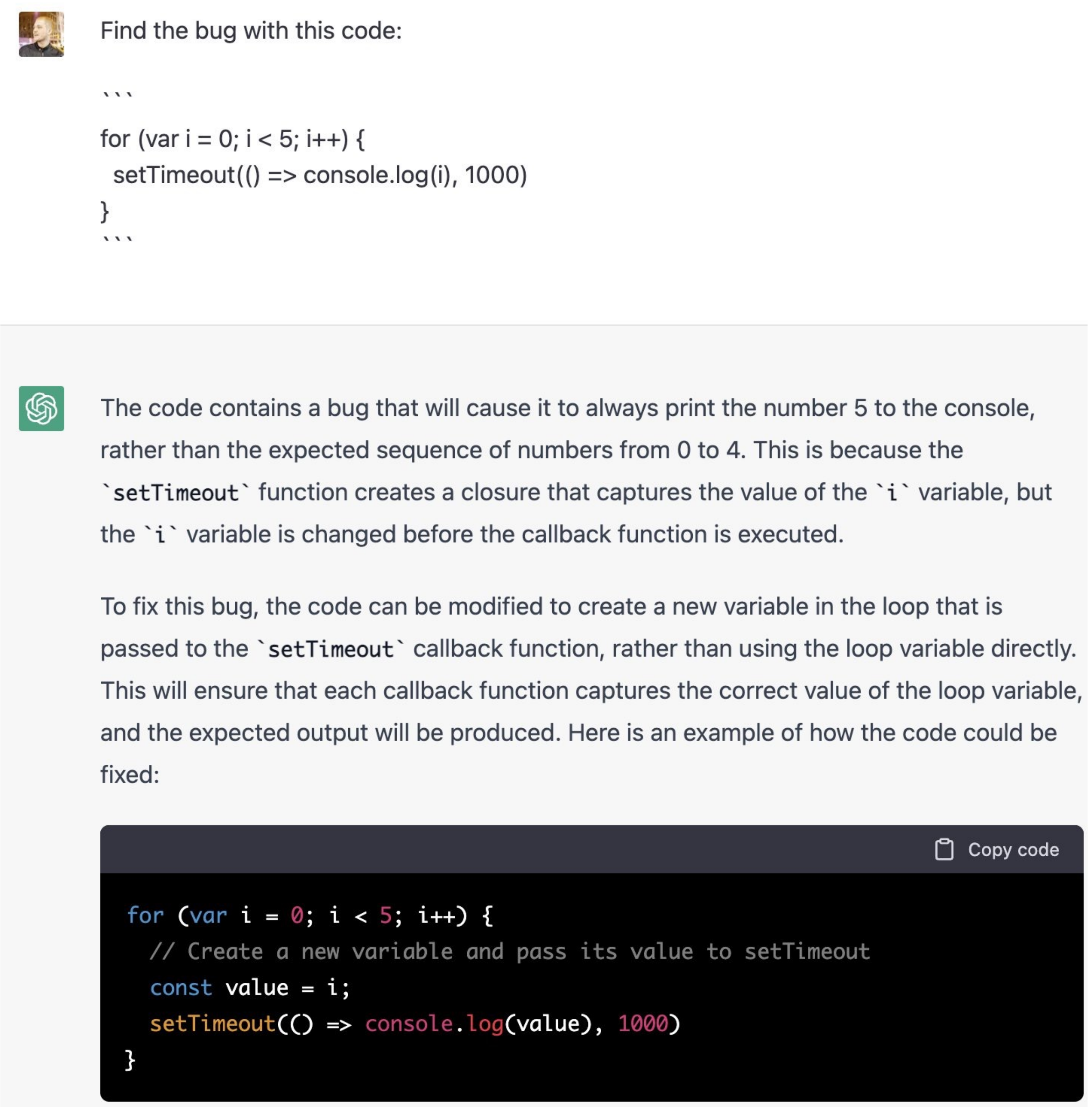
Open Datasets and Decentralized Compute for LLMs

Tri Dao
Together AI / Princeton University
https://tridao.me

Machine Learning Has Made Exciting Progress

Fix Bugs

(ChatGPT/GPT4 - OpenAI)



Find the bug with this code:

```
for (var i = 0; i < 5; i++) {
  setTimeout(() => console.log(i), 1000)
}
```

The code contains a bug that will cause it to always print the number 5 to the console, rather than the expected sequence of numbers from 0 to 4. This is because the `setTimeout` function creates a closure that captures the value of the `i` variable, but the `i` variable is changed before the callback function is executed.

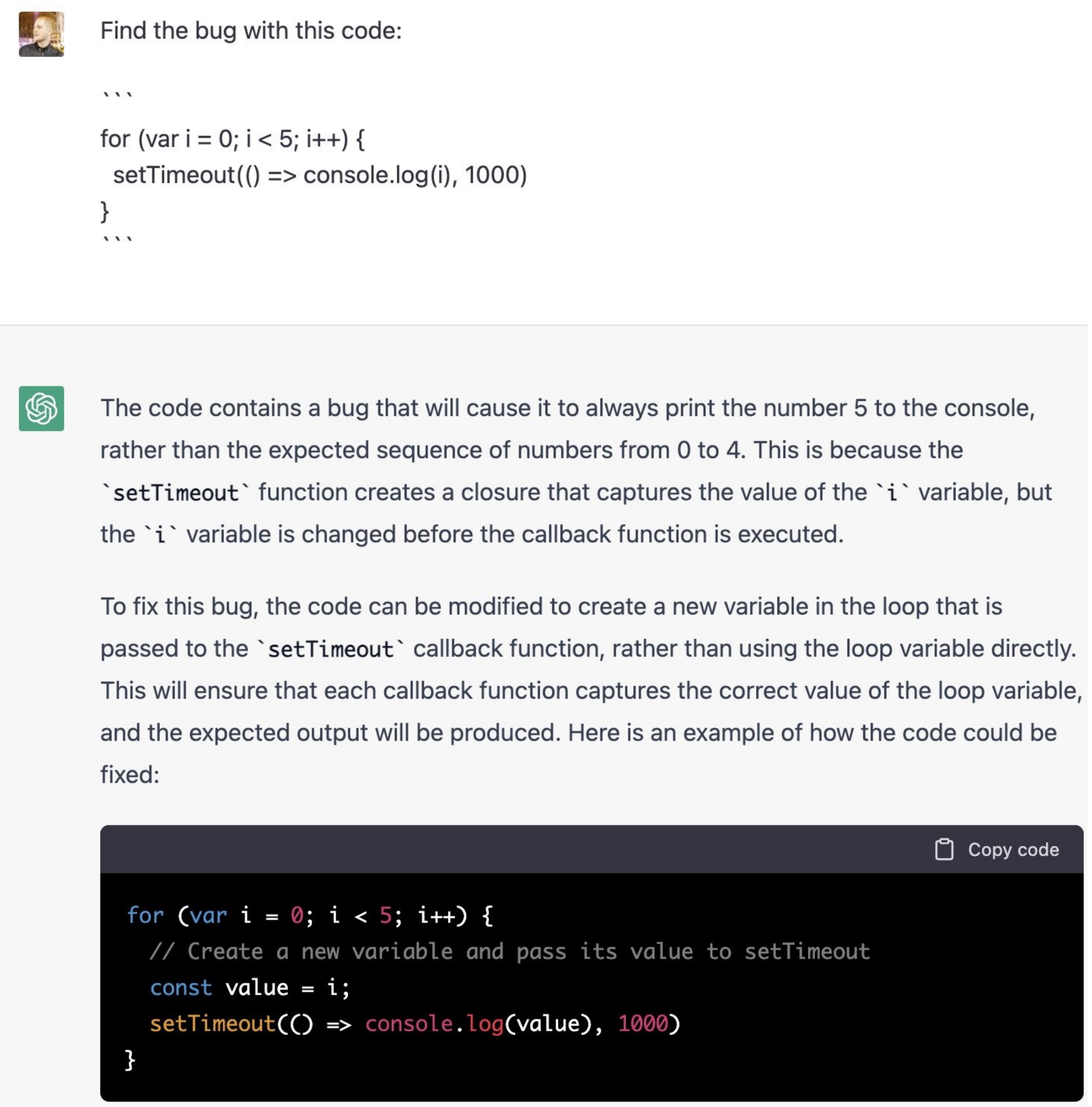
To fix this bug, the code can be modified to create a new variable in the loop that is passed to the `setTimeout` callback function, rather than using the loop variable directly. This will ensure that each callback function captures the correct value of the loop variable, and the expected output will be produced. Here is an example of how the code could be fixed:

```
for (var i = 0; i < 5; i++) {
  // Create a new variable and pass its value to setTimeout
  const value = i;
  setTimeout(() => console.log(value), 1000)
}
```

Machine Learning Has Made Exciting Progress

Fix Bugs

(ChatGPT/GPT4 - OpenAI)



Find the bug with this code:

```
for (var i = 0; i < 5; i++) {
  setTimeout(() => console.log(i), 1000)
}
```

The code contains a bug that will cause it to always print the number 5 to the console, rather than the expected sequence of numbers from 0 to 4. This is because the `setTimeout` function creates a closure that captures the value of the `i` variable, but the `i` variable is changed before the callback function is executed.

To fix this bug, the code can be modified to create a new variable in the loop that is passed to the `setTimeout` callback function, rather than using the loop variable directly. This will ensure that each callback function captures the correct value of the loop variable, and the expected output will be produced. Here is an example of how the code could be fixed:

```
for (var i = 0; i < 5; i++) {
  // Create a new variable and pass its value to setTimeout
  const value = i;
  setTimeout(() => console.log(value), 1000)
}
```

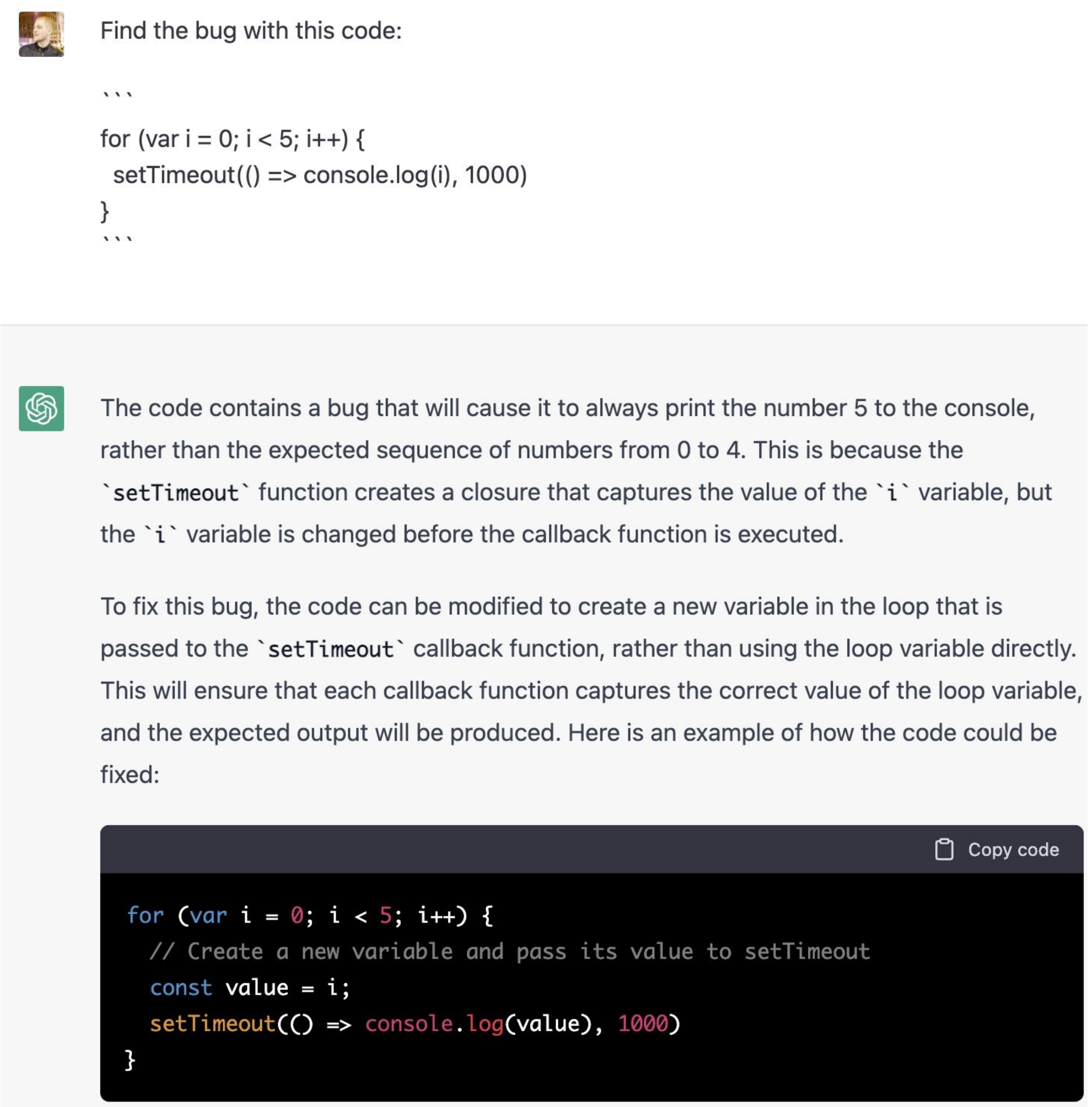
Generate Art

(Stable Diffusion – Stability.AI)

Machine Learning Has Made Exciting Progress

Fix Bugs

(ChatGPT/GPT4 - OpenAI)



Find the bug with this code:

```
```
for (var i = 0; i < 5; i++) {
  setTimeout(() => console.log(i), 1000)
}
```
```

The code contains a bug that will cause it to always print the number 5 to the console, rather than the expected sequence of numbers from 0 to 4. This is because the `setTimeout` function creates a closure that captures the value of the `i` variable, but the `i` variable is changed before the callback function is executed.

To fix this bug, the code can be modified to create a new variable in the loop that is passed to the `setTimeout` callback function, rather than using the loop variable directly. This will ensure that each callback function captures the correct value of the loop variable, and the expected output will be produced. Here is an example of how the code could be fixed:

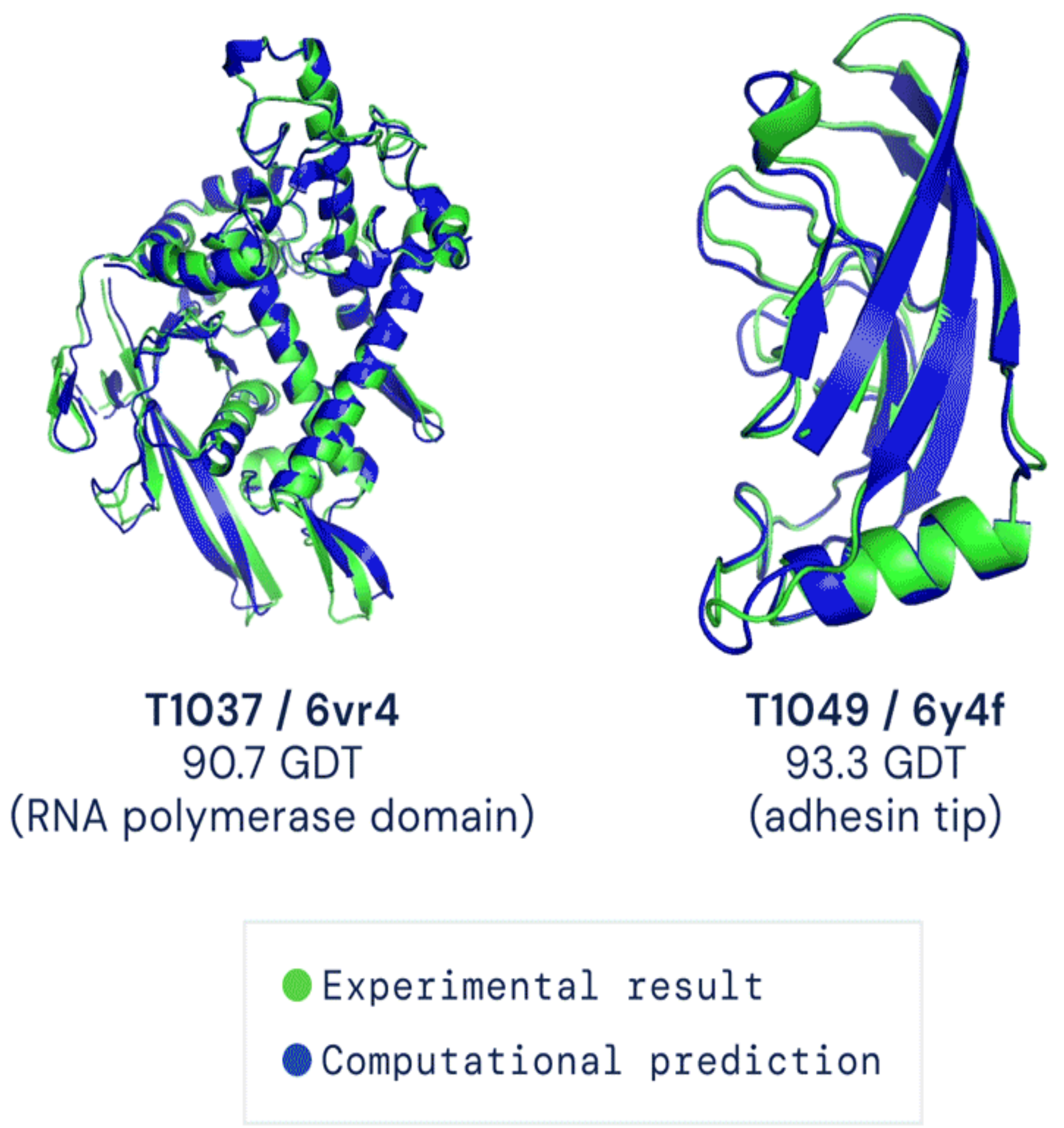
```
for (var i = 0; i < 5; i++) {
  // Create a new variable and pass its value to setTimeout
  const value = i;
  setTimeout(() => console.log(value), 1000)
}
```

