NEURAL ARCHITECTURE SEARCH WITH REINFORCEMENT LEARNING

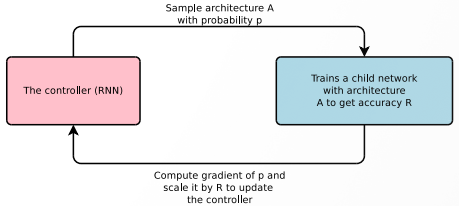
Barret Zoph,* **Quoc V. Le**
Google Brain
{barretzoph,qvl}@google.com

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.



Neural Architecture Search with Reinforcement Learning

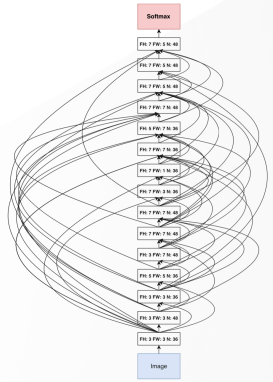
Zoph, B., & Le, Q. V. (2016). Neural Architecture Search with Reinforcement Learning. Retrieved from http://arxiv.org/abs/1611.01578

Computer Vision Lab, Hanyang University
Paper review, 8 Jul 2019
Jihun Kim

About the Paper

Introduction

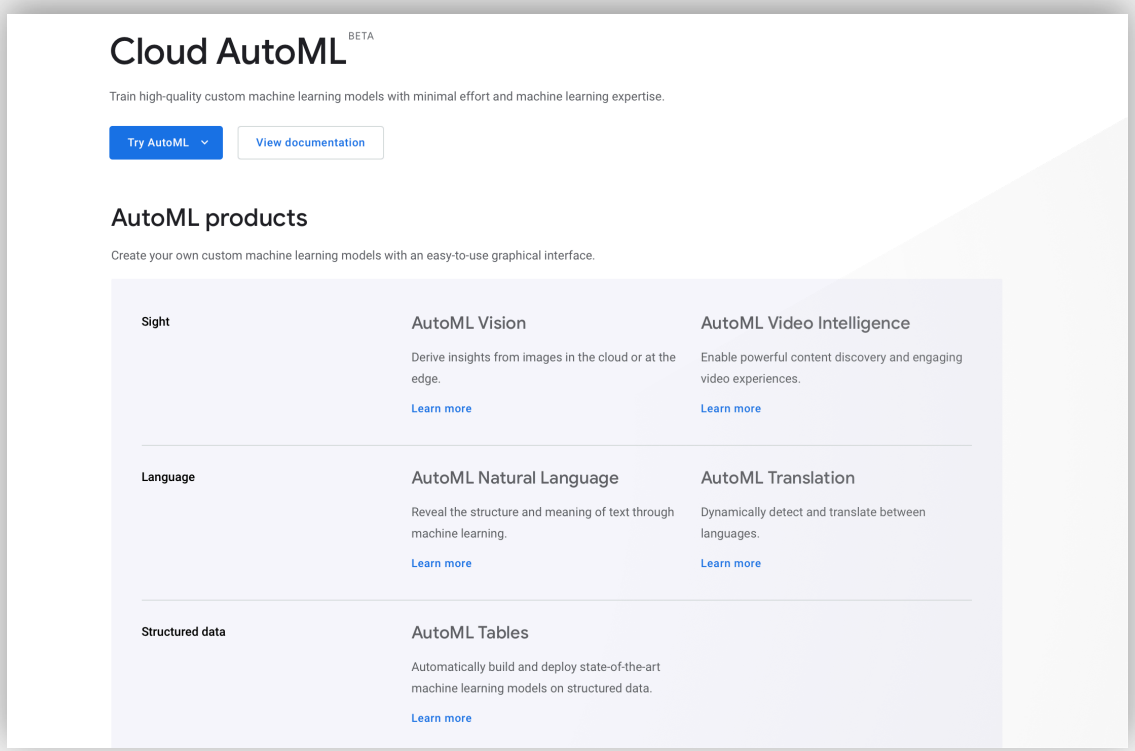
- NAS (Neural Architecture Search)
  - Advancing this trend, NAS has emerged as a promising direction for jointly searching wiring patterns and which operations to perform.
  - However, like the wiring patterns in ResNet and DenseNet, the NAS network generator is hand designed and the space of allowed wiring patterns is constrained in a small subset of all possible graphs.



