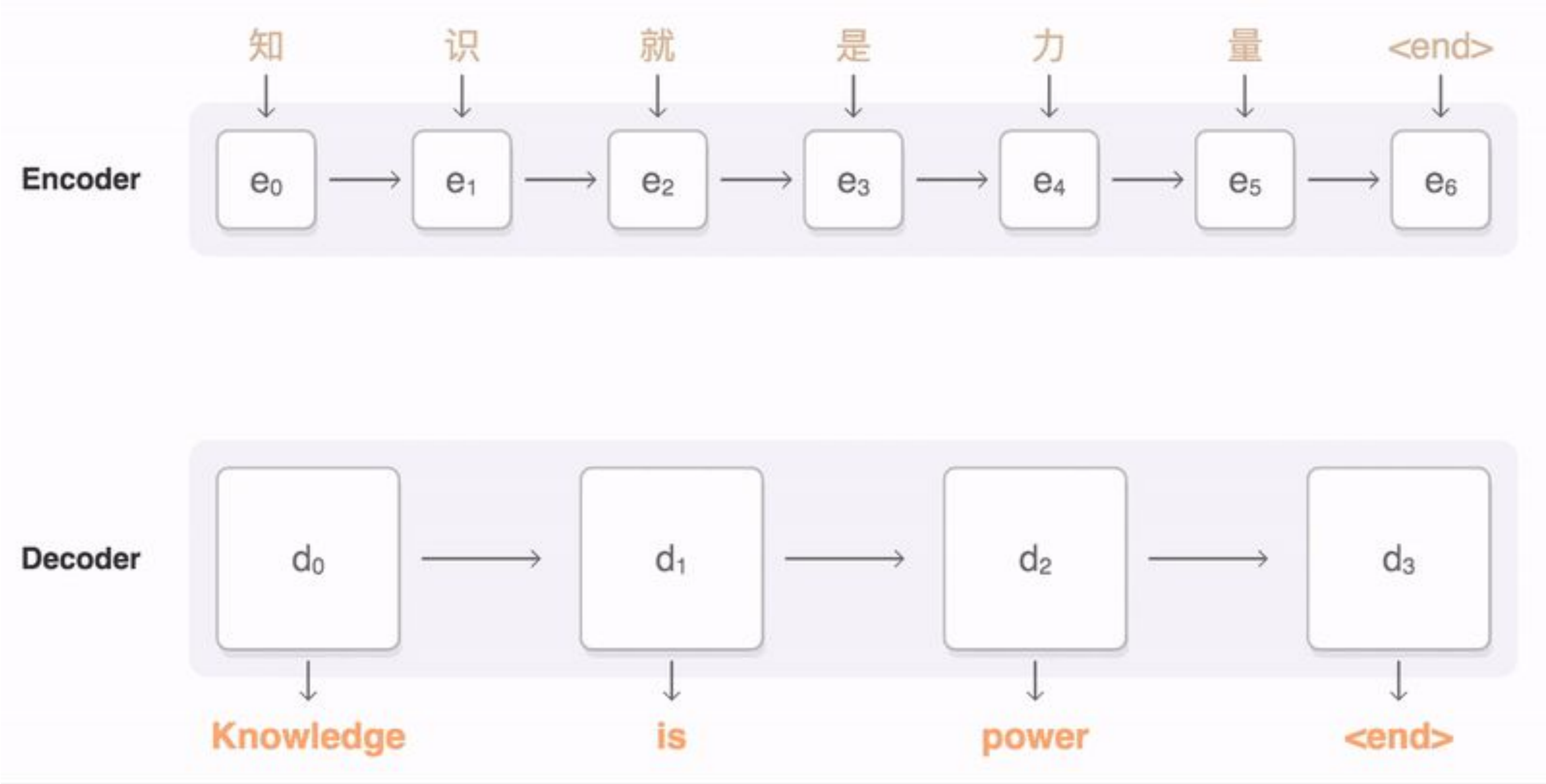
Google Research

Efficient Transformers

Nikita Kitaev, Łukasz Kaiser and Anselm Levskaya, Sebastian Jaszczur, Aakanksha Chowdhery, Afroz Mohiuddin, Wojciech Gajewski, Henryk Michalewski, Jonni Kanerva

Long long time go: RNNs Everywhere



But:

The Transformer

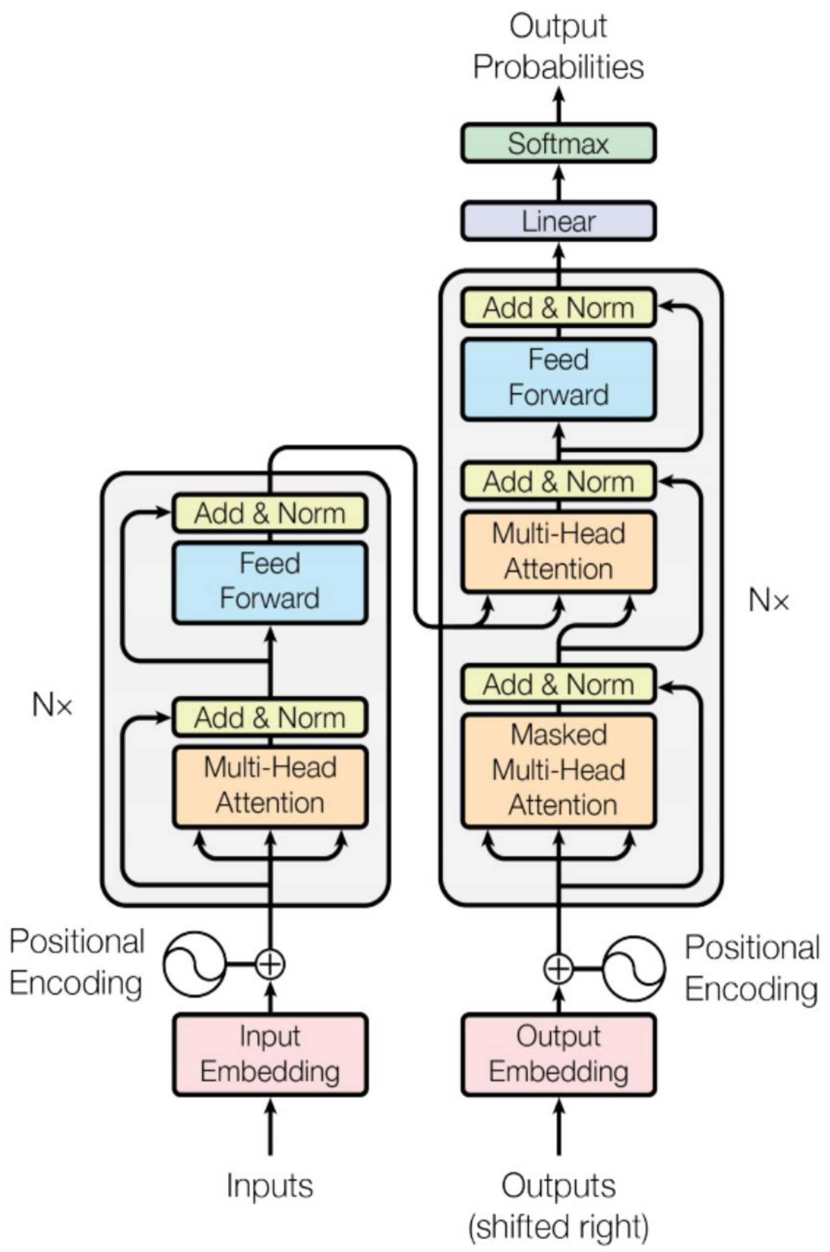


Figure 1: The Transformer - model architecture.

Machine Translation Results: WMT-14

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [17]	23.75			
Deep-Att + PosUnk [37]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [36]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [31]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [37]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [36]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	~~**28.4**~~ 29.1	~~**41.0**~~ 41.8	$2.3 \cdot 10^{19}$	

How about other NLP tasks?

BERT = Bidirectional Encoder Representations from Transformers

GLUE is a set of NLP tasks, we measure average score (higher is better)

• CBOW (bag of words)	58.6
• BiLSTM + Attention	65.6
• BiLSTM + ELMo + Attention	70.0
• BERT	80.5
• Human Baselines	87.1
• ALBERT	89.4

Transformer

Google Research

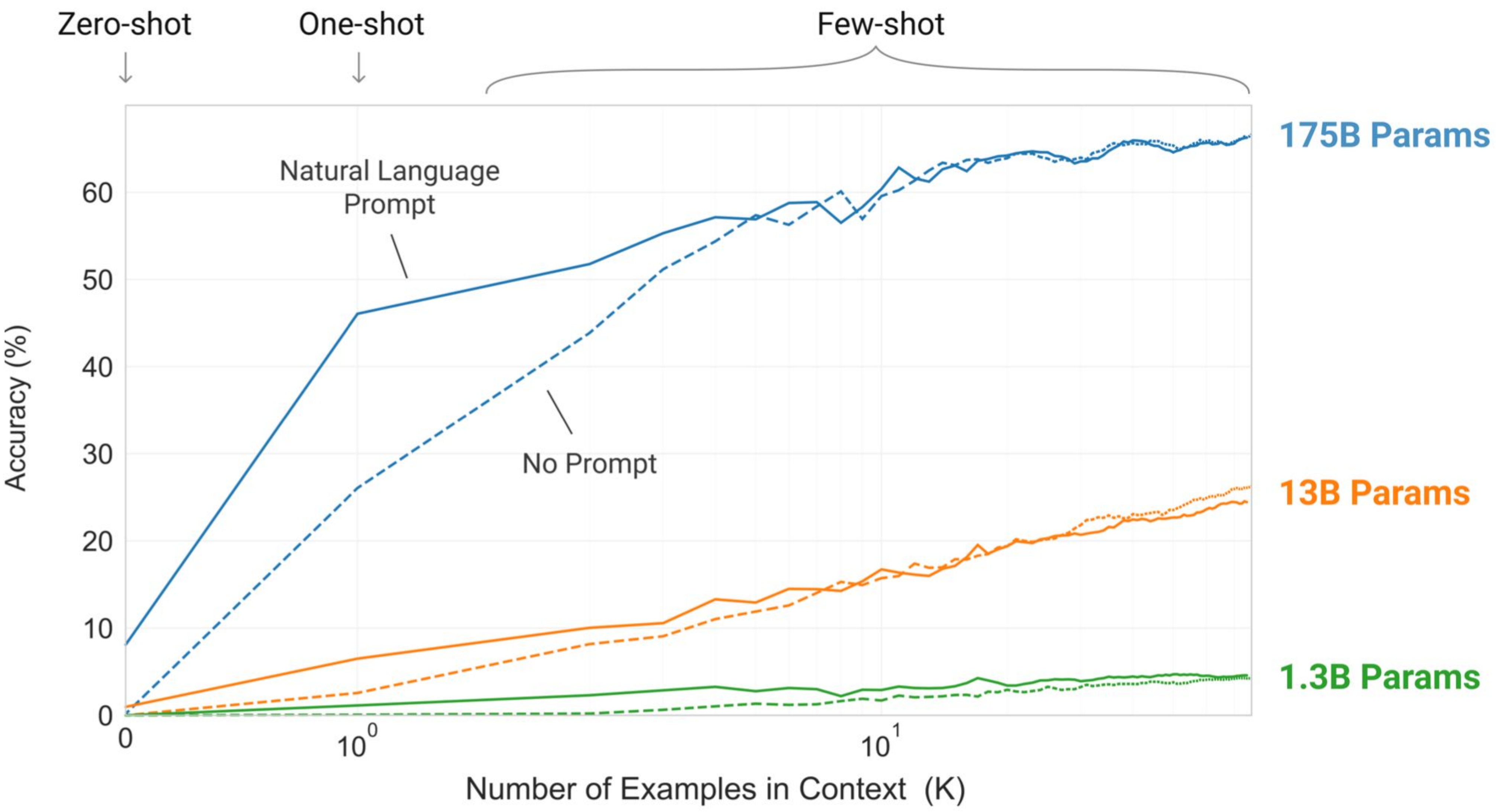
From the BERT documentation:

> Using the default training scripts (`run_classifier.py` and `run_squad.py`), we benchmarked the maximum batch size on single Titan X GPU (12GB RAM) with TensorFlow 1.11.0:

System	Seq Length	Max Batch Size
`BERT-Large`	64	12
...	128	6
...	256	2
...	320	1
...	384	0
...	512	0

ZERO!

GPT3



Zero-shot
One-shot
Few-shot
Natural Language
Prompt
No Prompt
175B Params
13B Params
1.3B Params
Accuracy (%)
60
50
40
30
20
10
0
0
10^0
10^1
Number of Examples in Context (K)

Outlook

In the near future, it will be impossible to even fine-tune state of the art models without datacenter-scale hardware resources.

Outlook

~~In the near future, it will be impossible to even fine tune state of the art models without datacenter-scale hardware resources.~~

Transformers can be adapted to run on today's hardware over entire chapters or documents of text -- up to 1 million tokens at a time.

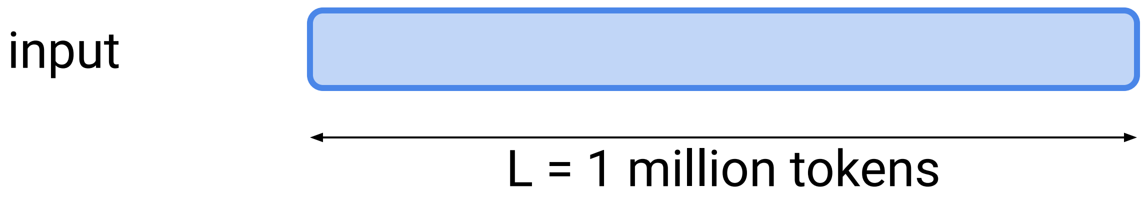
Moreover, the model should run on a single GPU or TPU device.

