Fine-Grained Conditional Computation in Transformers

Sebastian Jaszczur, University of Warsaw & IDEAS NCBR
PhD Advisor: Marek Cygan

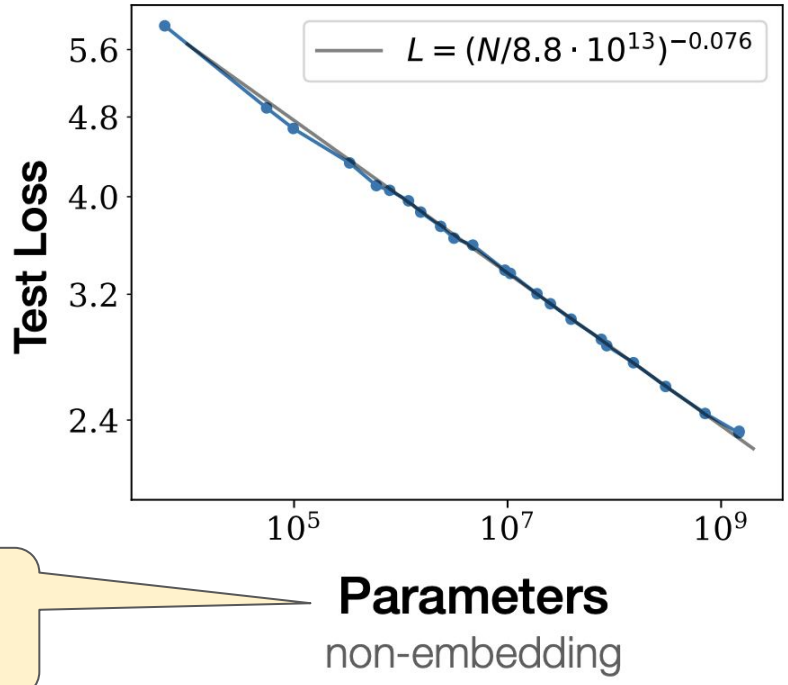
ML in PL Conference, November 2022

IDEAS NCBR

Motivation:
Constantly Increasing Size of Deep Neural Networks

Neural networks: the bigger the better

Model error



Model size

Figure from *Kaplan et al. 2020, Scaling Laws for Neural Language Models*

Why bigger networks are better?

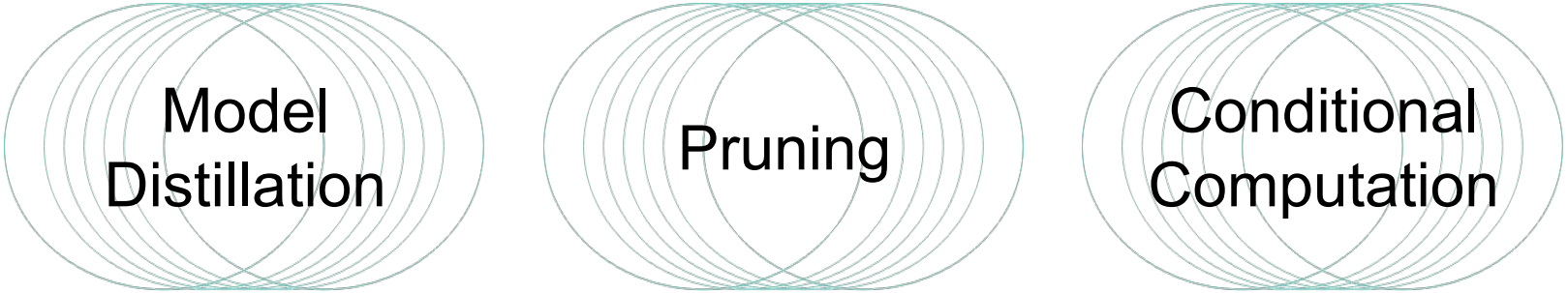
Two main reasons:

- More parameters and neurons ⇔ more "knowledge" stored!
- More layers ⇔ more "thinking" time.

There is a problem: bigger neural network is slower…

Reducing computation: possible approaches

Among others:



Conditional Computation

Neural networks are inefficient...

Consider the question:

"is there a cat on this picture?"

What do you think the model should pay attention to?

Neural networks are inefficient...

Standard neural network spends **the same amount of compute** on both halves of this image.

This is obviously inefficient.

Goals of conditional computation

We want to be able to:

- skip computation of parts of inputs
- skip computation of certain parts of model (neural network)

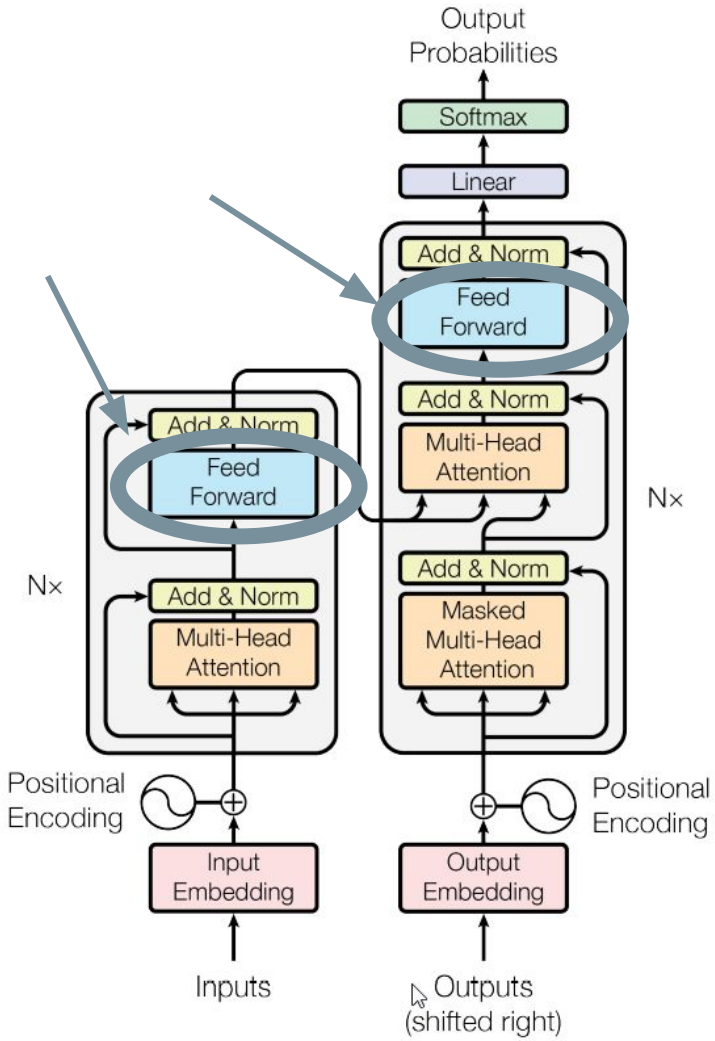
Mixture of Experts

Quick recap of Transformer

Every token is processed by multiple layers.

