Scaling Machine Learning with TensorFlow

Jeff Dean
Google Brain team
g.co/brain

Presenting the work of many people at Google
Our Mission:
Make Machines Intelligent.
Improve People’s Lives.
Google Brain Team: Research Impact

● Since 2012, published > 130 papers at top venues in machine learning

● Some highlights:
  - 2012: DistBelief, unsupervised learning to discover cats
  - 2013: opensource of word2vec
  - 2014: sequence to sequence learning, image captioning
  - 2015: Inception, DeepDream, TensorFlow
  - 2016: neural translation, medical imaging, architecture search
Main Research Areas

- General Machine Learning Algorithms and Techniques
- Computer Systems for Machine Learning
- Natural Language Understanding
- Perception
- Healthcare
- Robotics
- Music and Art Generation
Main Research Areas

- General Machine Learning Algorithms and Techniques
- Computer Systems for Machine Learning
- Natural Language Understanding
- Perception
- Healthcare
- Robotics
- Music and Art Generation
The Google Brain team – Looking Back on 2016

Thursday, January 12, 2017

Posted by Jeff Dean, Google Senior Fellow, on behalf of the entire Google Brain team

The Google Brain team’s long-term goal is to create more intelligent software and systems that improve people’s lives, which we pursue through both pure and applied research in a variety of different domains. And while this is obviously a long-term goal, we would like to take a step back and look at some of the progress our team has made over the past year, and share what we feel may be store for 2017.

Research Publications

One important way in which we assess the quality of our research is through publications in top-tier international machine learning venues such as ICLR, NIPS, and ICML. Last year our team had a total of 27 accepted papers at these venues, covering a wide range of topics including 

- machine learning
- knowledge transfer from one network to another
- distributed training of machine learning models
- generative models for language, unsupervised learning for robotics
- automated theorem proving
- better theoretical understanding of neural networks
- algorithms for improved reinforcement learning
- and many others.

We also had numerous other papers accepted at conferences in fields such as natural language processing (ACL, CoNLL), speech (ICASSP), vision (CVPR), robotics (IGIT), and computer systems (OSDI).

Spreading Machine Learning within Google

In addition to the public-facing activities outlined above, we have continued to work within Google to spread machine learning expertise throughout our product teams, and to ensure that the company as a whole is well positioned to take advantage of any new machine learning research that emerges. As one example, we worked closely with our platform teams to provide training for data scientists and developers for Google Tensor Processing Units (TPUs), a highly efficient hardware-accelerated machine learning accelerator that was discussed at Google I/O.

All in all, 2016 was an exciting year for the Google Brain team and our many collaborators and colleagues both within and outside of Google, and we look forward to our machine learning research having significant impact in 2017!
Accuracy

Scale (data size, model size)

1980s and 1990s

neural networks

other approaches
Accuracy

Scale (data size, model size)

1980s and 1990s

more compute

neural networks

other approaches
Accuracy

Scale (data size, model size)

Now

more compute

neural networks

other approaches
Growing Use of Deep Learning at Google

Across many products/areas:
- Android
- Apps
- drug discovery
- Gmail
- Image understanding
- Maps
- Natural language understanding
- Photos
- Robotics research
- Speech
- Translation
- YouTube
- ... many others ...
Experiment Turnaround Time and Research Productivity

- **Minutes, Hours:**
  - Interactive research! Instant gratification!

- **1-4 days**
  - Tolerable
  - Interactivity replaced by running many experiments in parallel

- **1-4 weeks**
  - High value experiments only
  - Progress stalls

- **>1 month**
  - Don’t even try
Build the right tools
Open, standard software for general machine learning

Great for Deep Learning in particular

First released Nov 2015

Apache 2.0 license

http://tensorflow.org/

and

https://github.com/tensorflow/tensorflow
Computation is a dataflow graph

Graph of Nodes, also called *Operations* or *ops.*
Computation is a dataflow graph

Edges are N-dimensional arrays: Tensors

- biases
- weights
- examples
- labels
Example TensorFlow fragment

- Build a graph computing a neural net inference.

```python
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets('MNIST_data', one_hot=True)
x = tf.placeholder("float", shape=[None, 784])
W = tf.Variable(tf.zeros([784,10]))
b = tf.Variable(tf.zeros([10]))
y = tf.nn.softmax(tf.matmul(x, W) + b)
```
Computation is a dataflow graph with state