Efficiency Challenges

- Memory Efficiency
  - Reduce memory usage with reversible residual layers, as in RevNet [Gomez+ 17]
  - Efficiently train with memory swapping to CPU and quantization
- Time Complexity
  - Introduce fast attention with locality sensitive hashing (LSH)
- Need to activate all weights for each token
  - Sparse layers that allow selective activations

Memory Efficiency

Memory Efficiency



Google Research

Memory Efficiency

input

$d_{model} = 512$

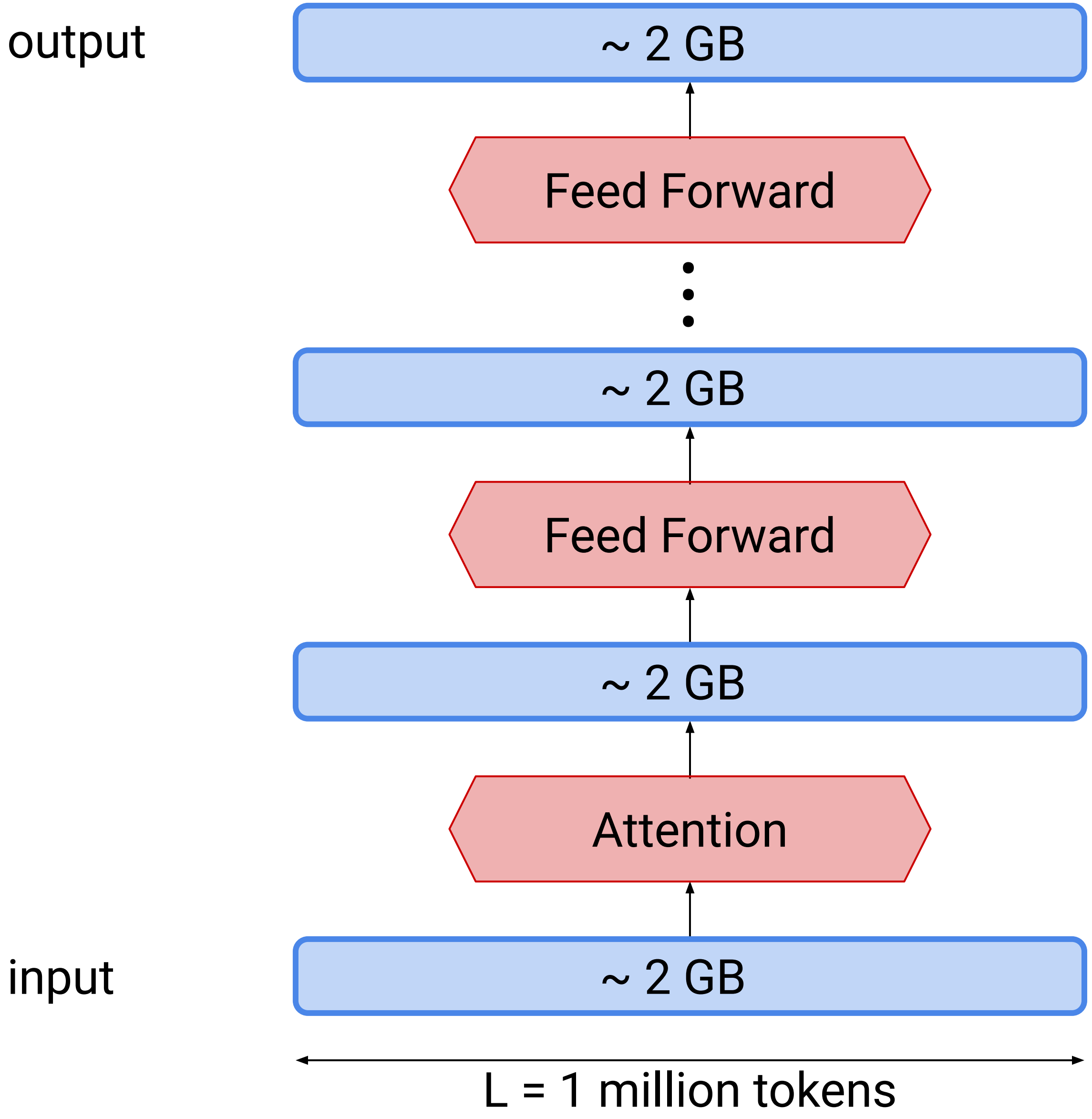
L = 1 million tokens

Memory Efficiency

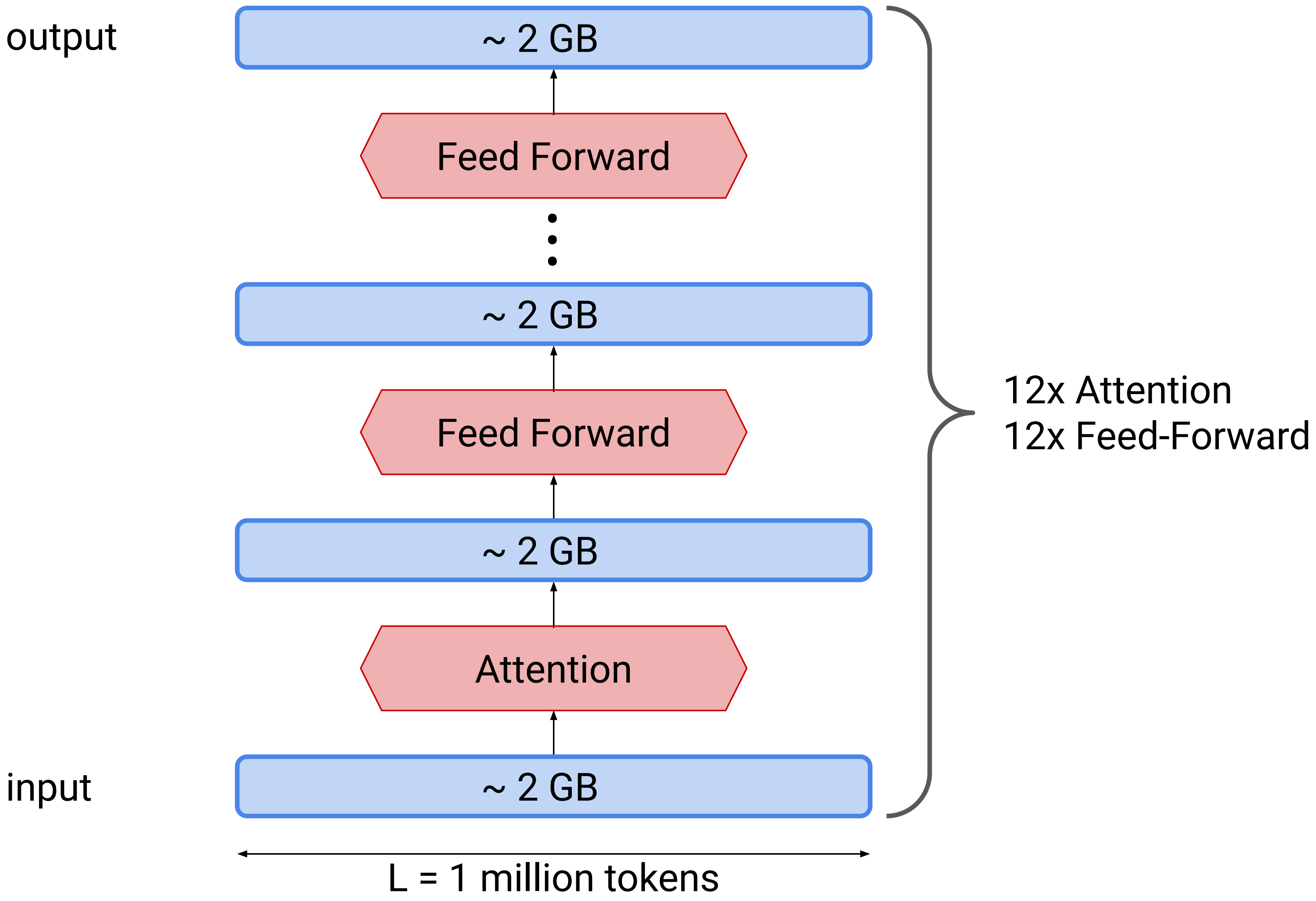
input ~ 2 GB $d_{model} = 512$