Let's focus on **Feed Forward** layer!

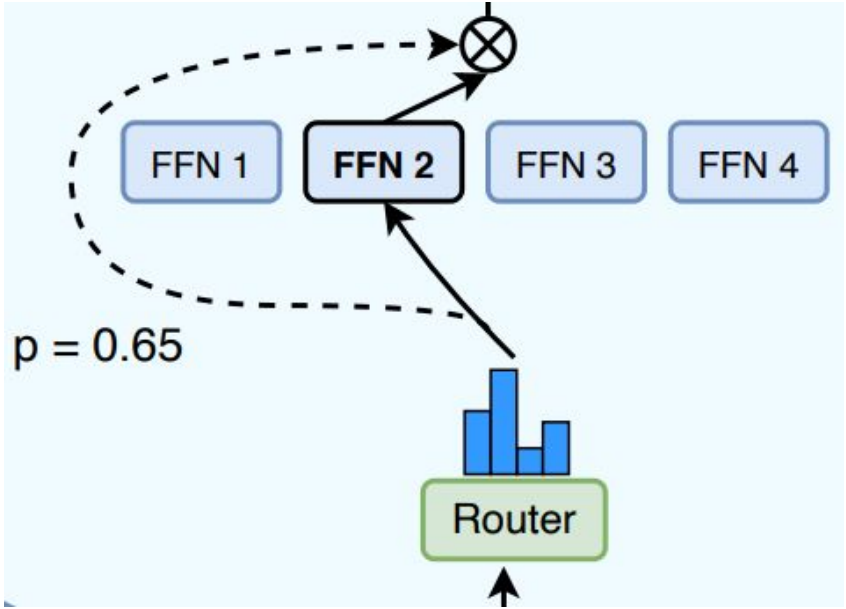
There are techniques making other layers faster, but we will not discuss them.



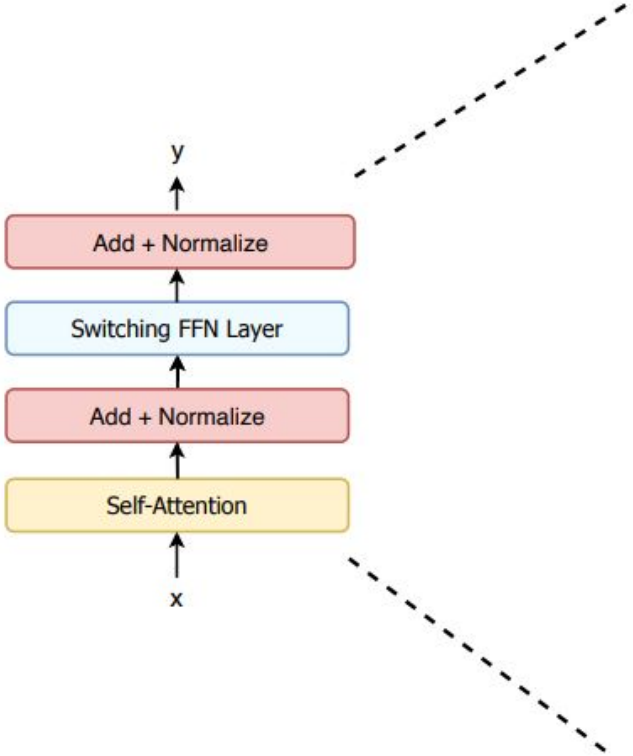
Example: Switch Transformer, Fedus et al. 2021

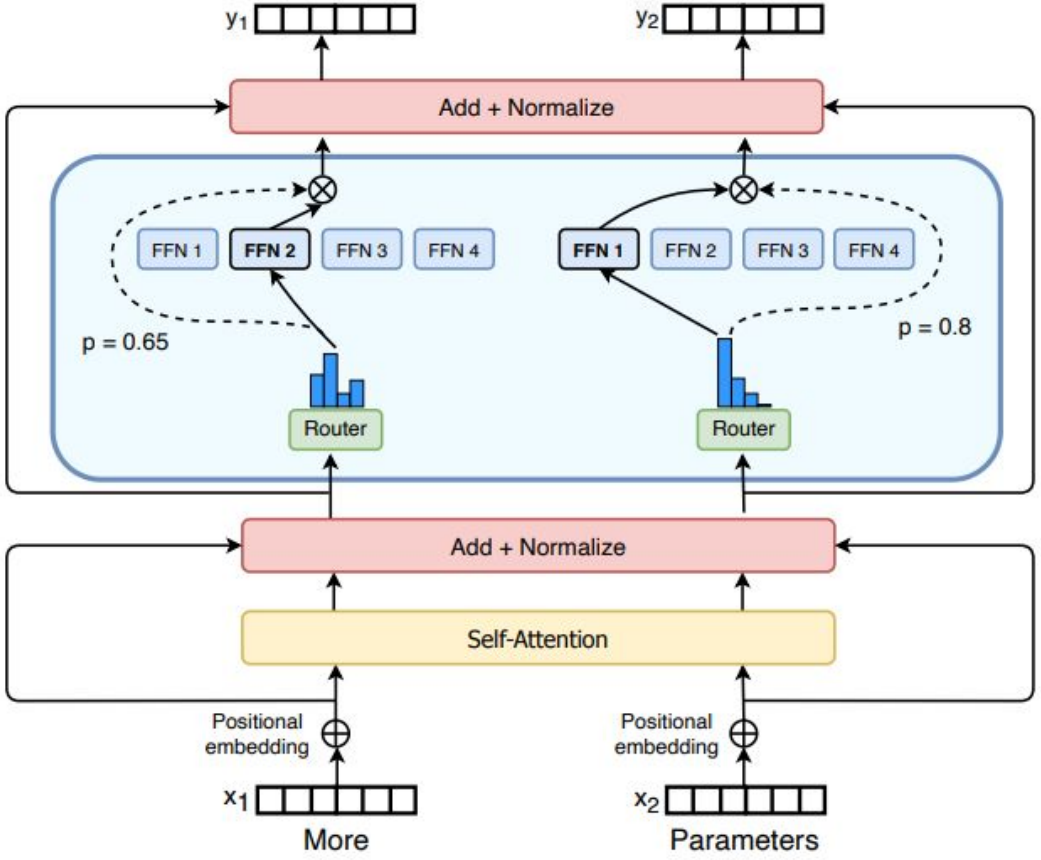
We duplicate Feed Forward layer N times, getting N experts.

