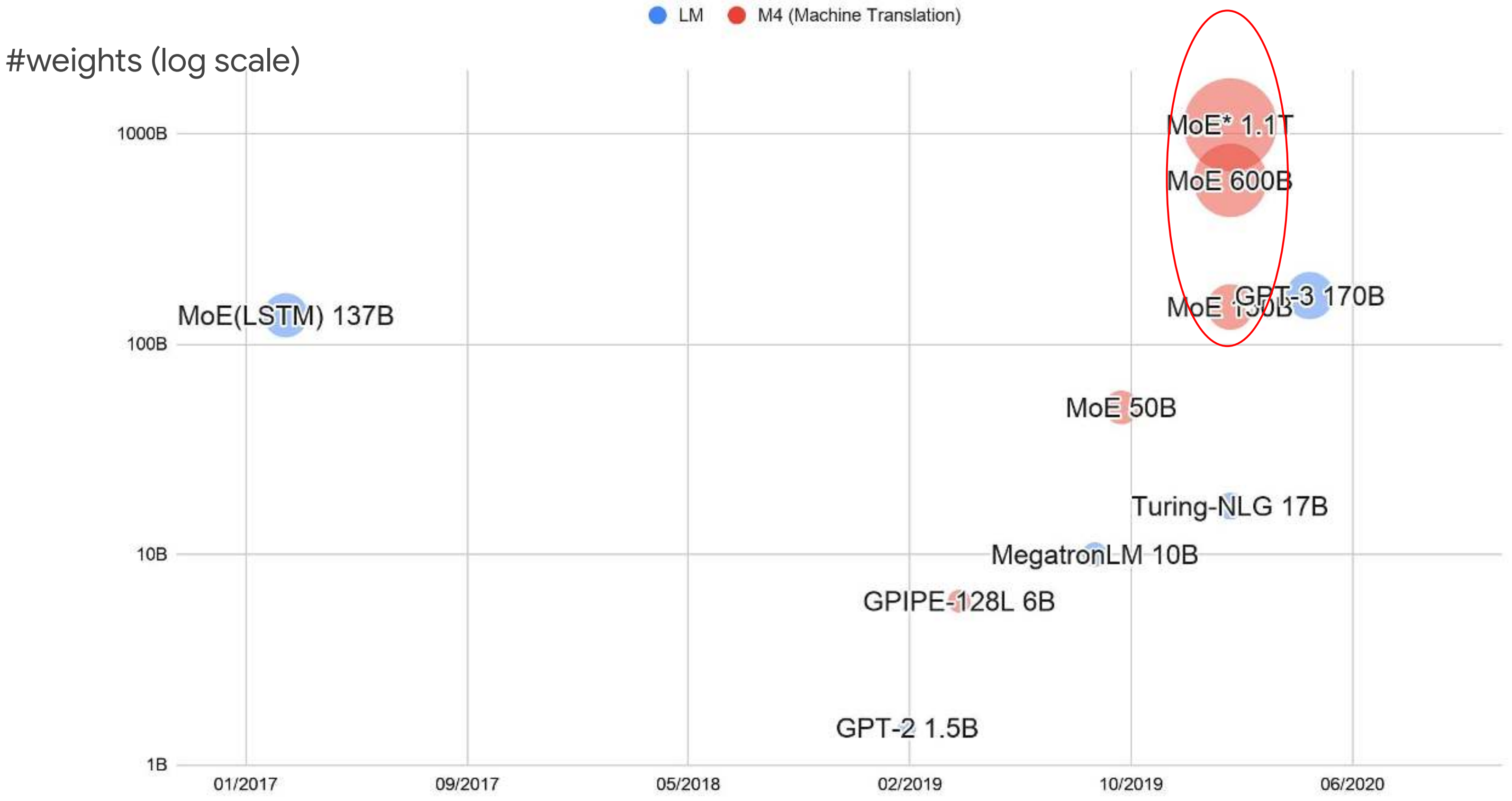
GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding

Zhifeng Chen
Google Inc.

Hot Chips 2020 Tutorial on “Machine Learning Scaleout”
08/16/2020

Big models



Google

Google Neural Machine Translation

Our goal

Develop a universal machine translation model (i.e. one model for all languages and domains)

"Perhaps the way [of translation] is to descend, from each language, down to the common base of human communication -- the real but as yet ***undiscovered universal language*** *-- and then re-emerge by whatever particular route is convenient."*

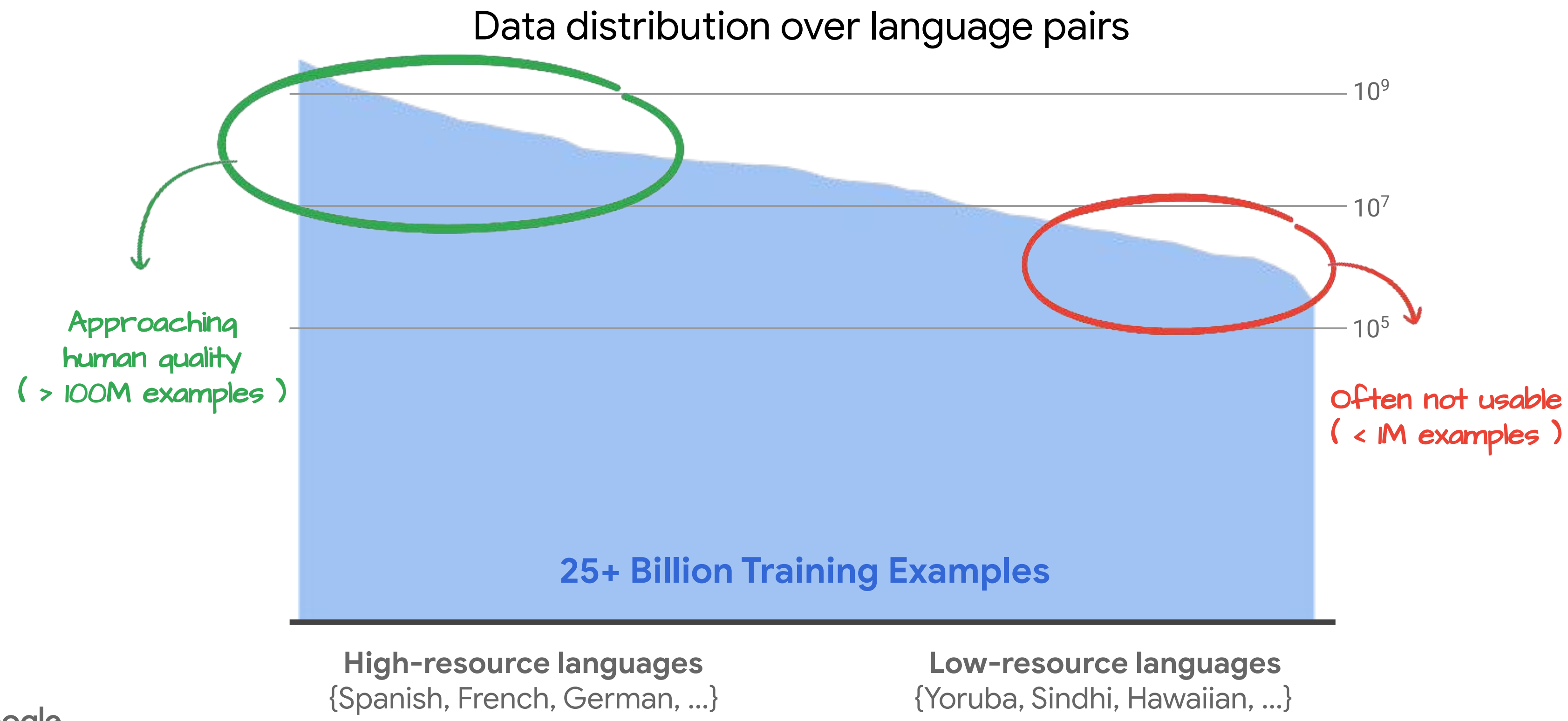
Warren Weaver (1949)

Google AI Blog

The latest news from Google AI

Exploring Massively Multilingual, Massive Neural Machine Translation

Motivation 1:
Improve translation quality for all language pairs



Google

Motivation 2:
Expand language coverage

In the world, there are...