L = 1 million tokens

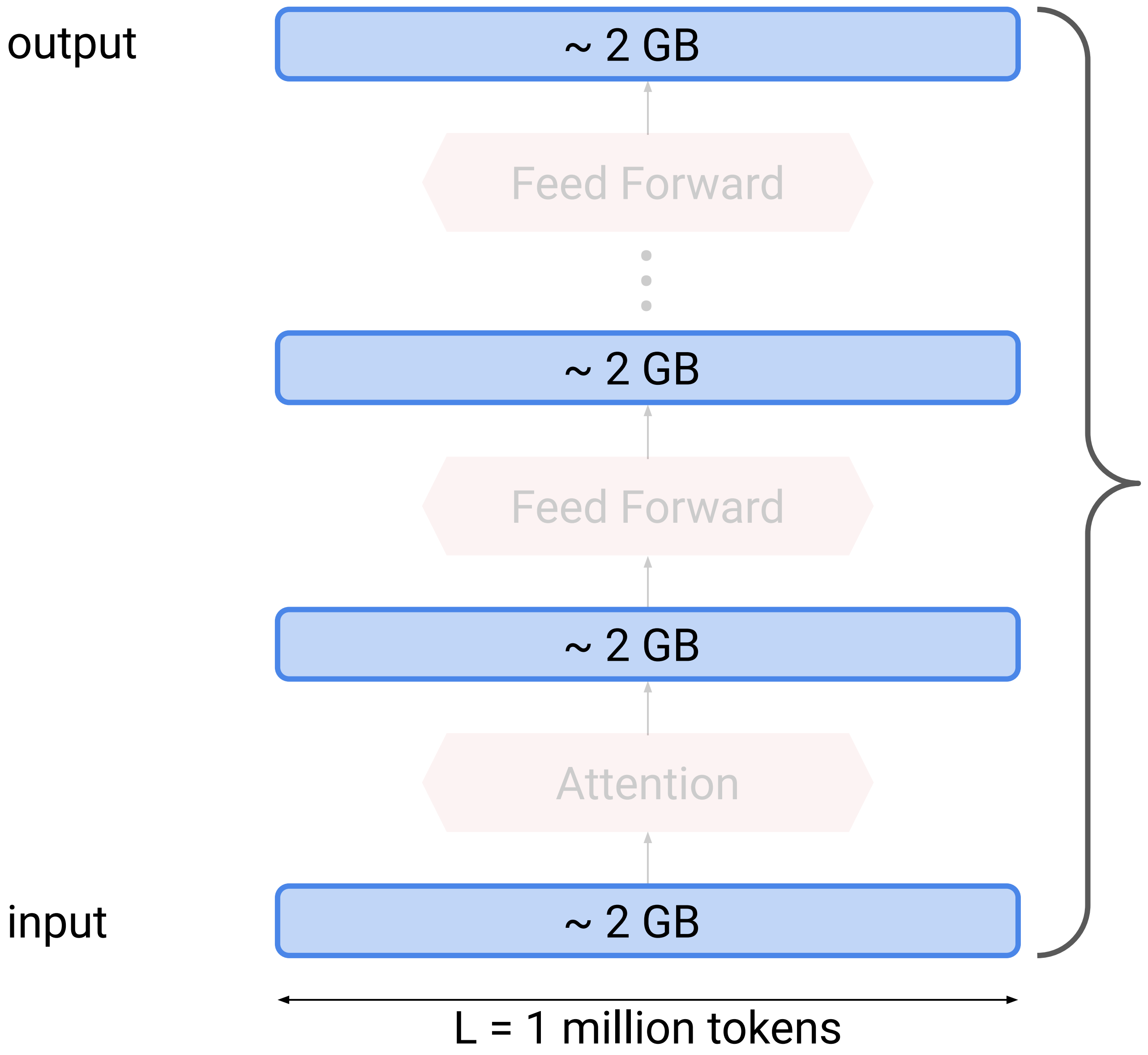
Memory Efficiency



Memory Efficiency



Memory Efficiency

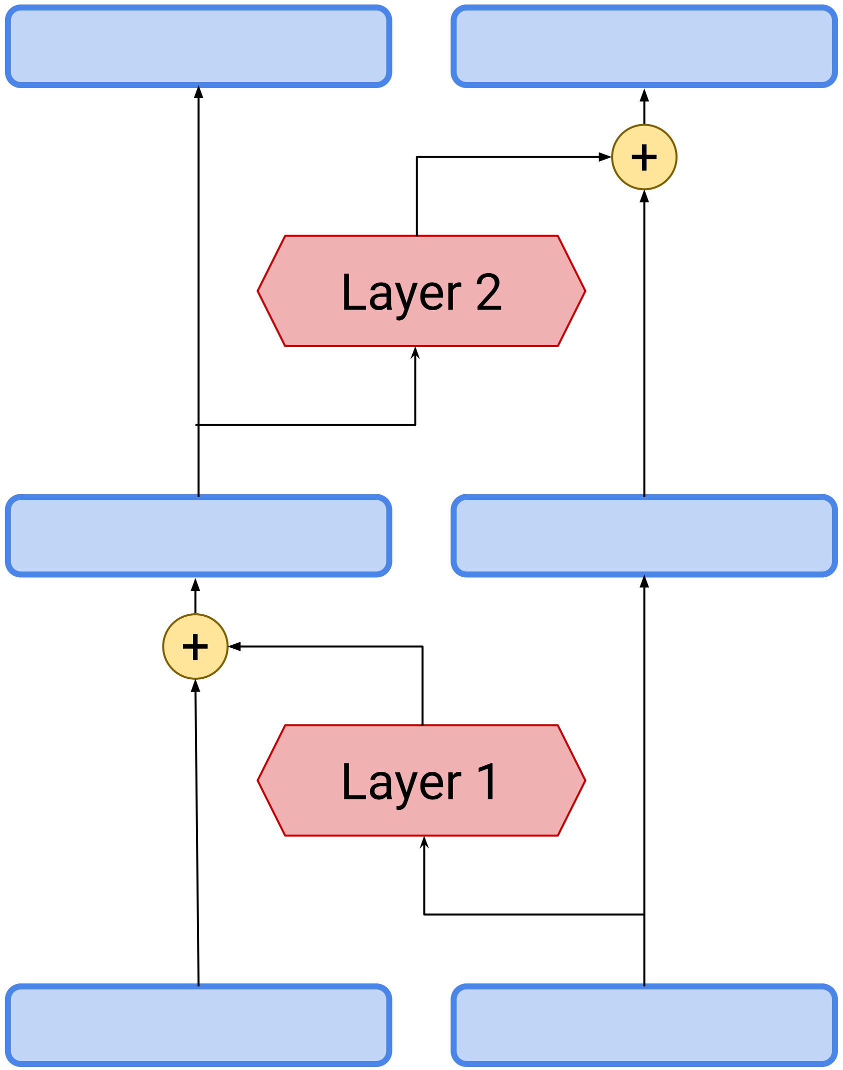


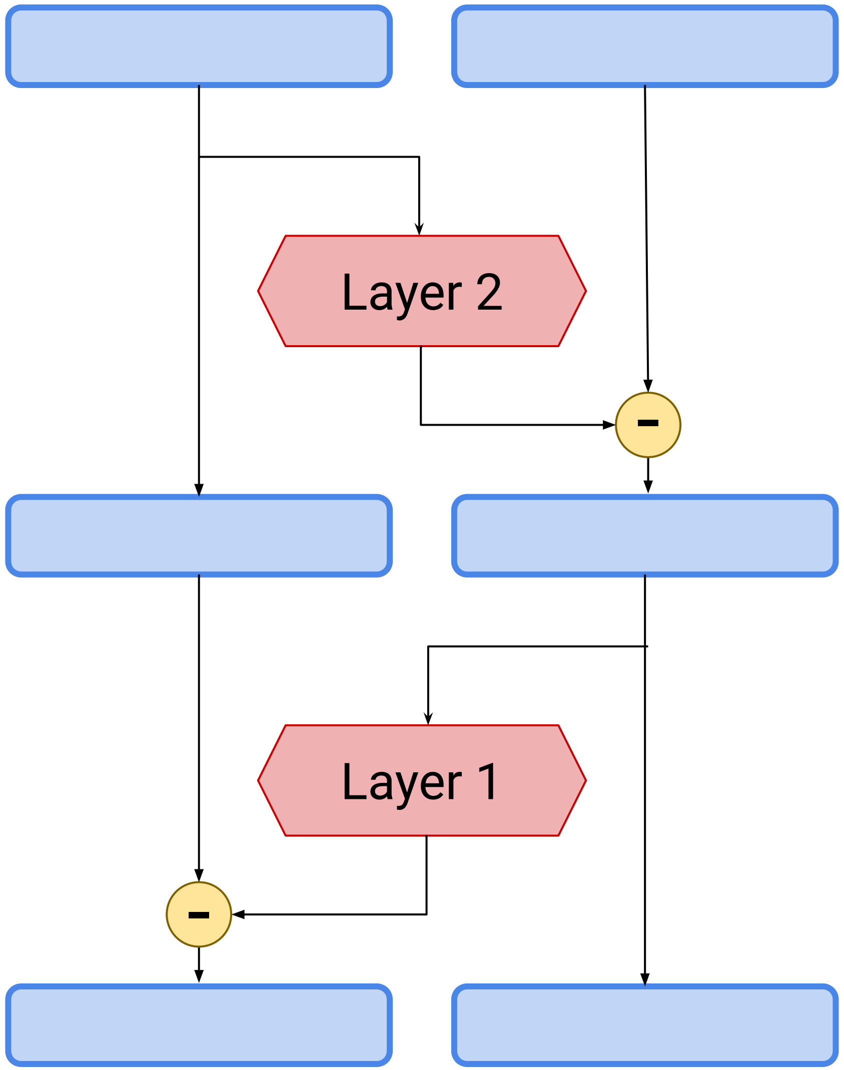
12x Attention
12x Feed-Forward

50 GB total

Google Research

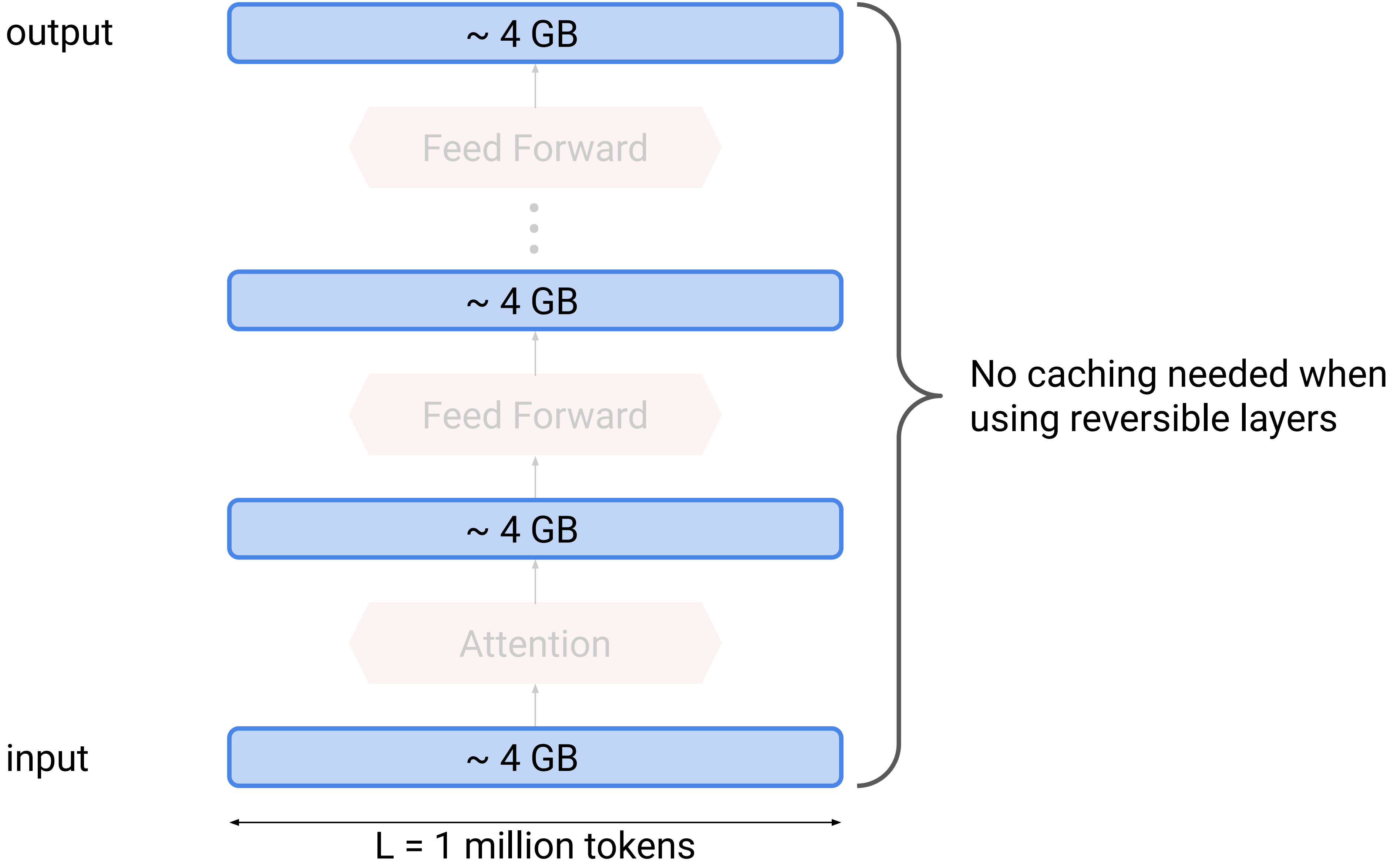
Memory Efficiency: RevNets



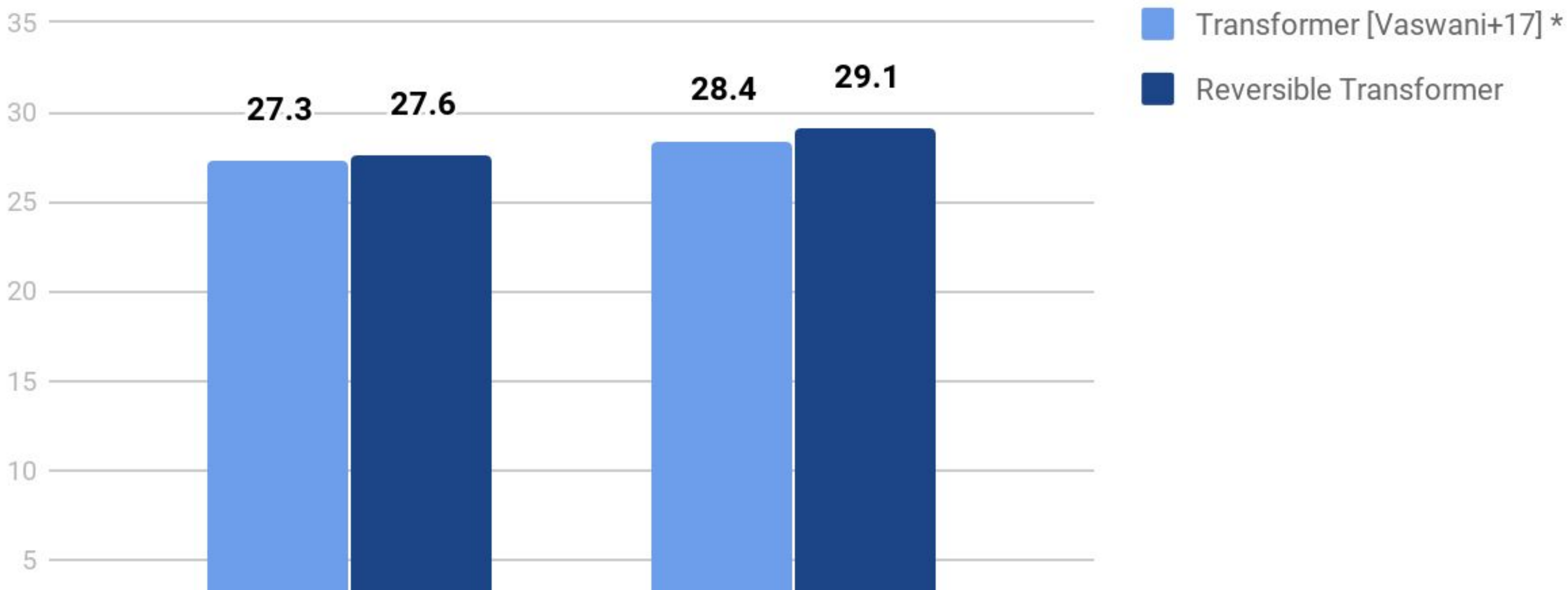


Memory Efficiency

Google Research



Reversible Transformer: BLEU Scores on WMT English-German

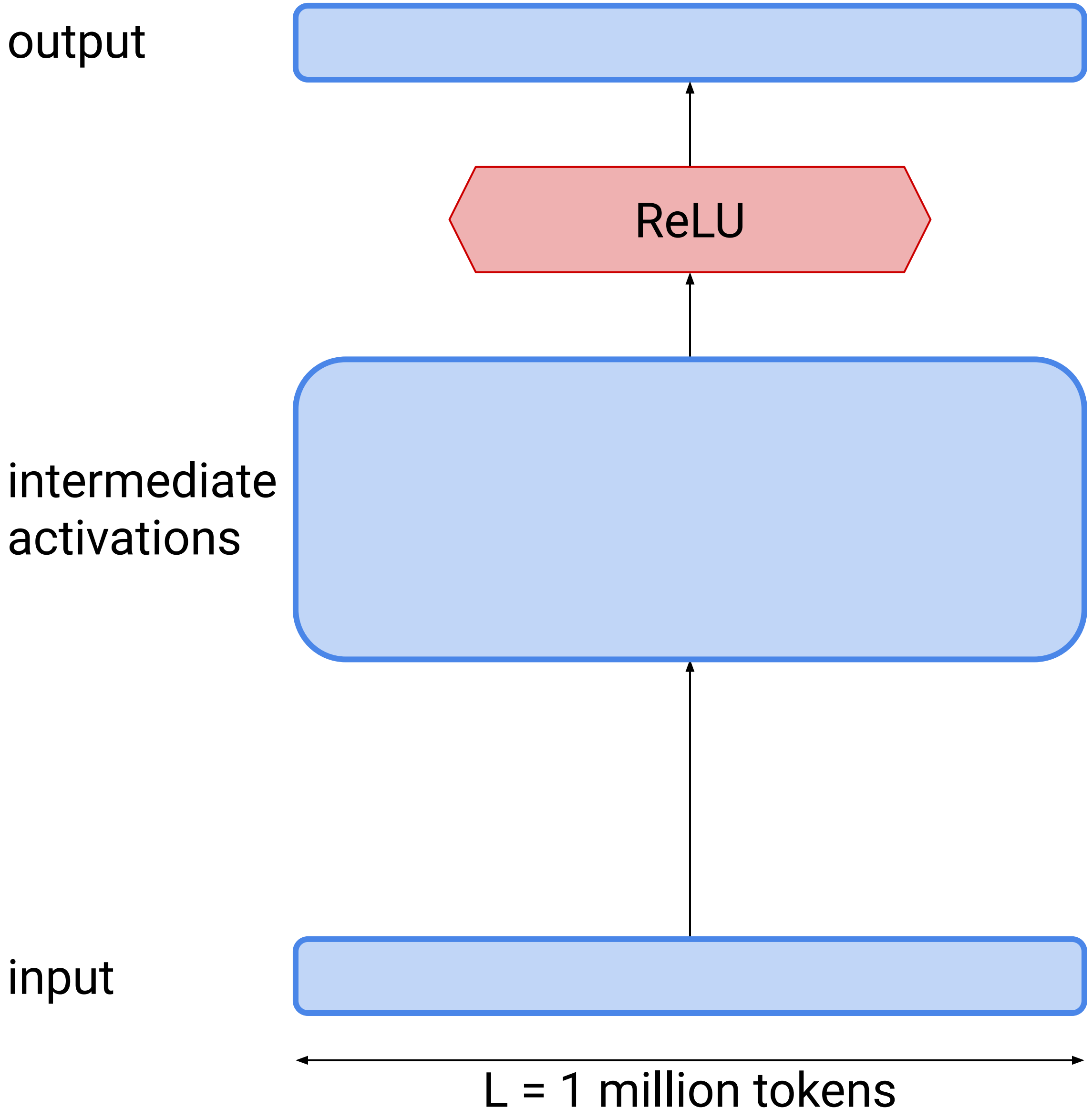


* original reported numbers; differences in BLEU from the Reversible Transformer are likely due to hyperparameter tuning