'Biases' is a variable

Some ops compute gradients

−= updates biases

biases

... Add ...

learning rate

Mul −=
Computation is a dataflow graph
Assign Devices to Ops

- TensorFlow inserts Send/Recv Ops to transport tensors across devices
- *Recv* ops pull data from *Send* ops
Assign *Devices to Ops*

- TensorFlow inserts *Send/Recv* Ops to transport tensors across devices
- *Recv* ops pull data from *Send* ops
Same mechanism supports large distributed systems

Computation spread across hundreds of machines and thousands of GPU cards
TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems
(Preliminary White Paper, November 9, 2015)

Google Research*


TensorFlow: A system for large-scale machine learning

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng

Google Brain

Paper in OSDI 2016
An open-source software library for Machine Intelligence

TensorFlow 1.0 has arrived!

We're excited to announce the release of TensorFlow 1.0! Check out the migration guide to upgrade your code with ease.

UPGRADE NOW

Dynamic graphs in TensorFlow

We've open-sourced TensorFlow Fold to make it easier than ever to work with input data with varying shapes and sizes.

LEARN MORE

The 2017 TensorFlow Dev Summit

Thousands of people from the TensorFlow community participated in the first flagship event. Watch the keynote and talks.

WATCH VIDEOS

http://tensorflow.org/
Why Did We Build TensorFlow?

Wanted system that was flexible, scalable, and production-ready

DistBelief, our first system, was good on two of these, but lacked flexibility

Most existing open-source packages were also good on 2 of 3 but not all 3
TensorFlow Goals

Establish **common platform** for expressing machine learning ideas and systems.

Make this platform the **best in the world** for both research and production use.

Open source it so that it becomes a **platform for everyone**, not just Google.
ML is done in many places

TensorFlow GitHub stars by GitHub user profiles w/ public locations
Source: http://jrvis.com/red-dwarf/?user=tensorflow&repo=tensorflow
Progress

v0.5
Initial Release

v0.7
TensorFlow Serving

v0.9
iOS; Mac GPU

v0.11
HDFS; CUDA 8, CuDNN 5

Nov '15 Dec '15 Feb '16 Apr '16 Jun '16 Aug '16 Oct '16 Nov '16

v0.6
Faster on GPUs; Python 3.3+

v0.8
Distributed TensorFlow

v0.10
Slim

v0.12
Windows 7, 10, and Server 2016; TensorBoard Embedding Visualizer

https://github.com/tensorflow/tensorflow/releases
Progress

v0.5
Initial Release

v0.7
TensorFlow Serving

v0.9
iOS; Mac GPU

v0.11
HDFS; CUDA 8, CuDNN 5

v1.0
released in Feb. 2017


v0.6
Faster on GPUs; Python 3.3+

v0.8
Distributed TensorFlow

v0.10
Slim

v0.12
Windows 7, 10, and Server 2016; TensorBoard Embedding Visualizer

https://github.com/tensorflow/tensorflow/releases
TensorFlow: A Vibrant Open-Source Community

- **Rapid development, many outside contributors**
  - 475+ non-Google contributors to TensorFlow 1.0
  - 15,000+ commits in 15 months
  - Many community created tutorials, models, translations, and projects
    - ~7,000 GitHub repositories with ‘TensorFlow’ in the title

- **Direct engagement between community and TensorFlow team**
  - 5000+ Stack Overflow questions answered
  - 80+ community-submitted GitHub issues responded to weekly

- **Growing use in ML classes: Toronto, Berkeley, Stanford, ...**
Tutorials

Basic Neural Networks

The first few TensorFlow tutorials guide you through training and testing a simple neural network to classify handwritten digits from the MNIST database of digit images.

MNIST For ML Beginners

If you're new to machine learning, we recommend starting here. You'll learn about a classic problem, handwritten digit classification (MNIST), and get a gentle introduction to multiclass classification.

View Tutorial

Deep MNIST for Experts

If you're already familiar with other deep learning software packages, and are already familiar with MNIST, this tutorial will give you a very brief primer on TensorFlow.

View Tutorial

TensorFlow Mechanics 101

This is a technical tutorial, where we walk you through the details of using TensorFlow infrastructure to train models at scale. We use MNIST as the example.

View Tutorial

Easy ML with tf.contrib.learn

tf.contrib.learn Quickstart

A quick introduction to tf.contrib.learn, a high-level API for TensorFlow. Build, train, and evaluate a neural network with just a few lines of code.

