronoun
20.	RB	Adverb
21.	RBR	Adverb, comparative
22.	RBS	Adverb, superlative
23.	RP	Particle
24.	SYM	Symbol
25.	TO	*to*
26.	UH	Interjection
27.	VB	Verb, base form
28.	VBD	Verb, past tense
29.	VBG	Verb, gerund or present participle
30.	VBN	Verb, past participle
31.	VBP	Verb, non-3rd person singular present
32.	VBZ	Verb, 3rd person singular present
33.	WDT	Wh-determiner
34.	WP	Wh-pronoun
35.	WP$	Possessive wh-pronoun
36.	WRB	Wh-adverb

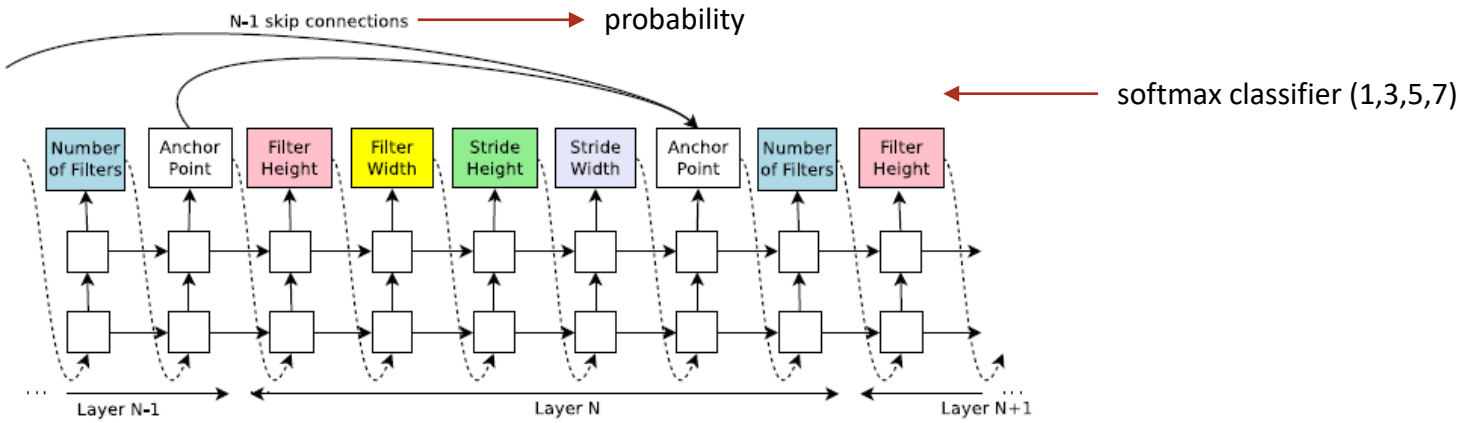
- Dataset: CIFAR-10(CNN), Penn Treebank(RNN)

Distributed Training



- Parameters of Controller is saved in Parameter Servers. Parameter Servers distribute parameters to Controllers.
- Each Controller samples m Child .Childs work in parallel.
- Accuracy of each Child is a reward signal for Controller.

Neural Architecture Search for Convolutional Networks



- Skip connections extend searching space for building network.
- Controller uses skip connections to decide what layers it wants as inputs to the current layer.
- Controller uses anchor points, and set-selection attention to form skip connections.
- Goal: finding [skip connection, filter height-width, stride height-width, no. of filters]