Time Complexity

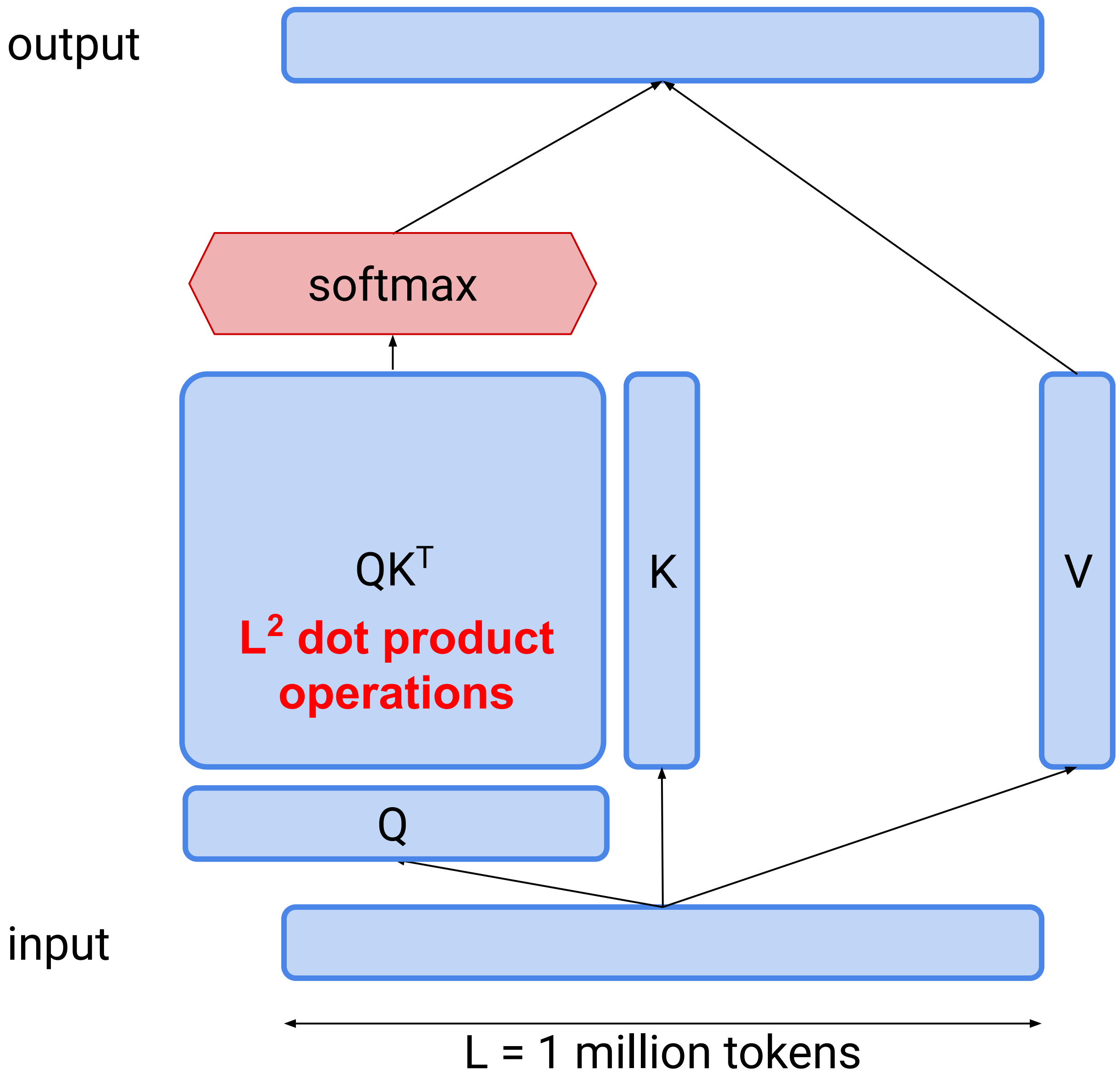
Google Research

Time Complexity: Feed Forward

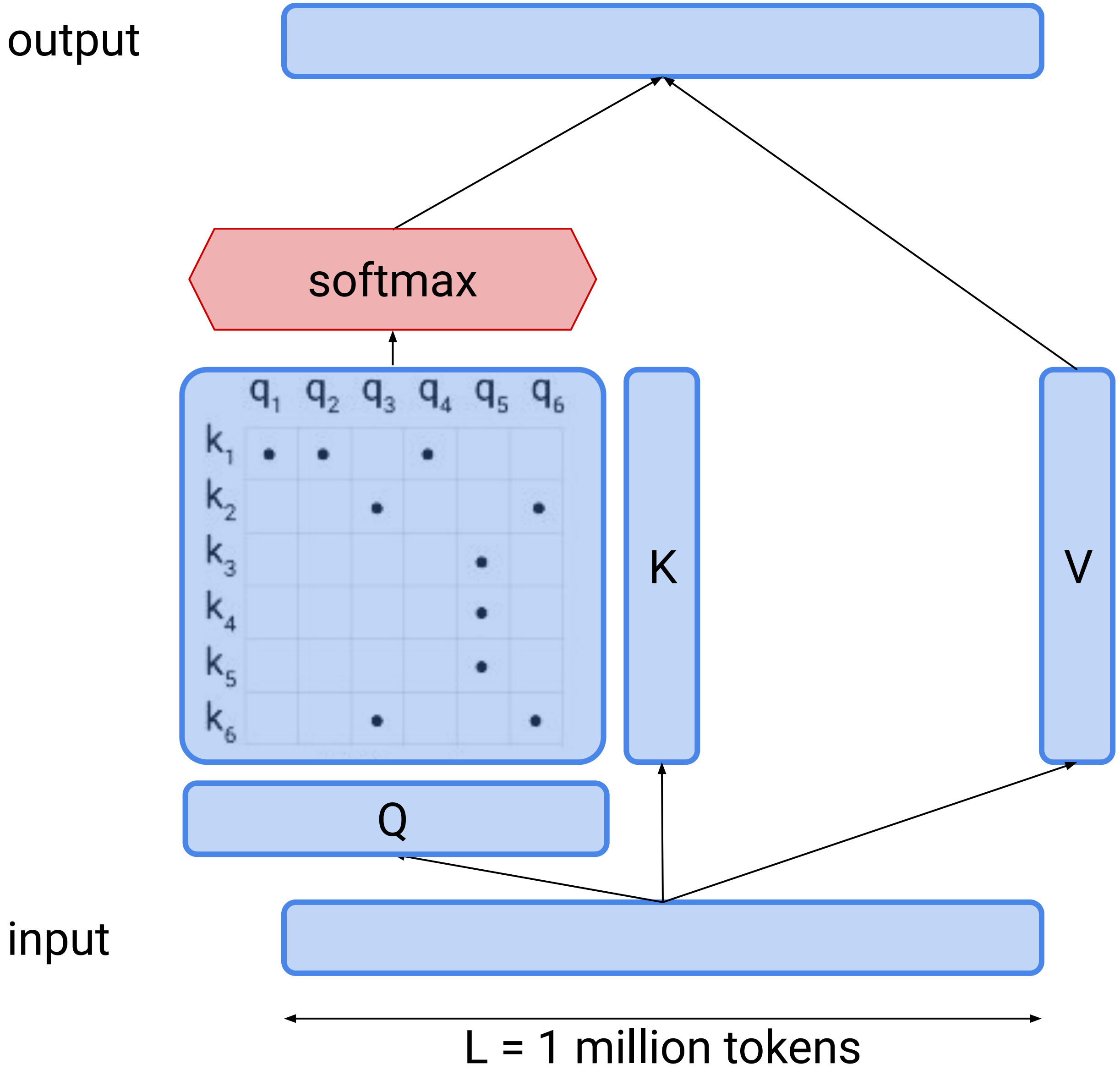


Linear: O(L)

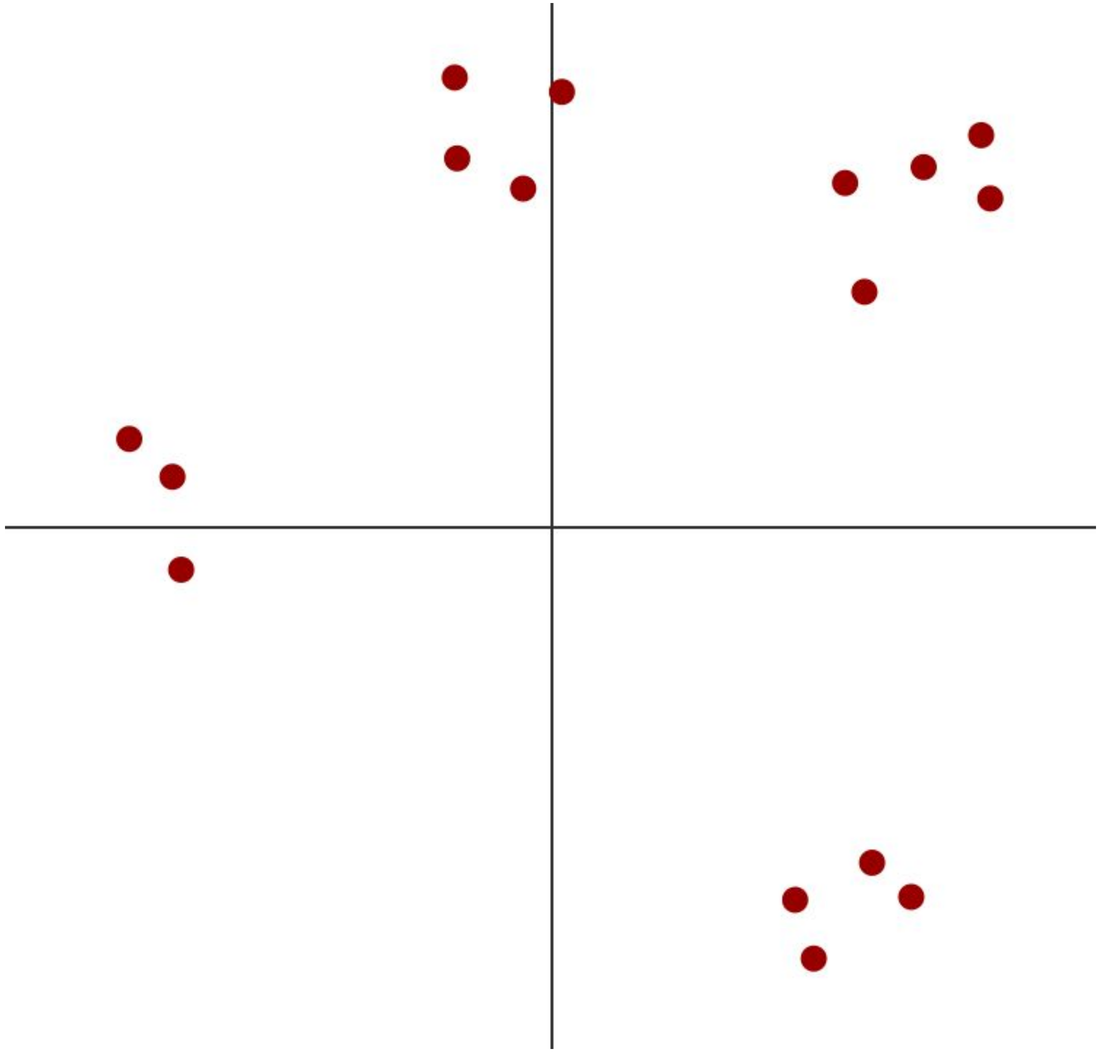
Time Complexity: Attention



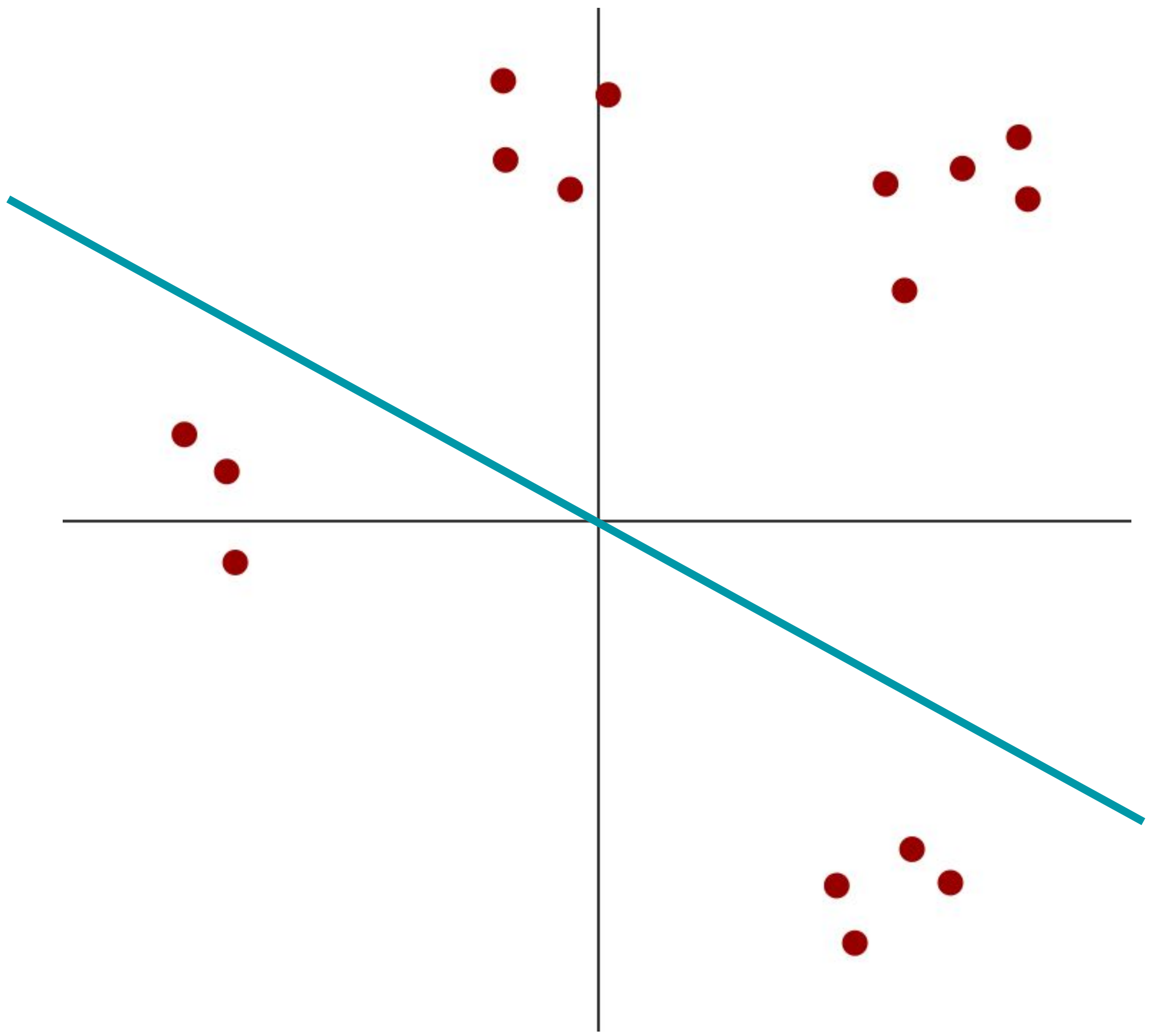
Attention is Sparse



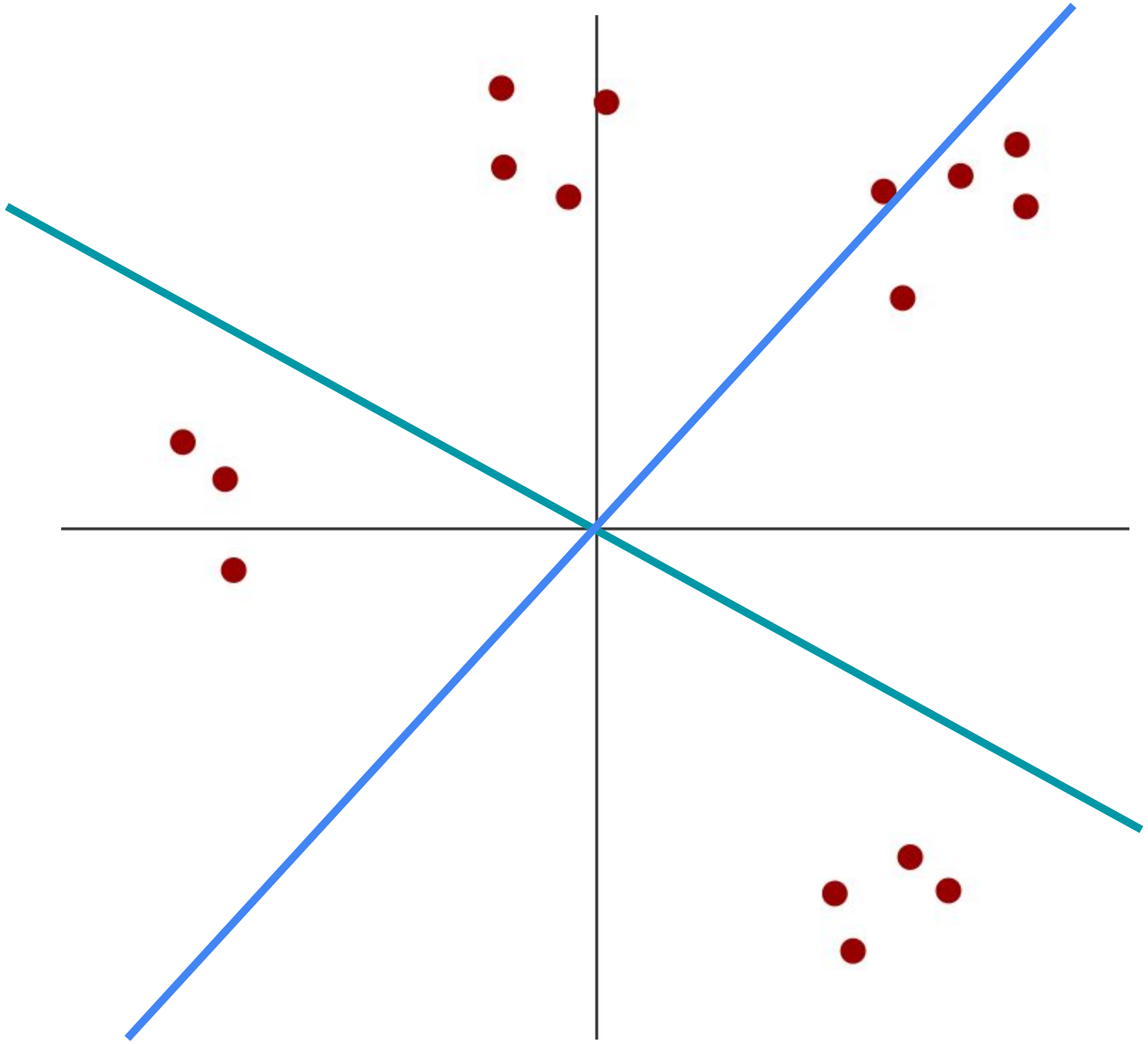
Locality Sensitive Hashing (LSH)



