# Mixture of Experts (MoE)

Yang You

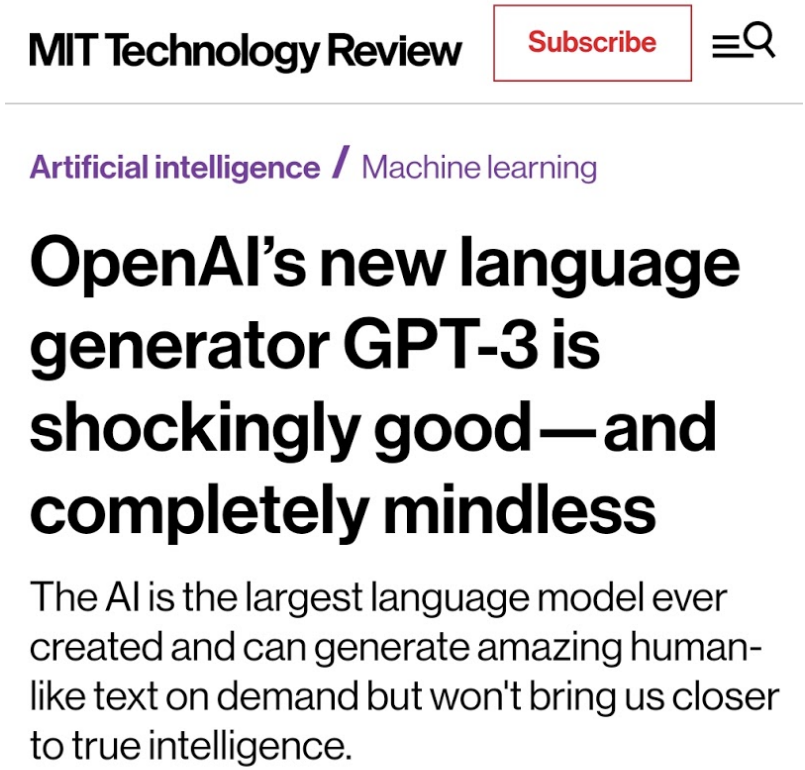
Presidential Young Professor at NUS Computer Science

ai.comp.nus.edu.sg

# Outline

- **Introduction**
- Sparsely-Gated MoE for LSTM
- Review on Transformers
- Sparsely-Gated MoE for Transformers

# GPT-3 shocked the world!



MIT Technology Review Subscribe

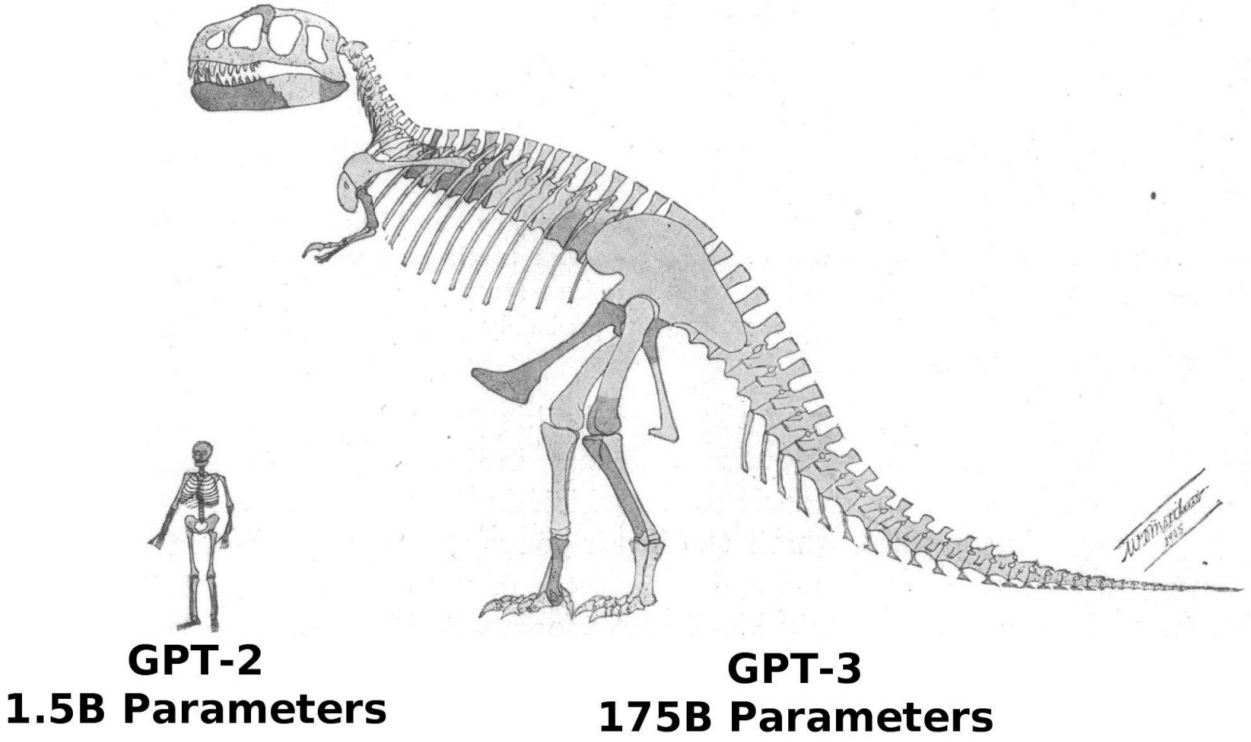
Artificial intelligence / Machine learning

## OpenAI's new language generator GPT-3 is shockingly good—and completely mindless

The AI is the largest language model ever created and can generate amazing human-like text on demand but won't bring us closer to true intelligence.

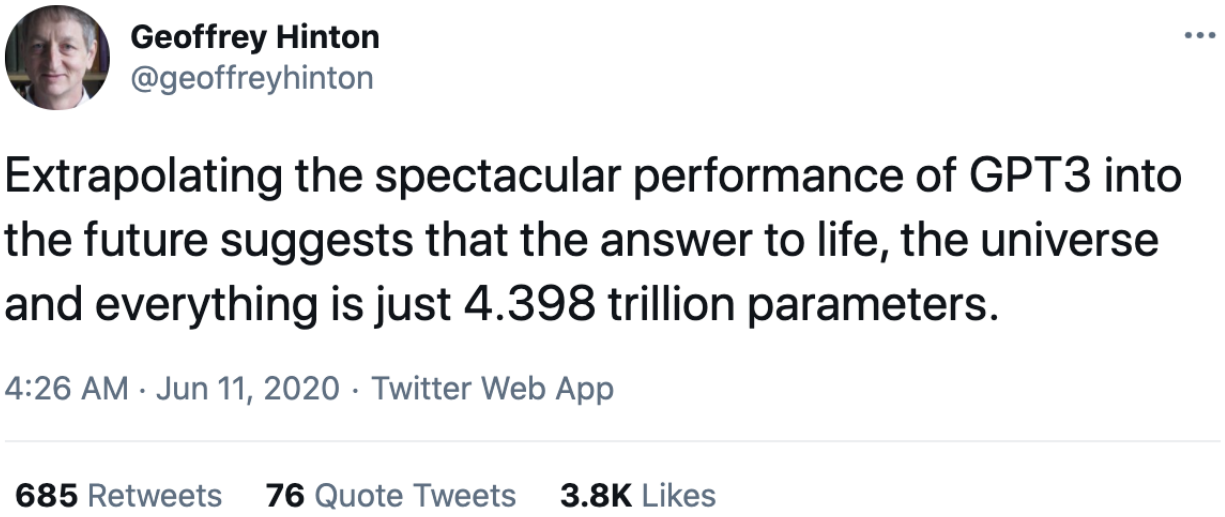
- https://www.technologyreview.com/2020/07/20/1005454/openai-machine-learning-language-generator-gpt-3-nlp/

# Why GPT-3 is so powerful?




- It finds a way to efficiently learn from a huge number of parameters

# Is it possible to train a model with trillions of parameters?



**Geoffrey Hinton**
@geoffreyhinton

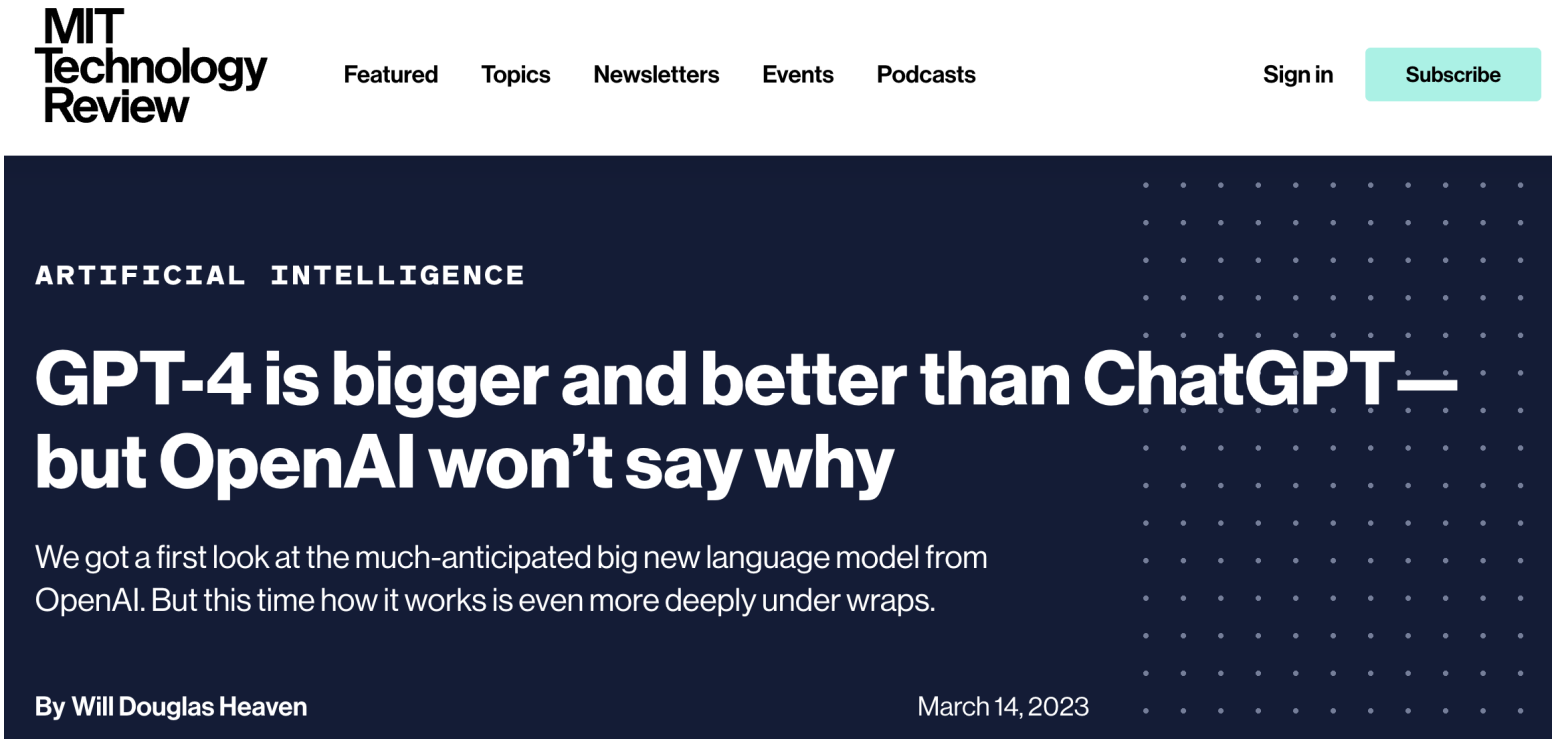
Extrapolating the spectacular performance of GPT3 into the future suggests that the answer to life, the universe and everything is just 4.398 trillion parameters.