Skip Connection ResNet and DenseNet

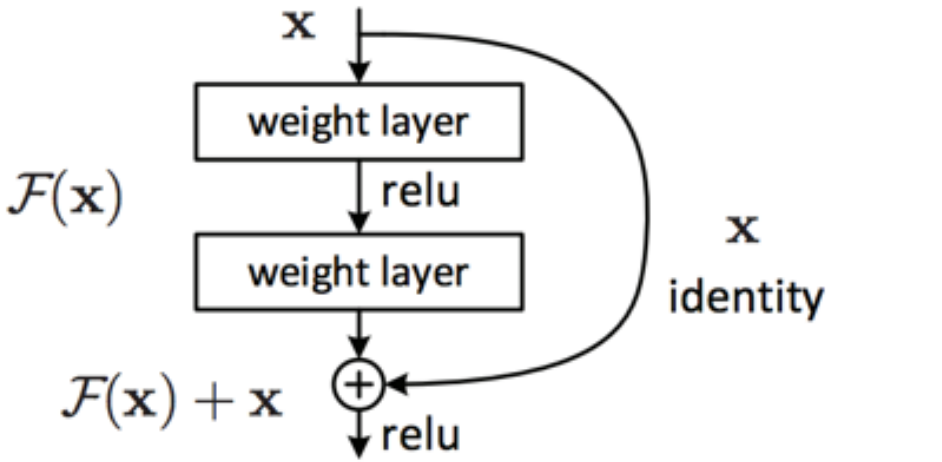
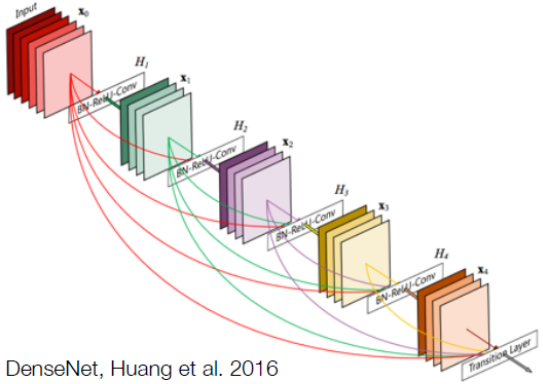
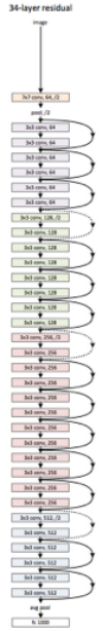


Figure 2. Residual learning: a building block.



DenseNet, Huang et al. 2016



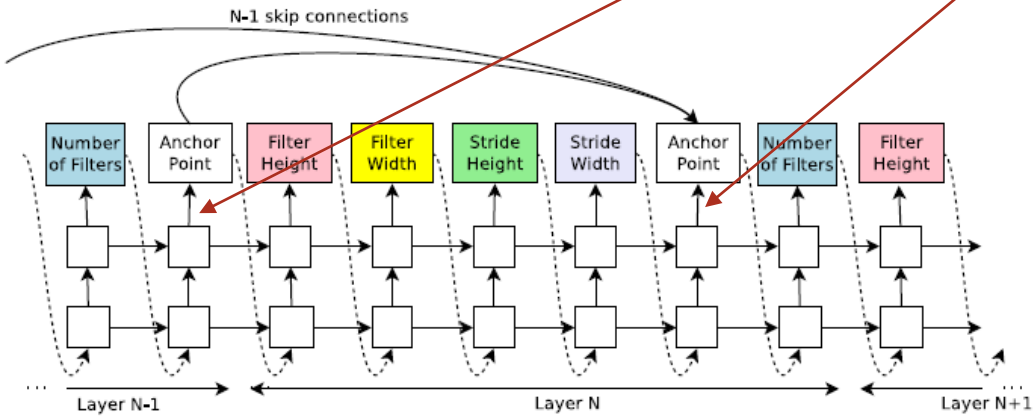
ResNet, He et al. 2015

DenseNets connects each layer to every other layer in a feed-forward fashion. They alleviate the vanishing-gradient problem, strengthen feature propagation, encourage feature reuse, and substantially reduce the number of parameters.