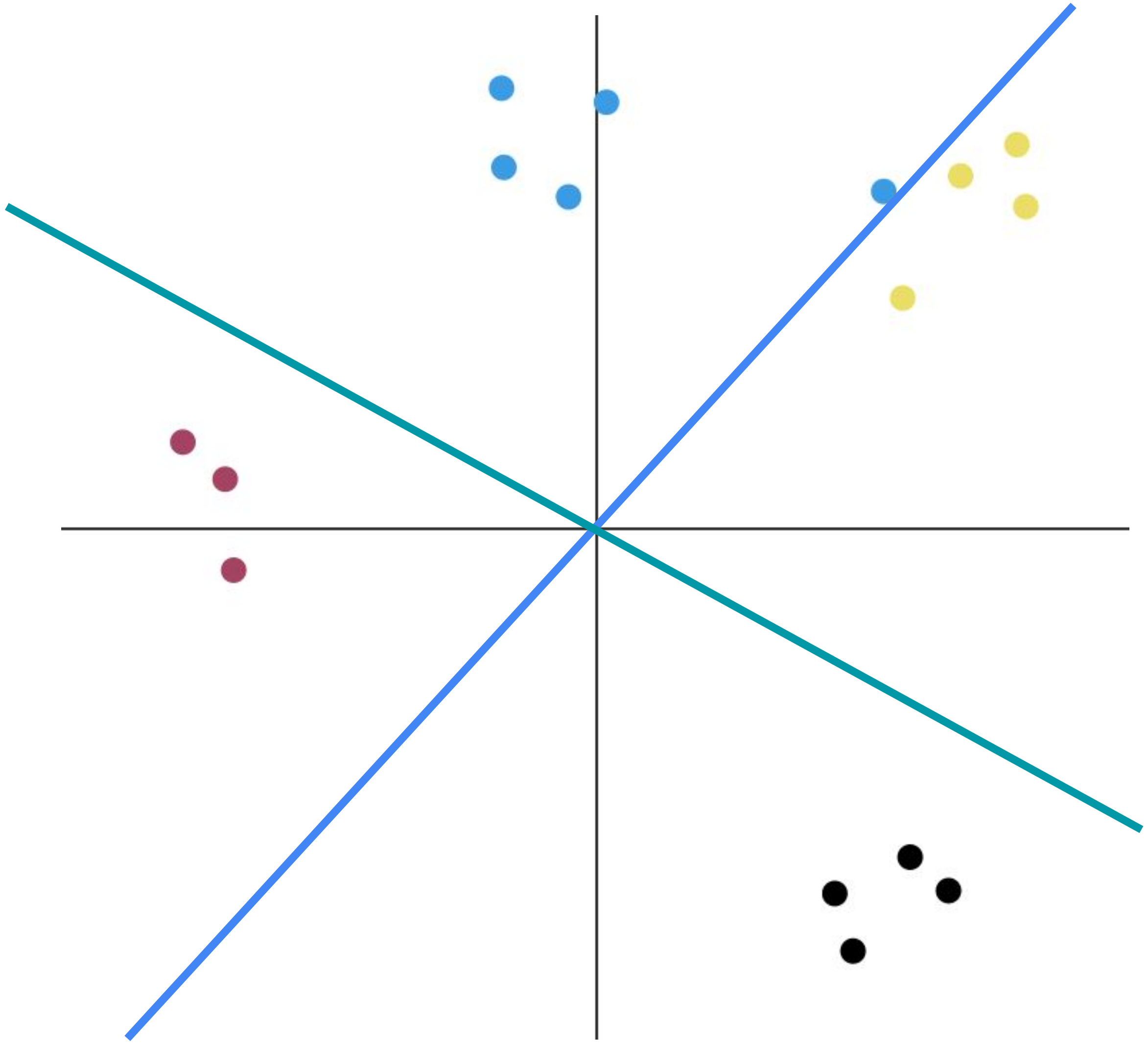
Locality Sensitive Hashing (LSH)



Locality Sensitive Hashing (LSH)

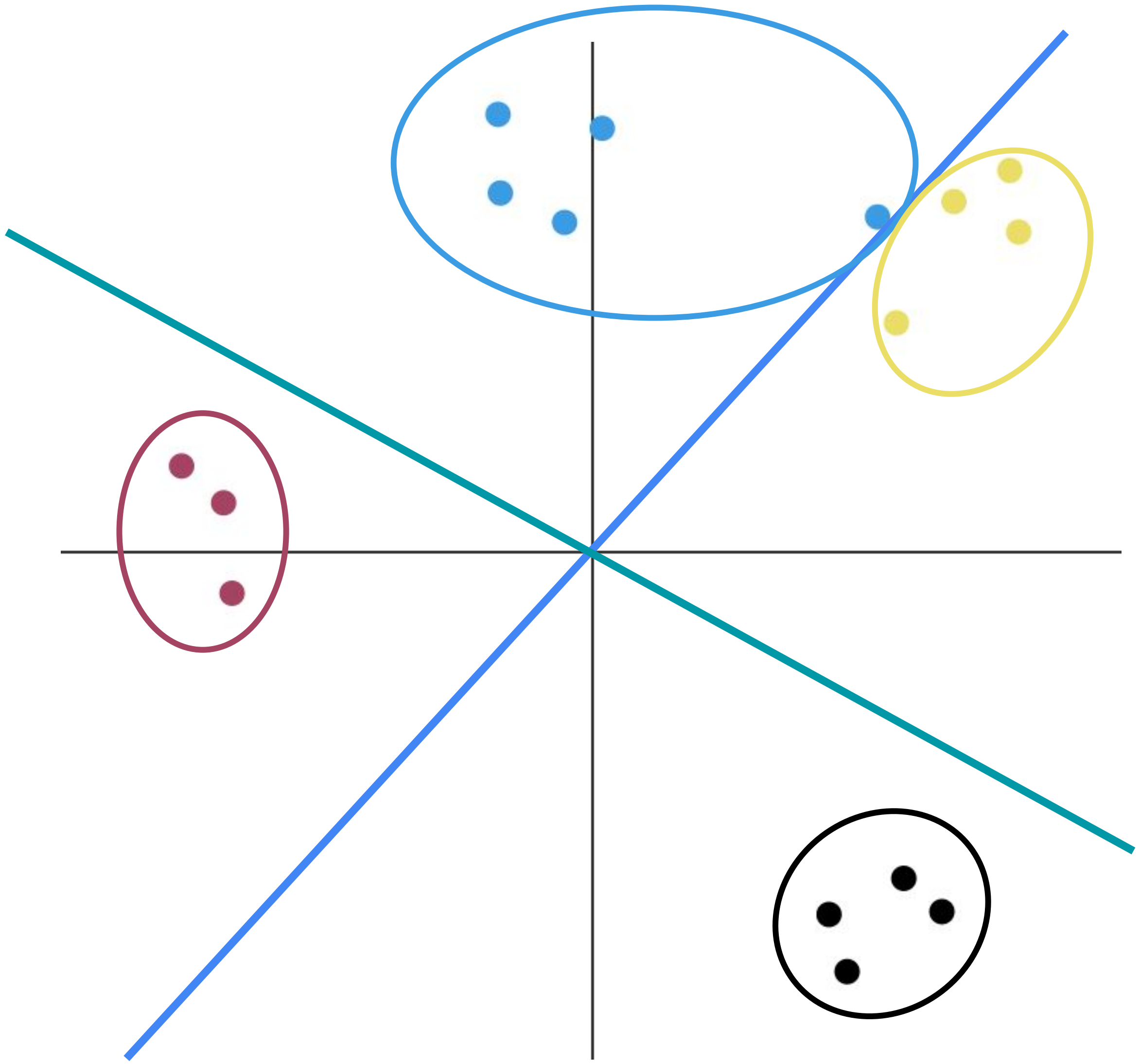


Locality Sensitive Hashing (LSH)



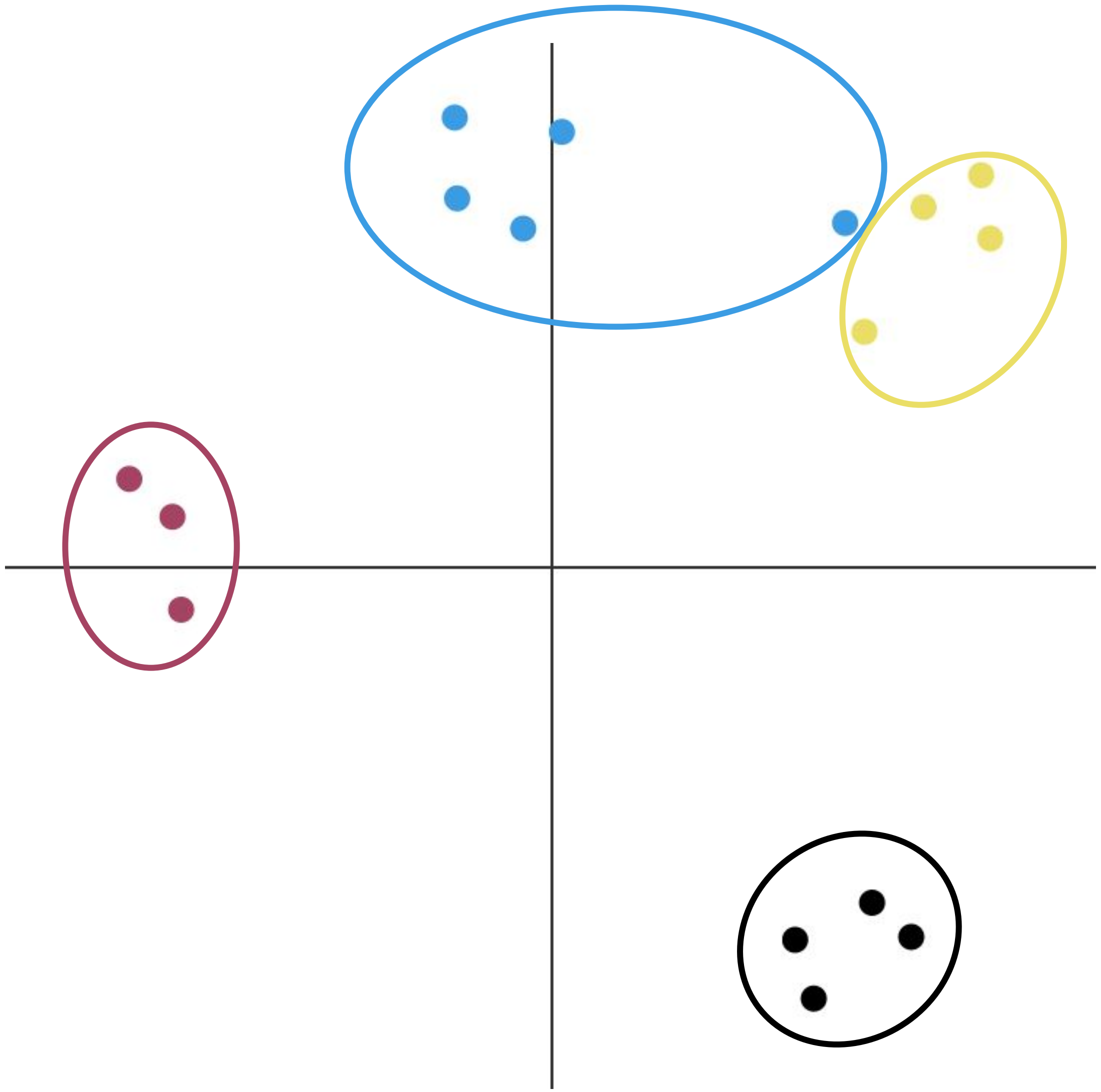
Google Research

Locality Sensitive Hashing (LSH)



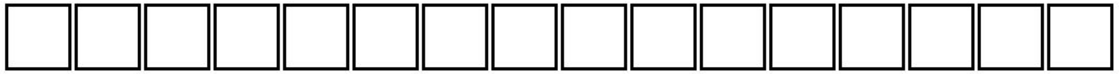
Google Research

Locality Sensitive Hashing (LSH)



