Natural Language Processing (NLP)

Generative Pre-trained Transformer

Large Language Modeling

By:

Dr. Zohair Ahmed

Generative Pre-trained Transformer (GPT-1)

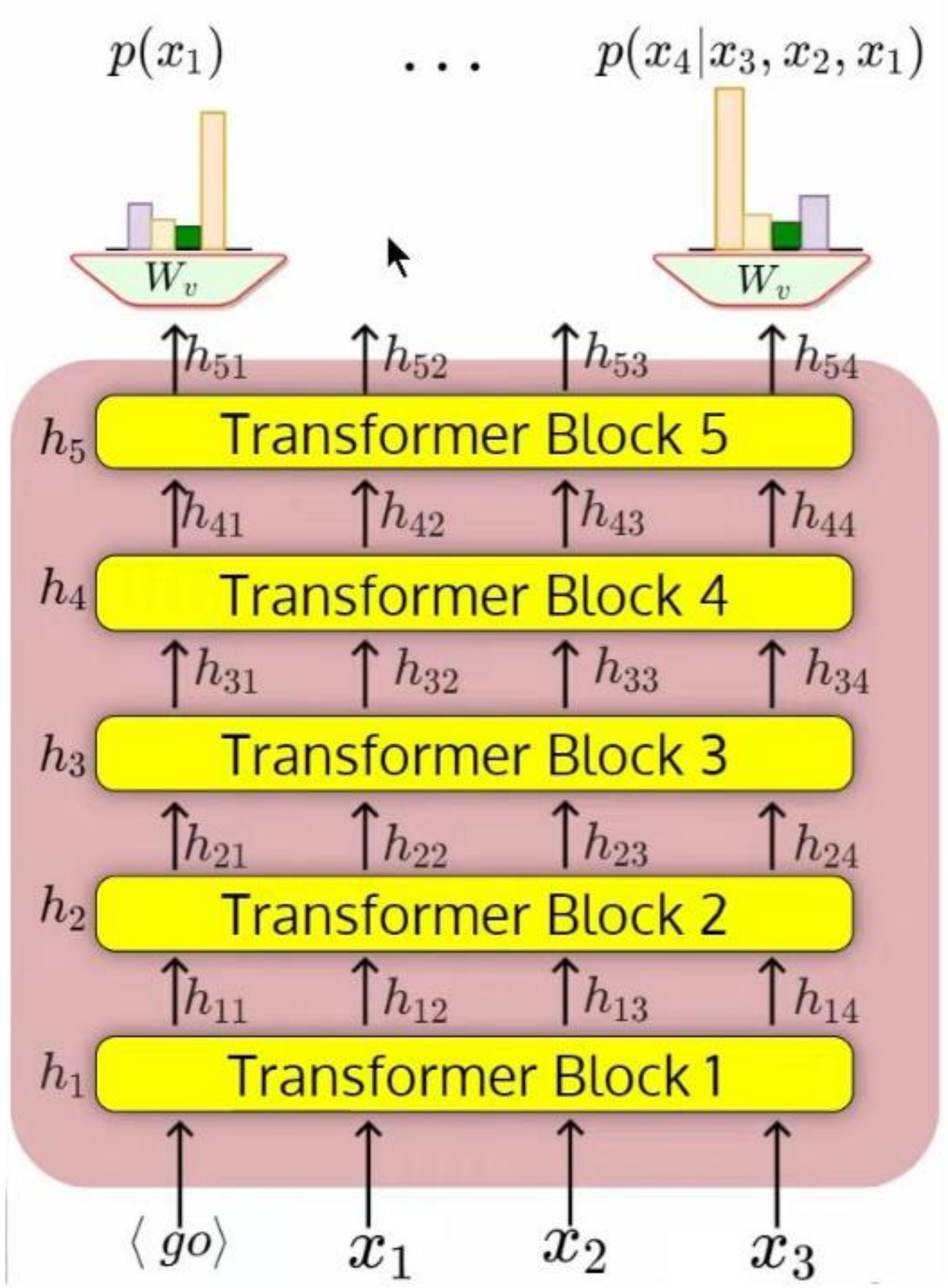
- The Birth of the Decoder-Only Era Paper:
- "Improving Language Understanding by Generative Pre-Training"
- (OpenAI) Released: 11 June 2018
- First model that showed:
- Train a big Transformer to just predict the next word
- magically solves many NLP tasks with minimal fine-tuning
- **Title:** Improving Language Understanding by Generative Pre-Training
- **Authors:** Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever **Date:** June 11, 2018
- **Direct PDF link (OpenAI official):** https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

What is a GPT Model?