4:26 AM · Jun 11, 2020 · Twitter Web App

**685** Retweets **76** Quote Tweets **3.8K** Likes

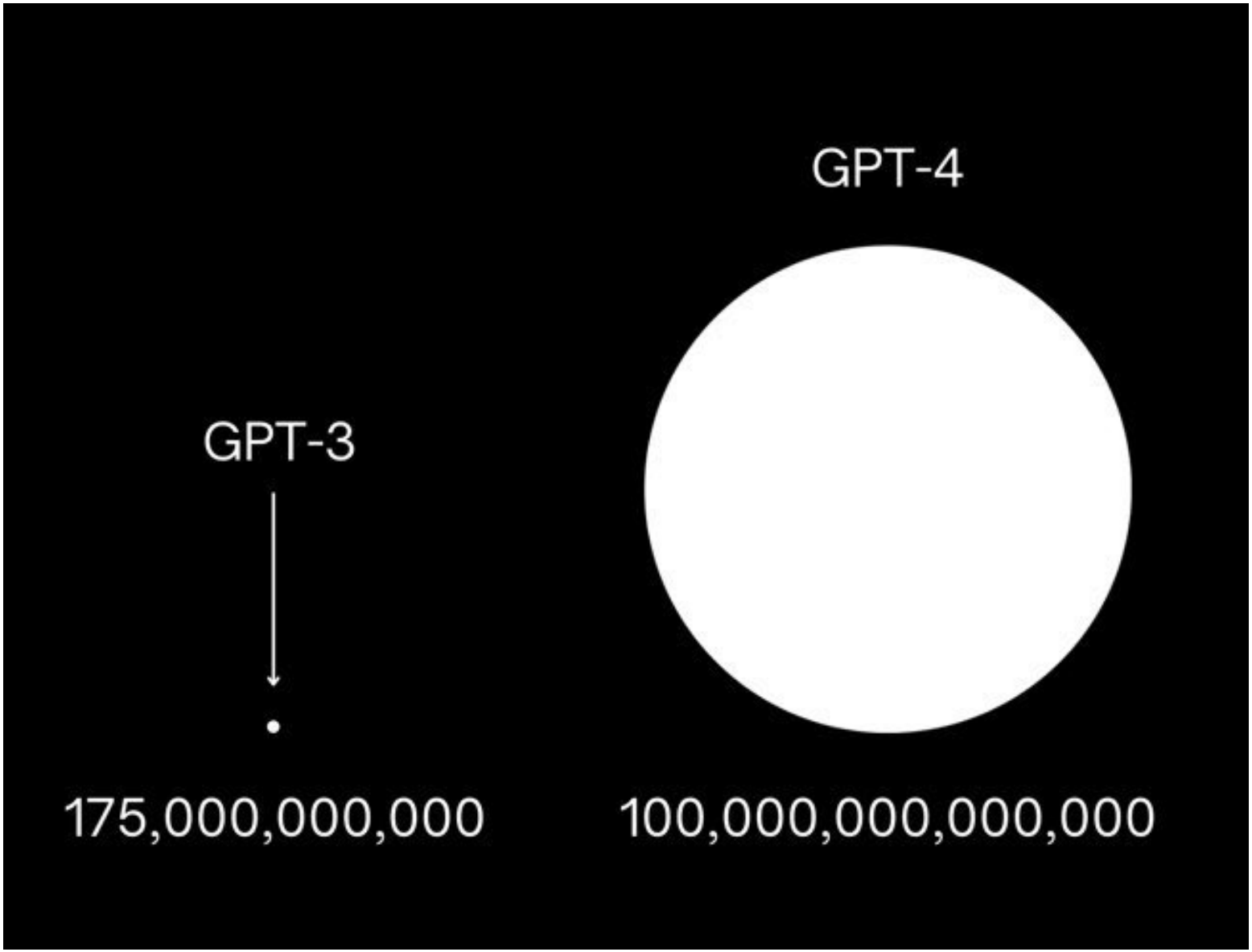
- How can we continue scaling up the model efficiently?

# GPT-4 shocked the world again!

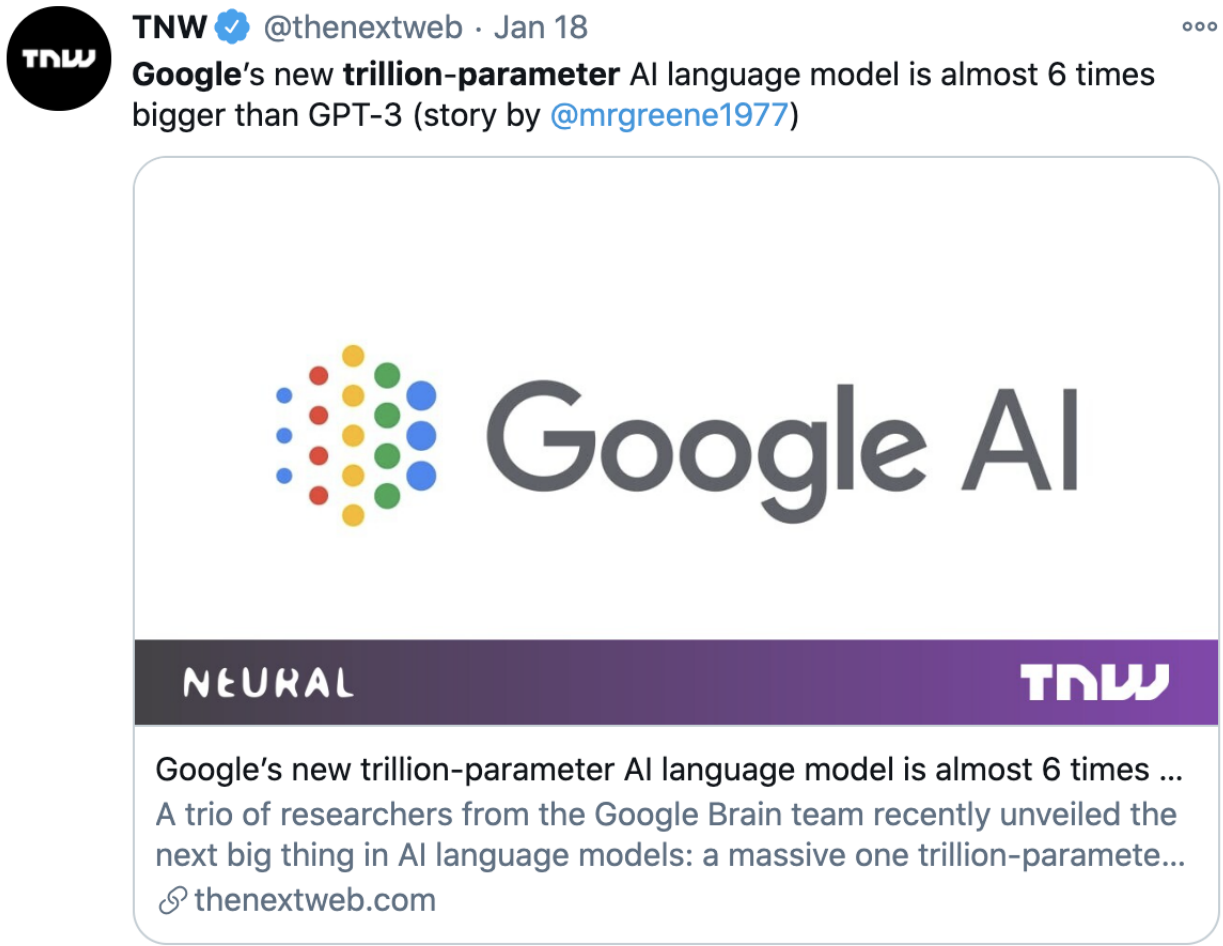



- https://www.technologyreview.com/2023/03/14/1069823/gpt-4-is-bigger-and-better-chatgpt-openai/

# GPT-4 may have 100 Trillion Parameters



# Is there another direction?




- Switch-Transformer is the first neural network model reaching trillions of parameters (1.6T)!

# Is there another direction?

AI Business

FINANCE HEALTH & PHARMA IT & DATA CENTER MANUFACTURING MARKETING & ECOMMERCE RETAIL & SUPPLY


AI PRACTITIONER

LinkedIn Twitter Facebook Email More 4

## Google Brain unveils trillion-parameter AI language model, the largest yet

*by Louis Stone* 1/15/2021


*Currently more of a research project than a commercial product*

Google Brain has developed an artificial intelligence language model with some 1.6 trillion parameters.

That puts it at nine times the size of the OpenAI's 175 billion parameter GPT-3, previously considered to be the world's largest language model.