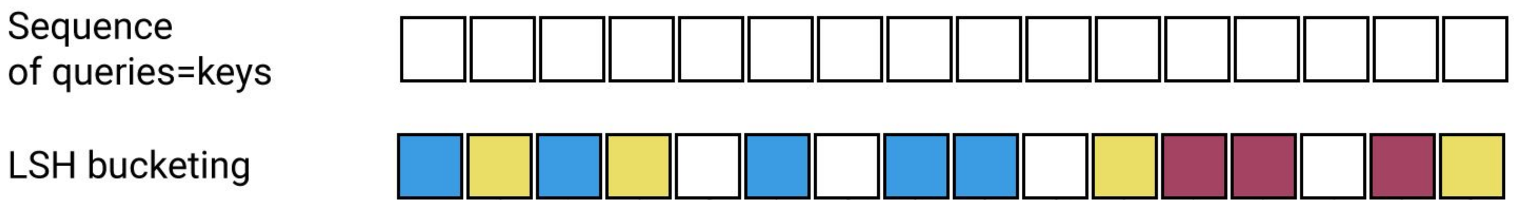
LSH Attention

Sequence
of queries=keys

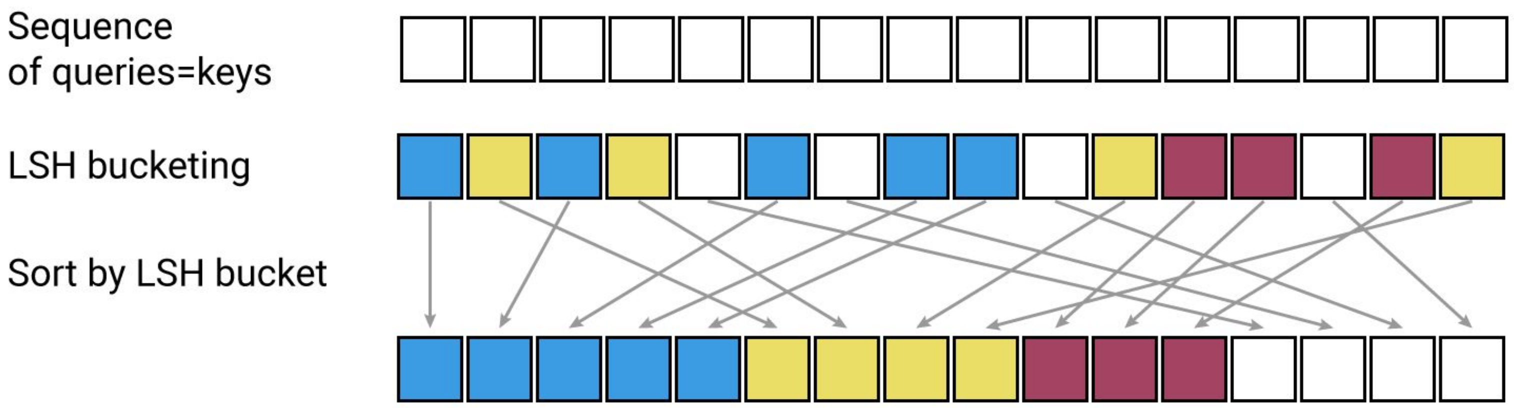


LSH Attention

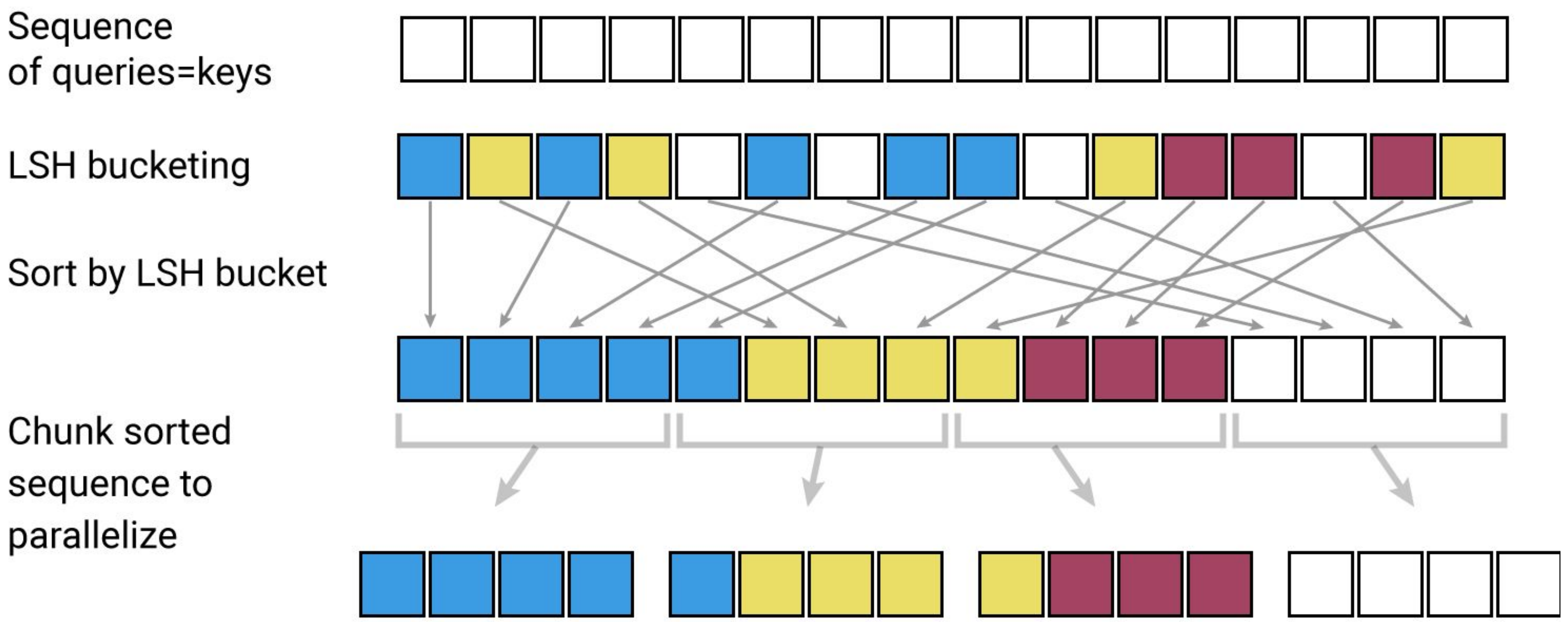
Google Research



LSH Attention

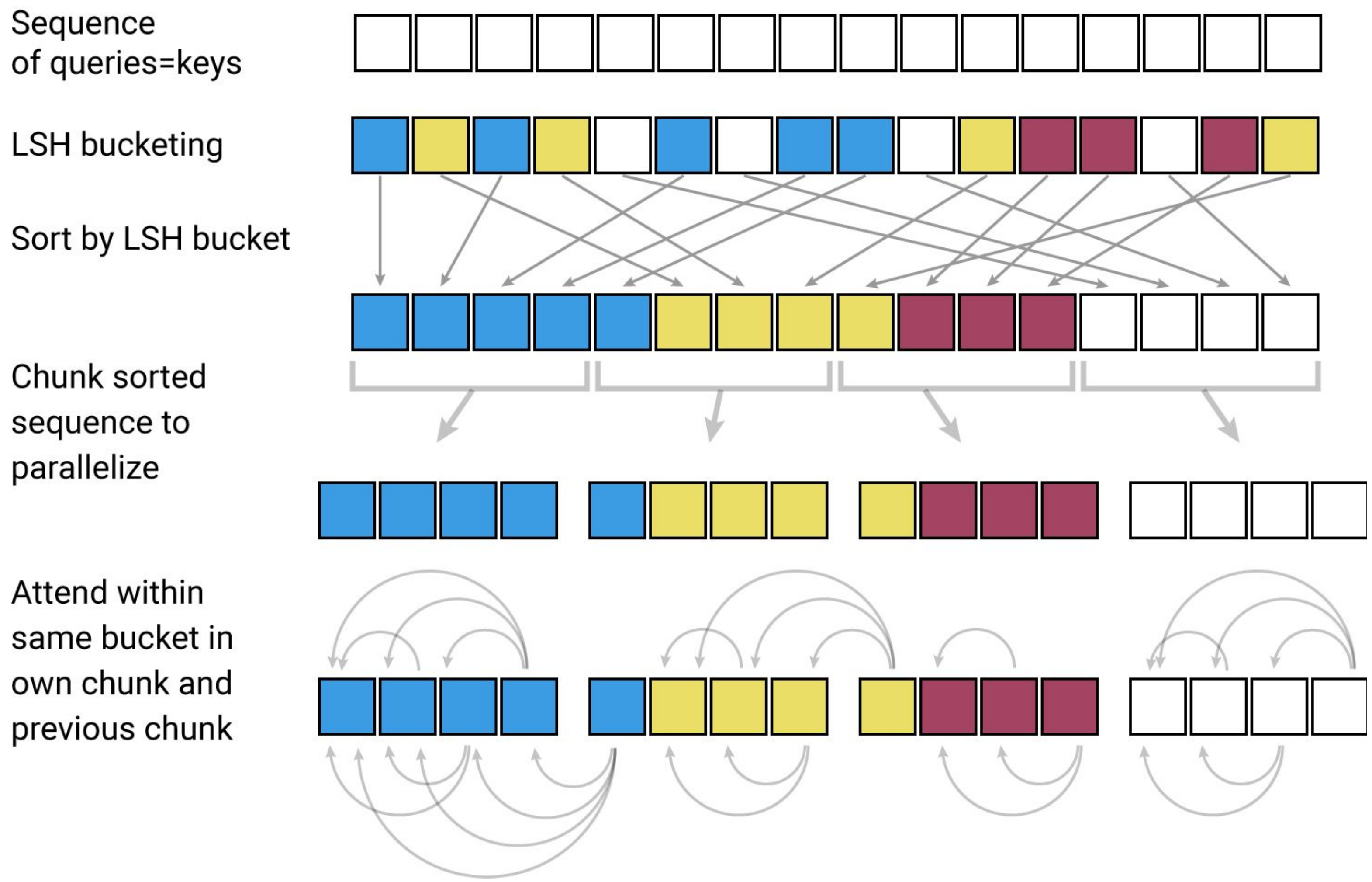


LSH Attention

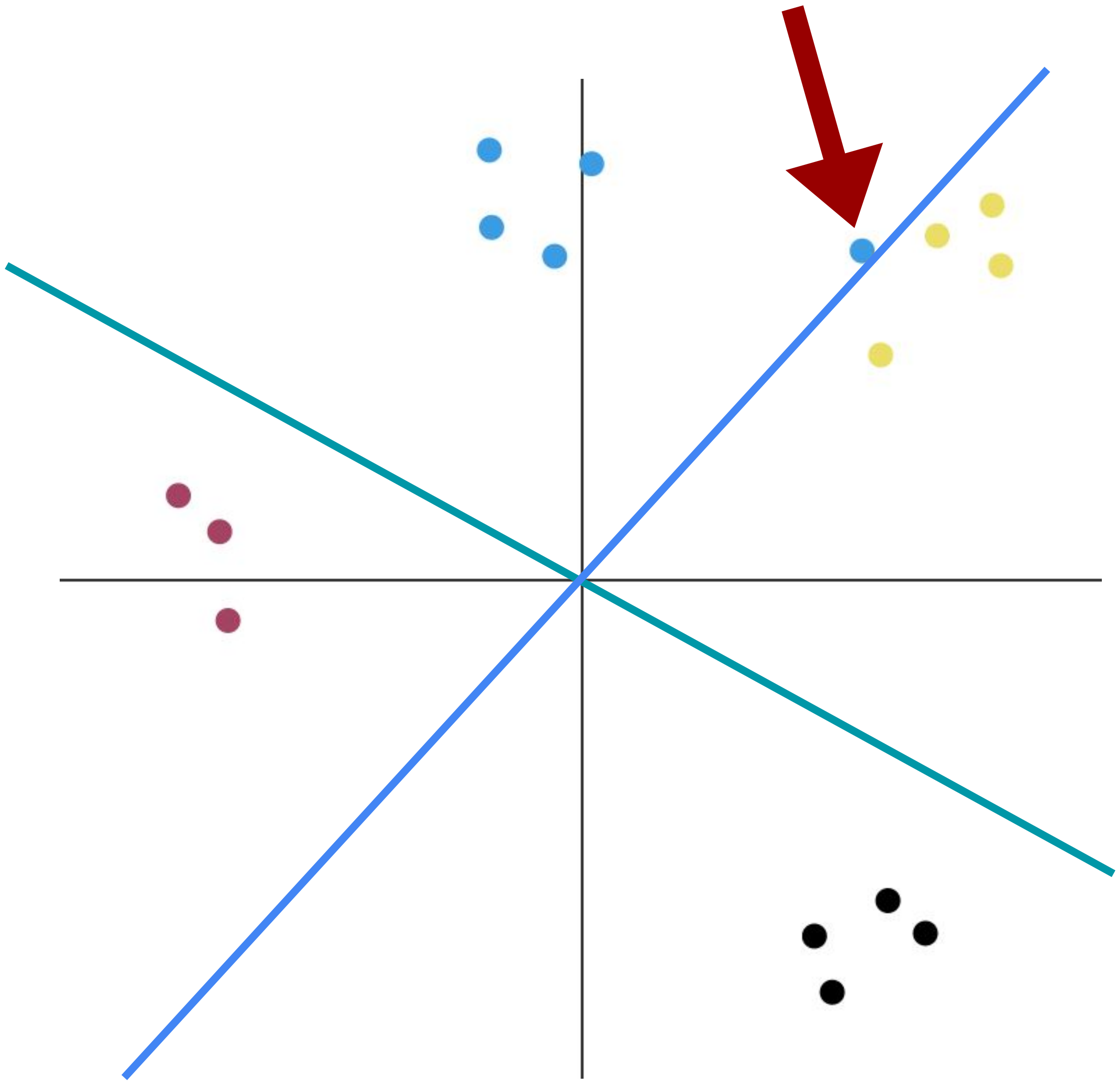


Google Research

LSH Attention

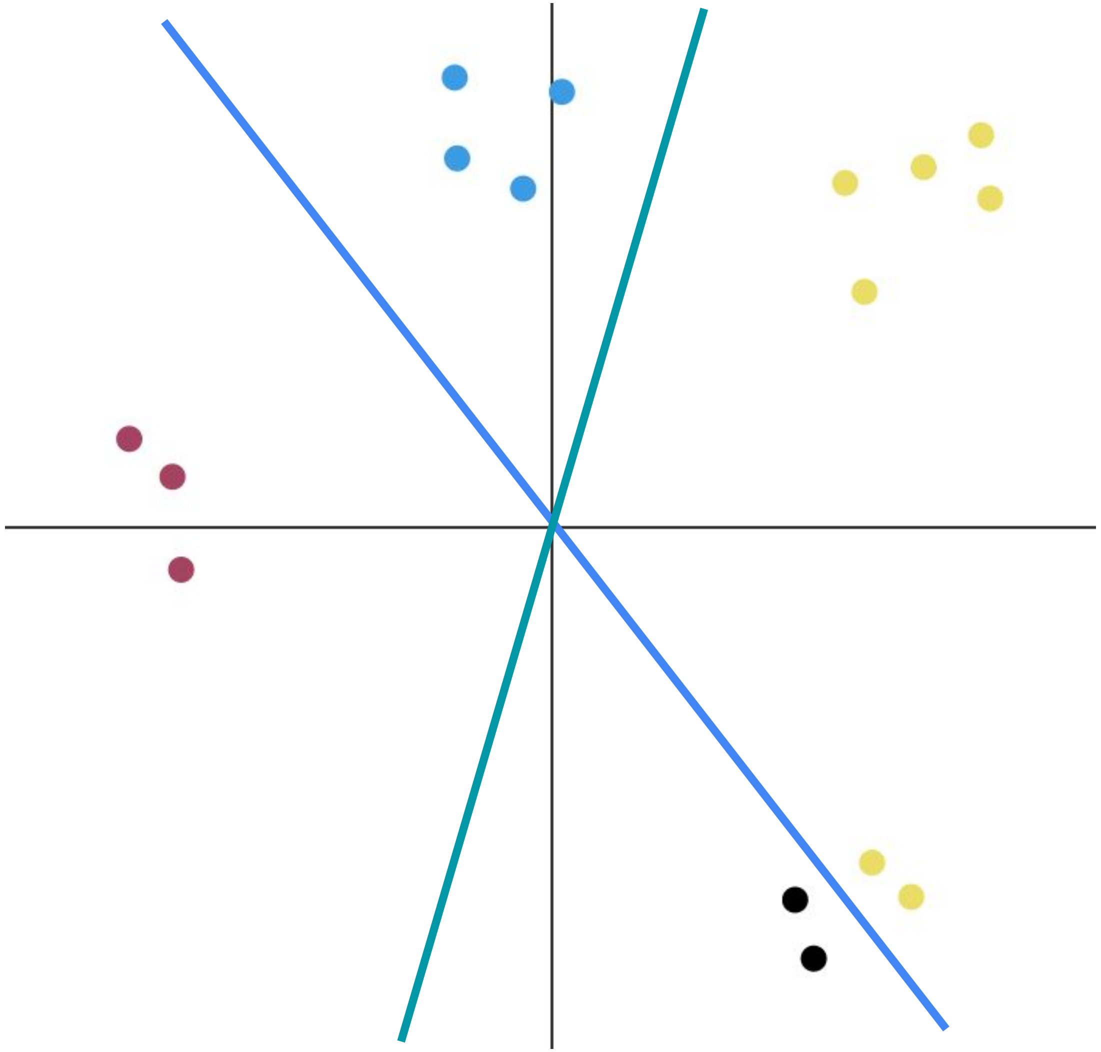


LSH Attention



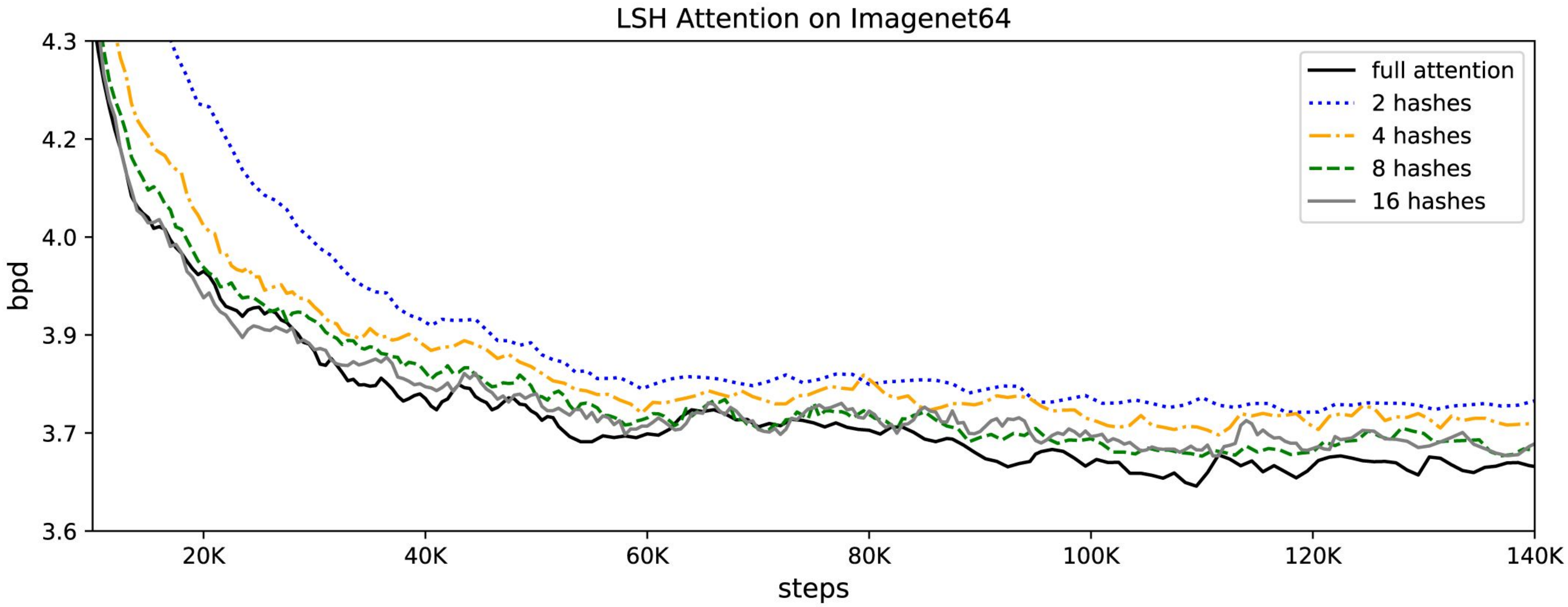
Google Research

LSH Attention



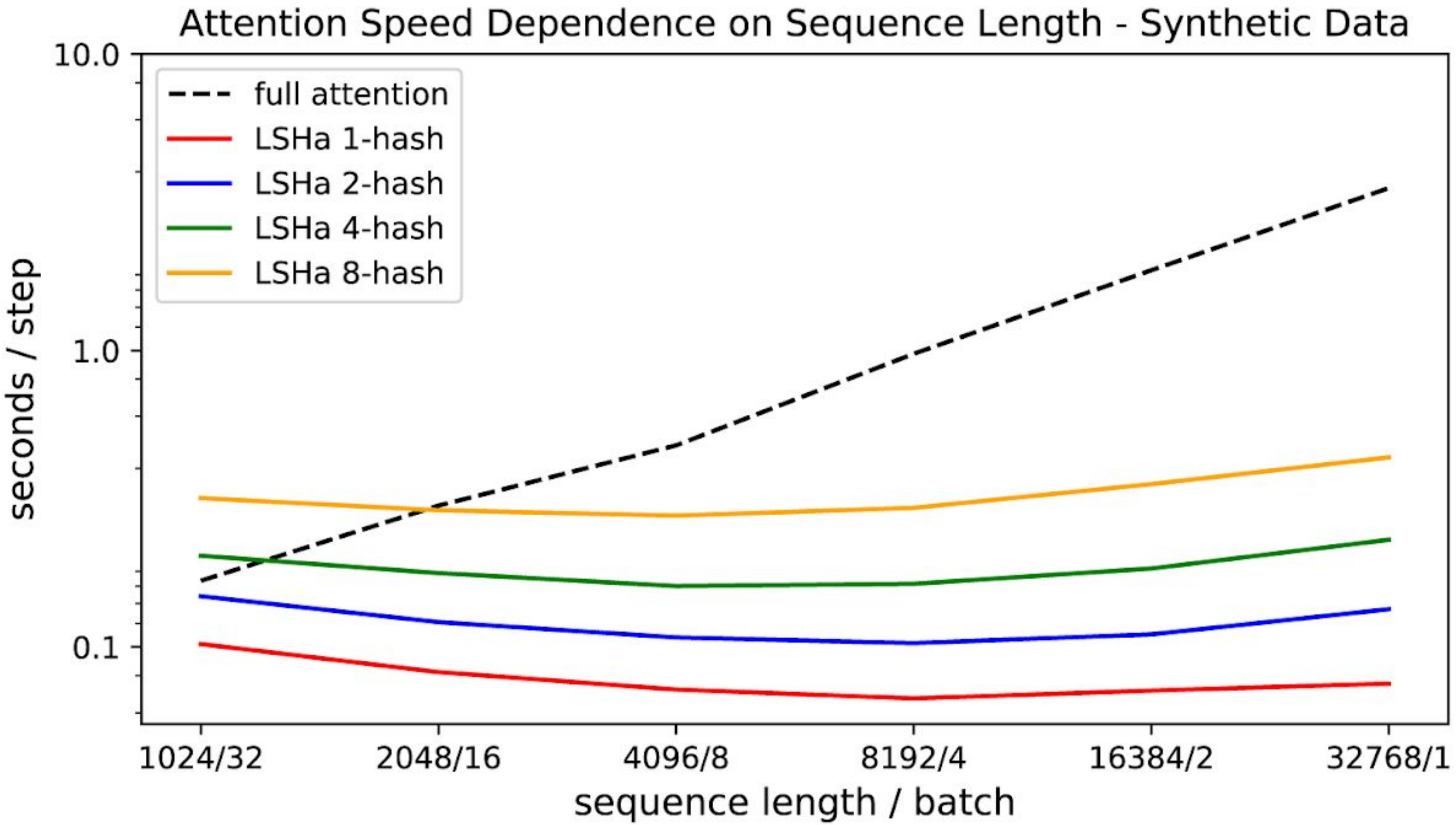
Google Research