Generate Art

(Stable Diffusion – Stability.AI)

Design Drugs

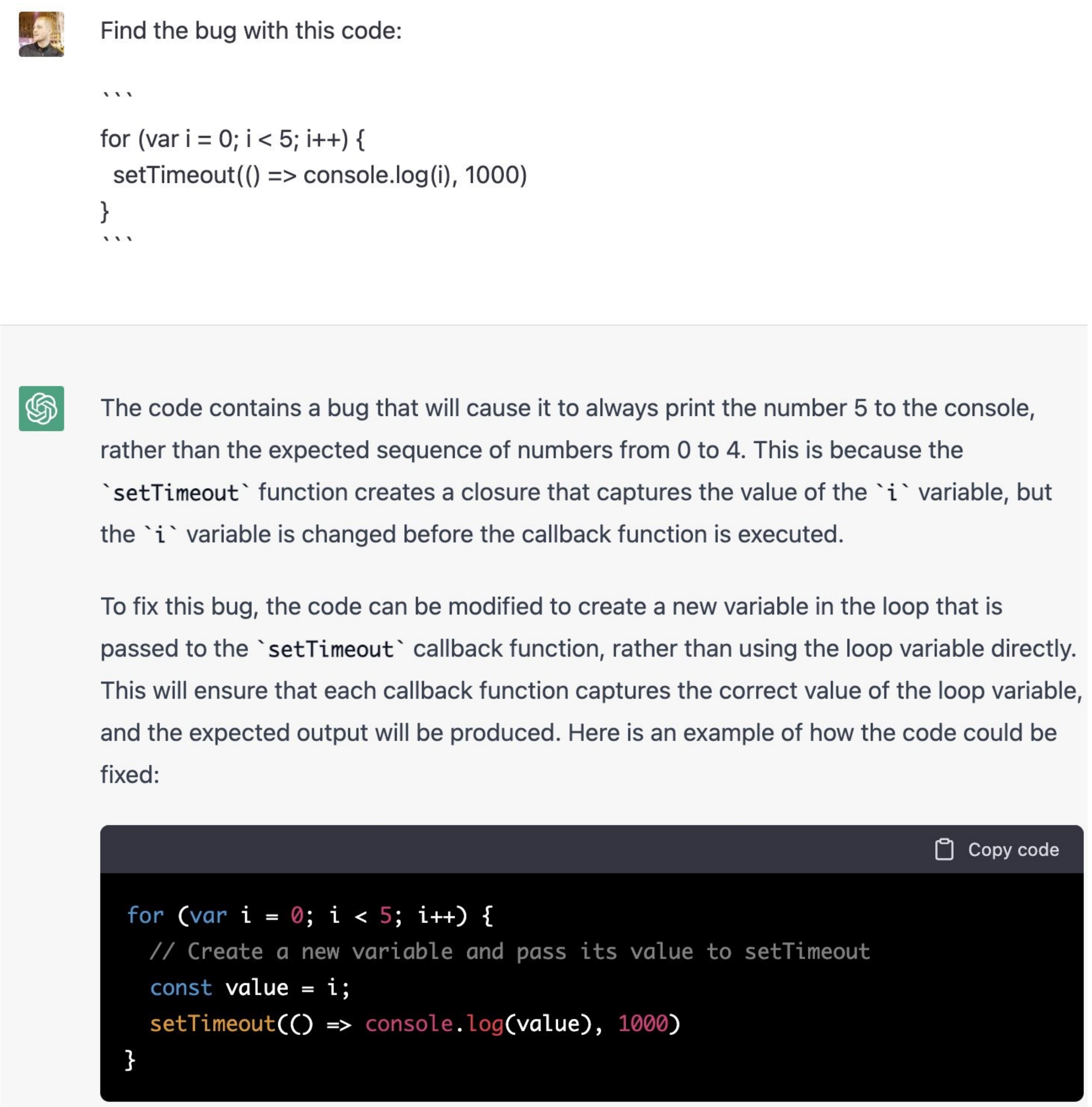
(AlphaFold – DeepMind)



Machine Learning Has Made Exciting Progress

Fix Bugs

(ChatGPT/GPT4 - OpenAI)



Find the bug with this code:

```
```
for (var i = 0; i < 5; i++) {
  setTimeout(() => console.log(i), 1000)
}
```
```

The code contains a bug that will cause it to always print the number 5 to the console, rather than the expected sequence of numbers from 0 to 4. This is because the `setTimeout` function creates a closure that captures the value of the `i` variable, but the `i` variable is changed before the callback function is executed.

To fix this bug, the code can be modified to create a new variable in the loop that is passed to the `setTimeout` callback function, rather than using the loop variable directly. This will ensure that each callback function captures the correct value of the loop variable, and the expected output will be produced. Here is an example of how the code could be fixed:

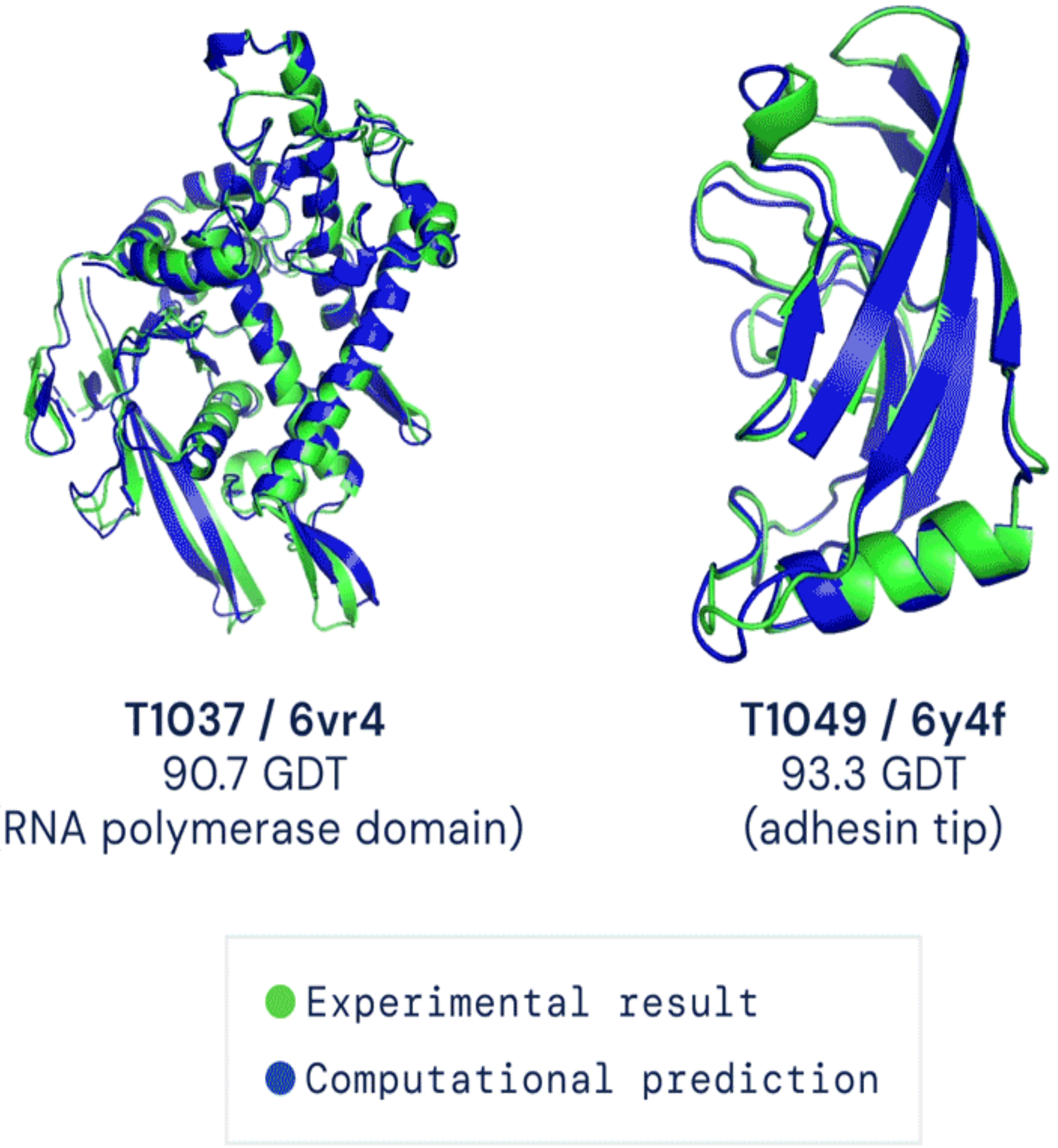
```
for (var i = 0; i < 5; i++) {
  // Create a new variable and pass its value to setTimeout
  const value = i;
  setTimeout(() => console.log(value), 1000)
}
```

Generate Art

(Stable Diffusion – Stability.AI)

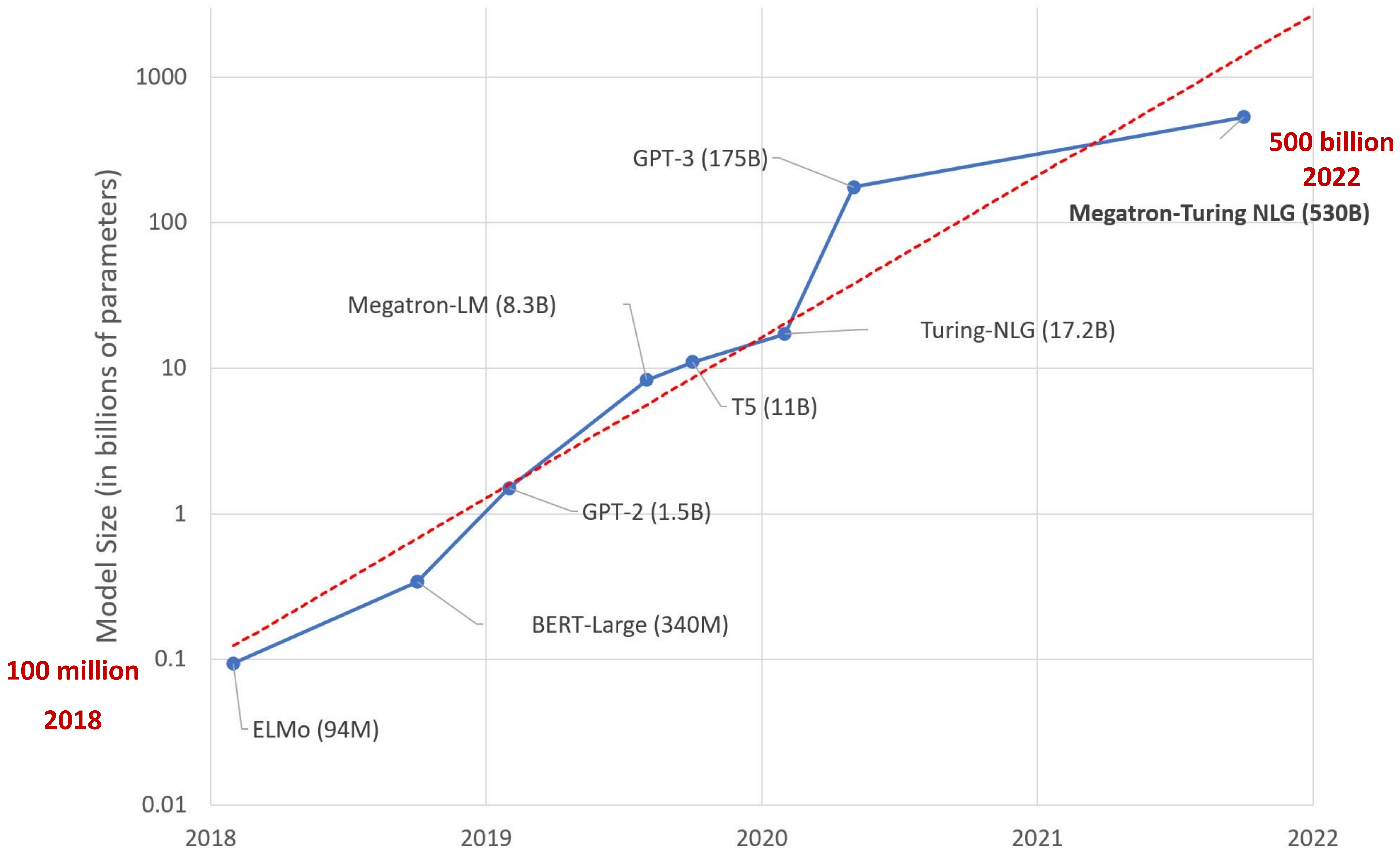
Design Drugs

(AlphaFold – DeepMind)



What enabled these advances? What are outstanding problems? How do we approach them?