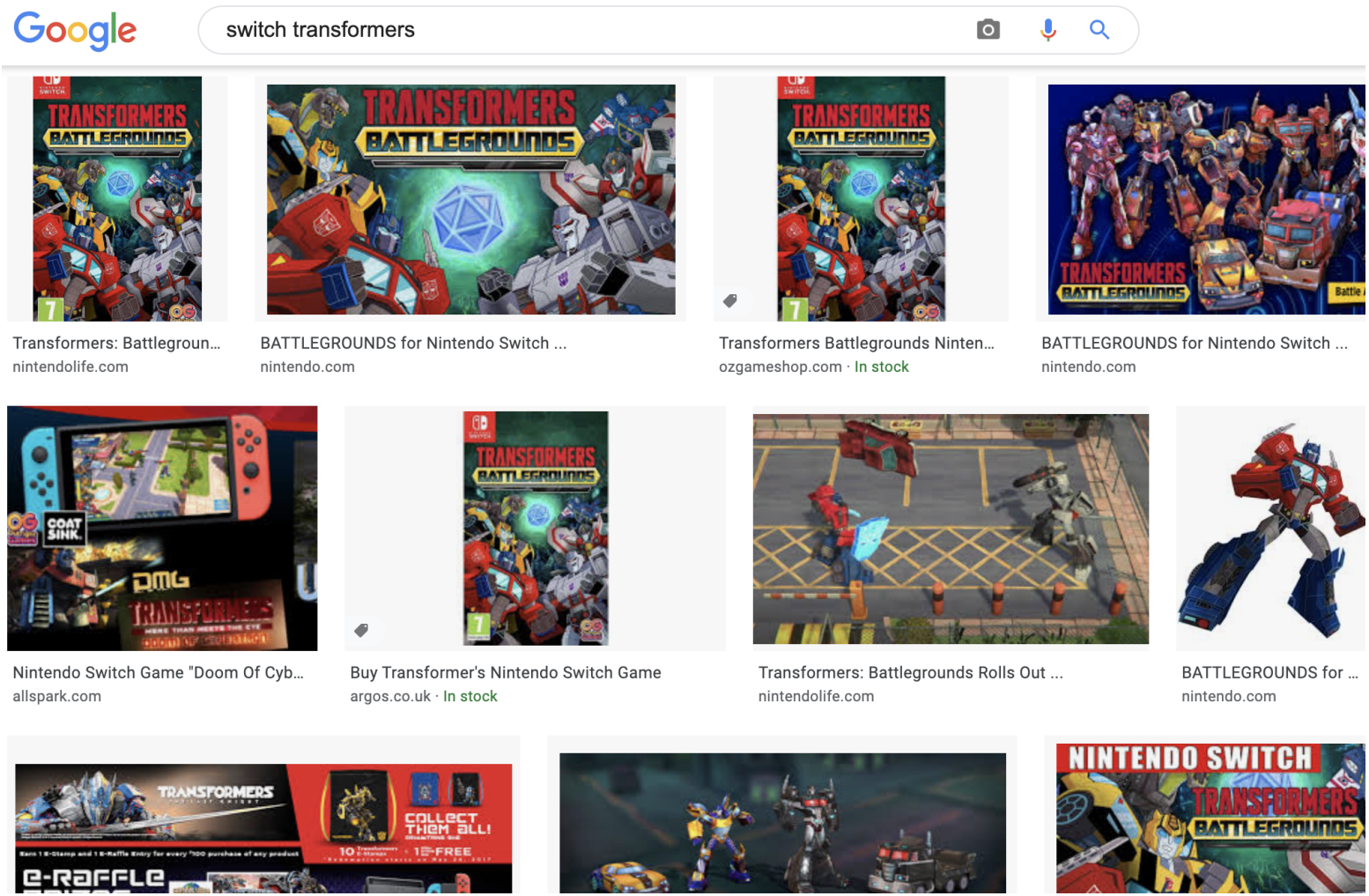
While it gives some indication of the project's scale, the models are too different in architecture for a meaningful 'apples-to-apples' comparison.

- Switch-Transformer is the first neural network model reaching trillions of parameters (1.6T)!

# Is there another direction?


- Switch-Transformer is the first neural network model reaching trillions of parameters (1.6T)!

# What are Switch-Transformers?



# What are Switch-Transformers?

- What Google has done
  - They figured out a way to make the model itself as simple as possible while squeezing in as much compute power as possible to make the increased number of parameters possible.
  - Google has a lot of money and that means it can afford to use as much hardware compute as the AI model can handle.

"Switch Transformers are scalable and effective natural language learners. We simplify Mixture of Experts to produce an architecture that is easy to understand, stable to train and vastly more sample efficient than equivalently-sized dense models. We find that these models excel across a diverse set of natural language tasks and in different training regimes, including pre-training, fine-tuning and multi-task training. These advances make it possible to train models with hundreds of billion to trillion parameters and which achieve substantial speedups relative to dense T5 baselines."

— Google Brain

# What are Switch-Transformers?

- The key idea behind Switch-Transformers is Mixture-of-Experts
  - Mixture-of-Experts (MoE) is a form of Conditional Computation
- A more relevant expert will have a stronger voice
- Each data point is processed by several experts
- Different data points are likely processed by different combinations of experts

# Why conditional computation?

- Larger datasets and models make deep learning flourish
- The entire model is typically activated for every example
  - If both the model size and the number of training examples increase
    - A roughly quadratic increase in training costs
  - The growth in computing power can't meet such demand
    - Moore's law[1]: computational power doubles in 18 months
    - Google's law[2]: deep learning model size doubles in 3.5 months
- Different forms of conditional computation were proposed
  - Large parts of a model are active or inactive on a per-example basis
  - Increase model capacity without a proportional increase in computational costs
  - The gating decisions may be binary or sparse and continuous, stochastic or deterministic
    - Different forms of reinforcement learning and back-propagation are proposed for training the gating decisions

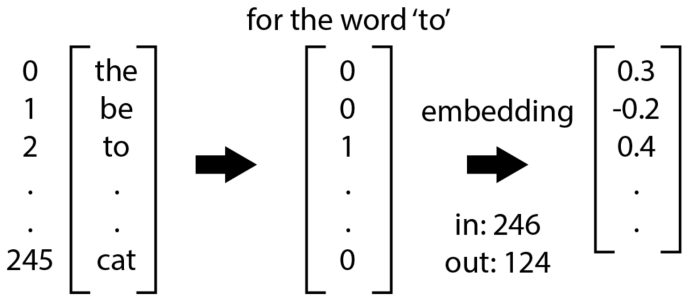
[1] Moore observed that the number of transistors on a chip was doubling about every 18–24 months.

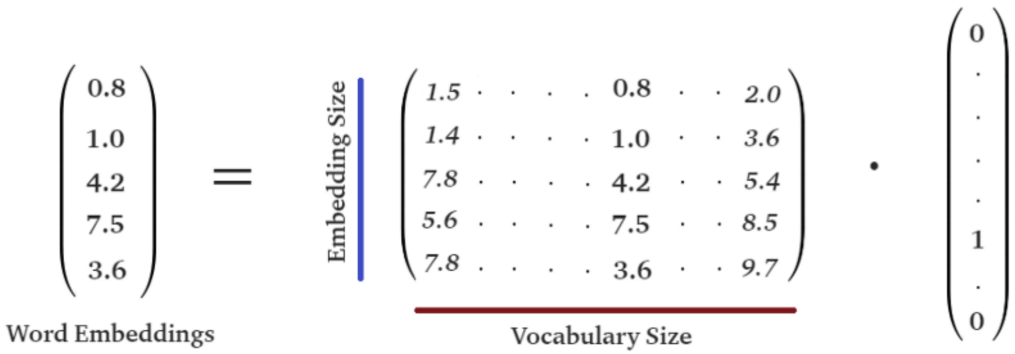
[2] https://arxiv.org/abs/2011.03641

# Outline

- Introduction
- **Sparsely-Gated MoE for LSTM**
- Review on Transformers
- Sparsely-Gated MoE for Transformers