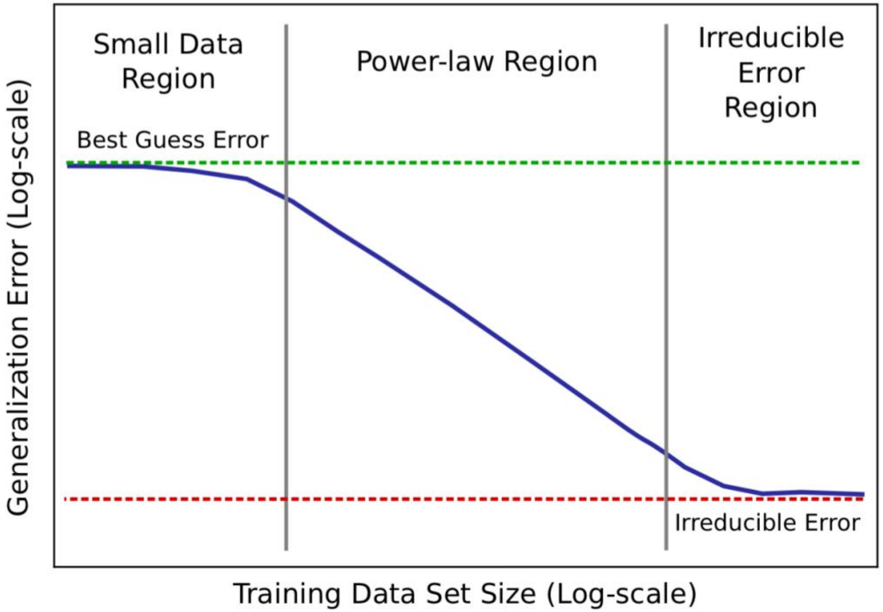
7,000+	2,000+	700+
Total languages	African languages	Native Am. languages[1]

But Translate only supports...

103	11	0
Total languages	African languages	Native Am. languages

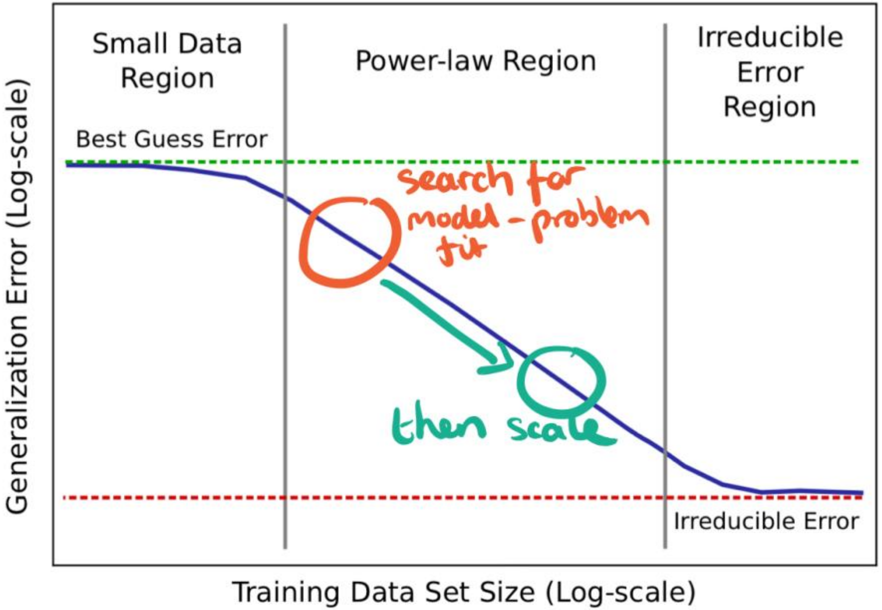
1.. Estimate is 766 Native Am languages globally: 383 in South America (source); 176 in central America (source); 207 in North America (source)

Motivation 3:
Neural network scaling and the new understanding of generalization



Hestness et al. 2018

Motivation 3:
Neural network scaling and the new understanding of generalization



Motivation 4:
This is a compelling test bed for ML research

Massive multilinguality requires advances in :

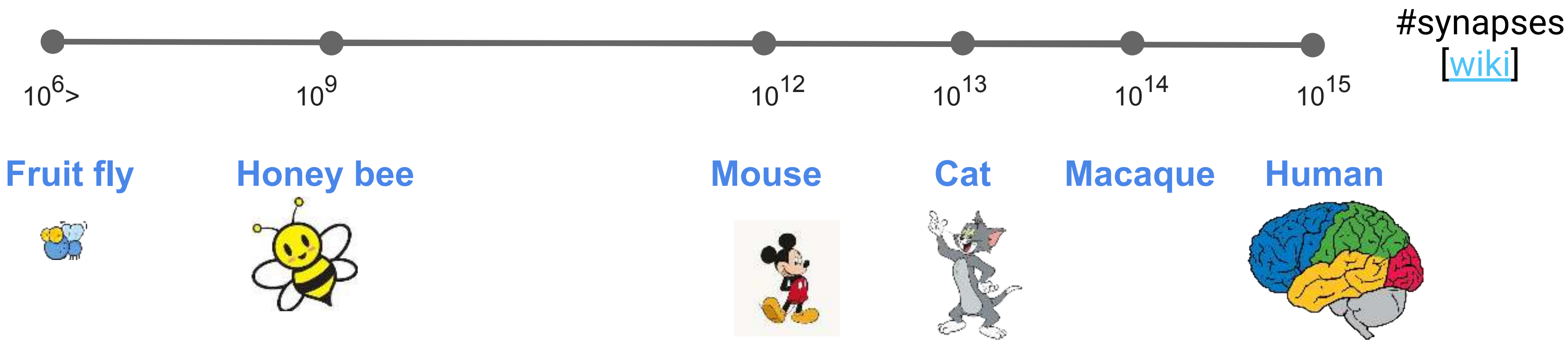
- Multi-task learning
- Meta-learning
- Continual learning

To achieve massive multilinguality, we need massive scale, requires advances in:

- **Model capacity**
- Trainability and optimization
- Efficiency improvements

Progress and Future

Number of Synapses



Number of Synapses

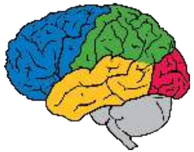
NMT with Attention
Resnet50
[25-50M]

#synapses
[wiki]

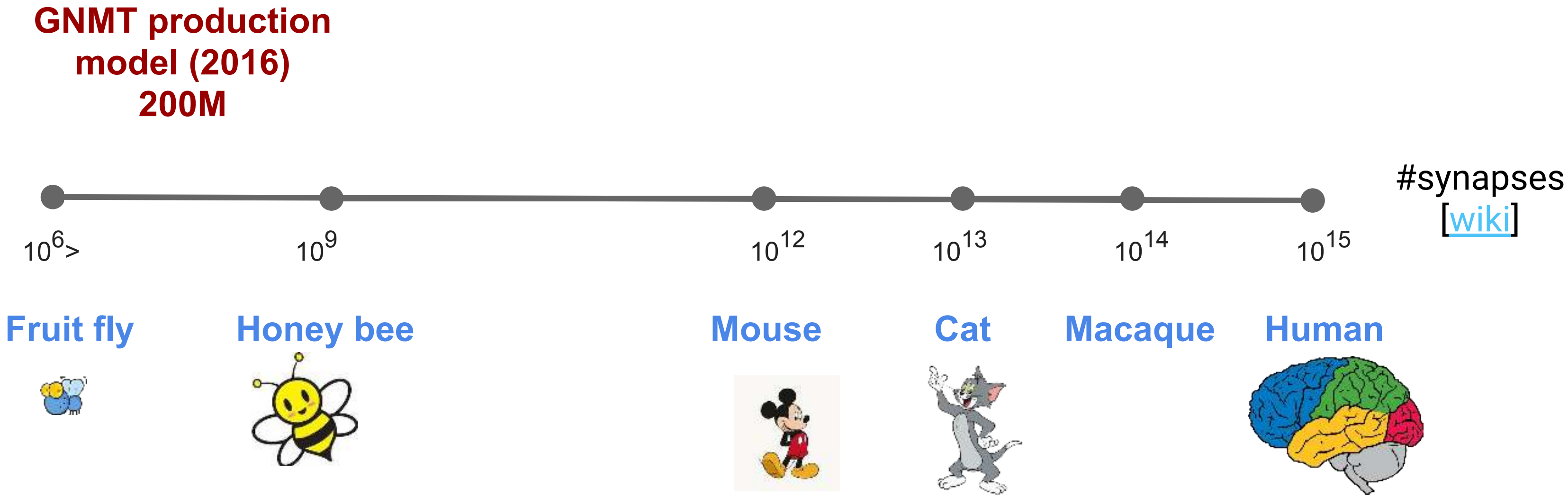
10^6>	10^9	10^{12}	10^{13}	10^{14}	10^{15}
Fruit fly	**Honey bee**	**Mouse**	**Cat**	**Macaque**	**Human**



