Long Text Sequences

Tasks In NLP:

Writing Books

Chatbots

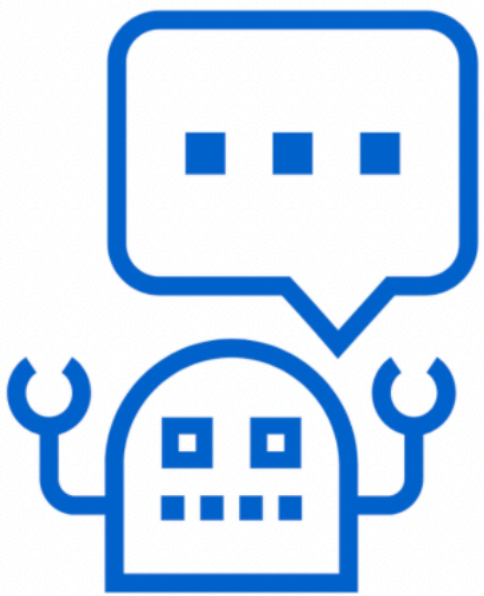
Tasks that make use of long sequences include

- Writing books
- Storytelling
- Building intelligent agents for conversations like chatbots.

This week you will learn about the bottlenecks in these larger transformer models, and solutions you can use to make them trainable for you. You will also learn about the re-former model (AKA the reversible transformer). Here is what you will be building for your programming assignment: A chatbot!

Context Windows:

User 1: What's for dinner?
Chatbot: Who's cooking, you or me?
User 1: Hey now chatbot.
Chatbot: I hope it's not hay, that's what horses eat.



Transformer Issues

- Attention on sequence of length L takes L^2 time and memory

L=100	L^2 = 10K	(0.001s at 10M ops/s)
L=1000	L^2 = 1M	(0.1s at 10M ops/s)
L=10000	L^2 = 100M	(10s at 10M ops/s)
L=100000	L^2 = 10B	(1000s at 10M ops/s)

- N layers take N times as much memory

  GPT-3 has 96 layers and new models will have more

Attention Complexity

- Attention: softmax(QK^T)V
- Q, K, V are all [L, d_model]
- QK^T is [L, L]
- Save compute by using area of interest for large L

Memory with N Layers

- Activations need to be stored for backprop
- Big models are getting bigger
- Compute vs memory tradeoff

What does Attention do?

Select Nearest Neighbors (K,Q) and return corresponding V

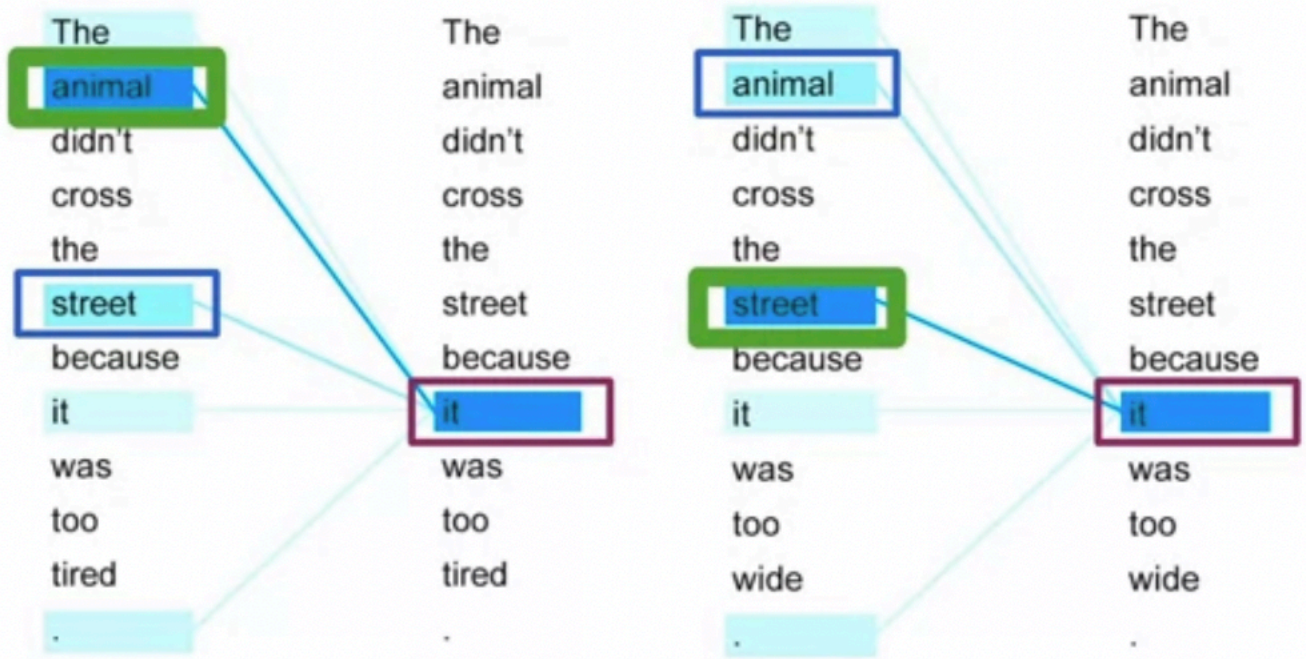


image © (Transformer: A Novel Neural Network Architecture for Language Understanding.)