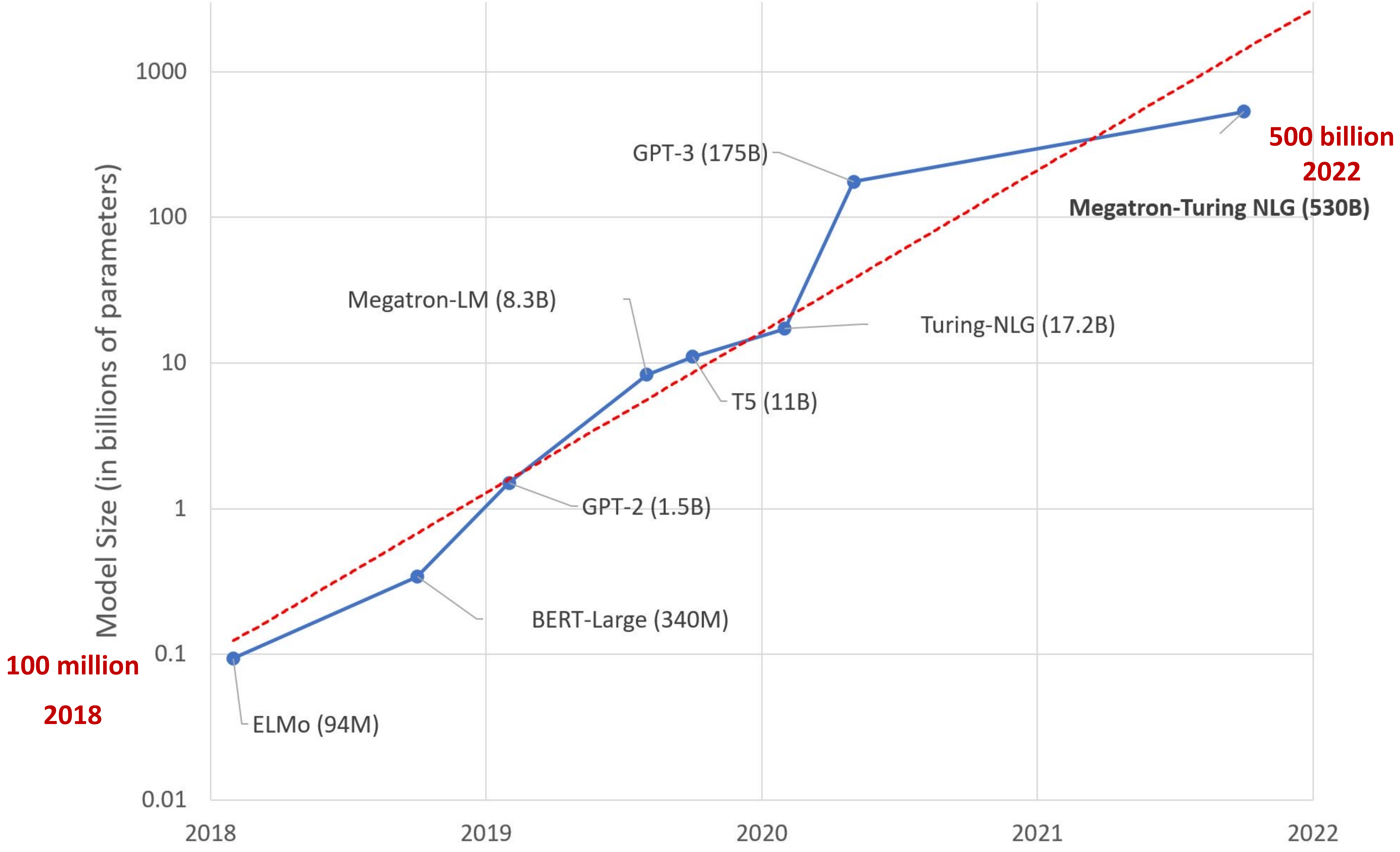
Scale in Data and Compute Brings Quality and Capabilities



100 million
2018

500 billion
2022

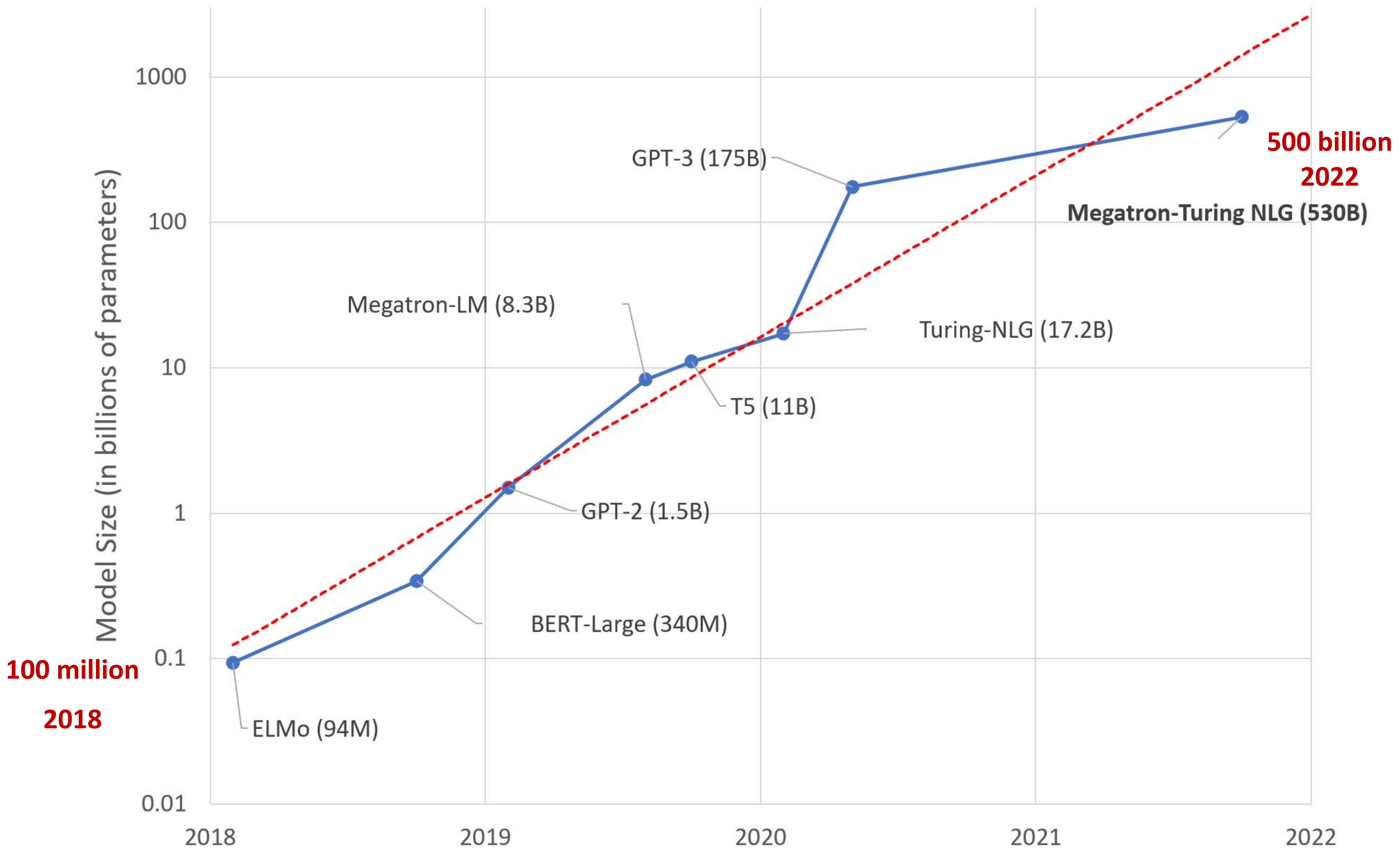
Scale in Data and Compute Brings Quality and Capabilities



Language models explaining jokes

Input: I tried 10000 random restarts of my neural network, but I was accused of overfitting. I guess no good seed goes unpunished.

Scale in Data and Compute Brings Quality and Capabilities

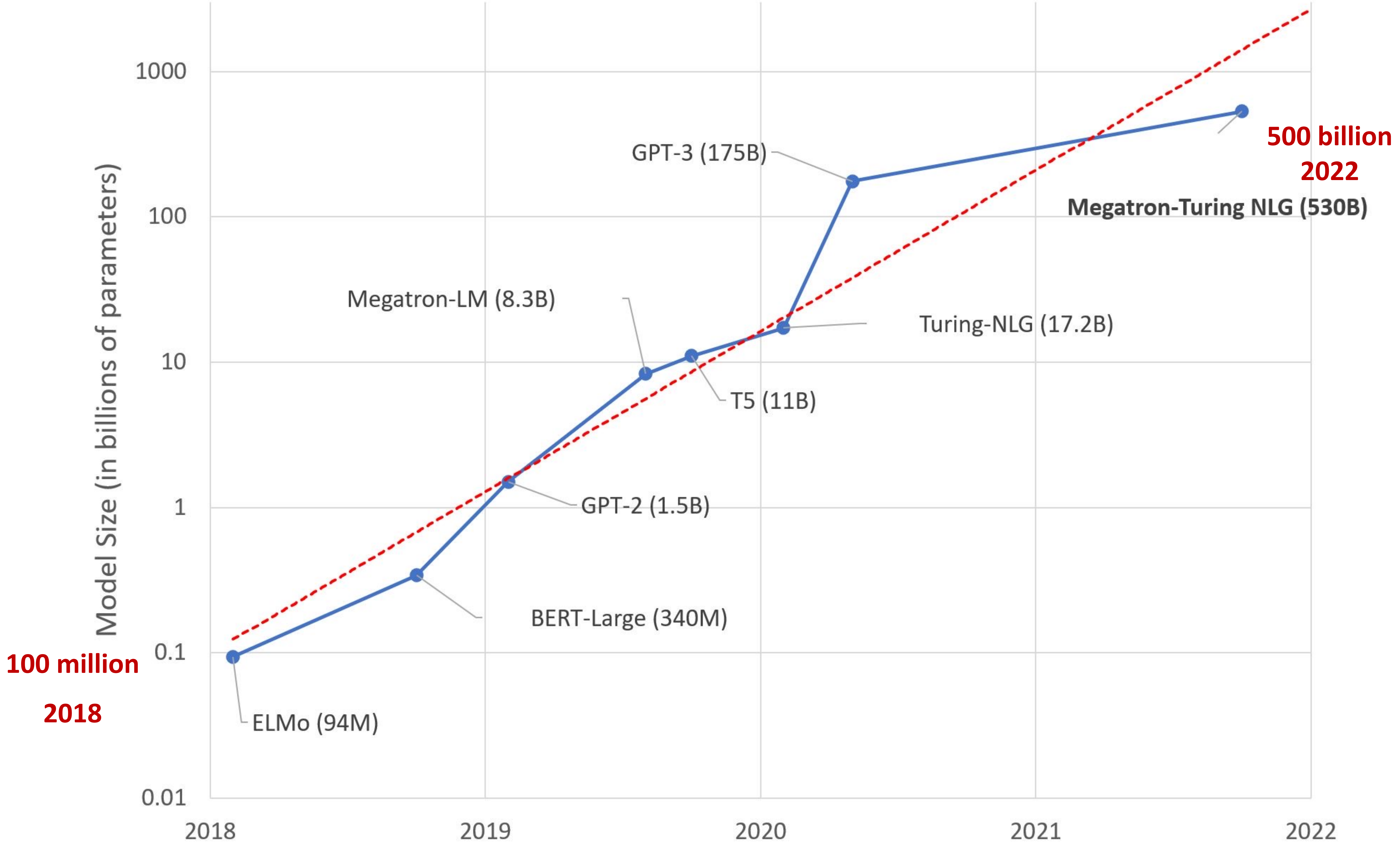


Language models explaining jokes

Input: I tried 10000 random restarts of my neural network, but I was accused of overfitting. I guess no good seed goes unpunished.

1.3B model: The joke is that if you try 10000 different seed choices, you'll eventually find one that works, but you'll be accused of overfitting.

Scale in Data and Compute Brings Quality and Capabilities



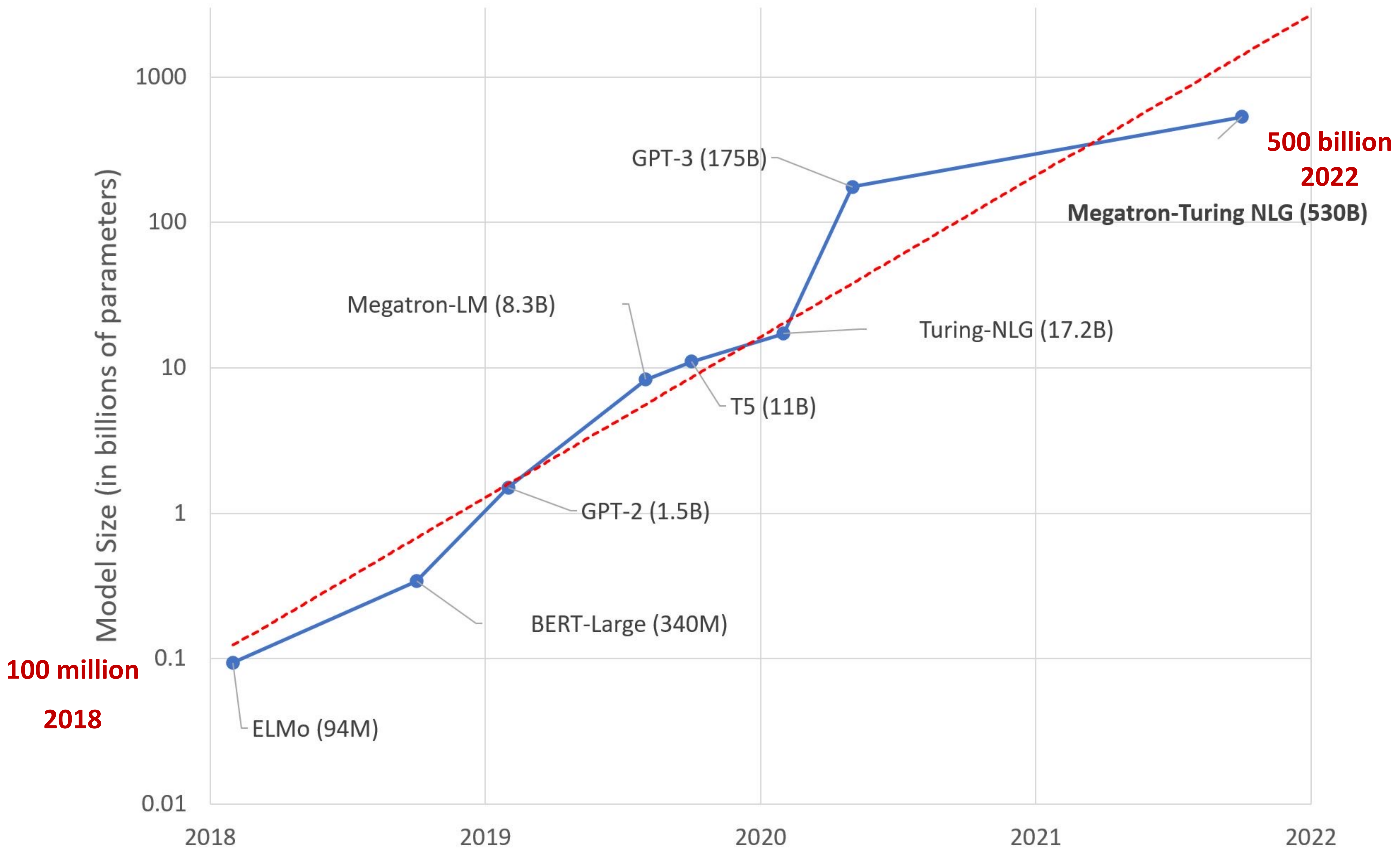
Language models explaining jokes

Input: I tried 10000 random restarts of my neural network, but I was accused of overfitting. I guess no good seed goes unpunished.

1.3B model: The joke is that if you try 10000 different seed choices, you'll eventually find one that works, but you'll be accused of overfitting.

175B model: This joke is a play on words related to neural networks, a type of machine learning algorithm.
The punchline, "I guess **no good seed goes unpunished**," is a play on the phrase "**no good deed goes unpunished**." In this case, "good seed" refers to a starting point for the random restarts, and the joke implies that even when trying to improve the neural network's performance, the person is still accused of overfitting.

Scale in Data and Compute Brings Quality and Capabilities



Language models explaining jokes

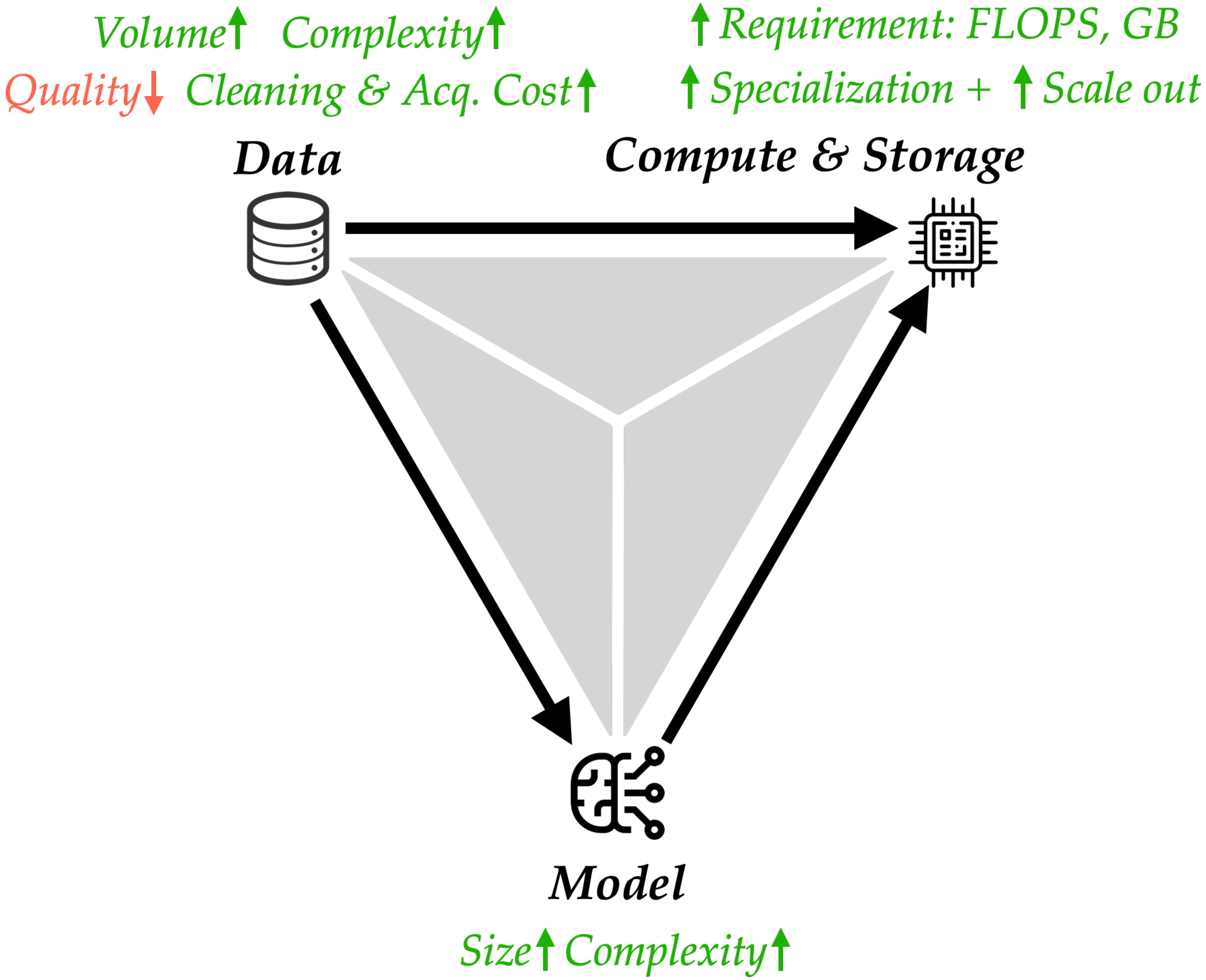
Input: I tried 10000 random restarts of my neural network, but I was accused of overfitting. I guess no good seed goes unpunished.