Convolutional architecture generated by NAS

5

"NAS has emerged as a promising direction for jointly searching wiring patterns and which operations to perform."

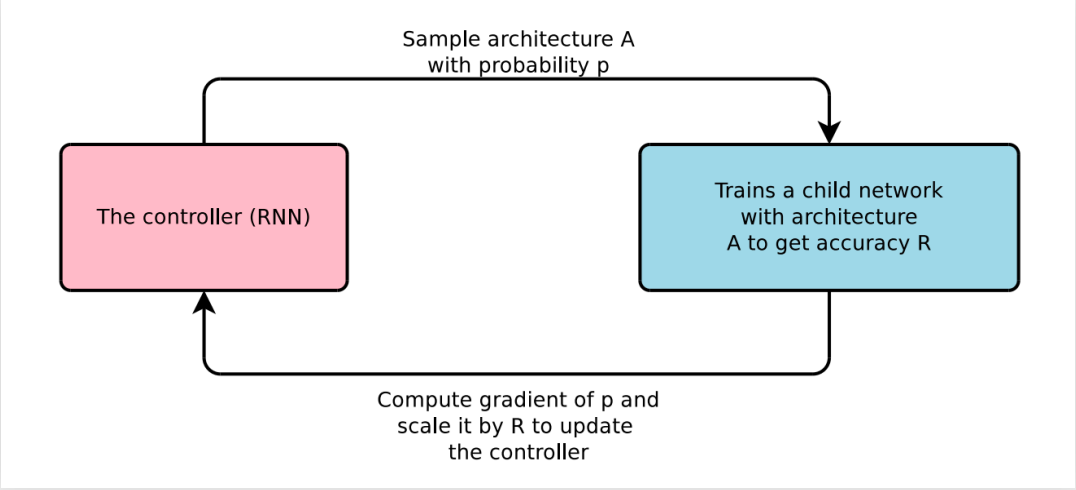


"Train custom machine learning models that are specific to your business needs, with minimum effort and machine learning expertise."

Introduction 👀

The last few years have seen much success of deep neural networks.
Although it has become easier, designing architectures still requires a lot of expert knowledge and takes ample time.

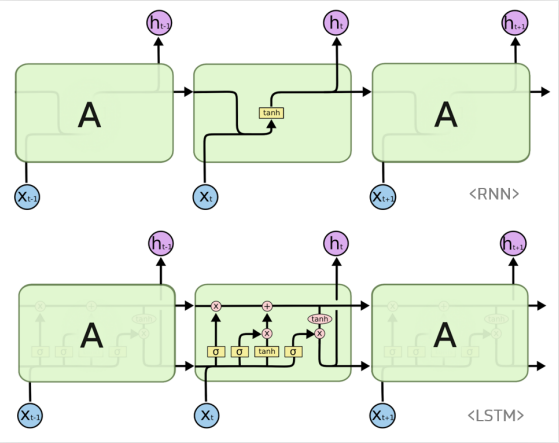
This paper presents Neural Architecture Search, a gradient-based method for finding good architectures.



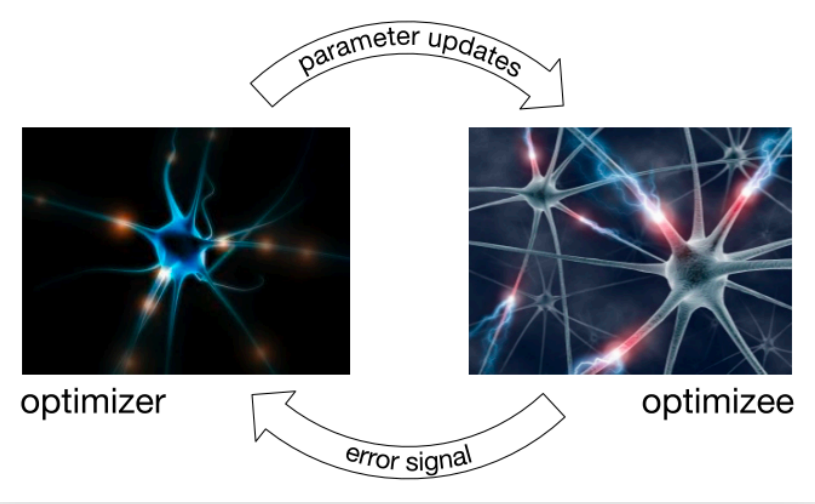
An overview of Neural Architecture Search.

