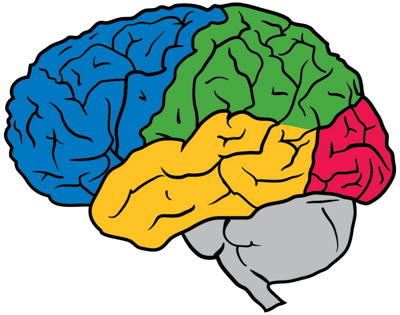


Recent Advances in Artificial Intelligence and the Implications for Computer System Design

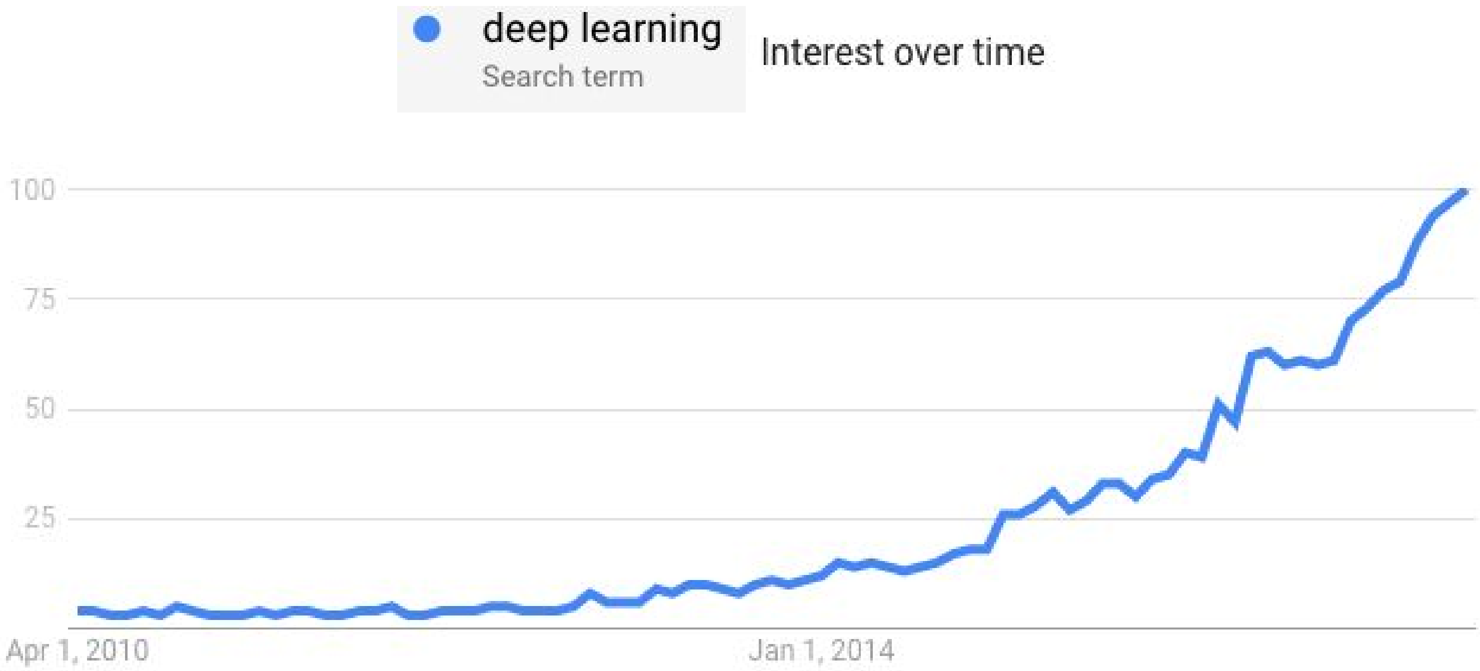
Jeff Dean

Google Brain team

g.co/brain

Presenting the work of **many** people at Google

Deep learning is causing a machine learning revolution



Deep Learning

Modern Reincarnation of Artificial Neural Networks

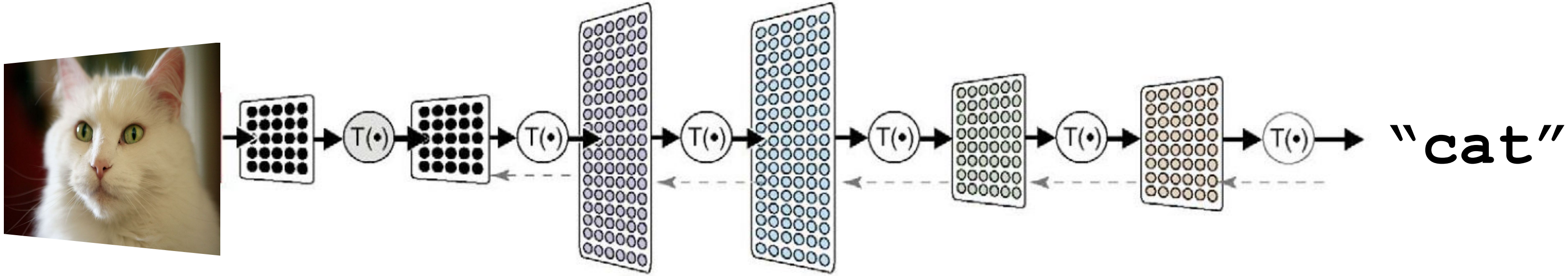
Collection of simple trainable mathematical units, organized in layers, that work together to solve complicated tasks

What's New

new network architectures,
new training math, ***scale***

Key Benefit

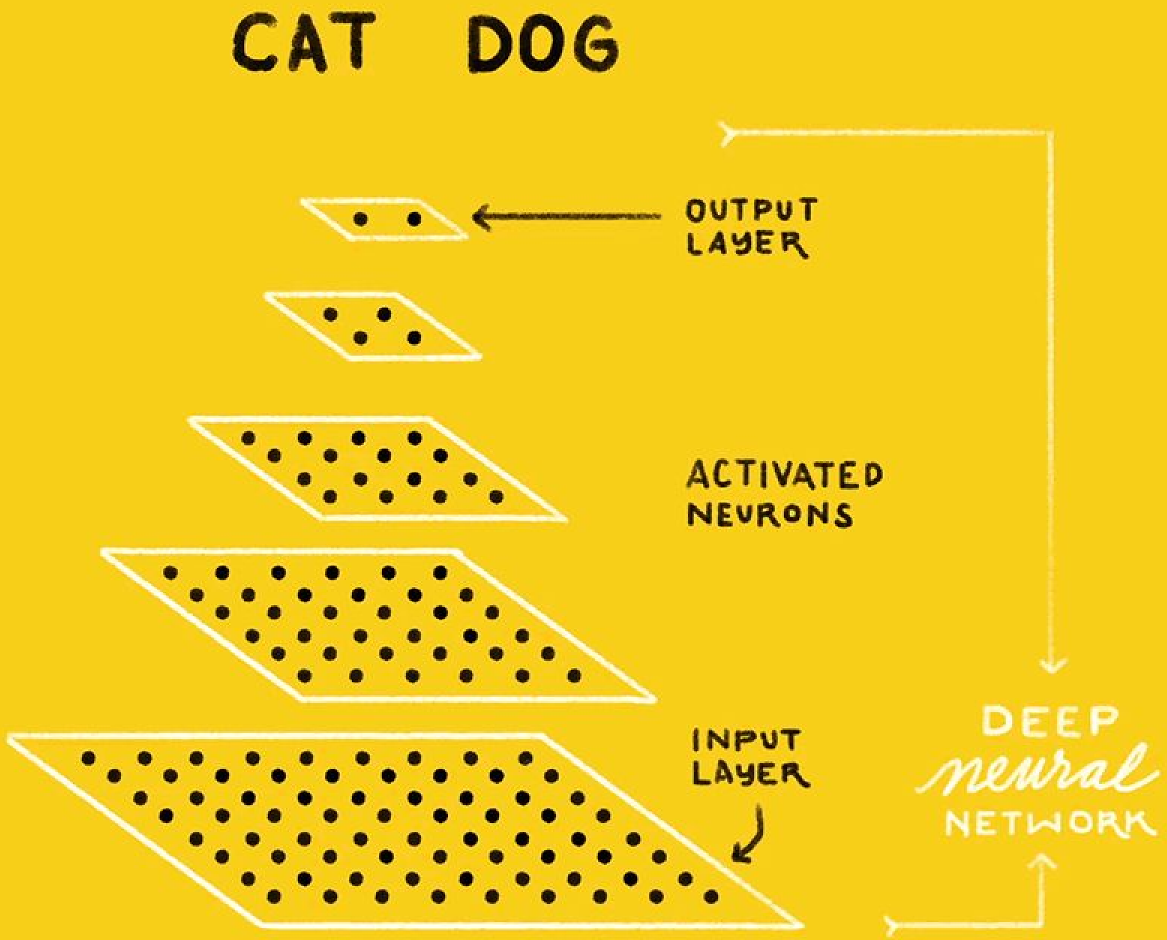
Learns features from raw, heterogeneous, noisy data
No explicit feature engineering required



Google

IS THIS A

CAT or DOG?

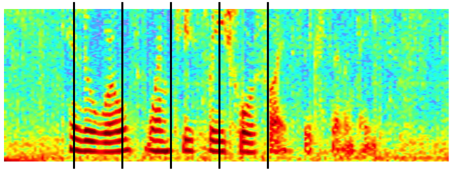


Functions a Deep Neural Network Can Learn

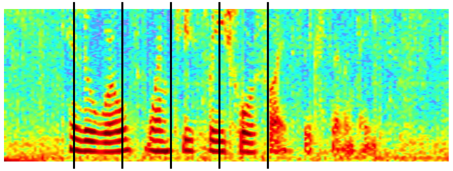
input | **output**

Pixels:  "lion"

Functions a Deep Neural Network Can Learn

	input	**output**
Pixels:		`"lion"`
Audio:		`"How cold is it outside?"`

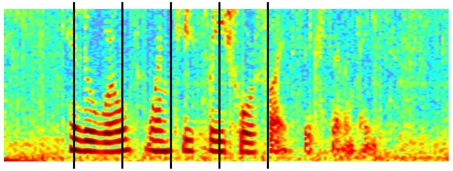
Functions a Deep Neural Network Can Learn

input		output
Pixels:		`“lion”`
Audio:		`“How cold is it outside?”`
`“Hello, how are you?”`		`“Bonjour, comment allez-vous?”`

Functions a Deep Neural Network Can Learn

input | **output**

Pixels:  "lion"

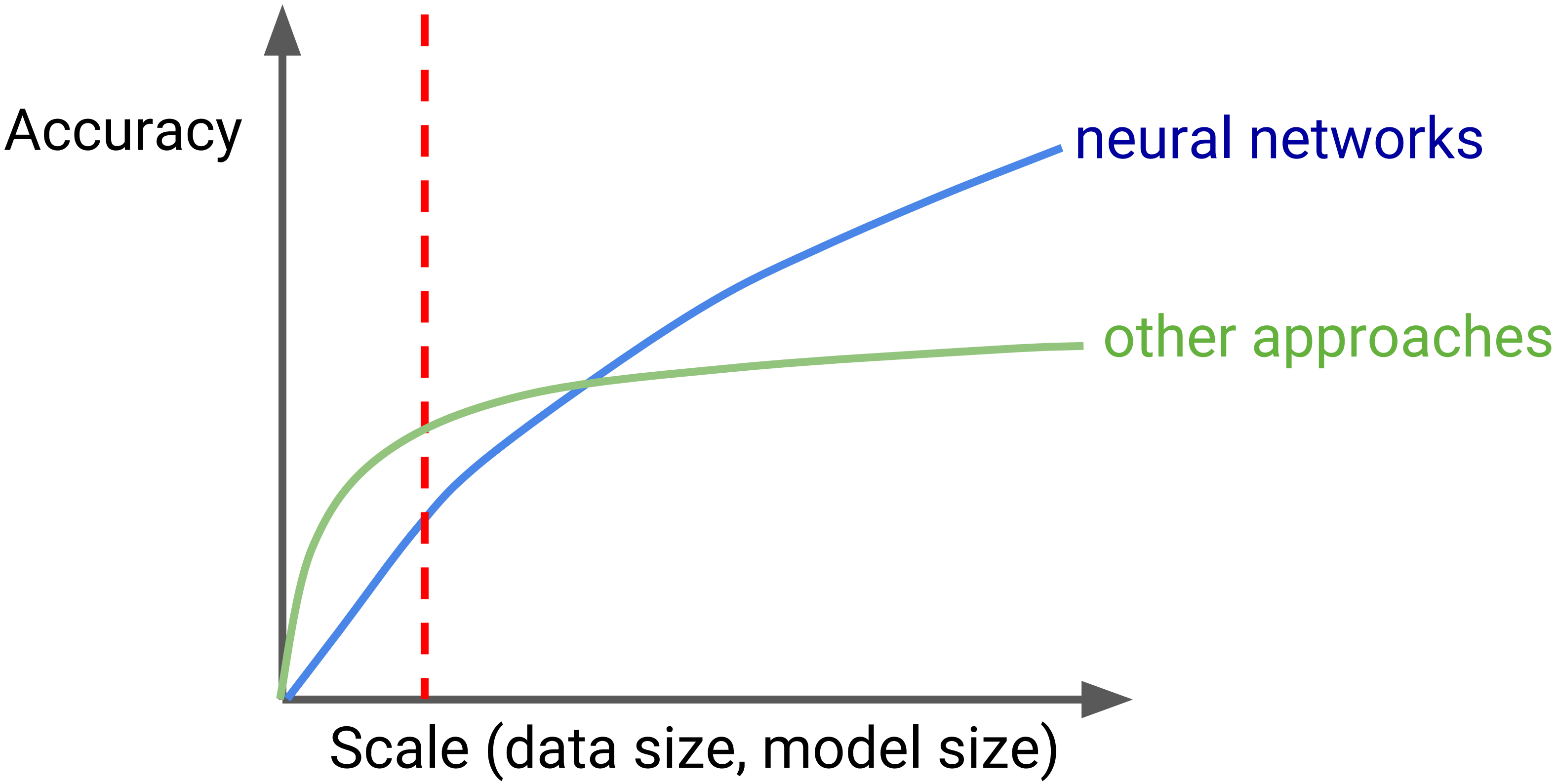
Audio:  "How cold is it outside?"

"Hello, how are you?" "Bonjour, comment allez-vous?"

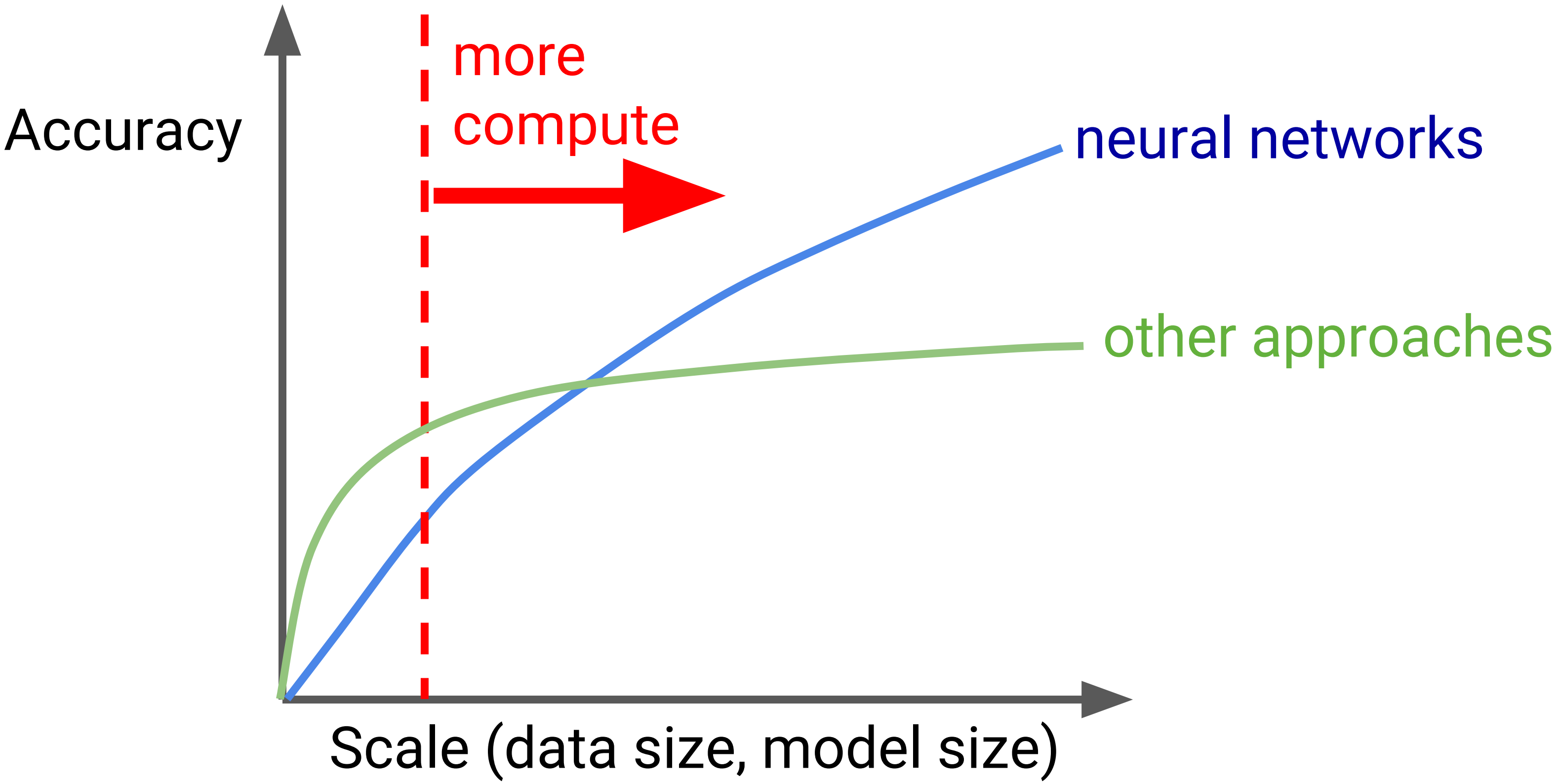
Pixels:  "A blue and yellow train travelling down the tracks"

But why now?

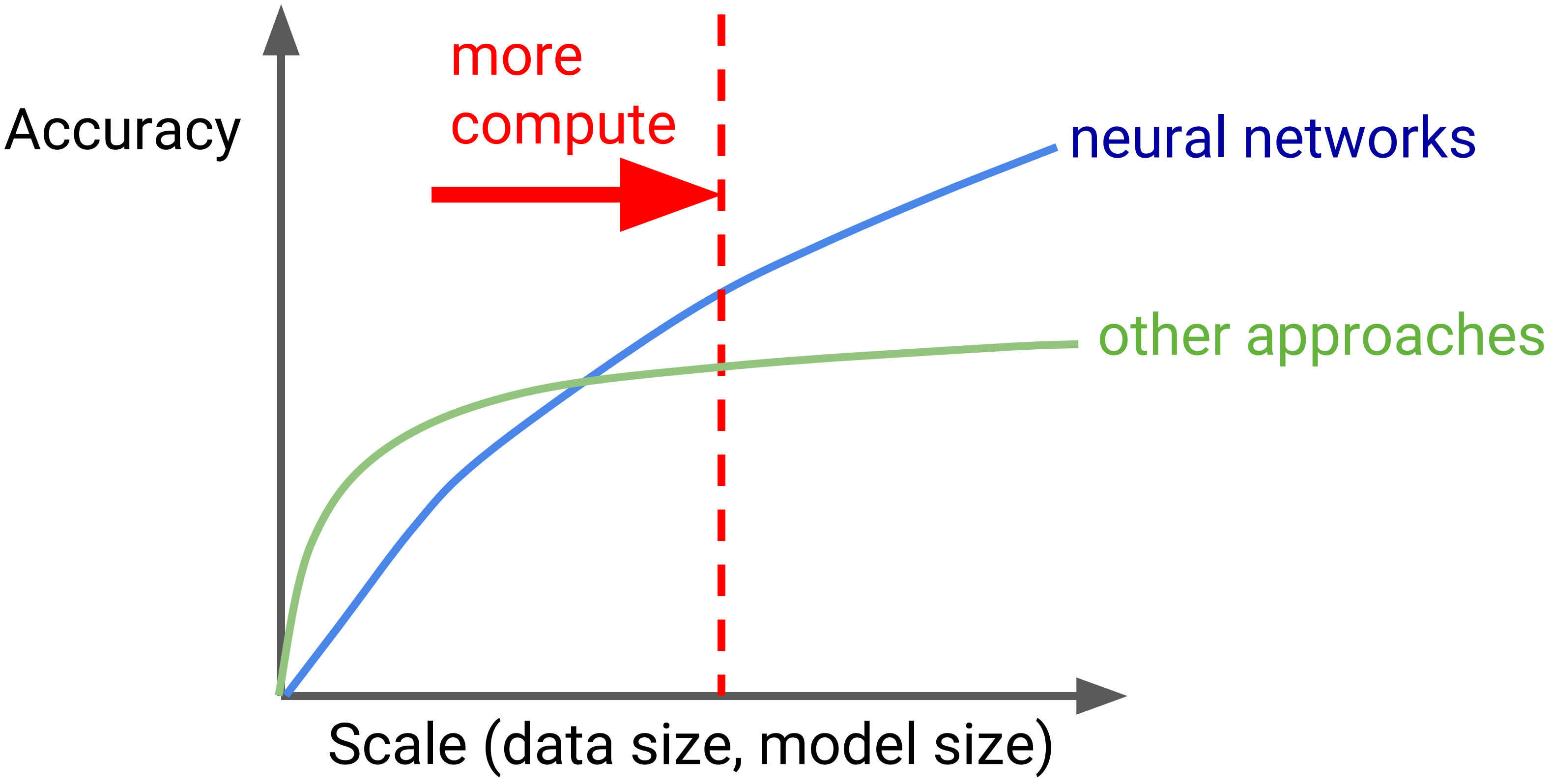
1980s and 1990s



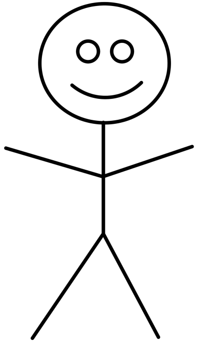
1980s and 1990s



Now

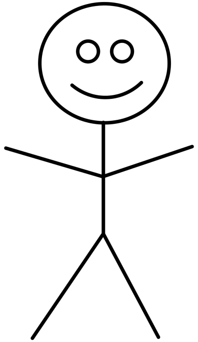

more
compute
Accuracy
neural networks
other approaches
Scale (data size, model size)

humans



5% errors

humans

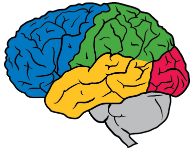


5% errors

2008: NAE Grand Engineering Challenges for 21st Century

- Make solar energy affordable
- Provide energy from fusion
- Develop carbon sequestration methods
- Manage the nitrogen cycle
- Provide access to clean water
- Restore & improve urban infrastructure
- Advance health informatics
- Engineer better medicines
- Reverse-engineer the brain
- Prevent nuclear terror
- Secure cyberspace
- Enhance virtual reality
- Advance personalized learning
- Engineer the tools for scientific discovery