View Tutorial

tensorflow.org/tutorials
Performance matters

Research
- Iterate quickly
- Train models faster
- Run more experiments in parallel

Production
- Server farms and embedded
- CPUs, GPUs, TPUs, and more
- Low-latency serving
TensorFlow v1.0 Performance
Inception-v3 Training - Ideal Scaling Synthetic Data

DGX-1:

K80:

Inception-v3 training on P100 GPUs

Inception-v3 training on K80 GPUs
TensorFlow v1.0 Performance

Inception-v3 Training - Synthetic Data

DGX-1: 7.37x speedup at 8 GPUs

K80: 7.5x speedup at 8 GPUs
TensorFlow v1.0 Performance
Inception-v3 Training - Real Data

DGX-1: 7.2x speedup at 8 GPUs

K80: 7.3x speedup at 8 GPUs
TensorFlow v1.0 Performance
Inception-v3 Distributed Training - Synthetic Data

58x speedup at 64 GPUs (8 Servers / 8 GPUs each)

- GPU: K80
- Network: 20 Gb/sec
Just-In-Time Compilation
via XLA, "Accelerated Linear Algebra" compiler

TF graphs go in,

```
0x00000000      movq    (%rdx), %rax
0x00000003      vmovaps (%rax), %xmm0
0x00000007      vmulps  %xmm0, %xmm0, %xmm0
0x0000000b      vmovaps %xmm0, (%rdi)
```

Optimized & specialized assembly comes out.

Let's explain that!
Demo: Inspect JIT code in TensorFlow iPython shell

```
In [1]: %cpaste
Pasting code: enter '--' alone on the line to stop or use Ctrl-D.
with tf.Session() as sess:
    x = tf.placeholder(tf.float32, [4])
    with tf.device("device:XLA_CPU:0"):
        y = x * x
    result = sess.run(y, {x: [1.5, 0.5, -0.5, -1.5]})
```
Computers can now see

Large implications for healthcare
Using similar model for detecting diabetic retinopathy in retinal images
Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs

Varun Gulshan, PhD; Lily Peng, MD, PhD; Marc Coram, PhD; et al

Author Affiliations

Performance on par or slightly better than the median of 8 U.S. board-certified ophthalmologists (F-score of 0.95 vs. 0.91).

Detecting Cancer Metastases on Gigapixel Pathology Images

Yun Liu¹*, Krishna Gadepalli¹, Mohammad Norouzi¹, George E. Dahl¹, Timo Kohlberger¹, Aleksey Boyko¹, Subhashini Venugopalan²**, Aleksei Timofeev², Philip Q. Nelson², Greg S. Corrado¹, Jason D. Hipp³, Lily Peng¹, and Martin C. Stumpe¹

{liuyun,mnorouzi,gdahl,lhpeng,mstumpe}@google.com

¹Google Brain, ²Google Inc, ³Verily Life Sciences, Mountain View, CA, USA

ML Challenges in Pathology

- Extremely large images (> 100k x 100k pixels)
- Multiscale problem - need detail as well as context
Multiscale model

- Detail $\leftrightarrow$ Context
- Resembles microscope magnifications
Detecting breast cancer metastases in lymph nodes

- Biopsy image
- Ground truth (from pathologist)
- Model prediction (early results)
- Model prediction (current results)

- Non-tumor regions
- Tumor
- Tumor not annotated in ground truth
- Reduced noise in normal regions (everywhere else)
- Tumor (in ground truth)
Model performance compared to pathologist

<table>
<thead>
<tr>
<th></th>
<th>our model</th>
<th>pathologist*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tumor localization score (FROC)</td>
<td>0.89</td>
<td>0.73</td>
</tr>
<tr>
<td>Sensitivity at 8 FP</td>
<td>0.92</td>
<td>0.73</td>
</tr>
<tr>
<td>Slide classification (AUC)</td>
<td>0.97</td>
<td>0.96</td>
</tr>
</tbody>
</table>

* pathologist given infinite time per image (reaching 0 FPs)

Evaluated using Camelyon16 dataset (just 270 training examples!)
Scaling language understanding models
Sequence-to-Sequence Model

[Sutskever & Vinyals & Le NIPS 2014]

\[ P(y_1, \ldots, y_{T'} | x_1, \ldots, x_T) = \prod_{t=1}^{T'} p(y_t | v, y_1, \ldots, y_{t-1}) \]
Sequence-to-Sequence Model: Machine Translation