- GPT stands for **Generative Pre-trained Transformer**.
- At its heart, it's a powerful text prediction engine.
- You give it a piece of text, and its only job is to predict the most likely next word. It does this over and over to "generate" new text.
- **Decoder-Only Architecture:** Think of the original Transformer model as having two parts: an *encoder* (for understanding input, like in translation) and a *decoder* (for generating output). GPT models are "decoder-only," meaning they are specialized solely for the task of generation.
- **Building Blocks:** The model is built by stacking multiple identical layers, called **Transformer Blocks**.
- Imagine a factory assembly line. Each Transformer Block is a workstation. A sentence goes into the first workstation, gets processed, and the improved version is passed to the next.
- GPT-1 had 12 of these workstations stacked on top of each other.

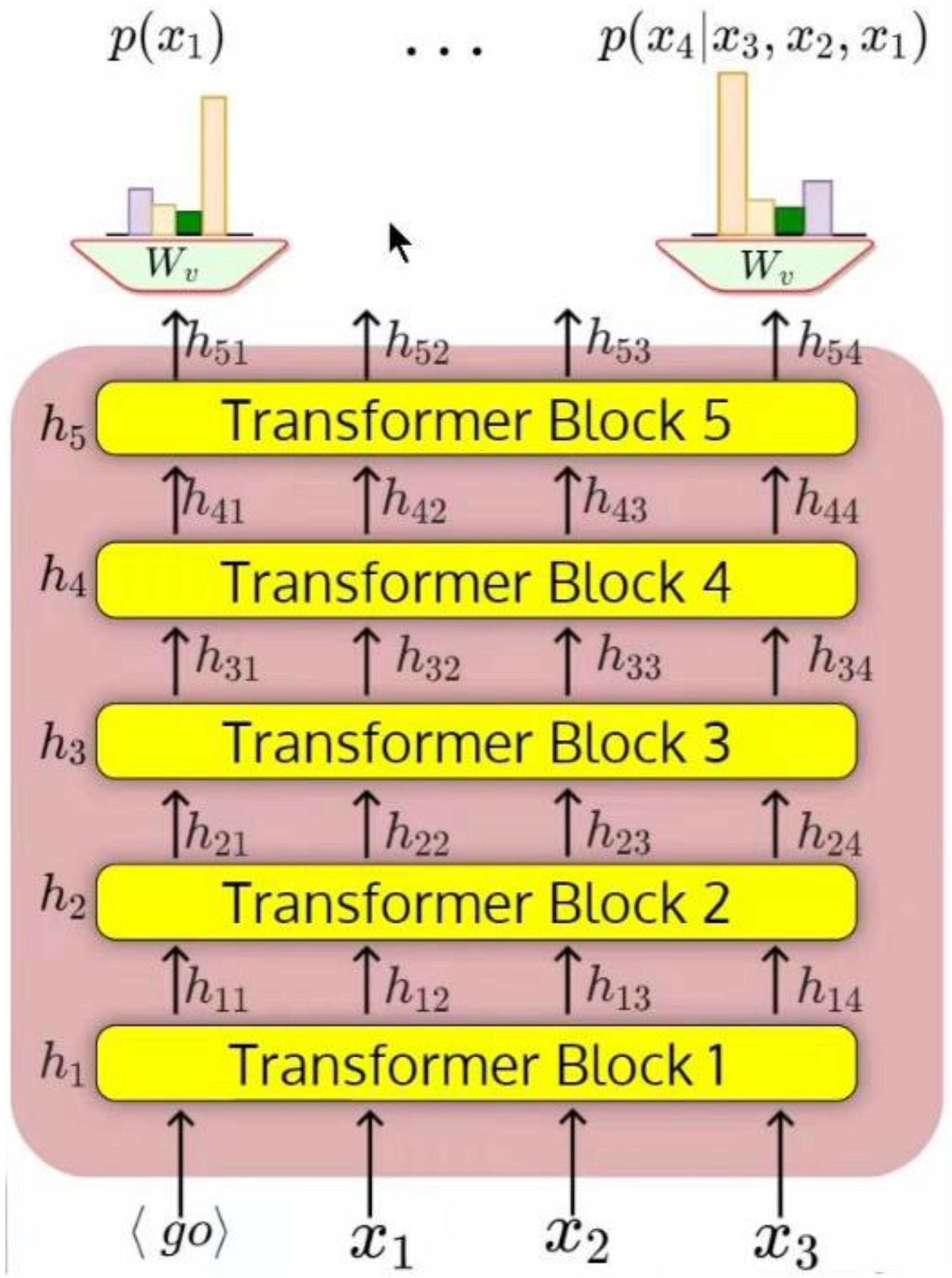
What is a GPT Model?