1.3B model: The joke is that if you try 10000 different seed choices, you'll eventually find one that works, but you'll be accused of overfitting.

175B model: This joke is a play on words related to neural networks, a type of machine learning algorithm.
The punchline, "I guess **no good seed goes unpunished**," is a play on the phrase "**no good deed goes unpunished**." In this case, "good seed" refers to a starting point for the random restarts, and the joke implies that even when trying to improve the neural network's performance, the person is still accused of overfitting.

Scale is more closely tied to advances in ML than ever before

Challenges with Scale



Volume↑ Complexity↑

Quality↓ Cleaning & Acq. Cost↑

Data

↑Requirement: FLOPS, GB

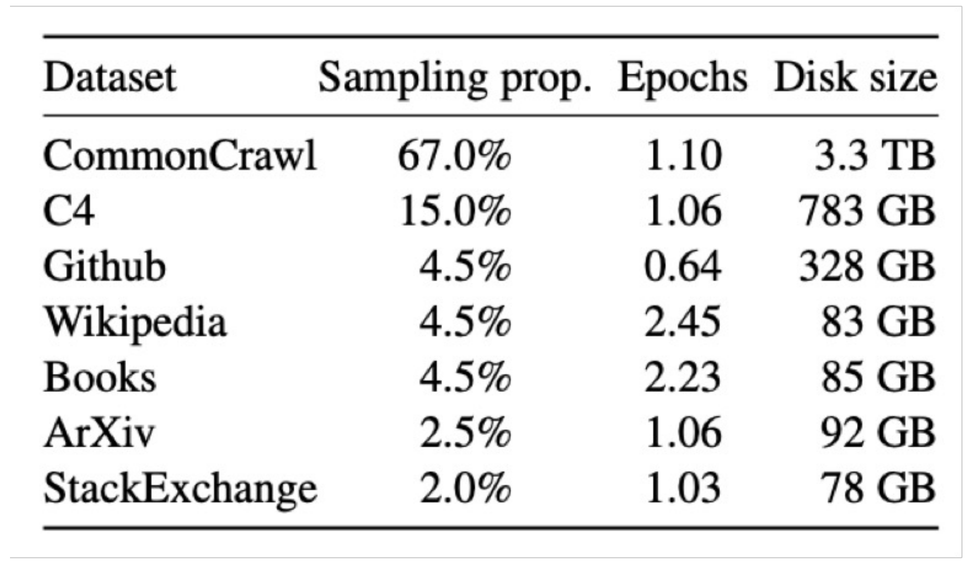
↑Specialization + ↑Scale out

Compute & Storage

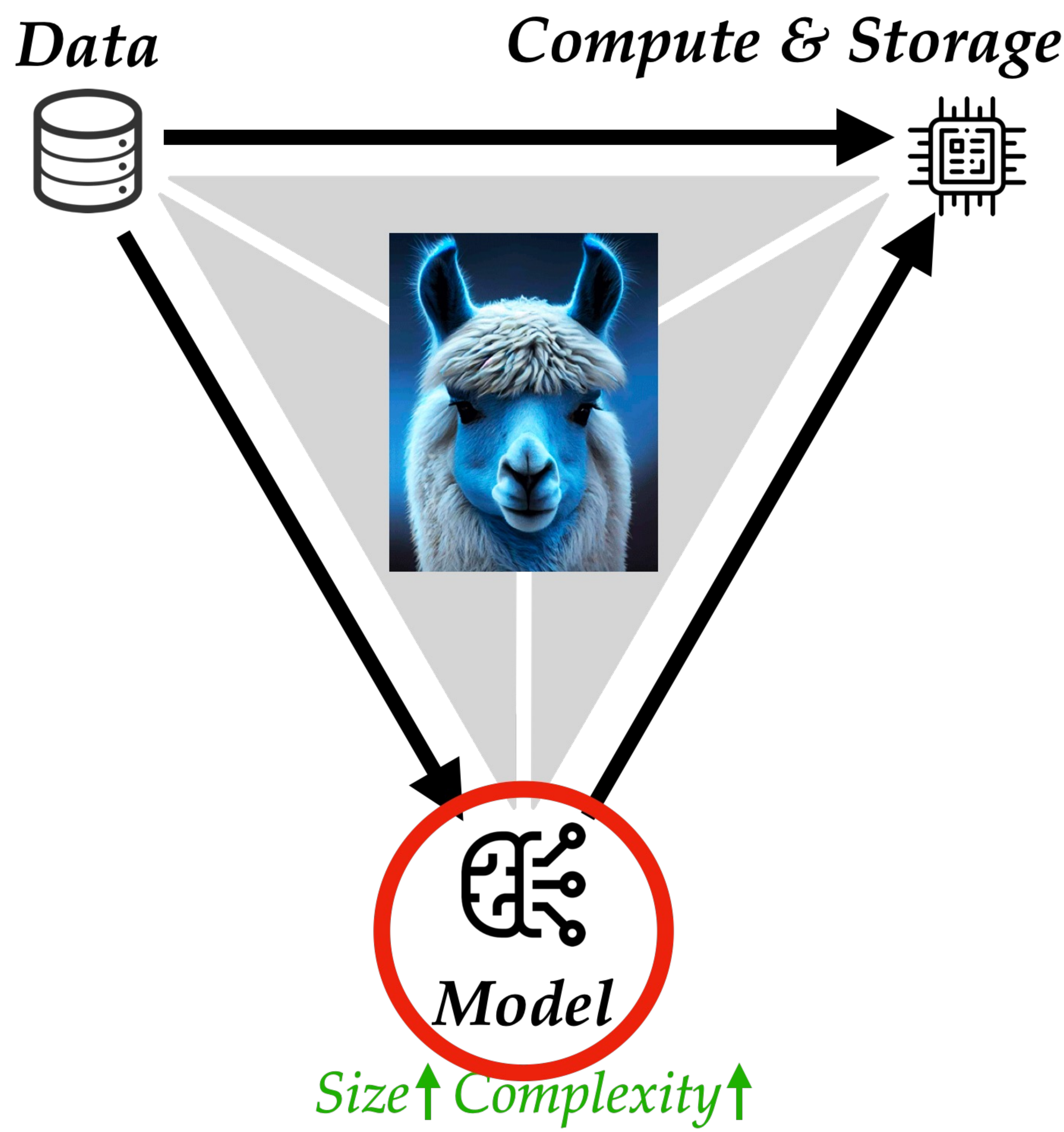
Model

Size↑ Complexity↑

The Llama Moment



Dataset	Sampling prop.	Epochs	Disk size
CommonCrawl	67.0%	1.10	3.3 TB
C4	15.0%	1.06	783 GB
Github	4.5%	0.64	328 GB
Wikipedia	4.5%	2.45	83 GB
Books	4.5%	2.23	85 GB
ArXiv	2.5%	1.06	92 GB
StackExchange	2.0%	1.03	78 GB



Volume↑ Complexity↑

Quality↓ Cleaning & Acq. Cost↑

Data

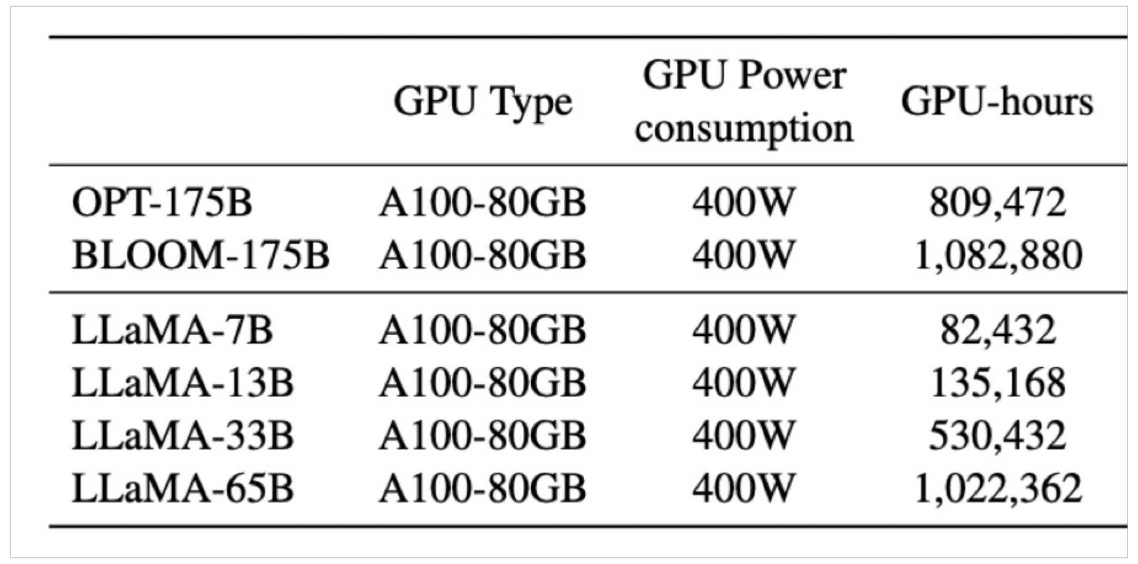
↑ Requirement: FLOPS, GB

↑ Specialization + ↑ Scale out

Compute & Storage

Model

Size↑ Complexity↑



	GPU Type	GPU Power consumption	GPU-hours
OPT-175B	A100-80GB	400W	809,472
BLOOM-175B	A100-80GB	400W	1,082,880
LLaMA-7B	A100-80GB	400W	82,432
LLaMA-13B	A100-80GB	400W	135,168
LLaMA-33B	A100-80GB	400W	530,432
LLaMA-65B	A100-80GB	400W	1,022,362

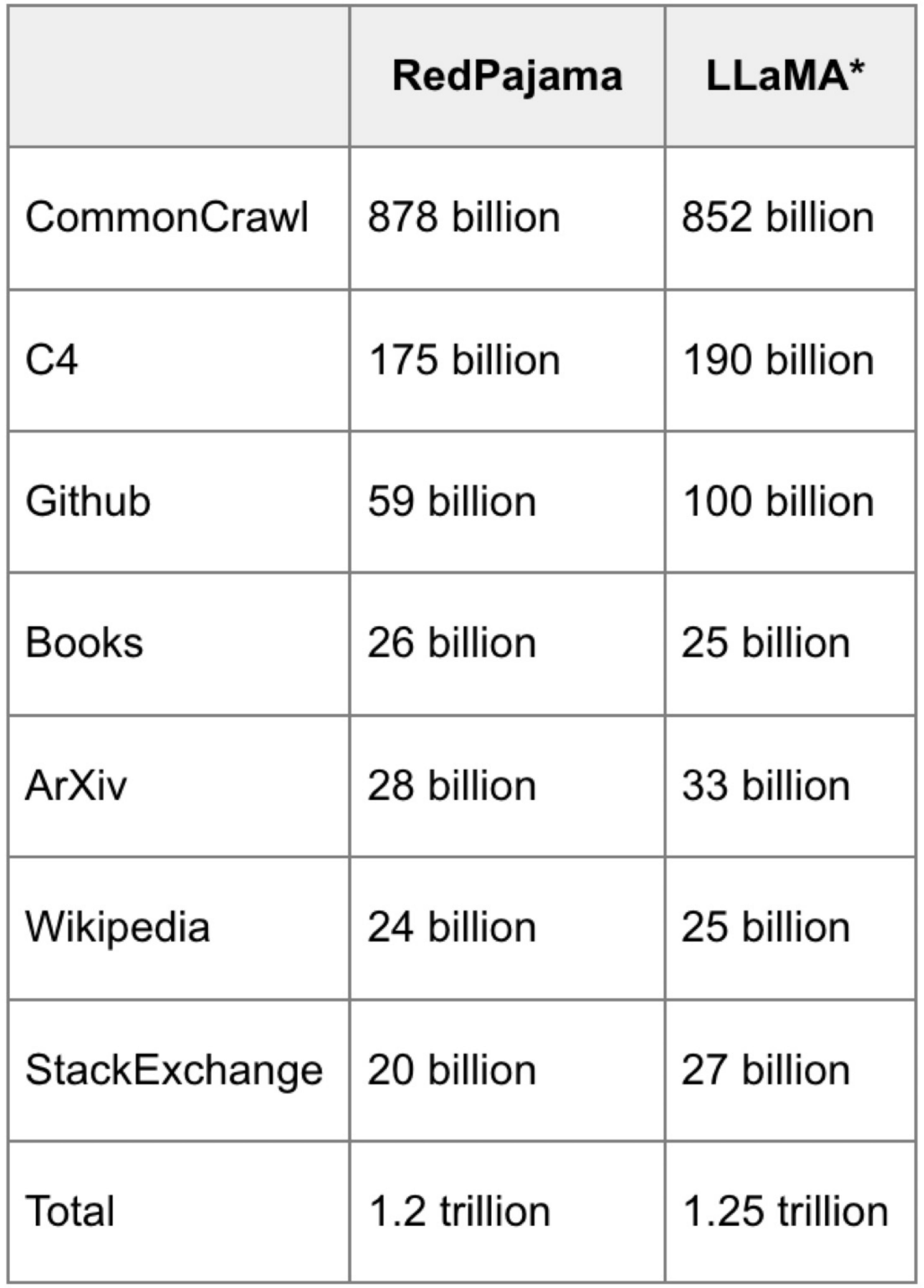
x

RedPajama v1: Data

- CommonCrawl
- C4
- GitHub
- arXiv
- Books
- Wikipedia
- StackExchange

RedPajama v1: Data

- CommonCrawl
- C4
- GitHub
- arXiv
- Books
- Wikipedia
- StackExchange



	RedPajama	LLaMA*
CommonCrawl	878 billion	852 billion
C4	175 billion	190 billion
Github	59 billion	100 billion
Books	26 billion	25 billion
ArXiv	28 billion	33 billion
Wikipedia	24 billion	25 billion
StackExchange	20 billion	27 billion
Total	1.2 trillion	1.25 trillion

Fueling and Exciting Generation of Open Models

TOGETHER
mosaicML
salesforce

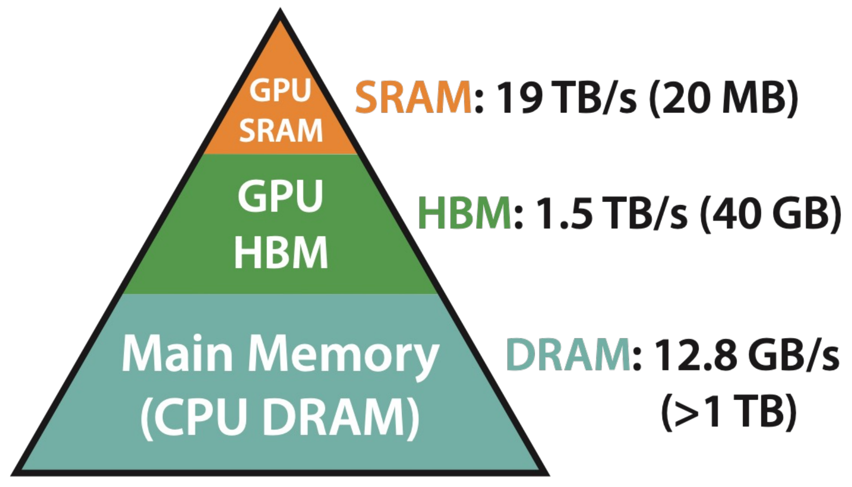
RedPajama-INCITE	OpenLlama	Mosaic MPT	Salesforce XGen
7/7 Slices	7/7 Slices	5/10 Slices	5/12 Slices