[Sutskever & Vinyals & Le NIPS 2014]

Input sentence

Quelle est votre taille? <EOS>

Target sentence

How
Sequence-to-Sequence Model: Machine Translation

[Sutskever & Vinyals & Le NIPS 2014]

Input sentence: Quelle est votre taille? <EOS>

Target sentence: How tall

Diagram showing the flow of translation from input sentence to target sentence.
Sequence-to-Sequence Model: Machine Translation

[Sutskever & Vinyals & Le NIPS 2014]

Input sentence

Quelle est votre taille? <EOS>

How tall are

Target sentence
Sequence-to-Sequence Model: Machine Translation

How tall are you?

Quelle est votre taille? <EOS>

Input sentence

Target sentence
Sequence-to-Sequence Model: Machine Translation

[Sequence-to-Sequence Model: Machine Translation]

[Sutskever & Vinyals & Le NIPS 2014]

At inference time:
 Beam search to choose most probable over possible output sequences

Input sentence

Quelle est votre taille? <EOS>
Sequence to Sequence model applied to Google Translate
Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi
yonghui,schuster,zhifengc,qvl,mnorouzi@google.com

Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey,
Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser,
Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens,
George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa,
Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, Jeffrey Dean

https://arxiv.org/abs/1609.08144
Google Neural Machine Translation Model

One model replica: one machine w/ 8 GPUs
Model + Data Parallelism

Parameters distributed across many parameter server machines
Neural Machine Translation

Closes gap between old system and human-quality translation by 58% to 87%

Enables better communication across the world

research.googleblog.com/2016/09/a-neural-network-for-machine.html
Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation,
Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat,
Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean
https://arxiv.org/abs/1611.04558

Bigger models, but sparsely activated
Bigger models, but sparsely activated

Motivation:
Want huge model capacity for large datasets, but want individual example to only activate tiny fraction of large model
Per-Example Routing
Table 7: Perplexity and BLEU comparison of our method against previous state-of-art methods on the Google Production En→Fr dataset.

<table>
<thead>
<tr>
<th>Model</th>
<th>Eval Perplexity</th>
<th>Eval BLEU</th>
<th>Test Perplexity</th>
<th>Test BLEU</th>
<th>Computation per Word</th>
<th>Total #Parameters</th>
<th>Training Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>MoE with 2048 Experts</td>
<td>2.60</td>
<td>37.27</td>
<td>2.69</td>
<td>36.57</td>
<td>100.8M</td>
<td>8.690B</td>
<td>1 day/64 k40s</td>
</tr>
<tr>
<td>GNMT (Wu et al., 2016)</td>
<td>2.78</td>
<td>35.80</td>
<td>2.87</td>
<td>35.56</td>
<td>214.2M</td>
<td>246.9M</td>
<td>6 days/96 k80s</td>
</tr>
</tbody>
</table>

Automated machine learning ("learning to learn")
Current:

Solution = ML expertise + data + computation
Current:

\[ \text{Solution} = \text{ML expertise} + \text{data} + \text{computation} \]

Can we turn this into:

\[ \text{Solution} = \text{data} + 100X \text{ computation} \]

???
Early encouraging signs

Trying multiple different approaches:

(1) RL-based architecture search
(2) Model architecture evolution
Idea: model-generating model trained via RL

(1) Generate ten models
(2) Train them for a few hours
(3) Use loss of the generated models as reinforcement learning signal

