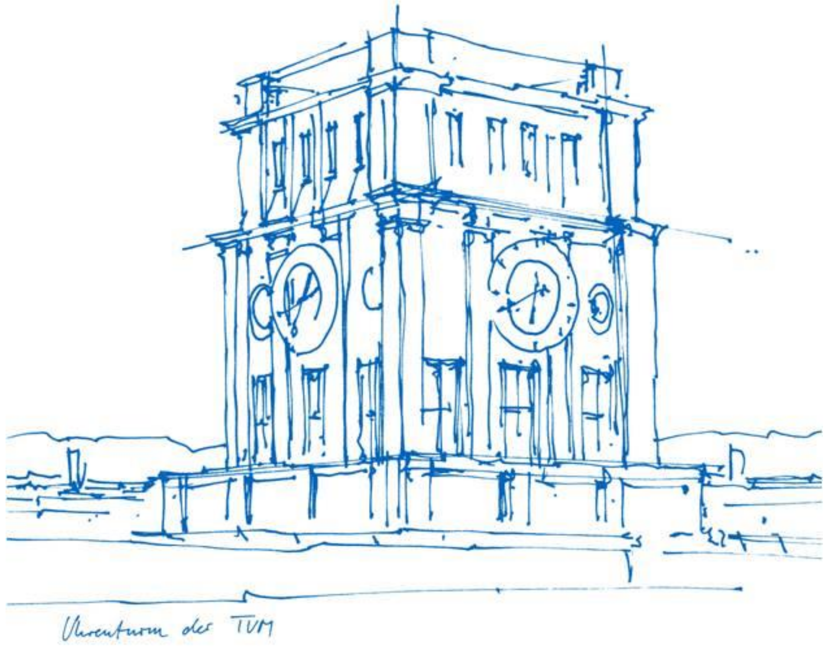
Neural Architecture Search with Reinforcement Learning

Recent Trends in Automated Machine Learning

Technische Universität München

Xiongyu Xie

May 26th, 2021



1. Motivation and problem statement

- Neural networks are powerful and widely used.
- A good model structure of NN will be beneficial.

- Many modern neural networks perform well/better only with specific structures, e.g., LSTM in RNN and skip connections in CNN
- **Those specific structures are difficult to design manually.**

2. Overview

- Generate model descriptions by the controller (RNN)
- Train the controller via Reinforcement Learning (**policy gradient ascent**) → maximize the prediction accuracy on a validation dataset.
- The generated models reach the same performance level as the previous state-of-the-art models.

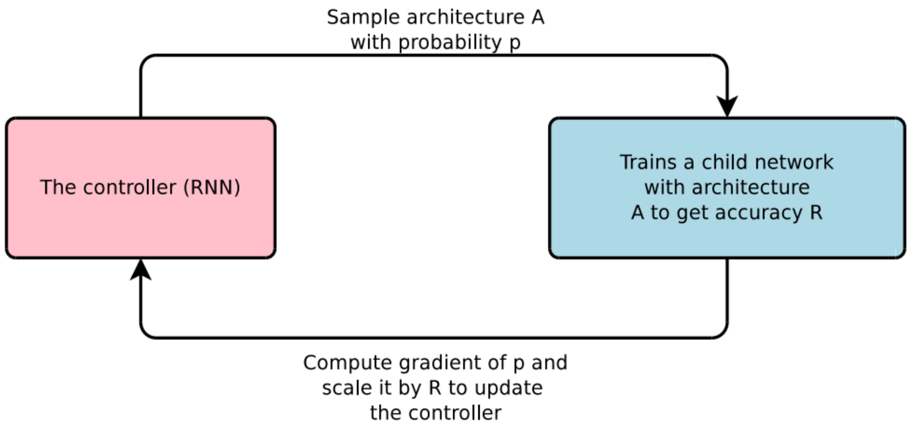
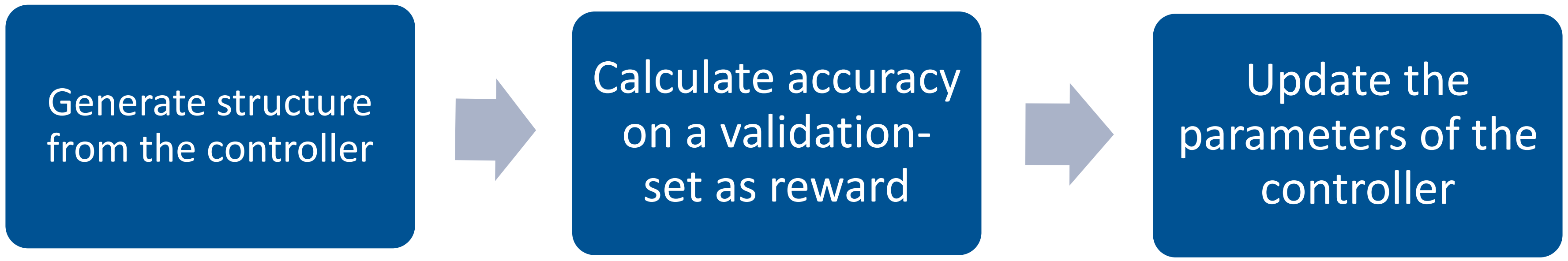

