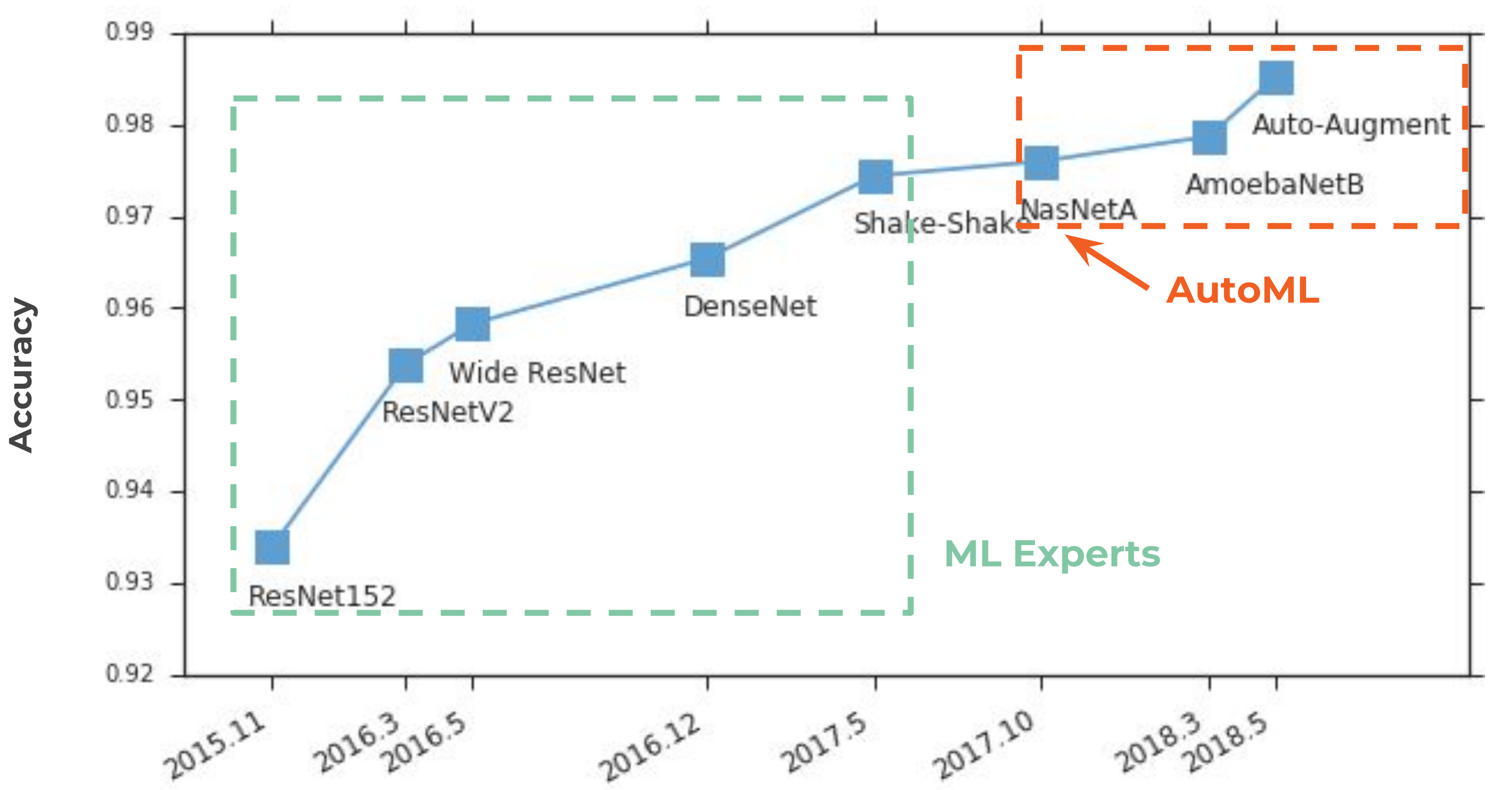
AutoML: Automated Machine Learning

Barret Zoph, Quoc Le

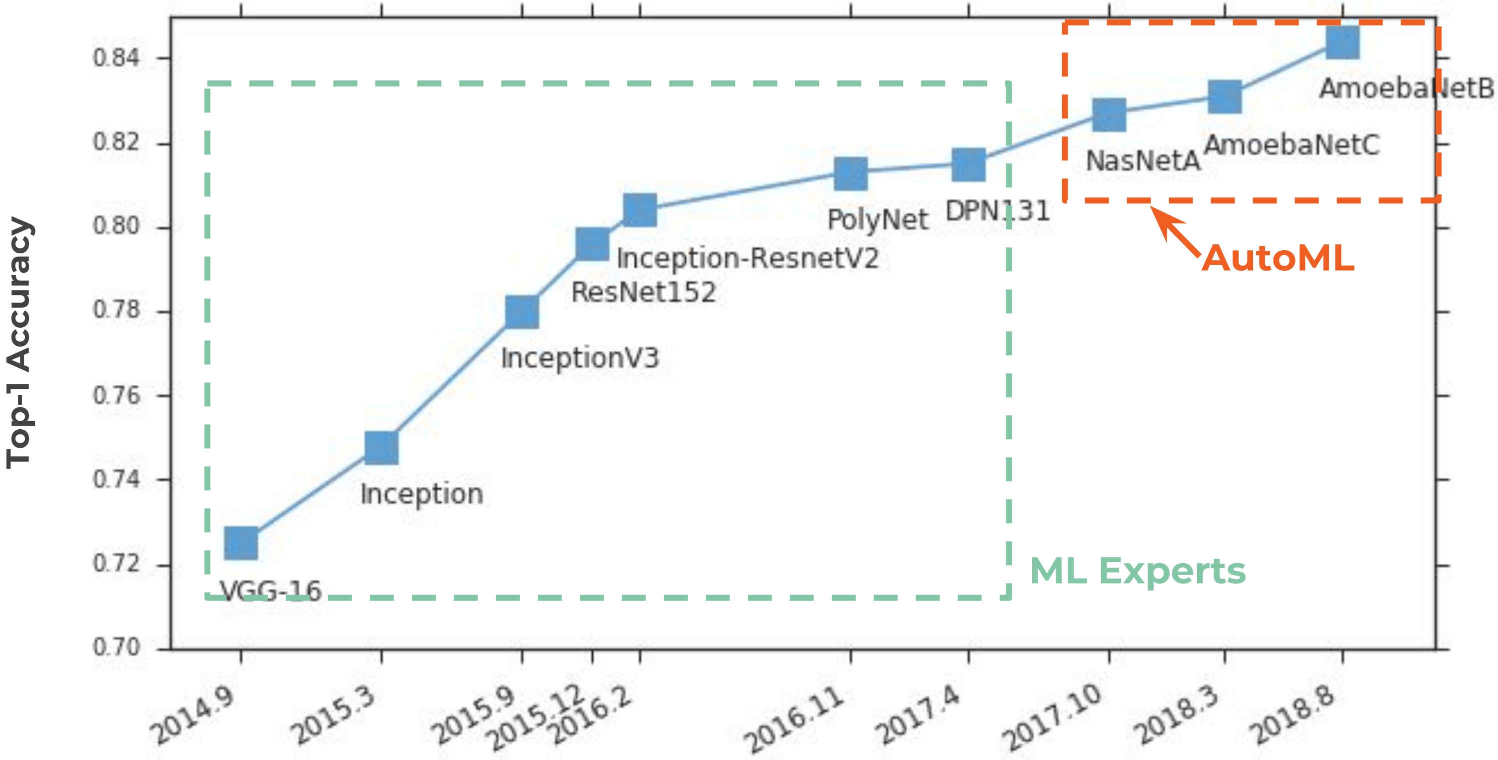
Thanks:
Google Brain team

CIFAR-10



Accuracy
0.99
0.98
0.97
0.96
0.95
0.94
0.93
0.92
ResNet152
ResNetV2
Wide ResNet
DenseNet
Shake-Shake
NasNetA
AmoebaNetB
Auto-Augment
AutoML
ML Experts
2015.11
2016.3
2016.5
2016.12
2017.5
2017.10
2018.3
2018.5

ImageNet



Google

Current:

Solution =

ML Expertise + Data + Computation

Current:

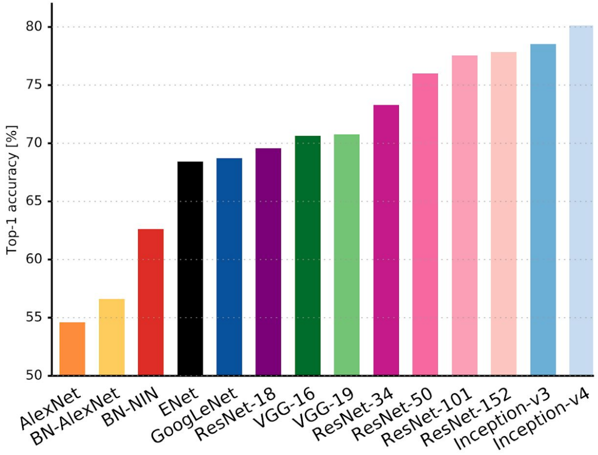
Solution =
ML Expertise + Data + Computation

But can we turn this into:

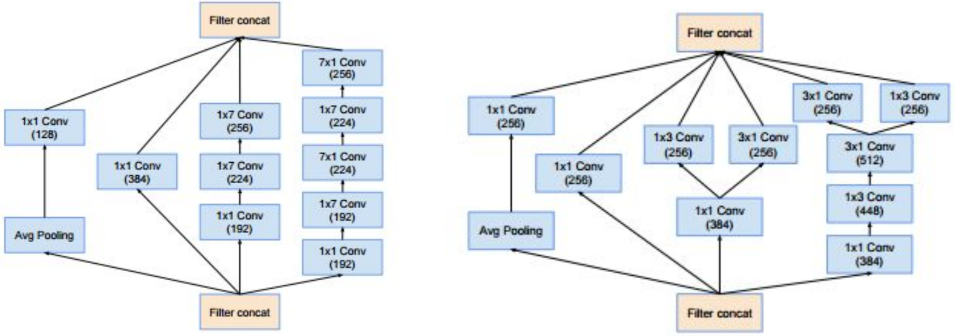
Solution =
Data + 100X Computation

Importance of architectures for Vision

- Designing neural network architectures is hard
- Lots of human efforts go into tuning them
- There is not a lot of intuition into how to design them well
- Can we try and learn good architectures automatically?