We insert a router, which decides which version of the layer (which expert) to use.



Switch Transformer





Fine-Grained Conditional Computation… during Inference

Sparse is Enough in Scaling Transformers

Sparse is Enough in Scaling Transformers

Sebastian Jaszczur* University of Warsaw

Aakanksha Chowdhery Google Research

Afroz Mohiuddin Google Research

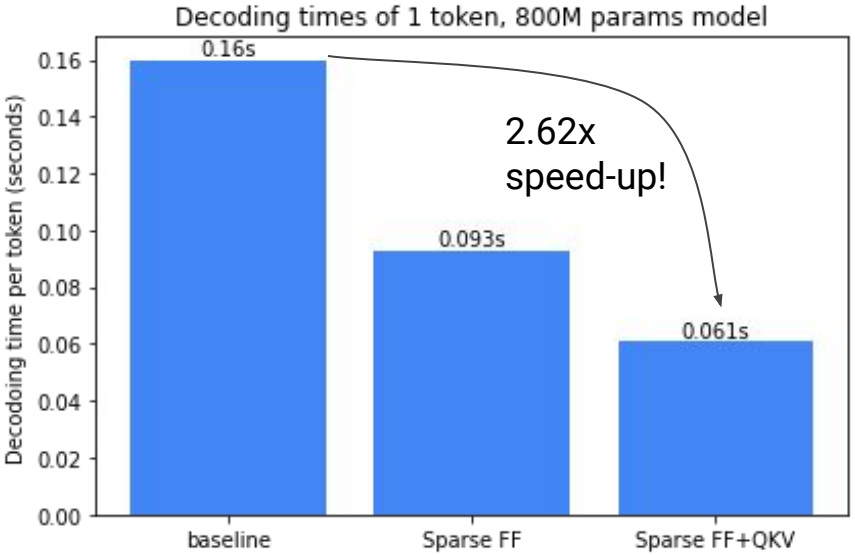
Łukasz Kaiser* OpenAI

Wojciech Gajewski Google Research

Henryk Michalewski Google Research

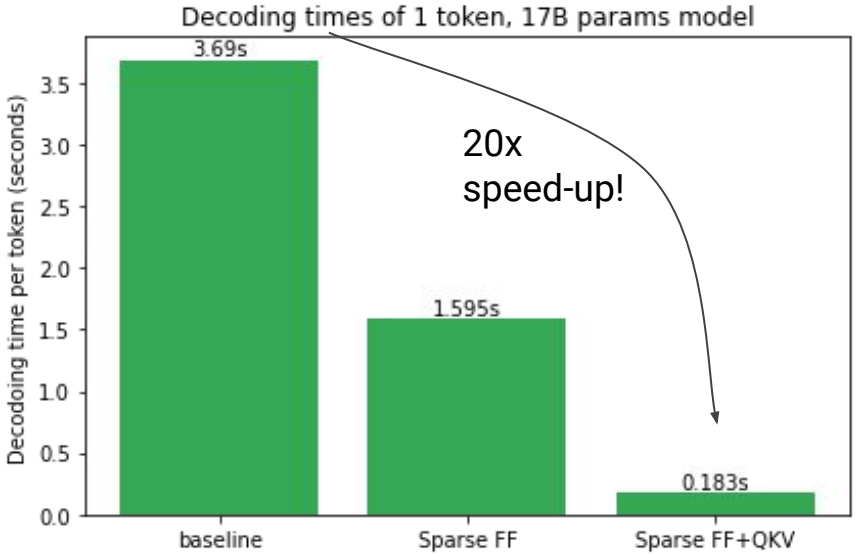
Jonni Kanerva Google Research

Sparse Transformer - Inference Time Improvement



~4 seconds per sentence

to ~1.5 seconds!



~90 seconds per sentence

to ~4.5 seconds!

Sparse is Enough in Scaling Transformers

	Params	Dec. time	Dec. time per block
baseline Transf.	800M	0.160s	5.9ms
+ Sparse FF	-	0.093s	3.1ms
+ Sparse QKV	-	0.152s	6.2ms
+ Sparse FF+QKV	-	0.061s	1.9ms
Speedup		2.62x	3.05x
baseline Transf.	17B	3.690s	0.581s
+Sparse FF	-	1.595s	0.259s
+Sparse QKV	-	3.154s	0.554s
+Sparse FF+QKV	-	0.183s	0.014s
Speedup		20.0x	42.5x

Table 1: Decoding speed (in seconds) of a single token. For Transformer model (equivalent to T5 large with approximately 800M parameters), Scaling Transformers with proposed sparsity mechanisms (FF+QKV) achieve up to 2x speedup in decoding compared to baseline dense model and 20x speedup for 17B param model. [2]