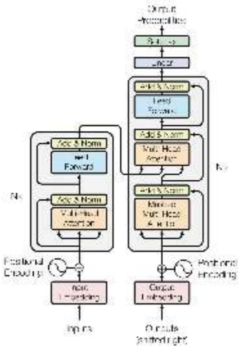




Number of Synapses



Number of Synapses



Transformer
[400M]

#synapses
[wiki]

10^6> | 10^9 | 10^{12} | 10^{13} | 10^{14} | 10^{15}

Fruit fly

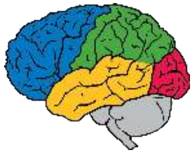
Honey bee

Mouse

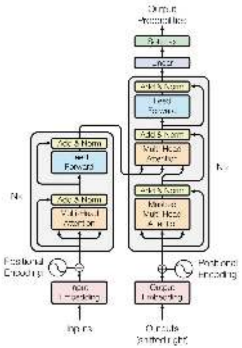
Cat

Macaque

Human



Number of Synapses



Transformer
[400M]

Facebook ResNeXt101
Open AI GPT2
MSR ZeRO
NVidia Megatron-LM
Google-T5
[1-10B]

M4:
Massively Multilingual Massive Machine Translation
[80B]

#synapses
[wiki]

10^6> | 10^9 | 10^{12} | 10^{13} | 10^{14} | 10^{15}

Fruit fly

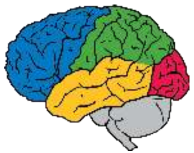
Honey bee

Mouse

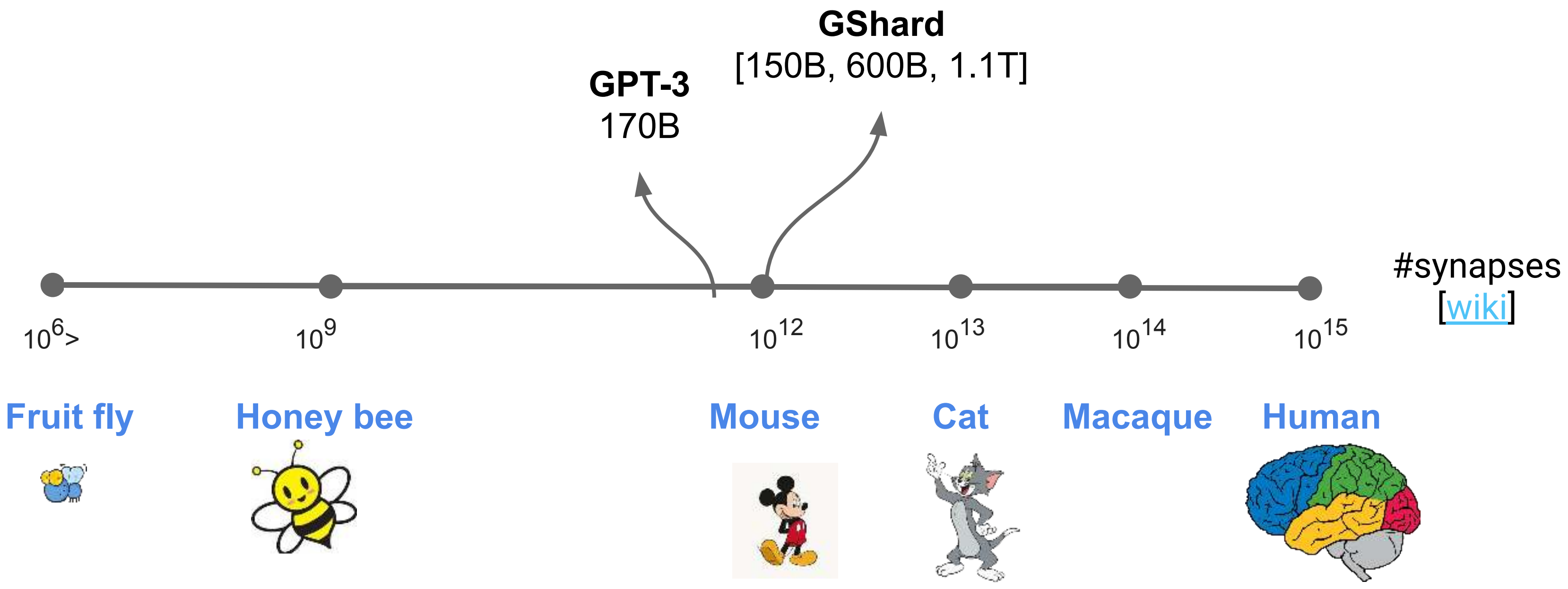
