Neural architecture search with reinforcement learning

Barret Zoph, Quoc V. Le
Google Brain
ICLR 2017 (Oral)

Introduction

- In this paper, the authors use RNN to automatically generate hyperparameters and structures of neural networks.

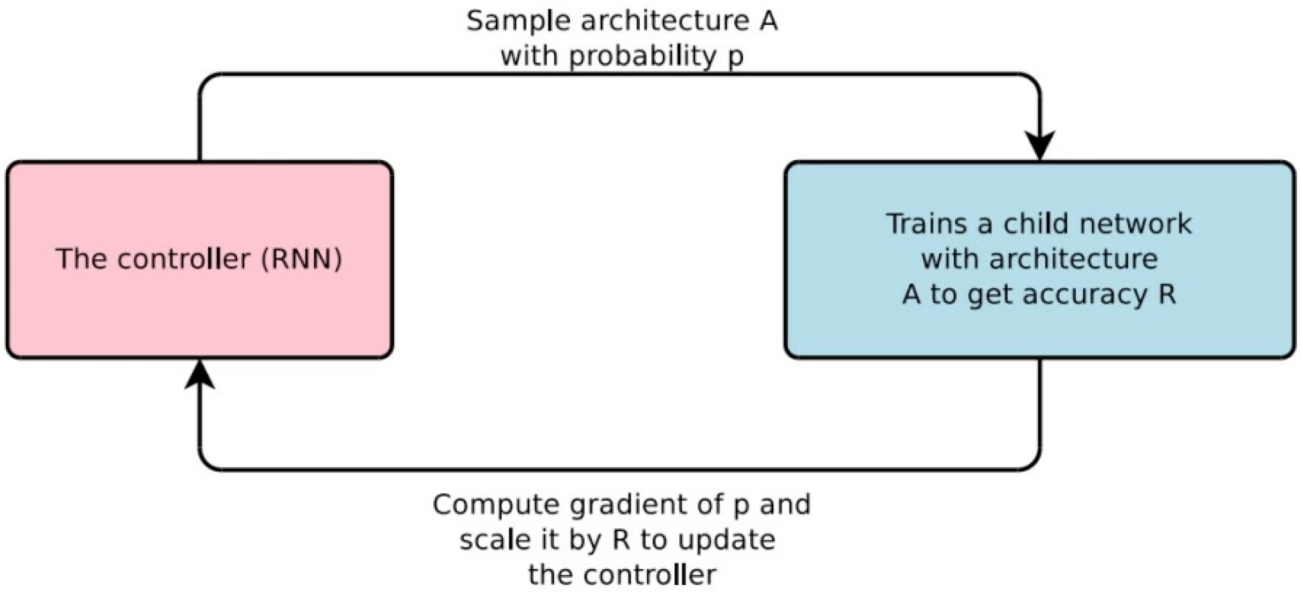
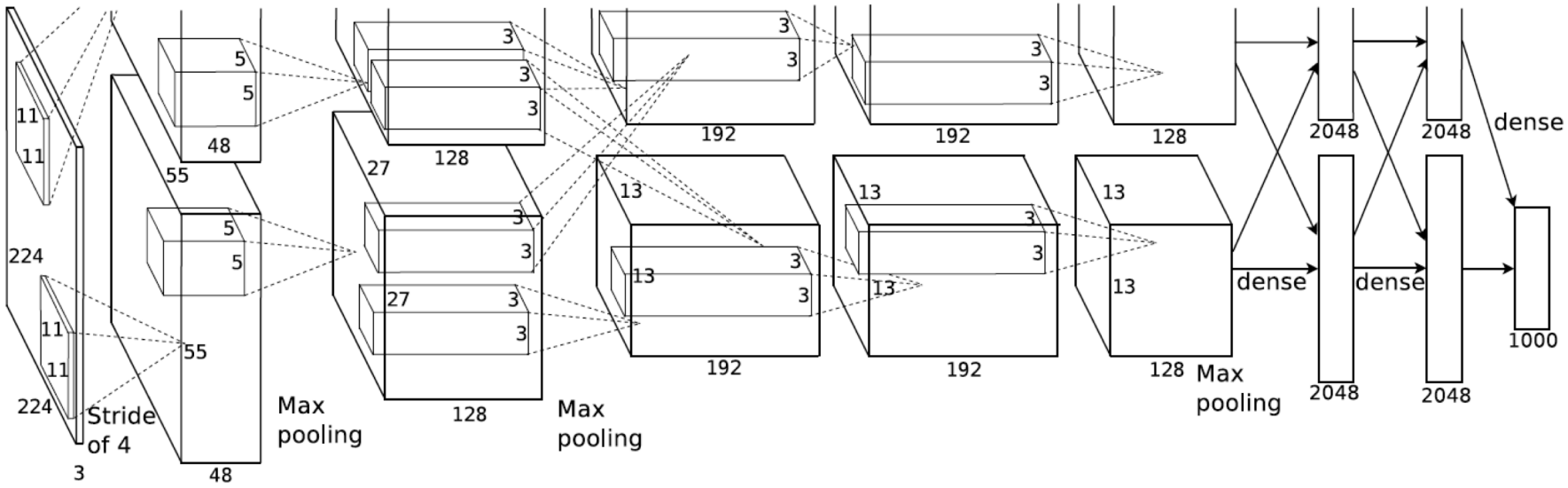
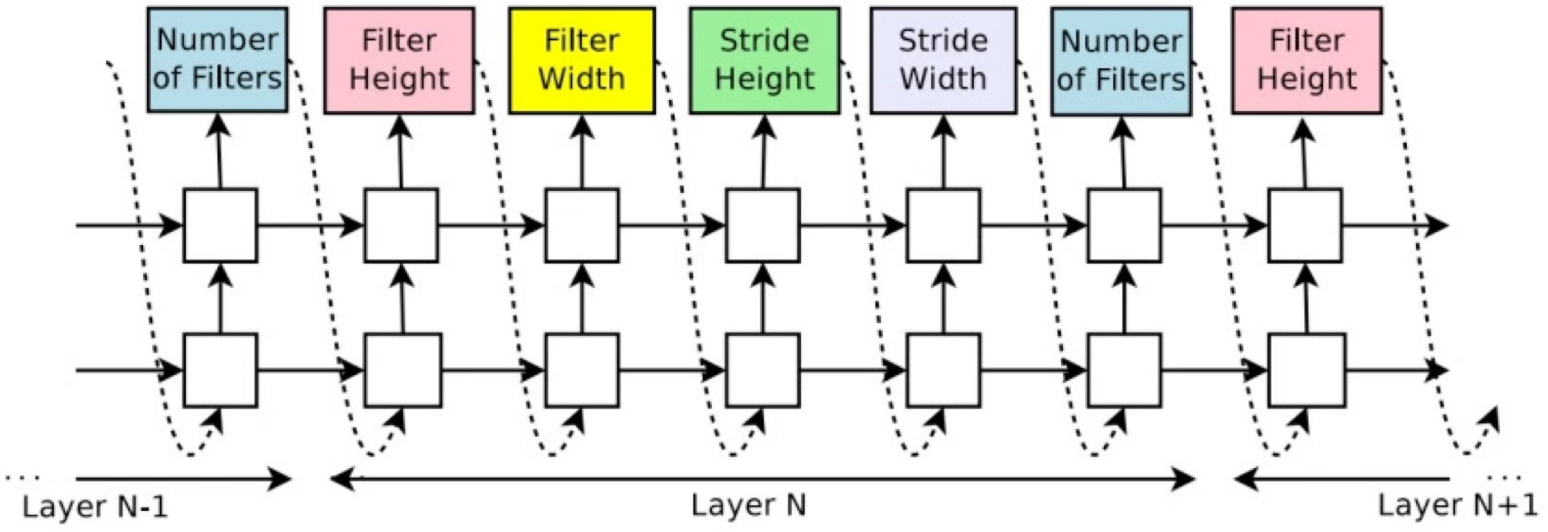


Figure 1: An overview of Neural Architecture Search.