Sample architecture A with probability p
The controller (RNN)
Trains a child network with architecture A to get accuracy R
Compute gradient of p and scale it by R to update the controller

Figure: An overview of Neural Architecture Search

Figure from Barret Zoph and Quoc V. Le, ‘Neural Architecture Search with Reinforcement Learning‘, 2017.

3. The core idea behind each step

Working flows:


Generate structure from the controller
Calculate accuracy on a validation-set as reward
Update the parameters of the controller

- Generation of network structures in arbitrary length
- Stop criterion is a predefined layer number

- Generate, build and train many architectures
- At convergence, calculate an accuracy R from the networks

- RL with gradient ascent

3. The core idea behind each step

Working flows:

Generate structure from the controller

- Generation of network structures in arbitrary length
- Stop criterion is a predefined layer number

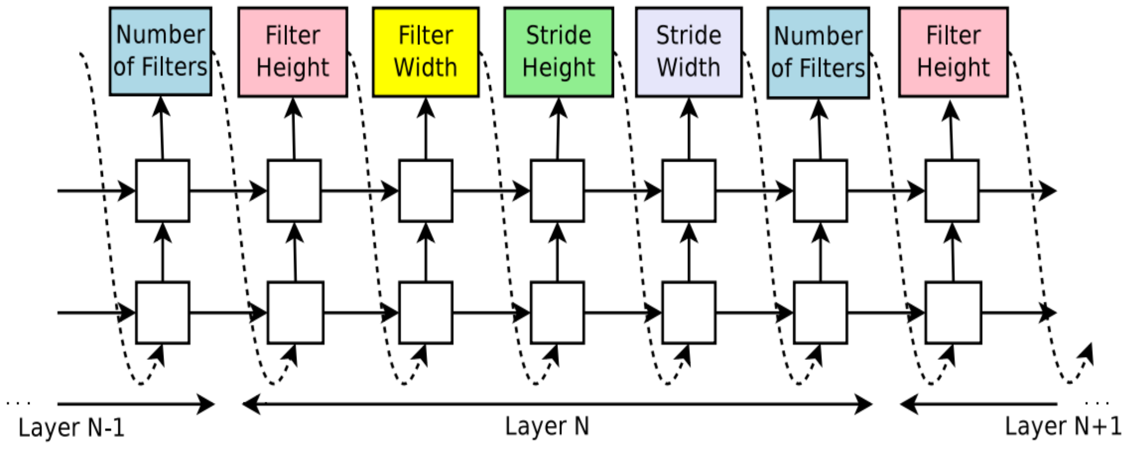
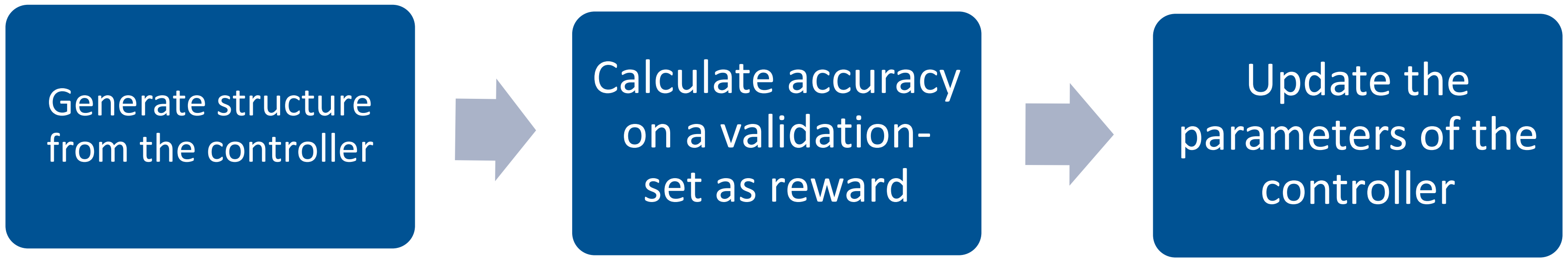