- Now we can create a stack (n) of modified decoder layers (called transformer block in the paper)
- Let, X denote the input sequence.
- $h_0 = X \in R^{T \times d_{\text{model}}}$
- $h_l = \text{transformer\_block}(h_{l-1}), \forall l \in [1, n]$
- Where $h_n[i]$ is the $i-$th output vector in h_n
- $P(x_i) = \text{softmax}(h_n[i] W_v)$



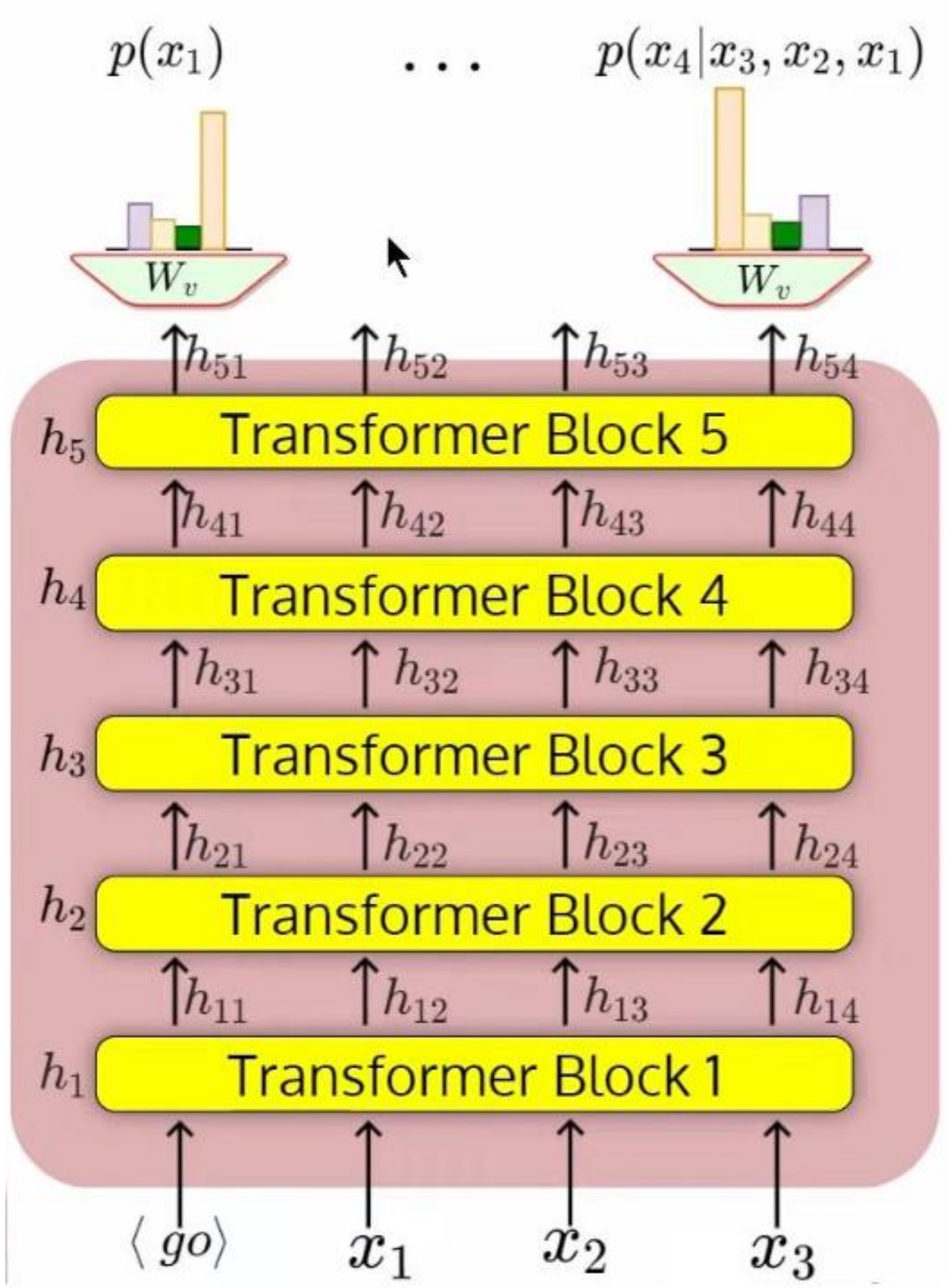
Step 1: GPT Model

- $h_0 = X \in R^{T \times d_{\text{model}}}$
- **Start with normal words → turn them into numbers**
- The input sentence (e.g., “I am going home”) is turned into a bunch of numbers.
- We call this starting bunch of numbers $\mathbf{h_0}$ (h-zero). It’s just a table:
- **Rows =** words (T rows, for example 512 words max)
- **Columns =** 768 numbers that describe each word → So h_0 is a big rectangle: 512 rows × 768 columns.