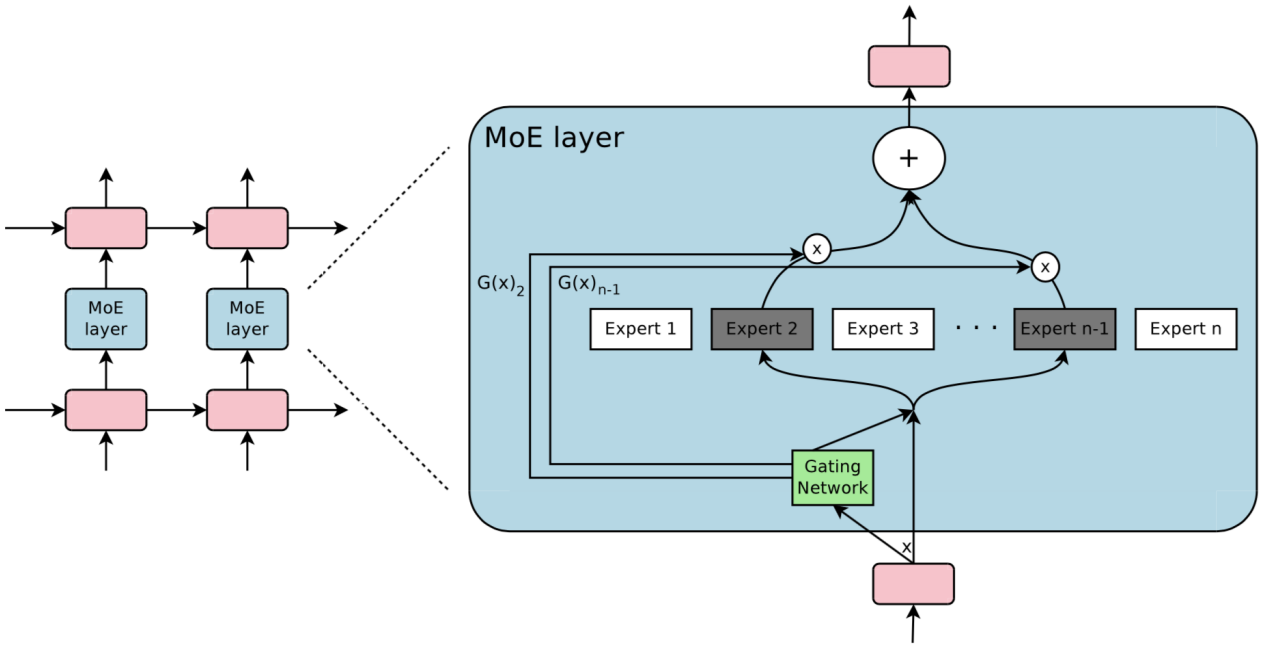
# Review: word embedding



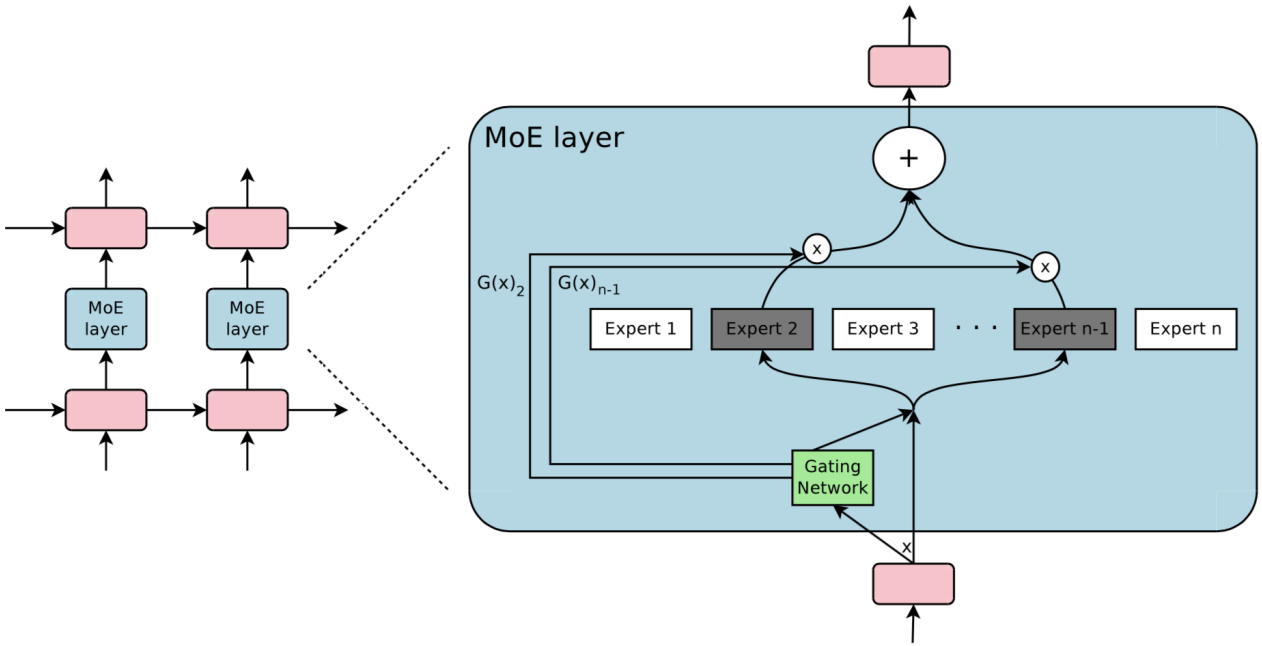

- Word-embedding can transfer a high-dimensional vector to a low-dimensional vector

# Overview of Sparsely-Gated MoE



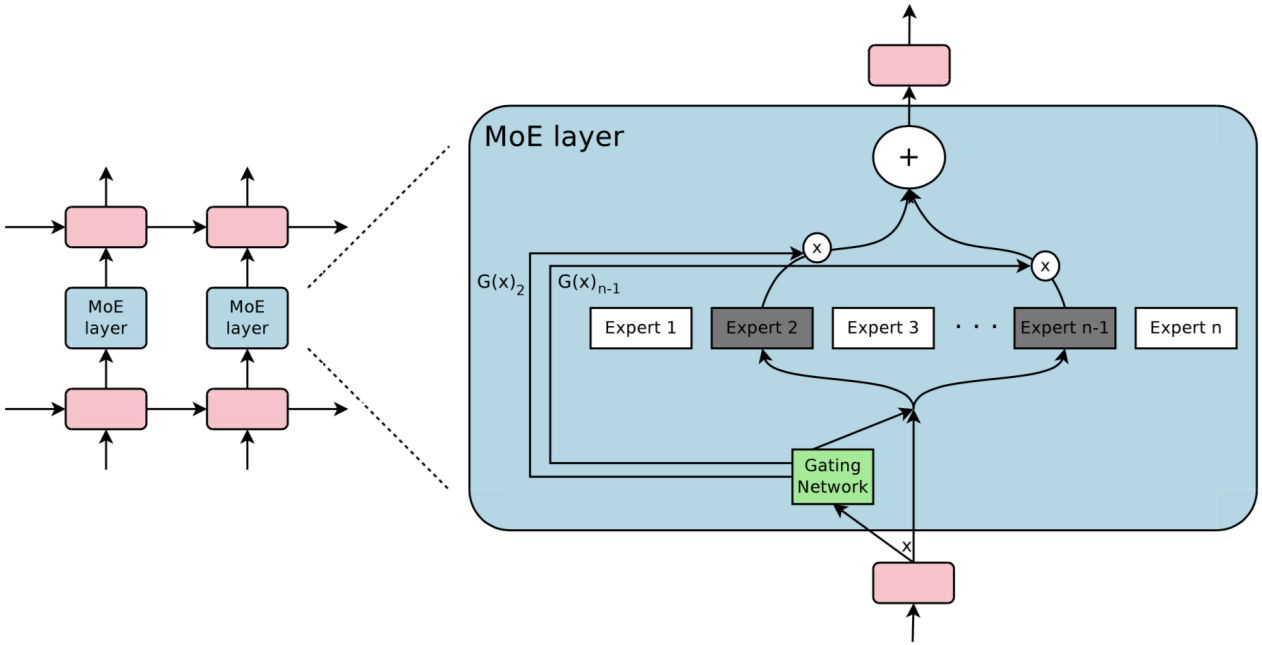

- MoE has a number of experts and a trainable gating network
  - The gate selects a sparse combination of experts to process an input
  - Each expert is a simple feed-forward neural network
- We focus on language modeling and machine translation tasks
  - They empirically benefit from very large models

# Overview of Sparsely-Gated MoE




- A MoE layer embedded in a recurrent language model
- In this case, the sparse gating function selects two experts to perform computations
  - Their outputs are modulated by the outputs of the gating network

# Overview of Sparsely-Gated MoE




- We apply a MoE convolutionally between stacked LSTM layers
- MoE is called once for each position in the text, selecting a potentially different combination of experts at each position
- The different experts tend to become highly specialized based on syntax and semantics (e.g. for subject, verb, object)

# Long history of MoE

- The mixture of experts is the whole model
  - Early attempts (Jacobs et al., 1991; Jordan & Jacobs, 1994)
  - SVMs (Collobert et al., 2002)
  - Gaussian Processes (Tresp, 2001; Deisenroth & Ng, 2015)
  - Dirichlet Processes (Shahbaba  Neal, 2009)
  - Different expert configurations
    - Hierarchical structure (Yao et al., 2009)
    - Infinite number of experts (Rasmussen & Ghahramani, 2002)
    - Adding experts sequentially (Aljundi et al., 2016)
    - Ensemble NMT for machine translation (Garmash & Monz, 2016)
- Using multiple MoEs with their own gating networks as parts of a deep model (Eigen et al. 2013)
  - Complex problems may contain many sub-problems each requiring different experts
    - e.g. fixing an Airbus A350 or Boeing 787 requires different experts to check different parts
  - A potential to introduce sparsity
    - Making MoE a form of conditional computation

# Sparsely-Gated MoE

- MoE layer has $n$ expert networks $E_1, ..., E_n$
  - Each expert has its own parameters
  - For simplicity, we assume they are feed-forward networks with identical architectures, but with separate parameters
- The output of gating network $G$ is a sparse $n$-dimensional vector
- $G(x)$ and $E_i(x)$ are the output of the gating network and the output of the $i$-th expert network for a given input $x$
- The output $y$ of the MoE can be written as:

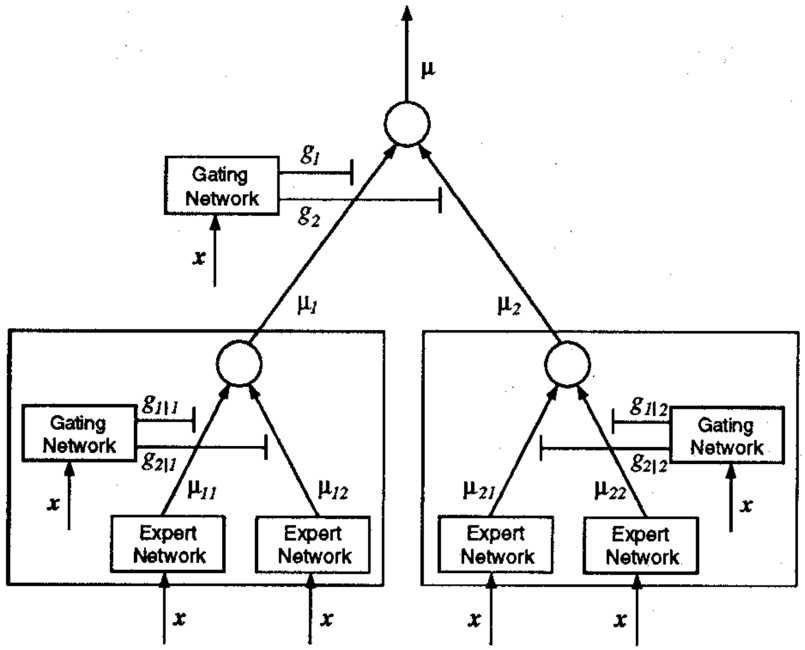
$$y = \sum_{i=1}^{n} G(x)_i E_i(x)$$

- Wherever $G(x)_i = 0$, we don't need to compute $E_i(x)$
  - Idea of conditional computation
  - We can have up to thousands of experts, but only need to evaluate a few of them for every example

# Early Gating Networks (Jordan & Jacobs, 1994)

- Softmax Gating: a simple choice of non-sparse function
  - Multiply the input by a trainable weight matrix $W_g$ and then apply the Softmax function.

$$G(x) = Softmax(xW_g)$$



# Noisy Top-K Gating

- Adding 2 components to Softmax gating: sparsity and noise.
- Before taking softmax function, we add tunable Gaussian noise, then keep only the top $k$ values, setting the rest to $-\infty$ (which causes the corresponding gate values to be 0).
- The sparsity can save computation.
- The noise term helps with load balancing.
- The amount of noise per component is controlled by a second trainable weight matrix $W_{noise}$.

$$G(x) = \text{Softmax}(\text{KeepTopK}(H(x), k))$$

$$H(x)_i = (xW_g)_i + \text{StandardNormal}() \times \text{softplus}((xW_{noise})_i)$$

$$\text{KeepTopK}(v, k)_i = \begin{cases} v_i & v_i \text{ is in the top } k \text{ elements of } v \\ -\infty & \text{otherwise} \end{cases}$$

# Training the Gating Network

- We train the gating network by simple back-propagation, along with the rest of the model.
- If we choose $k > 1$, the gate values for the top $k$ experts have nonzero derivatives with respect to the weights of the gating network.
- Gradients also back-propagate through the gating network to its inputs.