Figure: The controller samples a simple convolutional network

Figure from Barret Zoph and Quoc V. Le, 'Neural Architecture Search with Reinforcement Learning', 2017.

3. The core idea behind each step

Working flows:


Generate structure from the controller
Calculate accuracy on a validation-set as reward
Update the parameters of the controller

- Generation of network structures in arbitrary length
- Stop criterion is a predefined layer number

- Generate, build and train many architectures
- At convergence, calculate an accuracy R from the networks

- RL with gradient ascent

Training with REINFORCE

Objective function:

$$J(\theta) = E_{\tau\sim p_\theta(\tau)}[R(\tau)]$$

Where θ – the parameters of the controller, $\tau = a_{1:T}$ – the list of the outputs of the controller, R – the accuracy on a validation dataset.

The derivative of objective function (policy gradient):

$$\nabla_\theta J(\theta) = \sum_{t=1}^{T} E_{\tau\sim p_\theta(\tau)}[\nabla_\theta log P_\theta(a_t|a_{t-1:1}) R(\tau)]$$

Unbiased approximation for the above function:

$$\frac{1}{m}\sum_{k=1}^{m}\sum_{t=1}^{T} \nabla_\theta log P_\theta(a_t|a_{t-1:1})(R_k(\tau) - b)]$$

Where m – the number of sampled structures in one batch, T – the number of hyperparameters for a network, b – an exponential moving average of the previous architecture accuracies.