CNN example --- AlexNet



Generate hyperparameters of CNN



How to update the hidden state parameters in RNN

- Using REINFORCE rule (Williams 1992)

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

- *m* is the number of different architectures that the controller samples in one batch.
- *T* is the number of hyperparameters the controller must predict to design an architecture.
- *b* is a baseline function (exponential moving average of accuracy)

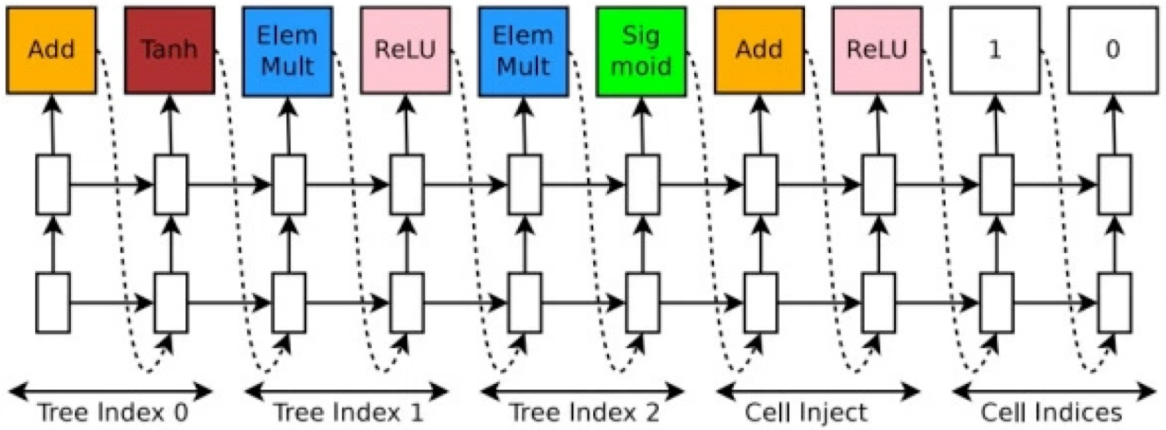
How to update the hidden state parameters in RNN

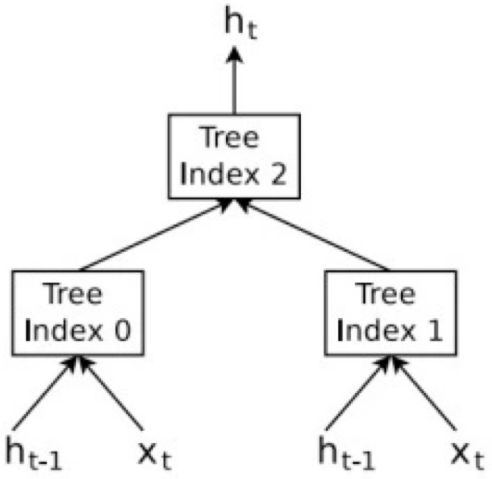
- Using REINFORCE rule (Williams 1992)

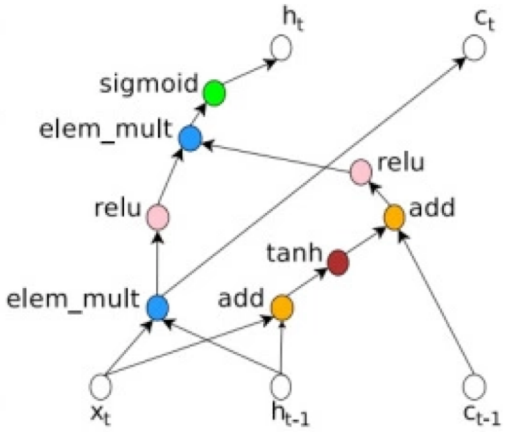
$$\frac{1}{m}\sum_{k=1}^{m}\sum_{t=1}^{T}\nabla_{\theta_c}\log P(a_t|a_{(t-1):1};\theta_c)(R_k-b)$$

- θ_c is the parameters of hidden states in RNN
- R_k is the accuracy of the given construction