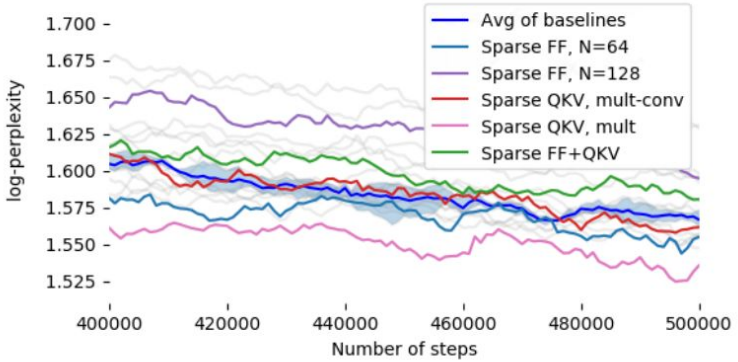
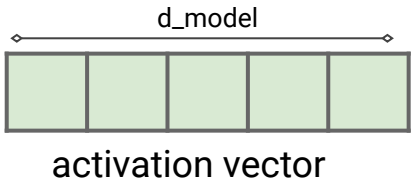


Figure 1: Log-perplexity of Scaling Transformers (equivalent to T5 large with approximately 800M parameters) on C4 dataset with proposed sparsity mechanisms (FF, QKV, FF+QKV) is similar to baseline dense model. Other models used in this paper are shown in grey lines; raw data is available in the appendix.