4. Generate CNN with more complex skip connections

Enable RNN to predict skip function

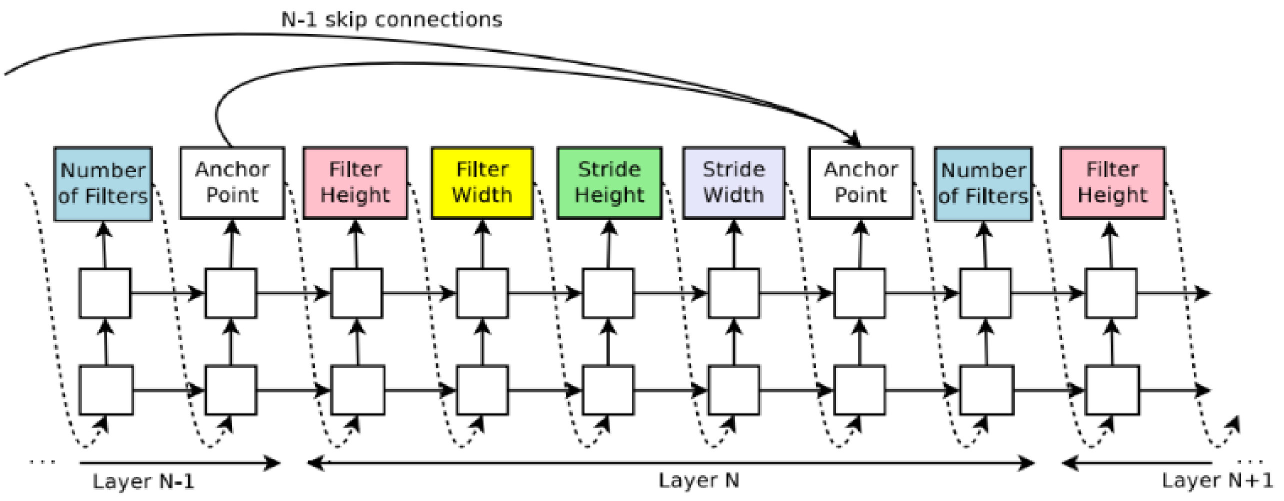


Figure: Adding anchor points to form skip connections

- Add **an anchor point** (a set-selection type attention)
- Sample outputs from previous layers as the input for current layer:

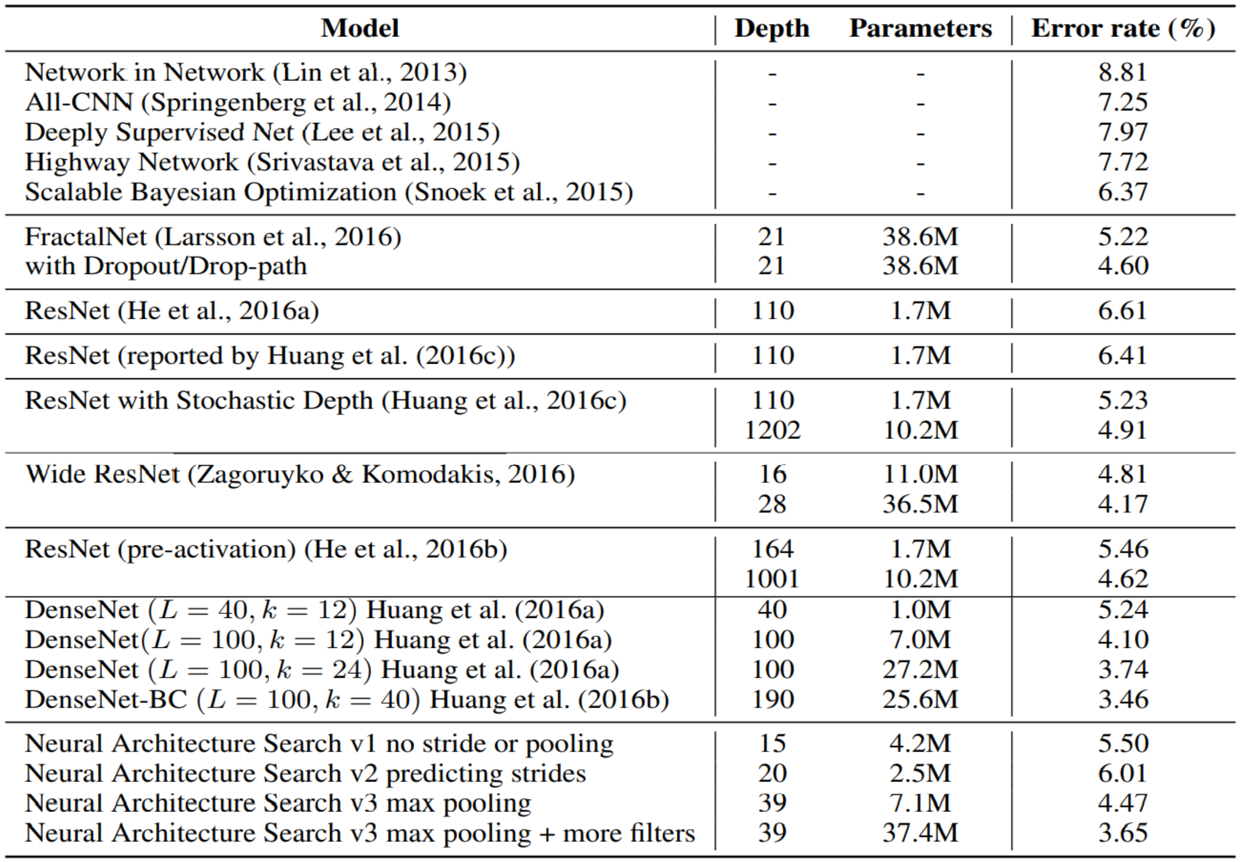
$$\mathrm{P}(\text{Layer j is an input to layer i}) = \mathrm{sigmoid}(v^{\mathrm{T}}\tanh(W_{prev} * h_j + W_{curr} * h_i)),$$