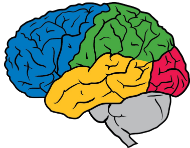
www.engineeringchallenges.org/challenges.aspx



2008: NAE Grand Engineering Challenges for 21st Century

- Make solar energy affordable
- Provide energy from fusion
- Develop carbon sequestration methods
- Manage the nitrogen cycle
- Provide access to clean water
- Restore & improve urban infrastructure
- Advance health informatics
- Engineer better medicines
- Reverse-engineer the brain
- Prevent nuclear terror
- Secure cyberspace
- Enhance virtual reality
- Advance personalized learning
- Engineer the tools for scientific discovery

www.engineeringchallenges.org/challenges.aspx



Restore & improve urban infrastructure

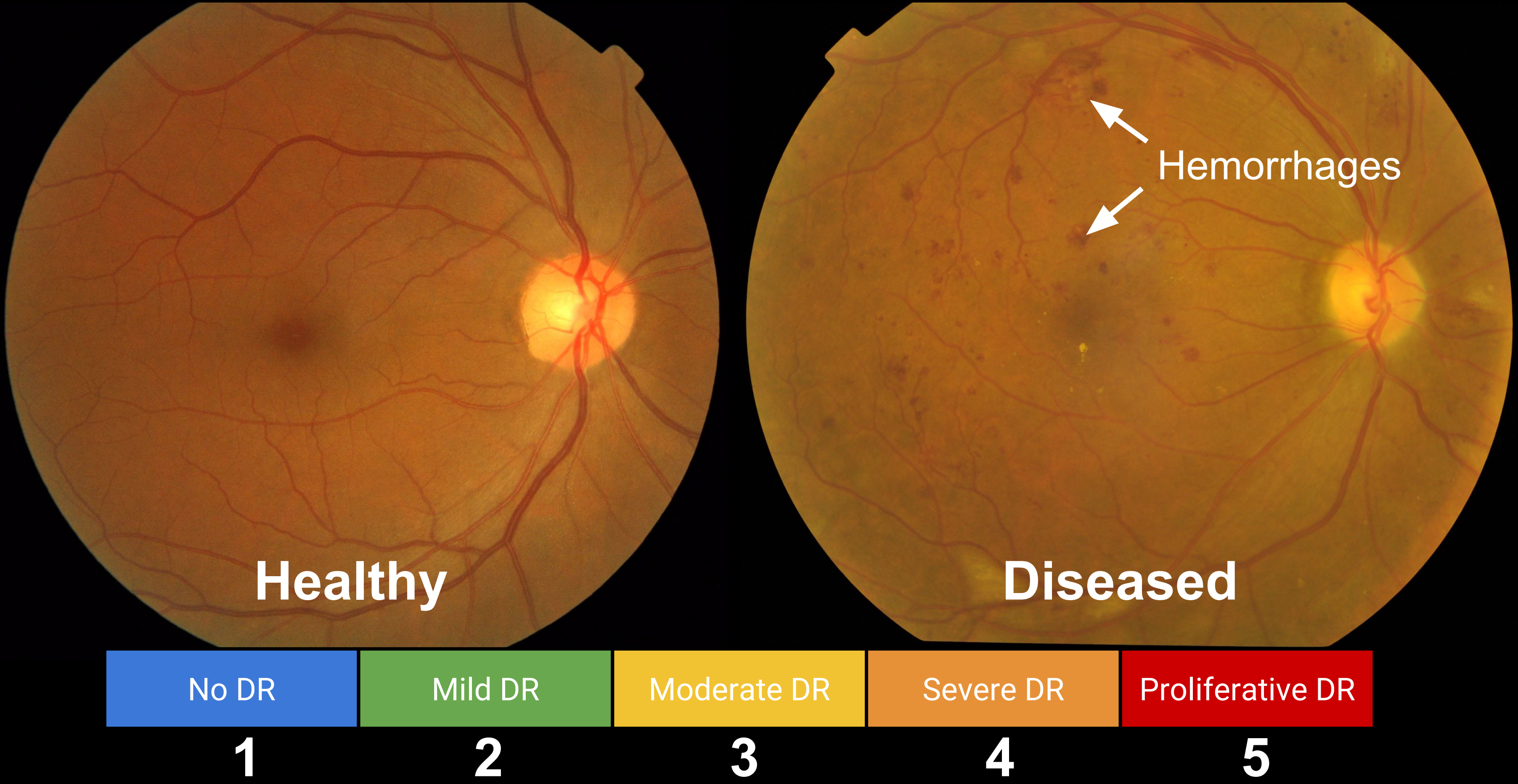
WAYMO

3 million miles self-driven

We drive more than 25,000 autonomous miles each week, largely on complex city streets. That's on top of 1 billion simulated miles we drove just in 2016.

https://waymo.com/tech/

Advance health informatics


Hemorrhages
Healthy
Diseased
No DR
Mild DR
Moderate DR
Severe DR
Proliferative DR
1
2
3
4
5

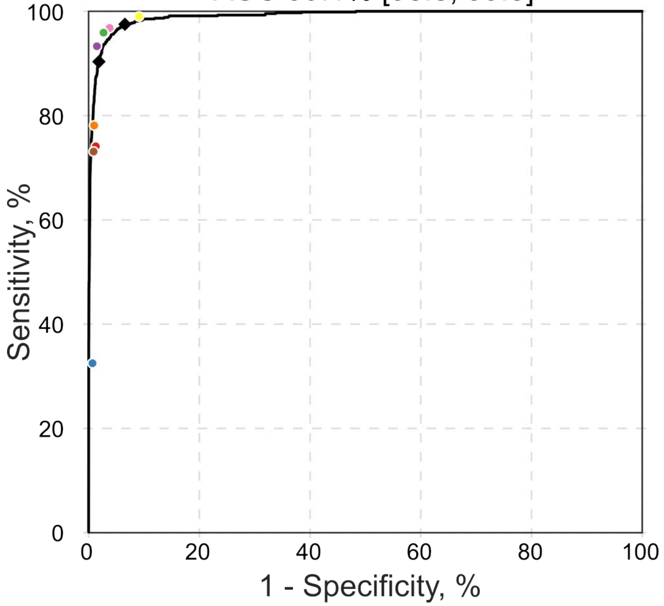
JAMA The Journal of the American Medical Association

JAMA | Original Investigation | INNOVATIONS IN HEALTH CARE DELIVERY

Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs

n = 9,963 images

AUC 99.1% [98.8, 99.3]



F-score

0.95
Algorithm

0.91
Ophthalmologist (median)

"The study by Gulshan and colleagues **truly represents the brave new world in medicine.**"

Dr. Andrew Beam, Dr. Isaac Kohane
Harvard Medical School

"Google just published this paper in JAMA (impact factor 37) [...] **It actually lives up to the hype**."

Dr. Luke Oakden-Rayner
University of Adelaide

Pathology

Detecting Cancer Metastases on Gigapixel Pathology Images

Yun Liu[1*], Krishna Gadepalli[1], Mohammad Norouzi[1], George E. Dahl[1], Timo Kohlberger[1], Aleksey Boyko[1], Subhashini Venugopalan[2**], Aleksei Timofeev[2], Philip Q. Nelson[2], Greg S. Corrado[1], Jason D. Hipp[3], Lily Peng[1], and Martin C. Stumpe[1]

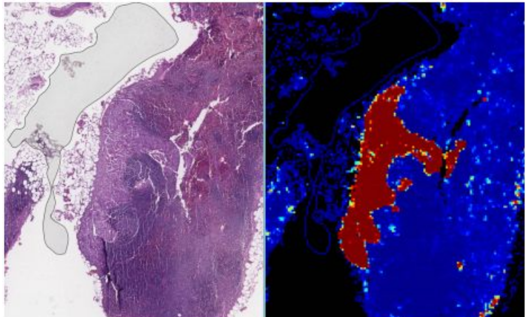
{liuyun,mnorouzi,gdahl,lhpeng,mstumpe}@google.com

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

Tumor localization score (FROC):

model:	**0.89**
pathologist:	0.73

arxiv.org/abs/1703.02442



Radiology

Acta Orthopaedica

Research-article

Artificial intelligence for analyzing orthopedic trauma radiographs

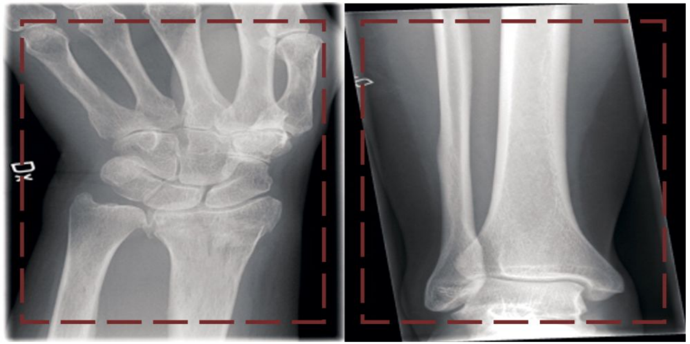
Deep learning algorithms—are they on par with humans for diagnosing fractures?

Jakub Olczak, Niklas Fahlberg, Atsuto Maki, Ali Sharif Razavian, Anthony Jilert, André Stark, Olof Sköldenberg & **Max Gordon** ...show less

Pages 1-6 | Received 01 Mar 2017, Accepted 06 Jun 2017, Published online: 06 Jul 2017

"The network performed similarly to senior orthopedic surgeons when presented with images at the same resolution as the network."

www.tandfonline.com/doi/full/10.1080/17453674.2017.1344459

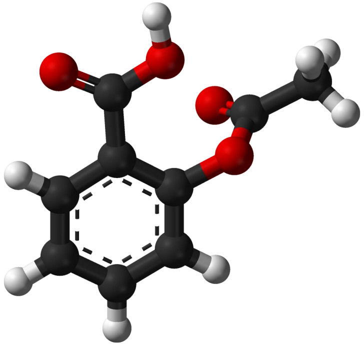


Engineer better medicines

and maybe...

Make solar energy affordable
Develop carbon sequestration methods
Manage the nitrogen cycle

Predicting Properties of Molecules



DFT (density functional theory) simulator

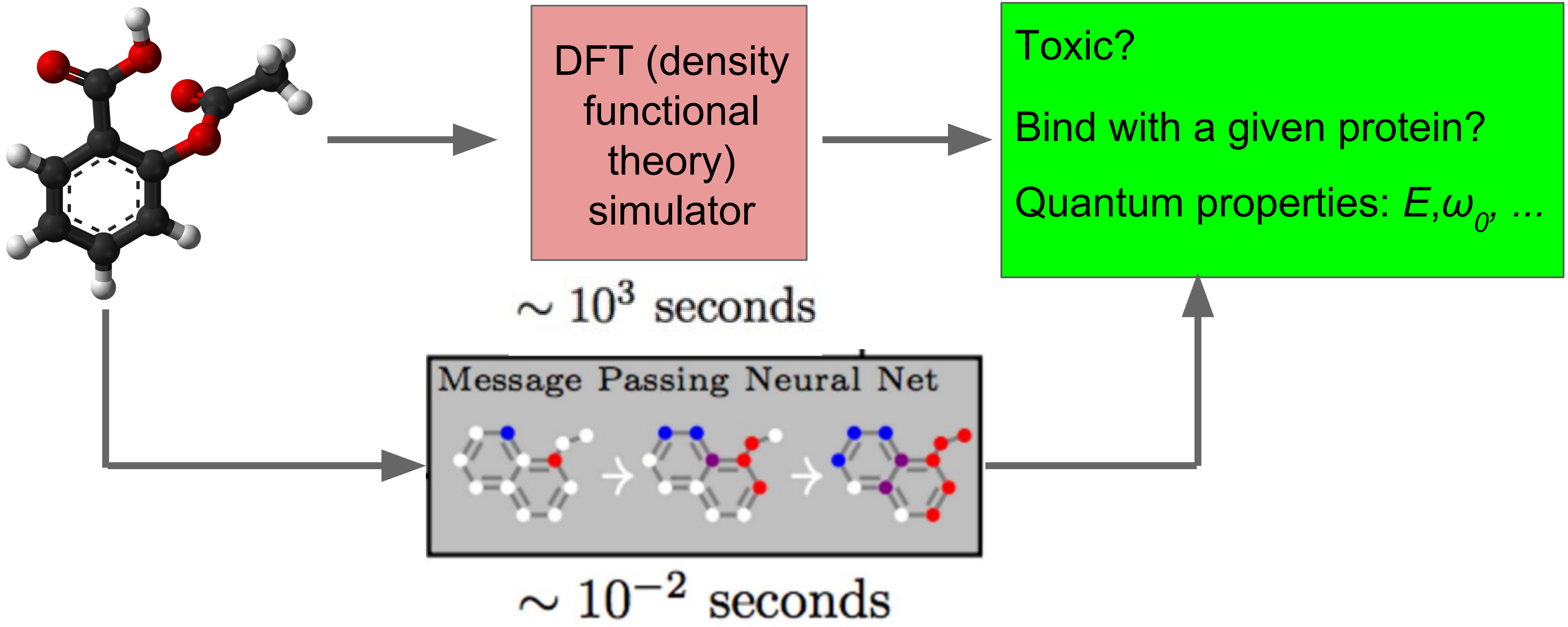
$\sim 10^3$ seconds

Toxic?

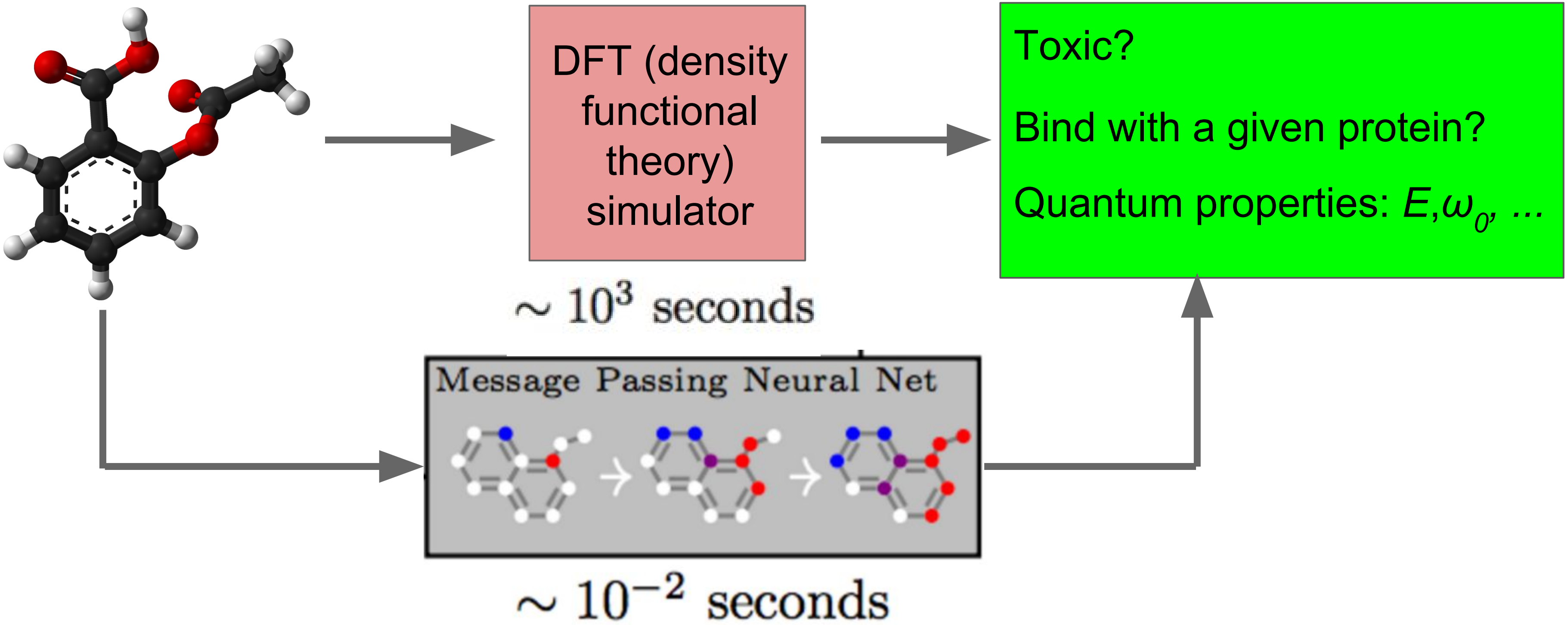
Bind with a given protein?

Quantum properties: $E, \omega_0, \ldots$

Predicting Properties of Molecules



Predicting Properties of Molecules

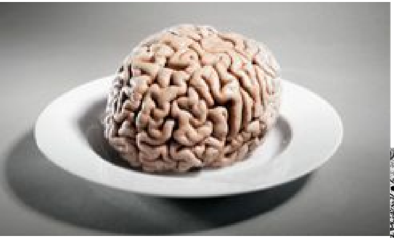


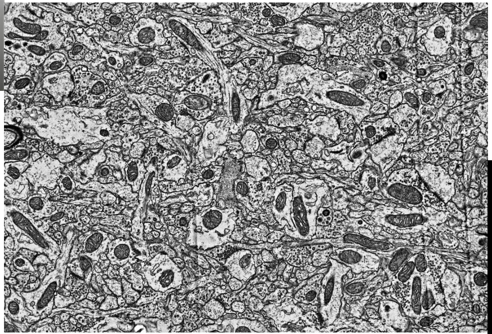
- State of the art results predicting output of expensive quantum chemistry calculations, but **~300,000 times faster**

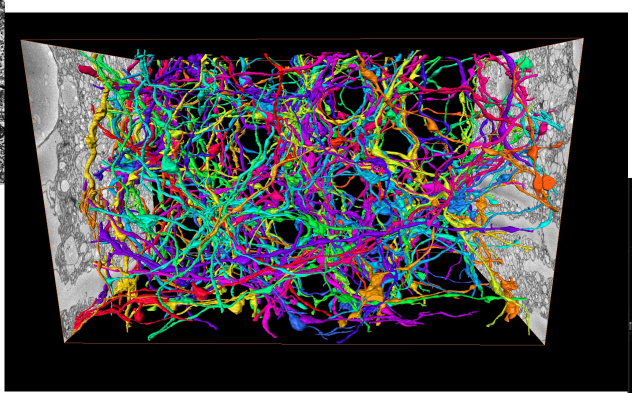
https://research.googleblog.com/2017/04/predicting-properties-of-molecules-with.html and https://arxiv.org/abs/1702.05532 and https://arxiv.org/abs/1704.01212 (appeared in ICML 2017)

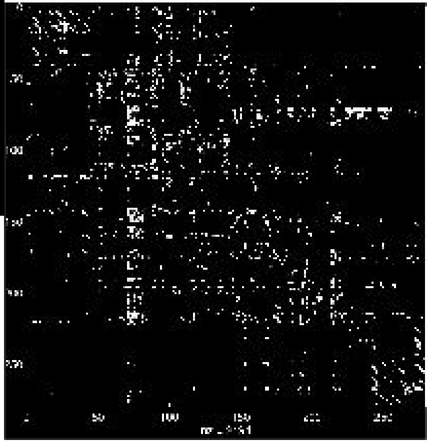
Reverse engineer the brain

Connectomics: Reconstructing Neural Circuits from High-Resolution Brain Imaging

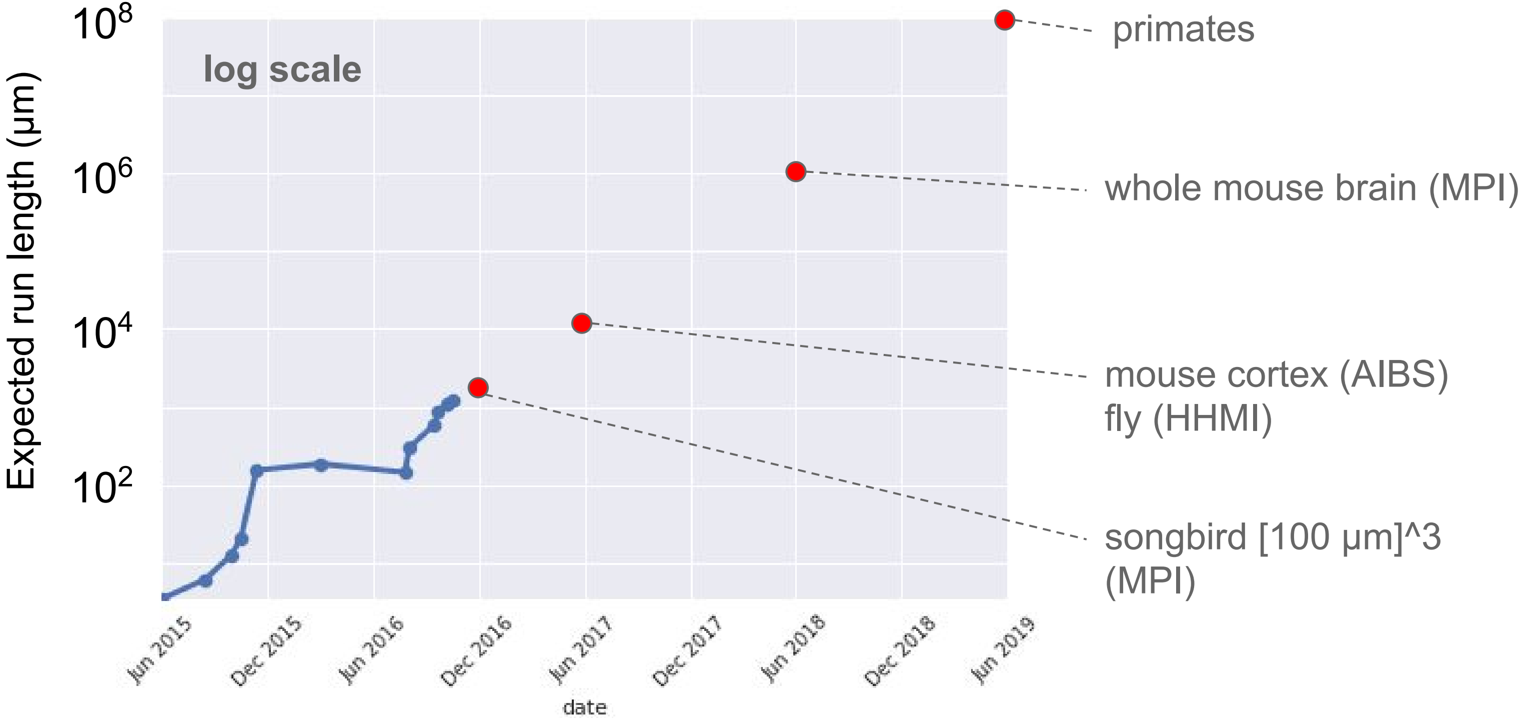








Automated Reconstruction Progress at Google



Metric: Expected Run Length (ERL)
"mean microns between failure" of automated neuron tracing