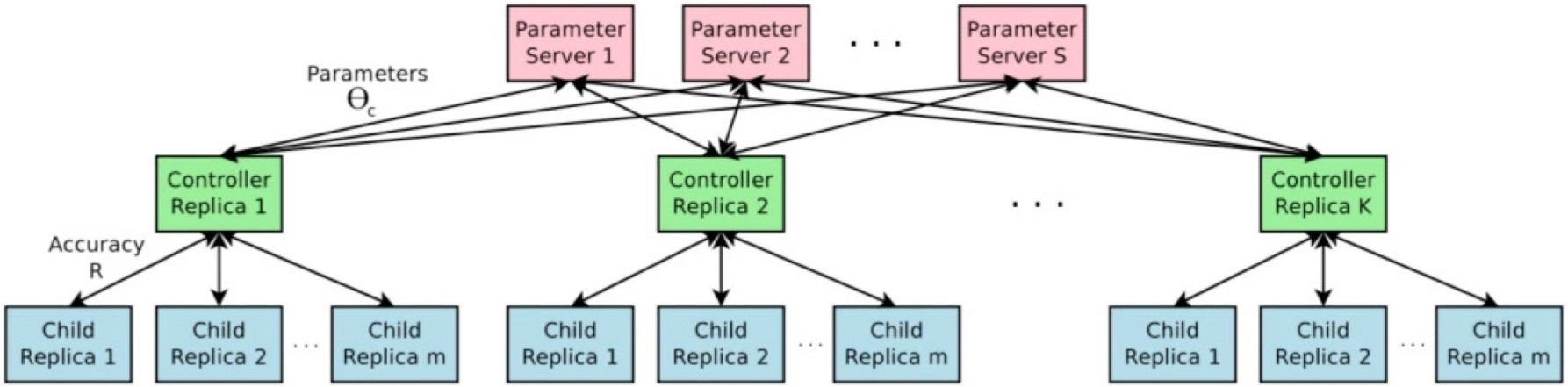
Generate hyperparameters of RNN







Accelerating training



- 20 parameter servers;
- 100 controller replica;
- 8 child replica;
- trained on 800 GPUs concurrently at any time

Experiment Result

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.

Comparing with random search

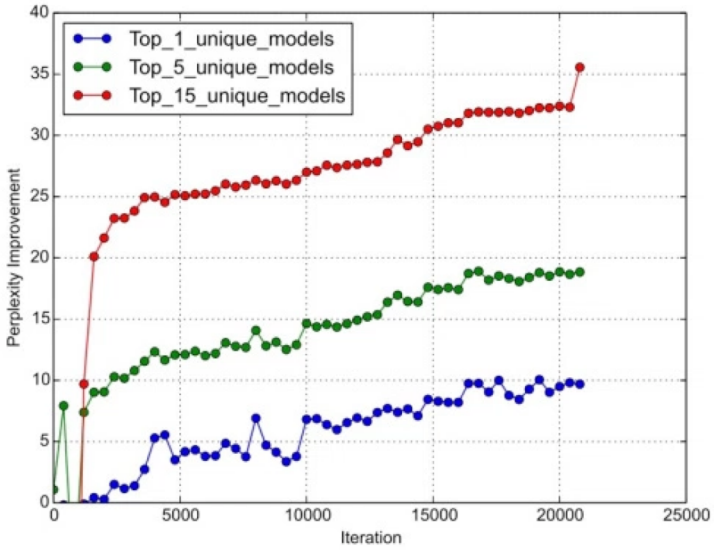


Figure 6: Improvement of Neural Architecture Search over random search over time. We plot the difference between the average of the top k models our controller finds vs. random search every 400 models run.

Takeaways

- Reinforcement learning updates (i.e., policy gradient) can be used for RNN models (i.e., the weights parameters of hidden states).
- In this paper, the controller (RNN) does not only generate the hyperparameters of CNNs (e.g., filter width, stride size) but also generate different structures (i.e., different modules) in RNNs
- It is indeed a computation intensive task to automatically find neural networks performed well.

Thanks!