Results of CNN generation for CIFAR-10

Training details:

1. The controller: two-layer LSTM with 35 hidden units on each layer (smaller than the generated neural networks).
2. Each child model is constructed and trained for 50 epochs.
3. Starting from 6-layer CNN, the controller increases the depth by two after every 1.600 samples of child networks.

Results of CNN generation for CIFAR-10



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

Table from Barret Zoph and Quoc V. Le, ‘Neural Architecture Search with Reinforcement Learning‘, 2017.

Results of CNN generation for CIFAR-10

Conclusions:

1. Classification accuracy can be as high as the state-of-art models.
2. Compared to the models with similar accuracy, the models from this paper have less layers.
3. The computation time of last model (40 filters added) is 1.05x as fast as the DenseNet that achieves 3.74 %

5. Generate RNN cell structures

Basic idea: combine the given nodes while following **tree structure**
→ The controller (RNN) generates combinations (i.e., addition) and activation functions, and the connection between cell variables C_t, C_{t-1} and the other nodes.

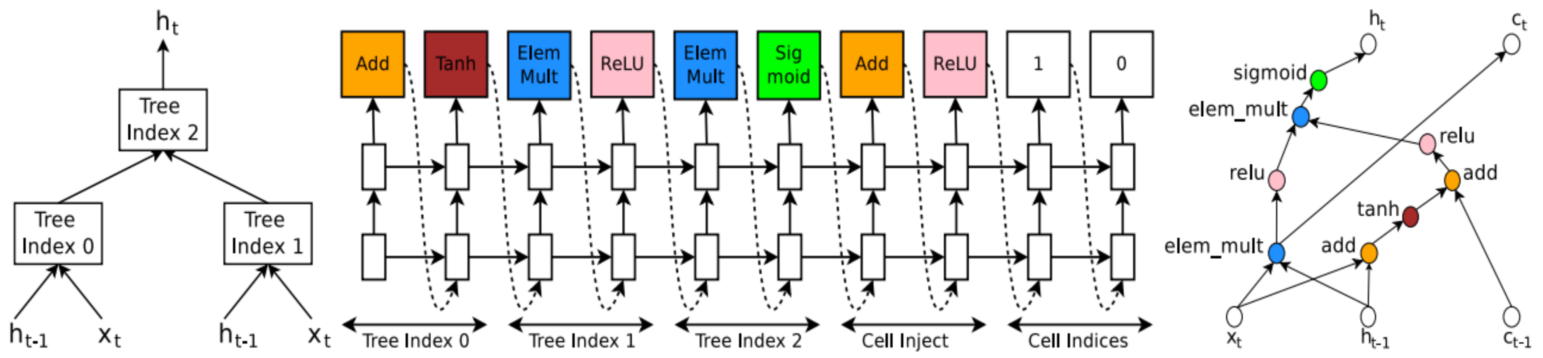


Figure: The example of a recurrent cell constructed from a tree

Figure from Barret Zoph and Quoc V. Le, 'Neural Architecture Search with Reinforcement Learning', 2017.