Related Work

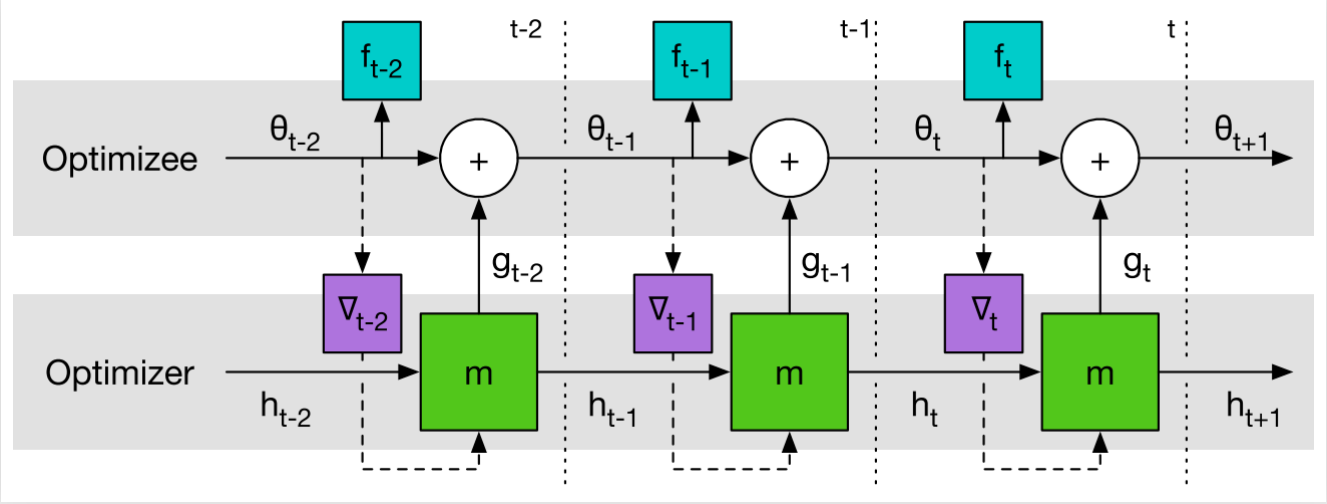
"Learning to Learn by Gradient Descent by Gradient Descent"



Graphic representation of basic RNN and LSTM.



The optimizer is provided with performance of the optimizee and proposes updates to increase the optimizee's performance.



Computational graph used for computing the gradient of the optimizer.

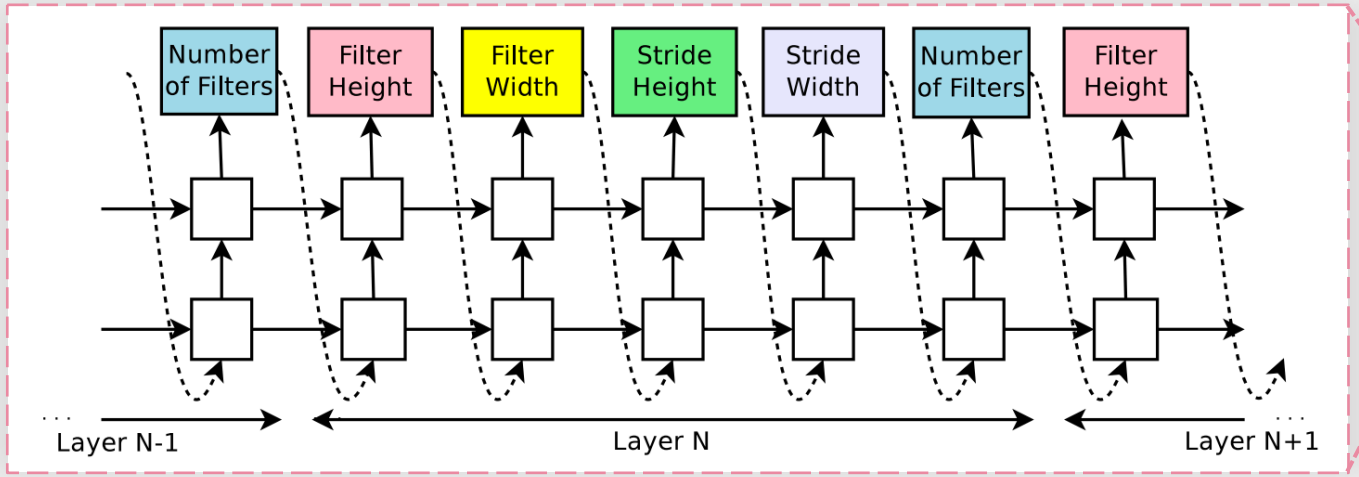
Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., ⋯ de Freitas, N. (2016). Learning to learn by gradient descent by gradient descent. Retrieved from http://arxiv.org/abs/1606.04474