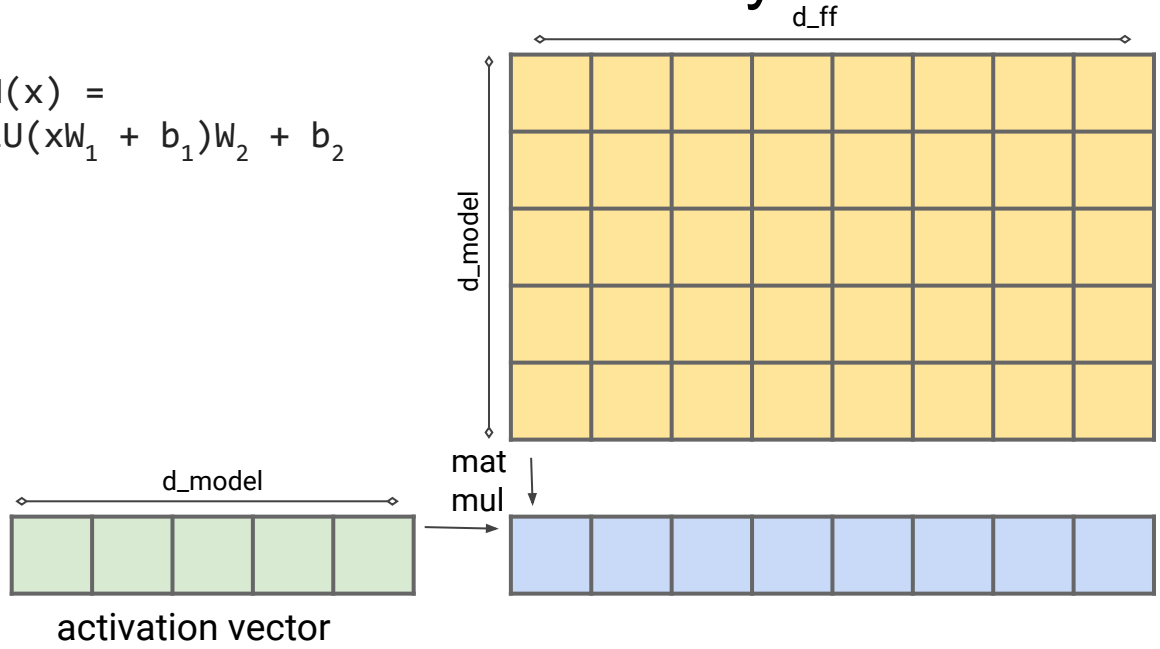
Standard Feed Forward Layer

$$FNN(x) = ReLU(xW_1 + b_1)W_2 + b_2$$



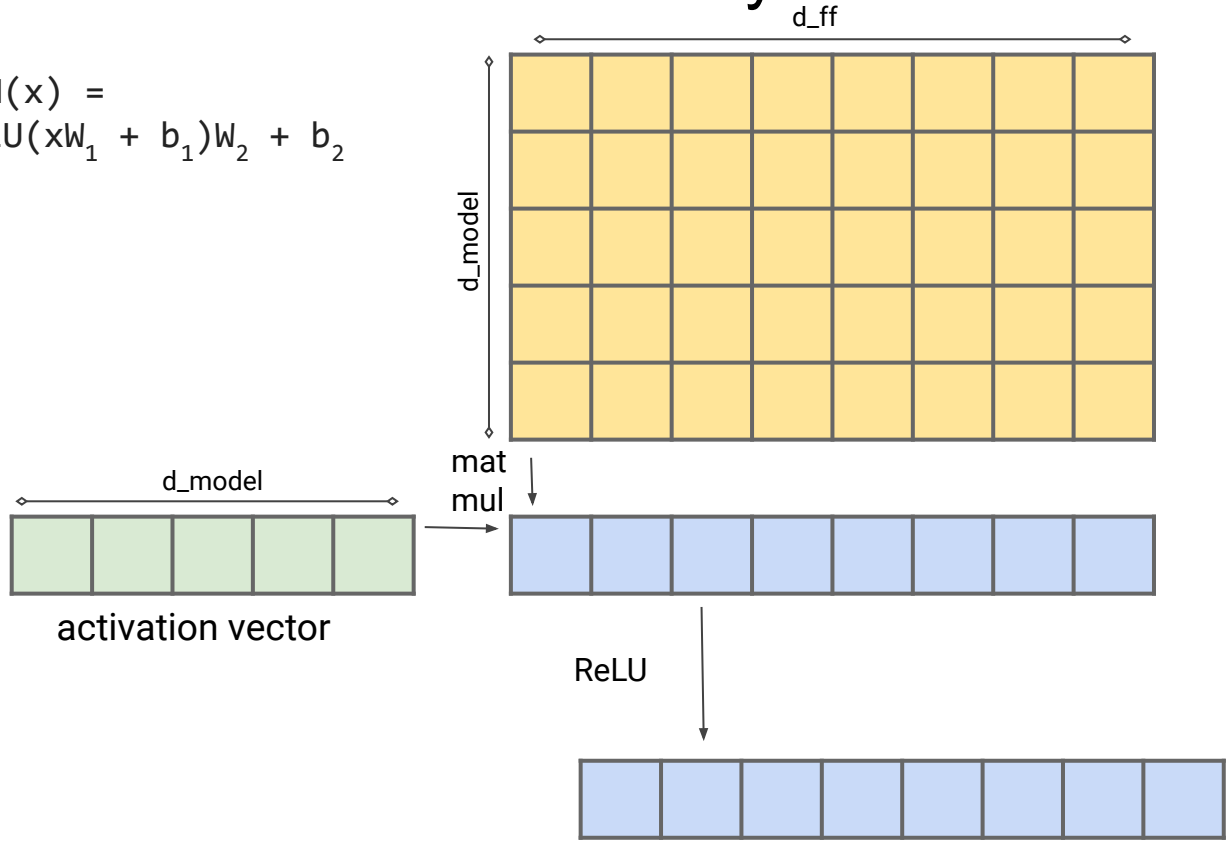
Standard Feed Forward Layer

$$FNN(x) = ReLU(xW_1 + b_1)W_2 + b_2$$



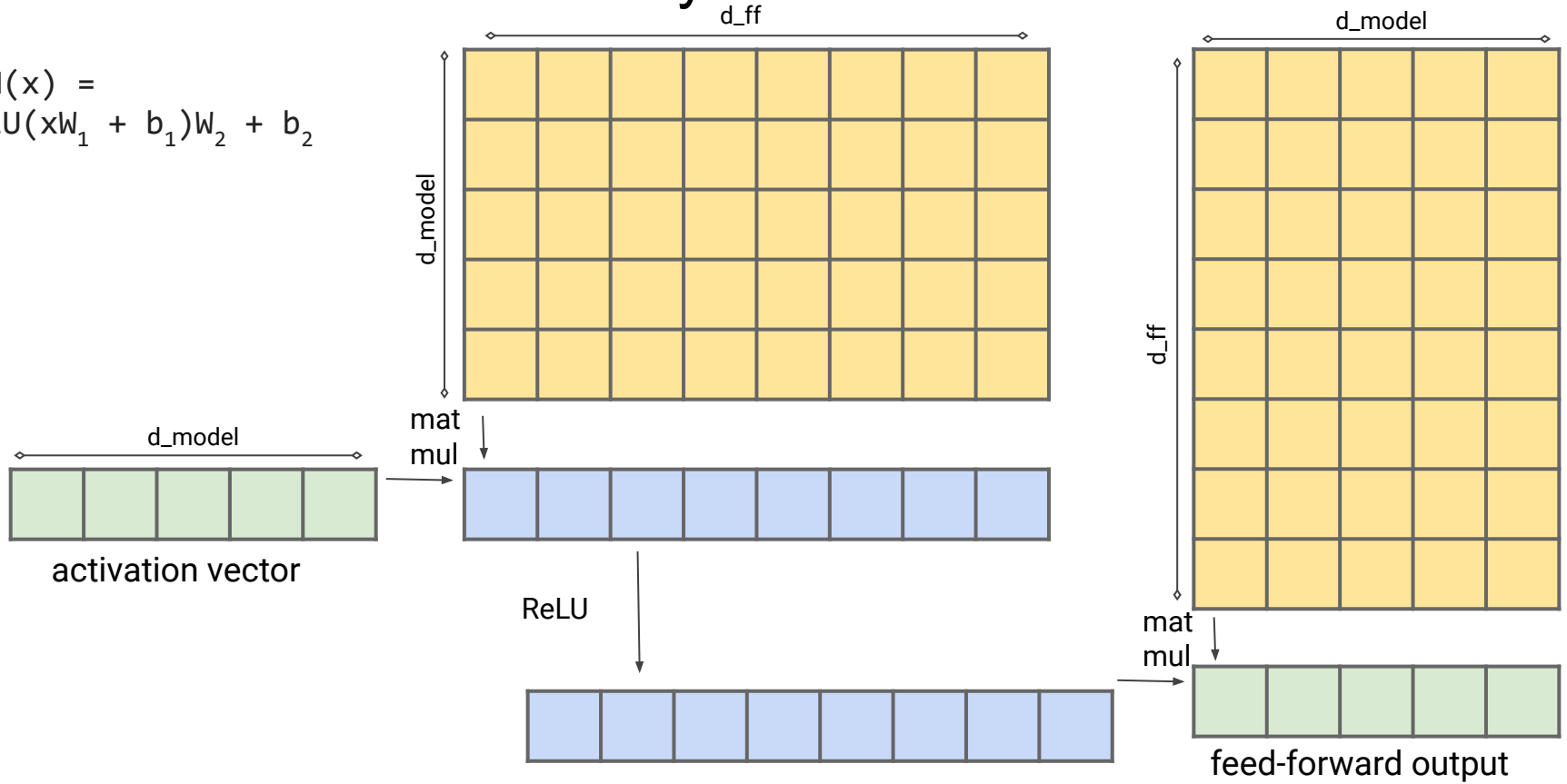
Standard Feed Forward Layer

$$FNN(x) = ReLU(xW_1 + b_1)W_2 + b_2$$