Nearest Neighbors

Course:

Natural Language Processing with Classification and Vector Spaces

Lessons:

- KNN
- Hash Tables and Hash Functions
- Locality Sensitive Hashing
- Multiple Planes

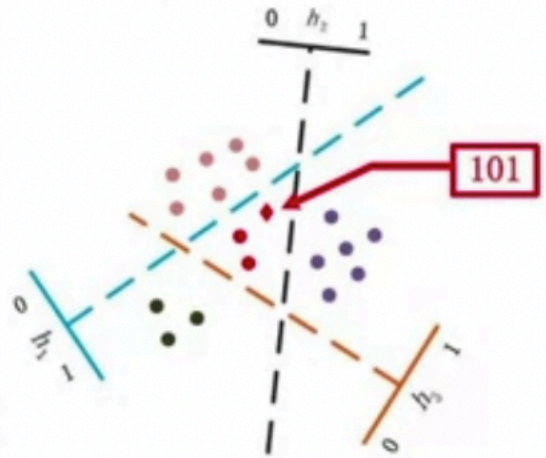
Nearest Neighbors

Compute the nearest neighbor to q among vectors $\{k_1, ..., k_n\}$

- Attention computes $d(q, k_i)$ for i from 1 to n which can be slow
- Faster *approximate* uses locality sensitive hashing (LSH)
- Locality sensitive: if q is close to k_i:

  hash(q) == hash(k_i)
- Achieve by randomly cutting space

  hash(x) = sign(xR) R: [d, n_hash_bins]



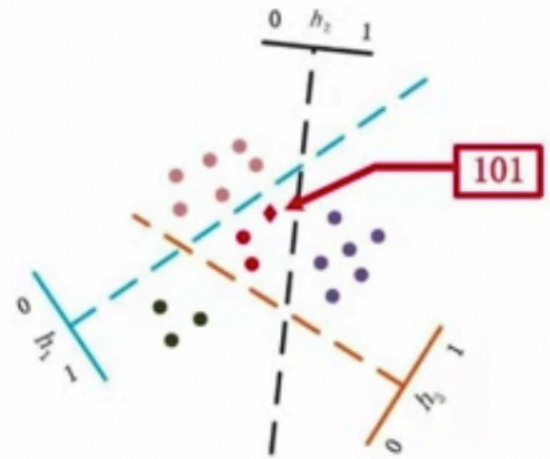
LSH Attention

Standard Attention:

$$\mathrm{A}(Q, K, V) = \mathrm{softmax}(QK^T)V$$

LSH Attention:

- Hash Q and K
- Standard attention within same-hash bins
- Repeat a few times to increase probability of key in the same bin



LSH Attention

image © (Reformer: The Efficient Transformer)