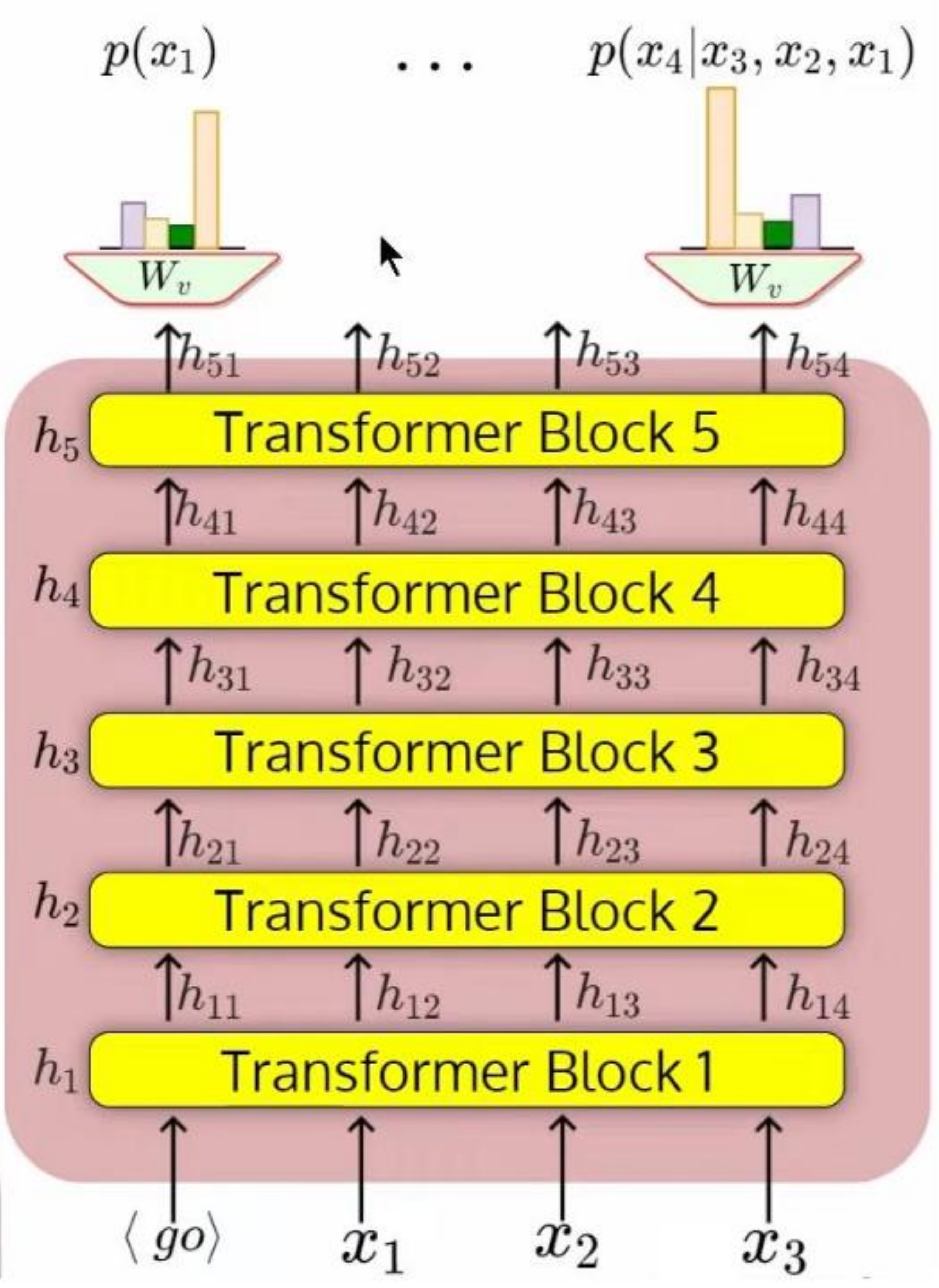
Step 2: GPT Model

- We have a magic **"thinking block"** We built one smart layer (called a transformer block)
- It looks at all the words so far and makes each word's numbers a little bit smarter.
- **We stack 12 of those magic blocks on top of each other** (n = 12 in GPT-1)
- The first block takes h_0 and gives us a better version → h_1
- The second block takes h_1 and gives us an even better version → h_2
- …
- The 12th block gives us the final super-smart version → h_{12} (we call it h_n)
- That's exactly what the equations say:
- h_1 = transformer_block(h_0)
