



May 16, 2018

AI and compute



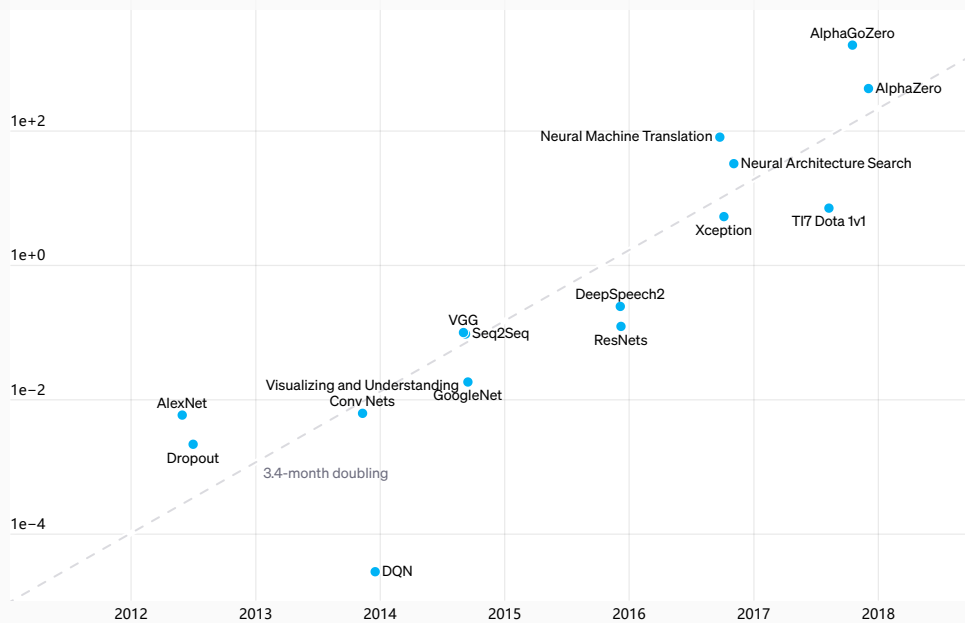
We're releasing an analysis showing that since 2012, the amount of compute used in the largest AI training runs has been increasing exponentially with a 3.4-month doubling time (by comparison, Moore's Law had a 2-year doubling period). Since 2012, this metric has grown by more than 300,000x (a 2-year doubling period would yield only a 7x increase). Improvements in compute have been a key component of AI progress, so as long as this trend continues, it's worth preparing for the implications of systems far outside today's capabilities.

AlexNet to AlphaGo Zero: 300,000x increase in compute

Log scale Linear Scale

▶ Read aloud





The total amount of compute, in petaflop/s-days,^[^footnote-petaflops] used to train selected results that are relatively well known, used a lot of compute for their time, and gave enough information to estimate the compute used.

[Download charts ↗](#)

Overview

Three factors drive the advance of AI: algorithmic innovation, data (which can be either supervised data or interactive environments), and the amount of compute available for training. Algorithmic innovation and data are difficult to track, but compute is unusually quantifiable, providing an opportunity to measure one input to AI progress. Of course, the use of massive compute sometimes just exposes the shortcomings of our current algorithms. But at least within many current domains, more compute seems to lead predictably to better performance, and is often complementary to algorithmic advances.

For this analysis, we believe the relevant number is not the speed of a single GPU, nor the capacity of the biggest datacenter, but the amount of compute that is used to train a single model—this is the number most likely to correlate to how powerful our best models are. Compute per model differs greatly from total bulk compute because limits on parallelism (both hardware and algorithmic) have constrained how big a model can be or how much it can be usefully trained. Of course, important breakthroughs are still made with modest amounts of compute—this analysis just covers compute capability.

The trend represents an increase by roughly a factor of 10 each year. It's been partly driven by custom hardware that allows more operations to be performed per second for a given price (GPUs and TPUs), but it's been primarily propelled by researchers repeatedly finding ways to use more chips in parallel and being willing to pay the economic cost of doing so.