Google

New Technology: Flood Filling Networks

Flood-Filling Networks

Michał Januszewski
Google
mjanusz@google.com

Jeremy Maitin-Shepard
Google
jbms@google.com

Peter Li
Google
phli@google.com

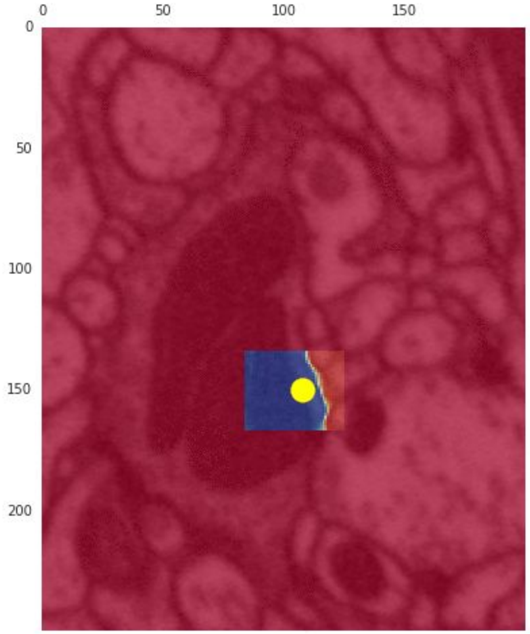
Jörgen Kornfeld
Max Planck Institute for Neurobiology
kornfeld@neuro.mpg.de

Winfried Denk
Max Planck Institute for Neurobiology
winfried.denk@neuro.mpg.de

Viren Jain
Google
viren@google.com

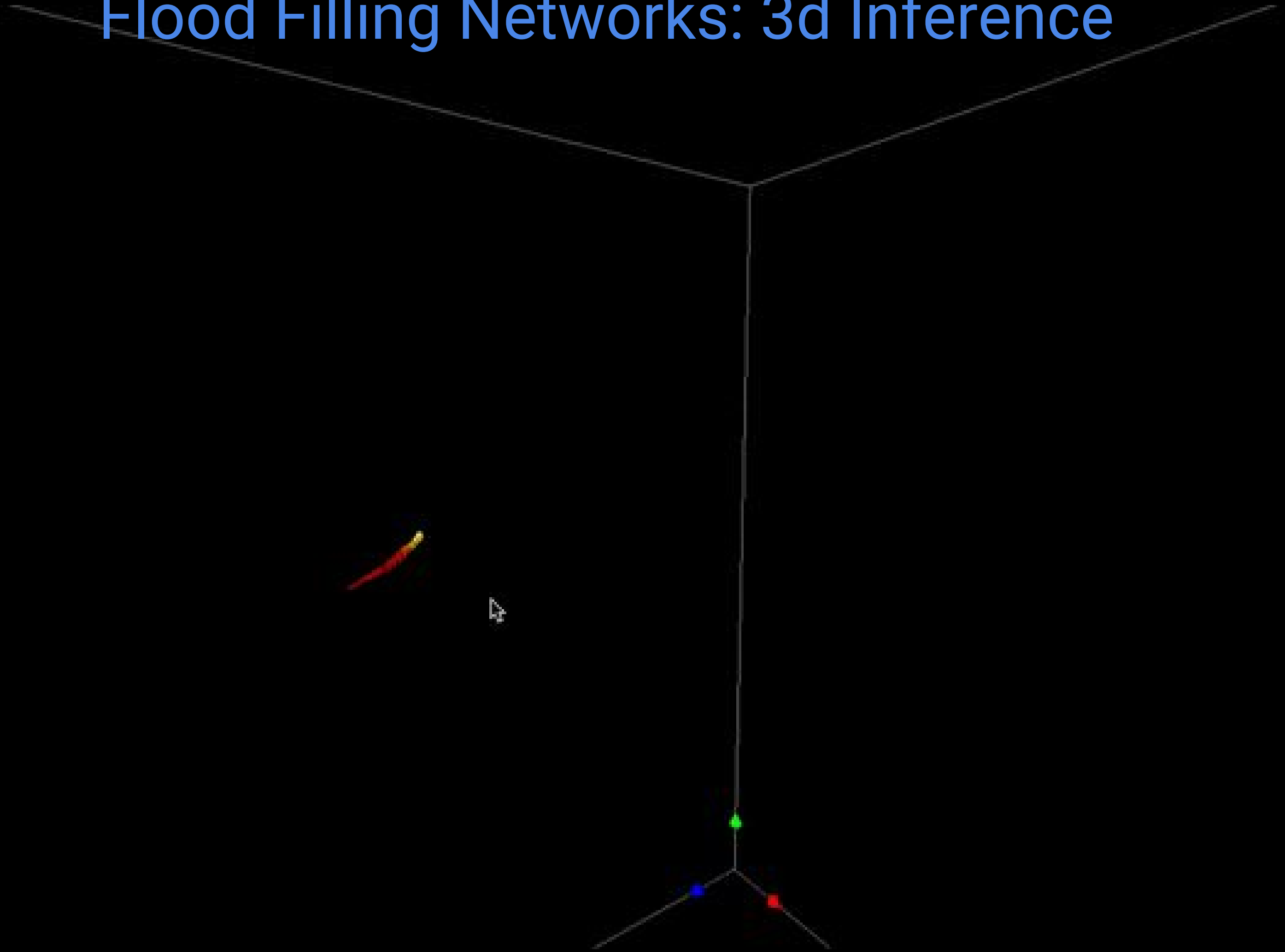
- Start with a seed point
- Recurrent neural network iteratively fills out an object based on image content and its own previous predictions

2d Inference

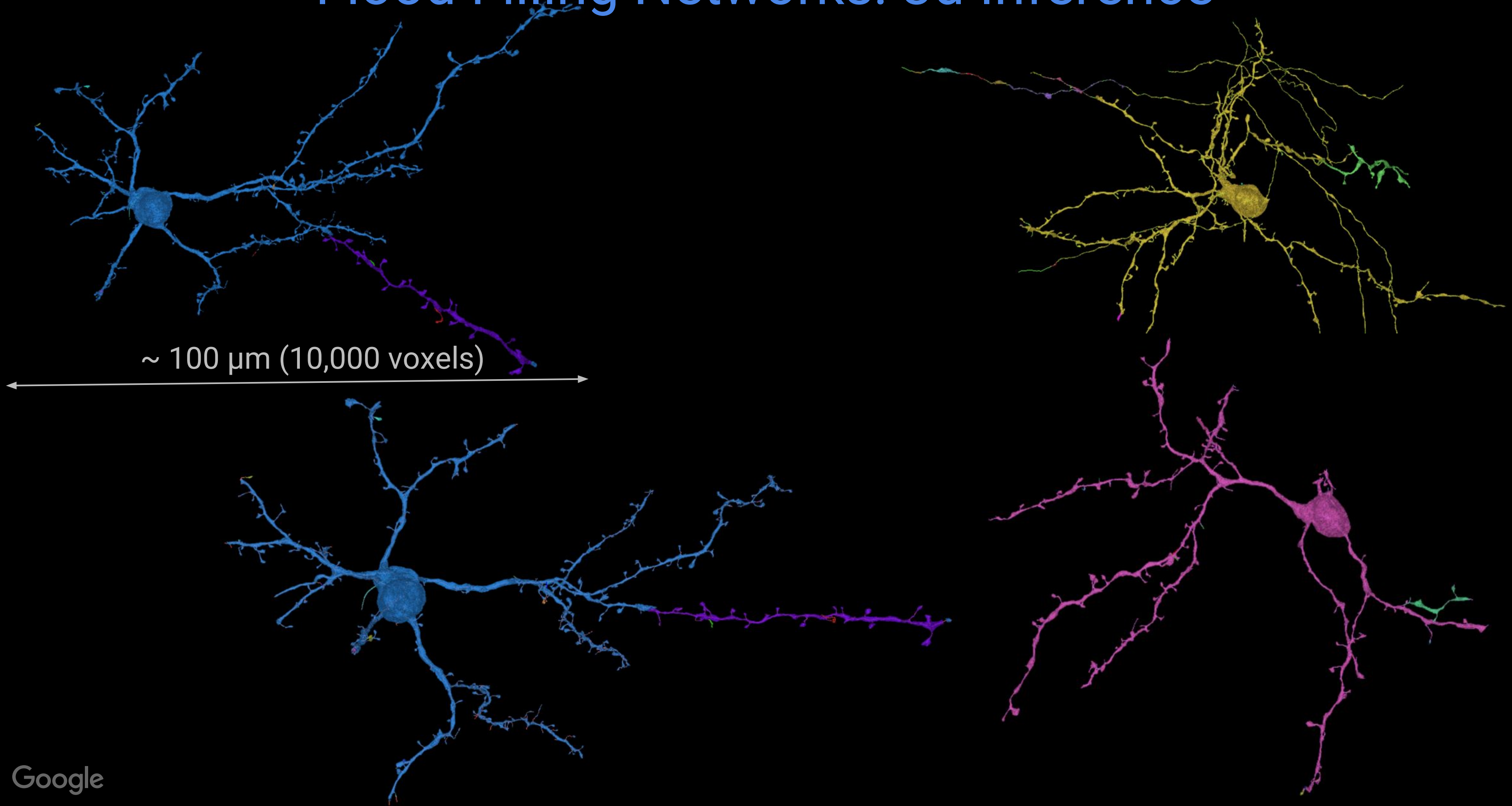


https://arxiv.org/abs/1611.00421

Flood Filling Networks: 3d Inference



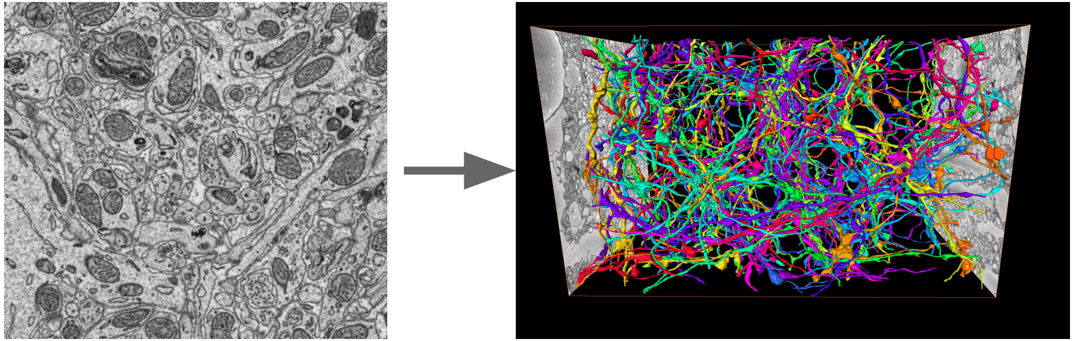
Flood Filling Networks: 3d Inference



Google

Songbird Brain Wiring Diagram

- Raw data produced by Max Planck Institute for Neurobiology using serial block face scanning electron microscopy
- 10,600 × 10,800 × 5,700 voxels =
  ~600 billion voxels
- Goal: Reconstruct **complete connectivity** and use to **test specific hypotheses** related to how biological nervous systems produce precise, sequential motor behaviors and perform reinforcement learning.



Courtesy Jorgen Kornfeld & Winfried Denk, MPI

Engineer the tools for scientific discovery

TensorFlow

http://tensorflow.org/

and

https://github.com/tensorflow/tensorflow

Open, standard software for general machine learning

Great for Deep Learning in particular

First released Nov 2015

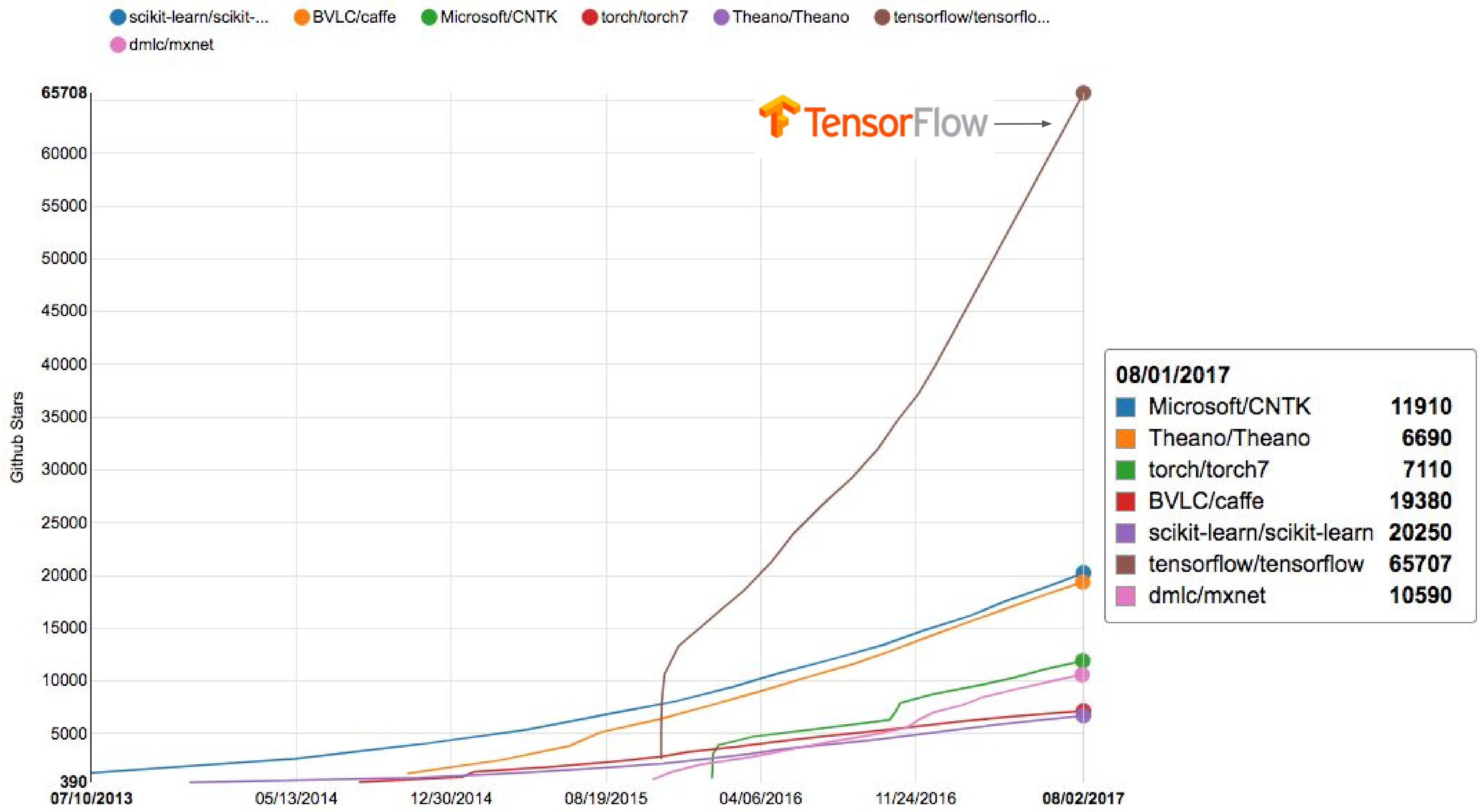
Apache 2.0 license

TensorFlow Goals

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

Open source it so that it becomes a **platform for everyone**, not just Google

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


scikit-learn/scikit-...
BVLC/caffe
Microsoft/CNTK
torch/torch7
Theano/Theano
tensorflow/tensorflo...
dmlc/mxnet
TensorFlow
Github Stars
65708
60000
55000
50000
45000
40000
35000
30000
25000
20000
15000
10000
5000
390
07/10/2013
05/13/2014
12/30/2014
08/19/2015
04/06/2016
11/24/2016
08/02/2017
08/01/2017
Microsoft/CNTK 11910
Theano/Theano 6690
torch/torch7 7110
BVLC/caffe 19380
scikit-learn/scikit-learn 20250
tensorflow/tensorflow 65707
dmlc/mxnet 10590

TensorFlow: A Vibrant Open-Source Community

- **Rapid development, many outside contributors**
  - ~800+ non-Google contributors to TensorFlow
  - 21,000+ commits in 21 months
  - Many community created tutorials, models, translations, and projects
    - ~16,000 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, ...**

More computational power needed

Deep learning is transforming how we design computers

Google

Special computation properties

reduced precision ok

$$\begin{array}{r} \text{about } 1.2 \\ \times\ \text{about } 0.6 \\ \hline \text{about } 0.7 \end{array} \quad \textbf{NOT} \quad \cancel{\begin{array}{r} 1.21042 \\ \times\ 0.61127 \\ \hline 0.73989343 \end{array}}$$

Special computation properties

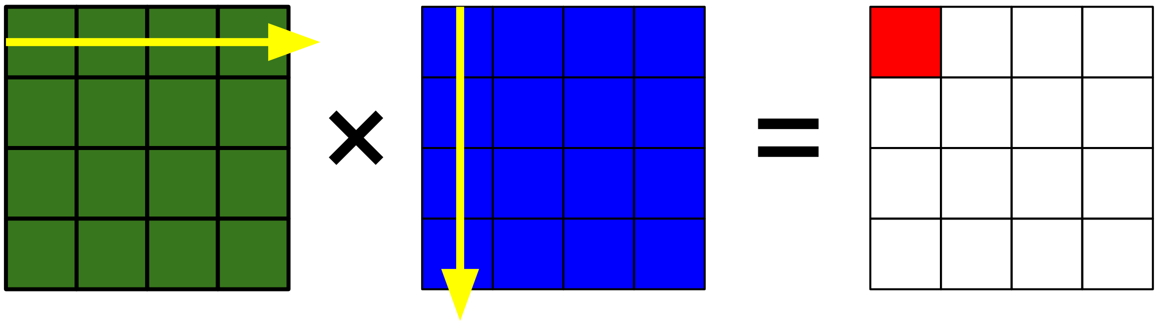
reduced precision ok

about 1.2
× about 0.6
about 0.7

NOT

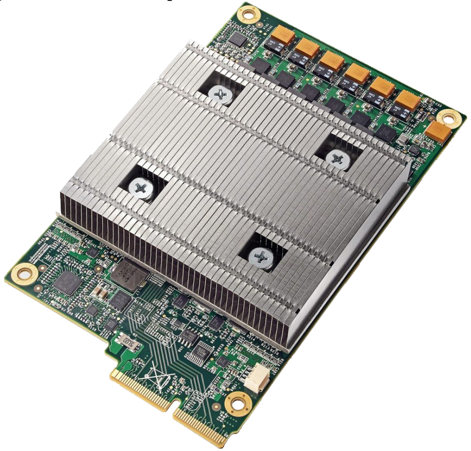
~~1.21042~~
~~× 0.61127~~
~~0.73989343~~

handful of specific operations



Tensor Processing Unit v1

Google-designed chip for neural net **inference**

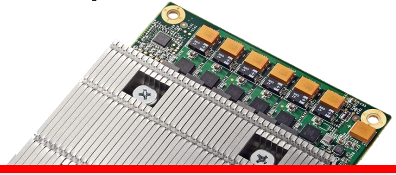


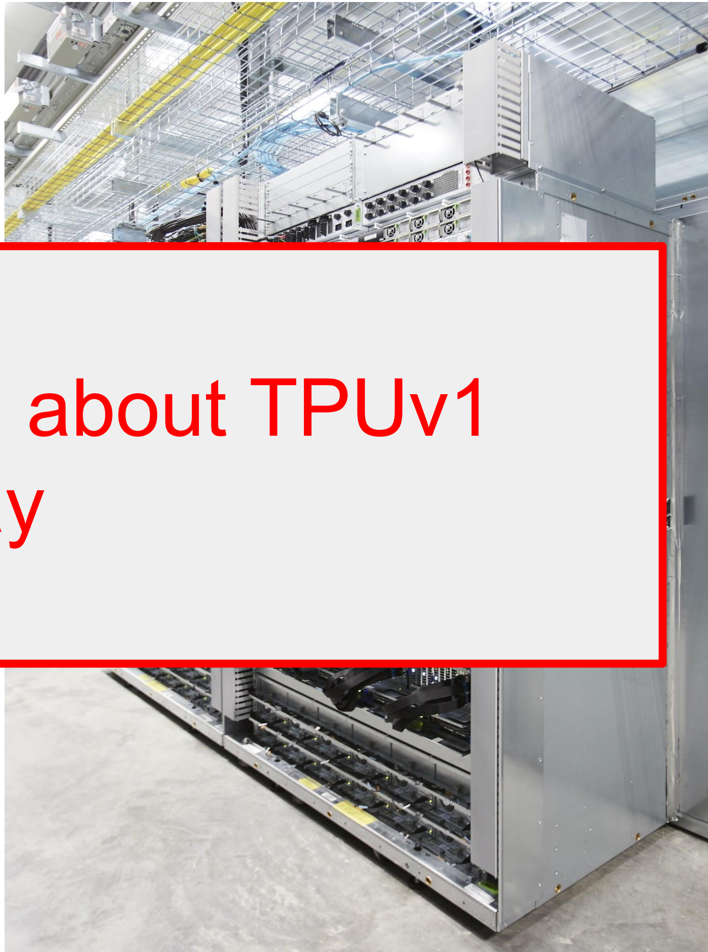
In production use for >30 months: used on search queries, for neural machine translation, for AlphaGo match, …