Generate Model Descriptions with a Controller

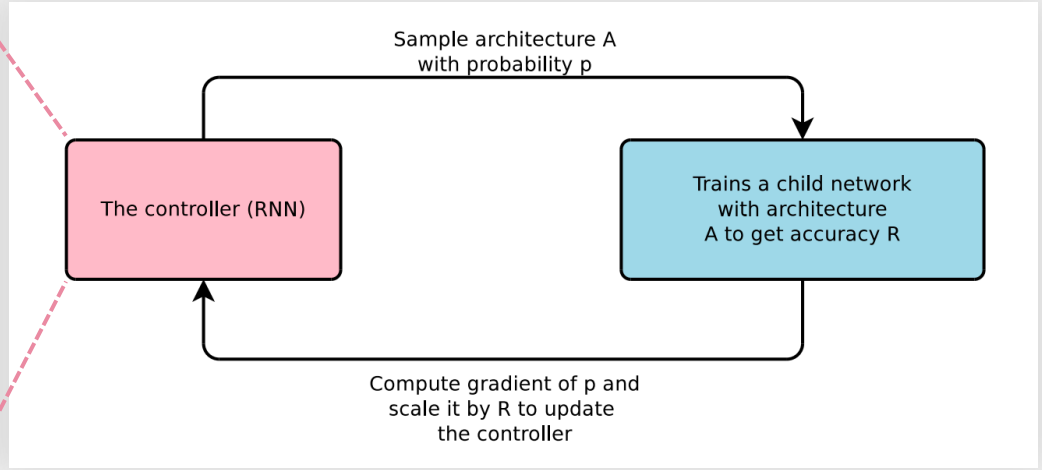
Use a controller to generate architectural hyperparameters of neural networks.
The controller is implemented as a recurrent neural network.

Once the controller RNN finishes generating an architecture, a neural network is built and trained.
At convergence, the accuracy of the network on a held-out validation set is recorded.

The parameters of the controller RNN are then optimized.



How controller recurrent neural network samples a simple convolutional network.



An overview of Neural Architecture Search.

Training with REINFORCE 💪

Ask controller to maximize its expected reward, represented by:

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

Since the reward signal R is non-differentiable, we need to use a policy gradient method to iteratively update θ_c. In this work, we use the REINFORCE rule from Williams (1992):

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

An empirical approximation of the above quantity is:

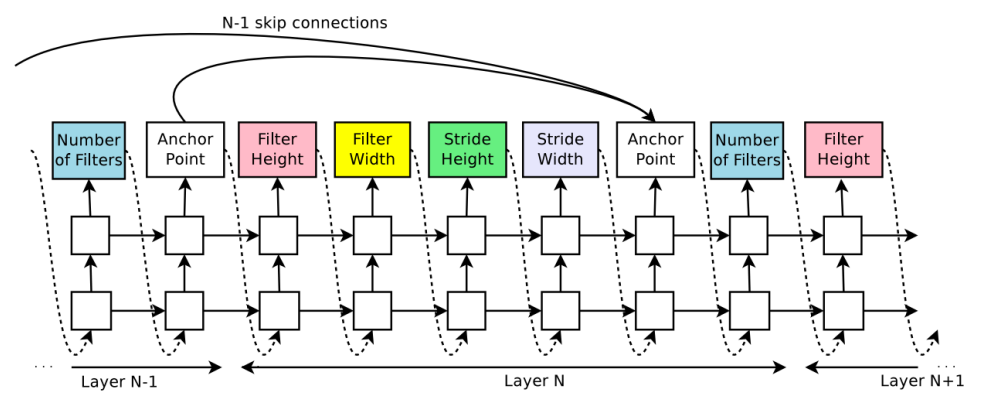
$$\frac{1}{m}\sum_{k=1}^{m}\sum_{t=1}^{T} \nabla_{\theta_c} \log P(a_t|a_{(t-1):1};\theta_c)R_k$$

Increasing Architecture Complexity

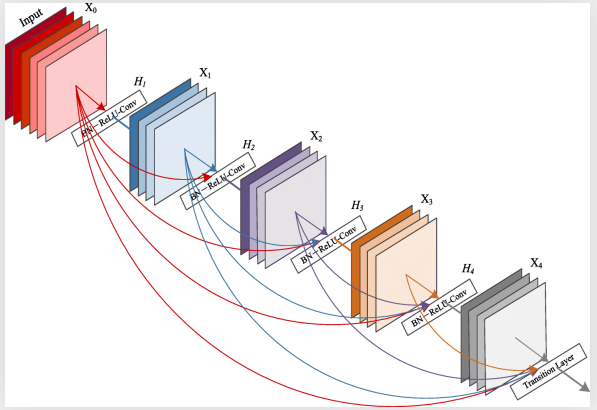
Introduce a method that allows the controller to propose skip connections or branching layers.

At layer N, we add an anchor point which has $N - 1$ content-based sigmoids to indicate the previous layers that need to be connected.

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



The controller uses anchor points, and set-selection attention to form skip connections.



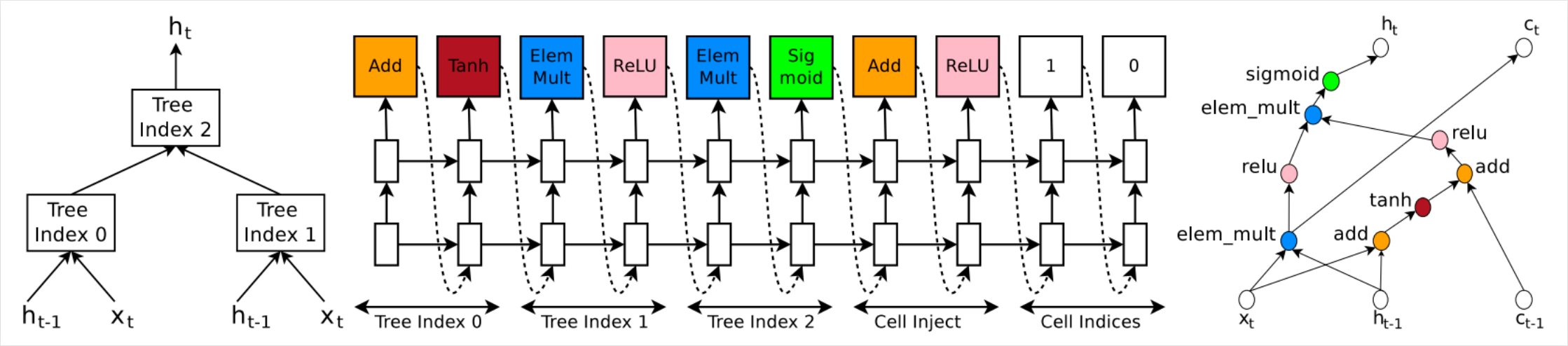
Graphic representation of DenseNet.

Generate Recurrent Cell Architectures 🏭

At every time step t, the controller needs to find a functional form for h_t that takes x_t and h_{t-1} as inputs.

The computations for basic RNN and LSTM cells can be generalized as a tree of steps that take x_t and h_{t-1} as inputs and produce h_t as final output.

