Large Language Model (LLM)

Key Concepts & Transformer

Large Language Model (LLM)

- An LLM is an AI model designed to understand, generate, and manipulate human language.
- “Large” refers to model capacity: trained on massive corpora and with billions+ of parameters.
- Parameters are the internal values adjusted during training to minimize prediction error.

Training Infrastructure and Duration for LLM

Model	GPUs Used	GPU Type	Training Duration
GPT-3 (175B)	~10,000	NVIDIA V100	~34 days
LLaMA 3 (70B)	~2,000	NVIDIA A100	~20 days
DeepSeek-R1 (671B)	~2,048 nodes with multiple H800	NVIDIA H800	~2 months (approx.)

What Can LLMs Do?

Key Concepts

Tokens & Tokenization

- A token can be a word, sub-word, or punctuation.
- Sub-word splits help generalization

“dog” → one token
“dogs” → “dog” + “s”
“unbelievable” → “un”, “believ”, “able”

- 128K tokens ≈ ~90–100k words

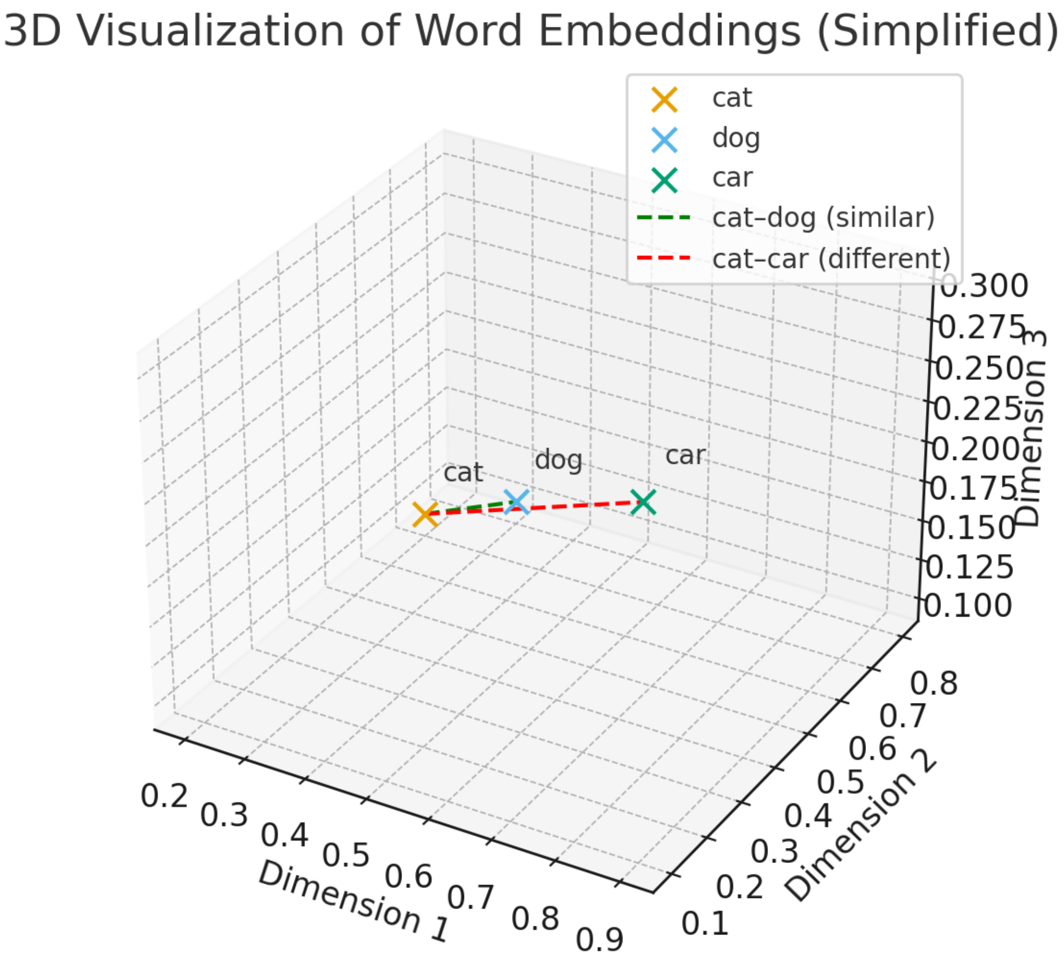
Embeddings

- An embedding is a way to turn words into numbers, so that a computer can understand and work with them.