In-Datacenter Performance Analysis of a Tensor Processing Unit, Jouppi, Young, Patil, Patterson et al., ISCA 2017, arxiv.org/abs/1704.04760

Tensor Processing Unit v1

Google-designed chip for neural net **inference**





Cliff Young will tell you all about TPUv1
at 3 PM today

In pro
querie
for AlphaGo match, …

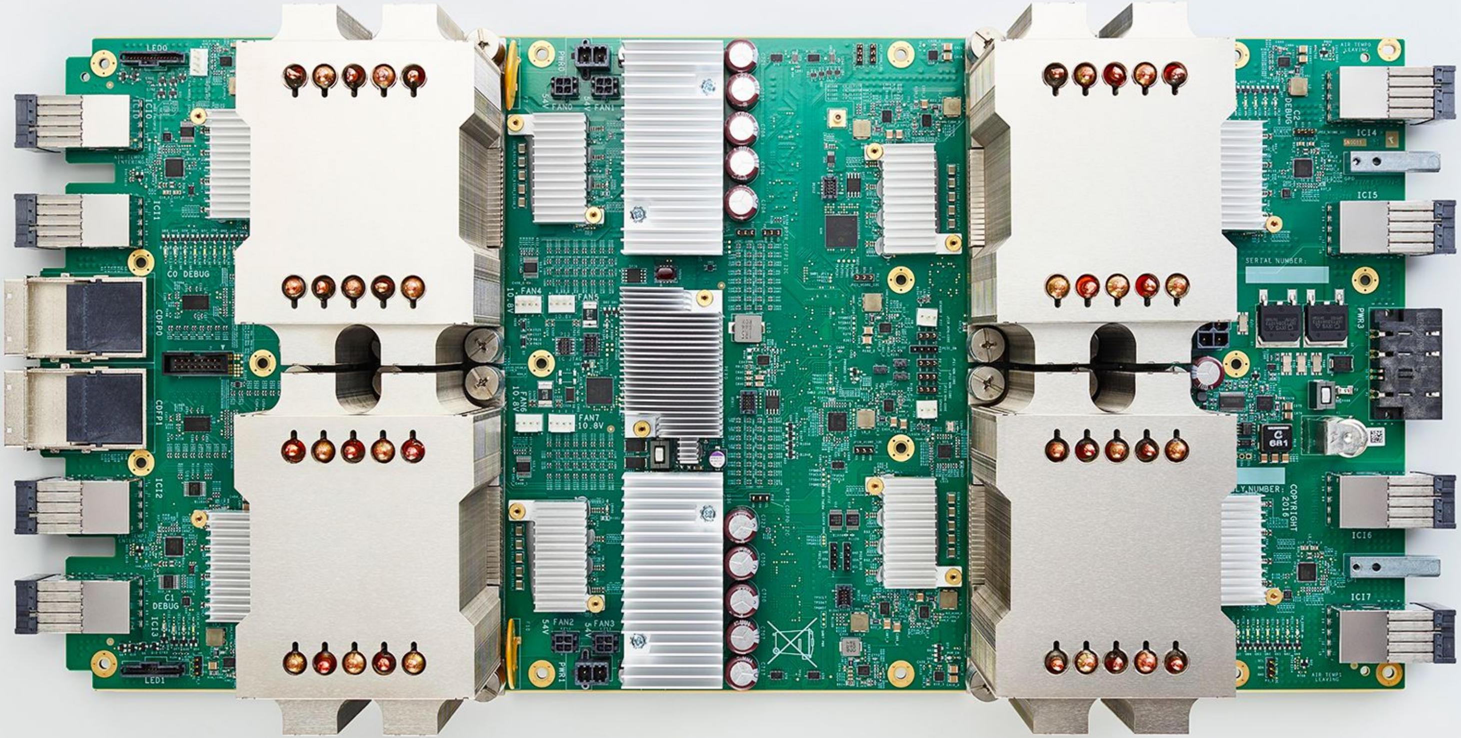
In-Datacenter Performance Analysis of a Tensor Processing Unit, Jouppi, Young, Patil, Patterson et al., ISCA 2017, arxiv.org/abs/1704.04760

TPUv1 is a huge help for inference

But what about training?

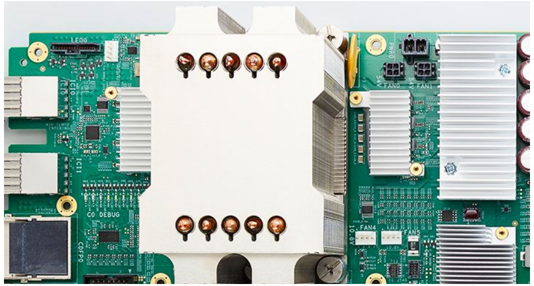
Speeding up training hugely important:
for researcher productivity, and
for increasing scale of problems that can be tackled

Tensor Processing Unit v2

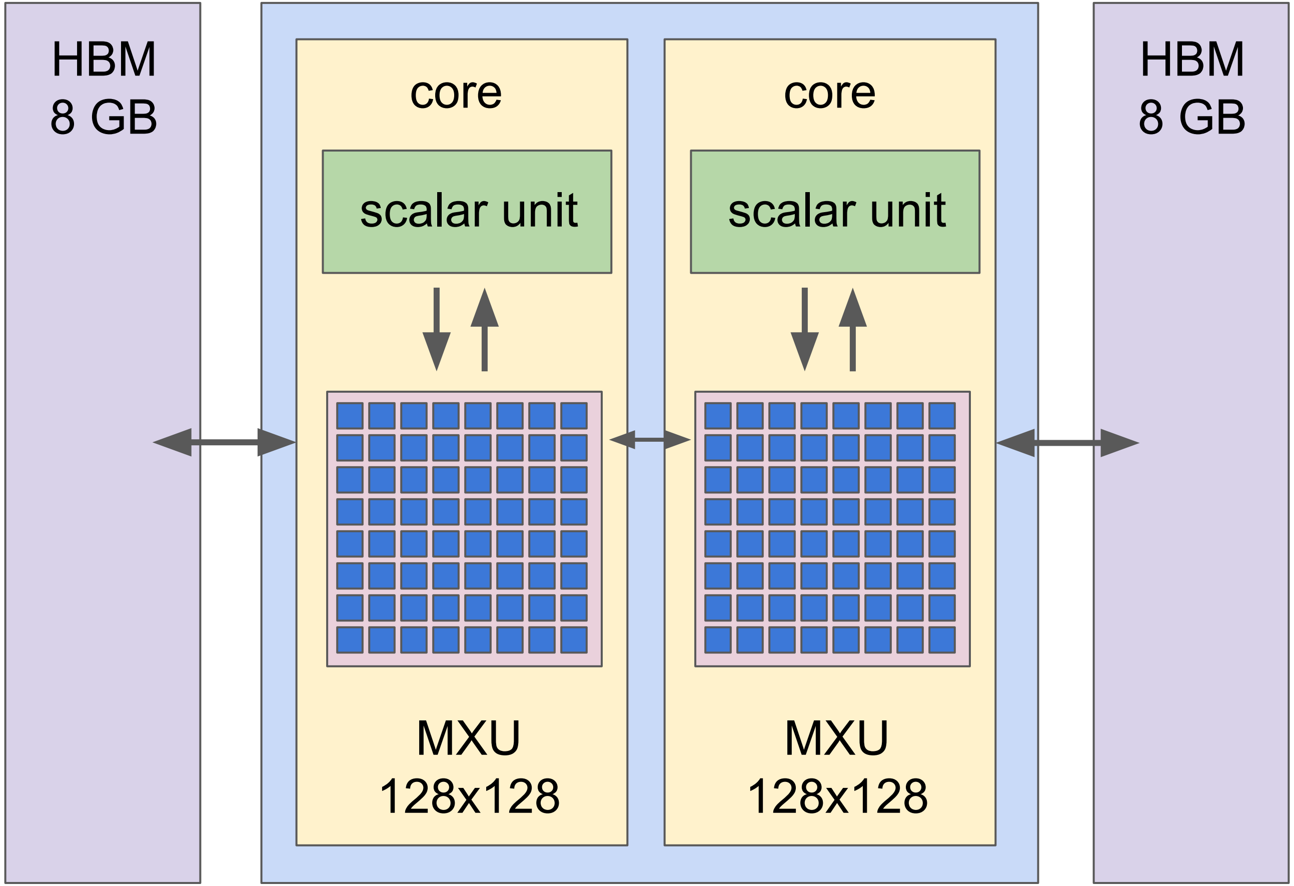


Google-designed device for neural net **training** and **inference**

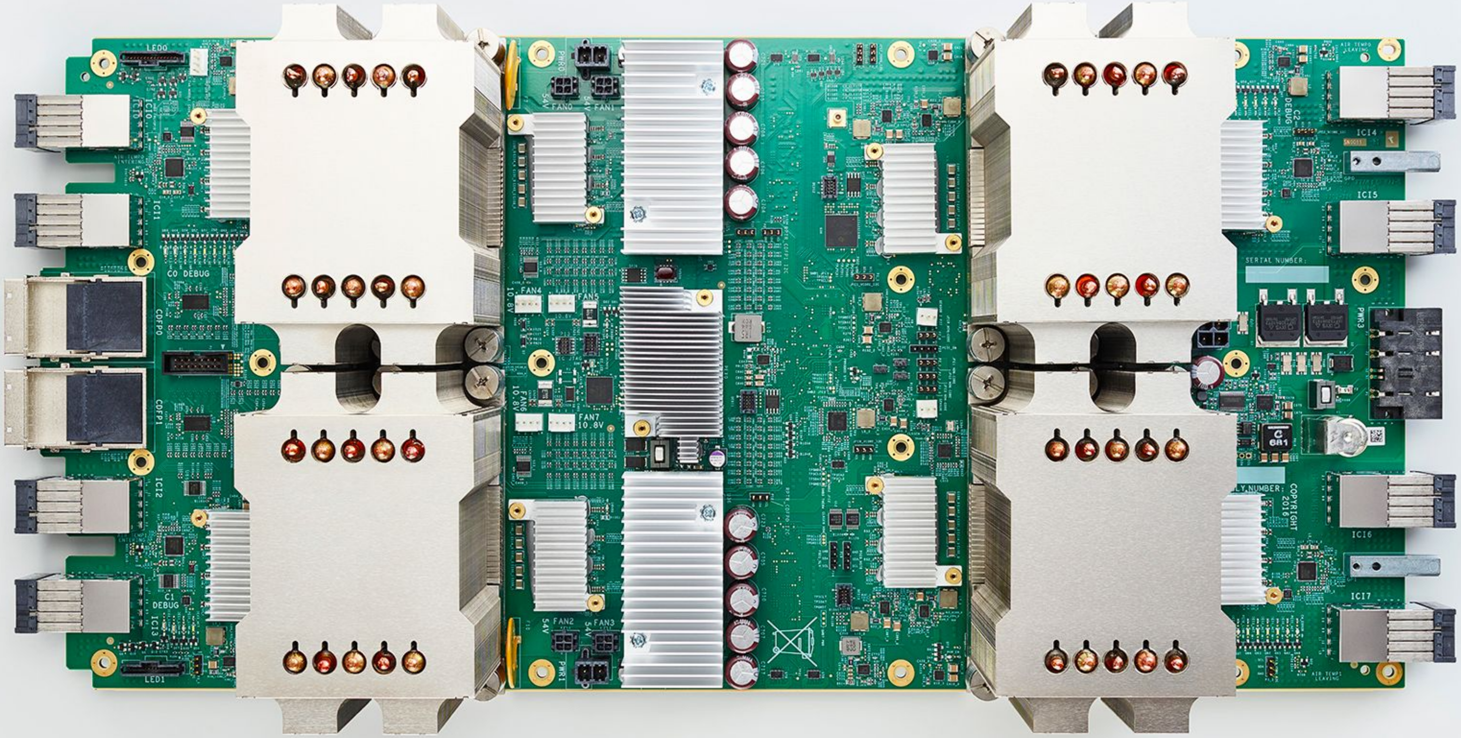
TPUv2 Chip



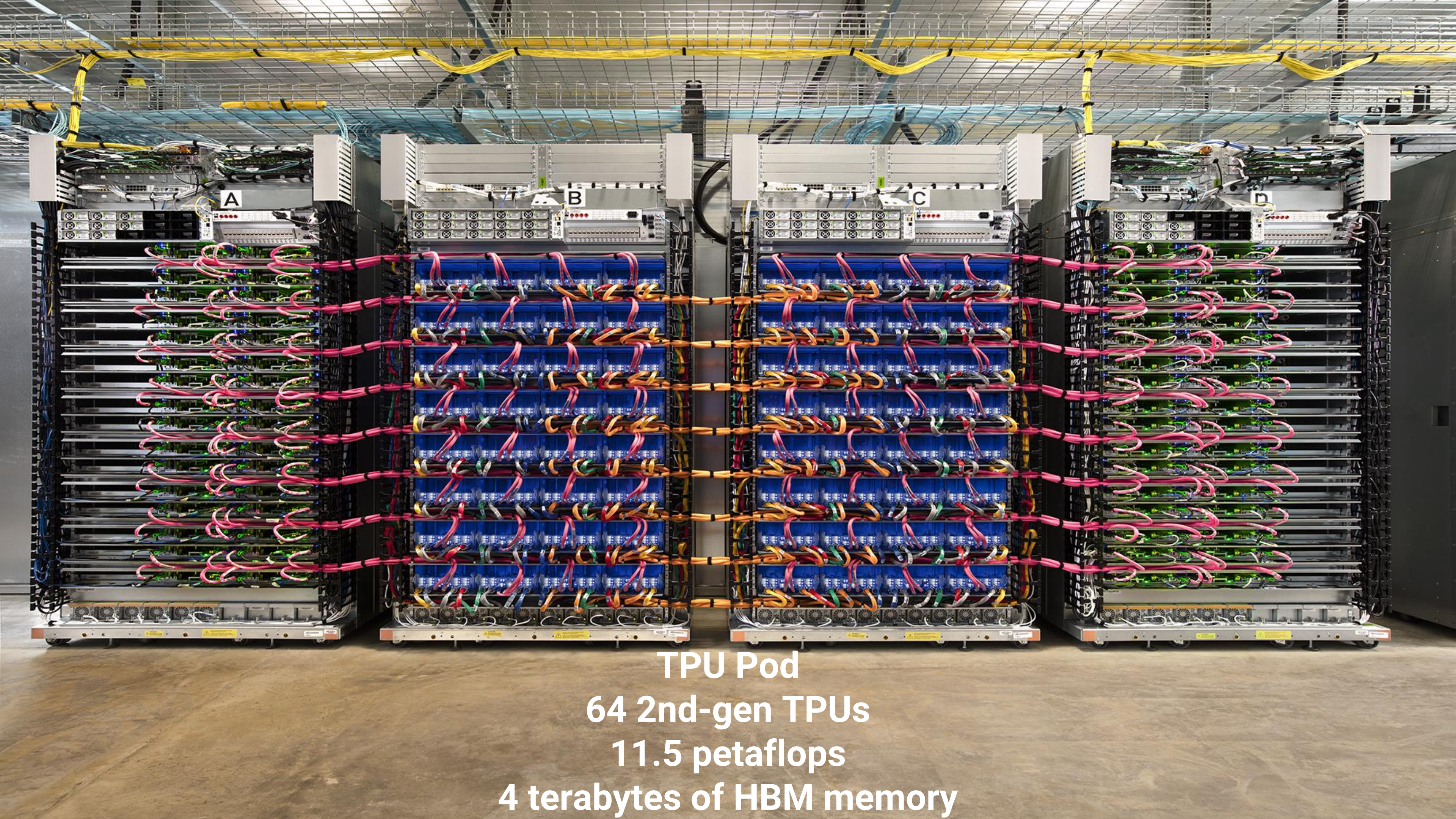
- 16 GB of HBM
- 600 GB/s mem BW
- Scalar unit: 32b float
- MXU: 32b float accumulation but reduced precision for multipliers
- 45 TFLOPS



Tensor Processing Unit v2



- 180 teraflops of computation, 64 GB of HBM memory, 2400 GB/s mem BW
- Designed to be connected together into larger configurations

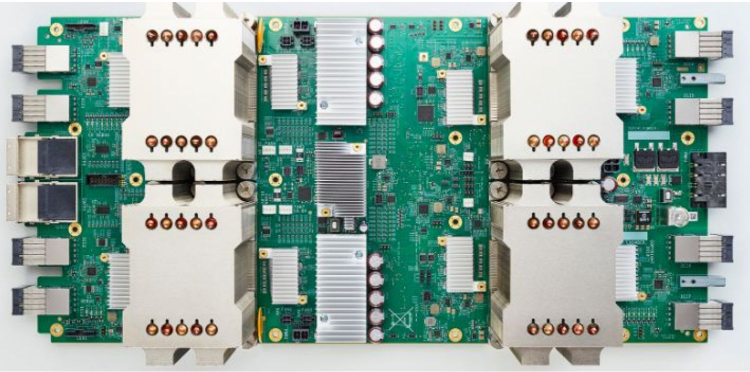

A
B
C
D
TPU Pod
64 2nd-gen TPUs
11.5 petaflops
4 terabytes of HBM memory

Programmed via TensorFlow

Same program will run with only minor modifications on CPUs, GPUs, & TPUs

Will be Available through Google Cloud

Cloud TPU - virtual machine w/180 TFLOPS TPUv2 device attached



TensorFlow
RESEARCH CLOUD

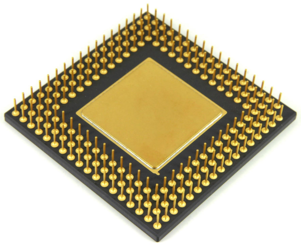
Making 1000 **Cloud TPUs** available **for free** to top researchers who are committed to **open machine learning research**

We're excited to see what researchers will do with much more computation!

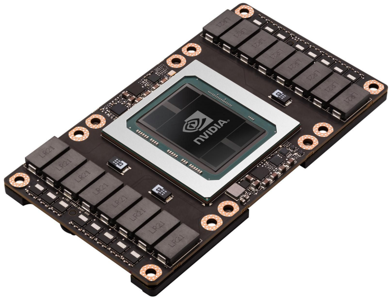
g.co/tpusignup

Machine learning needs to run in a growing set of environments

TensorFlow supports many platforms

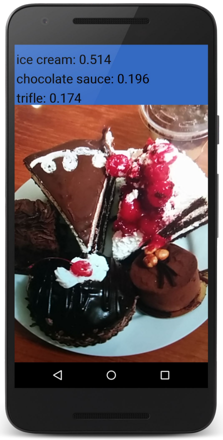


CPU

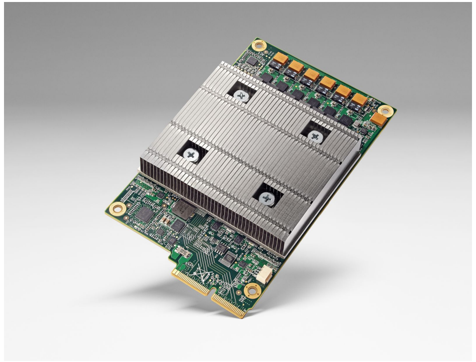


GPU

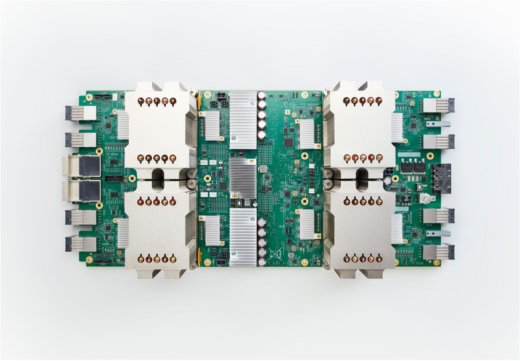
iOS



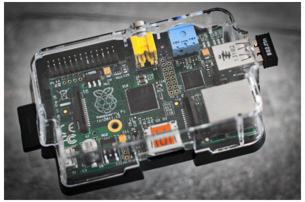
Android



1st-gen TPU



Cloud TPU



Raspberry Pi

TensorFlow supports many languages

Haskell

C++

Java

python™

julia

Go

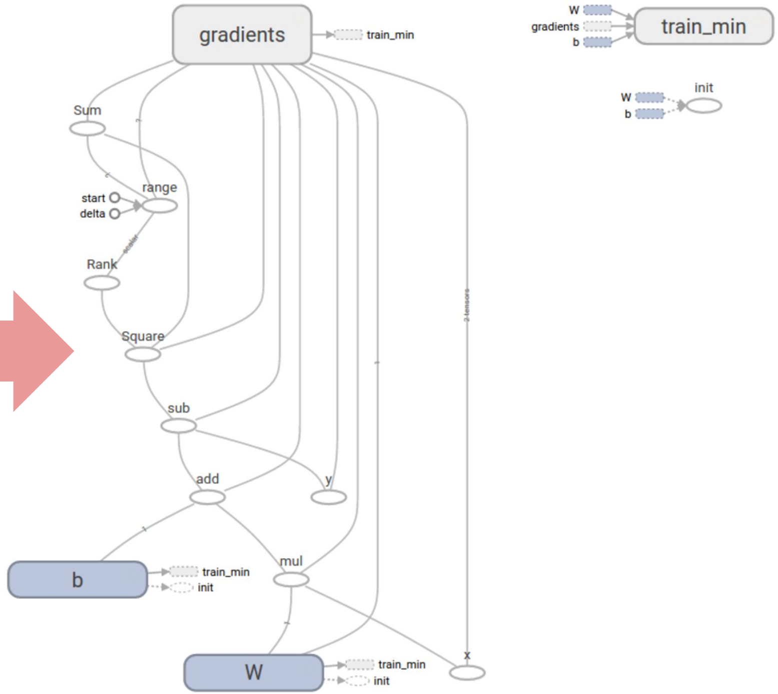
C#

TensorFlow Graph

Python Program

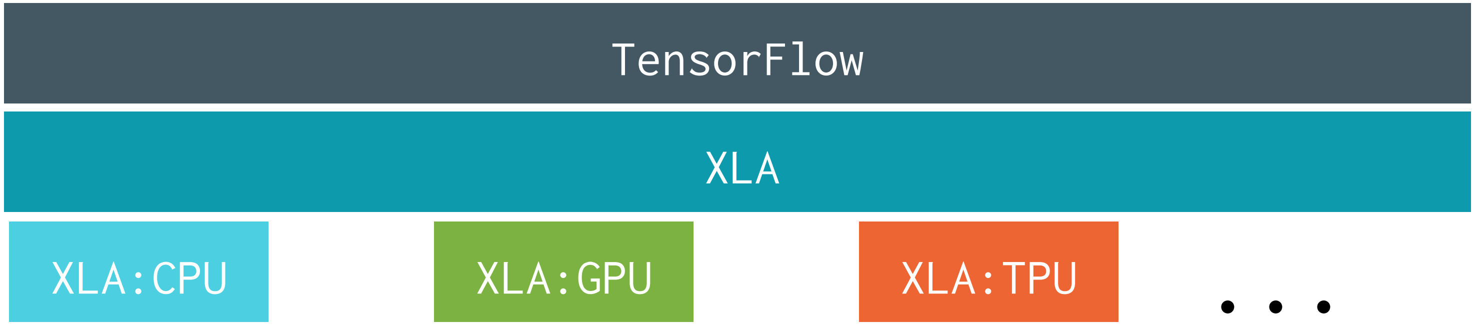
```
import numpy as np
import tensorflow as tf

# Model parameters
W = tf.Variable([.3], tf.float32)
b = tf.Variable([-.3], tf.float32)
# Model input and output
x = tf.placeholder(tf.float32)
linear_model = W * x + b
y = tf.placeholder(tf.float32)
# loss
loss = tf.reduce_sum(tf.square(linear_model - y)) # sum of the squares
# optimizer
optimizer = tf.train.GradientDescentOptimizer(0.01)
train = optimizer.minimize(loss)
# training data
x_train = [1,2,3,4]
y_train = [0,-1,-2,-3]
# training loop
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init) # reset values to wrong
for i in range(1000):
  sess.run(train, {x:x_train, y:y_train})

# evaluate training accuracy
curr_W, curr_b, curr_loss  = sess.run([W, b, loss], {x:x_train, y:y_train})
print("W: %s b: %s loss: %s"%(curr_W, curr_b, curr_loss))
```

TensorFlow Graph



https://www.tensorflow.org/get_started/get_started

TensorFlow + XLA Compiler



See: https://www.tensorflow.org/performance/xla/

Open-sourced code in https://github.com/tensorflow/tensorflow/tree/master/tensorflow/compiler

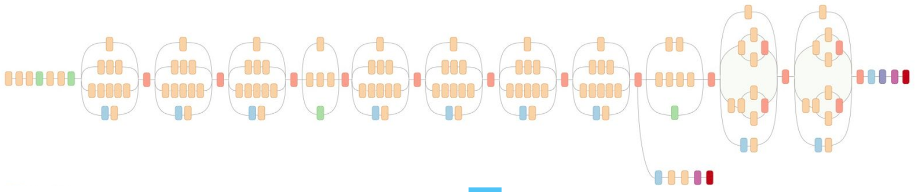
Why XLA?