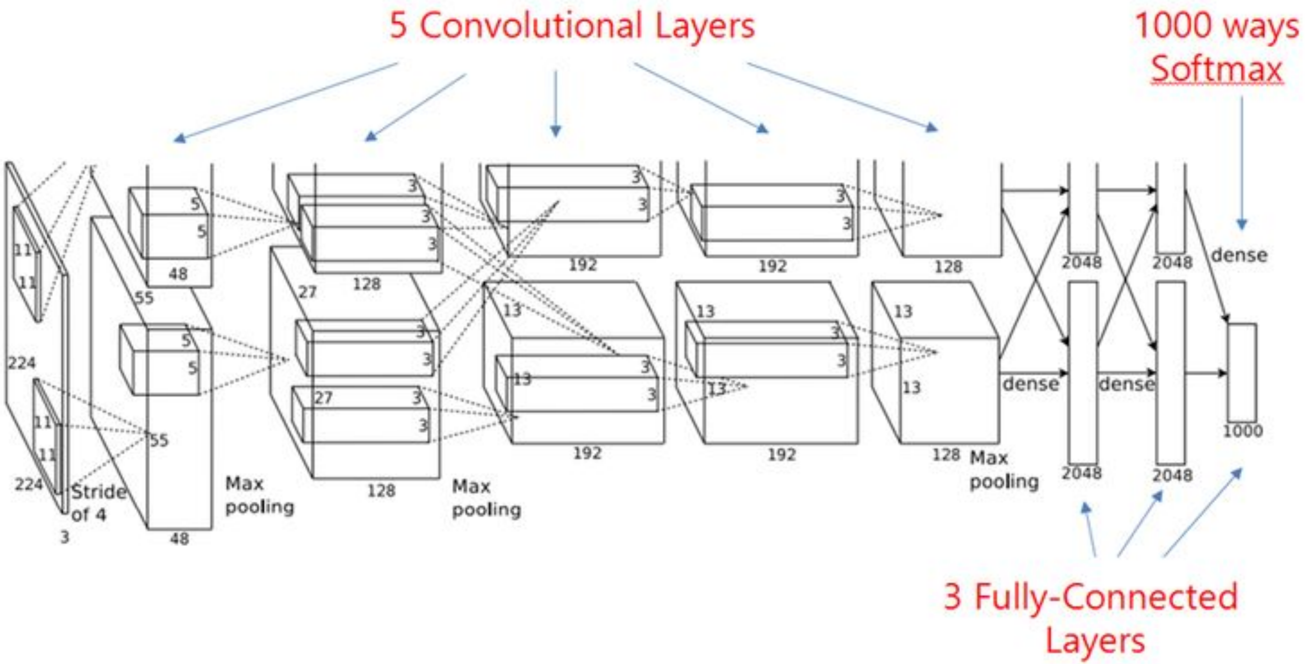


Canziani et al, 2017



Two layers from the famous Inception V4 computer vision model.
Szegedy et al, 2017

Convolutional Architectures

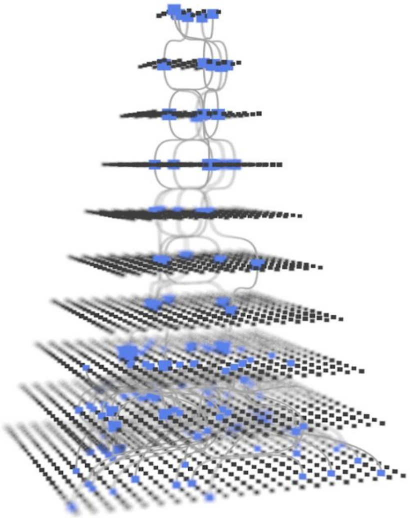


Krizhevsky et al, 2012

Neural Architecture Search

- Key idea is that we can specify the structure and connectivity of a neural network by using a configuration string
  - [“Filter Width: 5”, “Filter Height: 3”, “Num Filters: 24”]
- Our idea is to use a RNN (“Controller”) to generate this string that specifies a neural network architecture
- Train this architecture (“Child Network”) to see how well it performs on a validation set
- Use reinforcement learning to update the parameters of the Controller model based on the accuracy of the child model