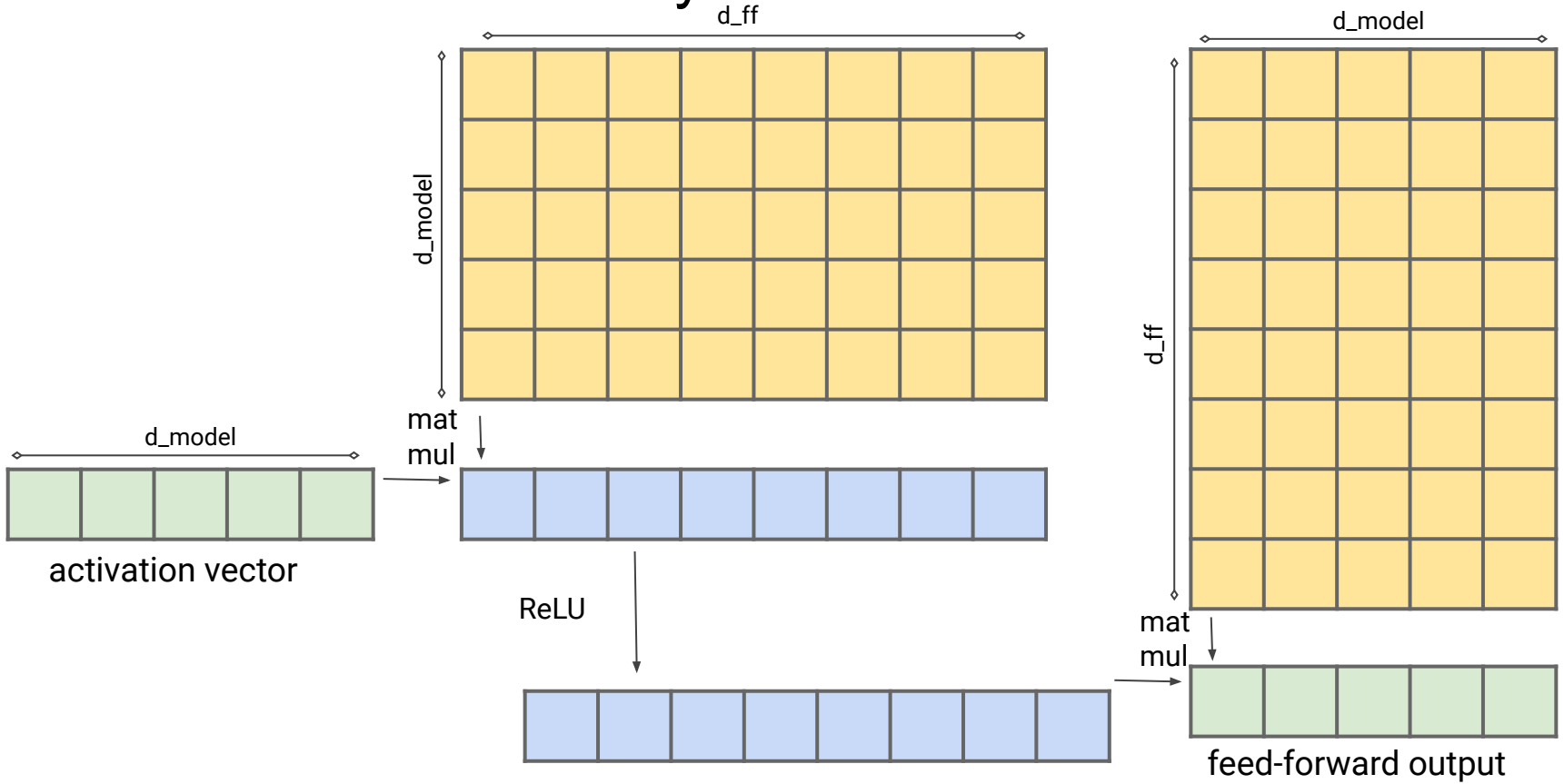
Standard Feed Forward Layer

$$FNN(x) = ReLU(xW_1 + b_1)W_2 + b_2$$

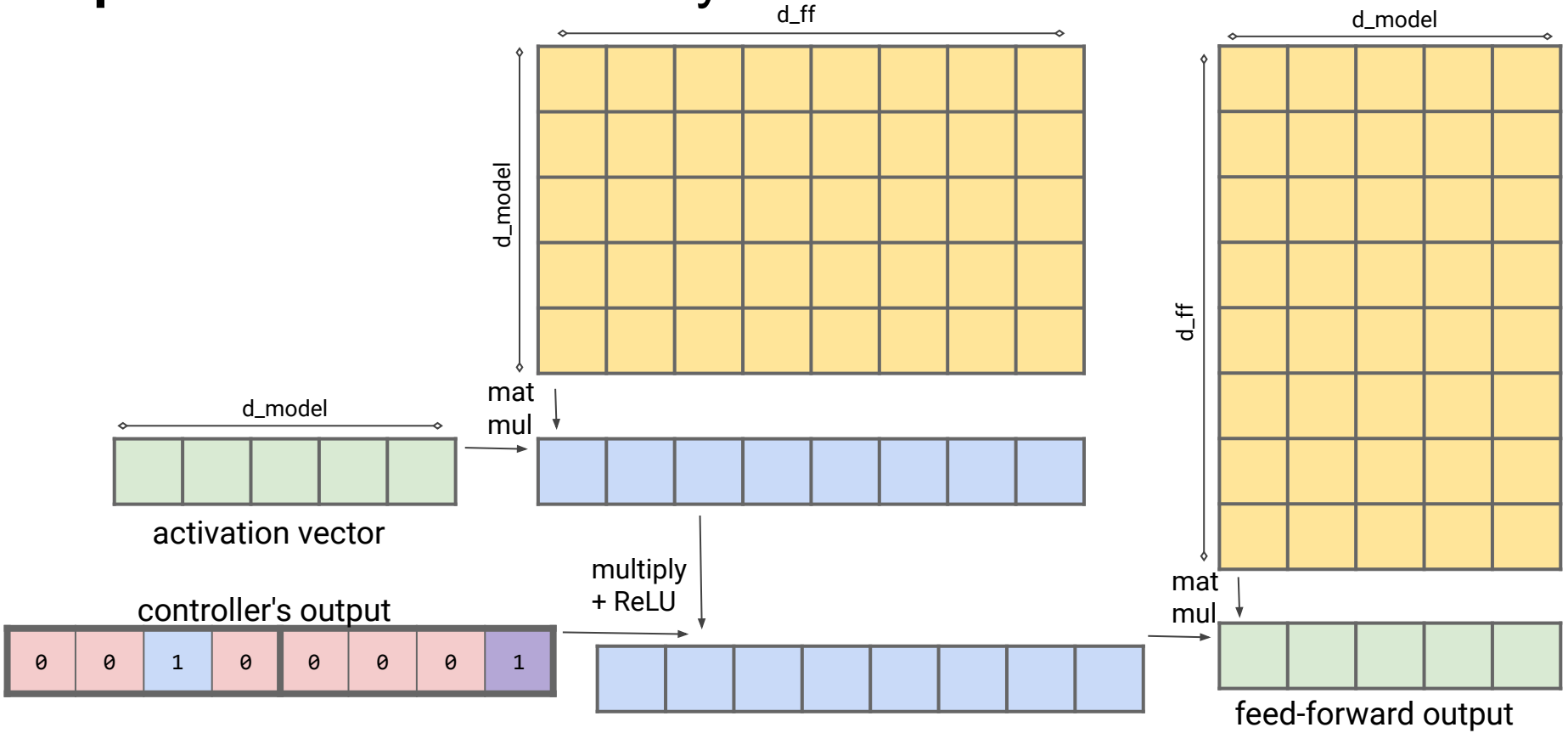


Introducing Fine-Grained Conditional Computation into FF Layer

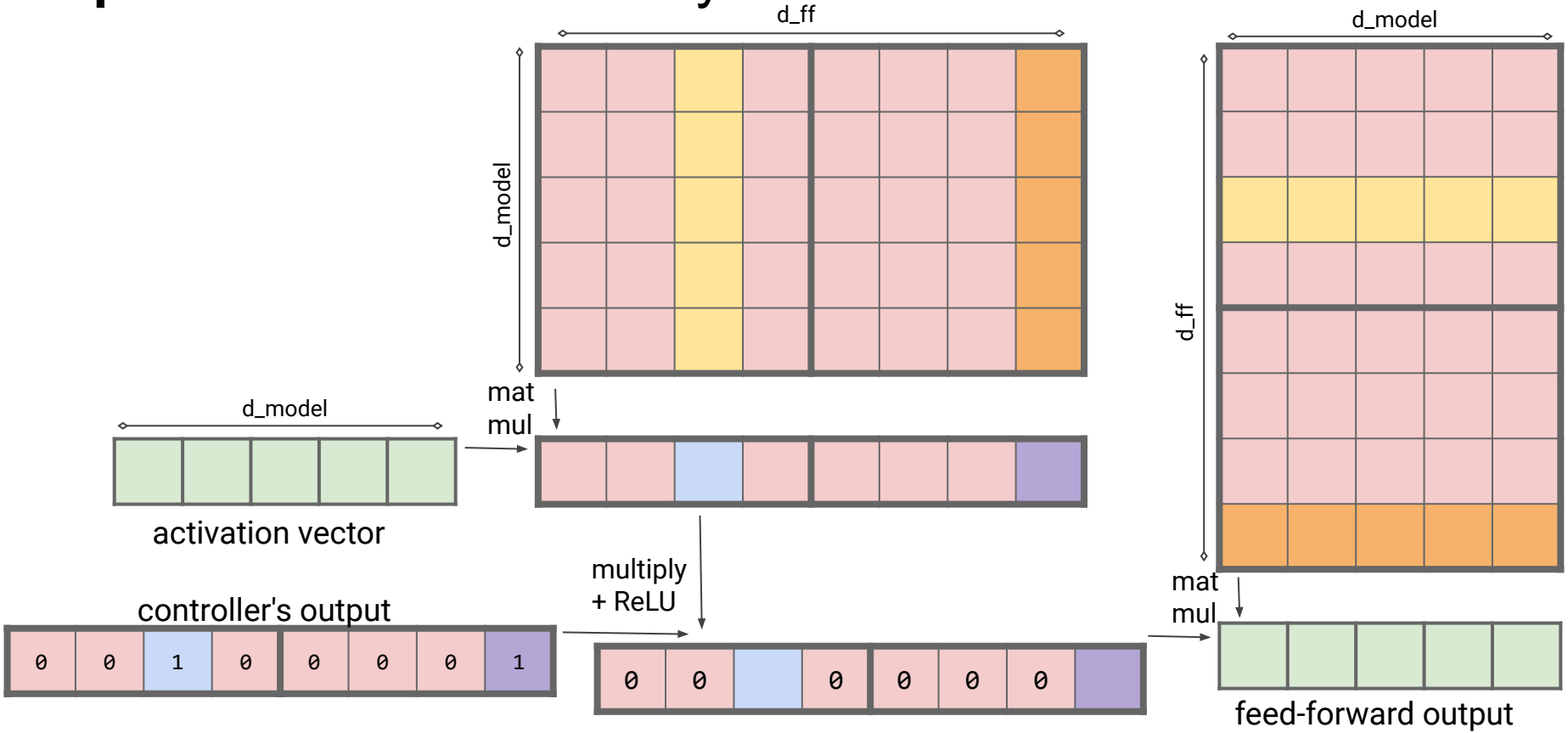
Standard Feed Forward Layer



Sparse Feed Forward Layer



Sparse Feed Forward Layer