TensorFlow Strengths

Flexible	Interpreted
Expressive	Dynamic Stateful
Extensible	"Black-Box" Modular

How do we keep the strengths but add more performance?

JIT Compilation via XLA



XLA program: static, decomposed TF ops

- Static data types
- Math-looking primitive ops

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

The Best of Both Worlds

TensorFlow Strengths

Flexible	Interpreted	Compiled
Expressive	Dynamic	Static
	Stateful	Pure
Extensible	"Black-Box" Modular	Primitives

Think & write this way...

But get optimization benefits of these!

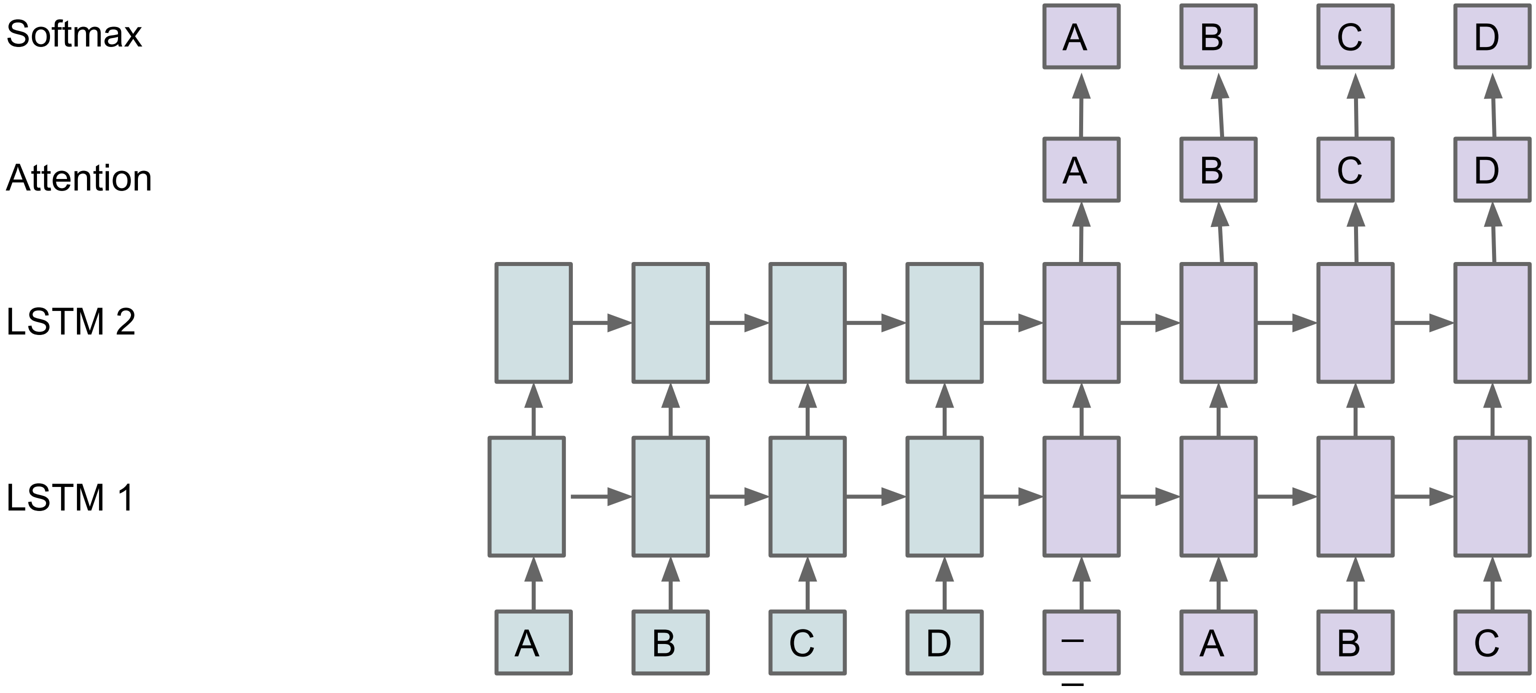
Machine Learning for Higher Performance Machine Learning Models

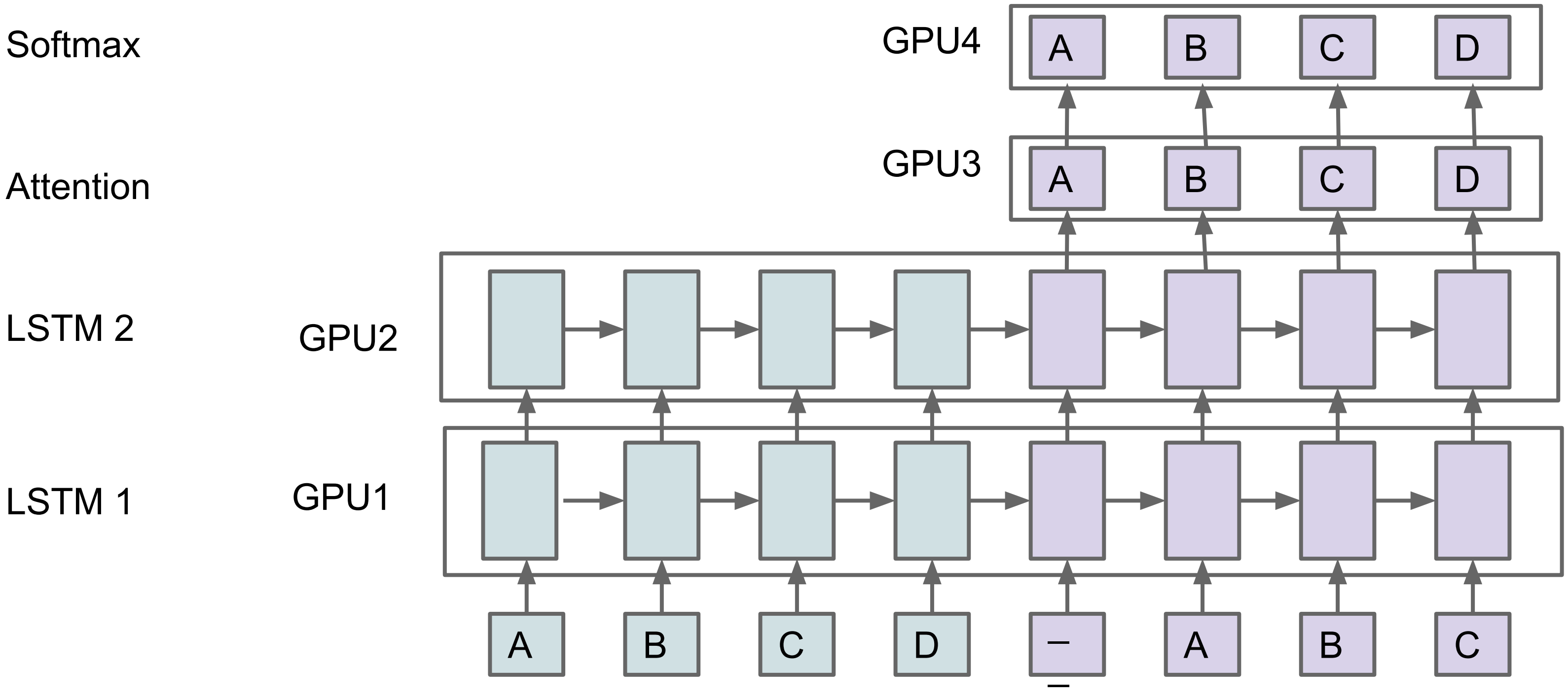
Google

For large models, model parallelism is important

For large models, model parallelism is important

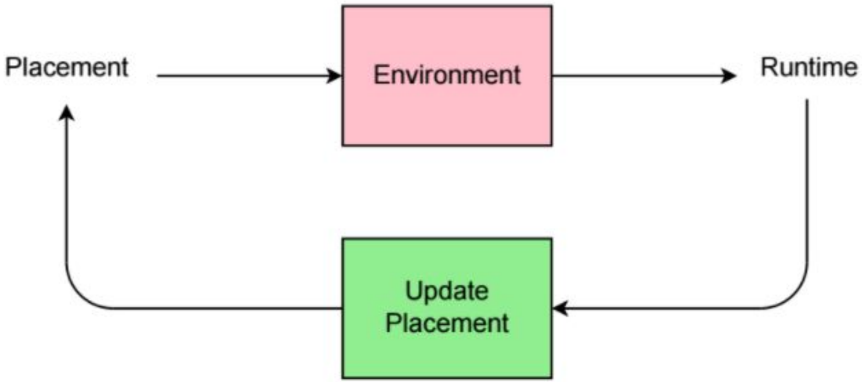
But getting good performance given multiple computing devices is non-trivial and non-obvious


Softmax
Attention
LSTM 2
LSTM 1
A
B
C
D
A
B
C
D
A
B
C
D
_
A
B
C



Softmax
Attention
LSTM 2
LSTM 1
GPU4
GPU3
GPU2
GPU1
A
B
C
D
A
B
C
D
A
B
C
D
_
A
B
C

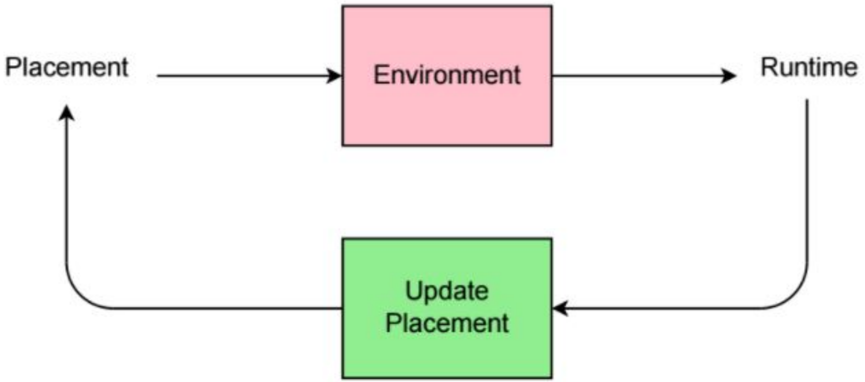
Reinforcement Learning for Higher Performance Machine Learning Models



Device Placement Optimization with Reinforcement Learning,
Azalia Mirhoseini, Hieu Pham, Quoc Le, Mohammad Norouzi, Samy Bengio, Benoit Steiner, Yuefeng Zhou, Naveen Kumar, Rasmus Larsen, and Jeff Dean, ICML 2017, arxiv.org/abs/1706.04972

Reinforcement Learning for Higher Performance Machine Learning Models

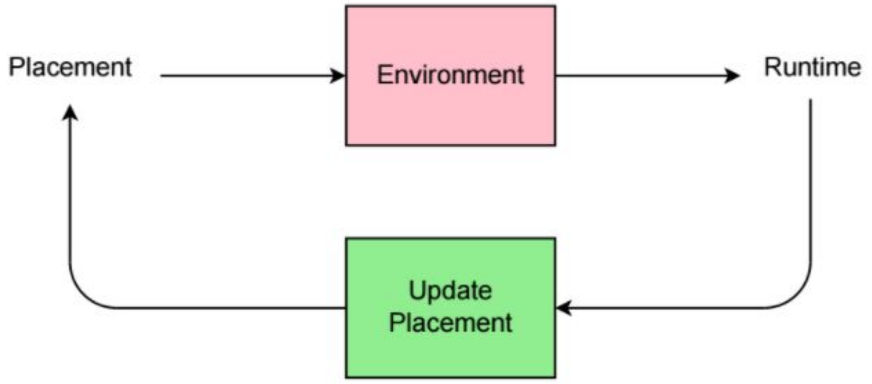
Placement model (trained via RL) gets graph as input + set of devices, outputs device placement for each graph node



Device Placement Optimization with Reinforcement Learning,
Azalia Mirhoseini, Hieu Pham, Quoc Le, Mohammad Norouzi, Samy Bengio, Benoit Steiner, Yuefeng Zhou, Naveen Kumar, Rasmus Larsen, and Jeff Dean, ICML 2017, arxiv.org/abs/1706.04972

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

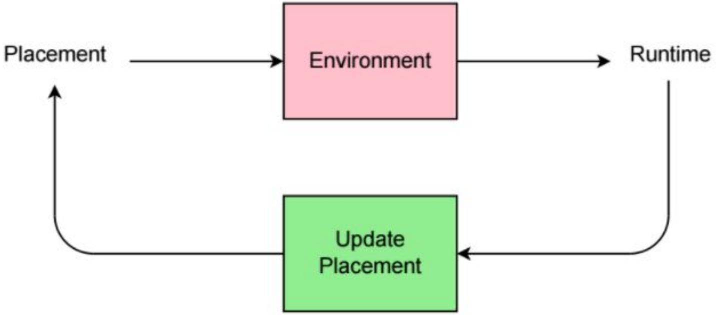


Measured time per step gives RL reward signal

Device Placement Optimization with Reinforcement Learning,
Azalia Mirhoseini, Hieu Pham, Quoc Le, Mohammad Norouzi, Samy Bengio, Benoit Steiner, Yuefeng Zhou, Naveen Kumar, Rasmus Larsen, and Jeff Dean, ICML 2017, arxiv.org/abs/1706.04972

Device Placement with Reinforcement Learning

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

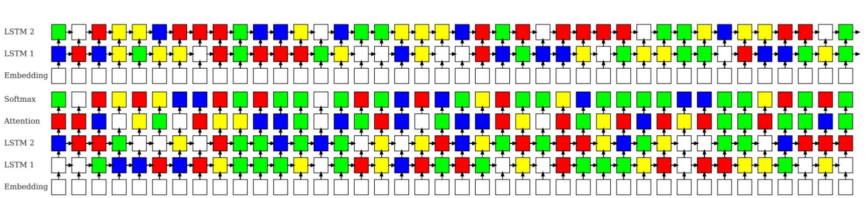


Figure 4. RL-based placement of Neural MT graph. Above: encoder, Below: decoder. Devices are denoted by colors, where the transparent color represents an operation on a CPU and each other unique color represents a different GPU. This placement achieves an improvement of 19.3% in running time compared to the fine-tuned hand-crafted placement.

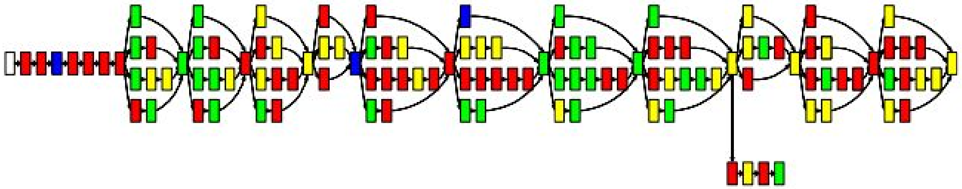


Figure 5. RL-based placement of Inception-V3. Devices are denoted by colors, where the transparent color represents an operation on a CPU and each other unique color represents a different GPU. RL-based placement achieves the improvement of 19.7% in running time compared to expert-designed placement.

+19.3% faster vs. expert human for neural translation model

+19.7% faster vs. expert human for InceptionV3 image model

Device Placement Optimization with Reinforcement Learning,
Azalia Mirhoseini, Hieu Pham, Quoc Le, Mohammad Norouzi, Samy Bengio, Benoit Steiner, Yuefeng Zhou, Naveen Kumar, Rasmus Larsen, and Jeff Dean, ICML 2017, arxiv.org/abs/1706.04972

Reducing inference cost

Google

Reducing inference cost

- **Bad feeling:** *"I have an awesomely good model that requires too much (computation, power, memory) to deploy! Oh no!"*

Fear not, there are lots of tricks:

- **Quantize**! Most models tolerate very low precision for weights (8 bits or even less).
    - 4X memory reduction, 4X computation efficiency

Distillation

- Suppose you have a giant, highly accurate model
  - (Or maybe an ensemble of many such models)
- Now you want a smaller, cheaper model with almost the same accuracy (maybe to run on a phone)

Distilling the Knowledge in a Neural Network, Hinton, Vinyals, and Dean. NIPS Deep Learning Workshop, 2014. http://arxiv.org/abs/1503.02531

.001	.04	.95		10^{-6}
Cow	Lion	Jaguar	…	Car

Softmax output

Expensive
but
accurate
model or
ensemble

The Main Idea

The ensemble implements a function from input to output. Forget the models in the ensemble and the way they are parameterized and focus on the function.

- After learning the ensemble, we have our hands on the function.
- Can we transfer the knowledge in the function into a single smaller model?

Training

0	0	**1**		0
Cow	Lion	**Jaguar**	…	Car

Hard targets

Small
model for
deployment

.001	.04	.95		10^{-6}
Cow	Lion	Jaguar	…	Car

Softmax output

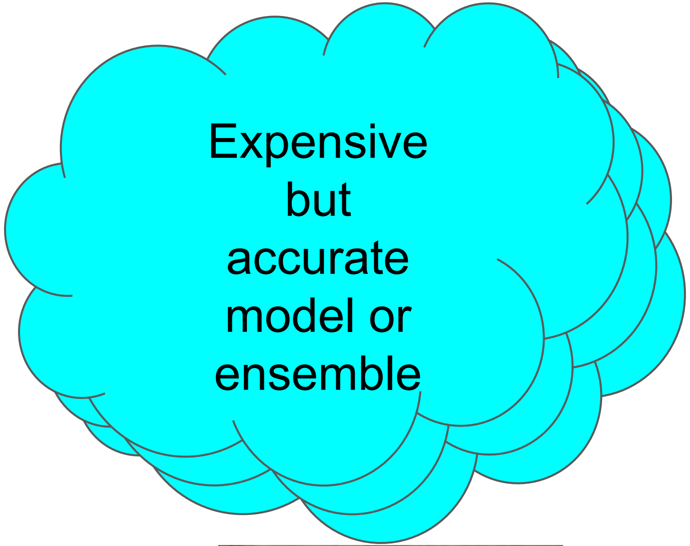
Expensive
but
accurate
model or
ensemble

.001	.04	.95		10^{-6}
Cow	Lion	Jaguar	…	Car

Softmax output

If we have the ensemble, we can divide the averaged logits from the ensemble by a "temperature" *T* to get a much softer distribution.

$$p_i = \frac{\exp\left(\frac{z_i}{T}\right)}{\sum_j \exp\left(\frac{z_j}{T}\right)}$$



.001	.04	.95	10^{-6}
Cow	Lion	Jaguar	… Car

Softmax output

.1	.2	.6	0
Cow	Lion	Jaguar	… Car

Softened softmax output

Expensive but accurate model or ensemble

If we have the ensemble, we can divide the averaged logits from the ensemble by a "temperature" *T* to get a much softer distribution.

$$p_i = \frac{\exp\left(\frac{z_i}{T}\right)}{\sum_j \exp\left(\frac{z_j}{T}\right)}$$

.001	.04	.95		10^{-6}
Cow	Lion	Jaguar	…	Car

Softmax output

.1	.2	.6		0
Cow	Lion	Jaguar	…	Car

Softened softmax output

If we have the ensemble, we can divide the averaged logits from the ensemble by a "temperature" *T* to get a much softer distribution.

$$p_i = \frac{\exp\left(\frac{z_i}{T}\right)}{\sum_j \exp\left(\frac{z_j}{T}\right)}$$

Expensive
but
accurate
model or
ensemble

This full distribution conveys lots of information about the function implemented by the large ensemble!