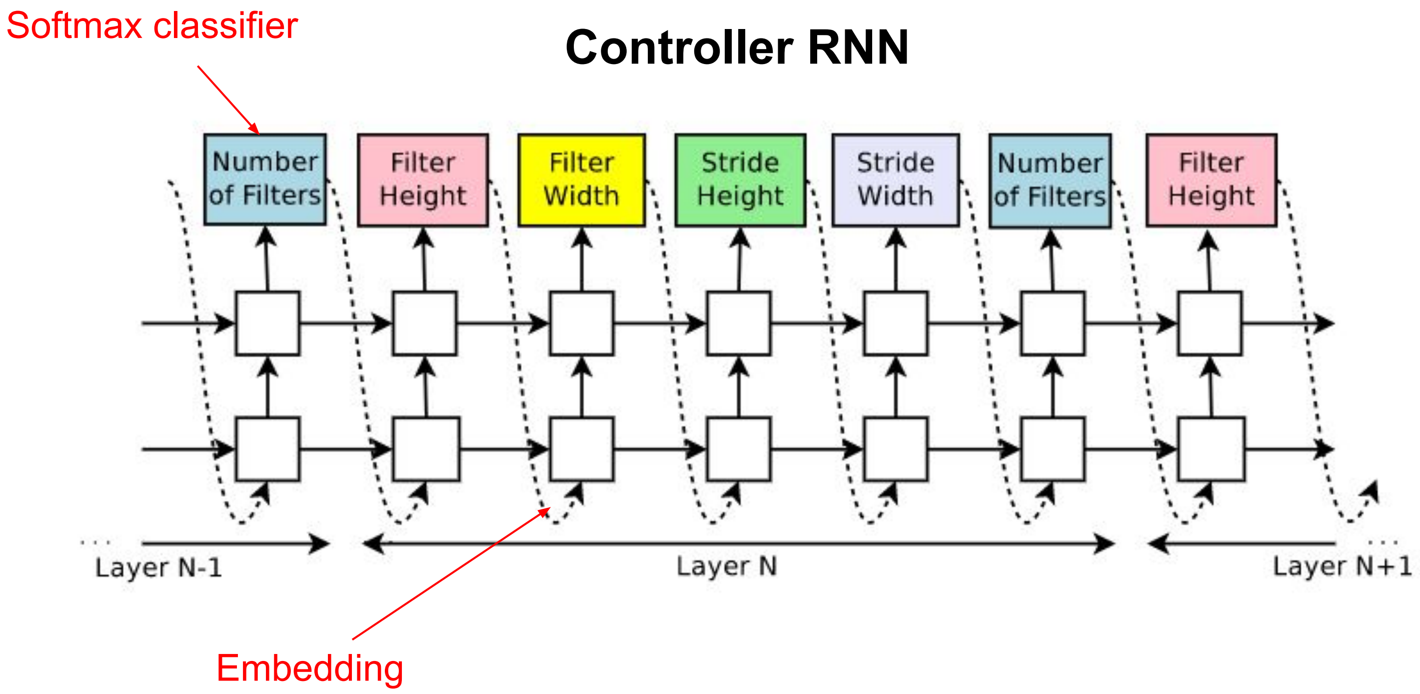
Controller: proposes ML models



Iterate to find the most accurate model

Train & evaluate models

Neural Architecture Search for Convolutional Networks



Training with REINFORCE

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

Google

Training with REINFORCE

Parameters of Controller RNN

Accuracy of architecture on held-out dataset

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

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

Training with REINFORCE

Parameters of Controller RNN

Accuracy of architecture on held-out dataset

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

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

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

Training with REINFORCE

Parameters of Controller RNN

Accuracy of architecture on
held-out dataset

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

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

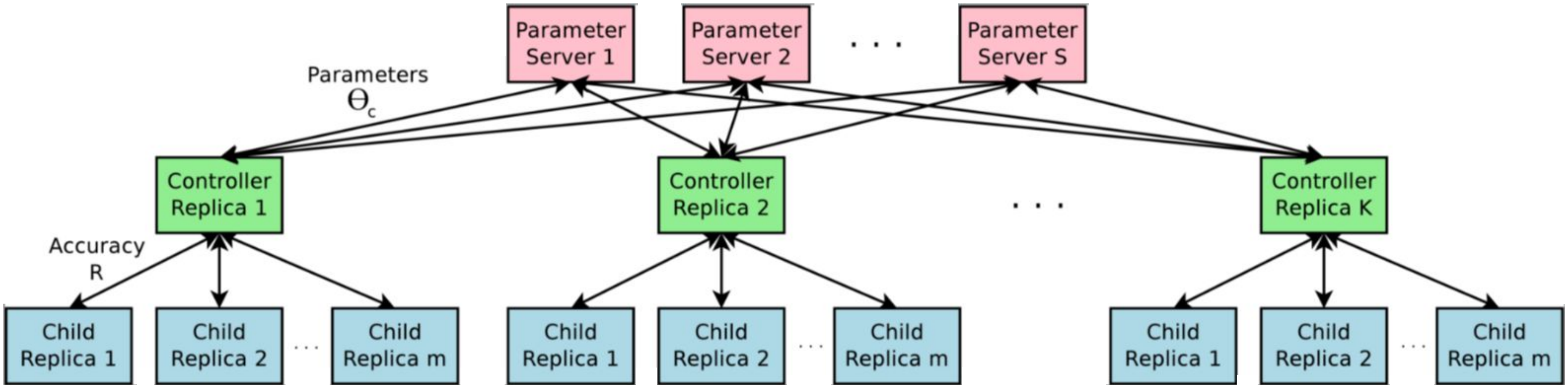
$$\nabla_{\theta_c} J(\theta_c) = \sum_{t=1}^{T} E_{P(a_{1:T};\theta_c)} \left[\nabla_{\theta_c} \log P(a_t | a_{(t-1):1}; \theta_c) R \right]$$

Number of models in minibatch

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

Google

Distributed Training



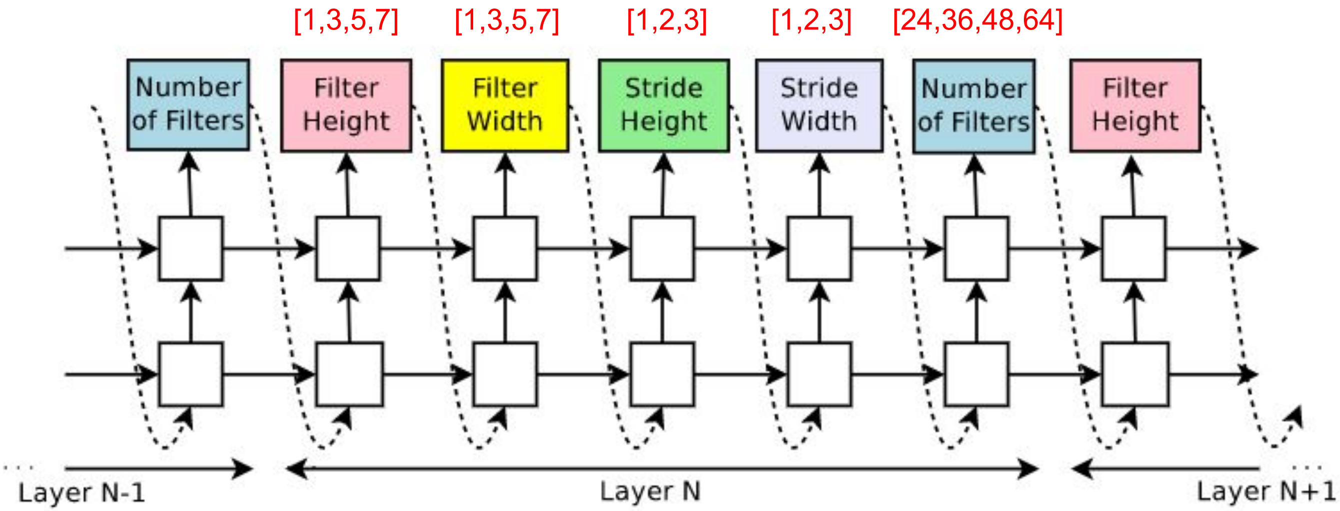
Overview of Experiments

- Apply this approach to Penn Treebank and CIFAR-10
- Evolve a convolutional neural network on CIFAR-10 and a recurrent neural network cell on Penn Treebank
- Achieve SOTA on the Penn Treebank dataset and almost SOTA on CIFAR-10 with a smaller and faster network
- Cell found on Penn Treebank beats LSTM baselines on other language modeling datasets and on machine translation

Neural Architecture Search for CIFAR-10

- We apply Neural Architecture Search to predicting convolutional networks on CIFAR-10
- Predict the following for a fixed number of layers (15, 20, 13):
  - Filter width/height
  - Stride width/height
  - Number of filters

Neural Architecture Search for CIFAR-10



Google

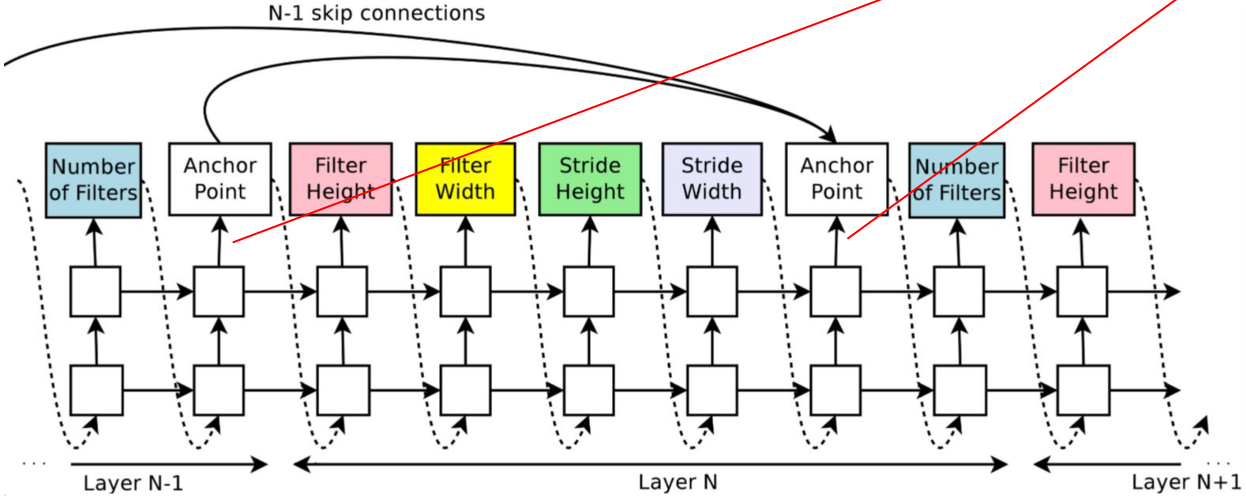
CIFAR-10 Prediction Method

- Expand search space to include branching and residual connections
- Propose the prediction of skip connections to expand the search space
- At layer N, we sample from N-1 sigmoids to determine what layers should be fed into layer N
- If no layers are sampled, then we feed in the minibatch of images
- At final layer take all layer outputs that have not been connected and concatenate them

Neural Architecture Search for CIFAR-10

Weight Matrices

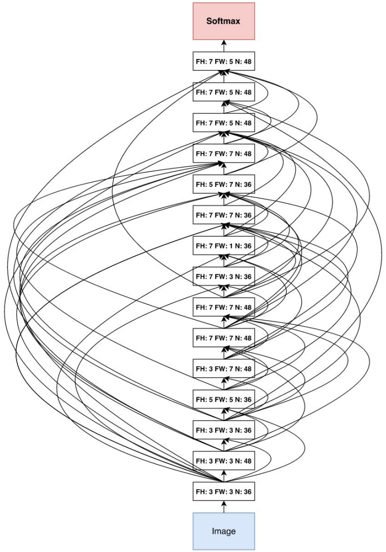
$$\text{P(Layer j is an input to layer i)} = \text{sigmoid}(v^{\mathrm{T}}\tanh(W_{prev} * h_j + W_{curr} * h_i))$$



CIFAR-10 Experiment Details

- Use 100 Controller Replicas each training 8 child networks concurrently
- Method uses 800 GPUs concurrently at one time
- Reward given to the Controller is the maximum validation accuracy of the last 5 epochs squared
- Split the 50,000 Training examples to use 45,000 for training and 5,000 for validation
- Each child model was trained for 50 epochs
- Run for a total of **12,800** child models
- Used curriculum training for the Controller by gradually increasing the number of layers sampled

Neural Architecture Search 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

5% faster

Best result of evolution (Real et al, 2017): 5.4%
Best result of Q-learning (Baker et al, 2017): 6.92%