- Tokens map to vectors that encode meaning in very **high-dimensional** space (often hundreds or even thousands of dimensions).

Similar concepts → nearby vectors;
Different concepts → farther apart.

Word	Embedding
cat	[0.2, 0.7, 0.1]
dog	[0.3, 0.8, 0.1]
car	[0.9, 0.1, 0.3]

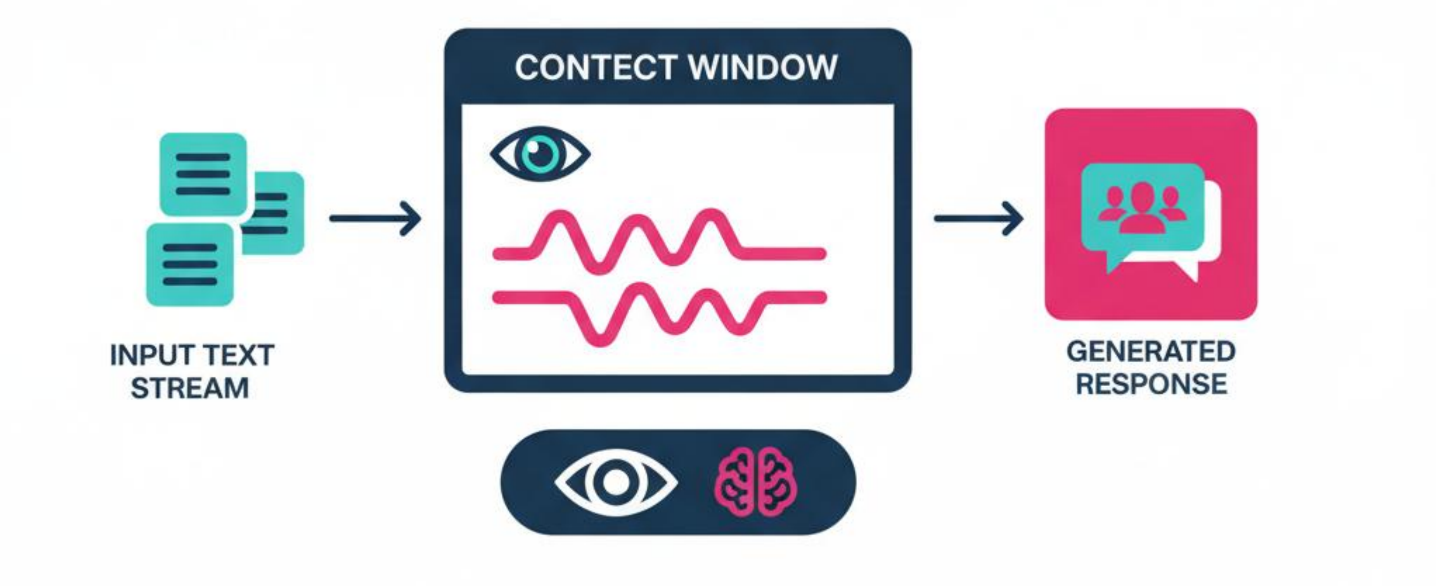


- “Cat” and “Dog” will be **close together** (because their meanings are similar, both animals).

- “Car” will be **farther away** (because it’s a machine, not an animal).

Context Window (“Memory”)

A context window is the amount of text (tokens) that an LLM can “**read**,” and “**remember**” at one time while generating a response.



Read

- **Single-Pass Fit**

A 100,000-token book can fit into a 128K-token window.

- **Too Large**

If the text is 200,000 tokens (65% of the book).
It must summarize parts outside the window.
Unless you specifically give parts to read it.