Eras

Looking at the graph we can roughly see four distinct eras:

- Before 2012: It was uncommon to use GPUs for ML, making any of the results in the graph difficult to achieve.
- 2012 to 2014: Infrastructure to train on many GPUs was uncommon, so most results used 1-8 GPUs rated at 1-2 TFLOPS for a total of 0.001-0.1 pfs-days.



2014 to 2016: Large-scale results used 10-100 GPUs rated at 5-10 TFLOPS, resulting in 0.1-10 pfs-days. Diminishing returns on data parallelism meant that larger training runs had limited value.



- 2016 to 2017: Approaches that allow greater algorithmic parallelism such as huge batch sizes, architecture search, and expert iteration, along with specialized hardware such as TPU's and faster interconnects, have greatly increased these limits, at least for some applications.

AlphaGoZero/AlphaZero is the most visible public example of massive algorithmic parallelism, but many other applications at this scale are now algorithmically possible, and may already be happening in a production context.

Looking forward

We see multiple reasons to believe that the trend in the graph could continue. Many hardware startups are developing AI-specific chips, some of which claim they will achieve a substantial increase in FLOPS/Watt (which is correlated to FLOPS/\$) over the next 1-2 years. There may also be gains from simply reconfiguring hardware to do the same number of operations for less economic cost. On the parallelism side, many of the recent algorithmic innovations described above could in principle be combined multiplicatively—for example, architecture search and massively parallel SGD.

On the other hand, cost will eventually limit the parallelism side of the trend and physics will limit the chip efficiency side. We believe the largest training runs today employ hardware that cost in the single digit millions of dollars to purchase (although the amortized cost is much lower). But the majority of neural net compute today is still spent on inference (deployment), not training, meaning companies can repurpose or afford to purchase much larger fleets of chips for training. Therefore, if sufficient economic incentive exists, we could see even more massively parallel training runs, and thus the continuation of this trend for several more years. The world's total hardware budget is 1 trillion dollars a year, so absolute limits remain far away. Overall, given the data above, the precedent for exponential trends in computing, work on ML specific hardware, and the economic incentives at play, we think it'd be a mistake to be confident this trend won't continue in the short term.

Past trends are not sufficient to predict how long the trend will continue into the future, or what will happen while it continues. But even the reasonable potential for rapid increases in capabilities means it is critical to start addressing both safety and malicious use of AI today. Foresight is essential to responsible policymaking and responsible technological development, and we must get out ahead of these trends rather than belatedly reacting to them.

If you'd like to help make sure that AI progress benefits all of humanity, join us at OpenAI. Our research and engineering roles range from machine learning researchers to policy researchers to infrastructure engineers.

Appendix: methods

Two methodologies were used to generate these data points. When we had enough information, we directly counted the number of FLOPs per training example and multiplied by the total number of forward and backward passes during training. When we didn't have enough training time and total number of GPUs used and assumed a utilization efficiency (usually 0.33). For the majority of the papers we used the first, and in a minority we relied on the second, and we computed both whenever possible as a consistency check. In the majority of cases we also intended to be precise but we aim to be correct within a factor 2-3. We provide some example calculations below.

Example of Method 1: Counting operations in the model

This method is particularly easy to use when the authors give the number of operations used in a forward pass, as in the Resnet paper

```
(add-multiplies per forward pass) * (2 FLOPs/add-multiply) * (3 for forward and backward pass) * (number of images)
= (11.4 * 10^9) * 2 * 3 * (1.2 * 10^6 images) * 128
= 10,000 PF = 0.117 pfs-days
```

Operations can also be counted programmatically for a known model architecture in some deep learning frameworks, or we can simulate. If we have enough information to make this calculation, it will be quite accurate, but in some cases papers don't contain all the necessary information

Example of Method 2: GPU Time

Q If we can't count operations directly, we can instead look at how many GPUs were trained for how long, and use reasonable guesses of operations performed. We emphasize that here we are not counting peak theoretical FLOPS, but using an assumed fraction of theoretical operations. We assume a 33% utilization for GPUs and a 17% utilization for CPU's, based on our own experience, except where we have more specific data. This is done at OpenAI).