We also need cell variables c_{t-1} and c_t to represent the memory states.



An example of a recurrent cell constructed from a tree that has two leaf nodes (base 2) and one internal node.

Experiment: Convolutional Architecture 🧪

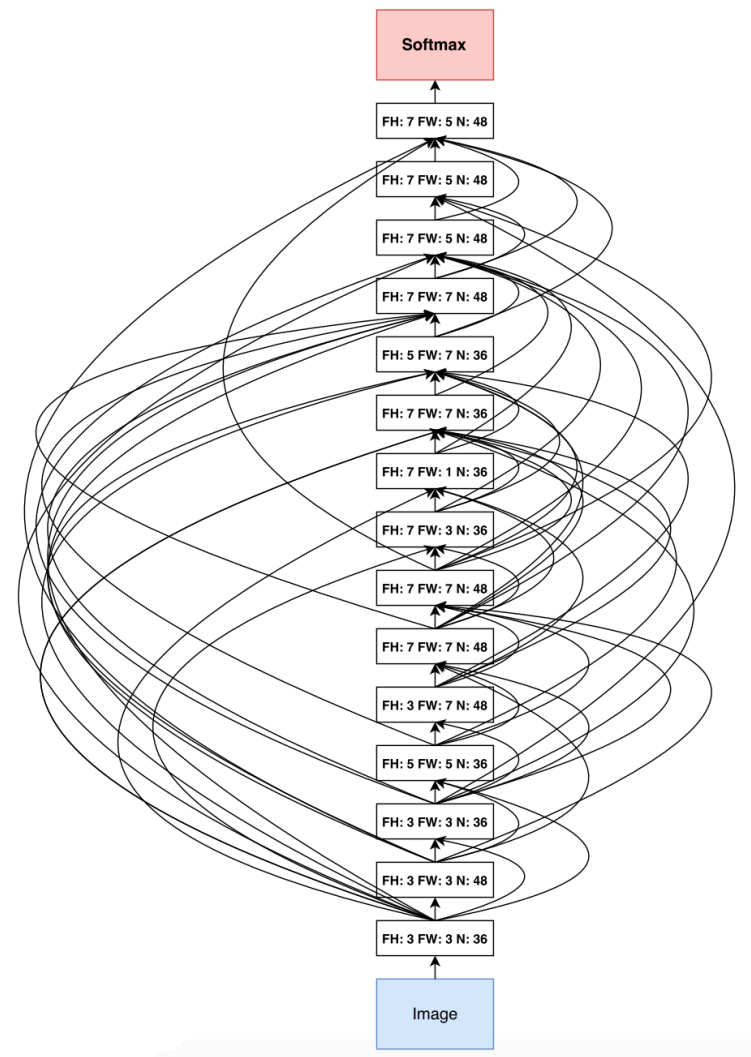
Dataset
CIFAR-10 dataset with
data preprocessing and augmentation.

Result
After the controller trains 12,800 architectures, we find the architecture that achieves the best validation accuracy.

We then run a small grid search over learning rate, weight decay, batchnorm epsilon and what epoch to decay the learning rate.

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

Appendix 📎


Softmax
FH: 7 FW: 5 N: 48
FH: 7 FW: 5 N: 48
FH: 7 FW: 5 N: 48
FH: 7 FW: 7 N: 48
FH: 5 FW: 7 N: 36
FH: 7 FW: 7 N: 36
FH: 7 FW: 1 N: 36
FH: 7 FW: 3 N: 36
FH: 7 FW: 7 N: 48
FH: 7 FW: 7 N: 48
FH: 3 FW: 7 N: 48
FH: 5 FW: 5 N: 36
FH: 3 FW: 3 N: 36
FH: 3 FW: 3 N: 48
FH: 3 FW: 3 N: 36
Image

Problems 😞

Used 800+ GPUs to train

Used CIFAR-10 dataset: Image size is small (32x32), dataset size is small (60,000).

References

http://arxiv.org/abs/1611.01578
https://www.slideshare.net/KihoSuh/neural-architecture-search-with-reinforcement-learning-76883153
https://www.youtube.com/watch?v=XP3vyVrrt3Q