Cat

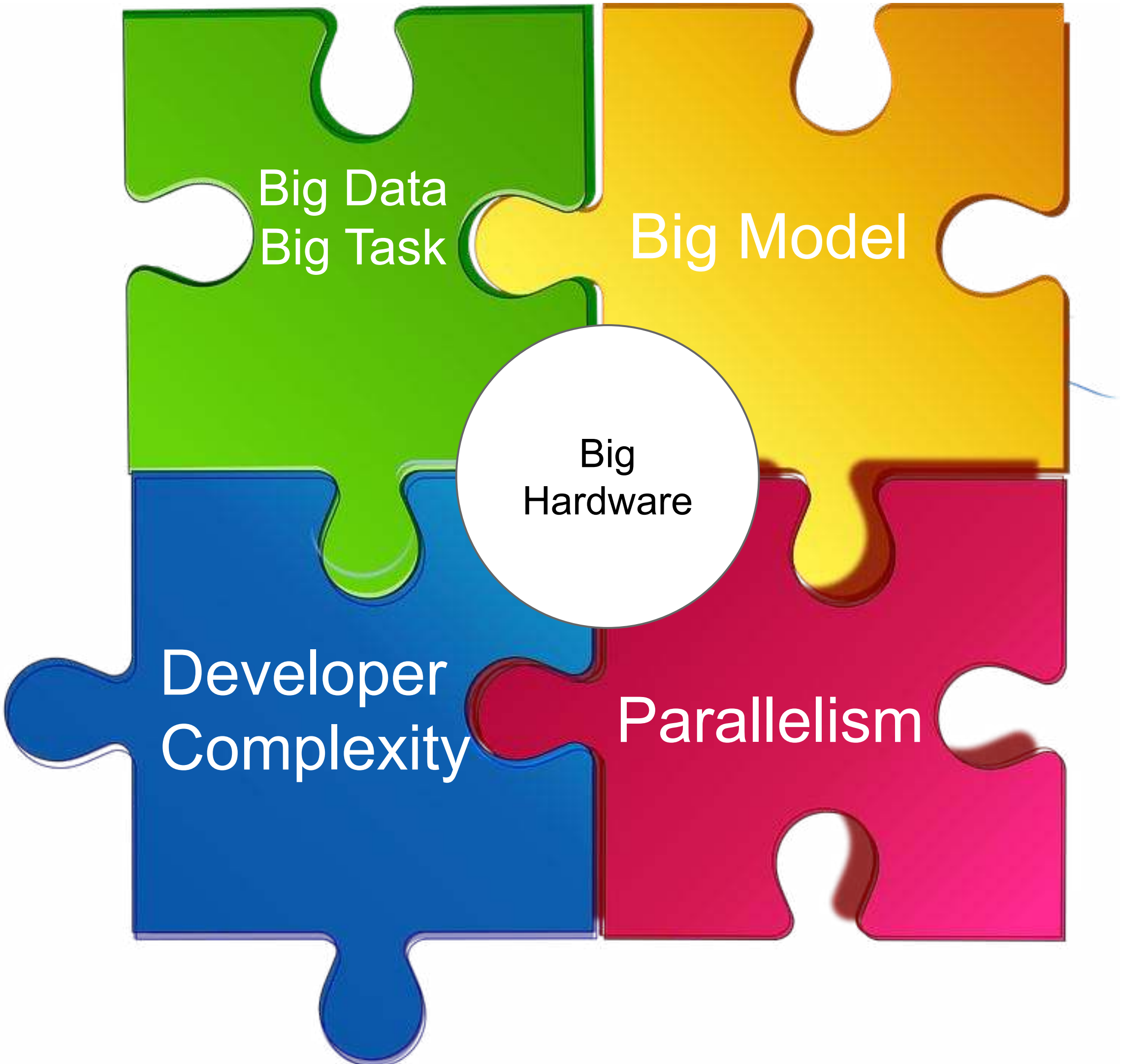
Macaque

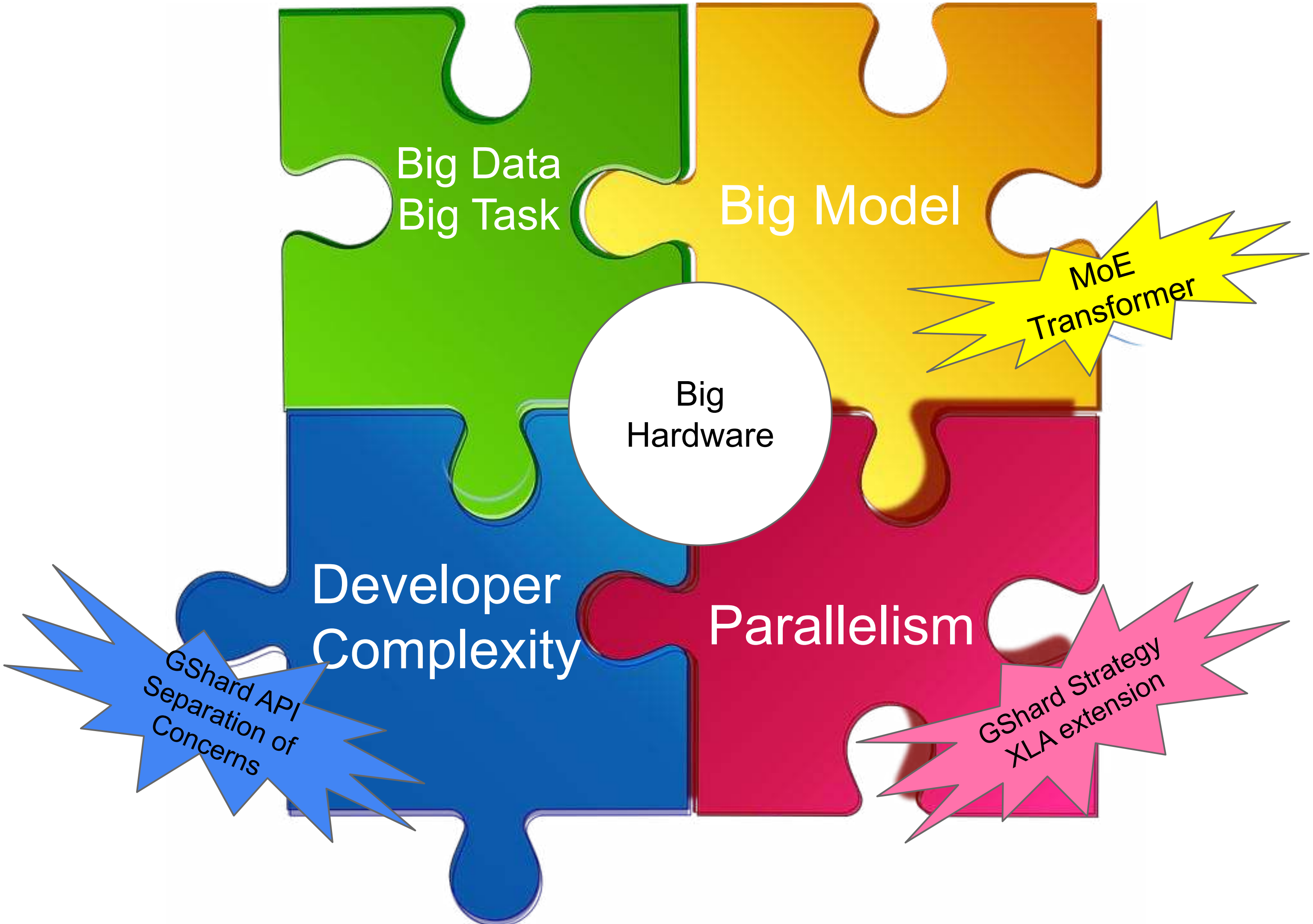
Human



Number of Synapses



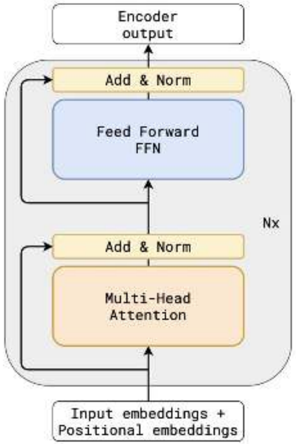

Big Data
Big Task
Big Model
Big
Hardware
Developer
Complexity
Parallelism


Big Data
Big Task
Big Model
MoE
Transformer
Big
Hardware
Developer
Complexity
Parallelism
GShard API
Separation of
Concerns
GShard Strategy
XLA extension

Google

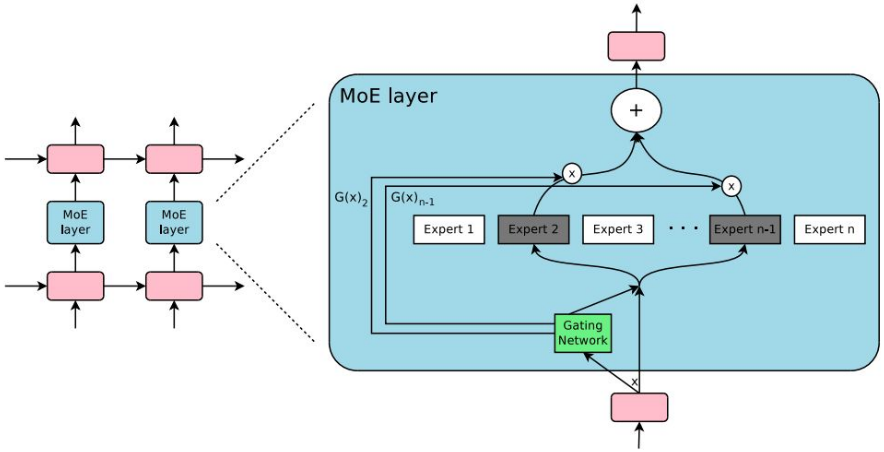
Transformer

Transfomer
Encoder



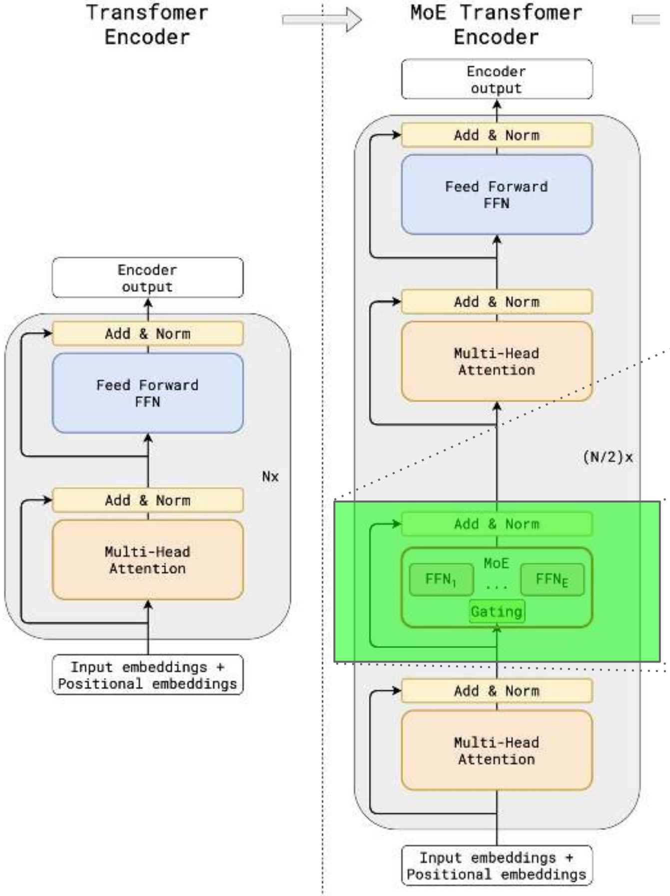
- Powerful
  - Cores to many SOTA results.
- Simple
  - Easy to express in linear algebra.
  - Reproduced many many times.
- Originally proposed in the [paper]