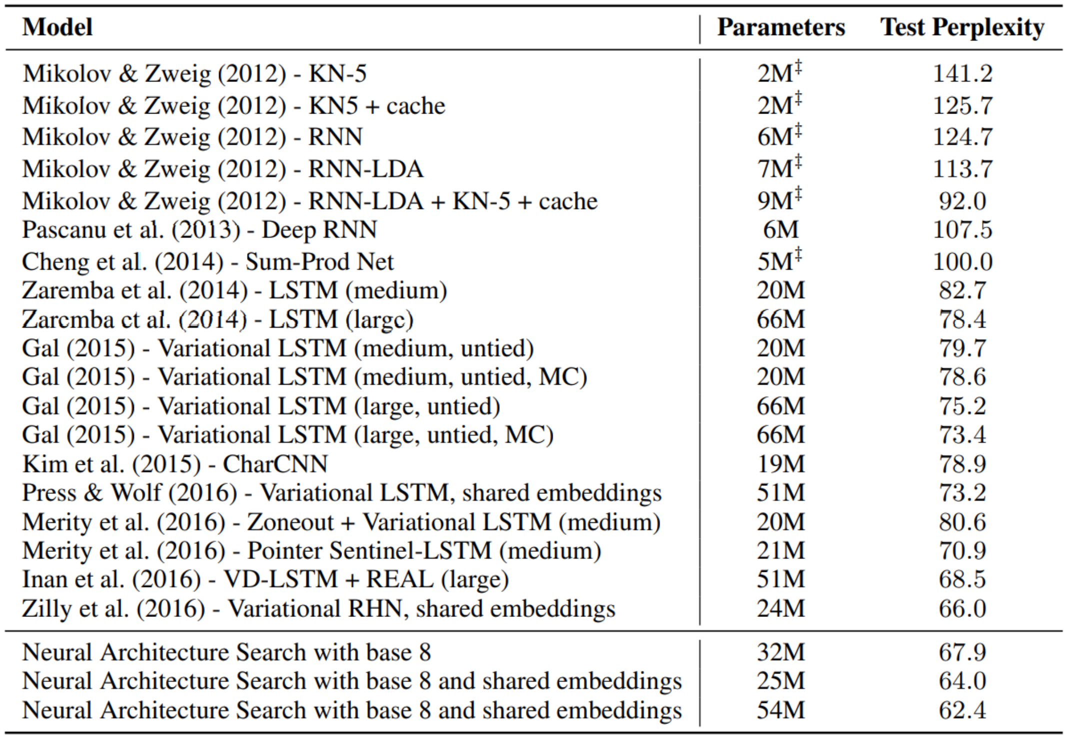
Results of cell structures generation (for Penn Treebank dataset)

Penn Treebank dataset is one of the most known and used corpus for the evaluation of models for sequence labelling.

Training details for cell structures generation:

- Base number: the number of input pairs, set to 8
- Only predict **the RNN cell structures** and fix all other parameters.
- After choosing the best cell structure, do grid search over learning rate, weight initialization etc.

Results of cell structures generation (for Penn Treebank dataset)



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
Zarcmba ct al. (2014) - LSTM (largc)	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: Single model perplexity on the test set of the PTB language modeling tasks

Conclusions:

1. The models found by this paper outperform the other models
2. The model (with 64 perplexity) is more than **two times faster** than the previous best network.

Table from Barret Zoph and Quoc V. Le, ‘Neural Architecture Search with Reinforcement Learning‘, 2017.

6 Small validation experiments

1. Transfer learning:
Applying the generated RNN cell structure from previous tasks to the character language modeling task on the same dataset → Result is better than LSTM cell

2. Adding more functions in the search space:
The model can achieve comparable performance

3. Comparison against random search:
Not only the best model from this paper is better, but also the average of top models is much better.

Conclusions

1. This paper introduces an idea of using RNN to **compose neural network architectures**, and the generated structures from this paper can even outperform some state-of-the-art models.
2. RNN makes it possible to search in different variable-length architecture spaces and different types of structures.
3. The generated structures from this methods can be generalized to other tasks.

Outlook:

1. This method is flexible, but still need to follow some patterns (**predefined recurrent structure**).
2. The child networks are always built and trained for 50 epochs, regardless of their structures.

Thanks for your attention!