LSH Attention: Model Quality



Google Research

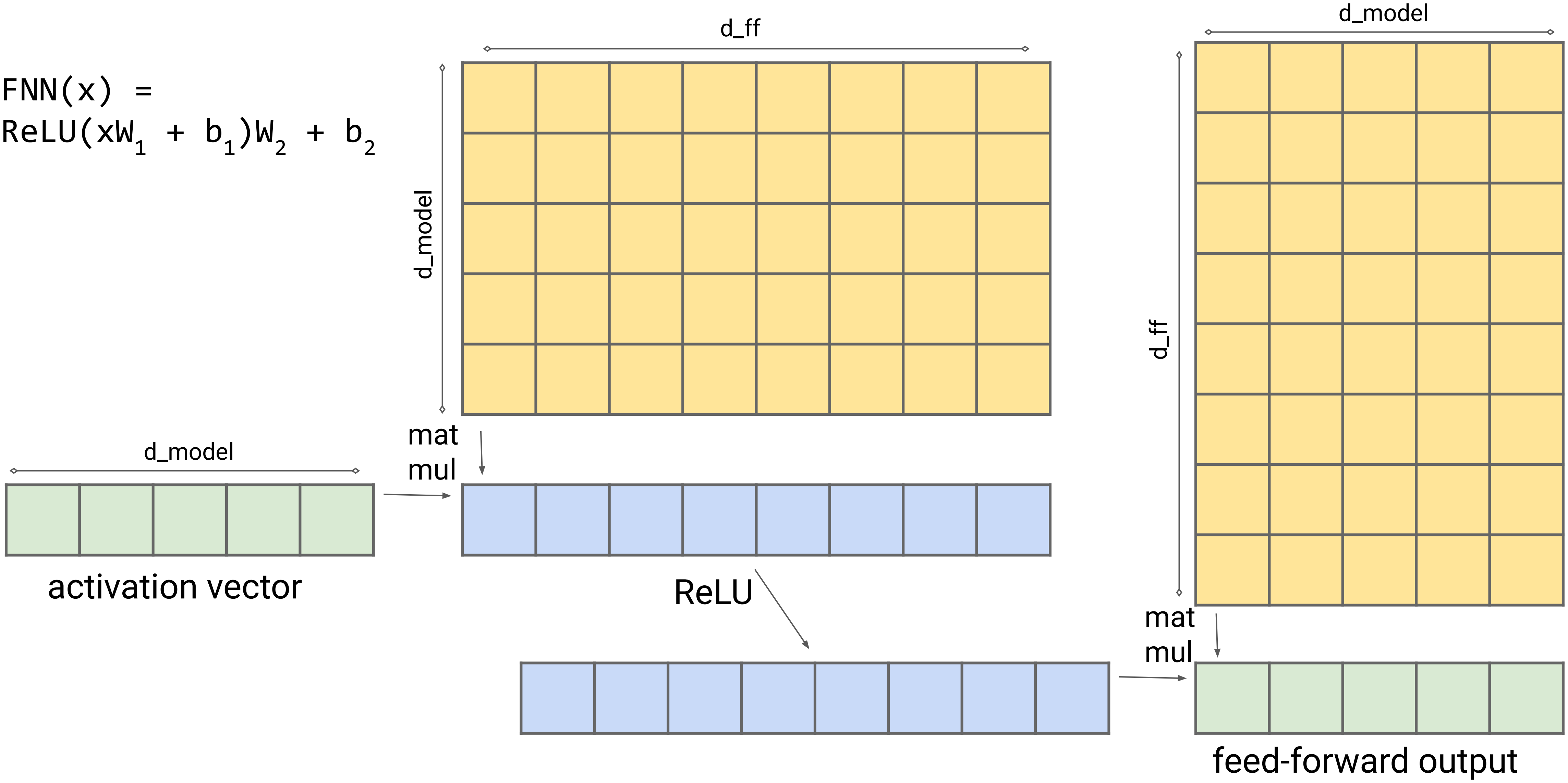
LSH Attention: Speed



Sparsity

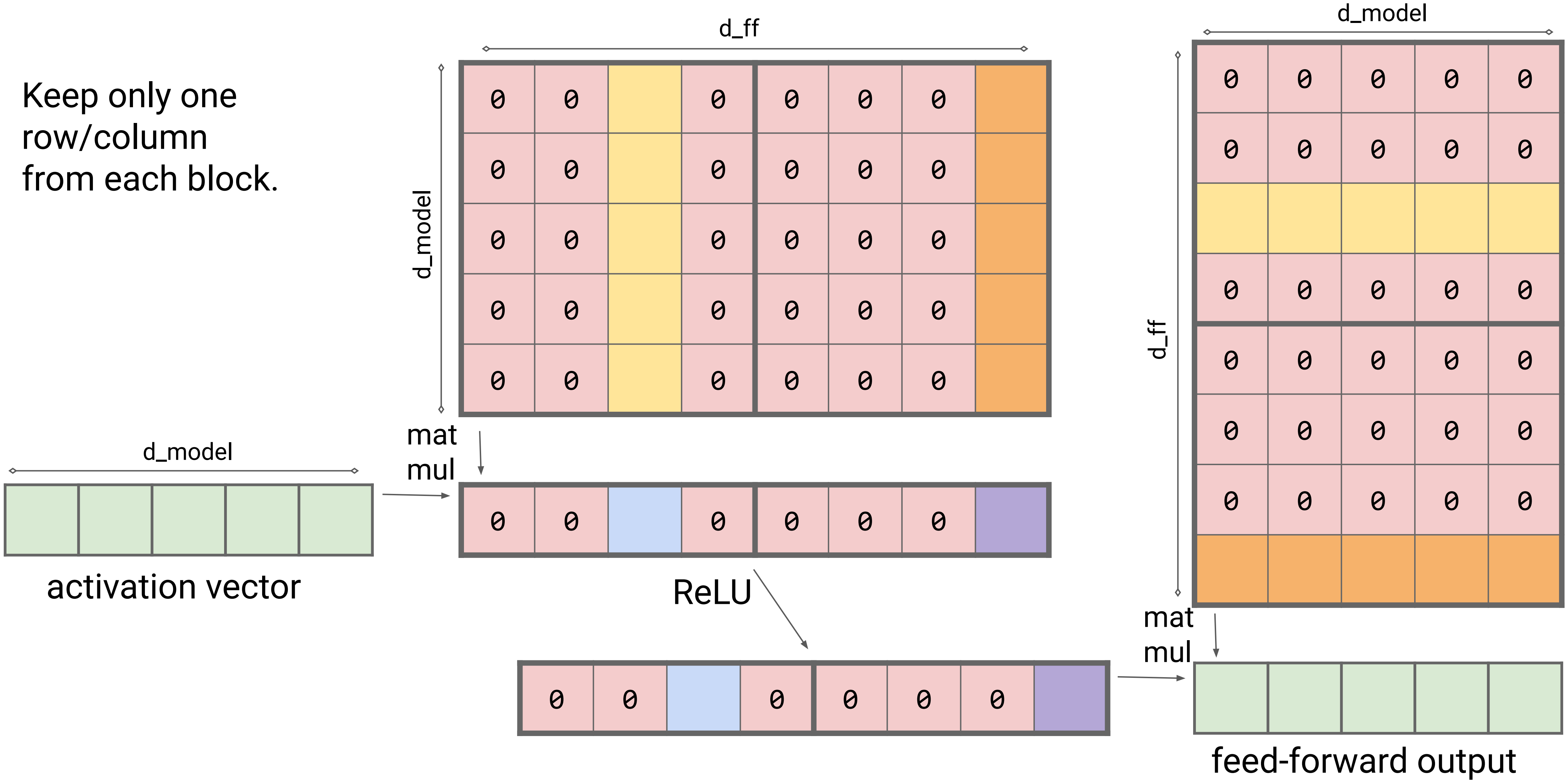
Standard Feed-Forward Layer

$$\mathrm{FNN}(x) = \mathrm{ReLU}(xW_1 + b_1)W_2 + b_2$$



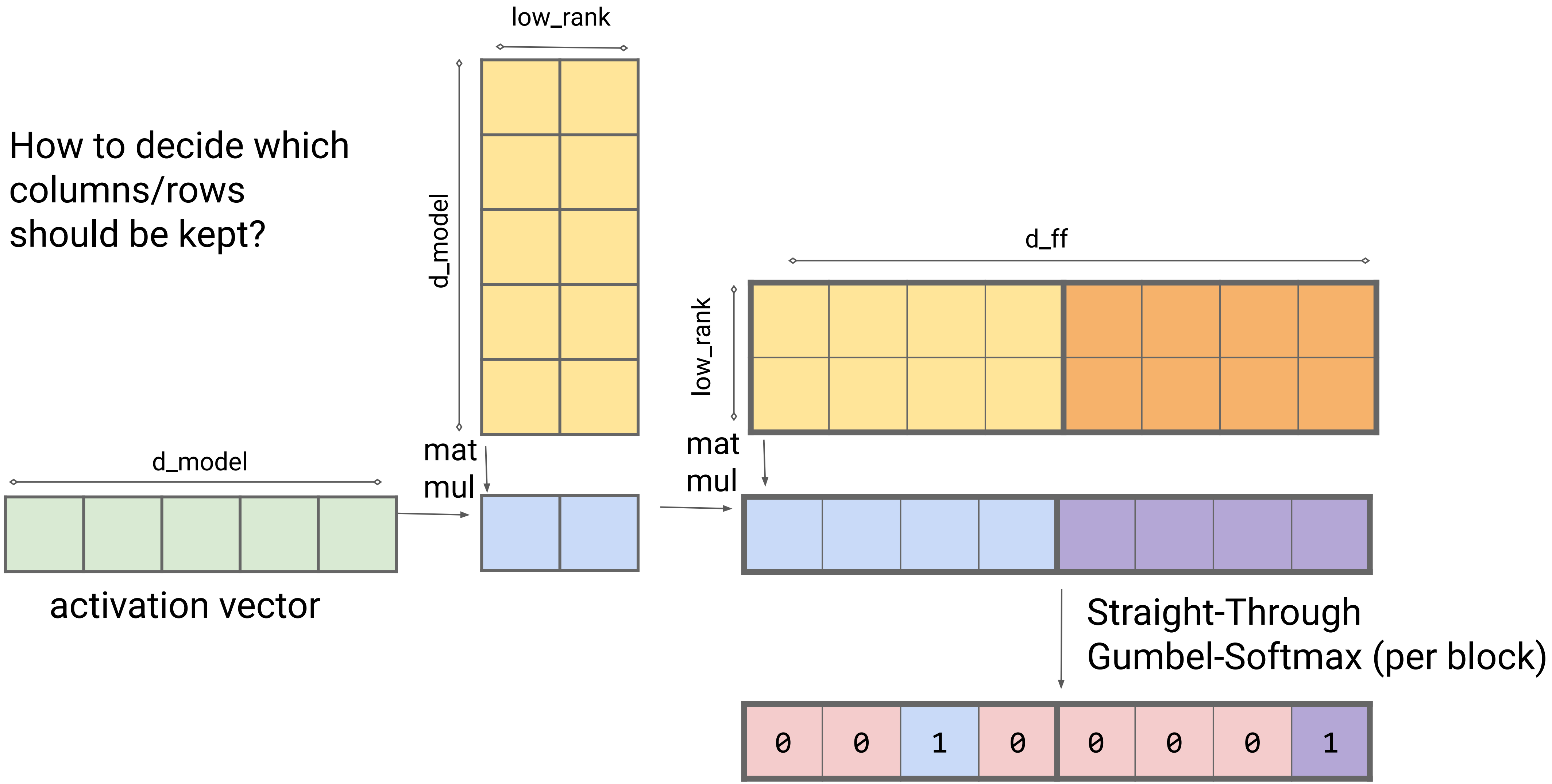
Sparse Feed-Forward Layer

Keep only one row/column from each block.

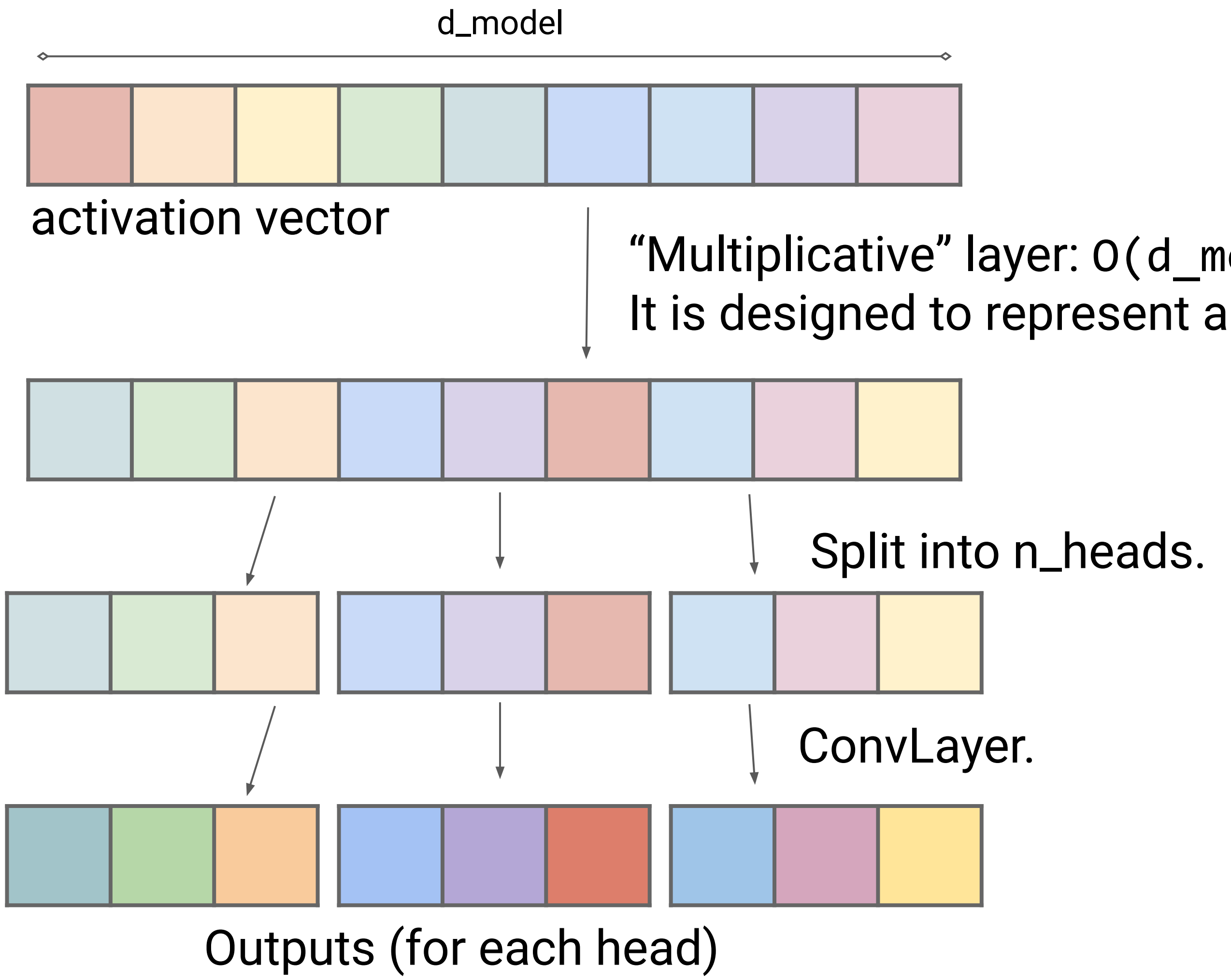


Sparse Feed-Forward Layer Controller

How to decide which columns/rows should be kept?



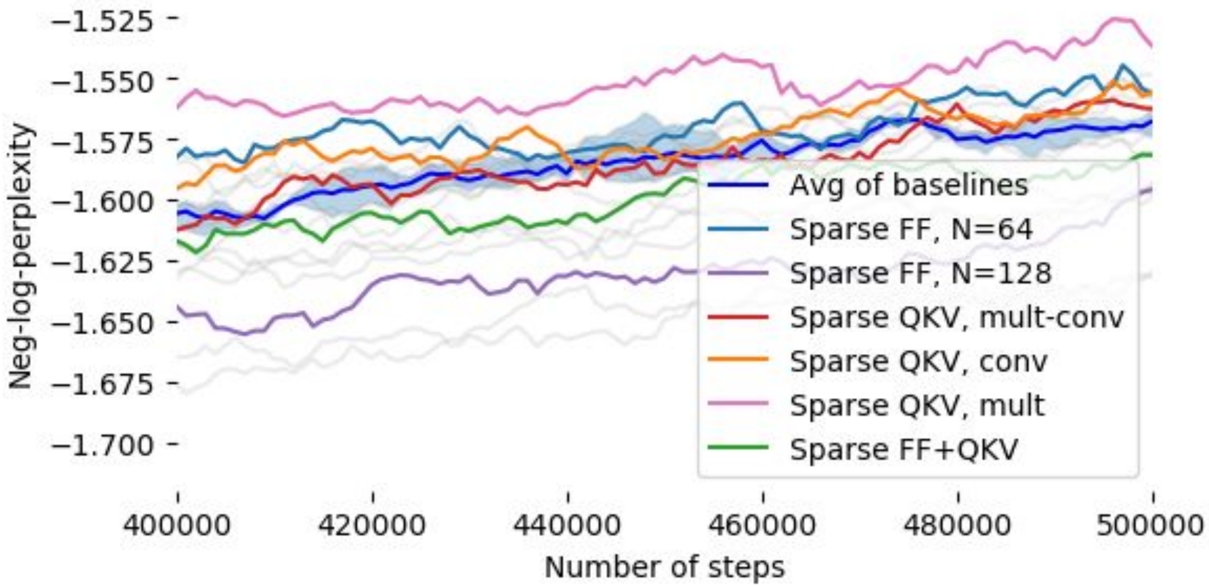
Sparsifying Dense QKV Layers in Attention