Google

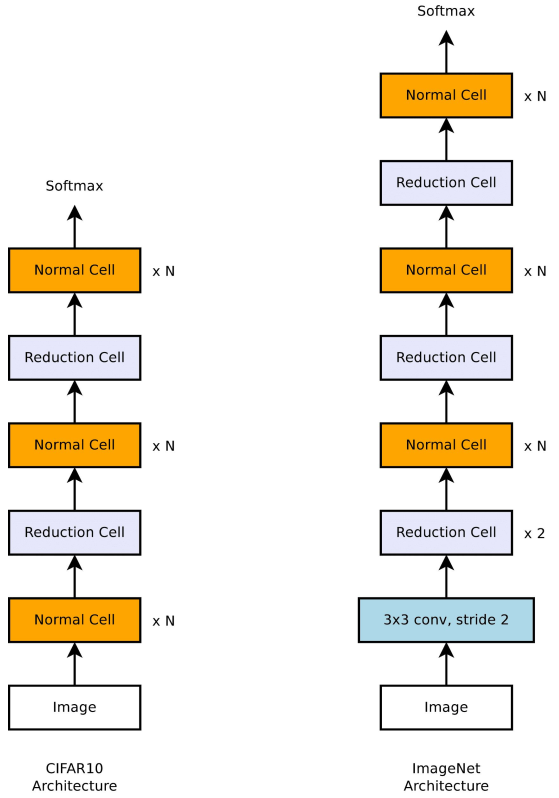
Neural Architecture Search for ImageNet

- Neural Architecture Search directly on ImageNet is expensive

- Key idea is to run Neural Architecture Search on CIFAR-10 to find a “cell”

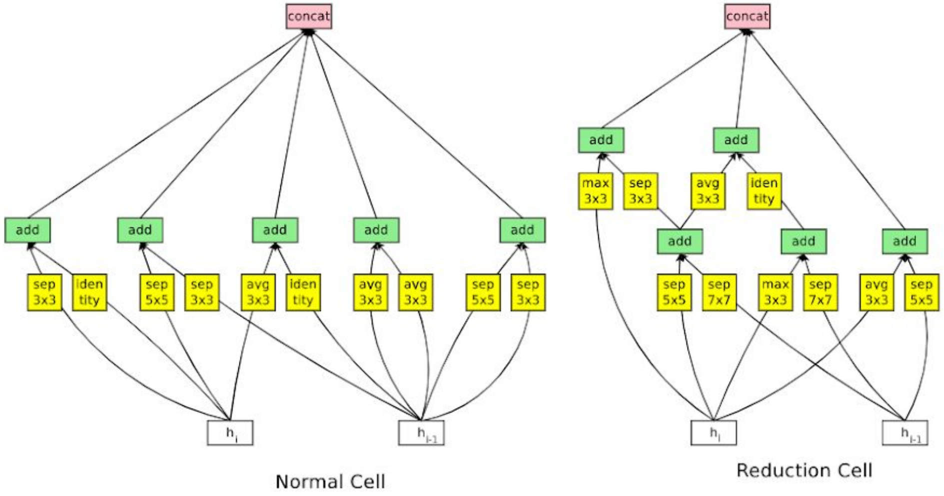
- Construct a bigger net from the “cell” and train the net on ImageNet

Neural Architecture Search for ImageNet



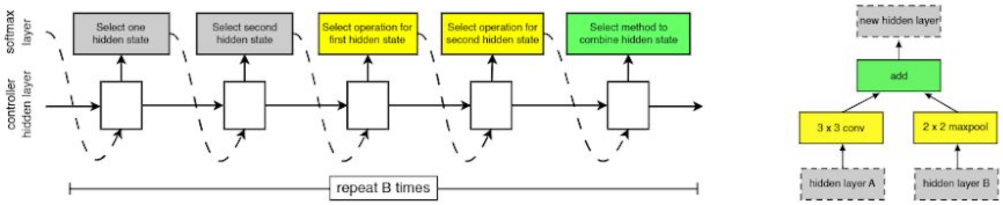
Google

Neural Architecture Search for ImageNet



Google

How the cell was found



Step 1. Select a hidden state from h or from the set of hidden states created in previous blocks.

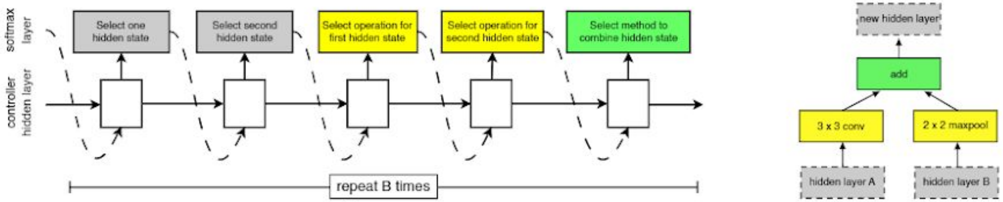
Step 2. Select a second hidden state from the same options as in Step 1.

Step 3. Select an operation to apply to the hidden state selected in Step 1.

Step 4. Select an operation to apply to the hidden state selected in Step 2.

Step 5. Select a method to combine the outputs of Step 3 and 4 to create a new hidden state.

How the cell was found



Step 1. Select a hidden state from h or from the set of hidden states created in previous blocks.

Step 2. Select a second hidden state from the same options as in Step 1.

Step 3. Select an operation to apply to the hidden state selected in Step 1.

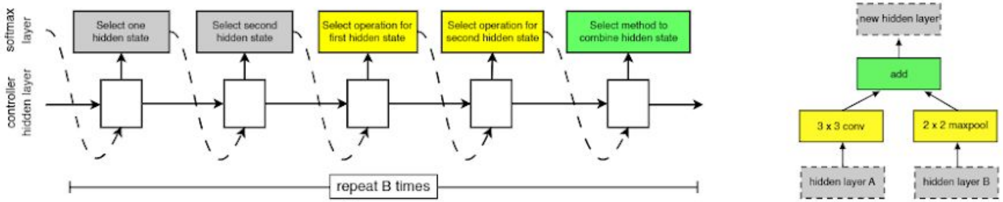
Step 4. Select an operation to apply to the hidden state selected in Step 2.

Step 5. Select a method to combine the outputs of Step 3 and 4 to create a new hidden state.

- identity
- 1x7 then 7x1 convolution
- 3x3 average pooling
- 5x5 max pooling
- 1x1 convolution
- 3x3 depthwise separable convolution
- 7x7 depthwise separable convolution
- 1x3 then 3x1 convolution
- 3x3 dilated convolution
- 3x3 max pooling
- 7x7 max pooling
- 3x3 convolution
- 5x5 depthwise separable convolution

Google

How the cell was found



Step 1. Select a hidden state from h or from the set of hidden states created in previous blocks.

Step 2. Select a second hidden state from the same options as in Step 1.

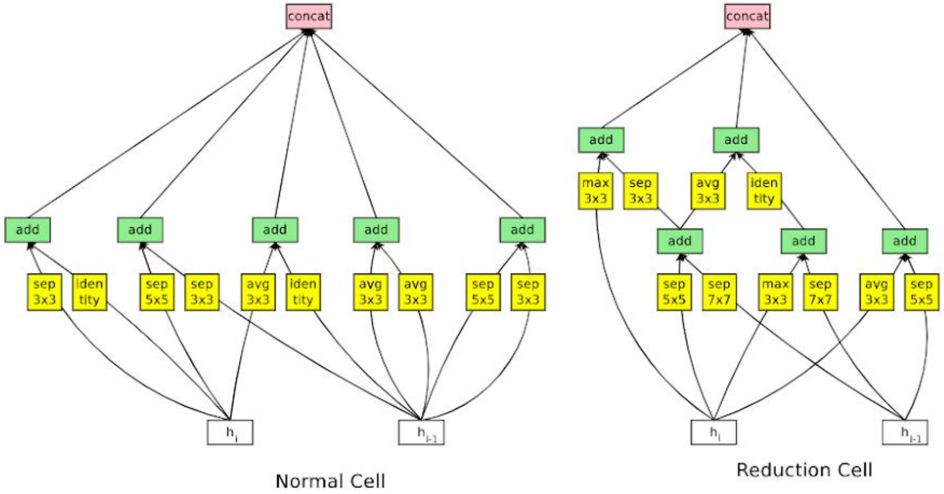
Step 3. Select an operation to apply to the hidden state selected in Step 1.

Step 4. Select an operation to apply to the hidden state selected in Step 2.

Step 5. Select a method to combine the outputs of Step 3 and 4 to create a new hidden state.