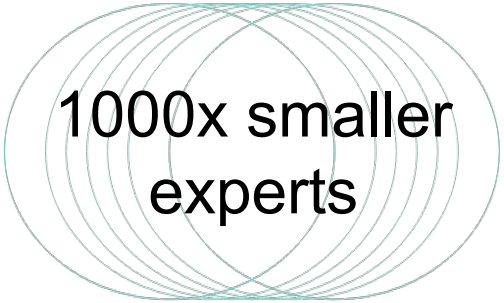


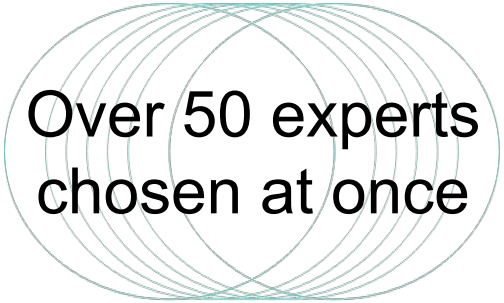
How to design Controller? (briefly)

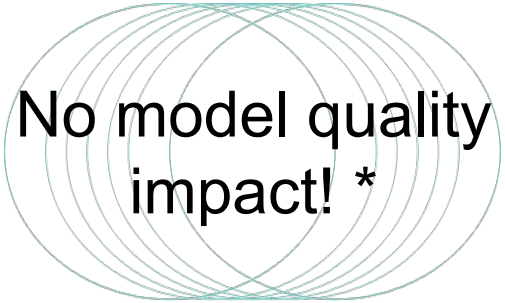
- Low-rank bottleneck
  - It has to be cheap to compute!

- Activation: Straight-Through Gumbel Softmax
  - This makes the output a discrete binary mask!
  - … and still provides gradients!