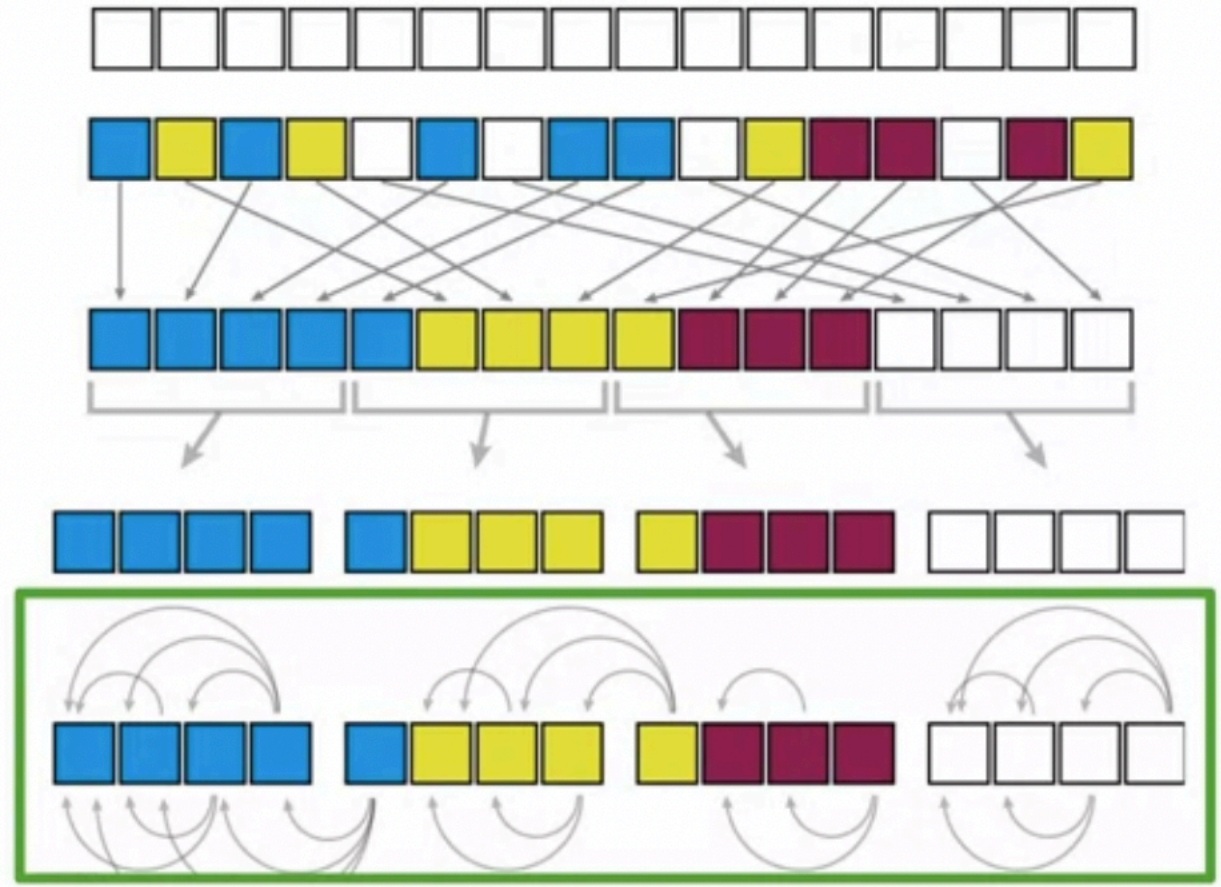
Sequence of Queries = Keys

LSH bucketing

Sort by LSH bucket

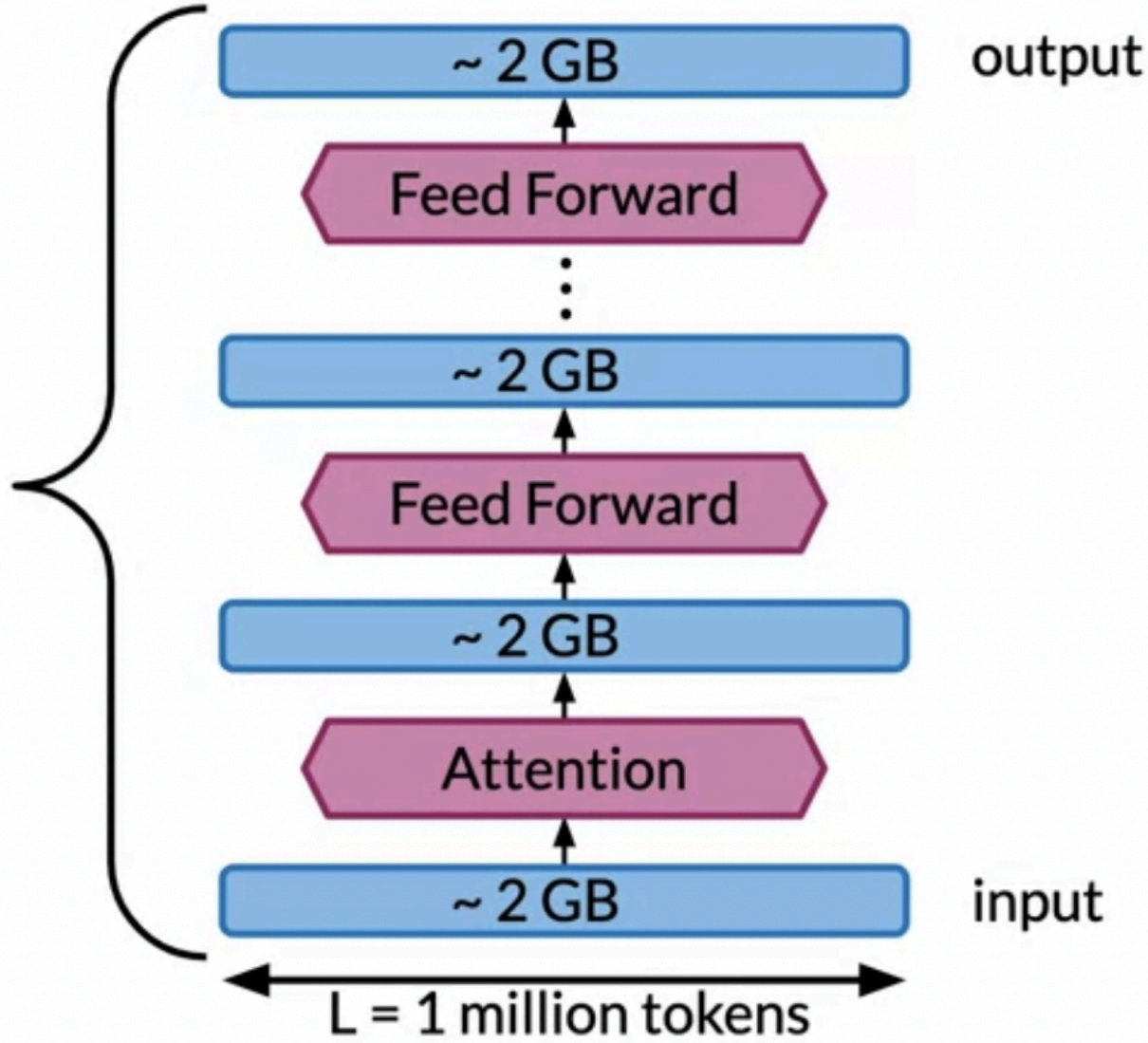
Chunk sorted sequence to parallelize

Attend within same bucket of own chunk and previous chunk