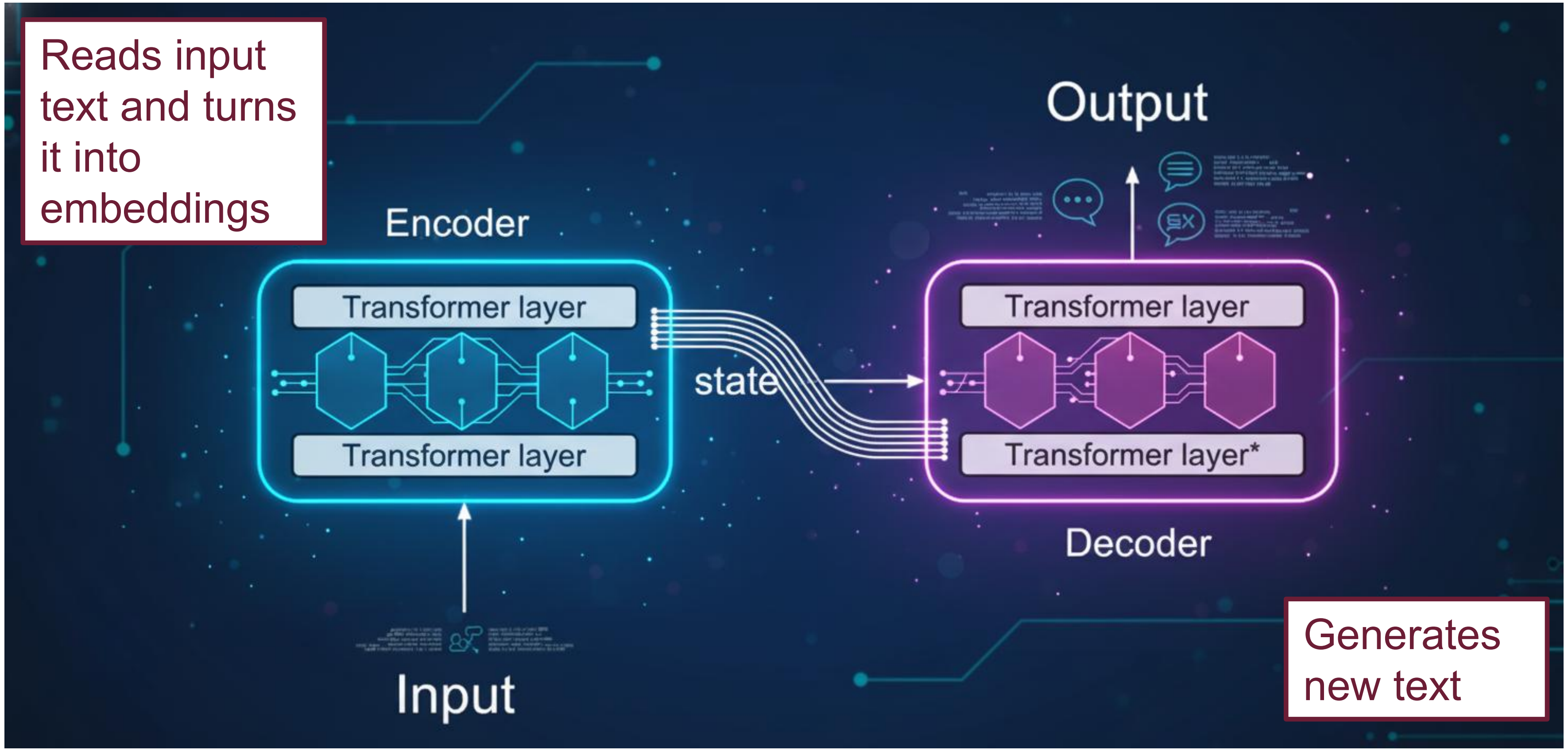
Remember

Type of “memory”	What it means
Context memory	The model “remembers” what you said earlier in the chat. (Temporary).
Training memory	Permanent knowledge learned from data
User memory	Saved info about you to personalize responses

Pipeline

$$\underbrace{\text{text} \rightarrow \text{tokens} \rightarrow \text{embeddings}}_{\text{input}} \rightarrow \overset{\text{Encode} \rightarrow \text{Decode}}{\boxed{\text{Memory}}} \rightarrow \underbrace{\text{tokens} \rightarrow \text{embeddings} \rightarrow \text{text}}_{\text{output}}$$

Transformers Architecture



UTM
UNIVERSITI TEKNOLOGI MALAYSIA

Transformers Architecture

A Transformer is a type of neural network architecture introduced by Vaswani et al. in 2017 in the paper: *"Attention Is All You Need."*
New mechanism called **self-attention**.

Self-attention lets the model look at all words in a sentence at once and learn which words are most relevant to each other.

"The cat sat on the mat because it was soft."
When processing "it," the model uses self-attention to see that "it" refers to "the mat," not "the cat."

Ashish Vaswani: The Mind that Rewrote the Rules of AI

“The cat sat on the mat”

$\underbrace{\text{The cat}}_{\text{Subject}}\ \underbrace{\text{sat on the mat}}_{\text{Predicate}}$

Clause element	Example	Question it answers
Subject	The cat	Who sat?
Verb	sat	What did the cat do?
Prepositional phrase	on the mat	Where did the cat sit?