Distillation

Softened softmax output

.1	.2	.6		0
Cow	Lion	Jaguar	…	Car

Hard targets

0	0	**1**		0
Cow	Lion	**Jaguar**	…	Car

Training objective tries to match both of these

Expensive but accurate model or ensemble

Small model for deployment

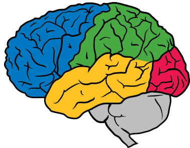
Some Results on Speech

Start with a model that classifies 58.9% of frames correctly.

Use that model to provide soft targets for smaller model (that also sees hard targets)

- The new model gets 57.0% correct even when trained on only 3% of the data
- With just hard targets, it only gets to 44.5% correct and then gets much worse.

Soft targets are a VERY good regularizer!
Also trains much faster (soft targets enrich gradients)



A few trends in the kinds of models we want to train

Google

Bigger models, but sparsely activated

Google

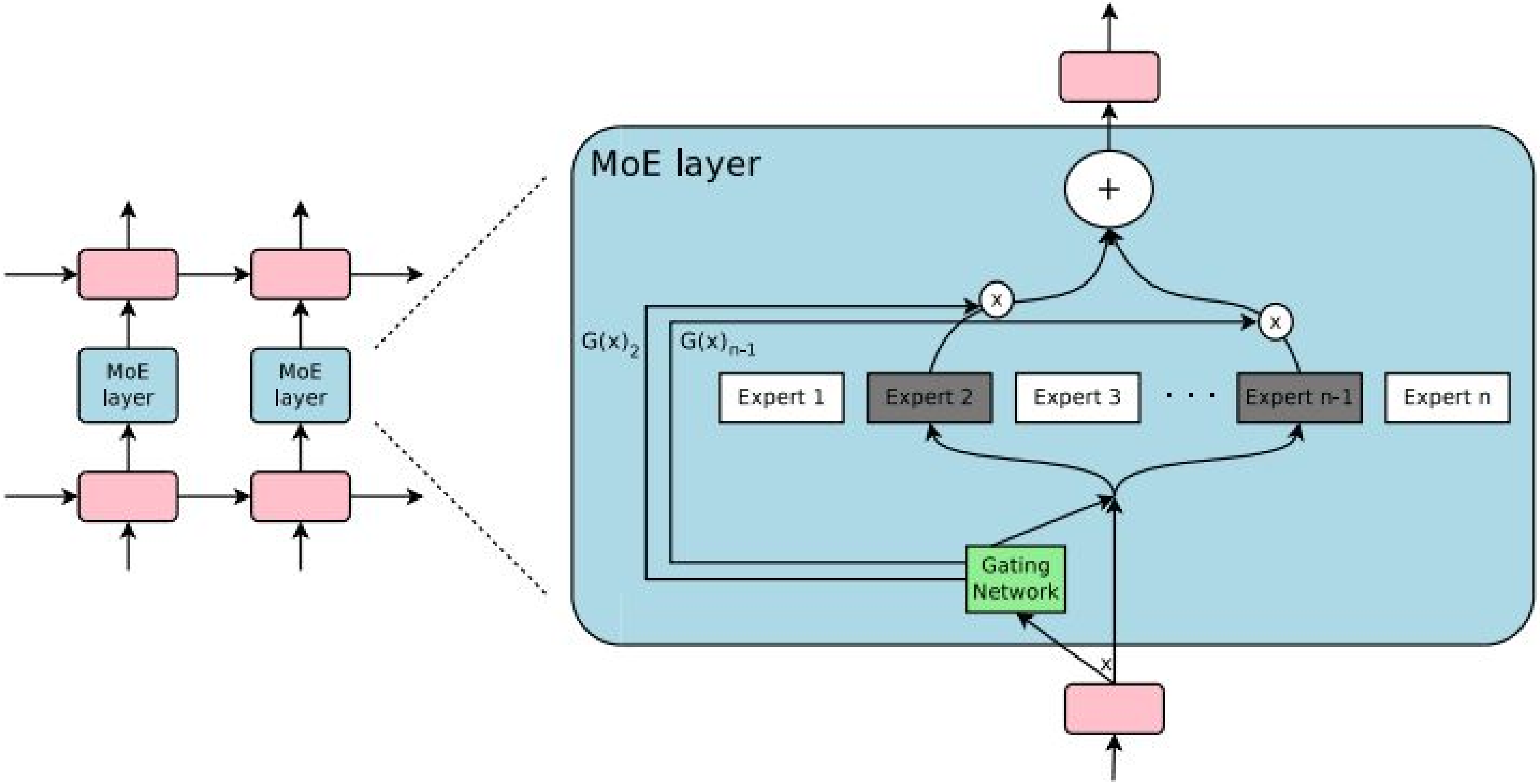
Bigger models, but sparsely activated

Motivation:

Want **huge model capacity** for large datasets, but want individual example to **only activate tiny fraction** of large model

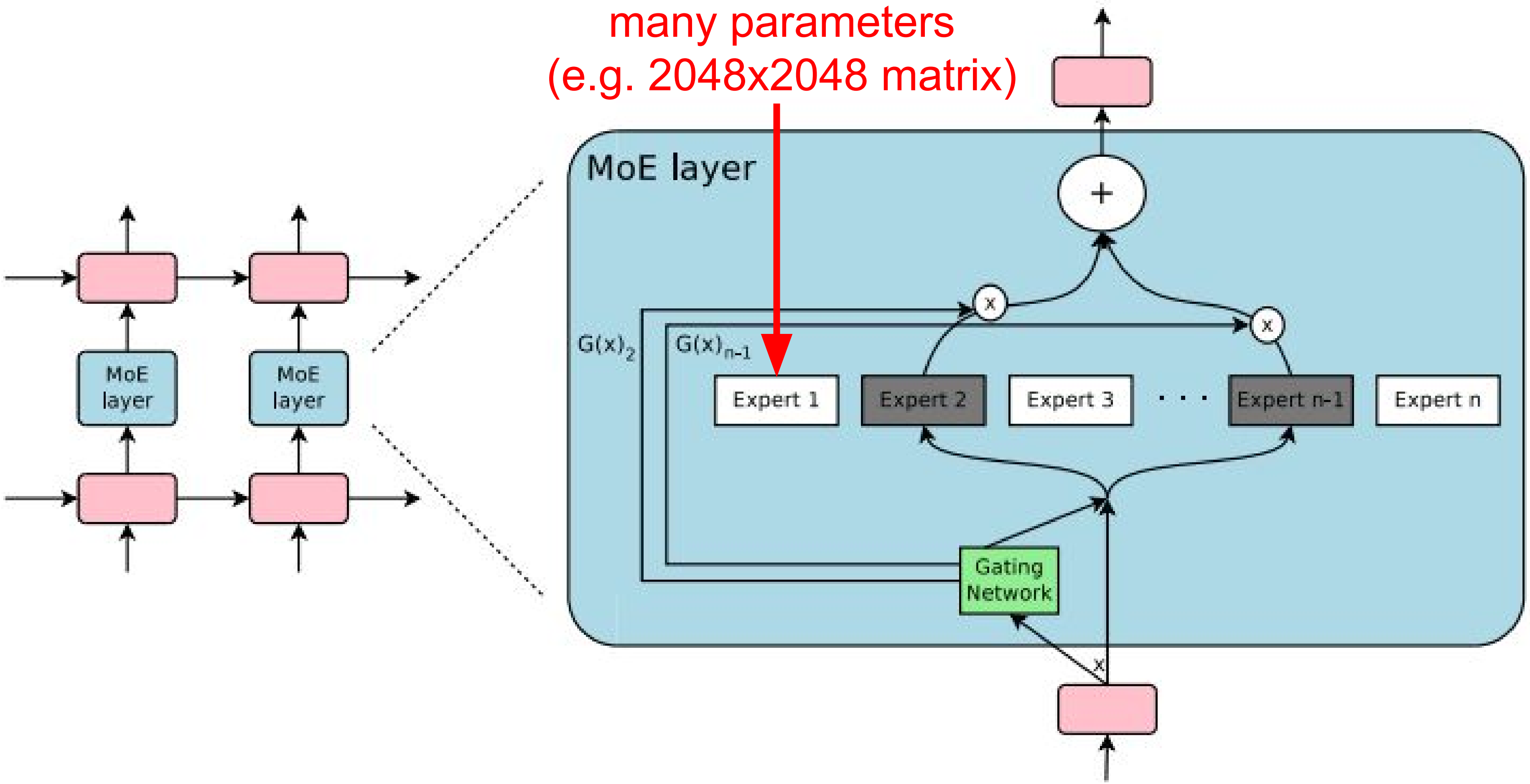
Google

Per-Example Routing

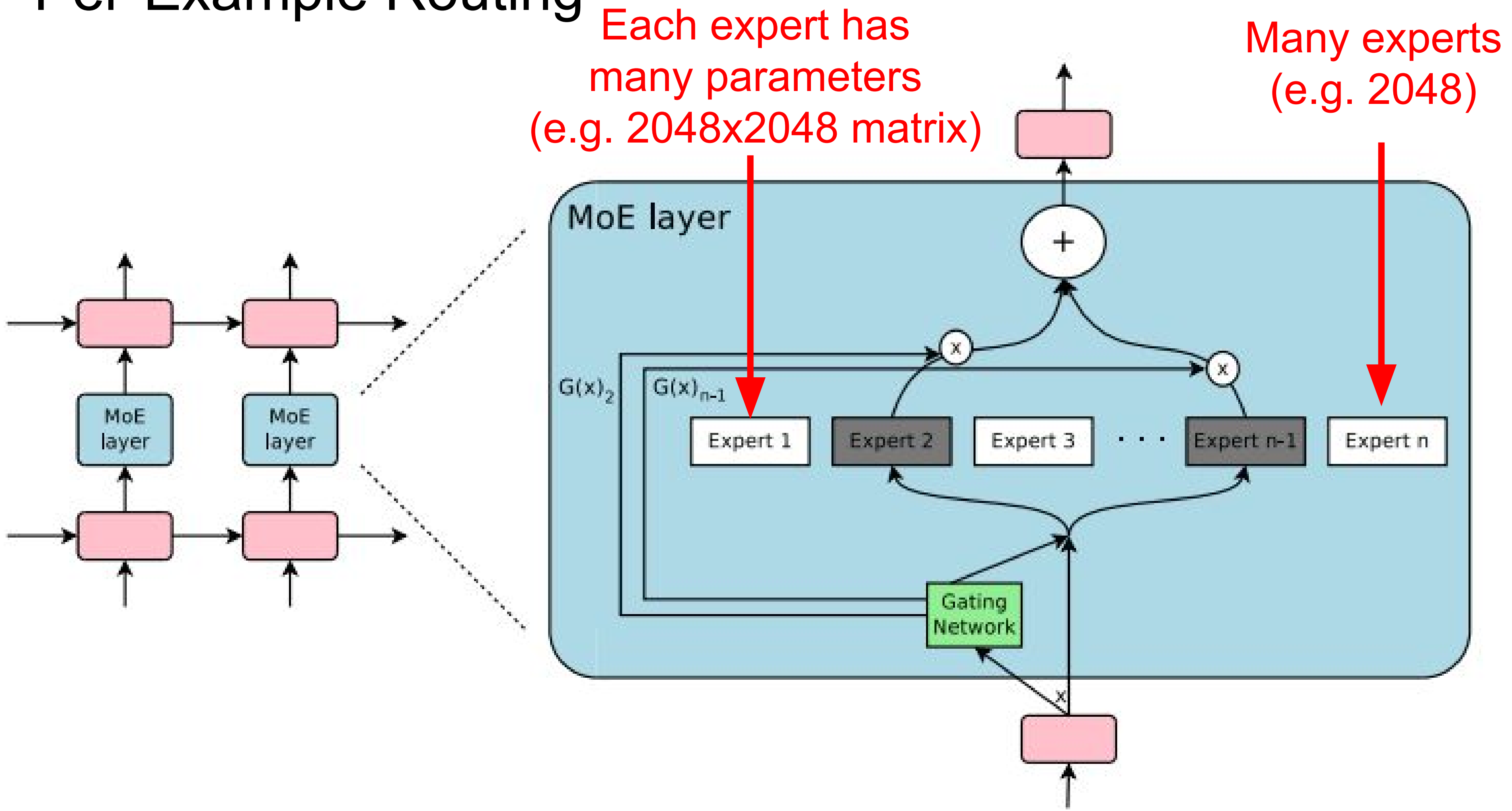


Per-Example Routing

Each expert has
many parameters
(e.g. 2048x2048 matrix)



Per-Example Routing



Per-Example Routing

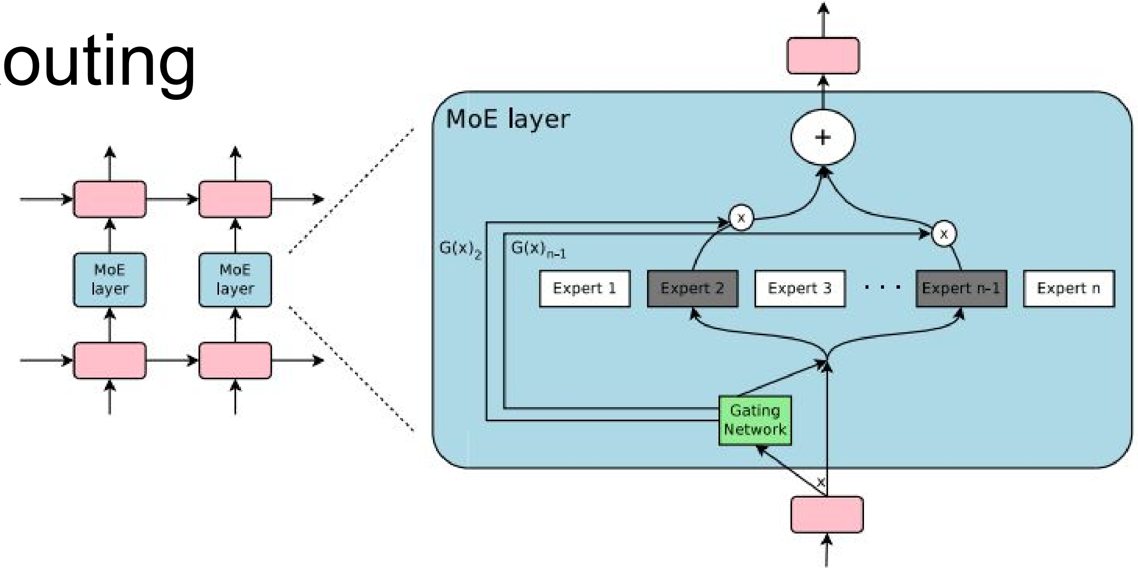


Table 7: Perplexity and BLEU comparison of our method against previous state-of-art methods on the Google Production En→ Fr dataset.

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

Outrageously Large Neural Networks: The Sparsely-gated Mixture-of-Experts Layer, Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le & Jeff Dean
Appeared in ICLR 2017, https://openreview.net/pdf?id=B1ckMDqlg

Automated machine learning ("learning to learn")

Current:

Solution = ML expertise + data + computation

Current:

Solution = ML expertise + data + computation

Can we turn this into:

Solution = data + 100X computation

???

Early encouraging signs

(1) Reinforcement learning-based architecture search
(2) Learn how to optimize

NEURAL ARCHITECTURE SEARCH WITH REINFORCEMENT LEARNING

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

Appeared in ICLR 2017

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

arxiv.org/abs/1611.01578

CIFAR-10 Image Recognition Task

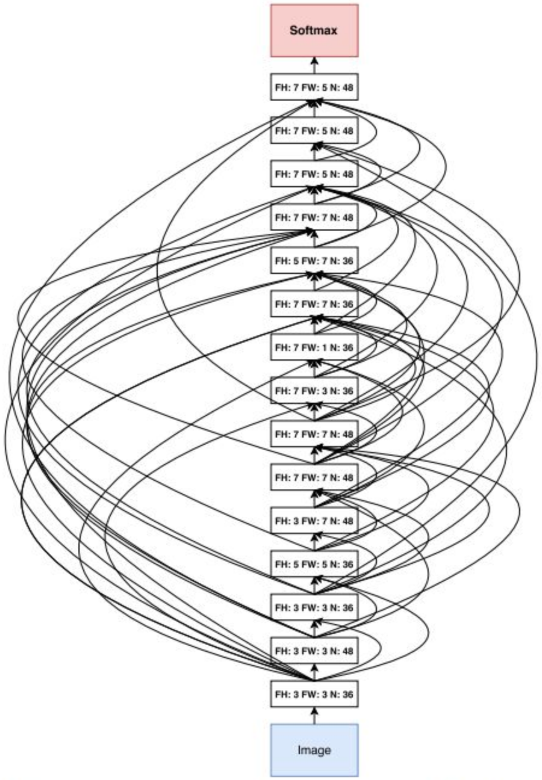


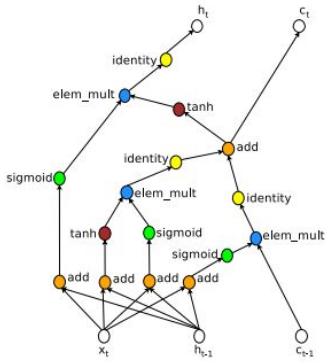
Figure 7: Convolutional architecture discovered by our method, when the search space does not have strides or pooling layers. FH is filter height, FW is filter width and N is number of filters.

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. (2016b))	110	1.7M	6.41
ResNet with Stochastic Depth (Huang et al., 2016b)	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
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	32.0M	3.84

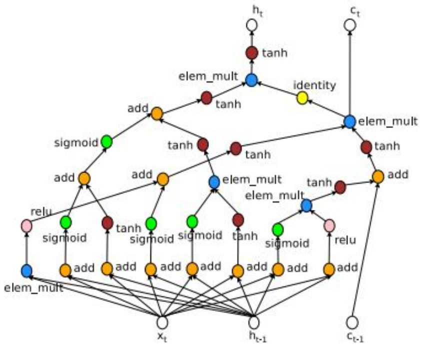
Table 1: Performance of Neural Architecture Search and other state-of-the-art models on CIFAR-10.

Penn Tree Bank Language Modeling Task

"Normal" LSTM cell



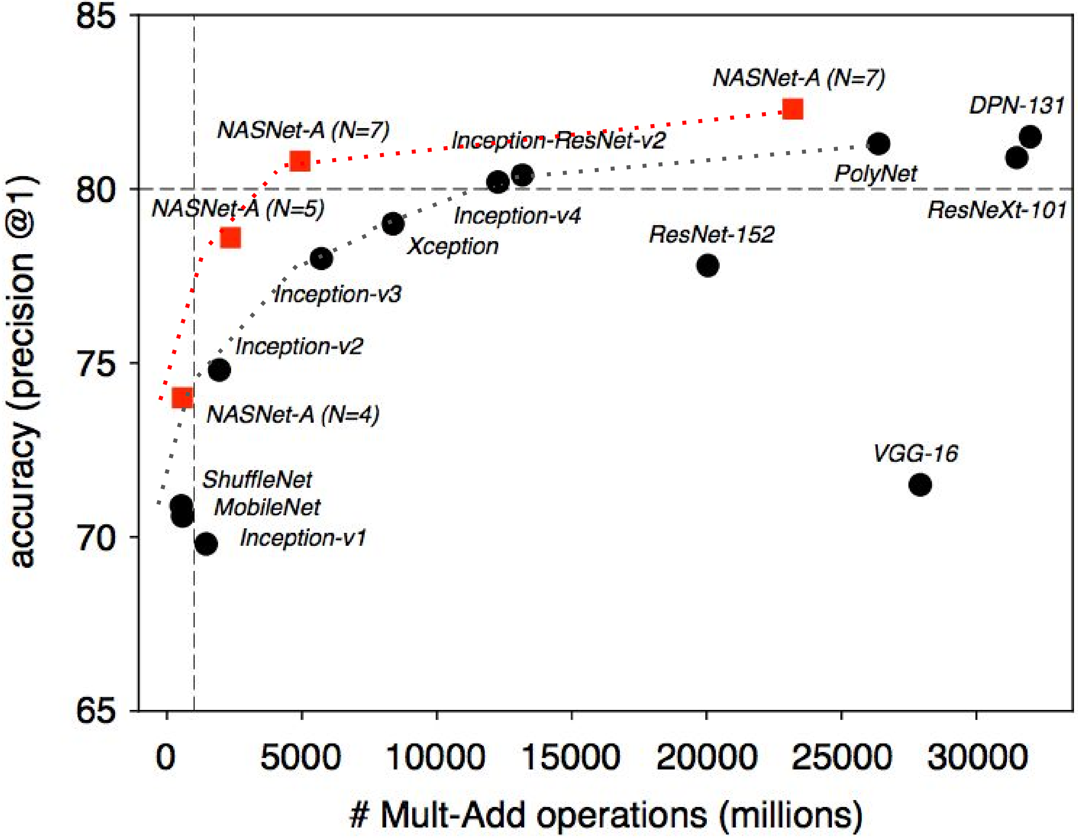
Cell discovered by architecture search



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
Zaremba et al. (2014) - LSTM (large)	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	24M	73.2
Merity et al. (2016) - Zoneout + Variational LSTM (medium)	20M	80.6
Merity et al. (2016) - Pointer Sentinel-LSTM (medium)	21M	70.9
Zilly et al. (2016) - Variational RHN, shared embeddings	24M	$\boxed{66.0}$
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	$\boxed{62.4}$

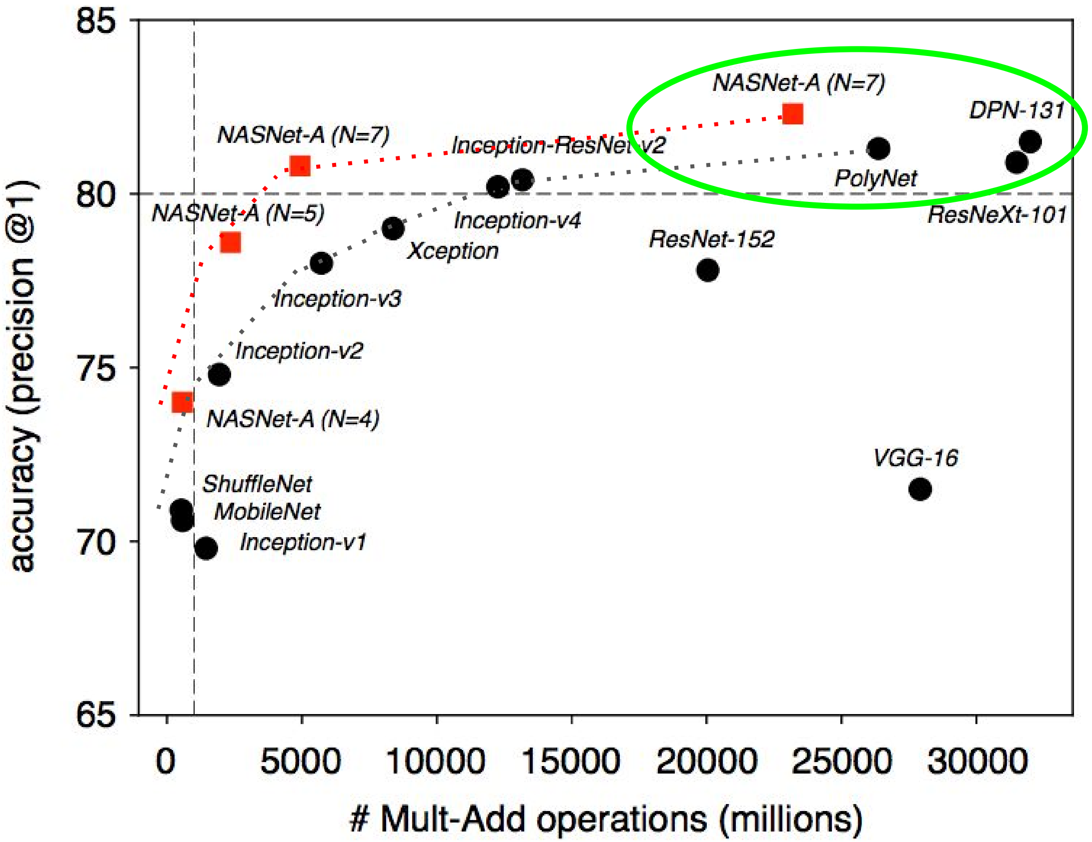
Table 2: Single model perplexity on the test set of the Penn Treebank language modeling task. Parameter numbers with ‡ are estimates with reference to Merity et al. (2016).