Memory Efficiency

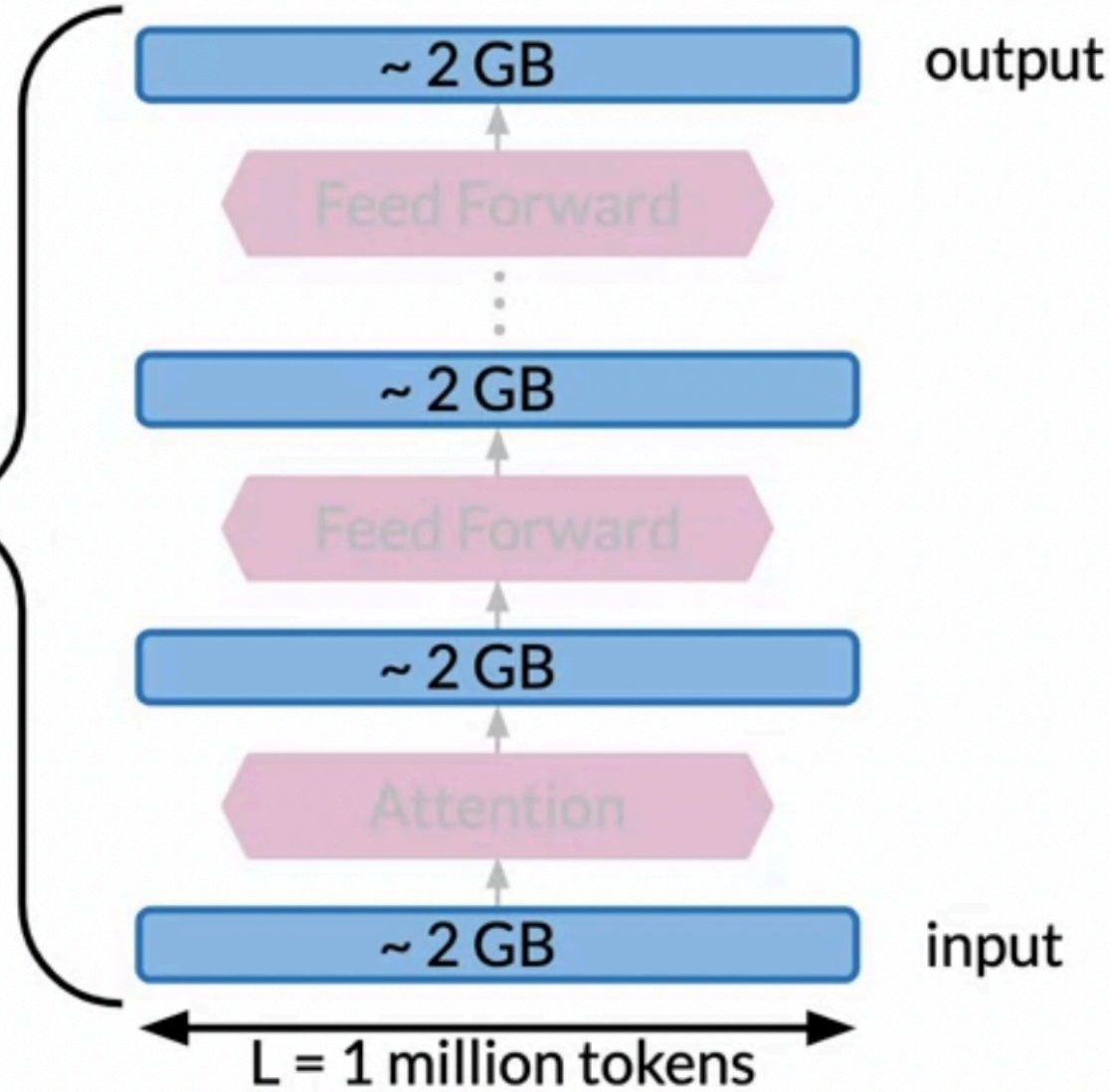
12 x Attention
12 x Feed-Forward



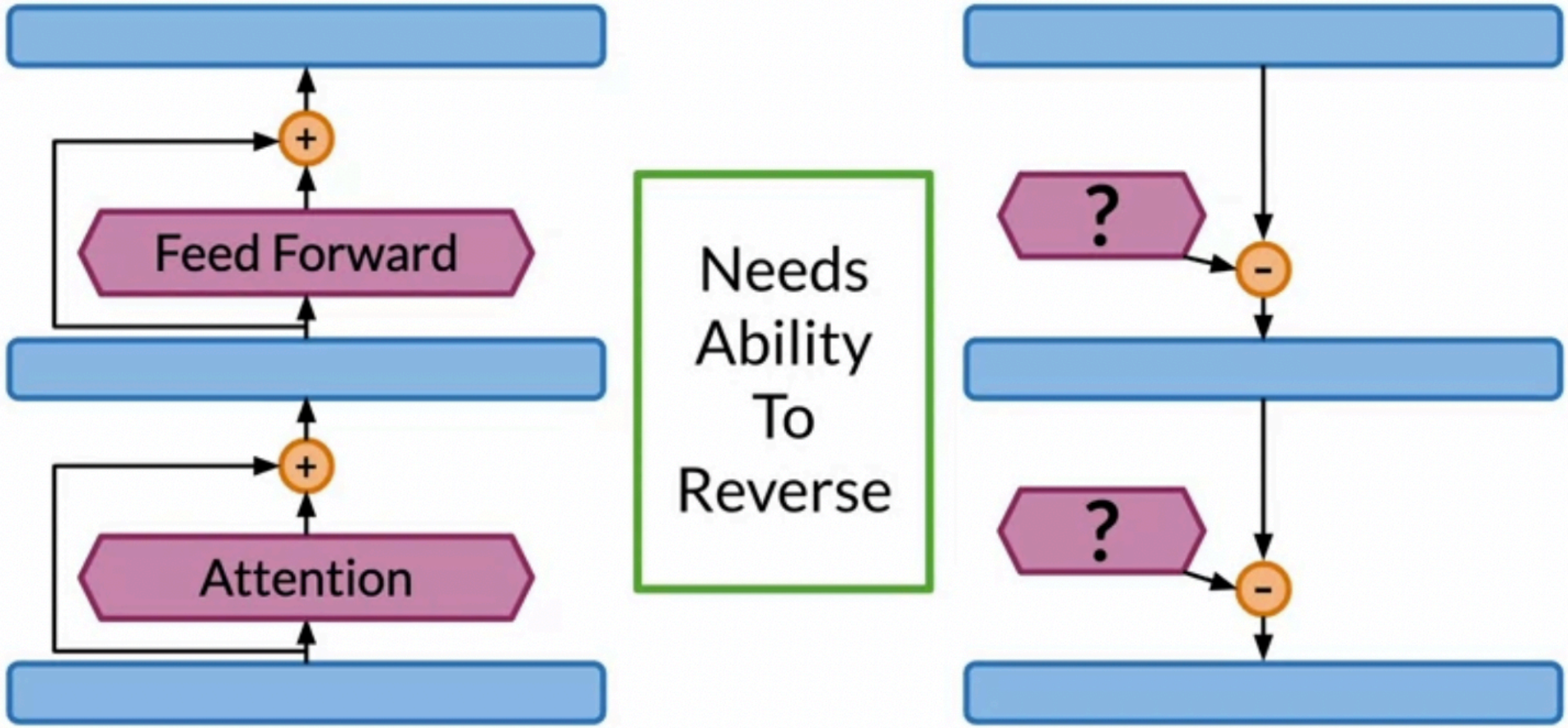
Memory Efficiency

12 x Attention
12 x Feed-Forward

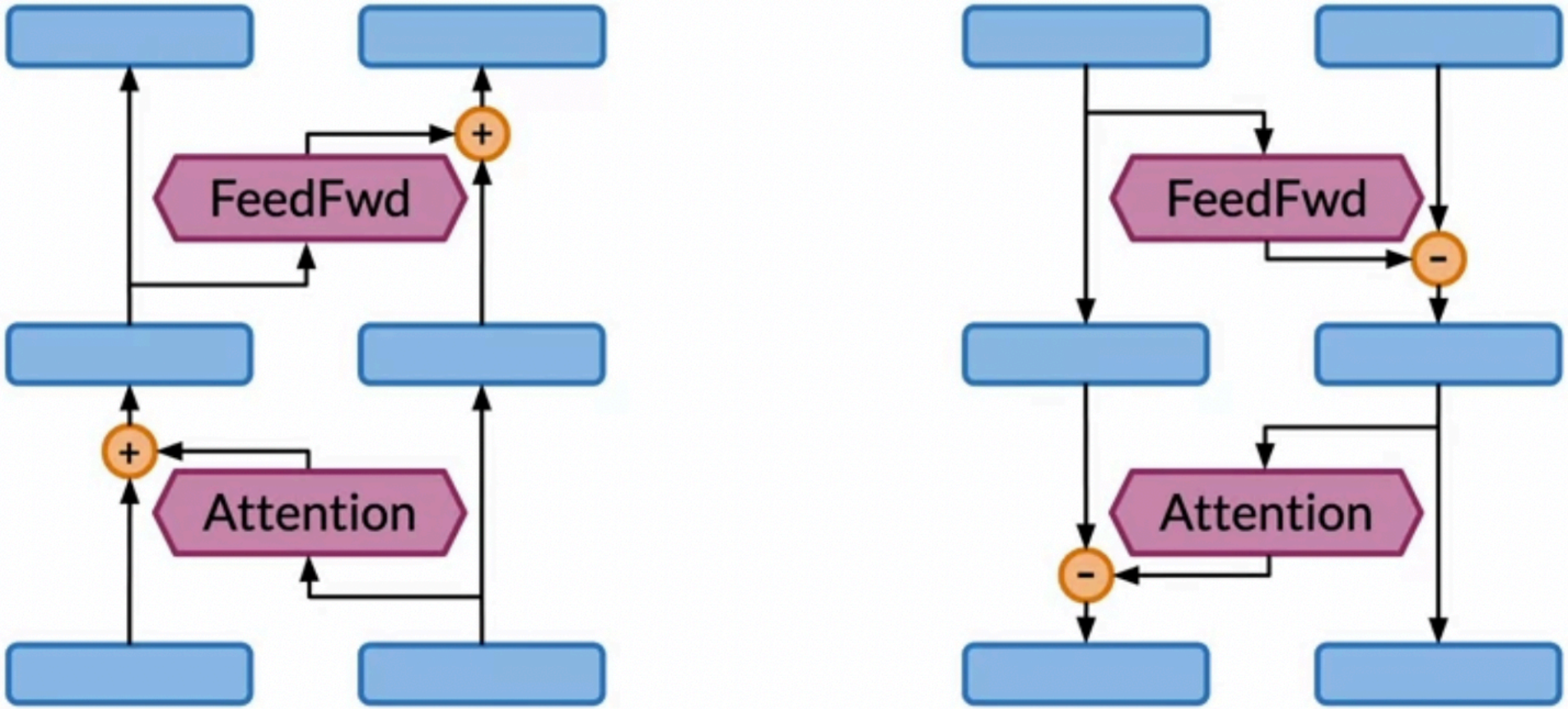
50 GB total