Mixture of Experts (MoE)

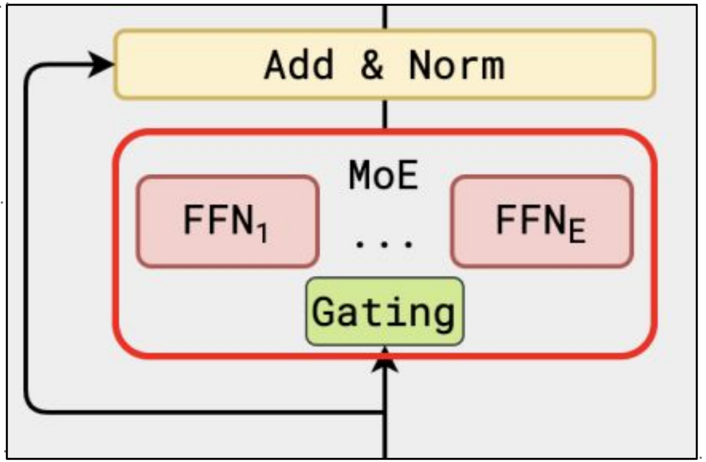


- Sparsely gated
  - Cost-effective inference
- Embarrassingly parallelizable
  - Nice to accelerators
- Originally proposed in this [paper]

Mixture-of-Experts Transformer



Replace every-other FFN with an MoE layer

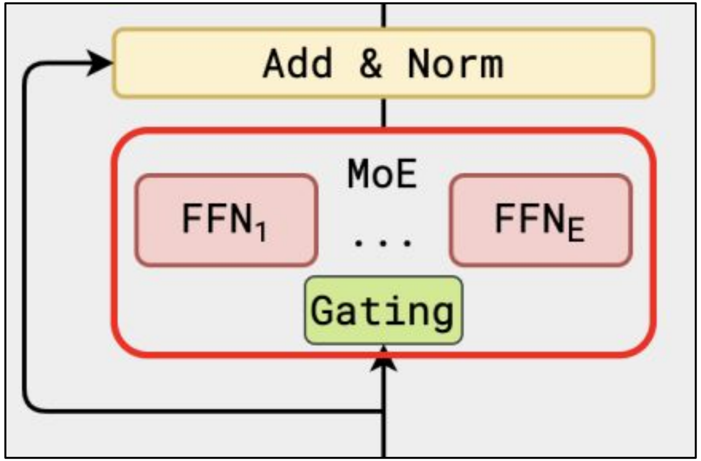


Position-wise Mixture-of-Experts Layer

x_s is the input token

$$\mathcal{G}_{s,E} = \mathrm{GATE}(x_s)$$
$$\mathrm{FFN}_e(x_s) = wo_e \cdot \mathrm{ReLU}(wi_e \cdot x_s)$$
$$y_s = \sum_{e=1}^{E} \mathcal{G}_{s,e} \cdot \mathrm{FFN}_e(x_s)$$

E feed-forward networks $\mathrm{FFN}_1 \ldots \mathrm{FFN}_E$

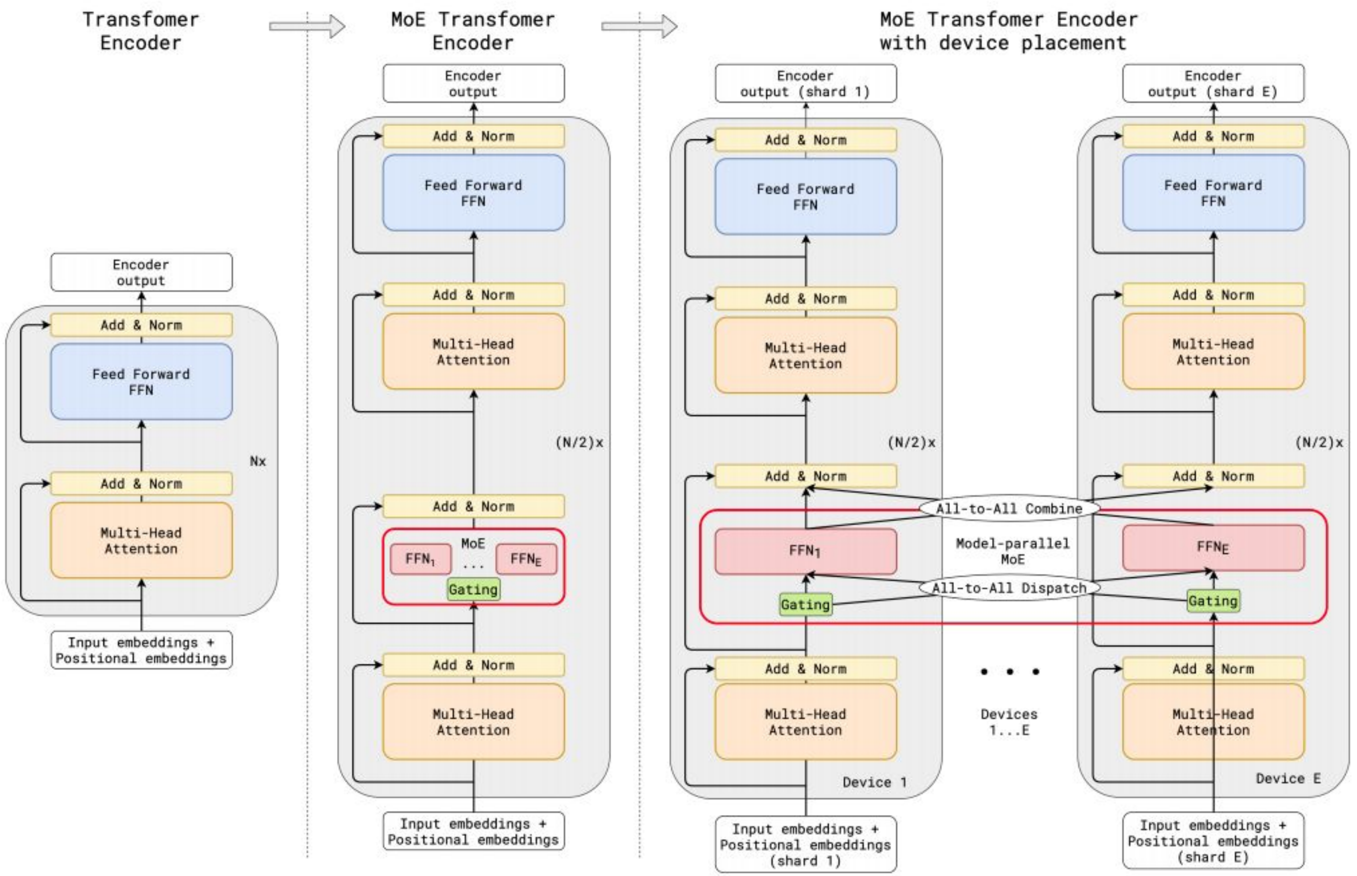


$\mathcal{G}_{s,E}$ is computed by a gating network.
y_s, is the weighted average

Google Research

Algorithm details

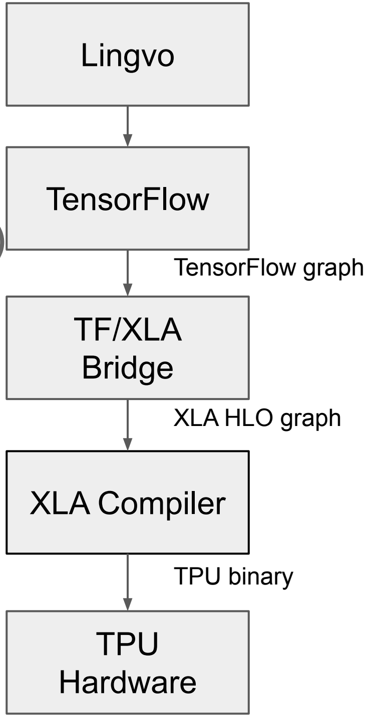
- **Gate function written in linear algebra**
  - Easy to express in a sequential program
- **Experts load balancing during training**
  - Auxiliary loss helps
- **Uniform routing during warming up phase**
- **Random second expert dispatch**
- **Flat beam search for inference**