Compute: Hardware-aware Algorithms

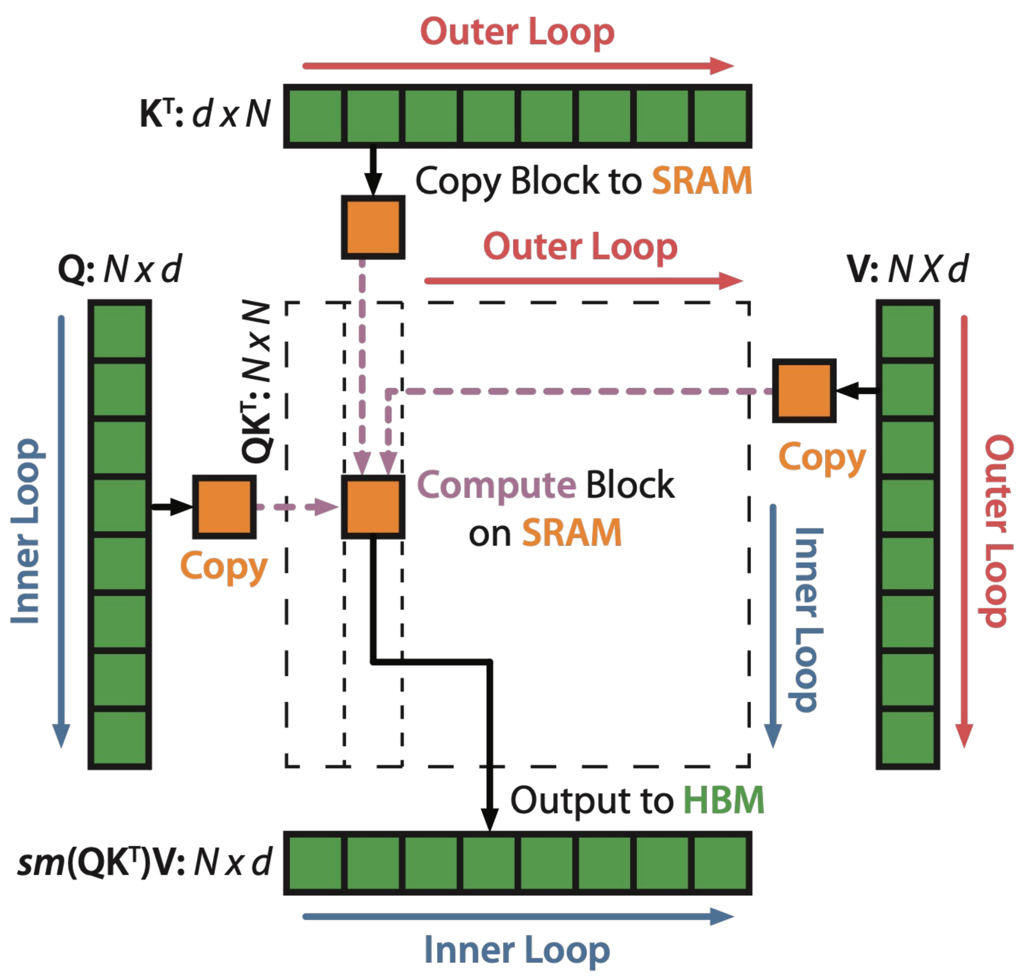
IO-awareness:
reducing reads/writes to GPU memory yields significant speedup

Compute: Hardware-aware Algorithms

IO-awareness:
reducing reads/writes to GPU memory yields significant speedup



Memory Hierarchy with
Bandwidth & Memory Size

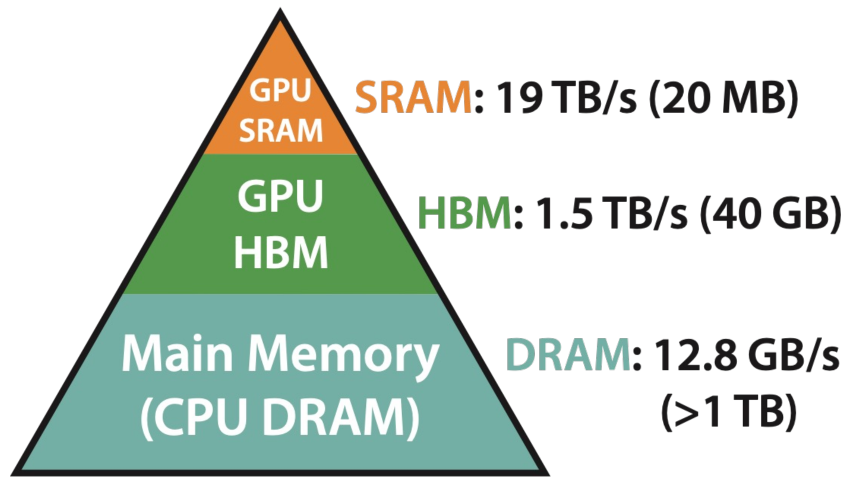


FlashAttention

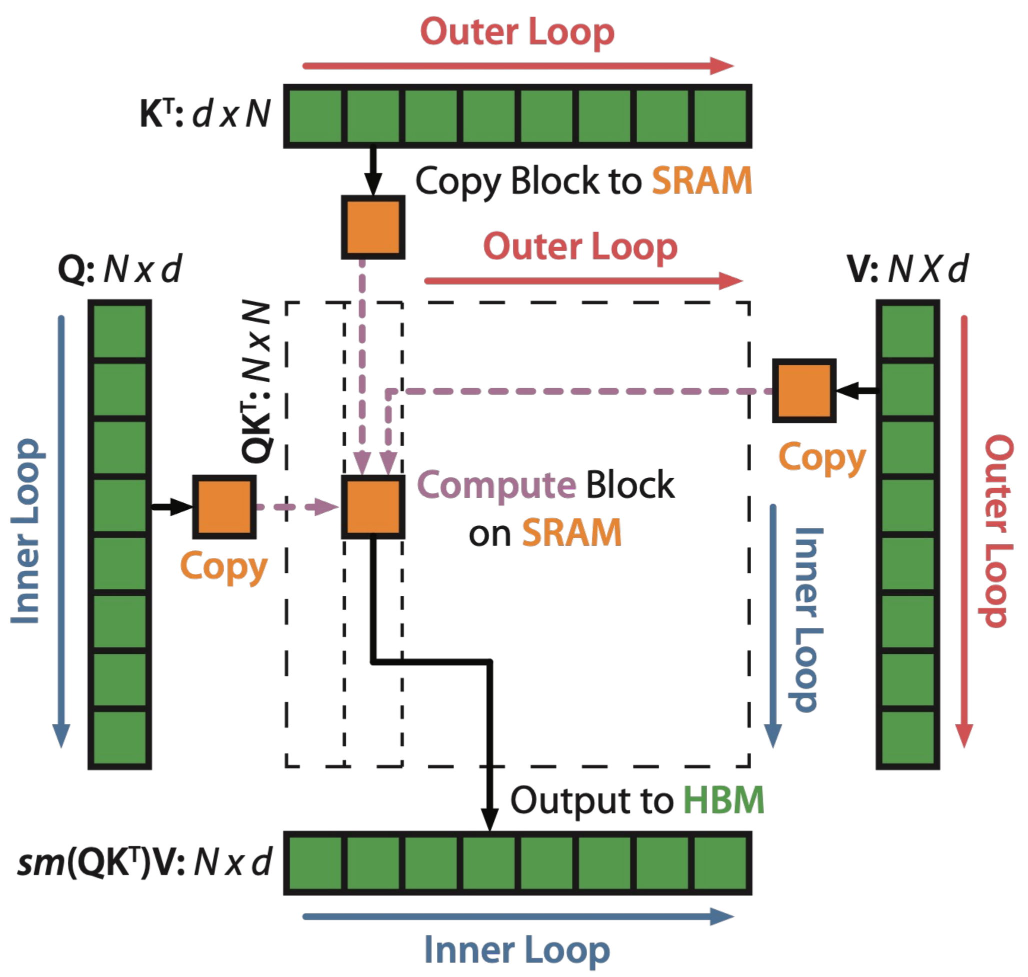
FlashAttention: fast (2-4x) and memory-efficient attention (10-20x) algorithm, with no approximation

Compute: Hardware-aware Algorithms

IO-awareness:
reducing reads/writes to GPU memory yields significant speedup



Memory Hierarchy with Bandwidth & Memory Size

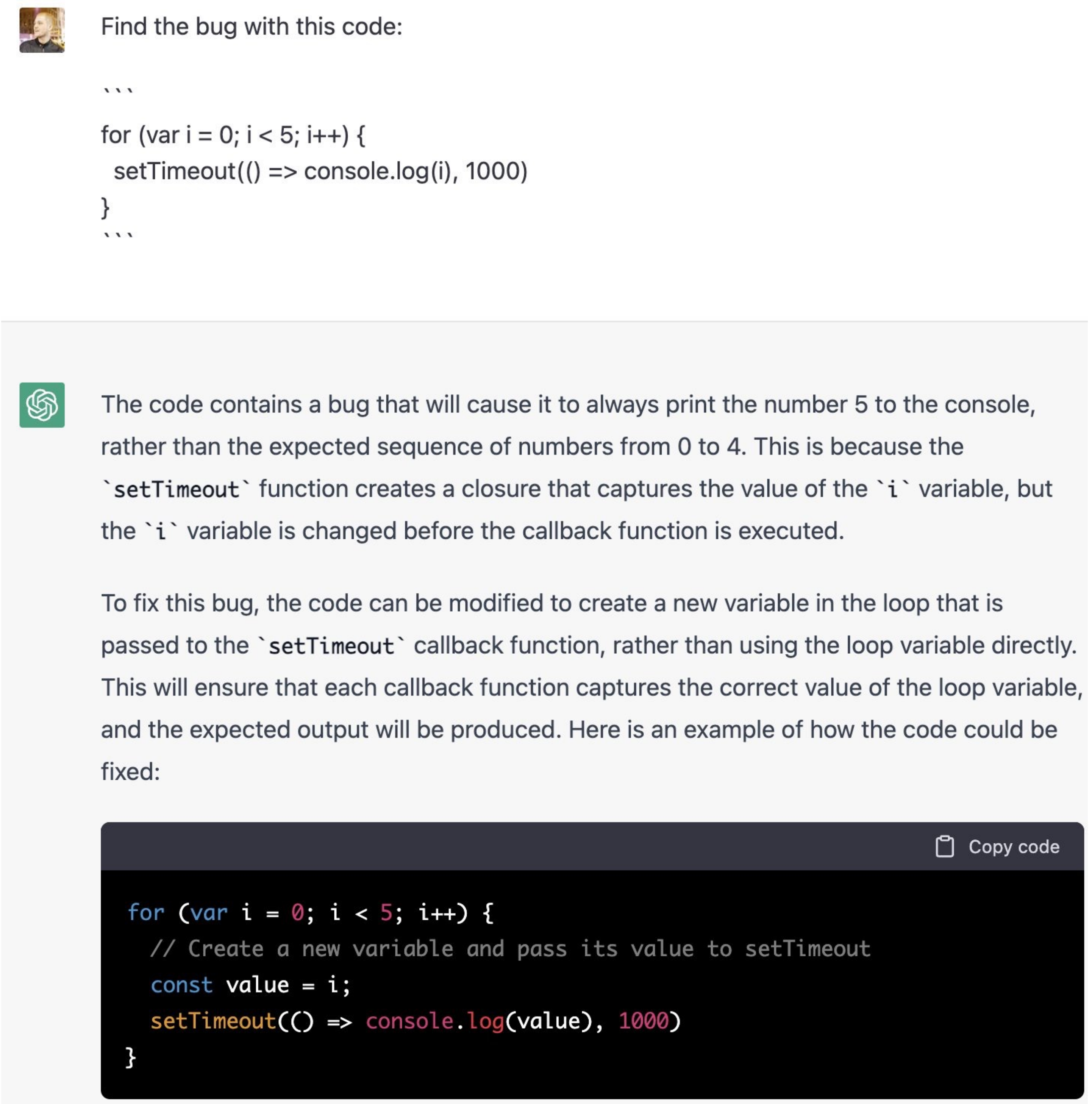


FlashAttention

FlashAttention: fast (4-8x) and memory-efficient attention (10-20x) algorithm, with no approximation

PyTorch OpenAI Meta NVIDIA Microsoft ANTHROP\C stability.ai mosaicML SambaNova SYSTEMS HUGGING FACE

FlashAttention Adoption Areas



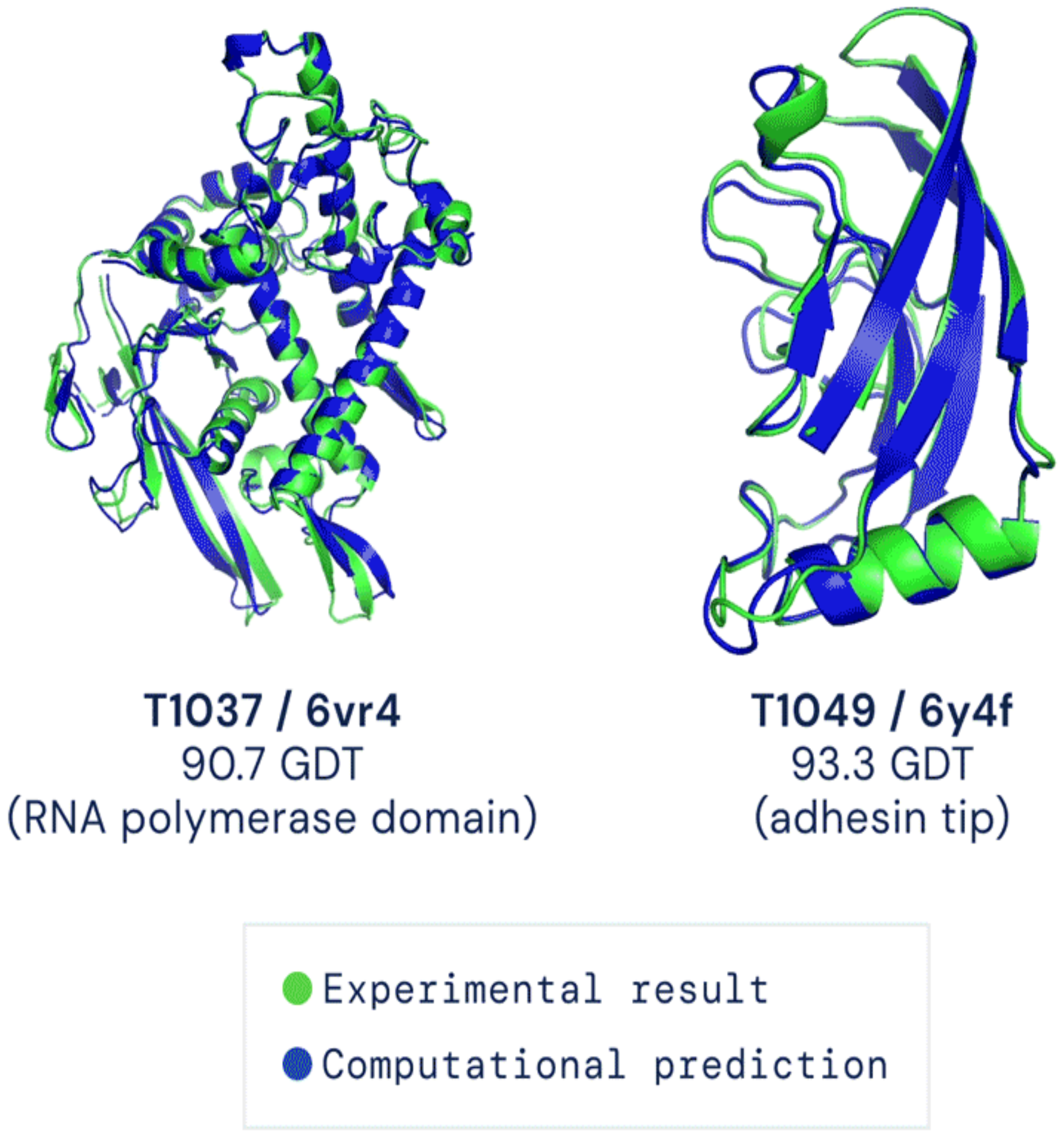
Find the bug with this code:

```
for (var i = 0; i < 5; i++) {
  setTimeout(() => console.log(i), 1000)
}
```

The code contains a bug that will cause it to always print the number 5 to the console, rather than the expected sequence of numbers from 0 to 4. This is because the `setTimeout` function creates a closure that captures the value of the `i` variable, but the `i` variable is changed before the callback function is executed.