# Load Balance

- Load-Unbalanced Issue
  - Some experts have too many tasks, other experts are idle
  - The system's speed will be limited by the slowest expert.
    - The busiest expert
  - This will be a very serious issue on distributed systems.
- The gating network tends to converge to a state where it always produces large weights for the same few experts
- The load-imbalance is self-reinforcing
  - The favored experts are trained more rapidly and thus are selected even more by the gating network

# A soft constraint method for load-balancing

- Importance of an expert relative to a batch of training examples
  - The batchwise sum of the gate values for that expert

$$I = Importance(X) = \sum_{x \in X} G(x)$$

- An additional loss ($L_{importance}$) is added to the overall loss function
  - $L_{importance}$ is the square of the coefficient of variation $(CV)^3$ of the set of importance values
    - It is multiplied by a hand-tuned scaling factor $w_{importance}$
  - This additional loss encourages all experts to have equal importance

$$L_{importance}(X) = w_{importance} \times [CV(I)] \times [CV(I)]$$

[3] CV is a measure of relative variability. It is the ratio of the standard deviation to the mean.

# Limitations of importance-based load-balancing

- This loss function can lead to equal importance
- Different experts may still have very different number of examples
  - One expert may receive a few examples with large weights
  - Another may receive many examples with small weights
- This can cause memory and performance problems on large-scale distributed hardware

# Load-based Loss for Load-Balancing

- The number of examples got by an expert is a discrete quantity
  - It can not be used in back-propagation
- We define a smooth estimator $Load(X)$ of the number of examples assigned to each expert for a batch $X$ of inputs
  - The smoothness can back-propagate gradients through the estimator
  - This is the purpose of the noise term in the gating function
- $P(x,i)$ is the probability that $G(x)_i$ is nonzero
  - Given a new random choice of noise on element $i$
  - Keeping the already-sampled choices of noise on the other elements
  - $G(x)_i$ is nonzero if and only if $H(x)_i$ is greater than the $k^{th}$-greatest element of $H(x)$ excluding itself
- $P(x,i) = P_r[H(x)_i > k^{th}\text{excluding}(H(x),k,i)]$
  - $H(x)_i = (xW_g)_i + \text{StandardNormal}() \times \text{softplus}((xW_{noise})_i)$
  - $[k^{th}\text{excluding}(v,k,i)]$ means the $k^{th}$ highest component of $v$, excluding component $i$

# Load-based Loss for Load-Balancing

- We can simply the equation
  - $\Phi$ is the CDF (cumulative distribution function) of the standard normal distribution
  - We want to select the top $k$ experts

$$P(x,i) = \Phi\left(\frac{(xW_g)_i - k^{th}\text{excluding}(H(x),k,i)}{\text{softplus}((xW_{noise})_i)}\right)$$

- Load loss is the square of coefficient of variation of the load vector
  - Multiplied by a hand-tuned scaling factor $w_{load}$
- $L_{load}(X) = w_{load} \times CV(Load(X)) \times CV(Load(X))$
  - Load vector: $Load(X)_i = \sum_{x \in X} P(x,i)$

# Different combinations of load-balance loss functions

- We trained a set of models with identical architecture using different values of $w_{importance}$ and $w_{load}$
  - We trained each model for 10 epochs, then measured perplexity on the testing dataset
  - Ratio of the load on the most overloaded expert to the average load
    - Achieving load-balance when this ratio is close to 1.0
  - $w_{importance} = 0.0$: not using importance-based loss
  - $w_{load} = 0.0$: not using load-based loss

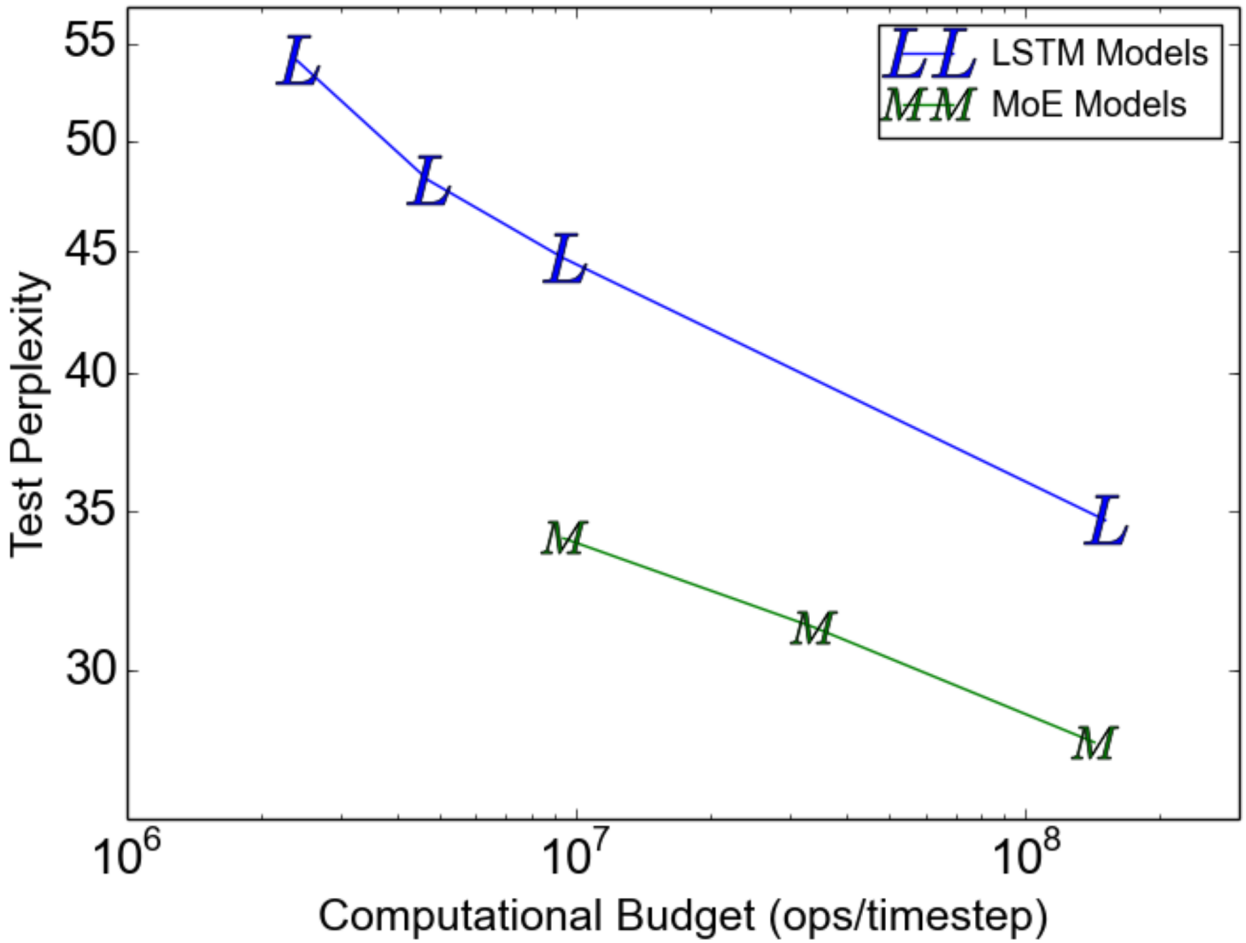
| $w_{importance}$ | $w_{load}$ | Test Perplexity | $CV(Importance(X))$ | $CV(Load(X))$ | $\frac{max(Load(X))}{mean(Load(X))}$ |
|---|---|---|---|---|---|
| 0.0 | 0.0 | 39.8 | 3.04 | 3.01 | 17.80 |
| 0.2 | 0.0 | **35.6** | 0.06 | 0.17 | 1.47 |
| 0.0 | 0.2 | 35.7 | 0.22 | 0.04 | 1.15 |
| 0.1 | 0.1 | **35.6** | 0.06 | 0.05 | 1.14 |
| 0.01 | 0.01 | 35.7 | 0.48 | 0.11 | 1.37 |
| 1.0 | 1.0 | 35.7 | 0.03 | 0.02 | **1.07** |

# Summary of MoE Performance

- 1 billion word language modeling benchmark
  - The dataset (Chelba et al., 2013) includes shuffled unique sentences from news articles, totaling approximately 829 million words, with a vocabulary of 793,471 words.
  - The best previously published results (Jozefowicz et al., 2016) use models consisting of one or more stacked LSTM layers.
  - Our models include 2 stacked LSTM layers with a MoE layer between them. We vary the sizes of layers and the number of experts.

| | Test Perplexity 10 epochs | Test Perplexity 100 epochs | #Parameters excluding embedding and softmax layers | ops/timestep | Training Time 10 epochs | TFLOPS /GPU |
|---|---|---|---|---|---|---|
| Best Published Results | 34.7 | 30.6 | 151 million | 151 million | 59 hours, 32 k40s | 1.09 |
| Low-Budget MoE Model | 34.1 | | 4303 million | 8.9 million | 15 hours, 16 k40s | 0.74 |
| Medium-Budget MoE Model | 31.3 | | 4313 million | 33.8 million | 17 hours, 32 k40s | 1.22 |
| High-Budget MoE Model | **28.0** | | 4371 million | 142.7 million | 47 hours, 32 k40s | **1.56** |

# LSTM vs MoE




- The top line is state-of-the-art LSTM model (Jozefowicz et al., 2016).
- The bottom line is 4-billion parameter MoE model with different computational budgets.