As an example, in the AlexNet paper it's stated that "our network takes between five and six days to train on two GTX 580 3GB GPU

```
Number of GPUs * (peta-flops/GTX580) * days trained * estimated utilization
= 2 * (1.58 * 10 ^ -3 PF) * 5.5 * 0.33
= 500 PF = 0.0058 pfs-days
```

This method is more approximate and can easily be off by a factor of 2 or occasionally more; our aim is only to estimate the order of magnitude. They often line up quite well (for AlexNet we can also directly count the operations, which gives us 0.0054 pfs-days vs 0.0058 with this method).

```
1.2M images * 90 epochs * 0.75 GFLOPS * (2 add-multiply) * (3 backward pass)
= 470 PF = 0.0054 pfs-days
```

Selected additional calculations

Dropout

```
1 GPU * 4 days * 1.54 TFLOPS/GTX 580 * 0.33 utilization
= 184 PF = 0.0021 pfs-days
```

Method 2

Visualizing and Understanding Conv Nets

```
1 GPU * 12 days * 1.54 TFLOPS/GTX 580 * 0.33 utilization
= 532 PF = 0.0062 pfs-days
```

Method 2

DQN

```
Network is 84x84x3 input, 16, 8x8, stride 4, 32 4x4 stride 2, 256 fully connected
First layer: 20*20*3*16*8*8 = 1.23M add-multiplies
Second layer: 9*9*16*32*4*4 = 0.66M add-multiplies
Third layer: 9*9*32*256 = 0.66M add-multiplies
Total ~ 2.55M add-multiplies
2.5 MFLOPS * 5M updates * 32 batch size * 2 multiply-add * 3 backward pass
= 2.3 PF = 2.7e-5 pfs-days
```

Method 1

Seq2Seq

```
(348M + 304M) words * 0.380 GF * 2 add-multiply * 3 backprop * 7.5 epoch
= 7,300 PF = 0.085 pfs-days
```

Method 1

```
10 days * 8 GPU's * 3.5 TFLOPS/ K20 GPU * 0.33 utilization
= 8,100 PF = 0.093 pfs-days
```

Method 2

VGG

```
1.2 M images * 74 epochs * 16 GFLOPS * 2 add-multiply * 3 backward pass
= 8524 PF = 0.098 pfs-days
```

Method 1

```
4 Titan Black GPU's * 15 days * 5.1 TFLOPS/GPU * 0.33 utilization
= 10,000 PF = 0.12 pfs-days
```

Method 2

DeepSpeech2

```
1 timestep = (1280 hidden units)^2 * (7 RNN layers * 4 matrices for bidirectional + 2 DNN layers) * (2 for forward + 2 for backward)
20 epochs * 12,000 hours * 3600 seconds/hour * 50 samples/sec * 98 MFLOPS * 3 add-multiply * 2 backprop
= 26,000 PF = 0.30 pfs-days
```

Method 1

```
16 TitanX GPU's * 5 days * 6 TFLOPS/GPU * 0.50 utilization
= 21,000 PF = 0.25 pfs-days
```

Method 2

Xception

```
60 K80 GPU's * 30 days * 8.5 TFLOPS/GPU * 0.33 utilization
= 4.5e5 PF = 5.0 pfs-days
```

Neural Architecture Search

50 epochs * 50,000 images * 10.0 GFLOPSs * 12800 networks * 2 add-multiply * 3 backward pass
= 1.9e6 PF = 22 pfs-days

Method 1

800 K40's * 28 days * 4.2 TFLOPS/GPU * 0.33 utilization
= 2.8e6 PF = 31 pfs-days

Method 2. Details given in a later paper.