To fix this bug, the code can be modified to create a new variable in the loop that is passed to the `setTimeout` callback function, rather than using the loop variable directly. This will ensure that each callback function captures the correct value of the loop variable, and the expected output will be produced. Here is an example of how the code could be fixed:

```
for (var i = 0; i < 5; i++) {
  // Create a new variable and pass its value to setTimeout
  const value = i;
  setTimeout(() => console.log(value), 1000)
}
```



Text Generation

(Llama – Meta, Falcon – TIIUAE, MPT, RedPajama)

Image Generation

(Stable Diffusion - Stability.AI)

Drug Discovery

(OpenFold, UniFold)

Decentralized Communication & Data Movement

Distributed training at scale is communication-intensive.

Decentralized Communication & Data Movement

Distributed training at scale is communication-intensive.

GPT−3

6.7B Parameters

1.20E+22

Floating Point Ops.

32 Machines, 4x A100 GPU each

Each machine send+recv ***4PB*** data

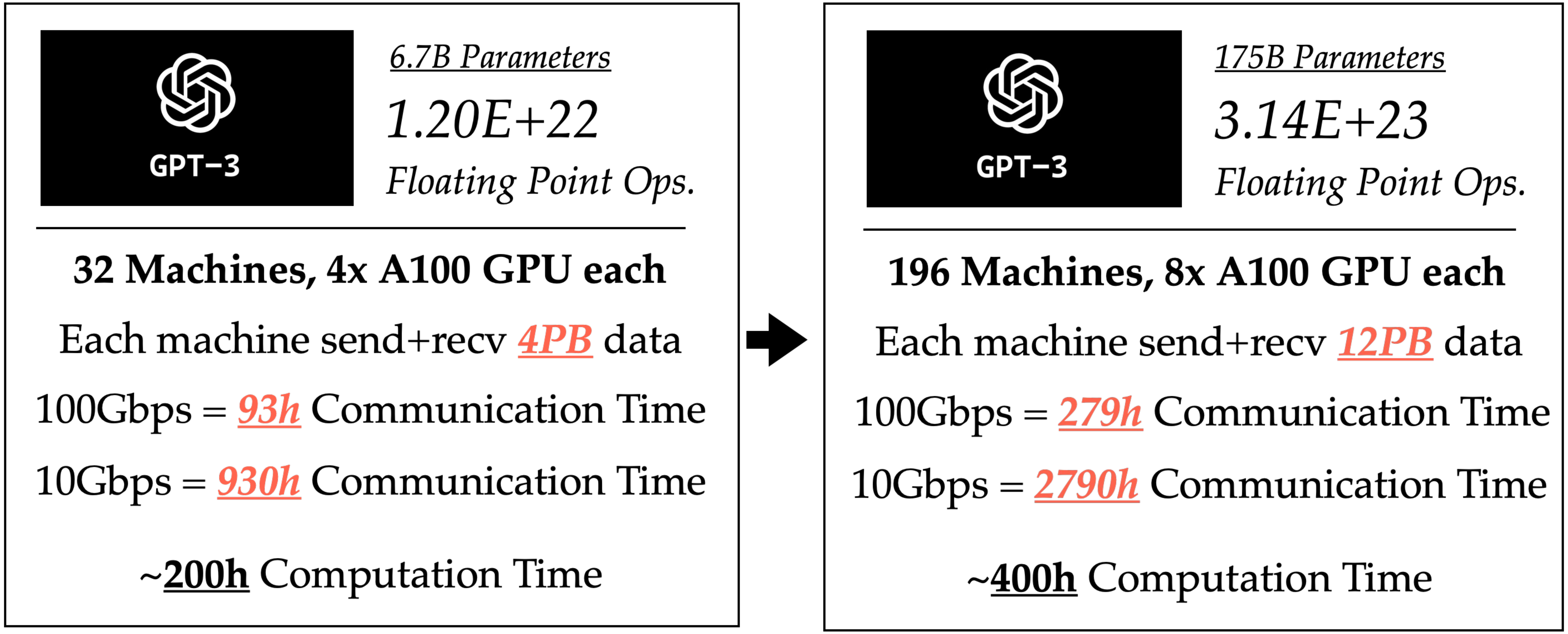
100Gbps = ***93h*** Communication Time

10Gbps = ***930h*** Communication Time

~**200h** Computation Time

Decentralized Communication & Data Movement

Distributed training at scale is communication-intensive.



GPT-3

6.7B Parameters

1.20E+22

Floating Point Ops.

32 Machines, 4x A100 GPU each

Each machine send+recv *4PB* data

100Gbps = *93h* Communication Time

10Gbps = *930h* Communication Time

~**200h** Computation Time

GPT-3

175B Parameters

3.14E+23

Floating Point Ops.

196 Machines, 8x A100 GPU each

Each machine send+recv *12PB* data

100Gbps = *279h* Communication Time

10Gbps = *2790h* Communication Time

~**400h** Computation Time

Decentralized Communication & Data Movement

Distributed training at scale is communication-intensive.

GPT−3

6.7B Parameters

1.20E+22

Floating Point Ops.

32 Machines, 4x A100 GPU each

Each machine send+recv *4PB* data

100Gbps = *93h* Communication Time

10Gbps = *930h* Communication Time

~**200h** Computation Time

GPT−3

175B Parameters

3.14E+23

Floating Point Ops.

196 Machines, 8x A100 GPU each

Each machine send+recv *12PB* data

100Gbps = *279h* Communication Time

10Gbps = *2790h* Communication Time

~**400h** Computation Time

(Today) Model training today is largely restricted to centralized data centers with fast network connections.

Hard to use cheaper alternatives (Tier 2-4 clouds, Spot Instances, Volunteer Computes, etc.).

Decentralized Communication & Data Movement

Distributed training at scale is communication-intensive.

GPT−3

6.7B Parameters

1.20E+22

Floating Point Ops.

32 Machines, 4x A100 GPU each

Each machine send+recv *4PB* data

100Gbps = *93h* Communication Time

10Gbps = *930h* Communication Time

~**200h** Computation Time

GPT−3

175B Parameters

3.14E+23

Floating Point Ops.

196 Machines, 8x A100 GPU each

Each machine send+recv *12PB* data

100Gbps = *279h* Communication Time

10Gbps = *2790h* Communication Time

~**400h** Computation Time

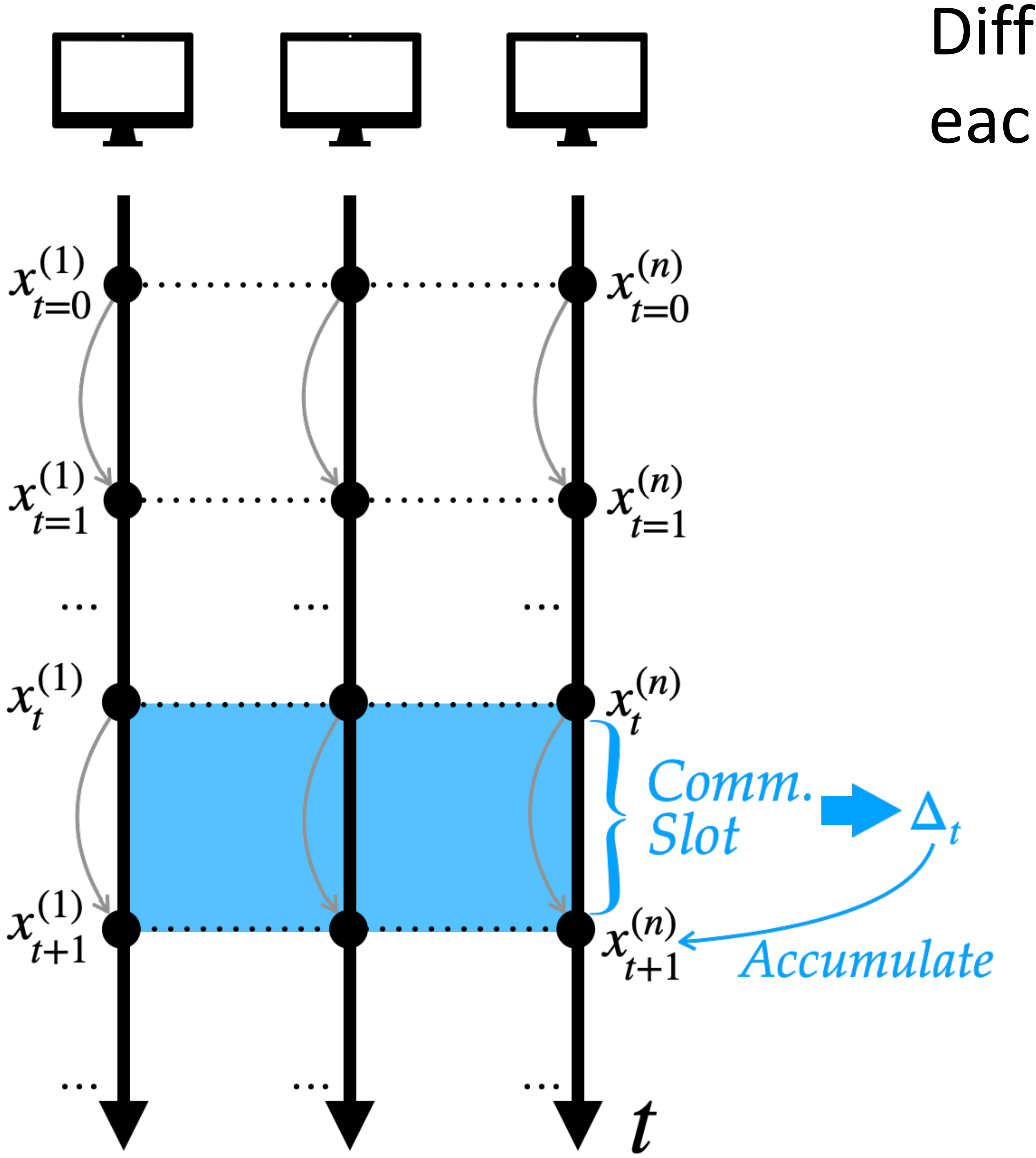
Future: 10x further scaling requires fast connections between 10x machines. Becoming challenging even for data center.

(Today) Model training today is largely restricted to centralized data centers with fast network connections.
Hard to use cheaper alternatives (Tier 2-4 clouds, Spot Instances, Volunteer Computes, etc.).

CocktailSGD: Mixture of Communication Compression Methods

Different communication compression techniques complement each other and compose well!

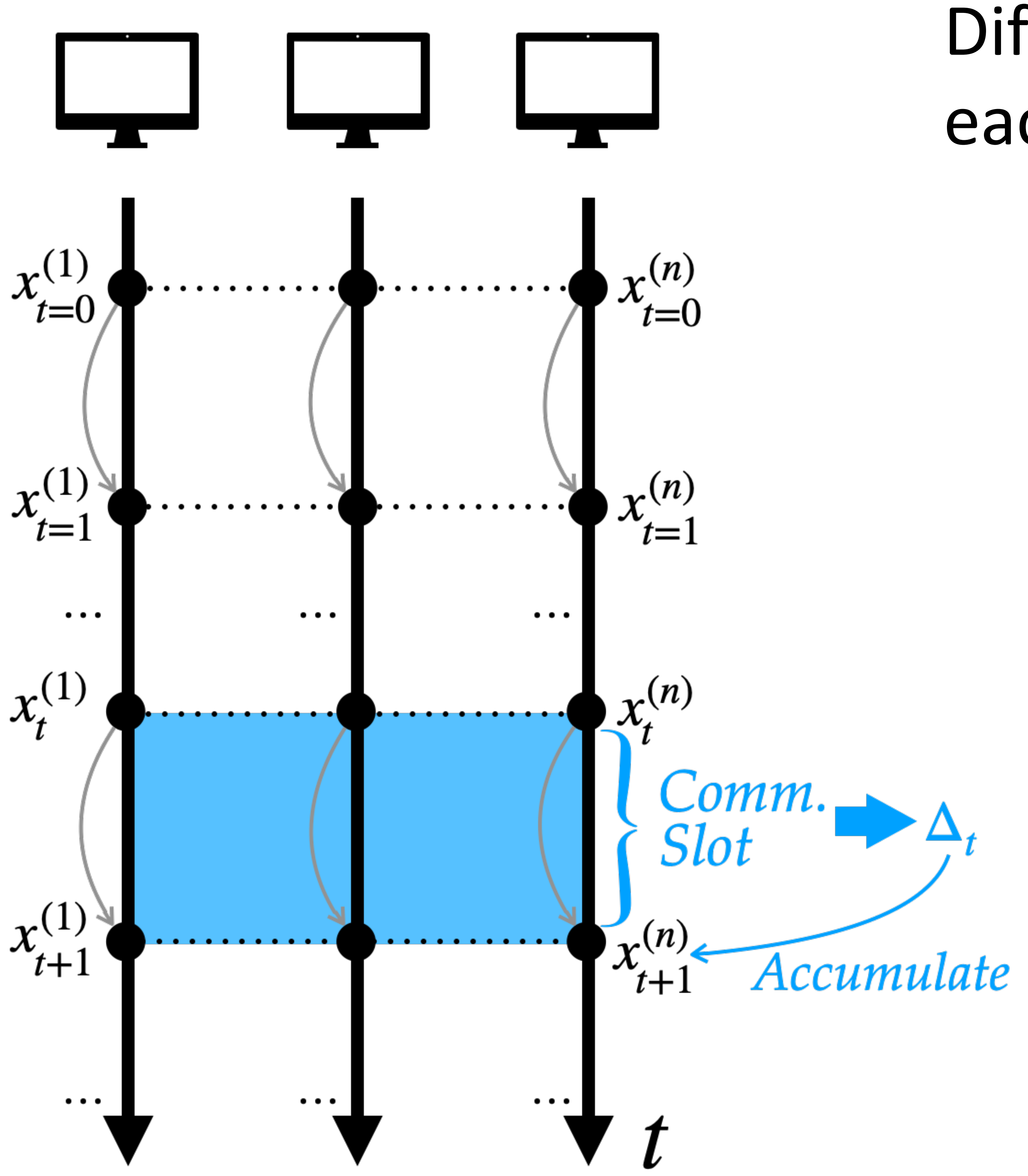
CocktailSGD: Mixture of Communication Compression Methods



As long as Communication fully fills the Comm. Slot, no slow down caused by communication.

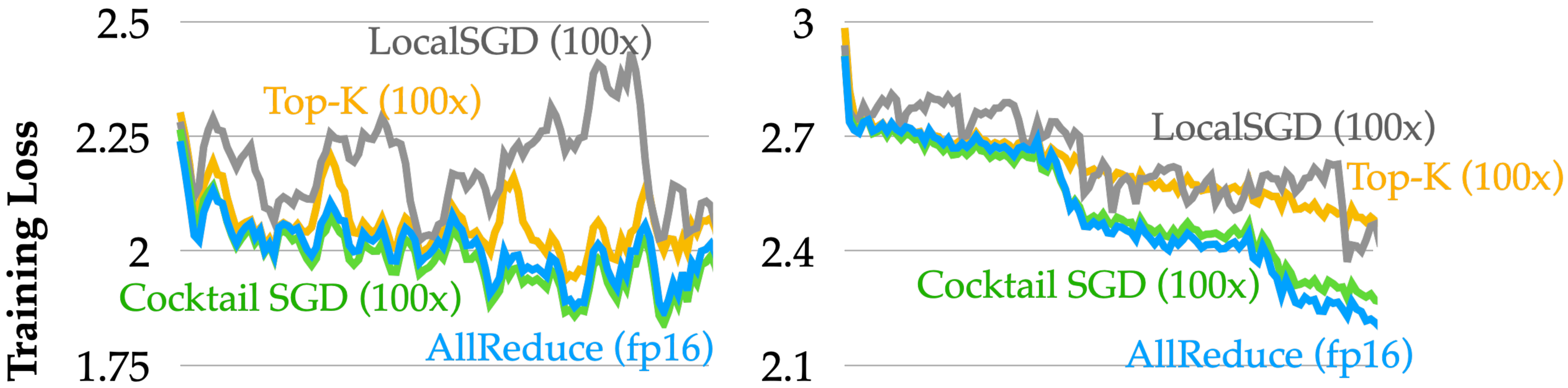
Different communication compression techniques complement each other and compose well!