# MoE for Machine Translation

| Model | Test Perplexity | Test BLEU | ops/timenstep | Total #Parameters | Training Time |
|---|---|---|---|---|---|
| MoE with 2048 Experts | 2.69 | 40.35 | 85M | 8.7B | 3 days/64 k40s |
| MoE with 2048 Experts (longer training) | **2.63** | **40.56** | 85M | 8.7B | 6 days/64 k40s |
| GNMT (Wu et al., 2016) | 2.79 | 39.22 | 214M | 278M | 6 days/96 k80s |
| GNMT+RL (Wu et al., 2016) | 2.96 | 39.92 | 214M | 278M | 6 days/96 k80s |
| PBMT (Durrani et al., 2014) | | 37.0 | | | |
| LSTM (6-layer) (Luong et al., 2015b) | | 31.5 | | | |
| LSTM (6-layer+PosUnk) (Luong et al., 2015b) | | 33.1 | | | |
| DeepAtt (Zhou et al., 2016) | | 37.7 | | | |
| DeepAtt+PosUnk (Zhou et al., 2016) | | 39.2 | | | |

- Results on WMT'14 En→Fr newstest2014

# MoE for Machine Translation

| Model | Test Perplexity | Test BLEU | ops/timestep | Total #Parameters | Training Time |
|---|---|---|---|---|---|
| MoE with 2048 Experts | **4.64** | **26.03** | 85M | 8.7B | 1 day/64 k40s |
| GNMT (Wu et al., 2016) | 5.25 | 24.91 | 214M | 278M | 1 day/96 k80s |
| GNMT +RL (Wu et al., 2016) | 8.08 | 24.66 | 214M | 278M | 1 day/96 k80s |
| PBMT (Durrani et al., 2014) | | 20.7 | | | |
| DeepAtt (Zhou et al., 2016) | | 20.6 | | | |

- Results on WMT'14 En→De newstest2014

# MoE for Machine Translation

| Model | Eval Perplexity | Eval BLEU | Test Perplexity | Test BLEU | ops/timestep | Total #Parameters | Training Time |
|---|---|---|---|---|---|---|---|
| MoE with 2048 Experts | **2.60** | **37.27** | **2.69** | **36.57** | 85M | 8.7B | 1 day/64 k40s |
| GNMT (Wu et al., 2016) | 2.78 | 35.80 | 2.87 | 35.56 | 214M | 278M | 6 days/96 k80s |

- Results on the Google Production En→Fr dataset

# MoE for Multi-Language Machine Translation

| | GNMT-Mono | GNMT-Multi | MoE-Multi | MoE-Multi vs. GNMT-Multi |
|---|---|---|---|---|
| Parameters | 278M / model | 278M | 8.7B | |
| ops/timestep | 212M | 212M | 102M | |
| training time, hardware | various | 21 days, 96 k20s | **12 days, 64 k40s** | |
| Perplexity (dev) | | 4.14 | **3.35** | -19% |
| French → English Test BLEU | 36.47 | 34.40 | **37.46** | +3.06 |
| German → English Test BLEU | 31.77 | 31.17 | **34.80** | +3.63 |
| Japanese → English Test BLEU | 23.41 | 21.62 | **25.91** | +4.29 |
| Korean → English Test BLEU | 25.42 | 22.87 | **28.71** | +5.84 |
| Portuguese → English Test BLEU | 44.40 | 42.53 | **46.13** | +3.60 |
| Spanish → English Test BLEU | 38.00 | 36.04 | **39.39** | +3.35 |
| English → French Test BLEU | 35.37 | 34.00 | **36.59** | +2.59 |
| English → German Test BLEU | **26.43** | 23.15 | 24.53 | +1.38 |
| English → Japanese Test BLEU | **23.66** | 21.10 | 22.78 | +1.68 |
| English → Korean Test BLEU | **19.75** | 18.41 | 16.62 | -1.79 |
| English → Portuguese Test BLEU | **38.40** | 37.35 | 37.90 | +0.55 |
| English → Spanish Test BLEU | 34.50 | 34.25 | **36.21** | +1.96 |

- GNMT-Mono: 12 separately trained single-pair GNMT models
- GNMT-Multi: one GNMT model on a very large combined dataset of 12 language pairs
- GNMT-Mono is better than GNMT-Multi
  - 12 models have 12 times the capacity and 12 times the aggregate training of 1 model
- MoE-Multi: one MoE model on a very large combined dataset of 12 language pairs

# Outline

- Introduction
- Sparsely-Gated MoE for LSTM
- **Review on Transformers**
- Sparsely-Gated MoE for Transformers

# What's next?

**hardmaru** @hardmaru · Jan 21

Transformers are the new LSTMs

23 61 648

**hardmaru** @hardmaru · Jan 21

This tweet didn't live up to its expectations:

We'll see in 46 years...

> **hardmaru** @hardmaru · May 15, 2017
>
> LSTMs are like the AK47 of neural nets. No matter how hard we try to replace it with something new, it will still be used 50 years from now.

7 17 252

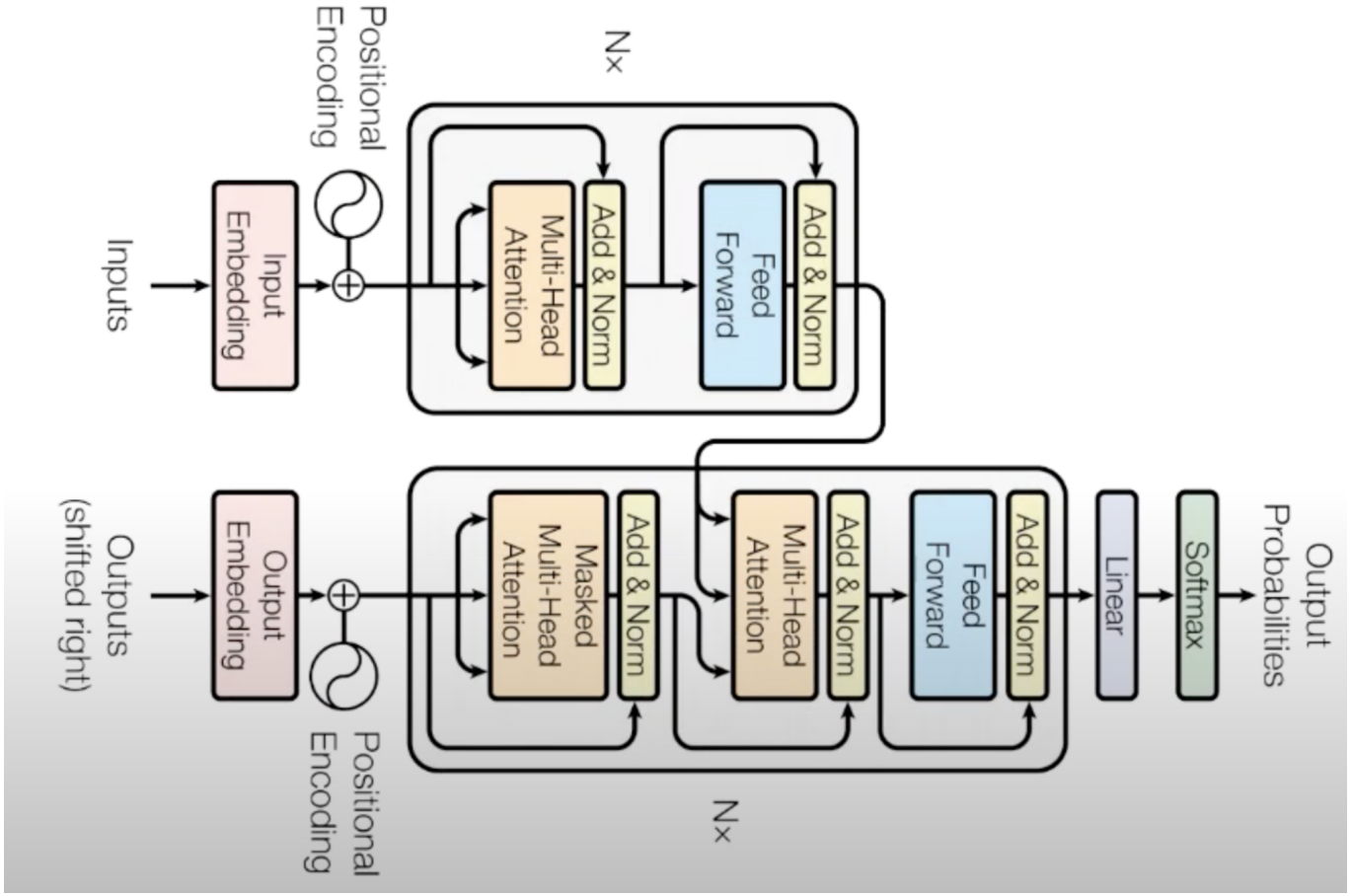
# What's next?

## 10 Novel Applications using Transformers [DL]

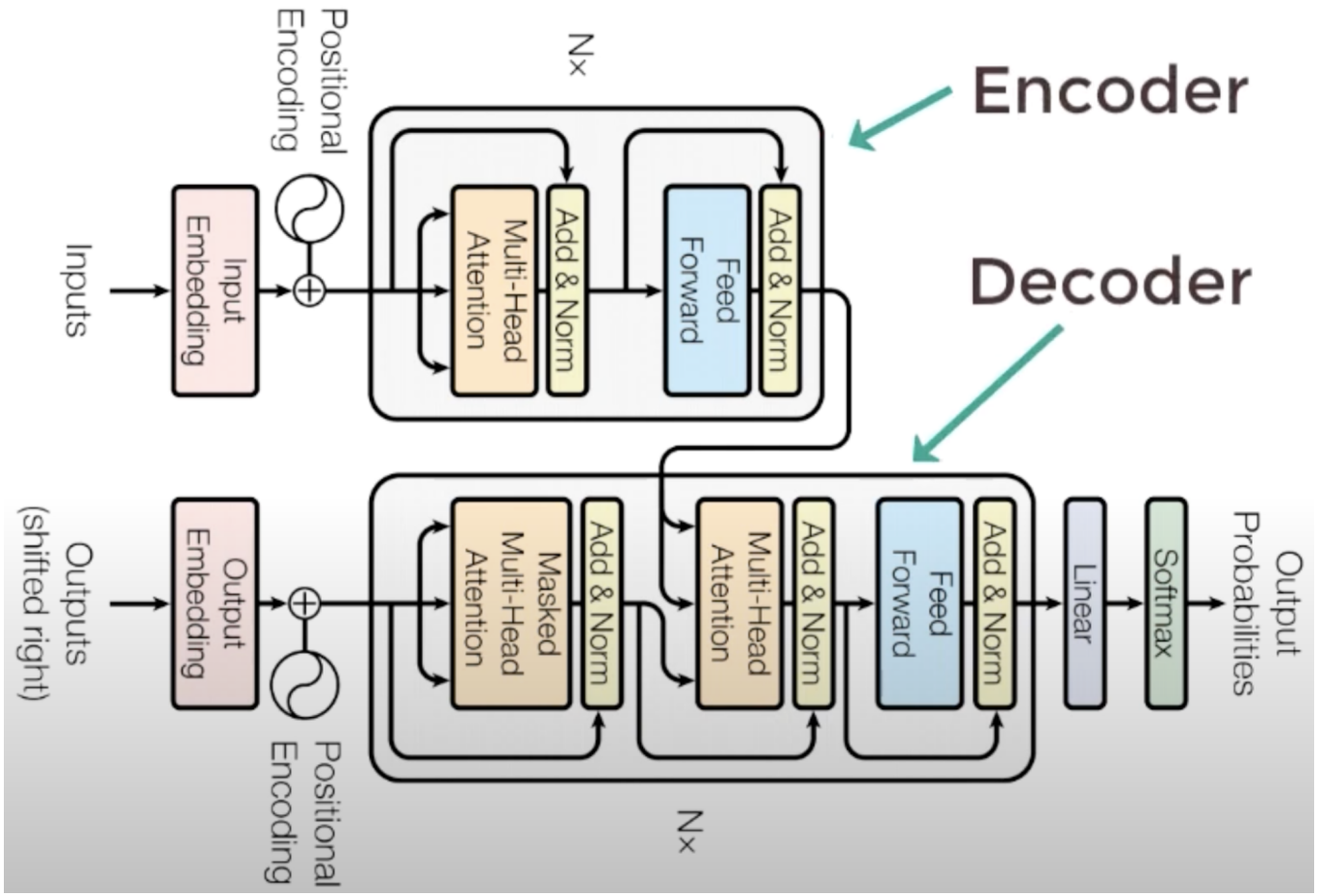
Transformers have had a lot of success in training neural language models. In the past few weeks, we've seen several trending papers with code applying Transformers to new types of task:

🏞 **Transformer for Image Synthesis** - 🔗 Esser et al. (2020)
🔵 **Transformer for Multi-Object Tracking** - 🔗 Sun et al. (2020)
🎶 **Transformer for Music Generation** - 🔗 Hsiao et al. (2021)
💃 **Transformer for Dance Generation with Music** - 🔗 Huang et al. (2021)
🔮 **Transformer for 3D Object Detection** - 🔗 Bhattacharyya et al. (2021)
🗿 **Transformer for Point-Cloud Processing** - 🔗 Guo et al. (2020)
⏰ **Transformer for Time-Series Forecasting** - 🔗 Lim et al. (2020)
👁 **Transformer for Vision-Language Modeling** - 🔗 Zhang et al. (2021)
🛣 **Transformer for Lane Shape Prediction** - 🔗 Liu et al. (2020)
🌠 **Transformer for End-to-End Object Detection** - 🔗 Zhu et al. (2021)

# Transformer Flow

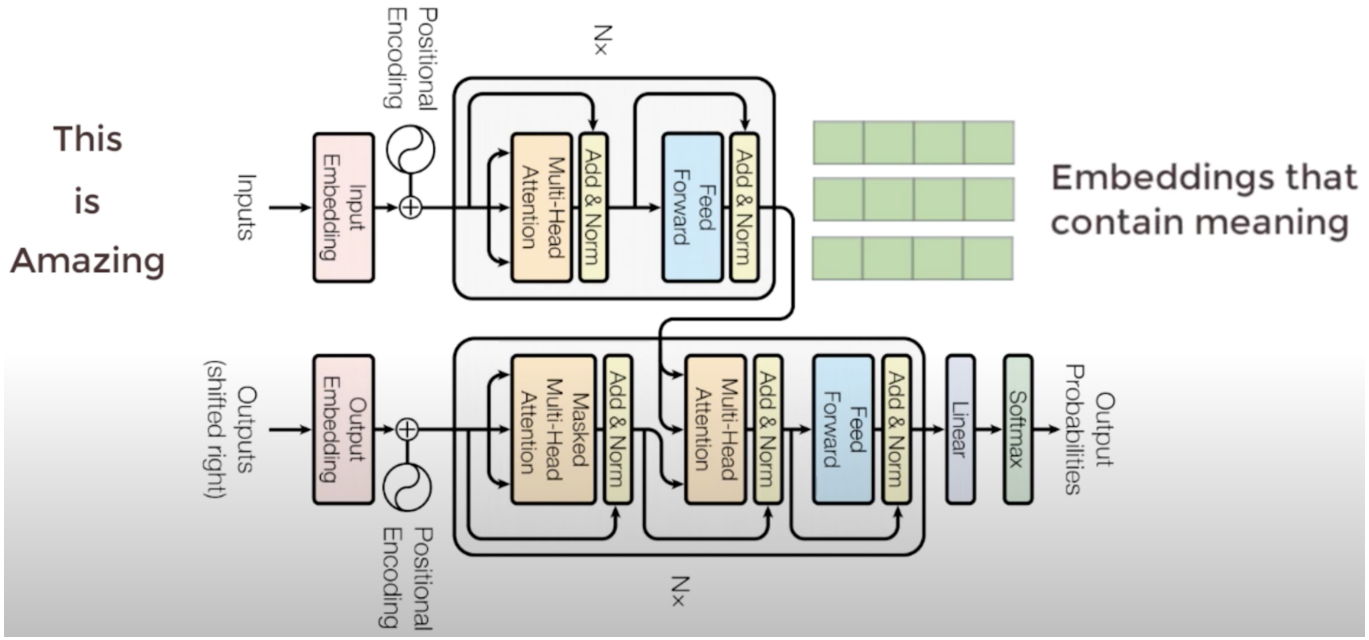


# Transformer Flow



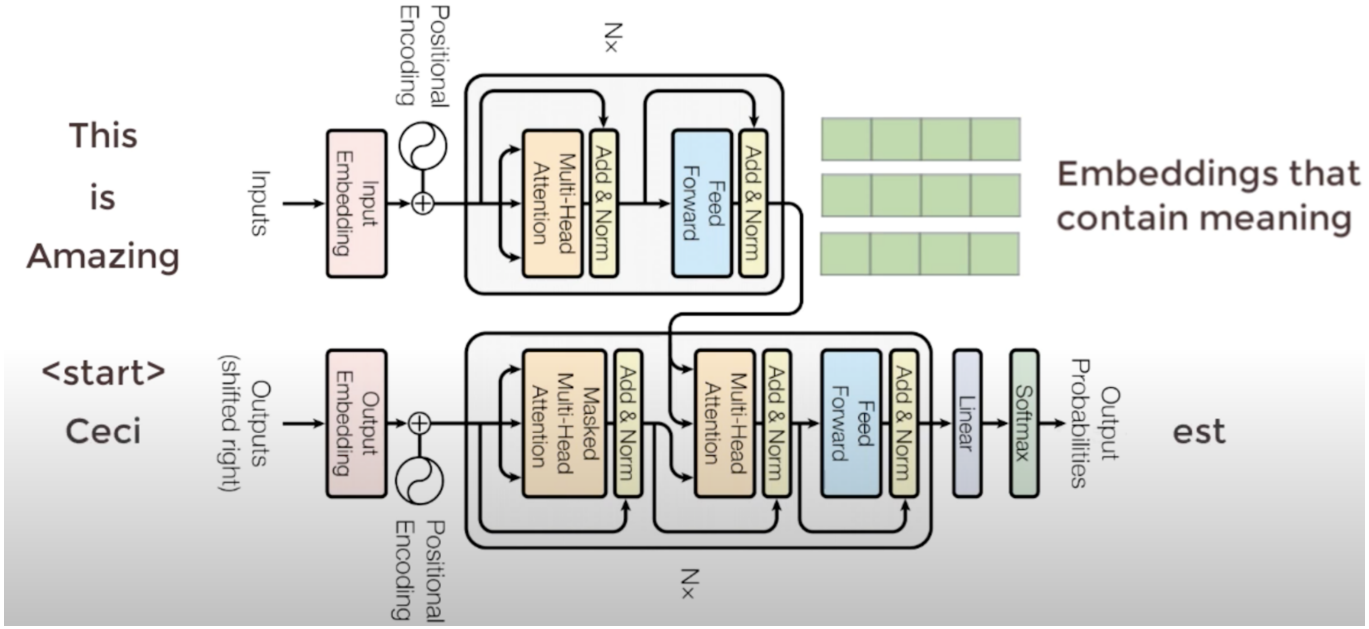

- e.g. English to French translation

# Transformer Flow



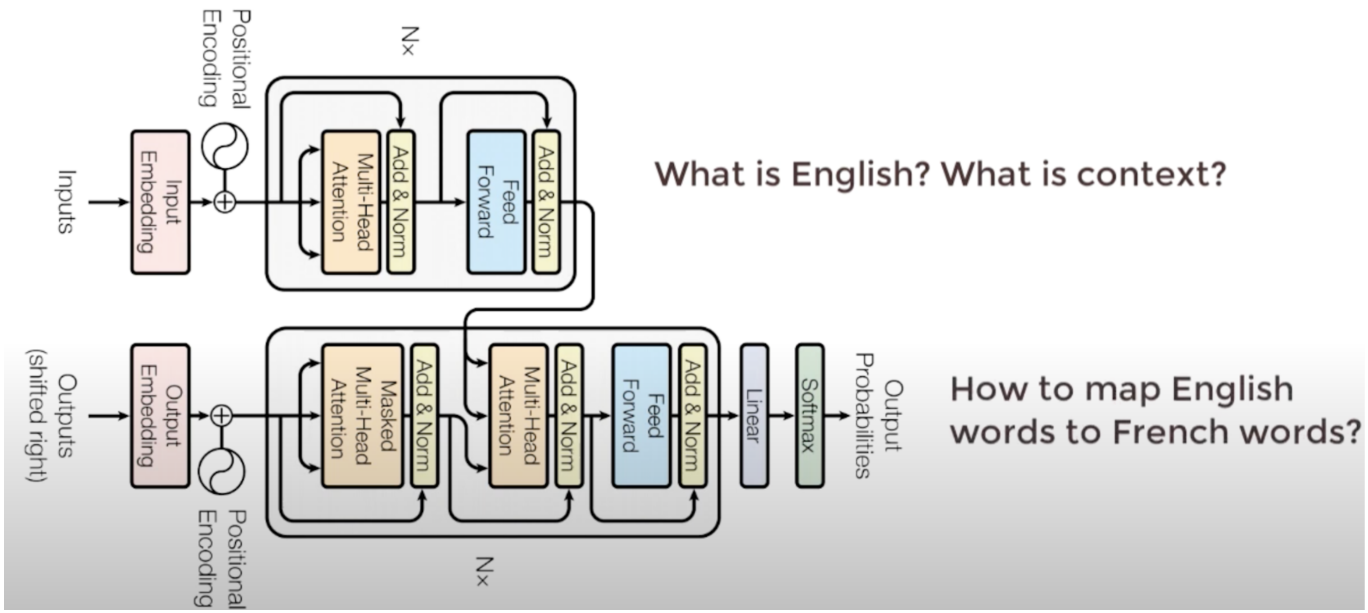

- The encoder takes English words simultaneously and generates embeddings for every word simultaneously.
- Embeddings are vectors that contain the meaning of the words.
- Similar words have closer numbers in the embeddings.