Scaling to Imagenet



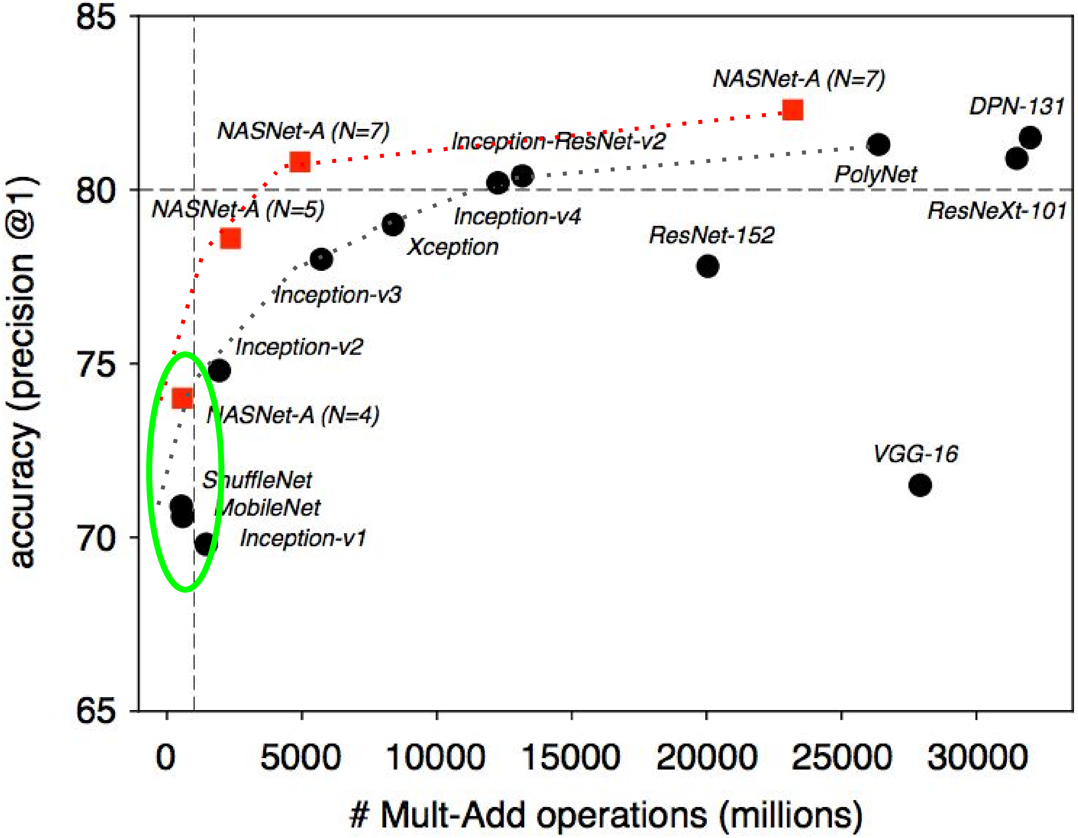
Learning Transferable Architectures for Scalable Image Recognition, Barret Zoph, Vijay Vasudevan, Jonathon Shlens and Quoc Le, https://arxiv.org/abs/1707.07012

Scaling to Imagenet



Learning Transferable Architectures for Scalable Image Recognition, Barret Zoph, Vijay Vasudevan, Jonathon Shlens and Quoc Le, https://arxiv.org/abs/1707.07012

Scaling to Imagenet



Learning Transferable Architectures for Scalable Image Recognition, Barret Zoph, Vijay Vasudevan, Jonathon Shlens and Quoc Le, https://arxiv.org/abs/1707.07012

Learn the Optimization Update Rule

Commonly Used Human-Designed Optimizers

parameters += *learning_rate* * ***expression***

SGD: g

Momentum: $g + \gamma\hat{m}$

ADAM: $\hat{m}/\sqrt{\hat{v}}$

RMSProp: $g/\sqrt{\hat{v}}$

Where:

g gradient

$\hat{m}$ bias-corrected running average of the gradient

$\hat{v}$ bias-corrected running average of the squared gradient

Learn the Optimization Update Rule

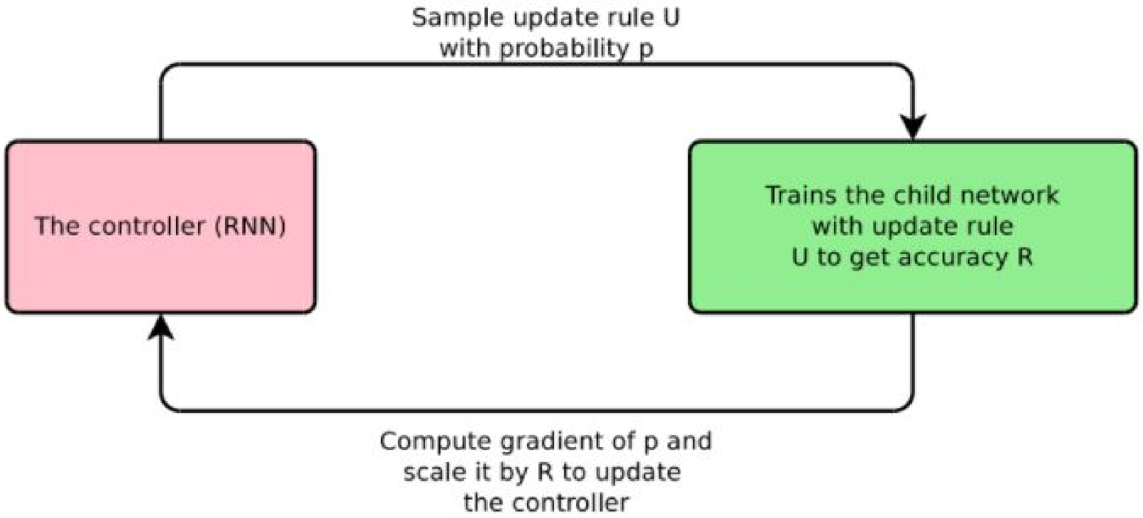


Figure 1. Overview of Neural Optimizer Search.

Neural Optimizer Search using Reinforcement Learning, Irwan Bello, Barret Zoph, Vijay Vasudevan, and Quoc Le, ICML 2017, proceedings.mlr.press/v70/bello17a/bello17a.pdf

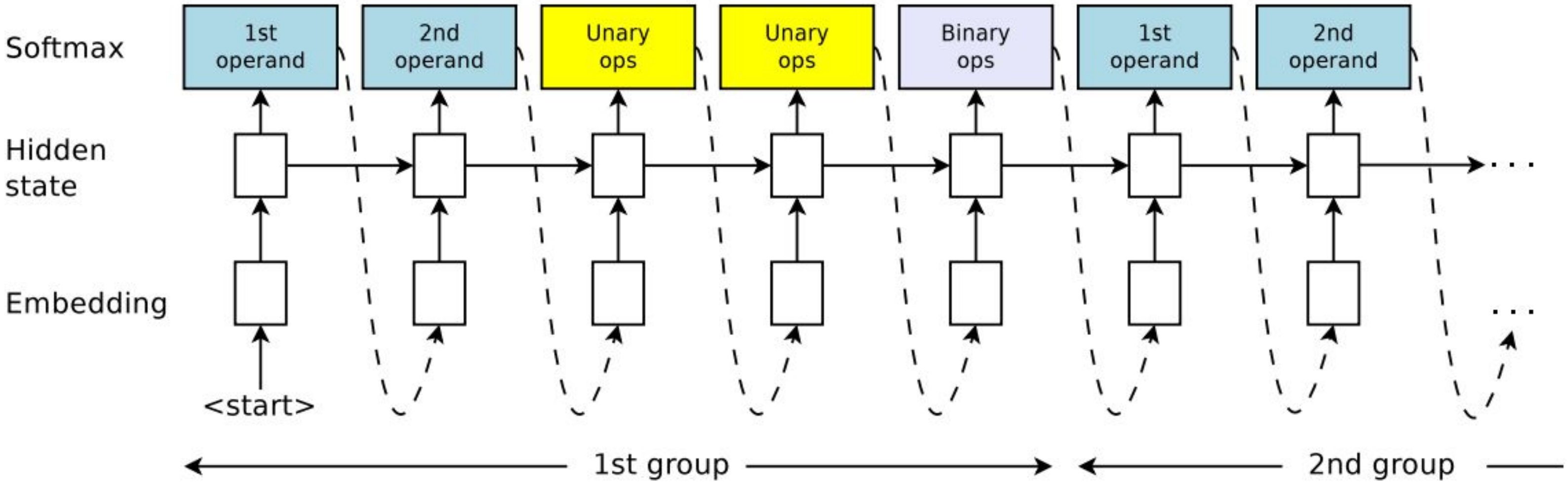


Figure 3. Overview of the controller RNN. The controller iteratively selects subsequences of length 5. It first selects the 1st and 2nd operands op_1 and op_2, then 2 unary functions u_1 and u_2 to apply to the operands and finally a binary function b that combines the outputs of the unary functions. The resulting $b(u_1(op_1), u_2(op_2))$ then becomes an operand that can be selected in the subsequent group of predictions or becomes the update rule. Every prediction is carried out by a softmax classifier and then fed into the next time step as input.

The operands, unary functions and binary functions that are accessible to our controller are as follows:

- Operands: g, g^2, g^3, $\hat{m}$, $\hat{v}$, $\hat{\gamma}$, $sign(g)$, $sign(\hat{m})$, $sign(g) * sign(m)$, 1, small constant noise, $10^{-4}w$, $10^{-3}w$, $10^{-2}w$, $10^{-1}w$, ADAM and RMSProp.

- Unary functions which map input x to: x, e^x, $\log|x|$, $clip(x, 10^{-5})$, $clip(x, 10^{-4})$, $clip(x, 10^{-3})$, $drop(x, 0.1)$, $drop(x, 0.3)$ or $drop(x, 0.5)$.

- Binary functions which map (x, y) to $x + y$ (addition), $x - y$ (subtraction), $x * y$ (multiplication), $\frac{x}{y+\epsilon}$ (division) or x (keep left).

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

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.

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	$\boxed{92.3}$
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	$\boxed{93.2}$

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.

Optimizer	Train perplexity	Test BLEU
Adam	1.49	24.5
$g * e^{\mathrm{sign}(g)*\mathrm{sign}(m)}$	1.39	25.0

Table 2. Performance of our optimizer versus ADAM in a state-of-the-art GNMT model on WMT 2014 English $\rightarrow$ German.

What might a plausible future look like?

Combine many of these ideas:

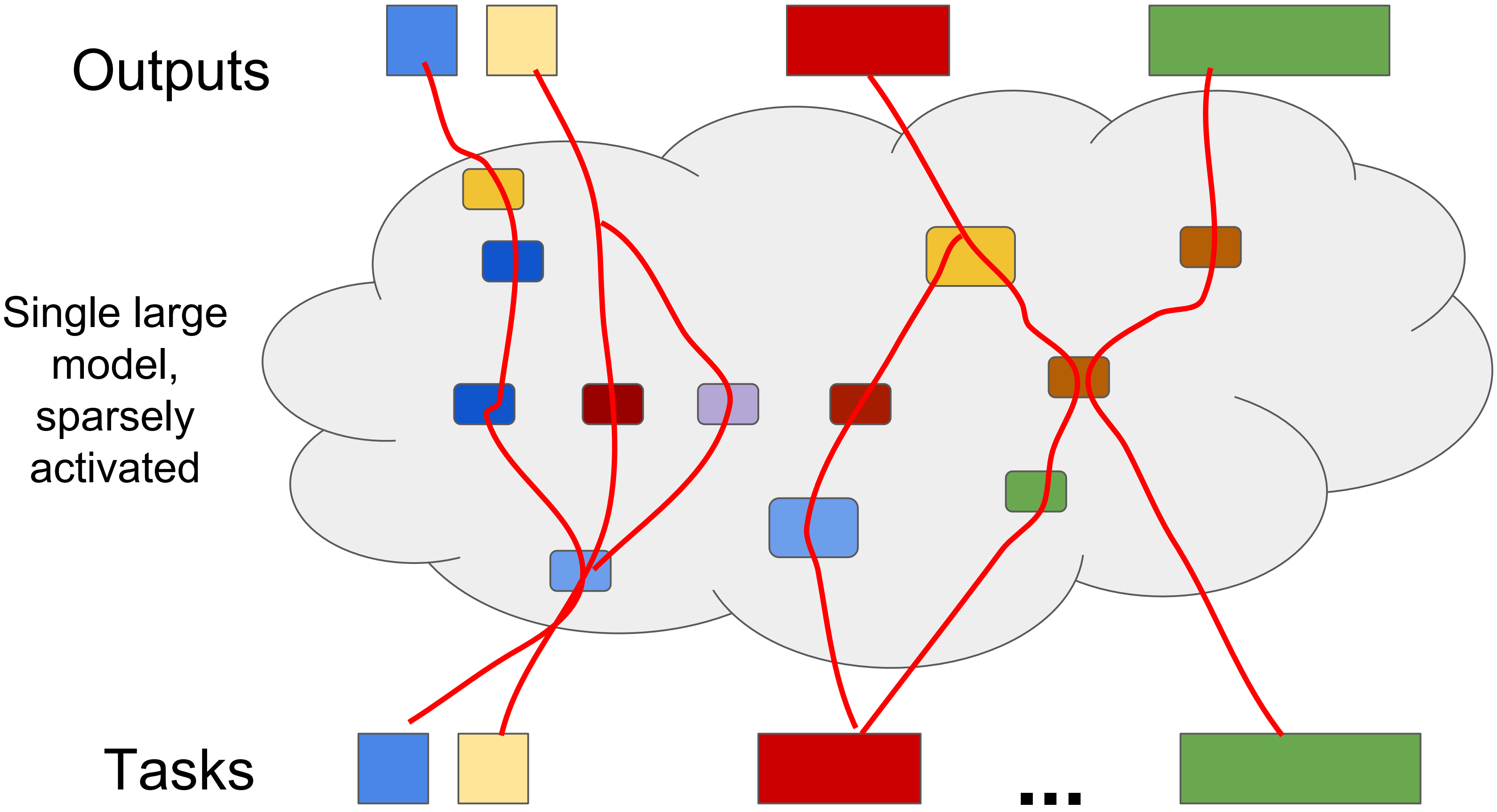
Large model, but **sparsely activated**

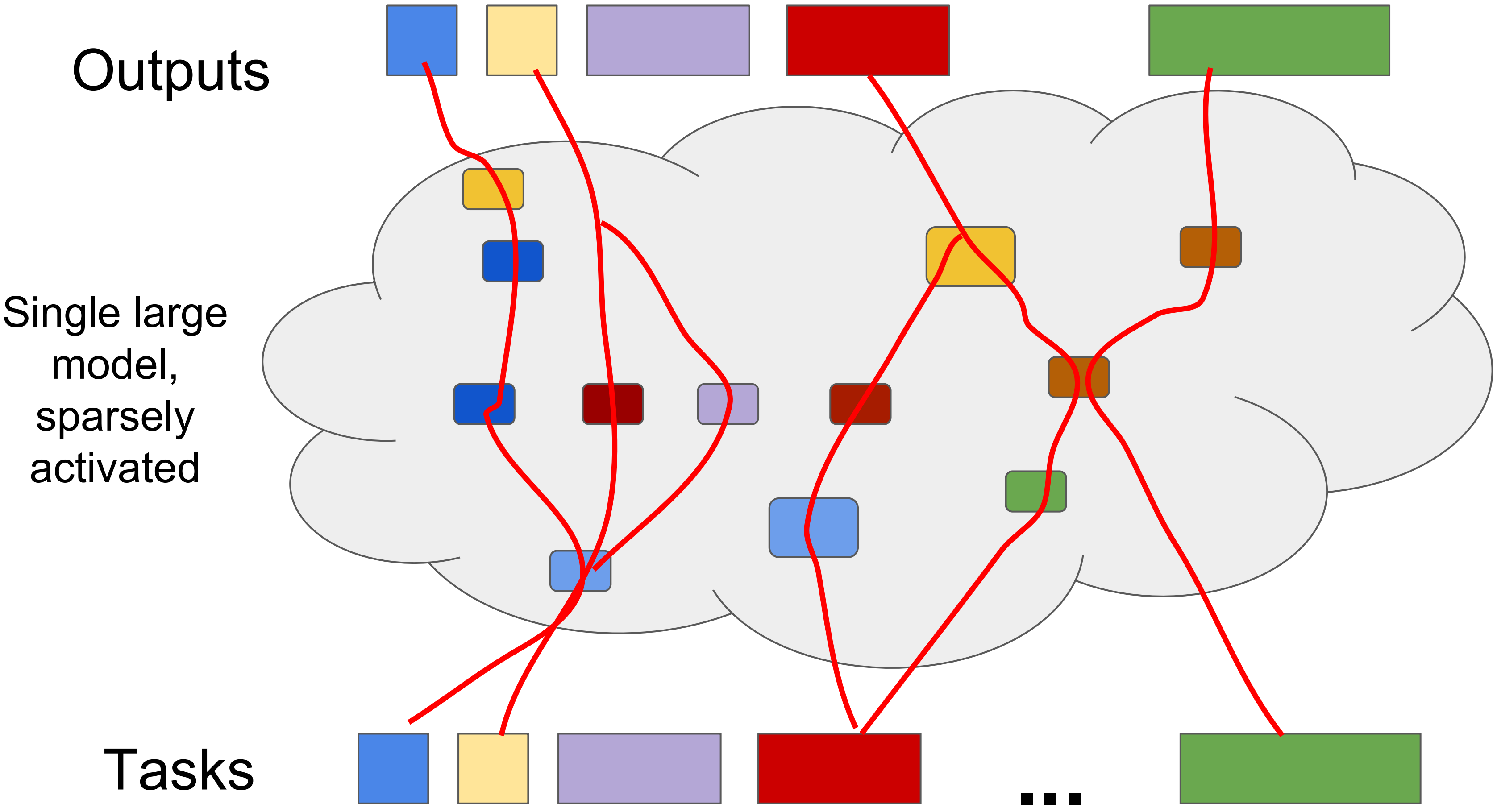
Single model to **solve many tasks** (100s to 1Ms)