Comparison to Mixture of Experts





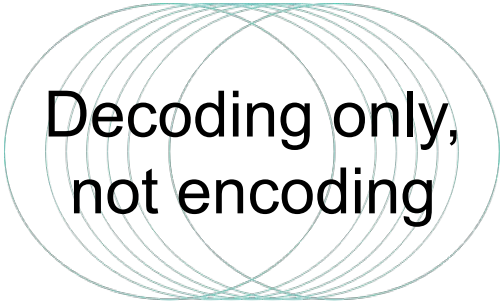


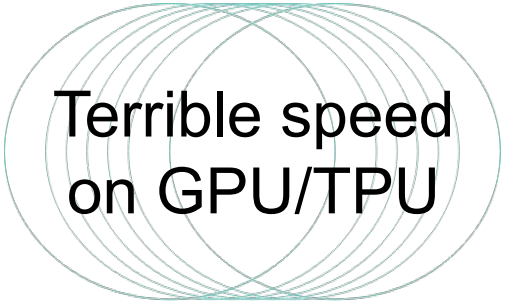
* With the same model size.

Problems and Limitations…

Problems and Limitations

Speed-up is achieved only in specific situations:





Lifting Limitations of Fine-Grained Conditional Computation

Motivation

- Verified hypothesis
  - Skipping parameters without decreasing quality is possible!

- If it works in specific circumstances, it can be generalized.
  - (at least it's likely to be possible)

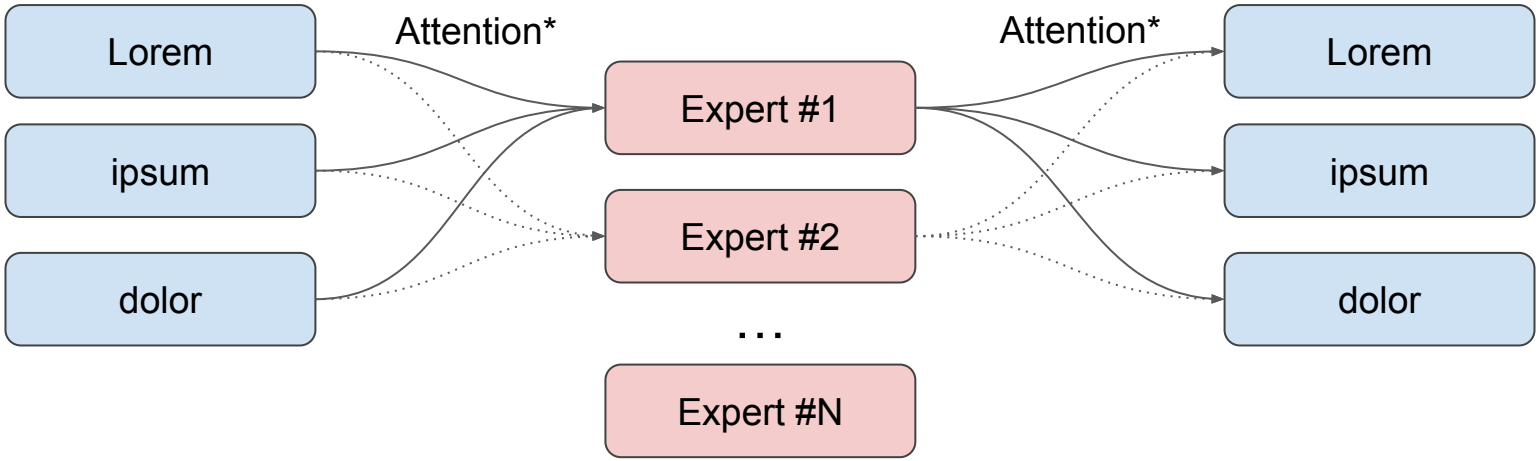
- So, currently I work on generalizing those gains to training and GPU/TPU.

Ideas for improvements…

- We need to route multiple tokens to multiple experts…
- … we need a seemingly sparse operation…
- … implemented efficiently as dense operations on GPU…

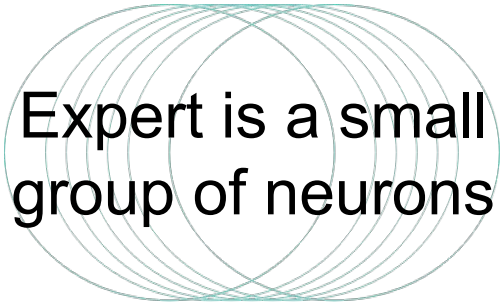
This sounds like the attention mechanism!

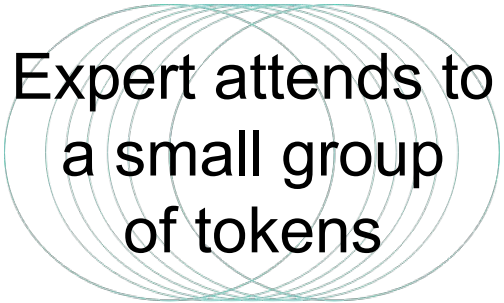
Attention in Fine-Grained Mixture-of-Experts

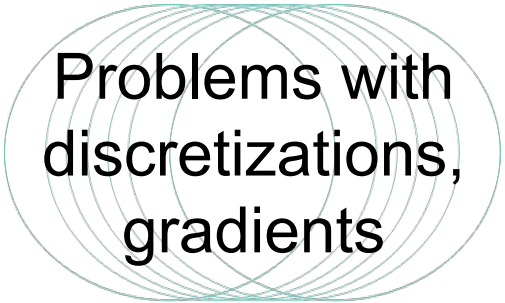


*This is actually just attention-**inspired** mechanism

Implementation " " " details " " "







There is much more - too much to cover completely

Results so far…

No presentable numbers…

- Similar speed-up of FF layer is possible during training on GPU!
  - 3x for small models, around 20x for larger models.
- However, the speedup of the whole model is small, because bottlenecks.
- So far it seems to be an improvement only early during training.

Stay tuned for more results in the near future!

Fine-Grained Conditional Computation in Transformers

Speaker: Sebastian Jaszczur, University of Warsaw & IDEAS NCBR
PhD Advisor: Marek Cygan

ML in PL Conference, November 2022

IDEAS NCBR