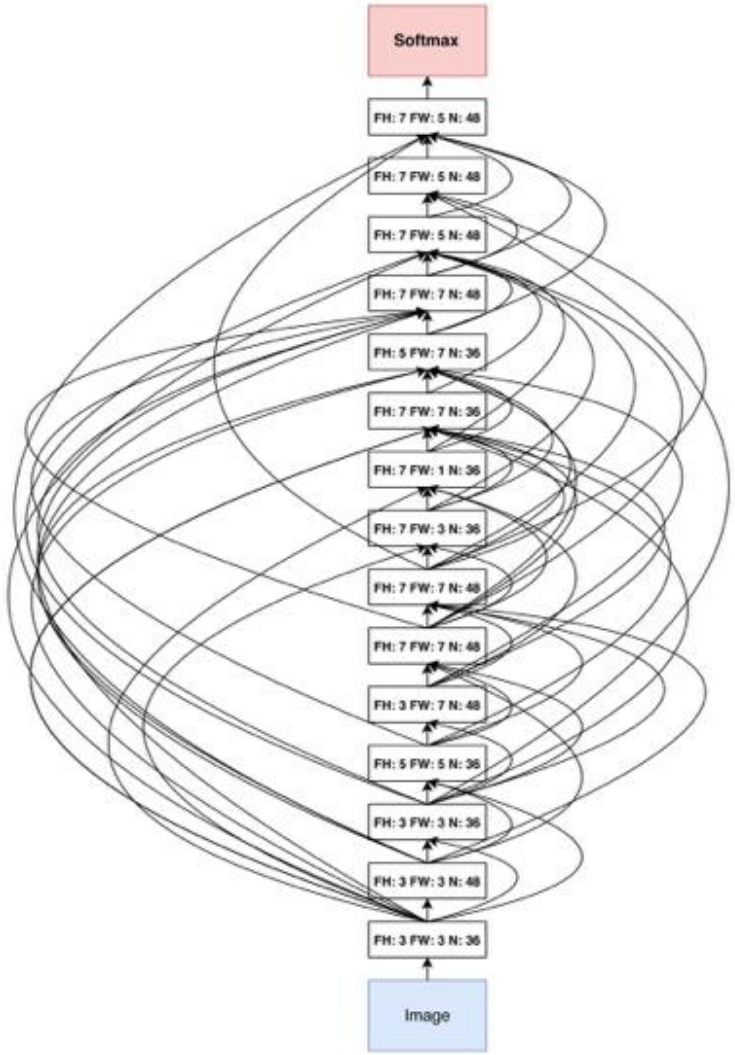
Generated Convolutional Network from Neural Architecture Search

Weight Matrices

$$\mathrm{P}(\text{Layer j is an input to layer i}) = \text{sigmoid}(v^{\mathrm{T}}\tanh(W_{prev} * h_j + W_{curr} * h_i))$$



Generated Convolutional Network from Neural Architecture Search



- There are lots of Skip Connection
- NAS like Rectangle filter

Generated Convolutional Network from Neural Architecture Search

Model	Depth	Parameters	Error rate (%)
Network in Network (Lin et al., 2013)	-	-	8.81
All-CNN (Springenberg et al., 2014)	-	-	7.25
Deeply Supervised Net (Lee et al., 2015)	-	-	7.97
Highway Network (Srivastava et al., 2015)	-	-	7.72
Scalable Bayesian Optimization (Snoek et al., 2015)	-	-	6.37
FractalNet (Larsson et al., 2016)	21	38.6M	5.22
with Dropout/Drop-path	21	38.6M	4.60
ResNet (He et al., 2016a)	110	1.7M	6.61
ResNet (reported by Huang et al. (2016c))	110	1.7M	6.41
ResNet with Stochastic Depth (Huang et al., 2016c)	110	1.7M	5.23
	1202	10.2M	4.91
Wide ResNet (Zagoruyko & Komodakis, 2016)	16	11.0M	4.81
	28	36.5M	4.17
ResNet (pre-activation) (He et al., 2016b)	164	1.7M	5.46
	1001	10.2M	4.62
DenseNet ($L = 40, k = 12$) Huang et al. (2016a)	40	1.0M	5.24
DenseNet($L = 100, k = 12$) Huang et al. (2016a)	100	7.0M	4.10
DenseNet ($L = 100, k = 24$) Huang et al. (2016a)	100	27.2M	3.74
DenseNet-BC ($L = 100, k = 40$) Huang et al. (2016b)	190	25.6M	3.46
Neural Architecture Search v1 no stride or pooling	15	4.2M	5.50
Neural Architecture Search v2 predicting strides	20	2.5M	6.01
Neural Architecture Search v3 max pooling	39	7.1M	4.47
Neural Architecture Search v3 max pooling + more filters	39	37.4M	3.65

Table 1: Performance of Neural Architecture Search and other state-of-the-art models on CIFAR-10.