Transfomer Encoder
MoE Transfomer Encoder
MoE Transfomer Encoder with device placement
Encoder output
Add & Norm
Feed Forward FFN
Multi-Head Attention
Nx
Input embeddings + Positional embeddings
(N/2)x
MoE
FFN_1
...
FFN_E
Gating
Encoder output (shard 1)
Encoder output (shard E)
All-to-All Combine
Model-parallel MoE
All-to-All Dispatch
Devices 1...E
Device 1
Device E
Input embeddings + Positional embeddings (shard 1)
Input embeddings + Positional embeddings (shard E)

GShard Overview

- User ***partially*** annotates tensors in the TensorFlow graph
  - Called sharding annotations
  - Specifies which dimension to split, across how many devices
  - Specifies which tensor to be replicated
- The annotation is delivered to the XLA compiler (as HLO graph)
- XLA Propagates user-provided sharding information to the rest of the computation with some heuristics
  - Users are not required to annotate every tensor in the computation. Just need to give enough hints that the compiler can do this propagation well.
  - Weights need to be annotated if they need to be partitioned (default is replicate)
- Transforms each op in the XLA graph, following the determined input/output sharding decision



tf.einsum recap

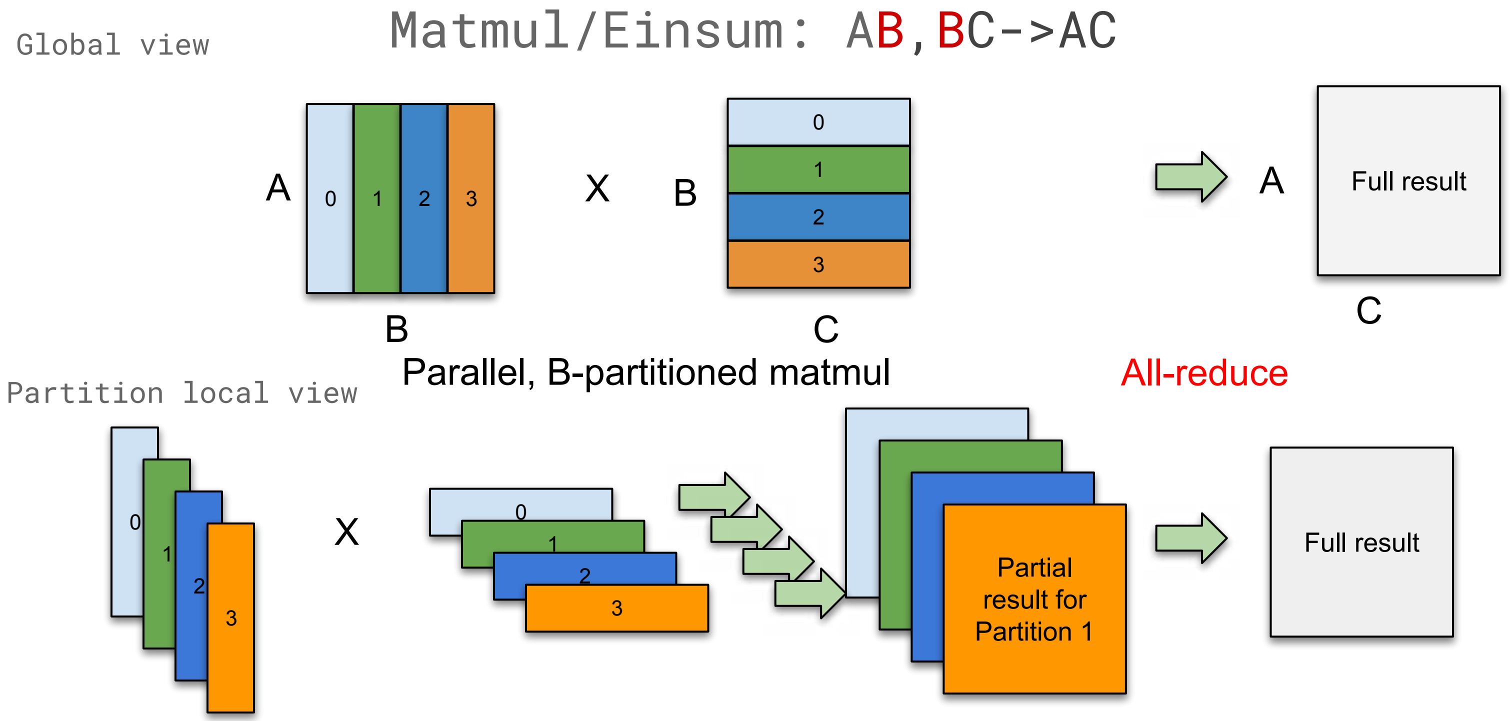
```
# Matrix multiplication
einsum('ij,jk->ik', m0, m1)

# Transpose
einsum('ij->ji', m)
```

User Annotation

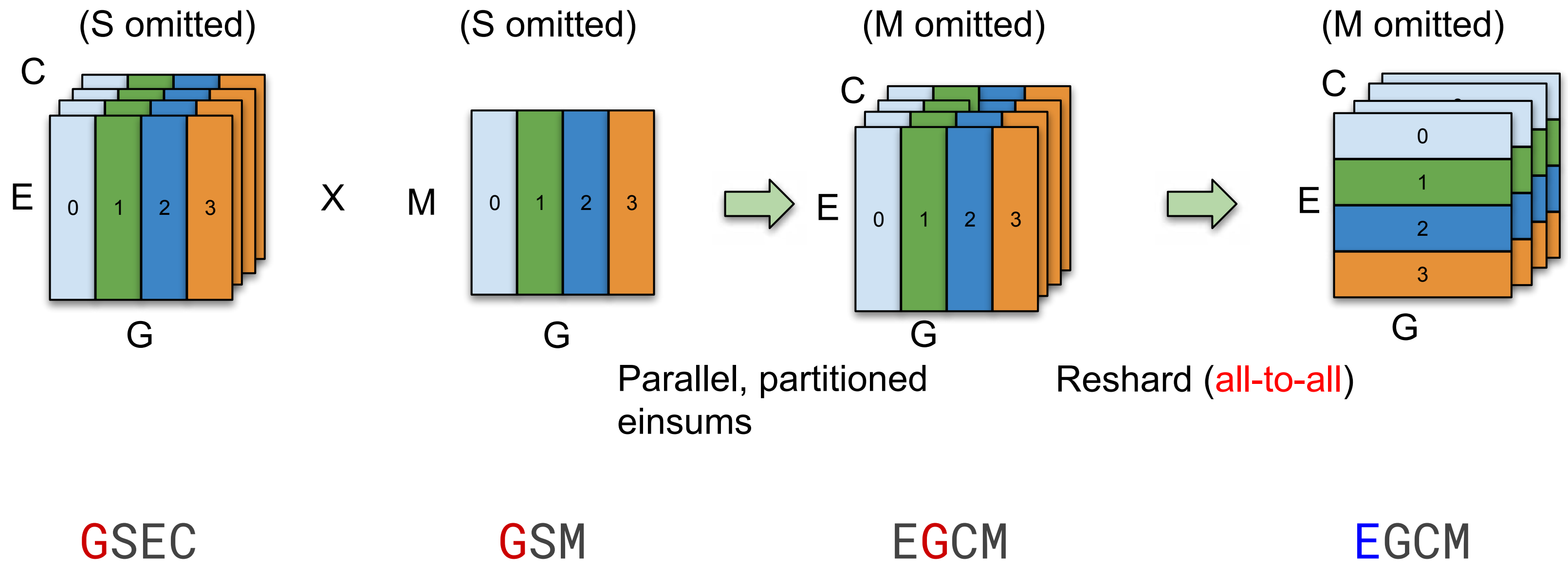
```
  # Partition inputs along group (G) dim.
+ inputs = split(inputs, 0, D)
  # Replicate the gating weights
+ wg = replicate(wg)
  gates = softmax(einsum("GSM,ME->GSE", inputs, wg))
  combine_weights, dispatch_mask = Top2Gating(gates)
  dispatched_expert_inputs = einsum(
    "GSEC,GSM->EGCM", dispatch_mask, reshaped_inputs)
  # Partition dispatched inputs along expert (E) dim.
+ dispatched_expert_inputs = split(dispatched_expert_inputs, 0, D)
  h = einsum("EGCM,EMH->EGCH", dispatched_expert_inputs, wi)
  ...
```