To appear in ICLR 2017
CIFAR-10 Image Recognition Task

<table>
<thead>
<tr>
<th>Model</th>
<th>Depth</th>
<th>Parameters</th>
<th>Error rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network in Network (Lin et al., 2013)</td>
<td>-</td>
<td>-</td>
<td>8.81</td>
</tr>
<tr>
<td>All-CNN (Springenberg et al., 2014)</td>
<td>-</td>
<td>-</td>
<td>7.25</td>
</tr>
<tr>
<td>Deeply Supervised Net (Lee et al., 2015)</td>
<td>-</td>
<td>-</td>
<td>7.97</td>
</tr>
<tr>
<td>Highway Network (Srivastava et al., 2015)</td>
<td>-</td>
<td>-</td>
<td>7.72</td>
</tr>
<tr>
<td>Scalable Bayesian Optimization (Snoek et al., 2015)</td>
<td>-</td>
<td>-</td>
<td>6.37</td>
</tr>
<tr>
<td>FractalNet (Larsson et al., 2016) with Dropout/Drop-path</td>
<td>21</td>
<td>38.6M</td>
<td>5.22</td>
</tr>
<tr>
<td>ResNet (He et al., 2016a)</td>
<td>110</td>
<td>1.7M</td>
<td>6.61</td>
</tr>
<tr>
<td>ResNet (reported by Huang et al. (2016b))</td>
<td>110</td>
<td>1.7M</td>
<td>6.41</td>
</tr>
<tr>
<td>ResNet with Stochastic Depth (Huang et al., 2016b)</td>
<td>1202</td>
<td>10.2M</td>
<td>4.91</td>
</tr>
<tr>
<td>Wide ResNet (Zagoruyko &amp; Komodakis, 2016)</td>
<td>16</td>
<td>11.0M</td>
<td>4.81</td>
</tr>
<tr>
<td></td>
<td>28</td>
<td>36.5M</td>
<td>4.17</td>
</tr>
<tr>
<td>ResNet (pre-activation) (He et al., 2016b)</td>
<td>164</td>
<td>1.7M</td>
<td>5.46</td>
</tr>
<tr>
<td></td>
<td>1001</td>
<td>10.2M</td>
<td>4.62</td>
</tr>
<tr>
<td>DenseNet ($L = 40, k = 12$) Huang et al. (2016a)</td>
<td>40</td>
<td>1.0M</td>
<td>5.24</td>
</tr>
<tr>
<td>DenseNet($L = 100, k = 12$) Huang et al. (2016a)</td>
<td>100</td>
<td>7.0M</td>
<td>4.10</td>
</tr>
<tr>
<td>DenseNet ($L = 100, k = 24$) Huang et al. (2016a)</td>
<td>100</td>
<td>27.2M</td>
<td>3.74</td>
</tr>
<tr>
<td>Neural Architecture Search v1 no stride or pooling</td>
<td>15</td>
<td>4.2M</td>
<td>5.50</td>
</tr>
<tr>
<td>Neural Architecture Search v2 predicting strides</td>
<td>20</td>
<td>2.5M</td>
<td>6.01</td>
</tr>
<tr>
<td>Neural Architecture Search v3 max pooling</td>
<td>39</td>
<td>7.1M</td>
<td>4.47</td>
</tr>
<tr>
<td>Neural Architecture Search v3 max pooling + more filters</td>
<td>39</td>
<td>32.0M</td>
<td>3.84</td>
</tr>
</tbody>
</table>

Table 1: Performance of Neural Architecture Search and other state-of-the-art models on CIFAR-10.
Penn Tree Bank Language Modeling Task

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

Table 2: Single model perplexity on the test set of the Penn Treebank language modeling task. Parameter numbers with $\dagger$ are estimates with reference to Merity et al. (2016).
Idea: evolve models via evolutionary algorithm
Large-Scale Evolution of Image Classifiers

Esteban Real¹ Sherry Moore¹ Andrew Selle¹ Saurabh Saxena¹
Yutaka Leon Suematsu² Quoc Le¹ Alex Kurakin¹

Evolve this DNA

ReLU
ReLU
ReLU
Linear
ReLU

Softmax
Softmax

0 1 2 3 4 5 6 7 8 9

Trained Model

W
W
W
W
W
W
W
B
B

W
W
W
W
W
W
W
B
B

B

DNA → Model → Trained Model → Fitness

https://arxiv.org/abs/1703.01041
Evolutionary Step

Mutations:
- Alter learning rate
- Identity
- Reset weights
- Insert convolution
- Remove convolution
- Alter strides
- Alter # of channels
- Alter horiz. filter size
- Alter vert. filters size
- Insert nonlinearity
- Remove nonlinearity
- Add-skip
- Remove skip
Evolutionary Step