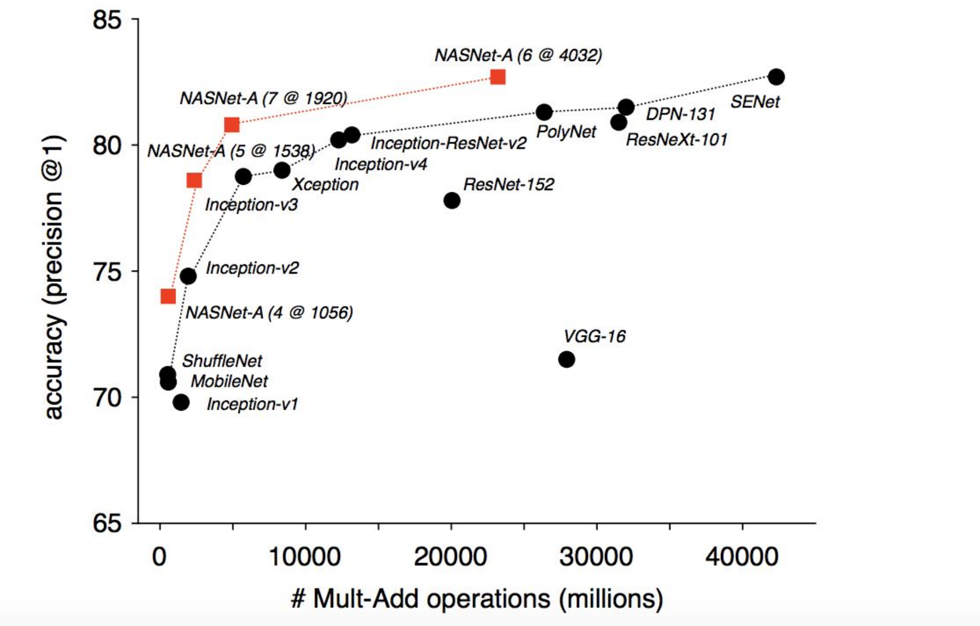
1. Elementwise addition
2. Concatenation along the filter dimension

The cell again



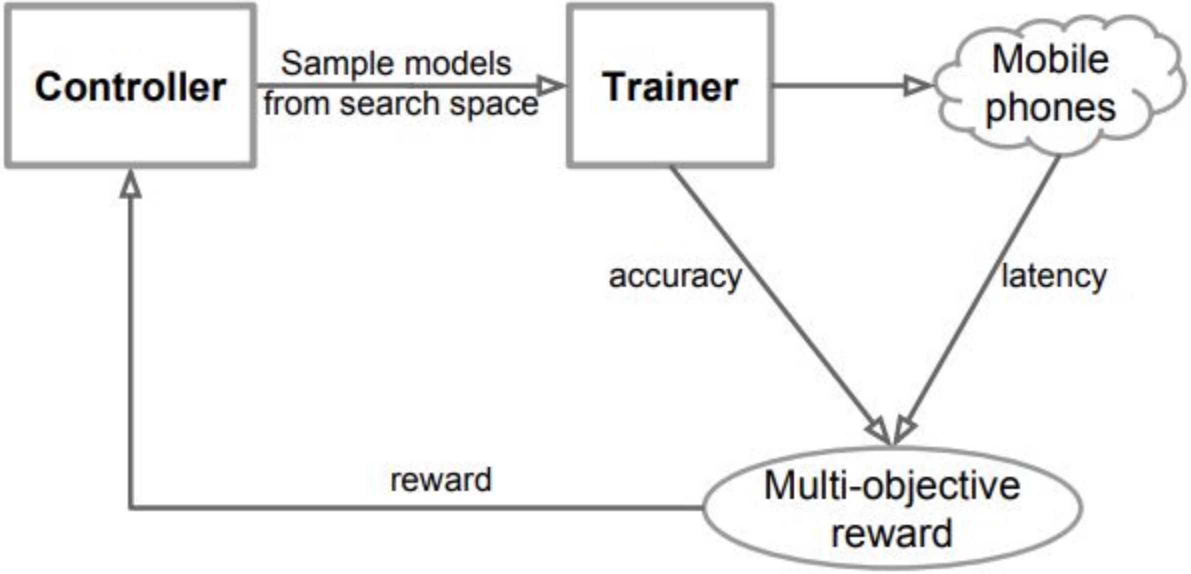
Google

Performance of cell on ImageNet

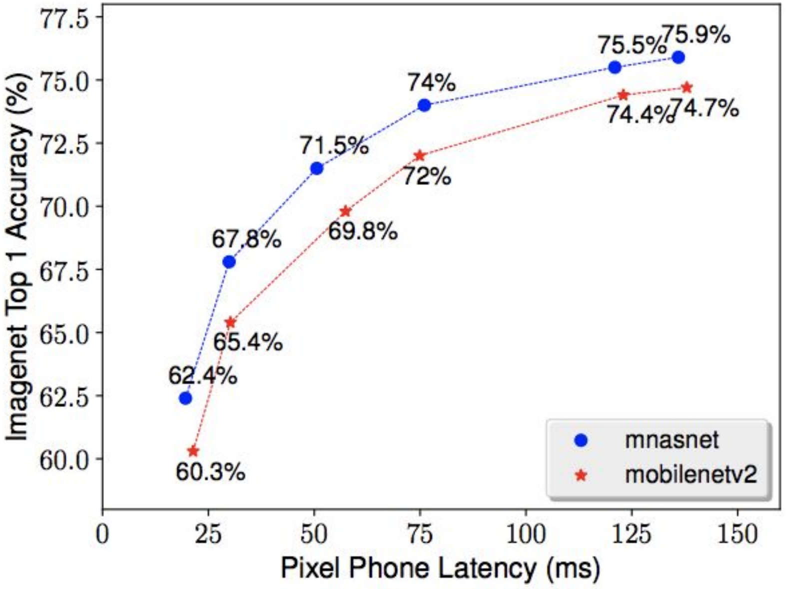


Google

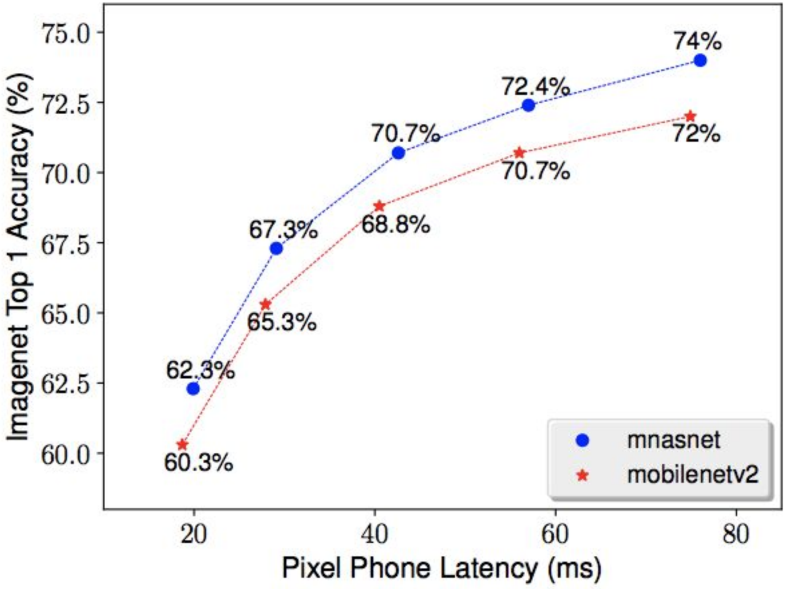
Platform aware Architecture Search



Platform aware Architecture Search

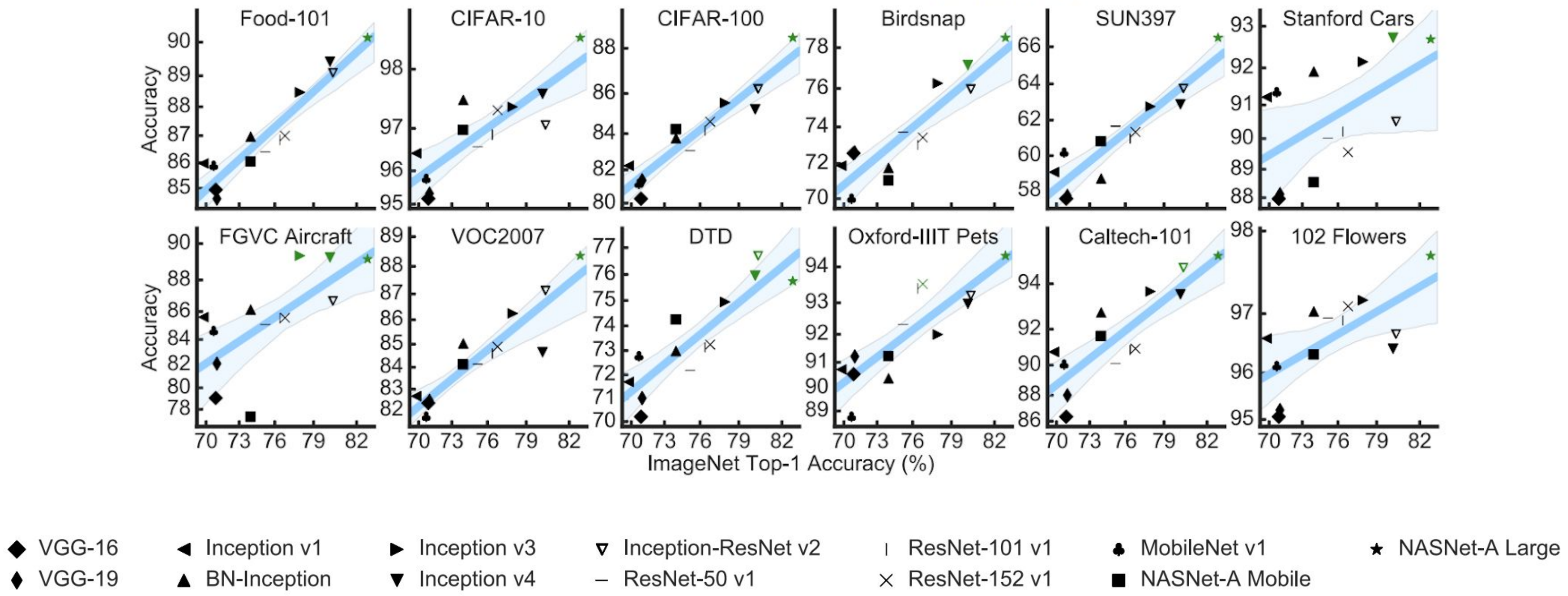


(a) Depth multiplier = 0.35, 0.5, 0.75, 1.0, 1.3, 1.4, corresponding to points from left to right.