CocktailSGD: Mixture of Communication Compression Methods



As long as Communication fully fills the Comm. Slot, no slow down caused by communication.

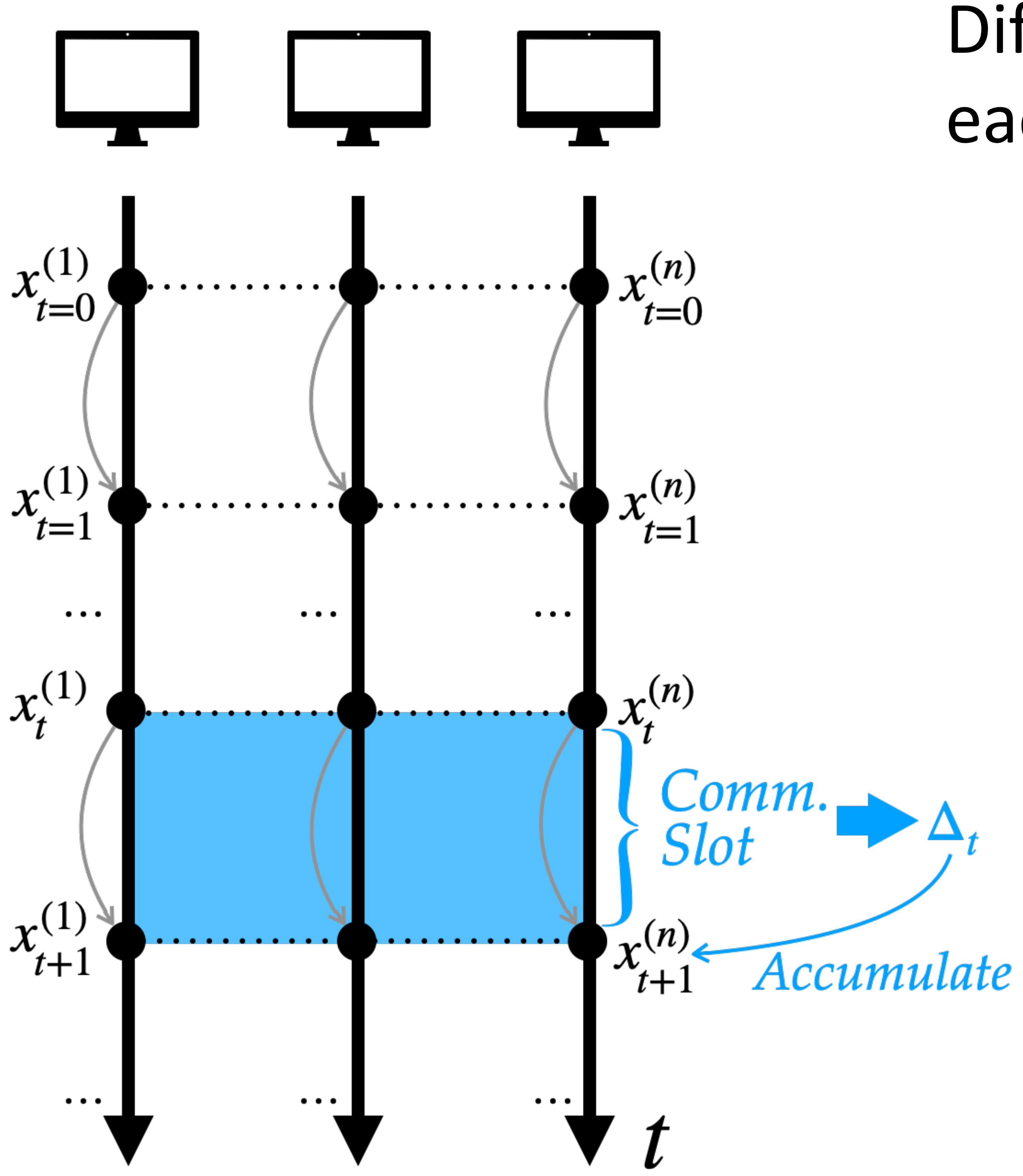
Different communication compression techniques complement each other and compose well!



GPT-J-6B Instruct Tuning

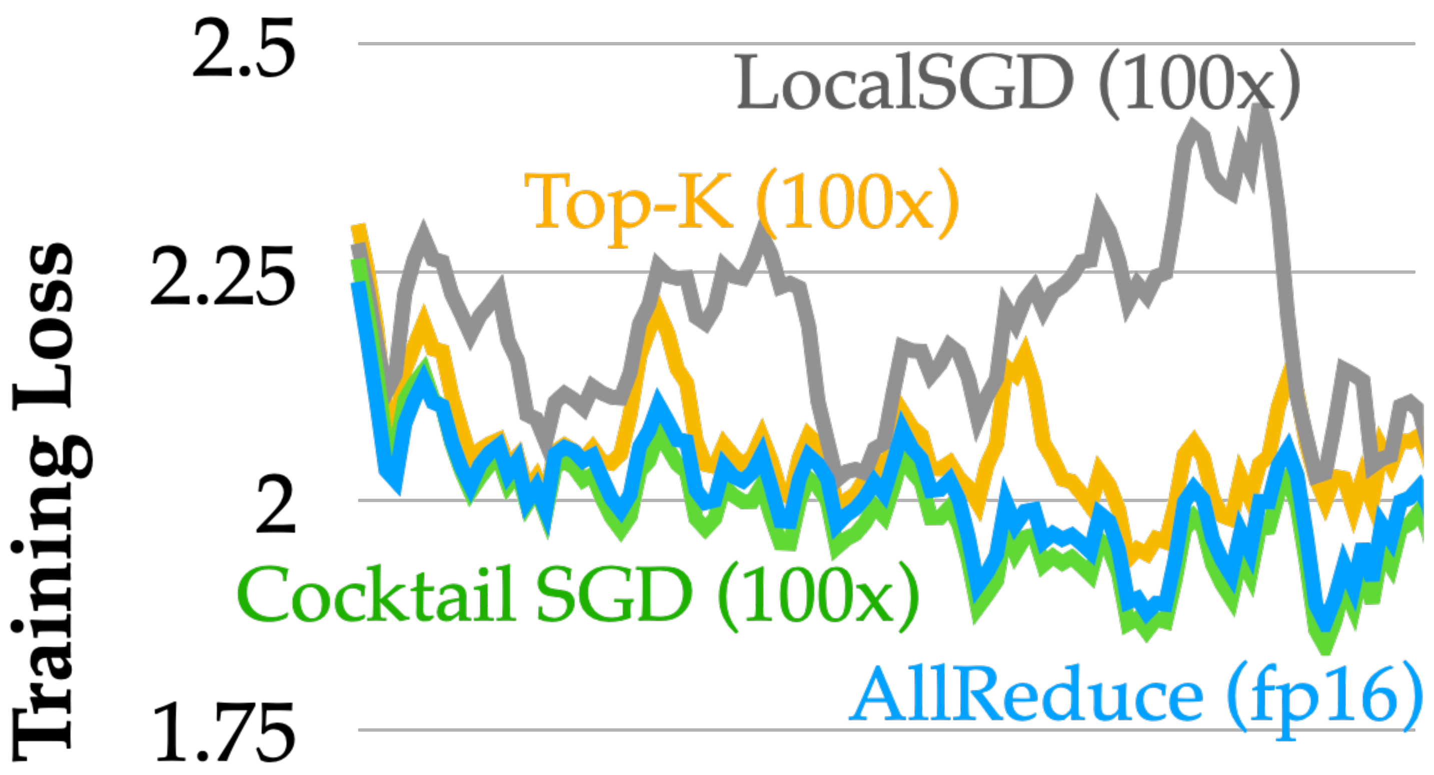
GPT-NeoX-20B Instruct Tuning

CocktailSGD: Mixture of Communication Compression Methods

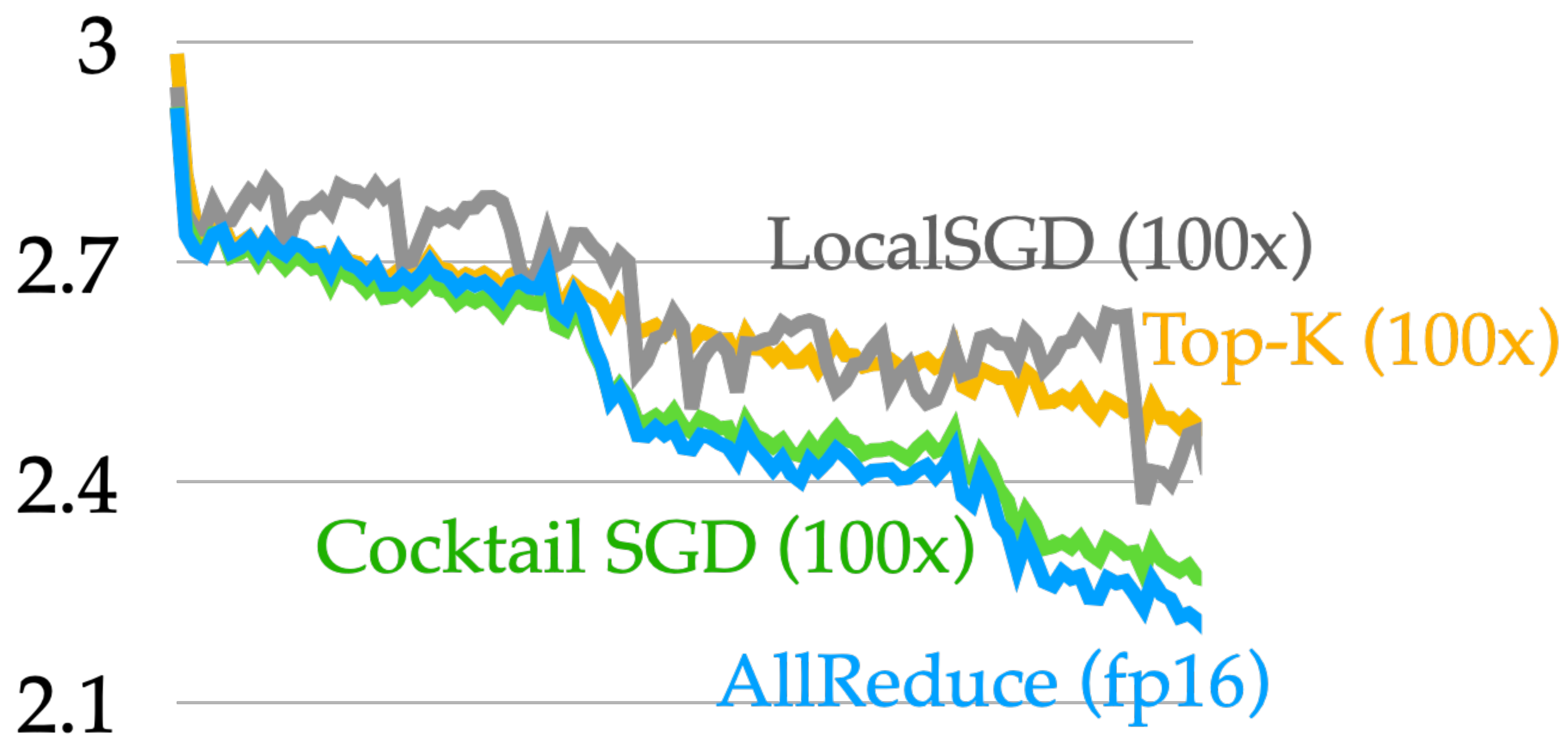


As long as Communication fully fills the Comm. Slot, no slow down caused by communication.

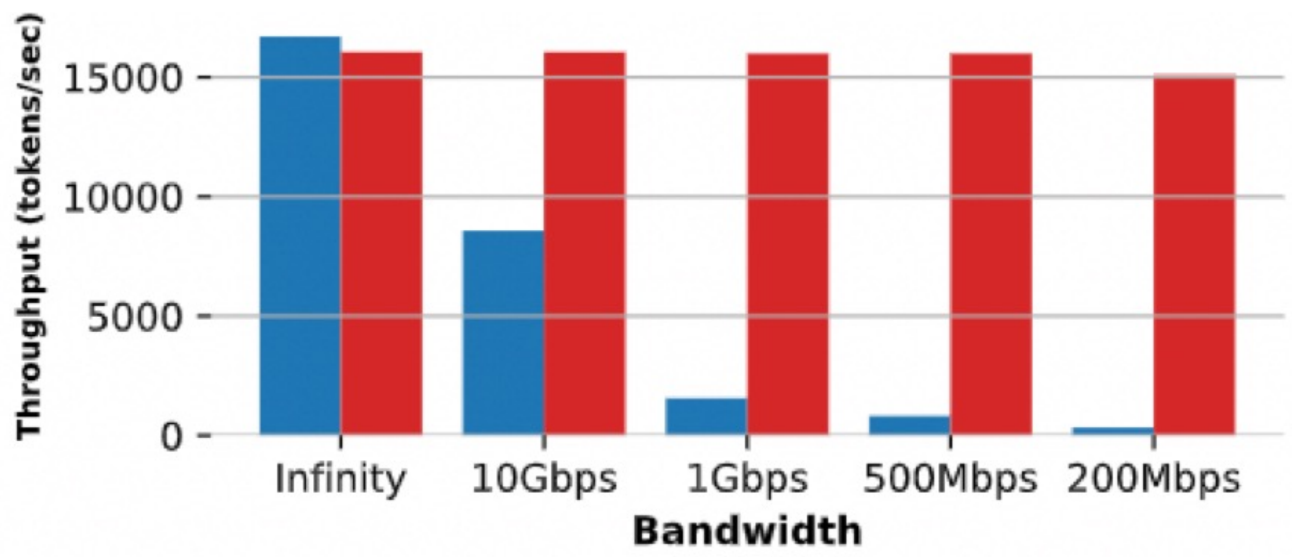
Different communication compression techniques complement each other and compose well!