- h_2 = transformer_block(h_1) … h_{12} = transformer_block(h_{11})



Step 3: GPT Model

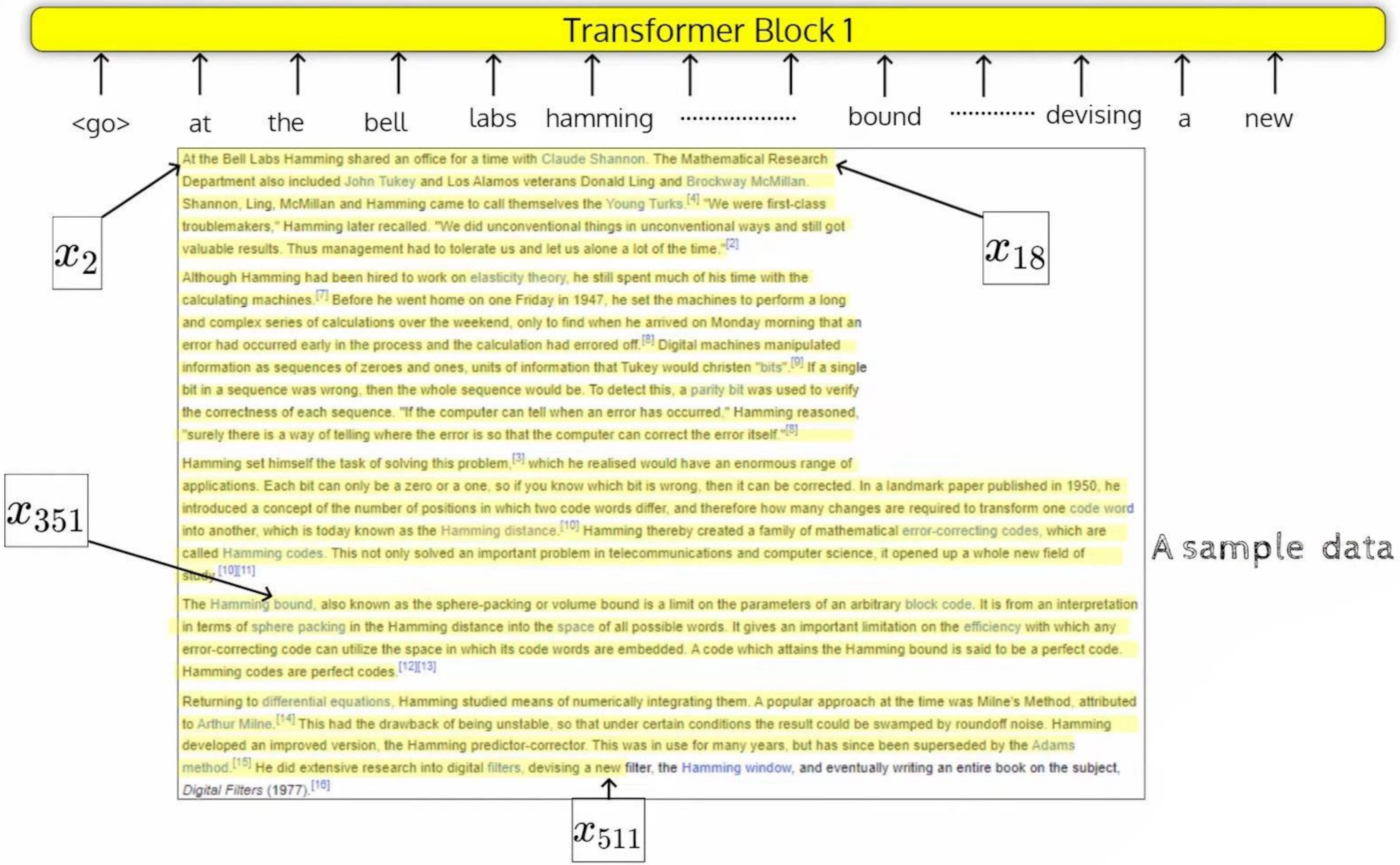
Operation	GPU does it on the whole rectangle at once?	Time taken
Make Q, K, V	Yes → 3 matrix multiplications on 512×768	~0.0001 sec
Q × K^T → attention scores	Yes → one big 512×512 matrix in one shot	~0.0002 sec
Apply causal mask	Yes → just subtract a 512×512 mask	instant
Softmax + multiply by V	Yes → back to 512×768	instant
Feed-forward (768→3072→768)	Yes → two big matrix multiplications	~0.0003 sec
Add & LayerNorm (residual)	Yes → element-wise on the whole 512×768	instant