(b) Input size = 96, 128, 160, 192, 224, corresponding to points from left to right.

Better ImageNet models transfer better

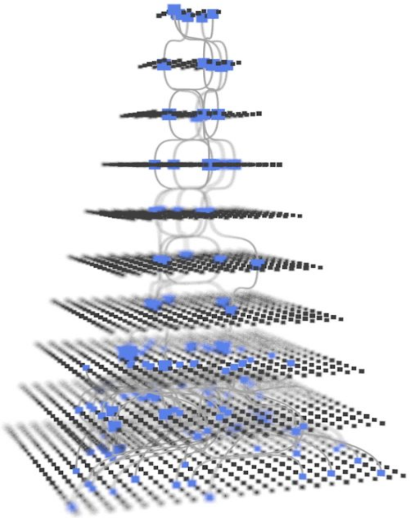


POC: skornblith@, shlens@, qvl@

Google

Controller: proposes Child Networks

Train & evaluate Child Networks



Iterate to find the most accurate Child Network

Reinforcement Learning or Evolution Search

Architecture / Optimization Algorithm / Nonlinearity

Google

Learn the Optimization Update Rule

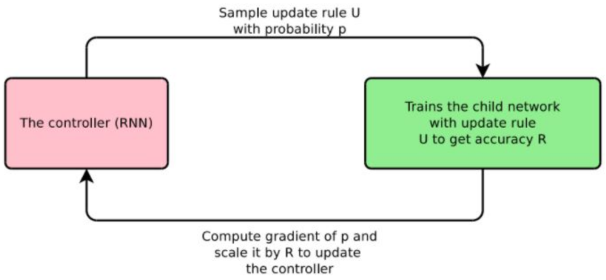


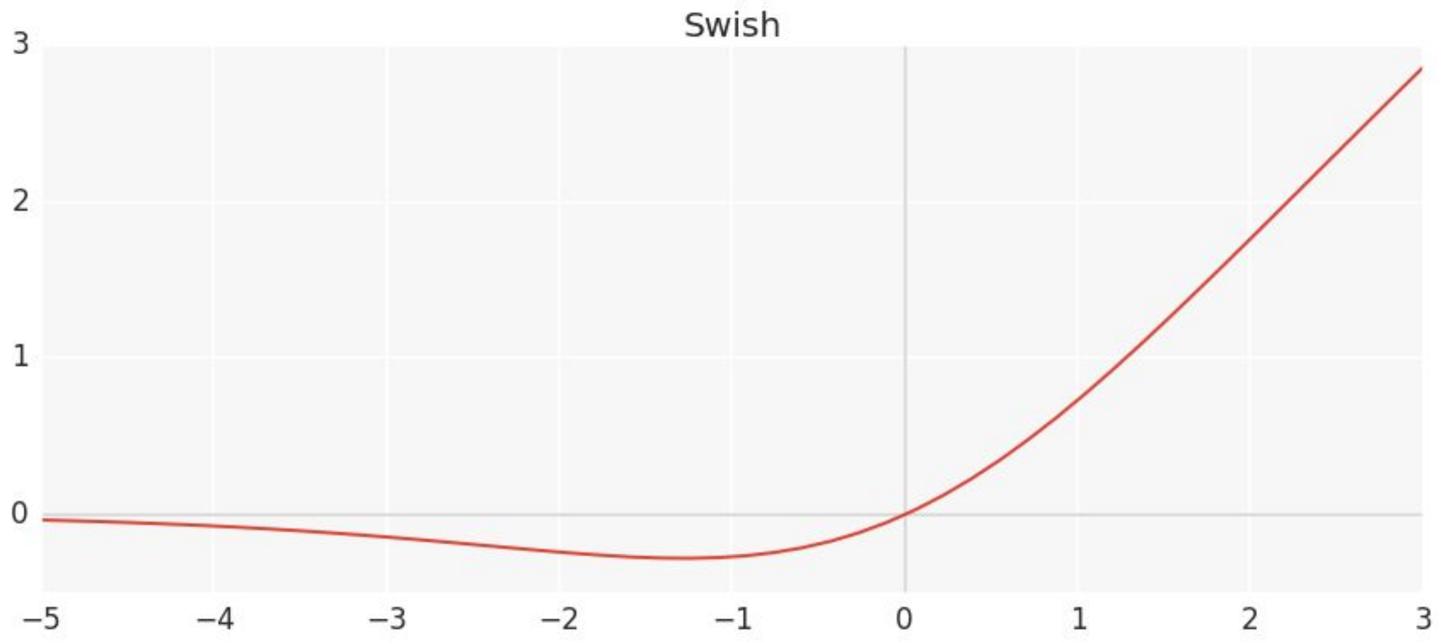
Figure 1. Overview of Neural Optimizer Search.

Optimizer	Final Val	Final Test	Best Val	Best Test
SGD	92.0	91.8	92.9	91.9
Momentum	92.7	92.1	93.1	92.3
ADAM	90.4	90.1	91.8	90.7
RMSProp	90.7	90.3	91.4	90.3
$[e^{\text{sign}(g)*\text{sign}(m)} + \text{clip}(g, 10^{-4})] * g$	92.5	92.4	93.8	93.1
$\text{clip}(\hat{m}, 10^{-4}) * e^{\hat{v}}$	93.5	92.5	93.8	92.7
$\hat{m} * e^{\hat{v}}$	93.1	92.4	93.8	92.6
$g * e^{\text{sign}(g)*\text{sign}(m)}$	93.1	92.8	93.8	92.8
$\text{drop}(g, 0.3) * e^{\text{sign}(g)*\text{sign}(m)}$	92.7	92.2	93.6	92.7
$\hat{m} * e^{g^2}$	93.1	92.5	93.6	92.4
$\text{drop}(\hat{m}, 0.1)/(e^{g^2} + \epsilon)$	92.6	92.4	93.5	93.0
$\text{drop}(g, 0.1) * e^{\text{sign}(g)*\text{sign}(m)}$	92.8	92.4	93.5	92.2
$\text{clip}(\text{RMSProp}, 10^{-5}) + \text{drop}(\hat{m}, 0.3)$	90.8	90.8	91.4	90.9
$\text{ADAM} * e^{\text{sign}(g)*\text{sign}(m)}$	92.6	92.0	93.4	92.0
$\text{ADAM} * e^{\hat{m}}$	92.9	92.8	93.3	92.7
$g + \text{drop}(\hat{m}, 0.3)$	93.4	92.9	93.7	92.9
$\text{drop}(\hat{m}, 0.1) * e^{g^3}$	92.8	92.7	93.7	92.8
$g - \text{clip}(g^2, 10^{-4})$	93.4	92.8	93.7	92.8
$e^g - e^{\hat{m}}$	93.2	92.5	93.5	93.1
$\text{drop}(\hat{m}, 0.3) * e^w$	93.2	93.0	93.5	93.2