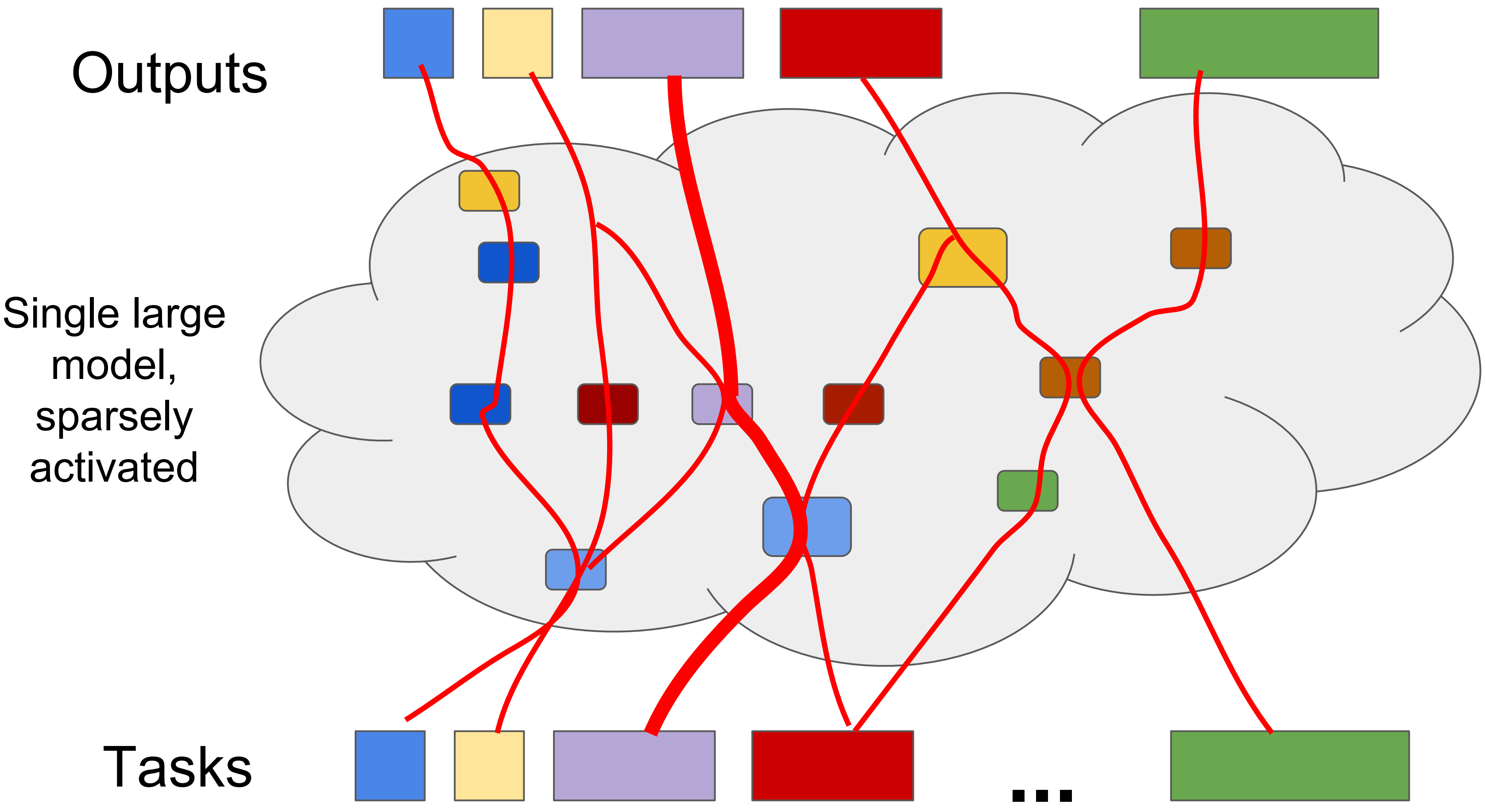
Dynamically learn and **grow pathways** through large model

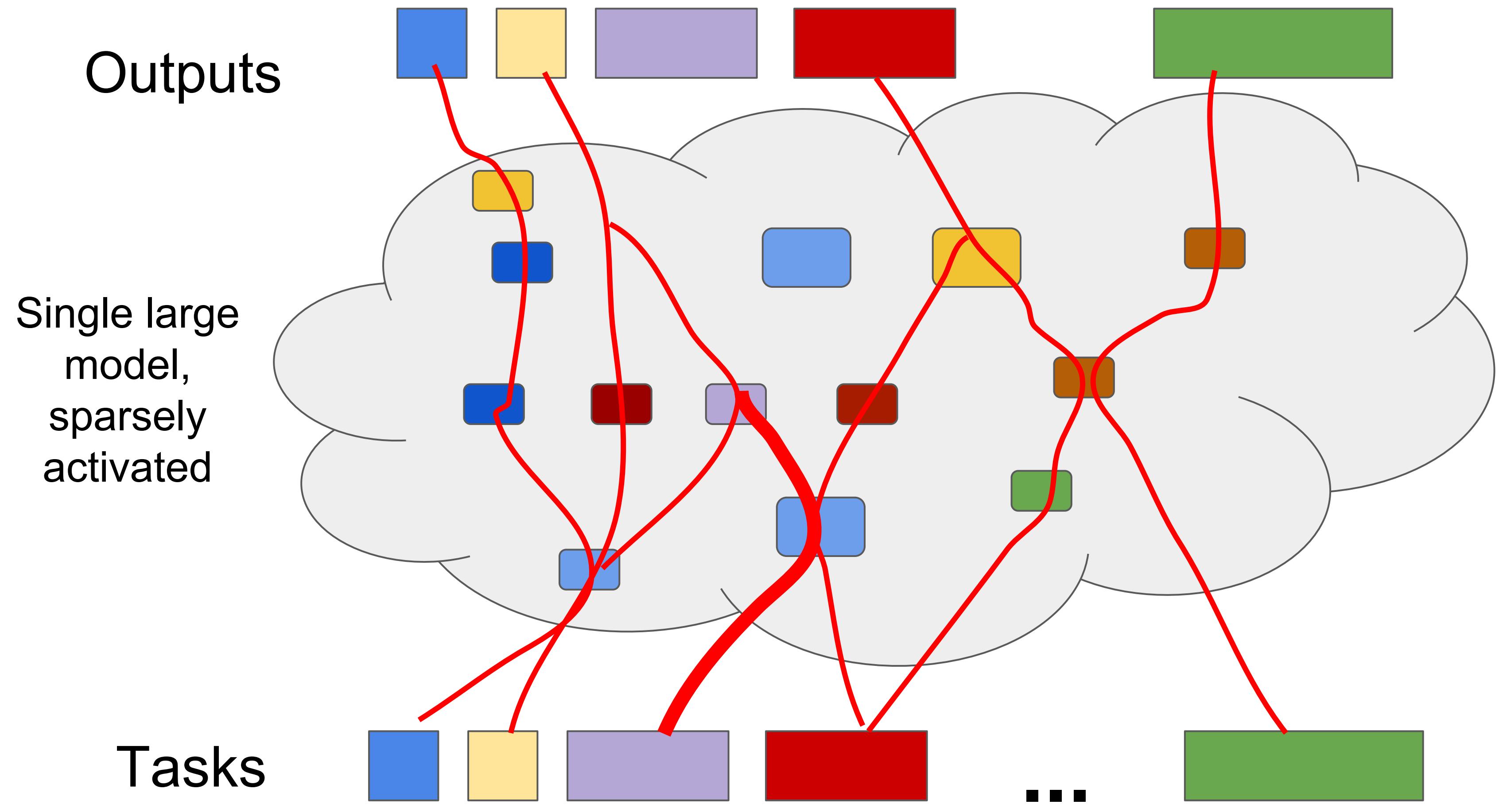
Hardware **specialized for ML supercomputing**

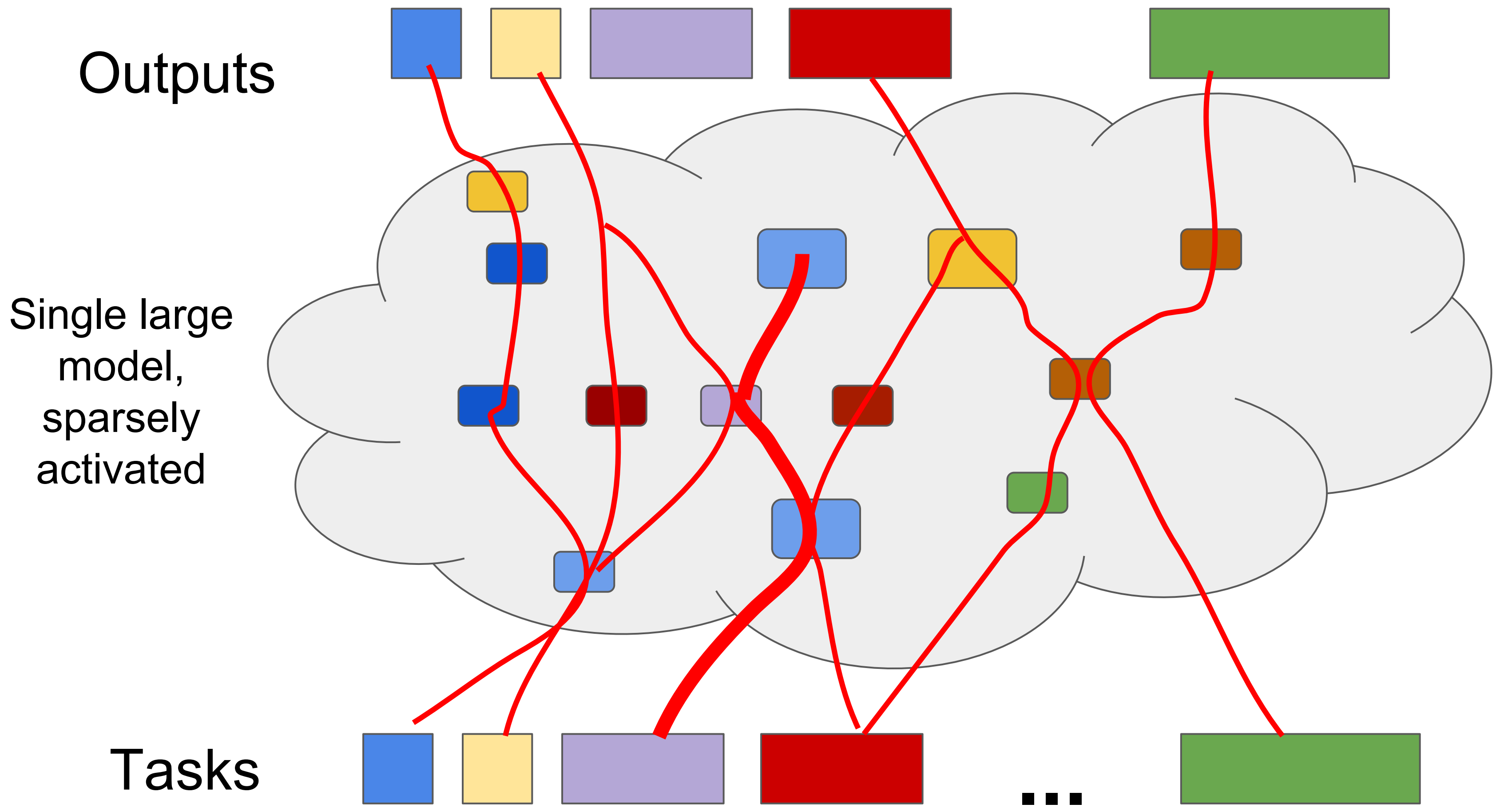
ML for efficient mapping onto this hardware

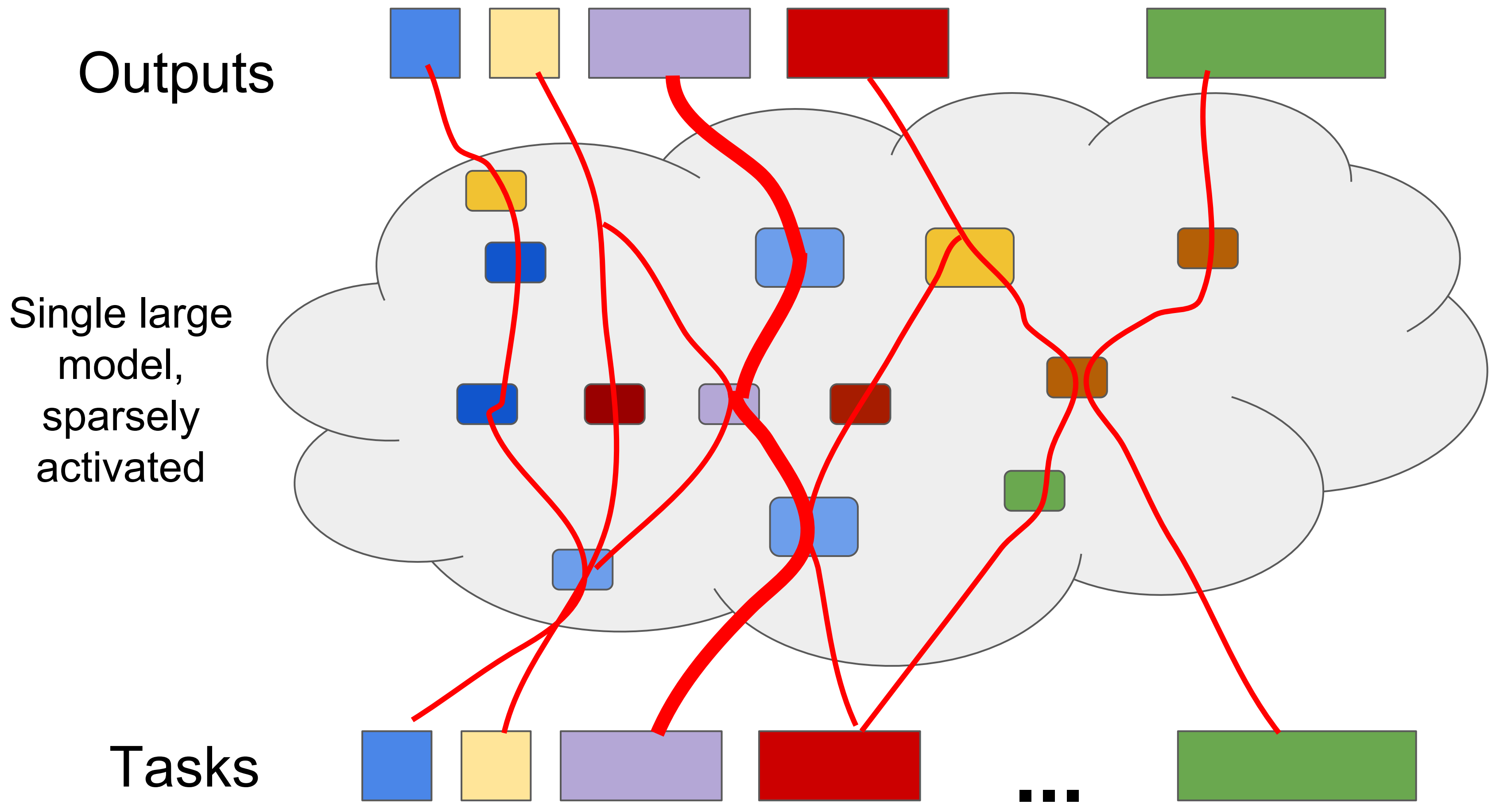

Outputs
Single large model, sparsely activated
Tasks
...


Outputs
Single large model, sparsely activated
Tasks
...


Outputs
Single large model, sparsely activated
Tasks
...


Outputs
Single large model, sparsely activated
Tasks
...


Outputs
Single large model, sparsely activated
Tasks
...

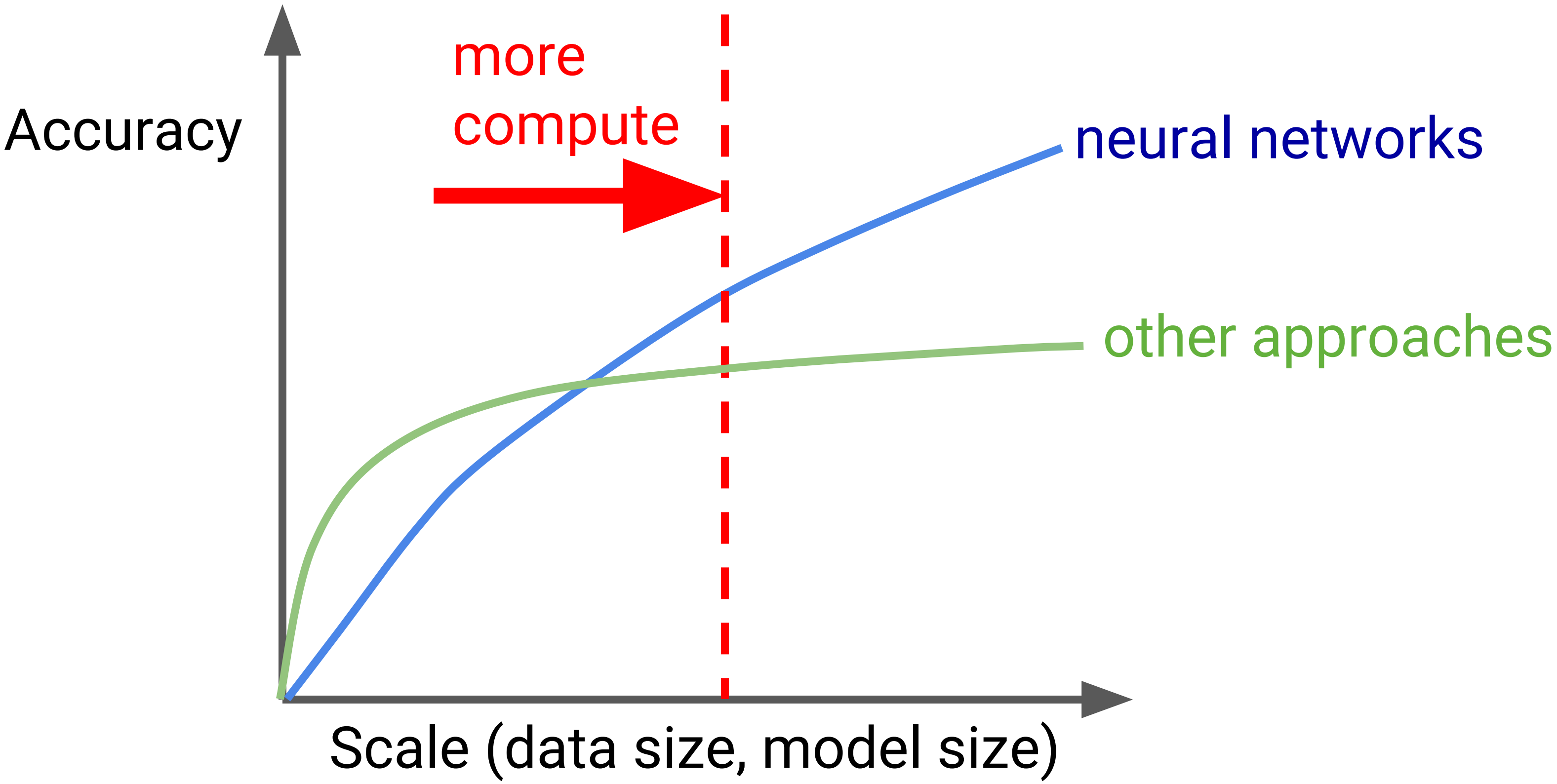

Outputs
Single large model, sparsely activated
Tasks
...

Questions/open-problems at the intersection of machine learning and systems/computer architecture

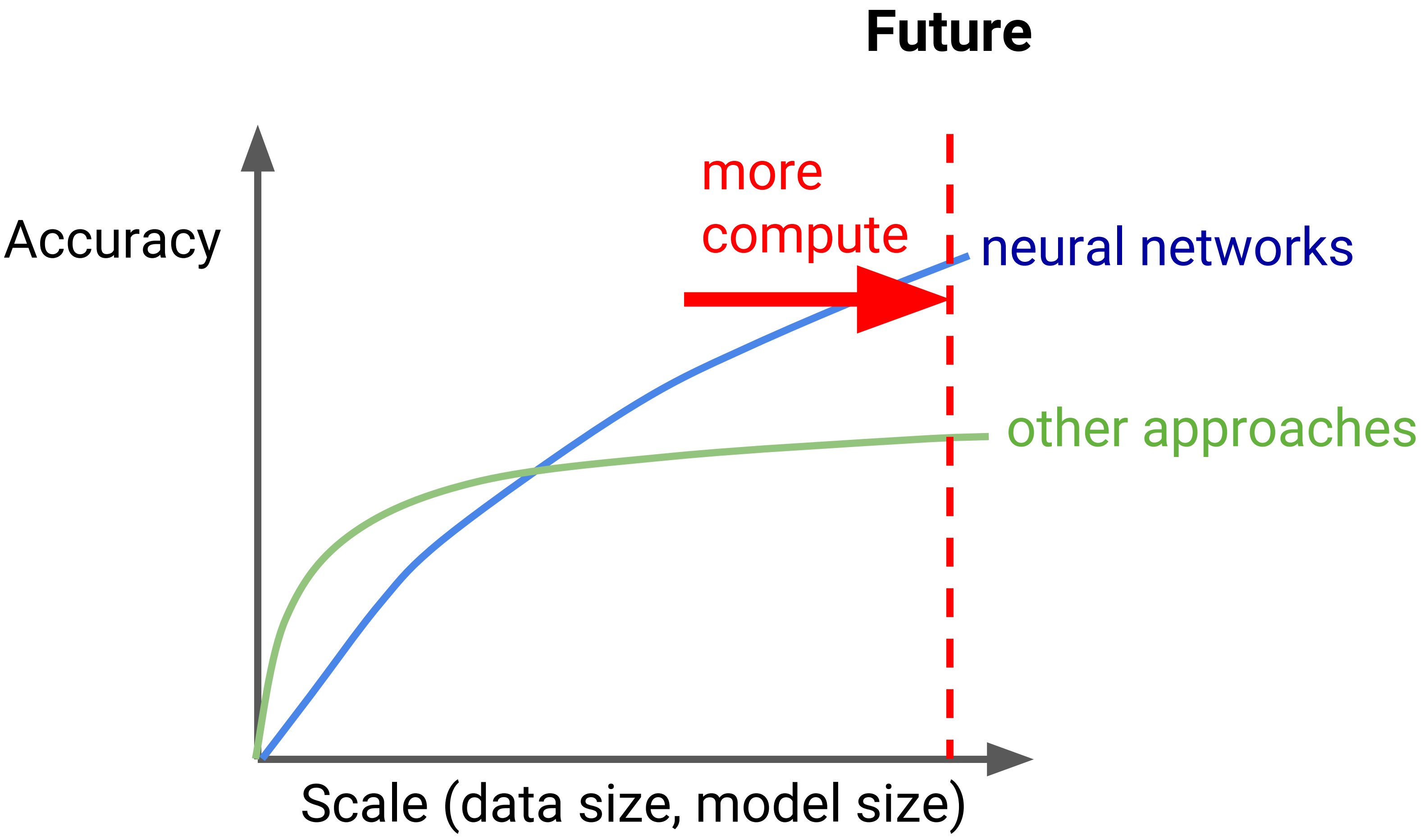
Questions/Open Issues

- Do **dramatically different numerics** make sense (e.g. 1- or 2-bit activations/parameters?)
- How can we **deal efficiently with very dynamic models** (different graph for every input example), especially on very large scale machines?
- What new approaches can help us with the **problem of diminishing returns from larger batch sizes**?
    - If we could train with batch_size = 1M, that would make things much easier
- What **ML algorithms/approaches will be important in 3-4 years**?

Now



more
compute
Accuracy
neural networks
other approaches
Scale (data size, model size)


Future
more
compute
Accuracy
neural networks
other approaches
Scale (data size, model size)

Conclusions

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

They are also **dramatically reshaping our computational devices**

If you're not considering how to use deep neural nets to solve your problems, **you almost certainly should be**

g.co/brain

More info about our work

Main Research Areas

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

MORE PAPERS

BLOG POSTS