Residual Blocks in Transformer



Reversible layers



Reversible layers equations

Standard Transformer:

$$y_a = x + \text{Attention}(x) \qquad y_b = y_a + \text{FeedFwd}(y_a)$$

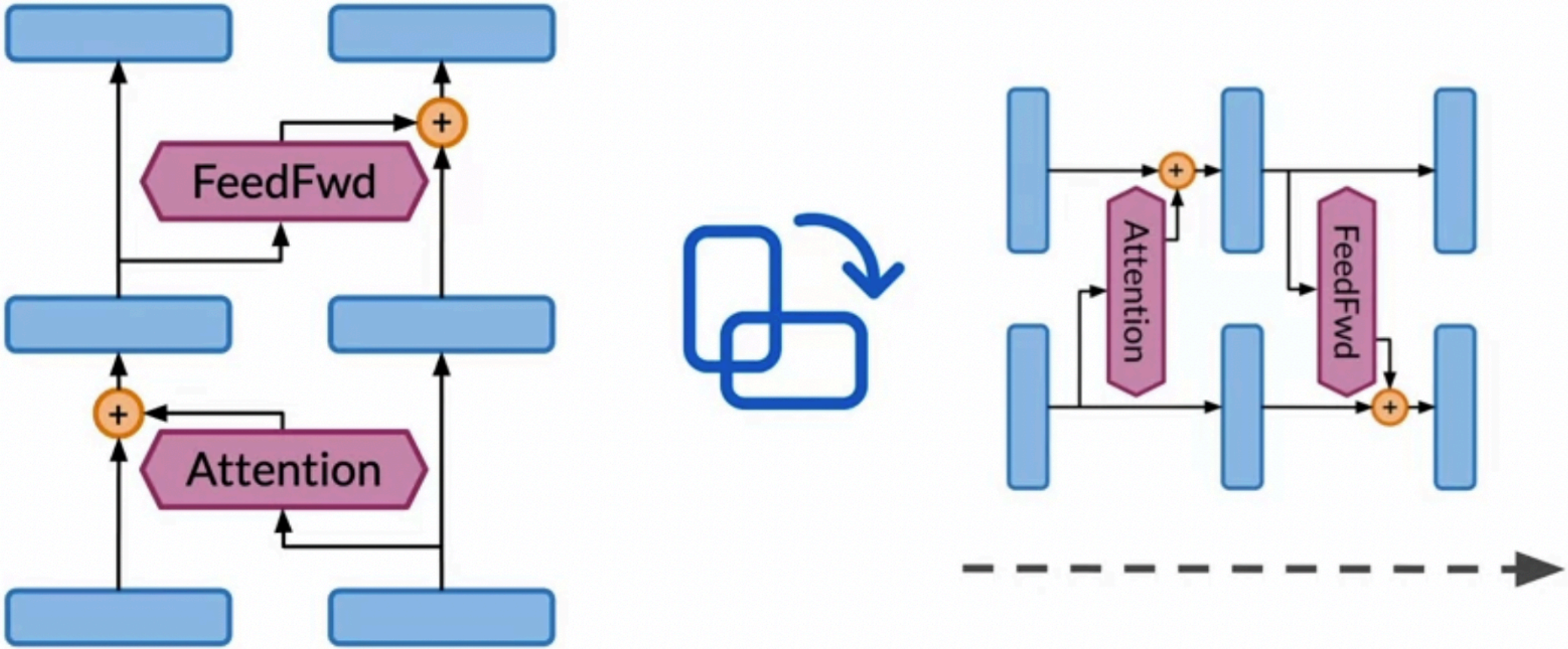
Reversible:

$$y_1 = x_1 + \text{Attention}(x_2) \qquad y_2 = x_2 + \text{FeedFwd}(y_1)$$

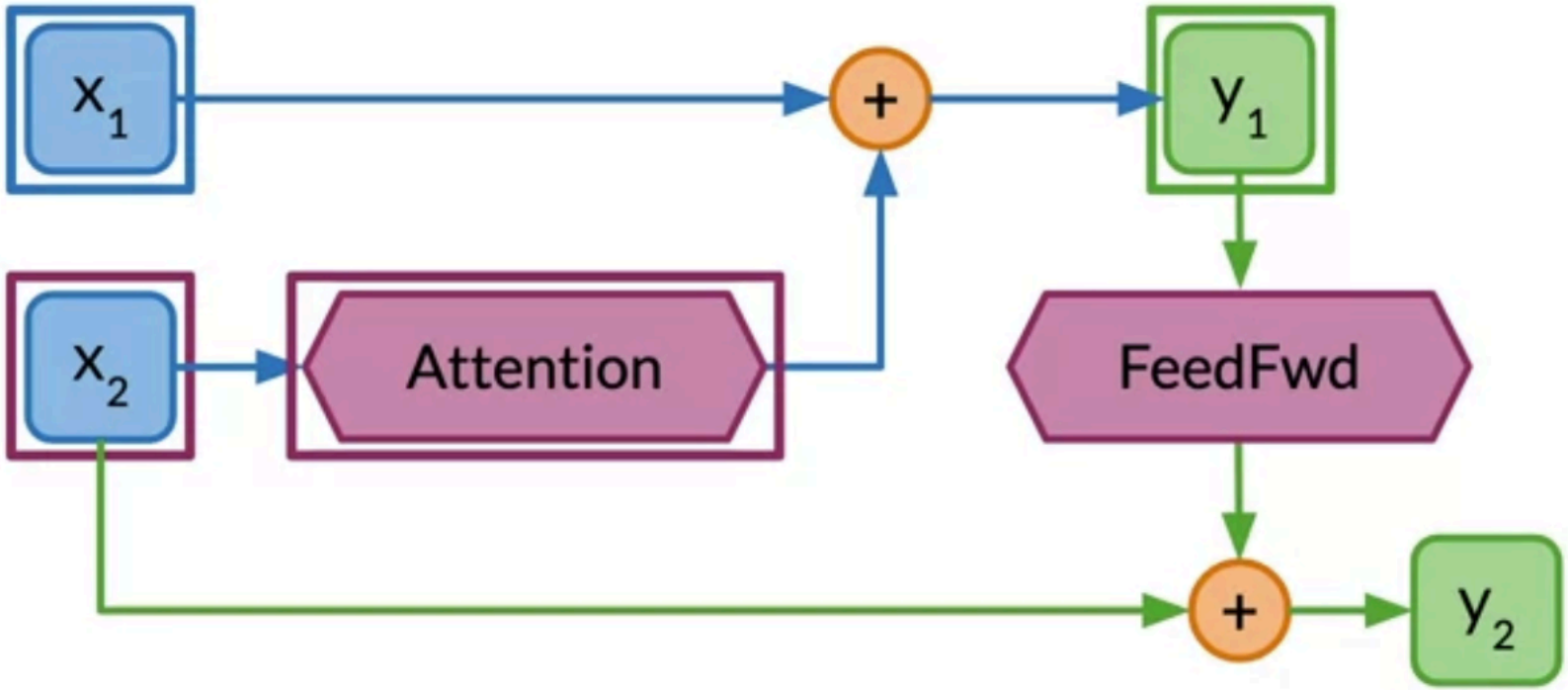
Recompute x_1, x_2 from y_1, y_2:

$$x_1 = y_1 - \text{Attention}(x_2) \qquad x_2 = y_2 - \text{FeedFwd}(y_1)$$

Reversible layers equations



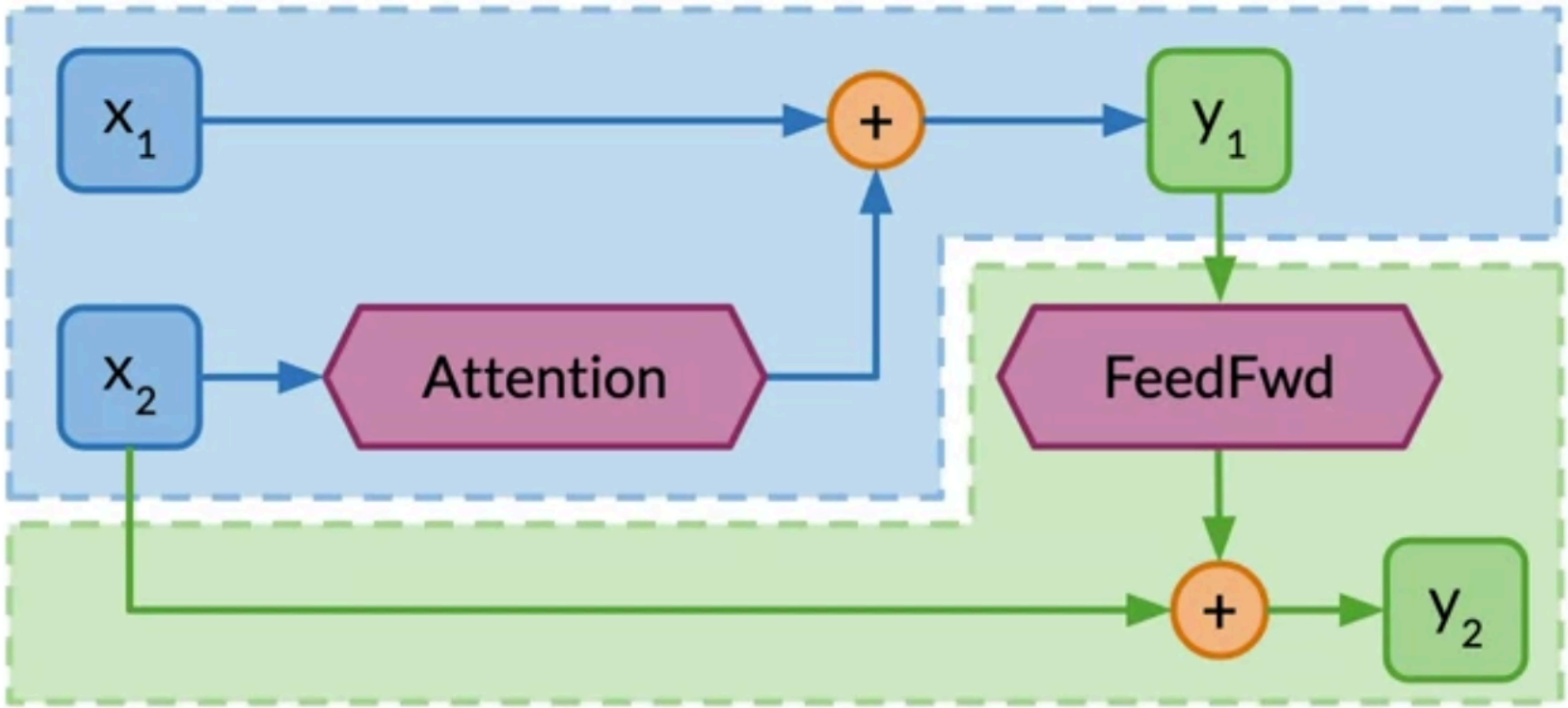
Reversible layers equations



$y_1 = x_1 + \text{Attention}(x_2)$

$y_2 = x_2 + \text{FeedFwd}(y_1)$

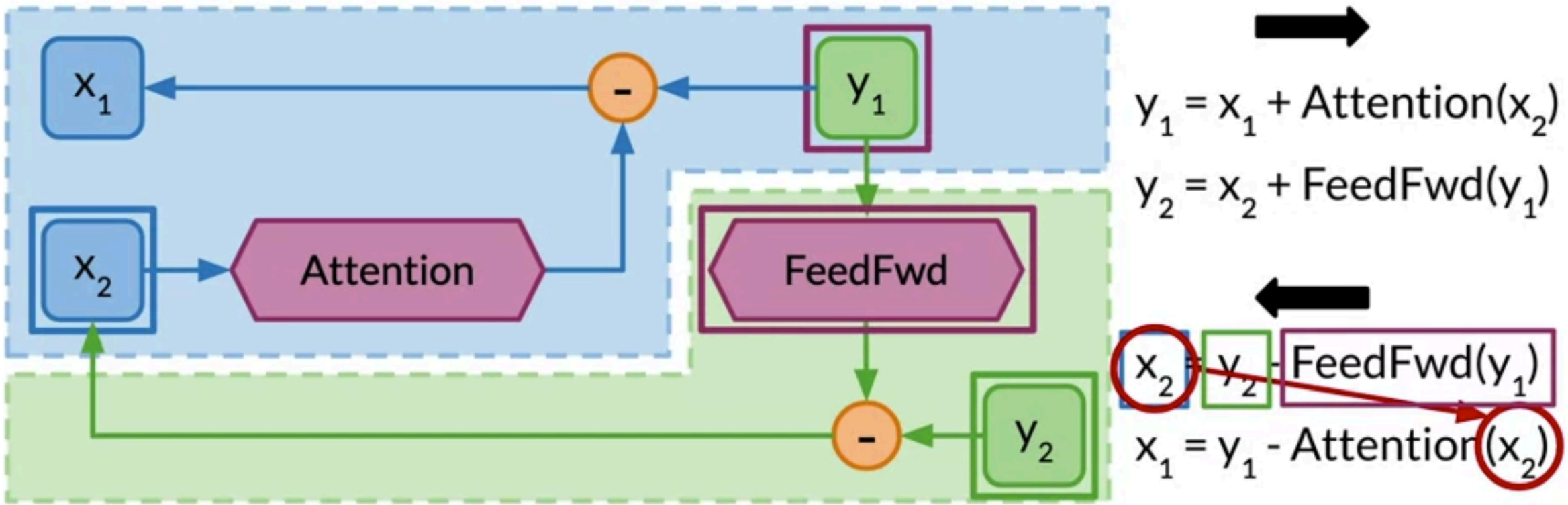
Reversible layers equations