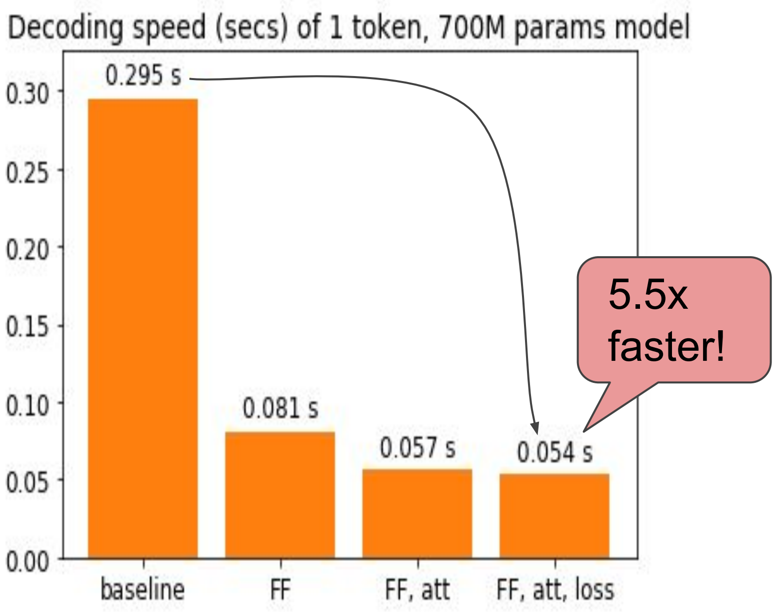


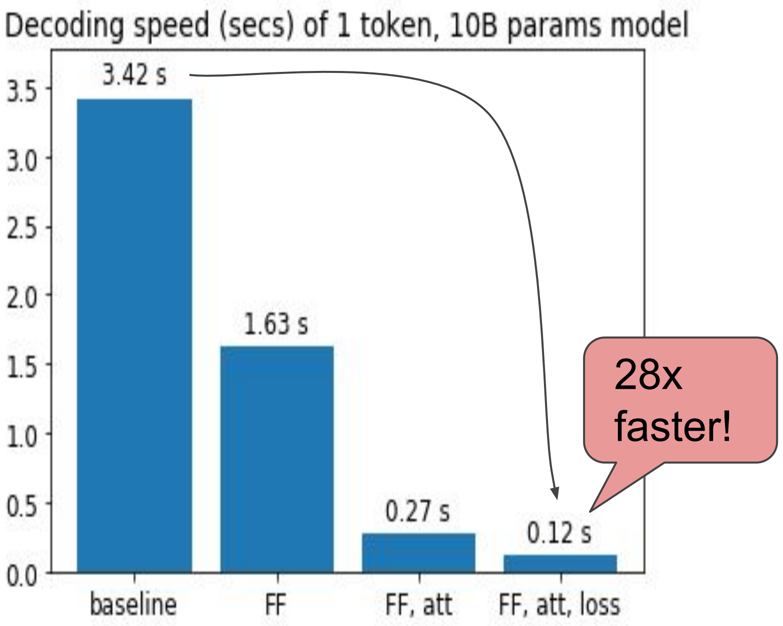
This layer has less parameters than Dense. To keep total number of model parameters, we always increase d_ff accordingly.

Scaling Transformer (Terraformer) Results

- Perplexity on par with dense model same size
- 5x+ decoding speedup on medium-sized model
- 28x+ decoding speedup on big model







Outlook

The future is promising!

- Efficient Transformers for all lengths
- Decoding fast enough even on CPUs
- Fine-tuning possible for everyone