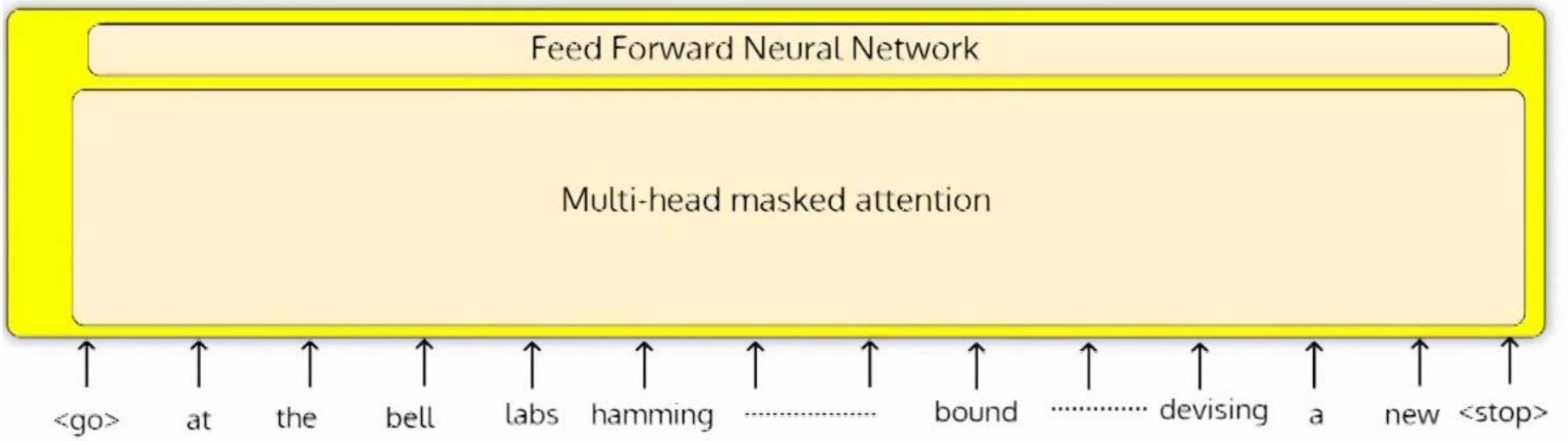
The Training

- **Training Data:** GPT-1 was trained on the **Book Corpus** dataset.
- A collection of over 7,000 unique books from various genres.
- They provide long, coherent stretches of text. This is crucial for learning grammar, reasoning, and long-range dependencies (e.g., connecting a character mentioned in Chapter 1 to their actions in Chapter 5).
- It contained roughly 1 billion words. While massive at the time, this is small compared to modern models trained on trillions of words from the entire internet.
- **Tokenization - Byte Pair Encoding (BPE):** The model needs to handle any word, even ones it has never seen.
- This allows the model to handle suffix which are frequent. Discuss Later (BERT)
- GPT-1 had a vocabulary of **40,000** such tokens.
- It Contains **12 decoder** layers (transformer blocks)
- **Context size:** 512
- **Attention heads:** 12
- **FFN hidden layer size:** 768 x 4 = 3072
- **Activation:** Gaussian Error Linear Unit (**GELU**)