$$y_1 = x_1 + \text{Attention}(x_2)$$

$$y_2 = x_2 + \text{FeedFwd}(y_1)$$

Reversible layers equations



Reformer

- LSH Attention
- Reversible Layers

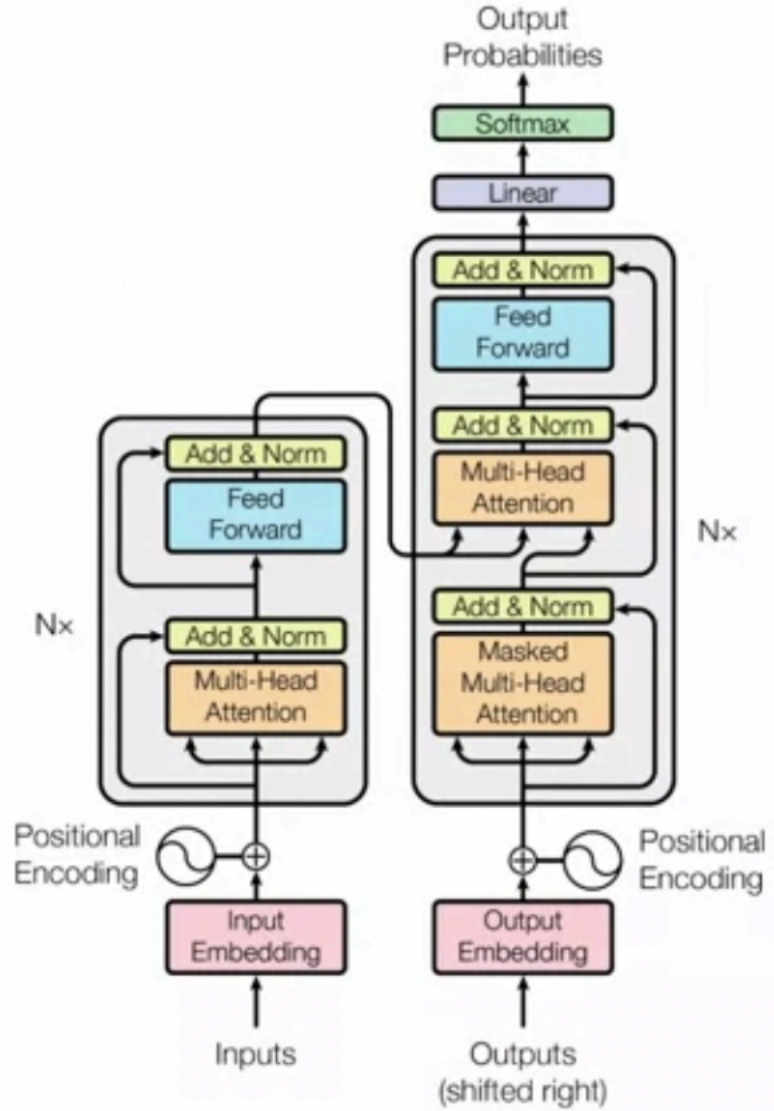


image © (Attention Is All You Need)

Reformer

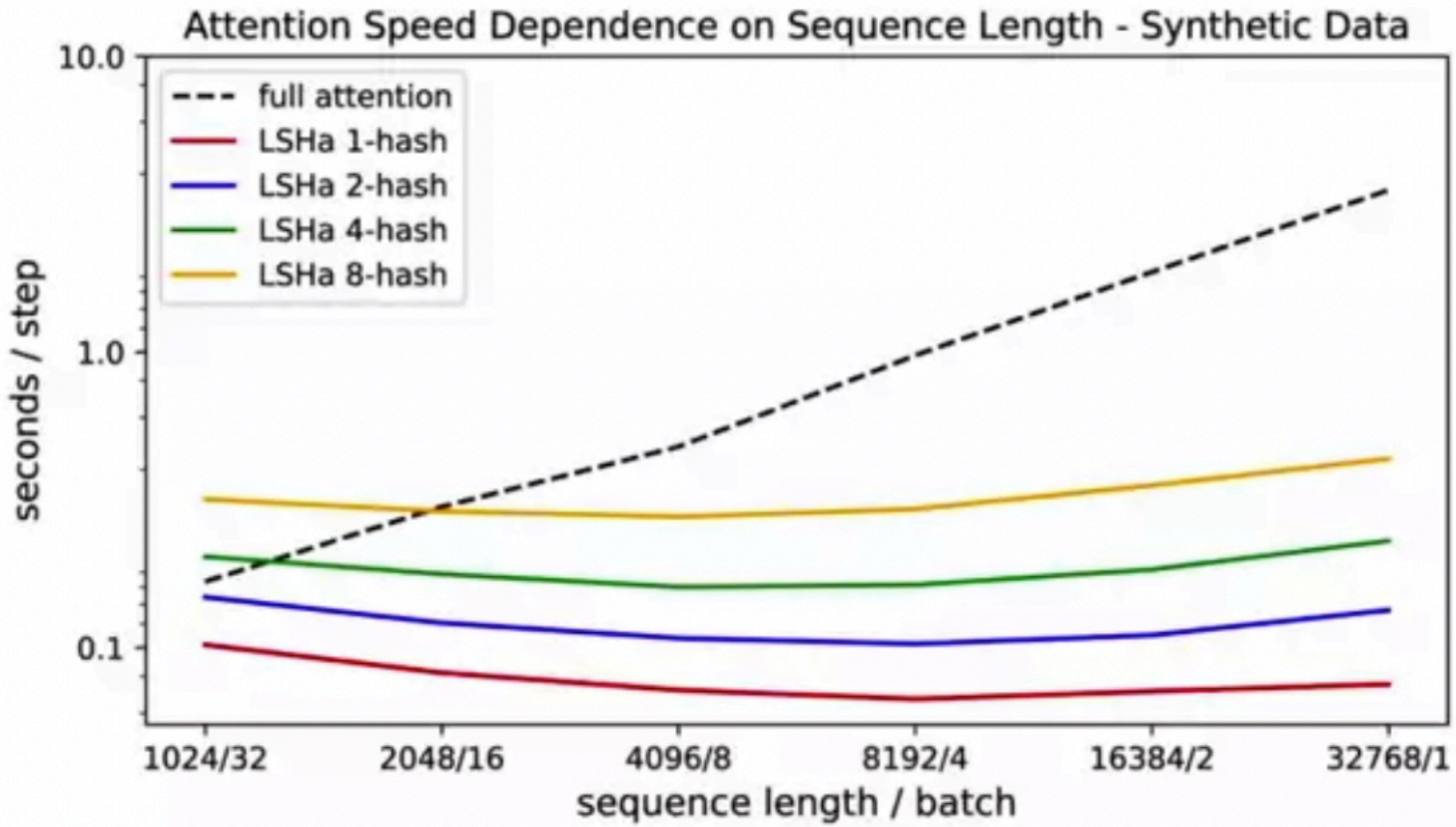


image © (Reformer: The Efficient Transformer)