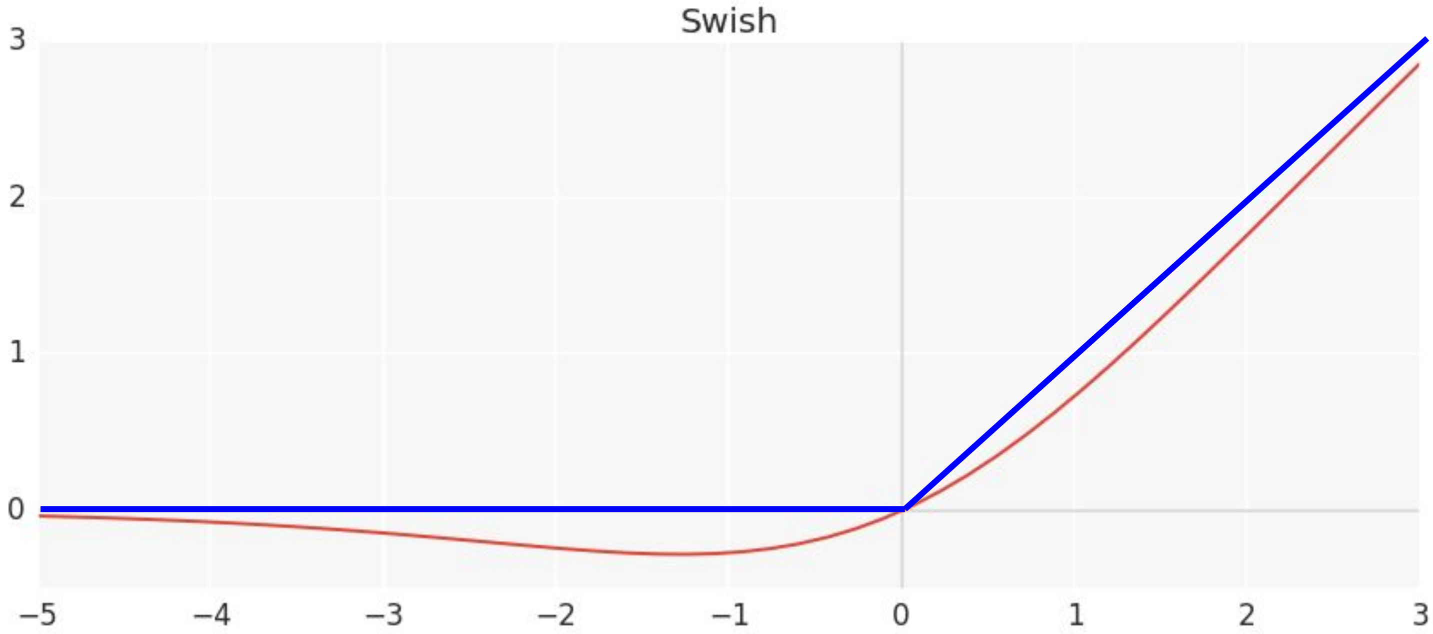
Table 1. Performance of Neural Optimizer Search and standard optimizers on the Wide-ResNet architecture (Zagoruyko & Komodakis, 2016) on CIFAR-10. Final Val and Final Test refer to the final validation and test accuracy after for training for 300 epochs. Best Val corresponds to the best validation accuracy over the 300 epochs and Best Test is the test accuracy at the epoch where the validation accuracy was the highest.

Neural Optimizer Search using Reinforcement Learning,
Irwan Bello, Barret Zoph, Vijay Vasudevan, and Quoc Le. ICML 2017

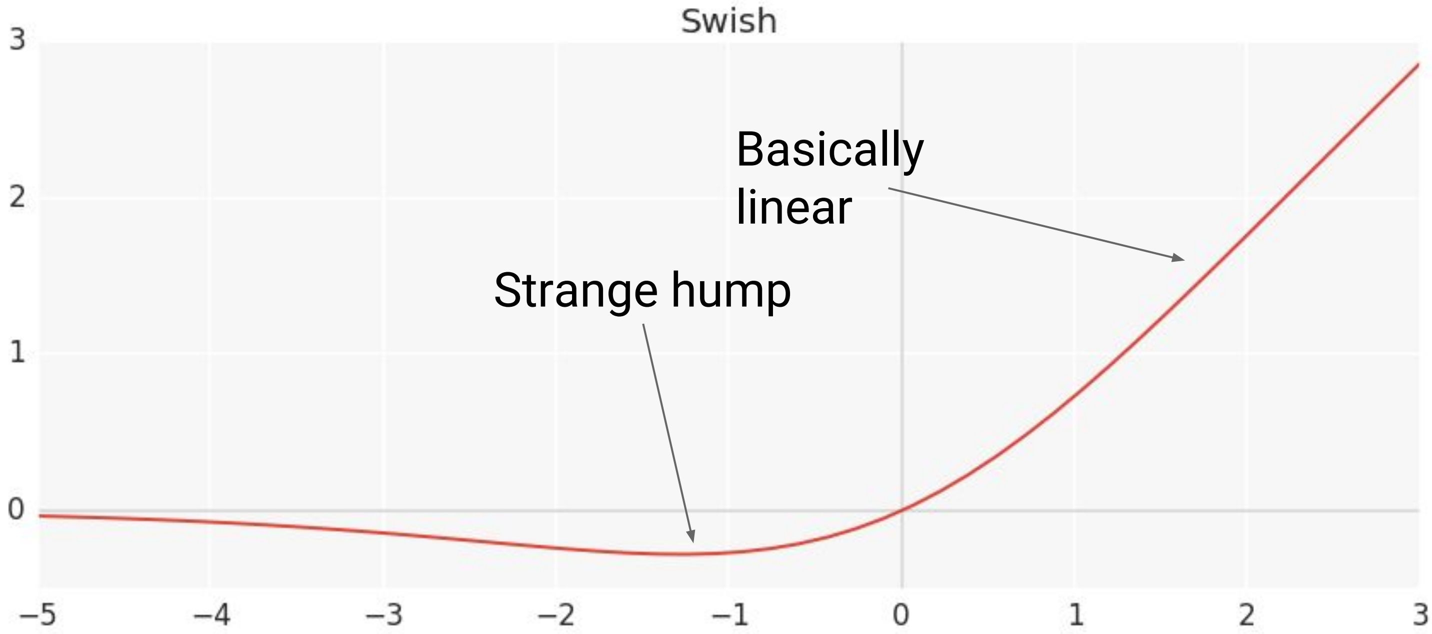
$$f(x) = x \cdot \mathrm{sigmoid}(x)$$



$$f(x) = x \cdot \mathrm{sigmoid}(x)$$



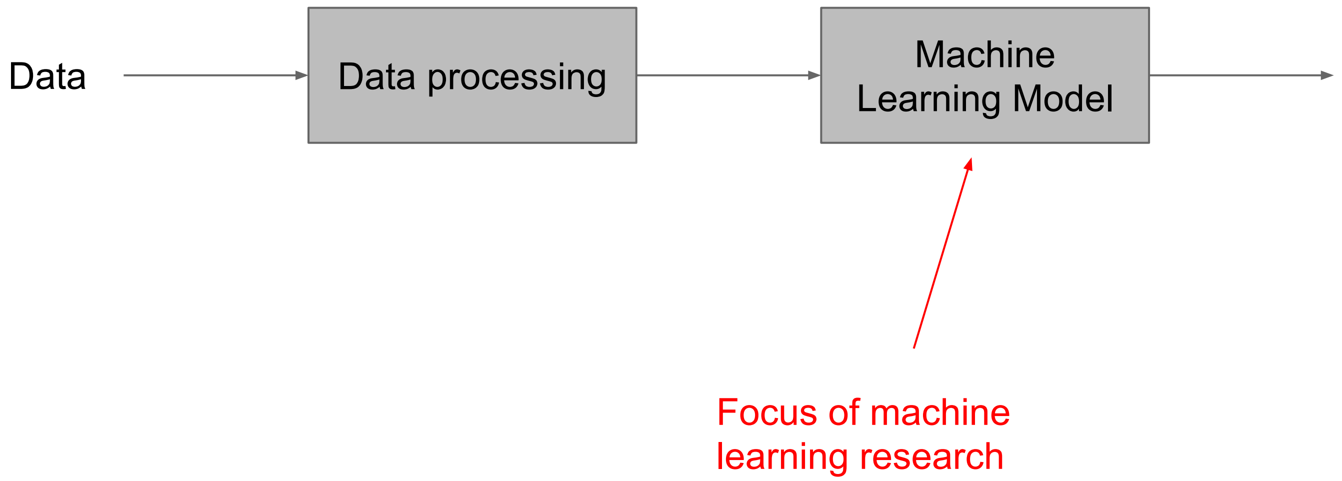
$$f(x) = x \cdot \mathrm{sigmoid}(x)$$



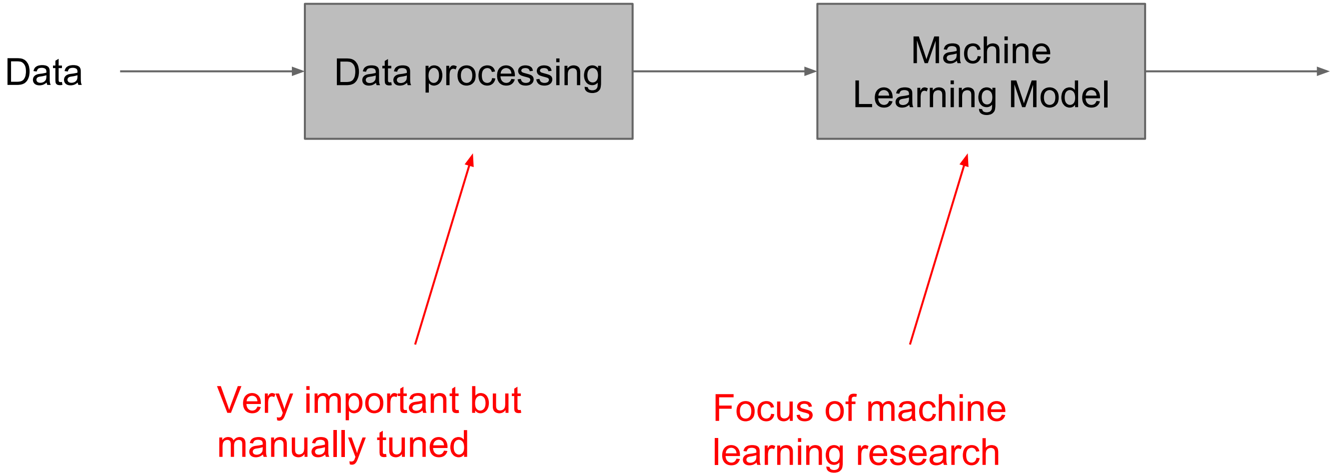
Mobile NASNet-A on ImageNet

Model	Top 1 Acc. (%)	Top 5 Acc. (%)
ReLU	73.5	91.4
LReLU	73.8	91.6
PReLU	**74.7**	92.3
ELU	74.2	91.8
SELU	73.7	91.7
Swish	**74.7**	92.0
Swish-β	**75.2**	**92.4**