- Pick 2 at random
- Copy-mutate best
- Kill worst

Worker 1
Worker 2
Worker 3
Worker 4
Worker N

Preempted by
vincentrobot.play_fetch
Evolve From Scratch

- Initialize with linear models
- Repeat evolutionary step
<table>
<thead>
<tr>
<th>Study</th>
<th>Params.</th>
<th>C10+</th>
<th>C100+</th>
<th>Within?</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAXOUT (GOODFELLOW ET AL., 2013)</td>
<td>–</td>
<td>90.7%</td>
<td>61.4%</td>
<td>No</td>
</tr>
<tr>
<td>NETWORK IN NETWORK (LIN ET AL., 2013)</td>
<td>–</td>
<td>91.2%</td>
<td>–</td>
<td>No</td>
</tr>
<tr>
<td>ALL-CNN (SPRINGENBERG ET AL., 2014)</td>
<td>1.3 M</td>
<td>92.8%</td>
<td>66.3%</td>
<td>Yes</td>
</tr>
<tr>
<td>DEEPLY SUPERVISED (LEE ET AL., 2015)</td>
<td>–</td>
<td>92.0%</td>
<td>65.4%</td>
<td>No</td>
</tr>
<tr>
<td>HIGHWAY (SRIVASTAVA ET AL., 2015)</td>
<td>2.3 M</td>
<td>92.3%</td>
<td>67.6%</td>
<td>No</td>
</tr>
<tr>
<td>RESNet (HE ET AL., 2016)</td>
<td>1.7 M</td>
<td>93.4%</td>
<td>72.8%</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Evolution (ours)</strong></td>
<td>5.4 M</td>
<td>94.6%</td>
<td>76.0%</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>Wide ResNet 28-10</strong> (ZAGORUYKO &amp; KOMODAKIS, 2016)</td>
<td>36.5 M</td>
<td>96.0%</td>
<td>80.0%</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Wide ResNet 40-10+d/o</strong> (ZAGORUYKO &amp; KOMODAKIS, 2016)</td>
<td>50.7 M</td>
<td>96.2%</td>
<td>81.7%</td>
<td>No</td>
</tr>
<tr>
<td><strong>DenseNet</strong> (HUANG ET AL., 2016A)</td>
<td>25.6 M</td>
<td>96.7%</td>
<td>82.8%</td>
<td>No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Study</th>
<th>Starting Point</th>
<th>Constraints</th>
<th>Post-Processing</th>
<th>Params.</th>
<th>C10+</th>
<th>C100+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bayesian (SNOEK ET AL., 2012)</td>
<td>3 layers</td>
<td>FIXED ARCHITECTURE, NO SKIPS</td>
<td>NONE</td>
<td>–</td>
<td>90.5%</td>
<td>–</td>
</tr>
<tr>
<td>Q-Learning (BAKER ET AL., 2016)</td>
<td>–</td>
<td>DISCRETE PARAMS., MAX. NUM. LAYERS, NO SKIPS</td>
<td>TUNE, RETRAIN</td>
<td>11.2 M</td>
<td>93.1%</td>
<td>72.9%</td>
</tr>
<tr>
<td>RL (ZOPH &amp; LE, 2016)</td>
<td>20 layers, 50%</td>
<td>DISCRETE PARAMS., EXACTLY 20 LAYERS</td>
<td>SMALL GRID</td>
<td>2.5 M</td>
<td>94.0%</td>
<td>–</td>
</tr>
<tr>
<td>RL (ZOPH &amp; LE, 2016)</td>
<td>39 layers, 2 pool layers at 13 and 26, 50%</td>
<td>DISCRETE PARAMS., EXACTLY 39 LAYERS, 2 POOL LAYERS AT 13 AND 26</td>
<td>SMALL GRID SEARCH, RETRAIN</td>
<td>32.0 M</td>
<td>96.2%</td>
<td>–</td>
</tr>
<tr>
<td><strong>Evolution (ours)</strong></td>
<td>LINEAR MODEL, ZERO CONVS.</td>
<td>POWER-OF-2 STRIDES</td>
<td>NONE</td>
<td>5.4 M</td>
<td>94.6%</td>
<td>76.0%</td>
</tr>
</tbody>
</table>
Where are we trying to go?
Where are we trying to go?