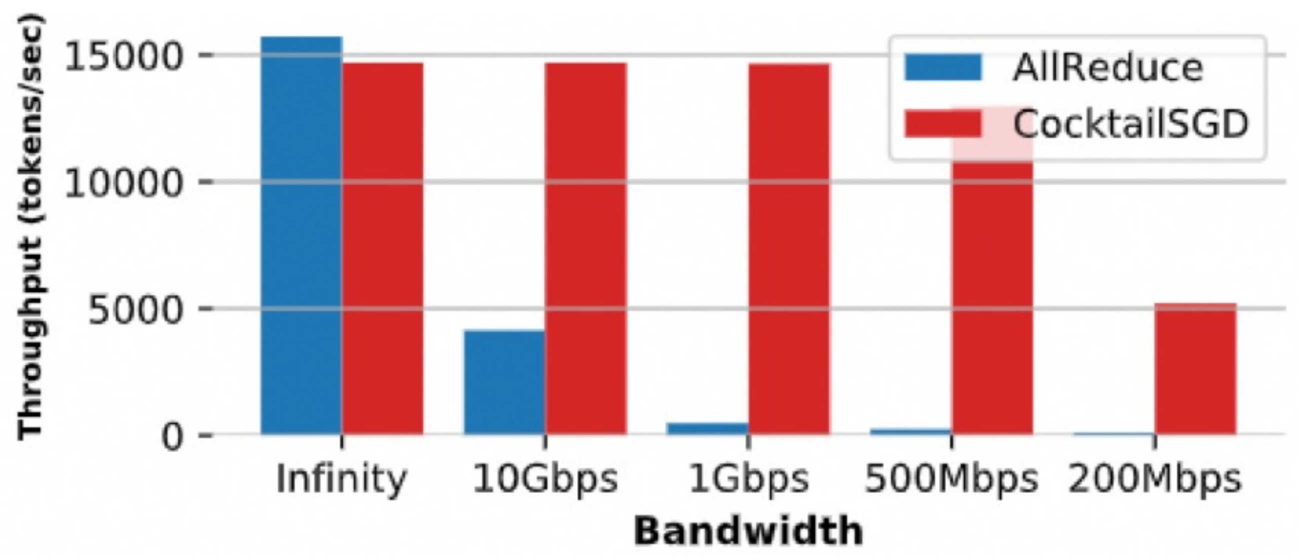
GPT-J-6B Instruct Tuning



GPT-NeoX-20B Instruct Tuning

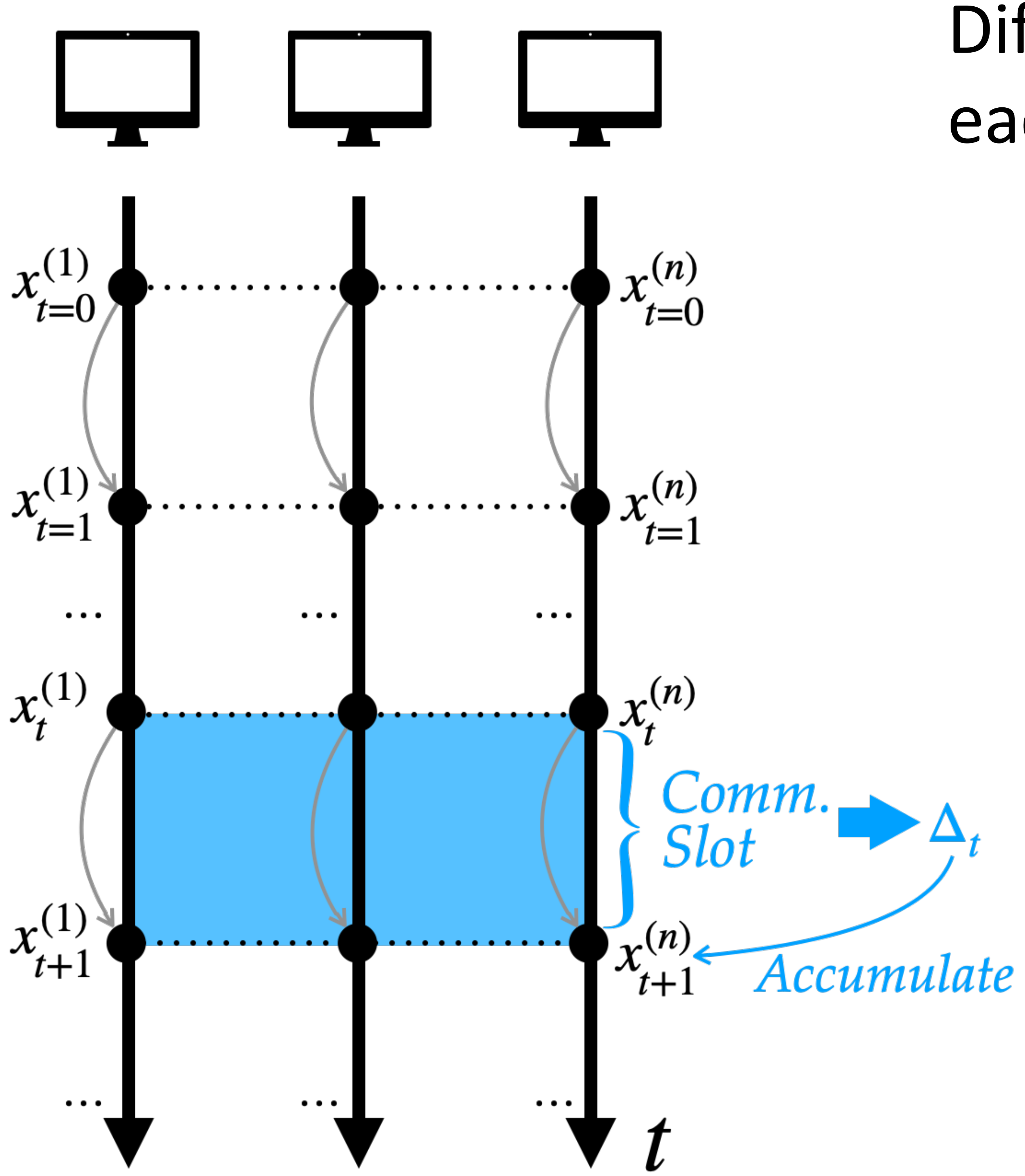


(b) GPT-J-6B



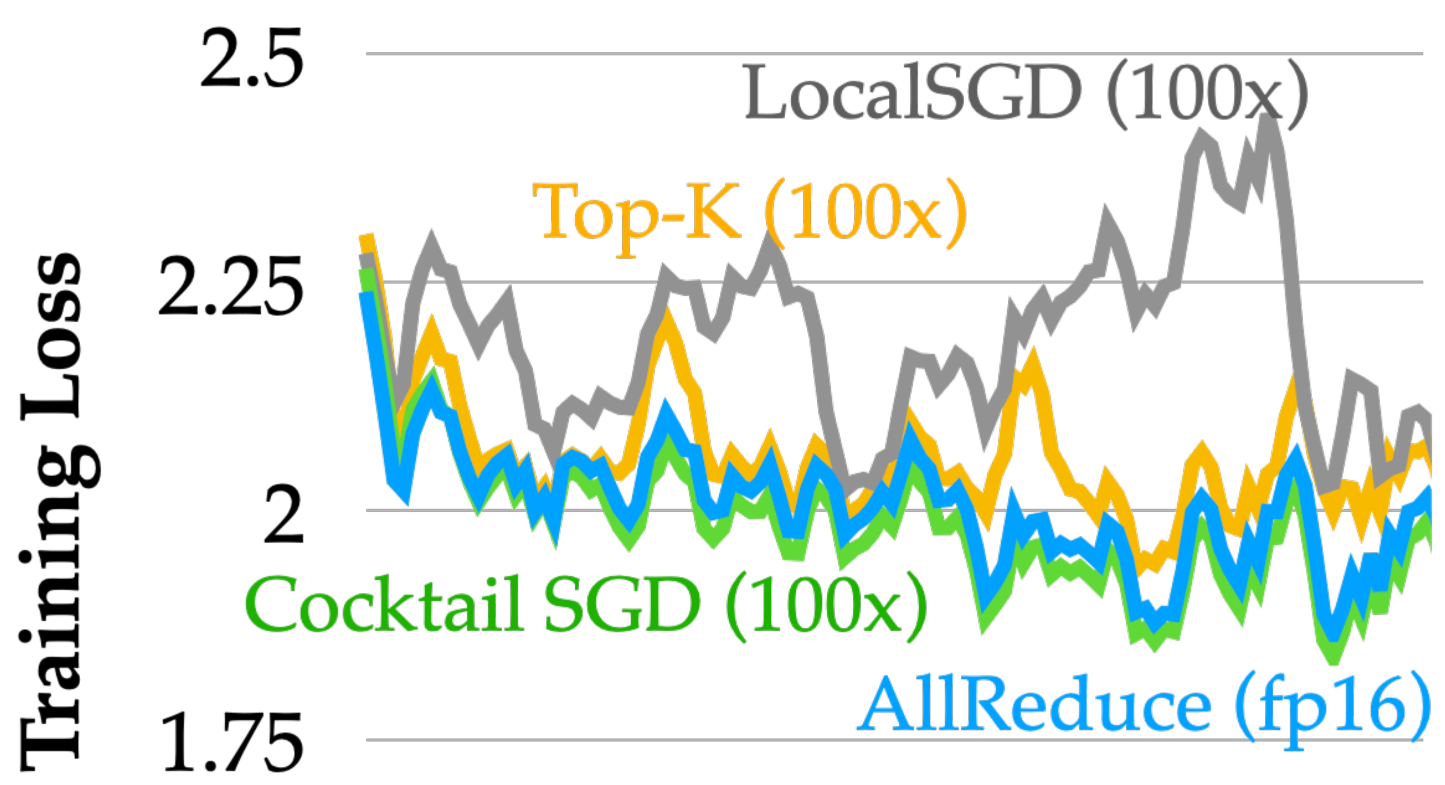
(c) GPT-NeoX-20B

CocktailSGD: Mixture of Communication Compression Methods

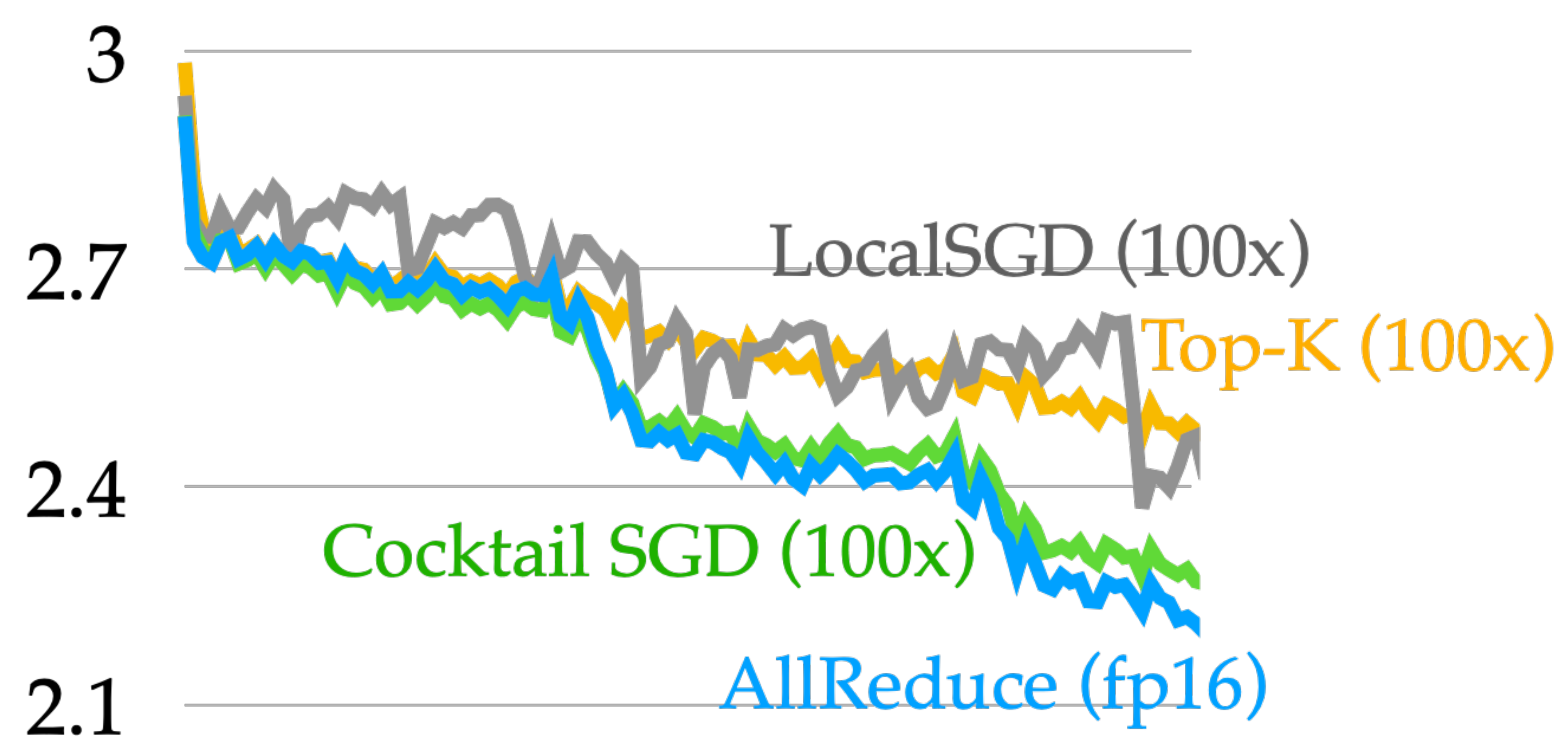


As long as Communication fully fills the Comm. Slot, no slow down caused by communication.

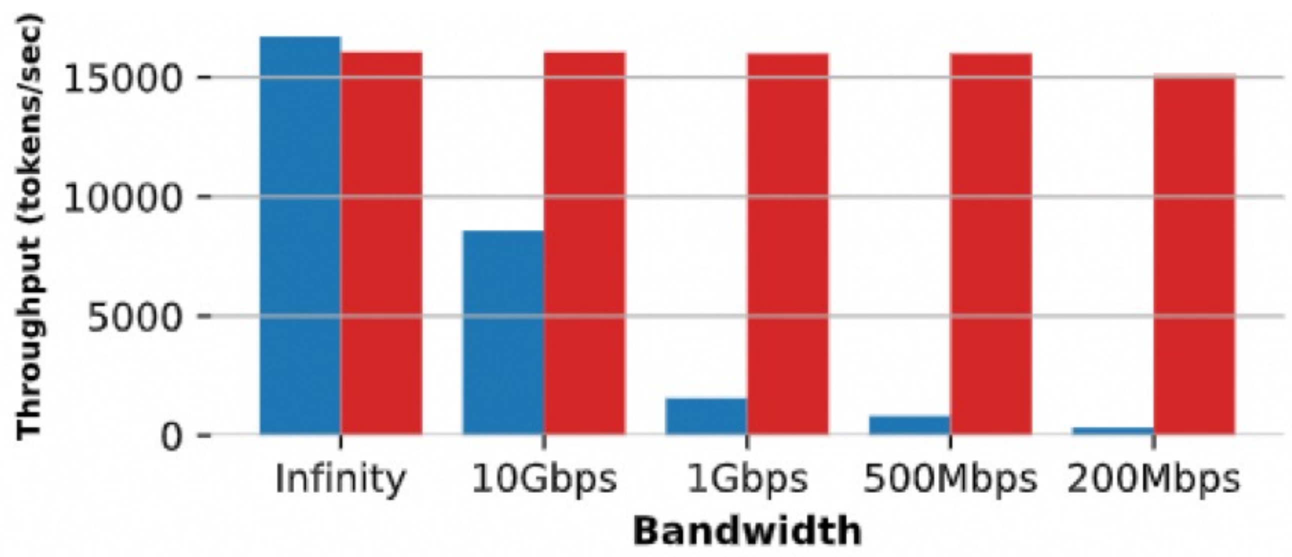
Different communication compression techniques complement each other and compose well!



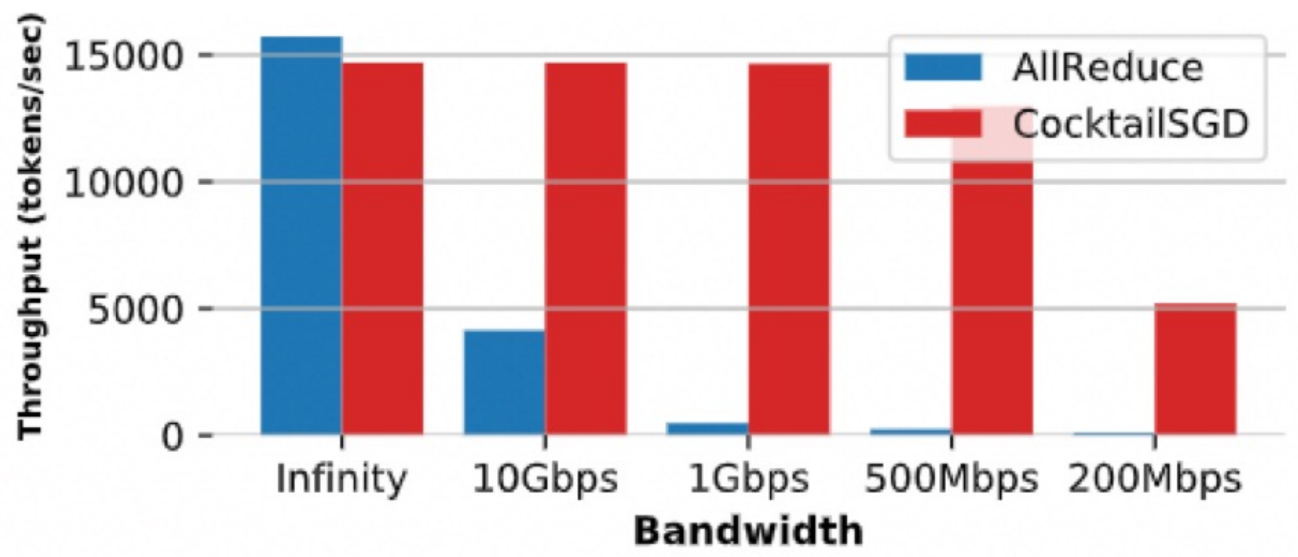
GPT-J-6B Instruct Tuning



GPT-NeoX-20B Instruct Tuning



(b) GPT-J-6B



(c) GPT-NeoX-20B

Data parallel over 1Gbps network!

Ongoing & Future Work: Optimizing Throughout the Stack

1. Different kinds of **hardware**

2. Efficient **algorithms** and kernels for training and inference

3. Diverse **capabilities** (long context) and new **applications** (multi-modal, genomics)