Grammar connects words logically

Self-Attention (K - Keys, Q - Queries, V - Values)

Word	Role	Meaning in Attention
Sat	Q	Asks “Who?” and “Where?”
Cat	K → Q+ K = V	Answers “Who” (the subject)

Attention math says “Query (verb) finds its Keys (subject)” and mixes their Values (V).

Self-Attention (K - Keys, Q - Queries, V - Values)

Phrase: "The cat sat"

Word	Embedding
The	[0.1, 0.2, 0.3]
Cat	[0.5, 0.4, 0.7]
Sat	[0.8, 0.9, 0.3]

Q = [[1, 0, 0],
[0, 1, 0],
[0, 0, 1]]

K = [[0.5, 0, 0],
[0, 0.5, 0],
[0, 0, 0.5]]

V = [[1, 0, 0],
[0, 1, 0],
[0, 0, 1]]

Q = embedding × Q
K = embedding × K
V = embedding × V

Example for “Cat”:

Q_cat = [0.5, 0.4, 0.7]
K_cat = [0.25, 0.2, 0.35]
V_cat = [0.5, 0.4, 0.7]

Compute Attention Scores (Q × K)

Phrase: "The cat sat"

Let's focus on **"Sat"**.

Q_sat = [0.8, 0.9, 0.3]

K_The = [0.05, 0.1, 0.15]
K_Cat = [0.25, 0.2, 0.35]
K_Sat = [0.4, 0.45, 0.15]

Pair	K	Q_sat·K
Sat–The	(0.8×0.05 + 0.9×0.1 + 0.3×0.15) = 0.08 + 0.09 + 0.045 = **0.215**	Low similarity
Sat–Cat	(0.8×0.25 + 0.9×0.2 + 0.3×0.35) = 0.2 + 0.18 + 0.105 = **0.485**	High similarity
Sat–Sat	(0.8×0.4 + 0.9×0.45 + 0.3×0.15) = 0.32 + 0.405 + 0.045 = **0.77**	Highest similarity

Apply Softmax to Get Attention Weights

Softmax converts the scores into probabilities (that sum to 1).

Let's focus on **"Sat"**.

exp(0.21) = 1.23
exp(0.49) = 1.63
exp(0.77) = 2.16

Total = 5.02

Weights = [1.23/5.02, 1.63/5.02, 2.16/5.02] = **[0.24, 0.32, 0.43]**

"The" "Cat" "Sat"

"Sat" paid **most attention to "Cat"** because the action depends on the subject.

Combine the Values (V) - Attention Output

Each word has a Value vector (V).

Let's focus on **"Sat"**.

V_The = [0.1, 0.2, 0.3]
V_Cat = [0.5, 0.4, 0.7]
V_Sat = [0.8, 0.9, 0.3]

Weights =**[0.24, 0.32, 0.43]**

Attention output for Sat = 0.24×**V_The** + 0.32×**V_Cat** + 0.43×**V_Sat**

= [0.24×0.1 + 0.32×0.5 + 0.43×0.8,
0.24×0.2 + 0.32×0.4 + 0.43×0.9,
0.24×0.3 + 0.32×0.7 + 0.43×0.3]

= [0.024 + 0.16 + 0.344, 0.048 + 0.128 + 0.387, 0.072 + 0.224 + 0.129]

= [0.528, 0.563, 0.425] "Sat"'s new meaning vector (new embedding)

Add & Norm

It also helps the model keep both the original meaning and the new context from attention.

Add

Original embedding for “sat” = [0.8, 0.9, 0.3]

Attention output for “sat” = [0.52, 0.56, 0.42]