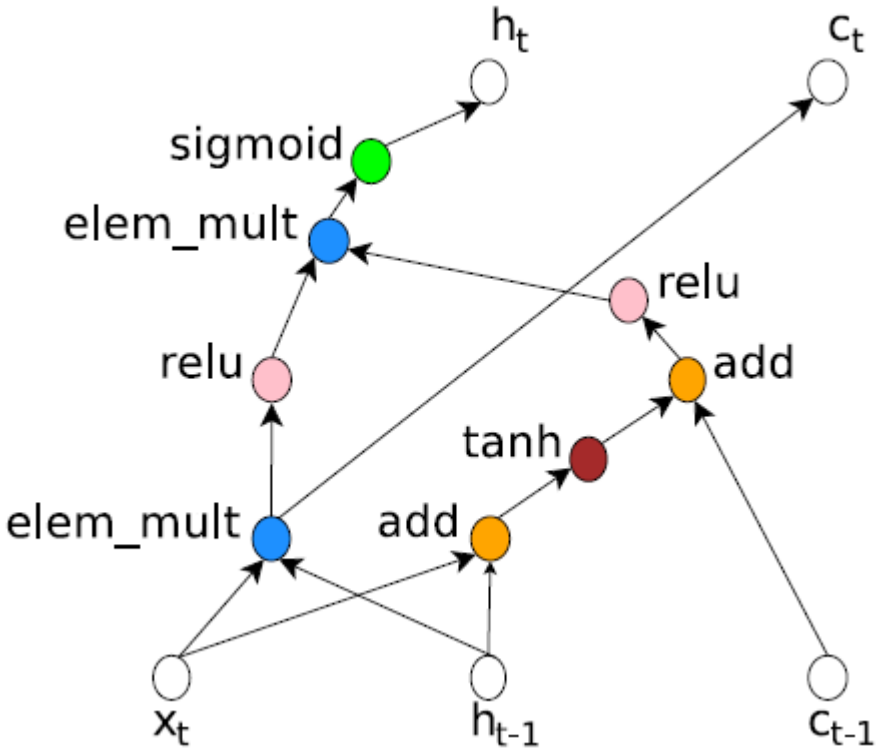
5% faster

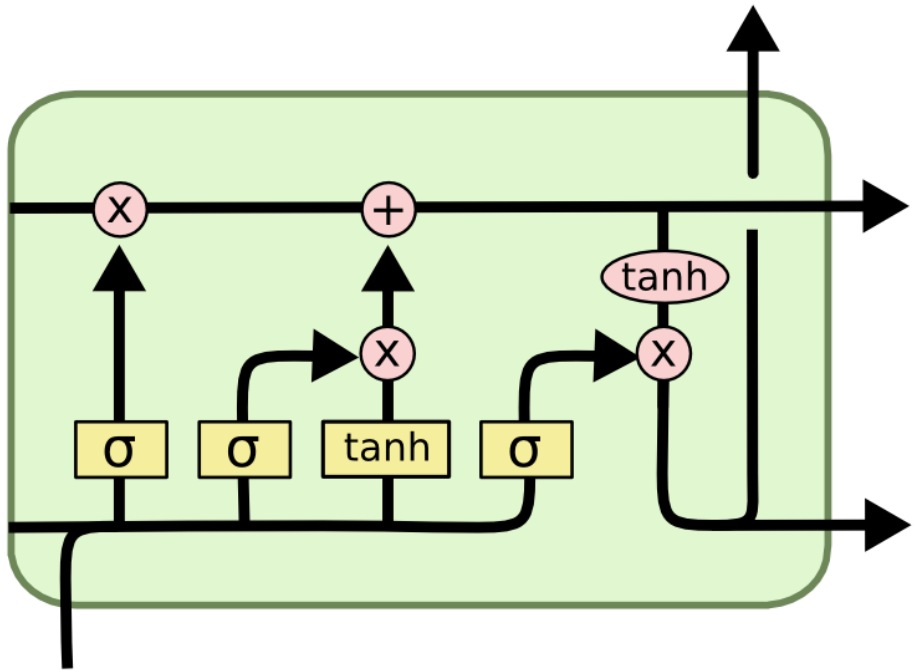
800 GPU

Best result of evolution (Real et al. 2017): 5.4%

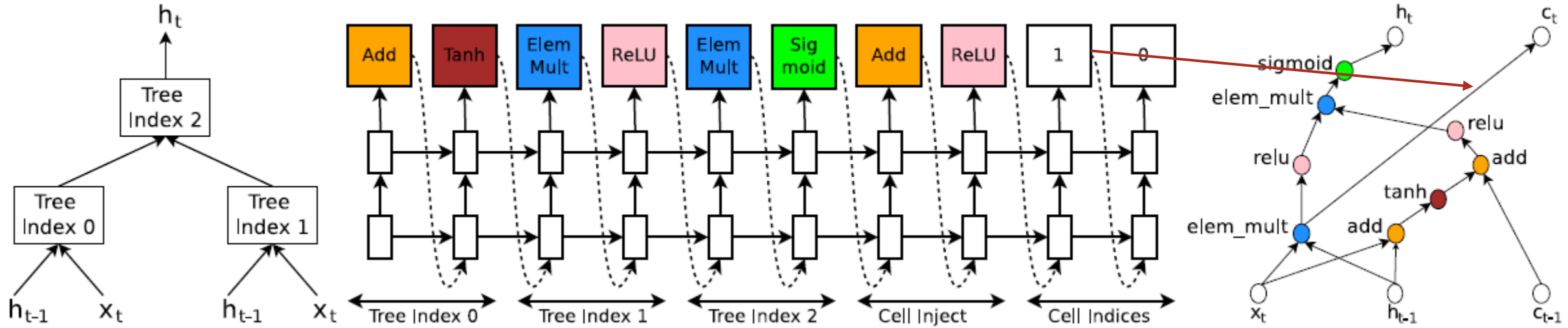
Best result of Q-learning (Baker et al. 2017): 6.92%

Recurrent Cell Prediction Method





Recurrent Cell Prediction Method



- Cell Search Space
  - Predefined
  - leaf node: 8
    - (2 here)
- RNN cell Controller
  - Controller RNN decide which function to use referring to Cell Search Space (Tree)
- cell represected by graph

Model	Parameters	Test Perplexity
Mikolov & Zweig (2012) - KN-5	2M‡	141.2
Mikolov & Zweig (2012) - KN5 + cache	2M‡	125.7
Mikolov & Zweig (2012) - RNN	6M‡	124.7
Mikolov & Zweig (2012) - RNN-LDA	7M‡	113.7
Mikolov & Zweig (2012) - RNN-LDA + KN-5 + cache	9M‡	92.0
Pascanu et al. (2013) - Deep RNN	6M	107.5
Cheng et al. (2014) - Sum-Prod Net	5M‡	100.0
Zaremba et al. (2014) - LSTM (medium)	20M	82.7
Zaremba et al. (2014) - LSTM (large)	66M	78.4
Gal (2015) - Variational LSTM (medium, untied)	20M	79.7
Gal (2015) - Variational LSTM (medium, untied, MC)	20M	78.6
Gal (2015) - Variational LSTM (large, untied)	66M	75.2
Gal (2015) - Variational LSTM (large, untied, MC)	66M	73.4
Kim et al. (2015) - CharCNN	19M	78.9
Press & Wolf (2016) - Variational LSTM, shared embeddings	51M	73.2
Merity et al. (2016) - Zoneout + Variational LSTM (medium)	20M	80.6
Merity et al. (2016) - Pointer Sentinel-LSTM (medium)	21M	70.9
Inan et al. (2016) - VD-LSTM + REAL (large)	51M	68.5
Zilly et al. (2016) - Variational RHN, shared embeddings	24M	66.0
Neural Architecture Search with base 8	32M	67.9
Neural Architecture Search with base 8 and shared embeddings	25M	64.0
Neural Architecture Search with base 8 and shared embeddings	54M	62.4

Table 2: Single model perplexity on the test set of the Penn Treebank language modeling task Parameter numbers with ‡ are estimates with reference to Merity et al. (2016).

100% faster

400 CPU

Conclusion

- NAS > Bayesian optimization
- NAS > Genetic Programming
- NAS > Random Search
- Cells created by NAS work well in other datasets.