Einsum Sharding Example



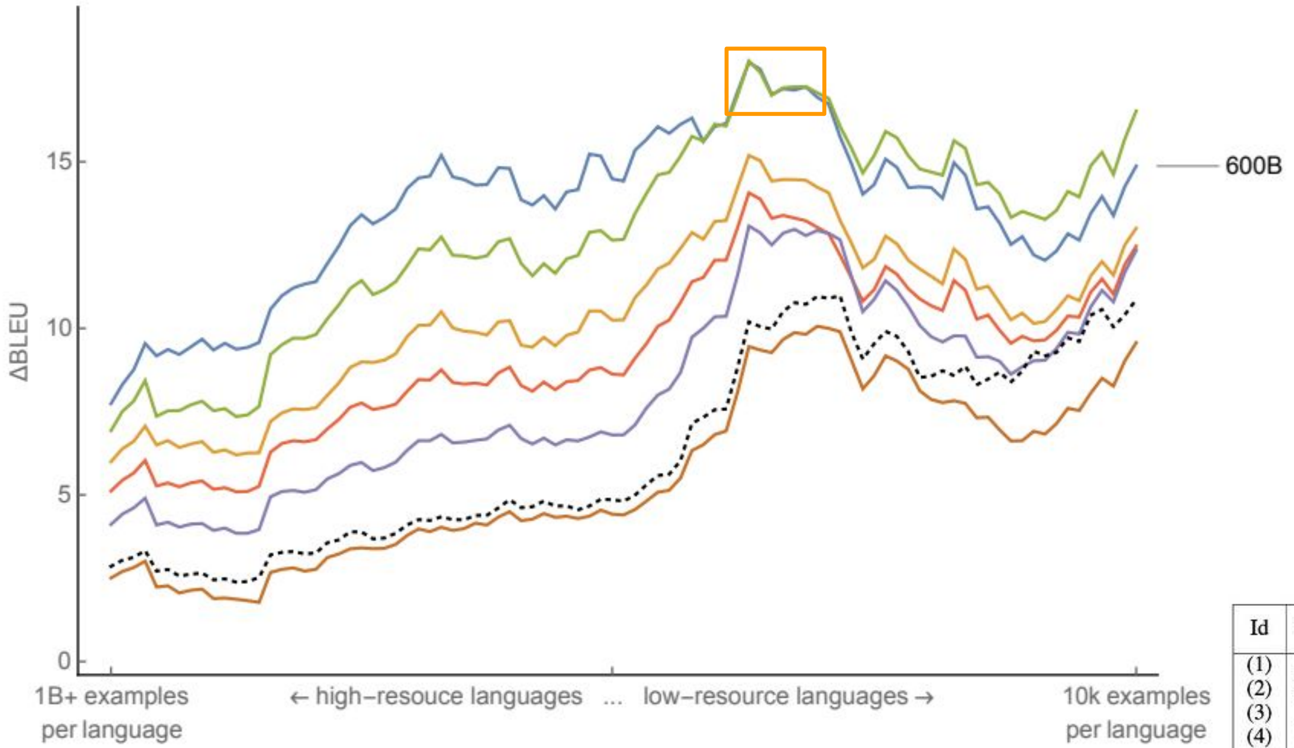
Einsum Sharding Example

Einsum: GSEC,GSM->EGCM



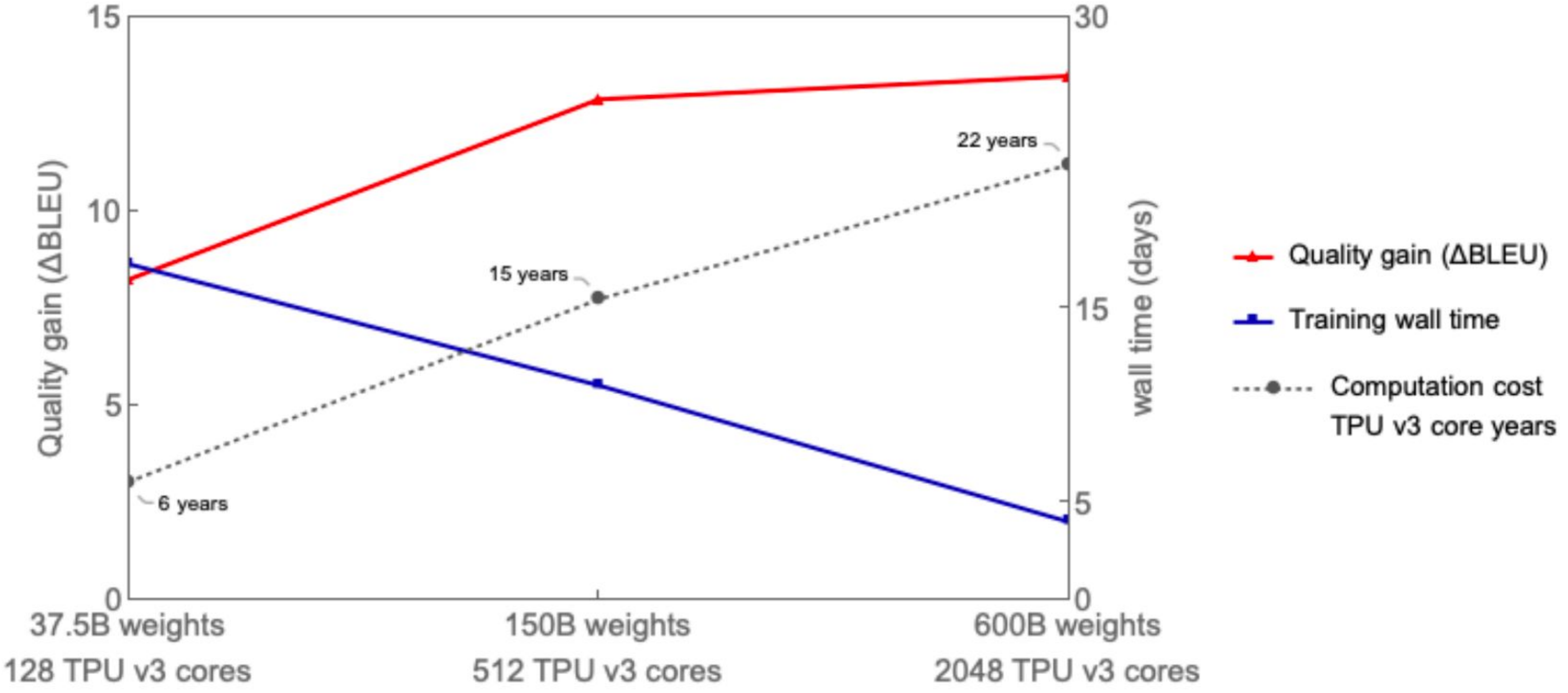
M4 ΔBLEU

Google Research



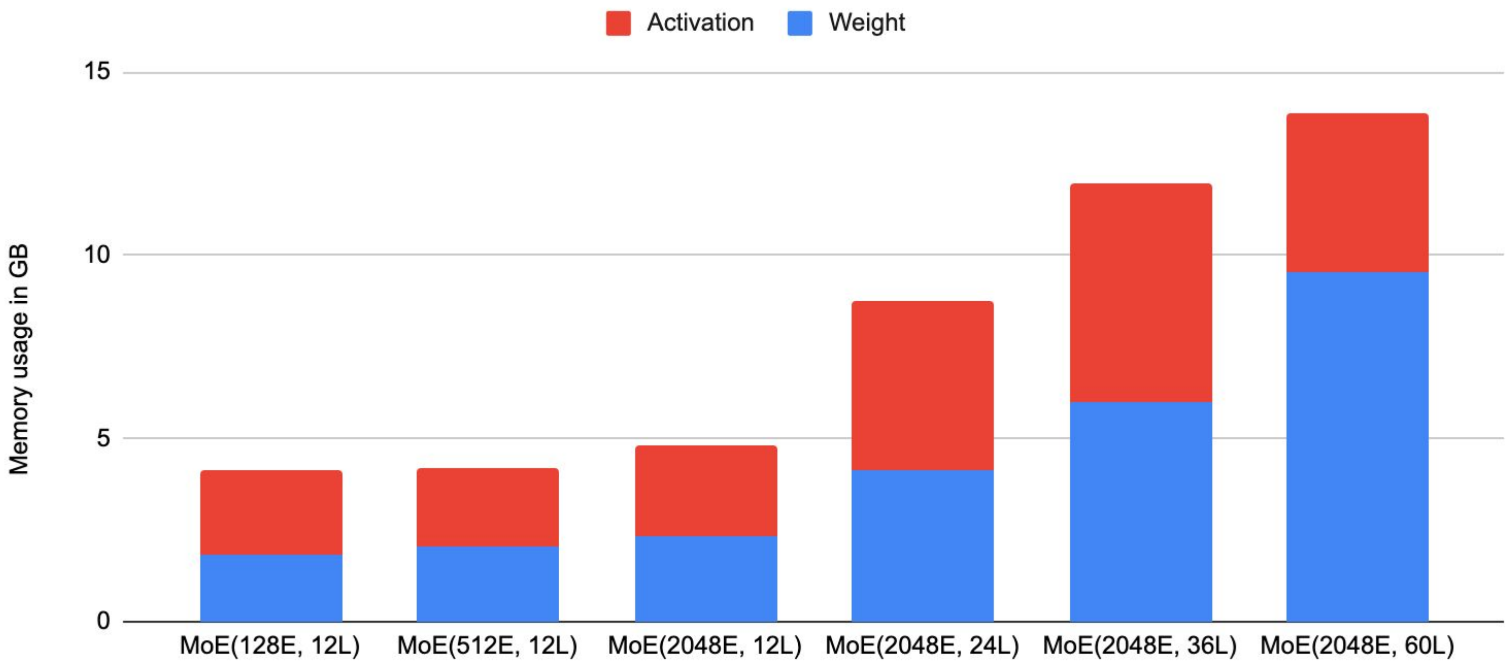
Id	Model	BLEU avg.	ΔBLEU avg.	Weights
(1)	MoE(2048E, 36L)	**44.3**	13.5	600B
(2)	MoE(2048E, 12L)	41.3	10.5	200B
(3)	MoE(512E, 36L)	43.7	12.9	150B
(4)	MoE(512E, 12L)	40.0	9.2	50B
(5)	MoE(128E, 36L)	39.0	8.2	37B
(6)	MoE(128E, 12L)	36.7	5.9	12.5B
*	T(96L)	36.9	6.1	2.3B
*	Baselines	30.8	-	100×0.4B

Quality vs. Cost



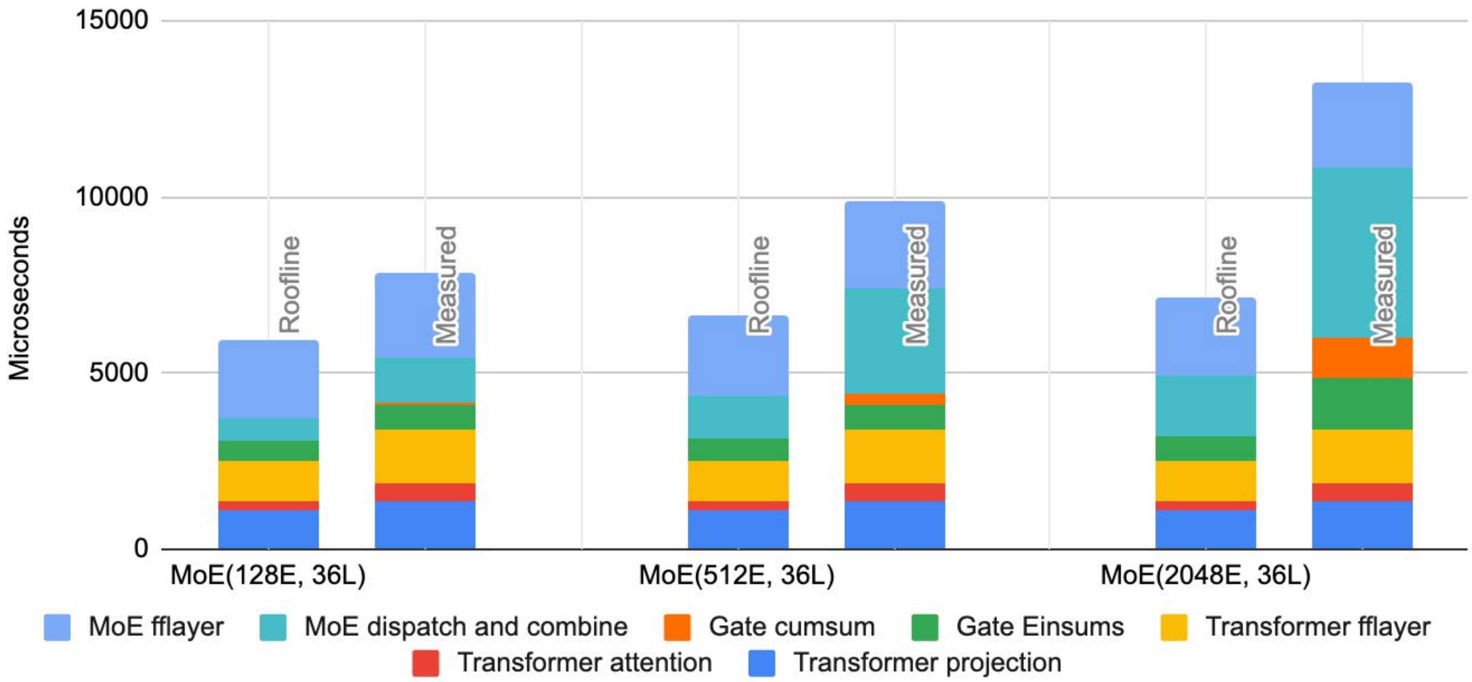
Google Research

HBM Profile



Google Research

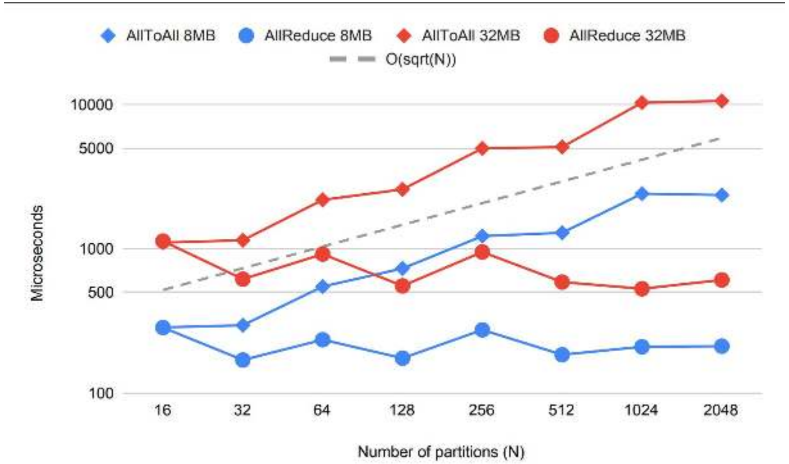
Execution Time Breakdown



Google

Communication Primitives for Sharding

- **AllReduce**: Performs elementwise reduction (e.g., summation) over the inputs from all participants
- **AllGather**: Concatenates tensors from all participants following a specified order
- **AllToAll**: Each participant splits its input of along a dimension, then sends each piece to all other participants. On receiving data pieces from others, each participant concatenates the pieces
- **Collective-Permute**: Given a list of source-destination pairs, the input data of a source device is sent to the corresponding destination device



Conclusion

- Giant neural networks are awesome.
- Mixture-of-expert makes giant nets cost effective.
- Simple API makes building such networks feasible.

Thank you

- [Paper]
- [Code]