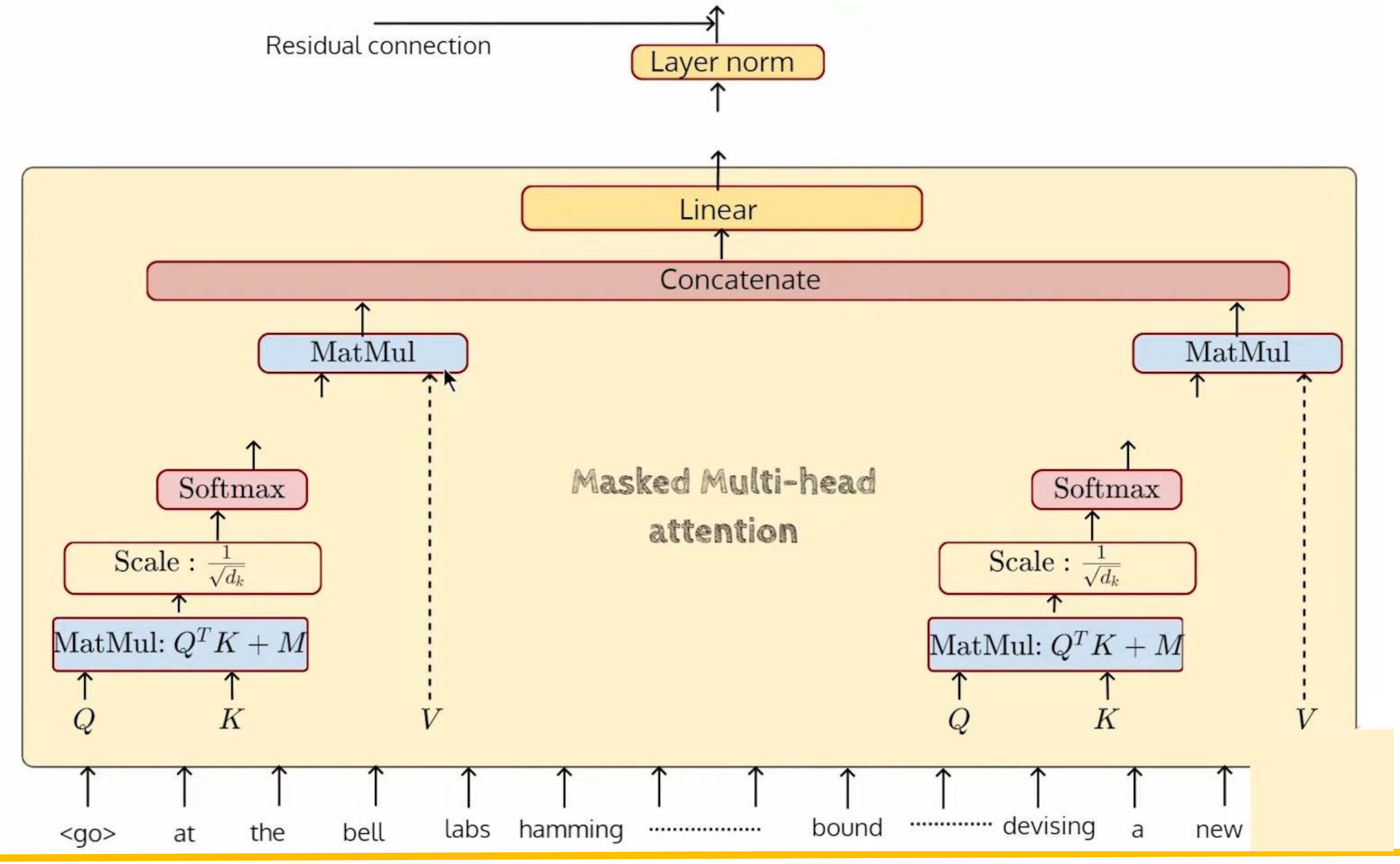
The Training



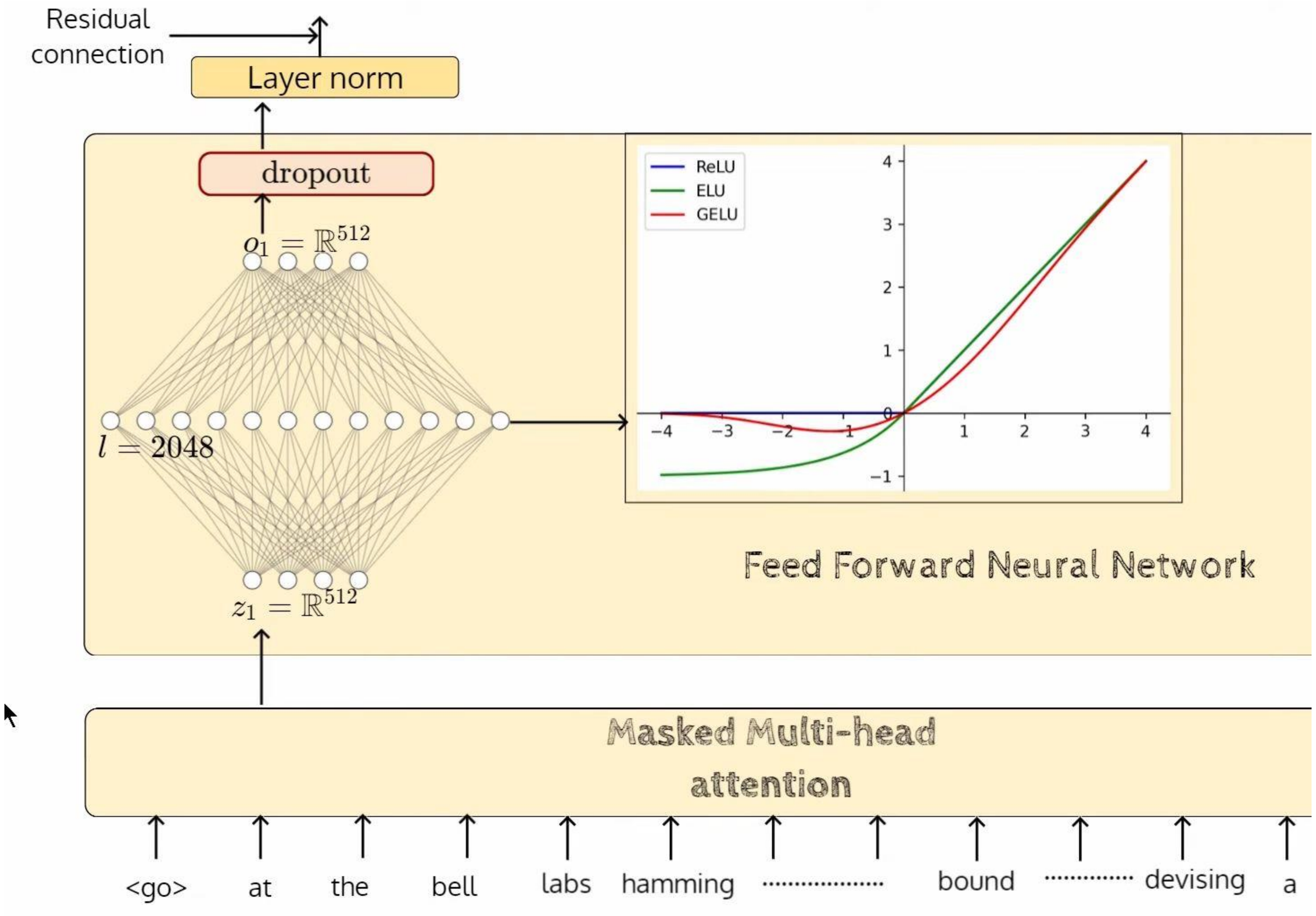
The Training



The Training



The Training



The Training

- **Token + Position Embeddings**
- **Embedding Layer Parameters**
- Token Embeddings: **|V| × embedding_dim**
- 40,478 × 768 = 31 × 10^6 = **31.1 M**
- **Position Embeddings:**
- **context length × embedding_dim**
- 512 × 768 = 0.3 × 10^6 = **0.3 M**
- **Total Embedding Parameters: 31.3 M**
- *The positional embeddings are also learned, unlike the original Transformer which uses fixed sinusoidal embeddings.*

- **Attention Parameters Per Block**
- **Attention Parameters (per transformer block)**
- Each attention head: $W^Q = W^K = W^V$
- (768 × 64): 3 × (768 × 64) ≈ **147 × 10^3** parameters per head
- **For 12 heads:** 12 × 147k ≈ **1.7 M**
- Linear output layer (W^O):**768 × 768 ≈ 0.6 M**
- **Total per block:** 1.7 M + 0.6 M = **2.3 M**
- For all 12 blocks: 12 × 2.3 M = **27.6 M**

The Training

- **Feed-Forward Network (FFN) Parameters**
- **FFN Parameters Per Block**
- **Two linear layers:** 2 × (768 × 3072) + biases (3072 + 768) = **4.7 × 10^6 = 4.7 M parameters per block**
- For all 12 blocks: 12 × 4.7 M = **56.4 M**
- **Final Number Everyone Remembers**
- **GPT-1 = ~117 Million Parameters** (the first real "large" language model in 2018, tiny by 2025 standards!)

Layer	Parameters (Millions)
Embedding Layer	31.3 M
Attention Layers (12)	27.6 M
FFN Layers (12)	56.4 M
Total	116.46 M