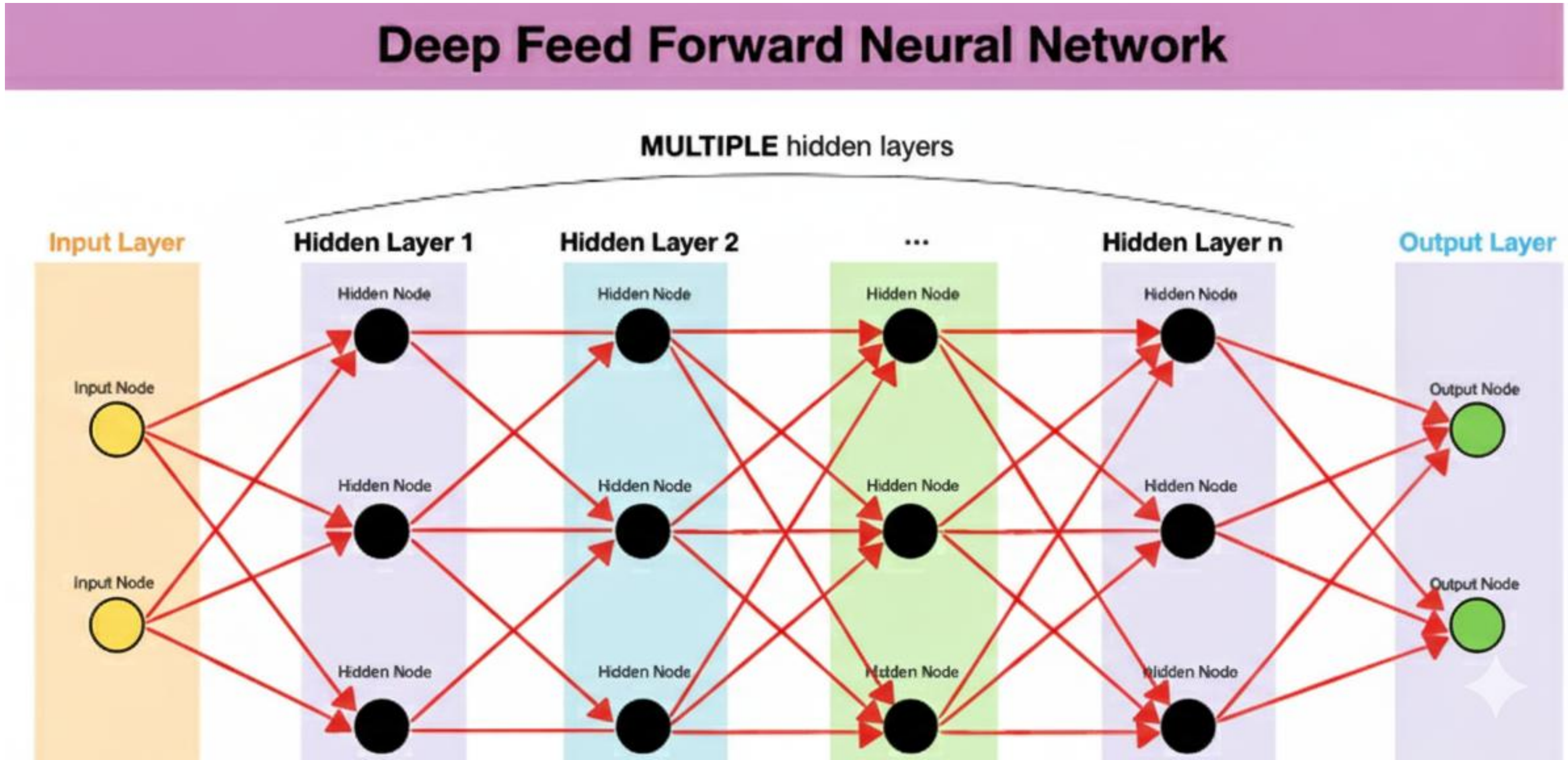
original + attention

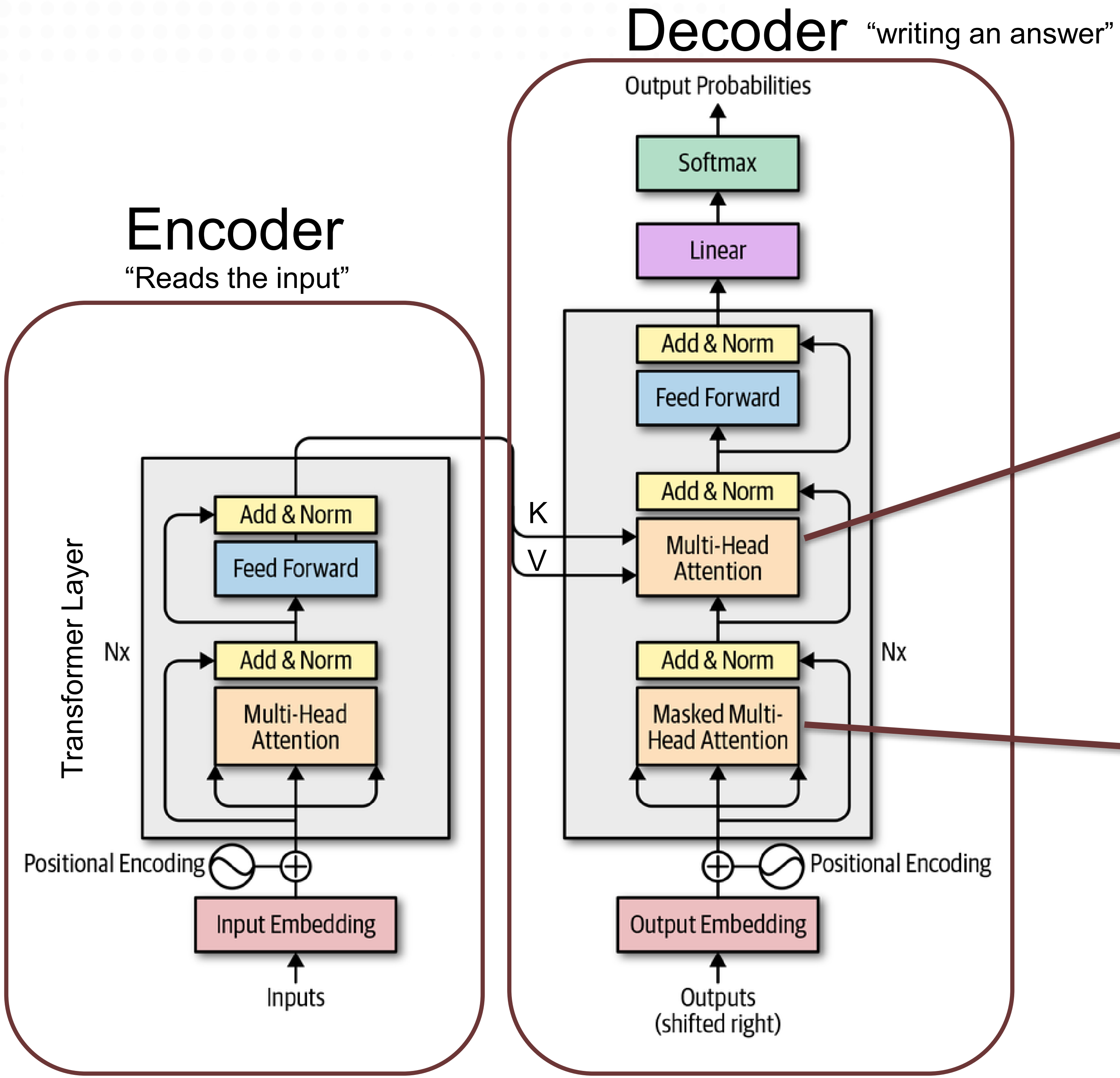
[0.8+0.52, 0.9+0.56, 0.3+0.42] = **[1.32, 1.46, 0.72]**

Norm

[1.32, 1.46, 0.72] → Norm (μ=0, σ=1) → **[0.40, 0.77, -1.17]**

Feed Forwad Neural Network





Uses Encoder outputs (K,V) + Decoder queries (Q)

Each output word looks only at *previous* words

"The"	"cat"
position 1	position 2

Comparison of Transformer and Modern Architectures

Model / Version	Launch Year	Uses Encoder Output?	Architecture
Transformer (original)	📅 2017	✅ Yes — decoder attends to encoder output (cross-attention)	Encoder + Decoder
BERT	📅 2018	✅ Yes — uses only encoder for understanding	Encoder-only
GPT-1	📅 2018	❌ No — decoder-only, next-token prediction	Decoder-only
GPT-2	📅 2019	❌ No — improved decoder-only generation	Decoder-only
T5	📅 2019	✅ Yes — both halves (encoder–decoder text-to-text)	Encoder–Decoder
BART	📅 2019	✅ Yes — both halves (denoising autoencoder)	Encoder–Decoder
GPT-3	📅 2020	❌ No — larger decoder-only model	Decoder-only
GPT-4	📅 2023	❌ No — advanced decoder-only with multimodal capability	Decoder-only
Llama 2	📅 2023	❌ No — open-source decoder-only model by Meta	Decoder-only
Llama 3	📅 2024	❌ No — improved decoder-only with larger context	Decoder-only
DeepSeek-R1	📅 2024	❌ No — decoder-only, reasoning-optimized (uses <think> tokens)	Decoder-only
GPT-5	📅 2025	❌ No — latest decoder-only generation model	Decoder-only

THANK YOU

Centre for Artificial Intelligence and Robotics

research.utm.my/cairo/