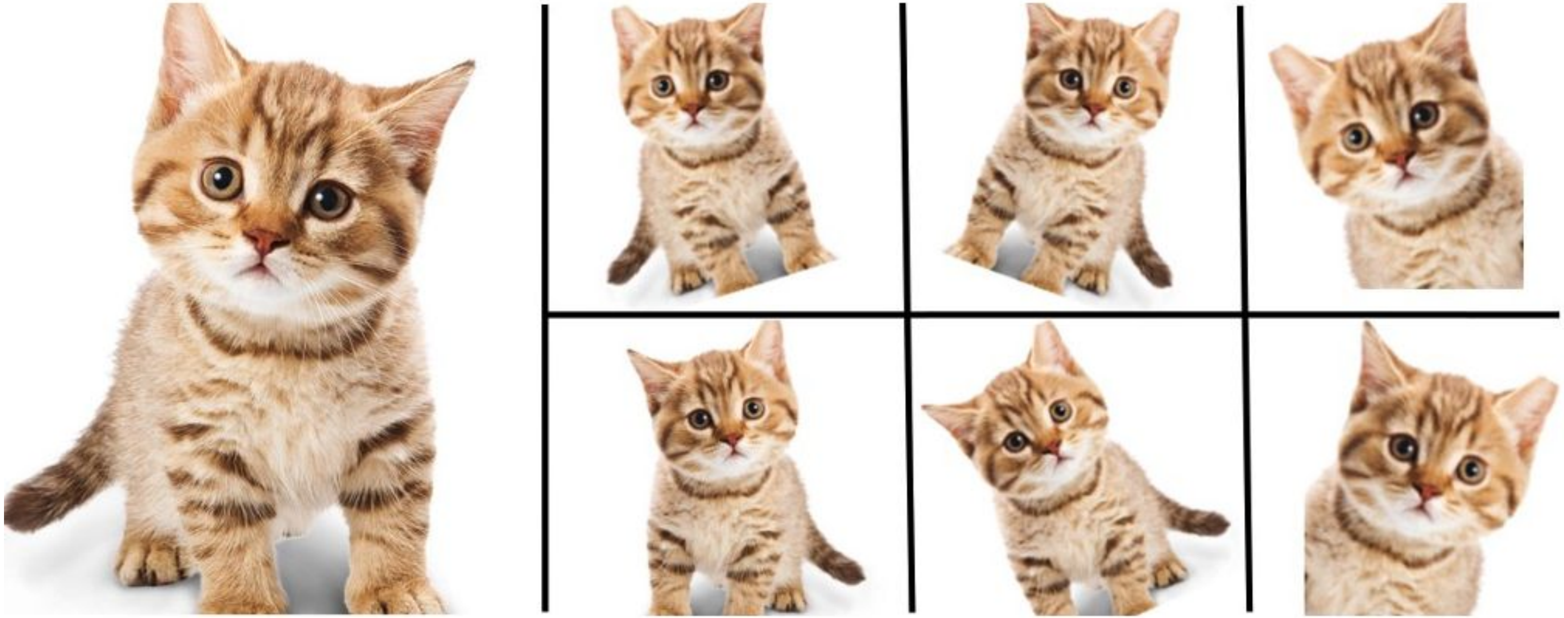
Mobile NASNet-A on ImageNet.



Data
Data processing
Machine Learning Model
Focus of machine learning research

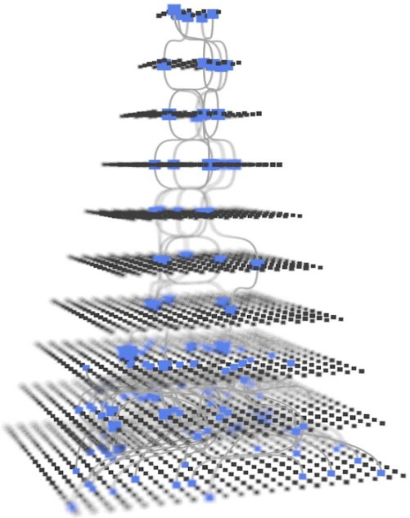

Data
Data processing
Machine
Learning Model
Very important but
manually tuned
Focus of machine
learning research

Data Augmentation



Enlarge your Dataset

Controller: proposes Child Networks



Reinforcement Learning
or Evolution Search

Iterate to find the most accurate Child Network

Train & evaluate Child Networks

Architecture / Optimization Algorithm /
Nonlinearity / Augmentation Strategy

AutoAugment: Example Policy

Original

Equalize, 0.4, 4
Rotate, 0.8, 8

Solarize, 0.6, 3
Equalize, 0.6, 7

Posterize, 0.8, 5
Equalize, 1.0, 2

Rotate, 0.2, 3
Solarize, 0.6, 8

Equalize, 0.6, 8
Posterize, 0.4, 6

Probability of applying

Magnitude

CIFAR-10

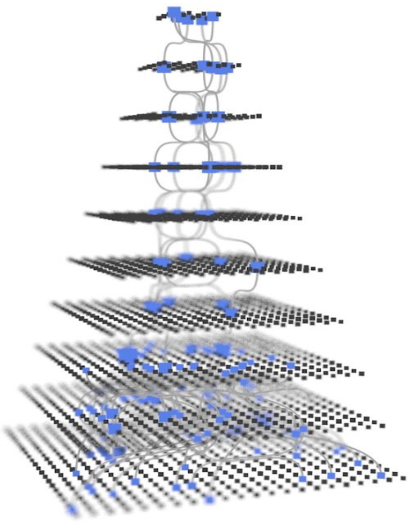
State-of-art: 2.1% error
AutoAugment: 1.5% error

ImageNet

State-of-art: 3.9% error
AutoAugment: 3.5% error

Summary of AutoML and its progress

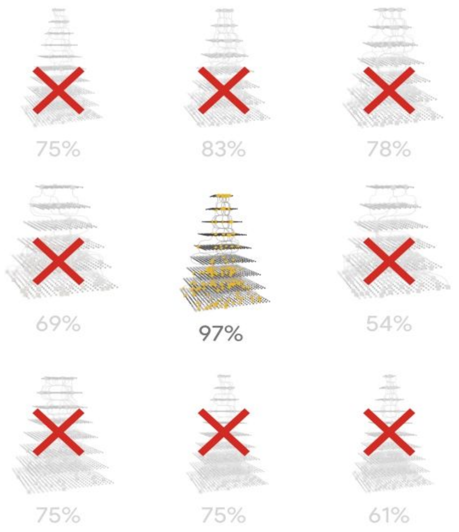
Controller: proposes Child Networks



Reinforcement Learning
or Evolution Search

Iterate to
find the
most
accurate
Child
Network

Train & evaluate Child Networks



Architecture / Optimization Algorithm /
Nonlinearity / Augmentation Strategy

References

- *Neural Architecture Search with Reinforcement Learning*. Barret Zoph and Quoc V. Le. ICLR, 2017
- *Learning Transferable Architectures for Large Scale Image Recognition*. Barret Zoph, Vijay Vasudevan, Jonathon Shlens, Quoc V. Le. CVPR, 2018
- *AutoAugment: Learning Augmentation Policies from Data*. Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, Quoc V. Le. Arxiv, 2018

- *Searching for Activation Functions*. Prajit Ramachandran, Barret Zoph, Quoc Le. ICLR Workshop, 2018

RL vs random search

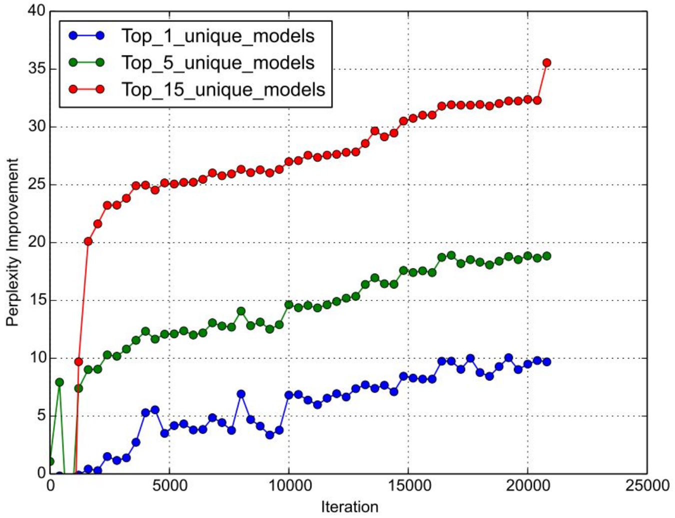


Figure 6: Improvement of Neural Architecture Search over random search over time. We plot the difference between the average of the top k models our controller finds vs. random search every 400 models run.