Neural Machine Translation

$\sqrt{10 * 100}$ factor added because production model used 2-3 orders of magnitude more data, but only 1 ep
96 K80 GPU's * 9 days * 8.5 TFLOPS * 0.33 utilization * $\sqrt{10 * 100}$
= 6.9e6 PF = 79 pfs-days

Method 2

Appendix: Recent novel results that used modest amounts of compute

Massive compute is certainly not a requirement to produce important results. Many recent noteworthy results have used only mode using modest compute that gave enough information to estimate their compute. We didn't use multiple methods to estimate the con conservative estimates around any missing information, so they have more overall uncertainty. They aren't material to our quantitat sharing:

Attention is all you need: 0.089 pfs-days (6/2017)

Adam Optimizer: less than 0.0007 pfs-days (12/2014)

Learning to Align and Translate: 0.018 pfs-days (9/2014)

GANs: less than 0.006 pfs-days (6/2014)


Word2Vec: less than 0.00045 pfs-days (10/2013)

Variational Auto Encoders: less than 0.0000055 pfs-days (12/2013)

Addendum: Compute used in older headline results

We've updated our [analysis](#) with data that span 1959 to 2012. Looking at the data as a whole, we clearly see two distinct eras of training AI systems in terms of compute-usage: (a) a first era, from 1959 to 2012, which is defined by results that roughly track Moore's law, and (b) the modern era, from 2012 to now, of results using computational power that substantially outpaces macro trends. The history of investment in AI broadly is usually told as a story of booms and busts, but we don't see that reflected in the historical trend of compute used by learning systems. It seems that AI winters and periods of excitement had a small effect on compute used to train models^B over the last half-century.

Two distinct eras of compute usage in training AI systems

Show error bars All 



Footnotes

A A petaflop/s-day (pfs-day) consists of performing 10¹⁵ neural net operations per second for one day, or a total of about 10²⁰ operations. The compute-time product serves as a mental convenience, similar to kW-hr for energy. We don't measure peak theoretical FLOPS of the hardware but instead try to estimate the number of actual operations performed. We count adds and multiplies as separate operations, we count any add or multiply as a single operation regardless of numerical precision (making "FLOP" a slight misnomer), and we ignore ensemble models. Example calculations that went into this graph are provided in this [appendix](#). Doubling time for line of best fit shown is 3.4 months. ↩



- B Just as in the original analysis, we focus on the costs to train models. This doesn't include AI systems like expert systems, which attracted substantial investment in the first era. ↩
- C For one vivid account of the history of computing in AI in this period, see the "False Start" section in Hans Moravec's [article](#). ↩
- D We've already advocated for additional funding for academia in our [testimony in Congress](#) this year, and for the creation of dedicated compute clusters to help academia and industry collaboratively benchmark and assess the safety of AI systems in response to a [request for information from NIST](#). ↩

Original post

[Dario Amodei](#), [Danny Hernandez](#)

Addendum

[Girish Sastry](#), [Jack Clark](#), [Greg Brockman](#), [Ilya Sutskever](#)

Acknowledgments

The authors thank Katja Grace, Geoffrey Irving, Jack Clark, Thomas Anthony, and Michael Page for assistance with this post.

Related articles

[View all Research](#) >



Improved Techniques for Training Consistency Models

Consistency Models

Our research

Overview

Index



DALL·E 3

Sora

ChatGPT

For Everyone

For Teams

For Enterprises

ChatGPT login ↗

Download

API

Platform overview

Pricing

Documentation ↗

API login ↗

Explore more

OpenAI for business

Stories

Safety overview

Safety overview

Safety standards

Teams

Safety Systems

Preparedness

Superalignment

Company

About us

News

Our Charter

Security

Residency

Careers

Terms & policies

Terms of use

Privacy policy

Brand guidelines