# Transformer Flow



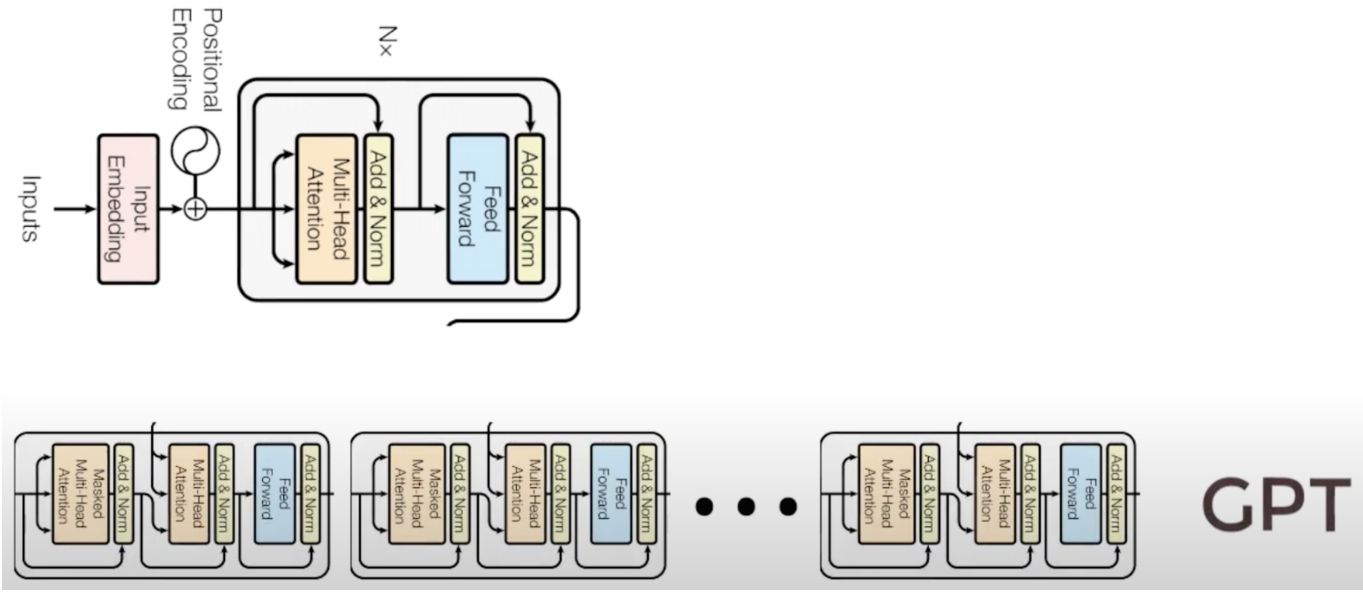

- The decoder takes these embeddings from the encoder and previously generated words of the translated French sentence.
- The decoder uses them to generate the next French word.
- We keep generating the French translation one word a time until the end of the sentence is reached.

# Why this is so much better than LSTM?



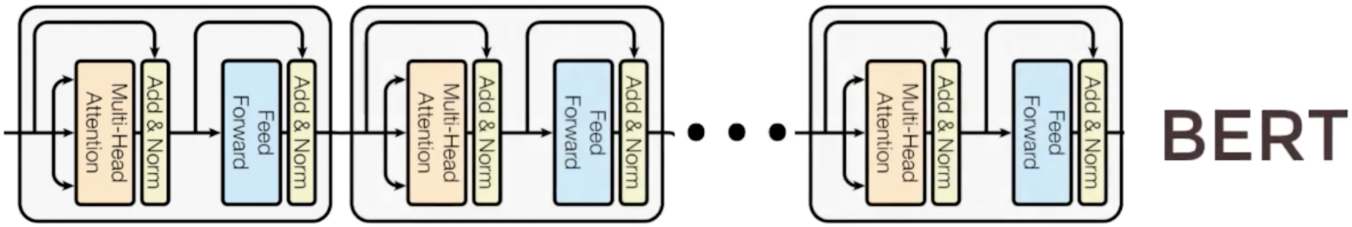

- We can physically see a separation in tasks.
  - The encoder learns what is English.
    - What is grammar and what is context?
  - The decoder learns how do English words relate to French words.
- They separately have some underlying understandings of languages.

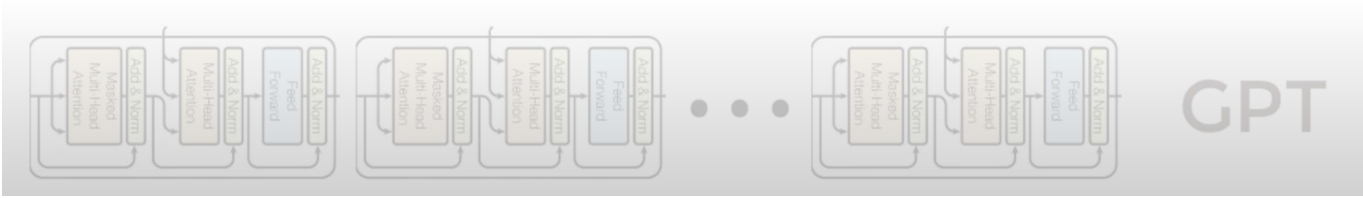
# Transformer Flow




- We stack the decoders and we get GPT transformer architecture.

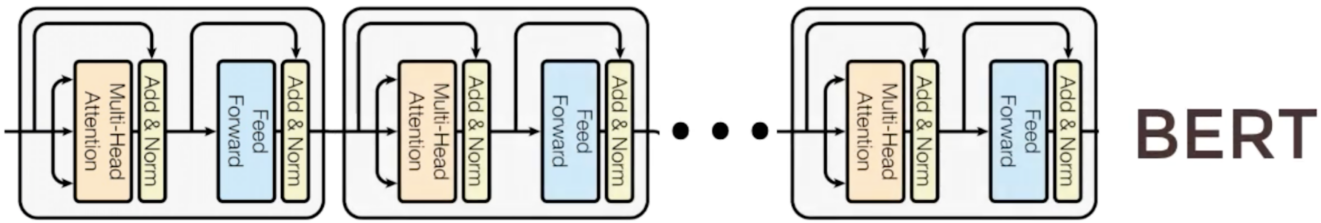
# Transformer Flow



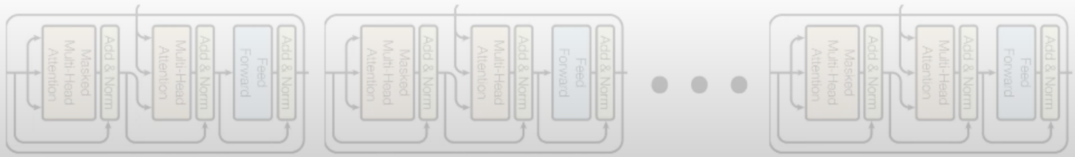




- We stack the encoders and we get BERT

# Transformer Flow

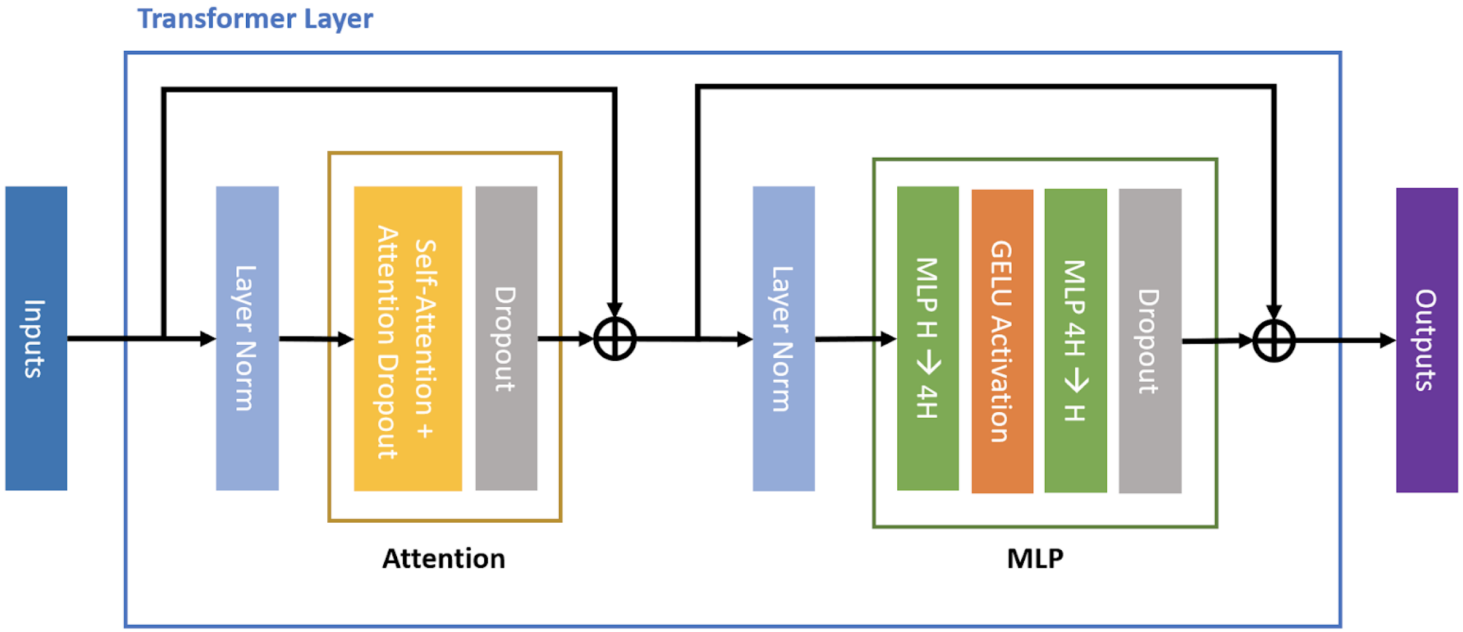



Bidirectional Encoder Representation from Transformers




- We stack the encoders and we get BERT

# Transformer Flow



# Question

- What is the best application of MoE? What do you think?
  - Something related to different tasks?
  - Different experts can process different tasks!
- e.g. Multilingual Translation

# Outline

- Introduction
- Sparsely-Gated MoE for LSTM
- Review on Transformers
- **Sparsely-Gated MoE for Transformers**

# References for further reading


- Shazeer, Noam, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer." arXiv preprint arXiv:1701.06538 (2017).
  - ICLR reviews are at https://openreview.net/forum?id=B1ckMDqlg
- Kaiser, Lukasz, Aidan N. Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. "One model to learn them all." arXiv preprint arXiv:1706.05137 (2017).
- Lepikhin, Dmitry, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. "Gshard: Scaling giant models with conditional computation and automatic sharding." arXiv preprint arXiv:2006.16668 (2020).
  - ICLR reviews are at https://openreview.net/forum?id=qrwe7XHTmYb
- Fedus, William, Barret Zoph, and Noam Shazeer. "Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity." arXiv preprint arXiv:2101.03961 (2021).