Combine Several of These Ideas:

Large model, but sparsely activated
Single model to solve many tasks (100s to 1Ms)
Dynamically learn and grow pathways through large model
Tasks

Single large model, sparsely activated

Outputs
Single large model, sparsely activated
Single large model, sparsely activated

Tasks

Outputs
Single large model, sparsely activated

Tasks

Outputs
Tasks

Single large model, sparsely activated

Outputs
Single large model, sparsely activated
More computational power needed

Deep learning is transforming how we design computers
Special computation properties

reduced precision
ok

about 1.2
× about 0.6
about 0.7

NOT

1.21042
× 0.61127
0.73989343
Special computation properties

- handful of specific operations
- reduced precision ok

\[
\begin{align*}
\text{about 1.2} & \times \text{about 0.6} = \text{about 0.7} \\
1.21042 & \times 0.61127 \\
& = 0.73989343
\end{align*}
\]
Tensor Processing Unit

Custom Google-designed chip for neural net computations

In production use for >24 months: used on every search query, for neural machine translation, for AlphaGo match, …

Talk at Computer History Museum on April 5th:
sites.google.com/view/naeregionalsymposium
Machine Learning for Higher Performance Machine Learning Models
For large models, model parallelism is important
For large models, model parallelism is important.

But getting good performance given multiple computing devices is non-trivial and non-obvious.
Reinforcement Learning for Higher Performance Machine Learning Models
Placement model (trained via RL) gets graph as input + set of devices, outputs device placement for each graph node.
Reinforcement Learning for Higher Performance Machine Learning Models

Placement model (trained via RL) gets graph as input + set of devices, outputs device placement for each graph node.

Measured time per step gives RL reward signal.
Early results, but it seems to work

Per-step running times (secs)

<table>
<thead>
<tr>
<th>Model</th>
<th>Hardware</th>
<th>Baseline</th>
<th>RL</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neural MT (2 layers) + attention</td>
<td>4 Tesla K80</td>
<td>3.20s</td>
<td>2.47s</td>
<td>22.8%</td>
</tr>
<tr>
<td>Inception</td>
<td>4 Tesla K80</td>
<td>4.60s</td>
<td>3.85s</td>
<td>16.3%</td>
</tr>
</tbody>
</table>

Baselines:
NMT: human expert placement shown on earlier slide
Inception: default placement on GPU/0
Early results, but it seems to work

Per-step running times (secs)

<table>
<thead>
<tr>
<th>Model</th>
<th>Hardware</th>
<th>Baseline</th>
<th>RL</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neural MT (2 layers) + attention</td>
<td>4 Tesla K80</td>
<td>3.20s</td>
<td>2.47s</td>
<td>22.8%</td>
</tr>
<tr>
<td>Inception</td>
<td>4 Tesla K80</td>
<td>4.60s</td>
<td>3.85s</td>
<td>16.3%</td>
</tr>
</tbody>
</table>

Baselines:
NMT: human expert placement shown on earlier slide
Inception: default placement on GPU/0
Figure 4: Placement of the NMT graph. Due to space limit, we show only the last 12 steps of the encoder and the first 12 steps of the decoder. Devices are denoted by colors, where gray represents the CPU and each other colors represents a different GPU.
Now

Accuracy

Scale (data size, model size)

more compute

neural networks

other approaches
Future

Accuracy

Scale (data size, model size)

more compute

neural networks

other approaches
Example queries of the future

Which of these eye images shows symptoms of diabetic retinopathy?

Please fetch me a cup of tea from the kitchen

Describe this video in Spanish

Find me documents related to reinforcement learning for robotics and summarize them in German
Conclusions

Deep neural networks are making significant strides in speech, vision, language, search, robotics, healthcare, ...

If you’re not considering how to use deep neural nets to solve your problems, you almost certainly should be
More info about our work

<table>
<thead>
<tr>
<th>Main Research Areas</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine Learning Algorithms and Techniques</td>
</tr>
<tr>
<td>Computer Systems for Machine Learning</td>
</tr>
<tr>
<td>Natural Language Understanding</td>
</tr>
<tr>
<td>Perception</td>
</tr>
<tr>
<td>Healthcare</td>
</tr>
<tr>
<td>Robotics</td>
</tr>
<tr>
<td>Music and Art Generation</td>
</tr>
</tbody>
</table>

**More papers**

**Blog posts**
Join the Team

**Full Time Roles**
We’re looking for talented research scientists and software engineers enthusiastic about deep learning to join us.

**Brain Residency**
This 12-month program is designed to jumpstart your career in deep learning, working with our scientists and engineers from the Google Brain Team.

**Visiting Faculty**
Visiting Faculty work closely with our scientists and engineers, and have the opportunity to explore projects at industrial scale with state-of-the-art technology.

**Interns**
Our interns work on projects utilizing the latest techniques in deep learning. In your application, indicate your research interests in the ‘Cover letter/other notes’ section so it can be routed to the appropriate recruiter.

Thanks!