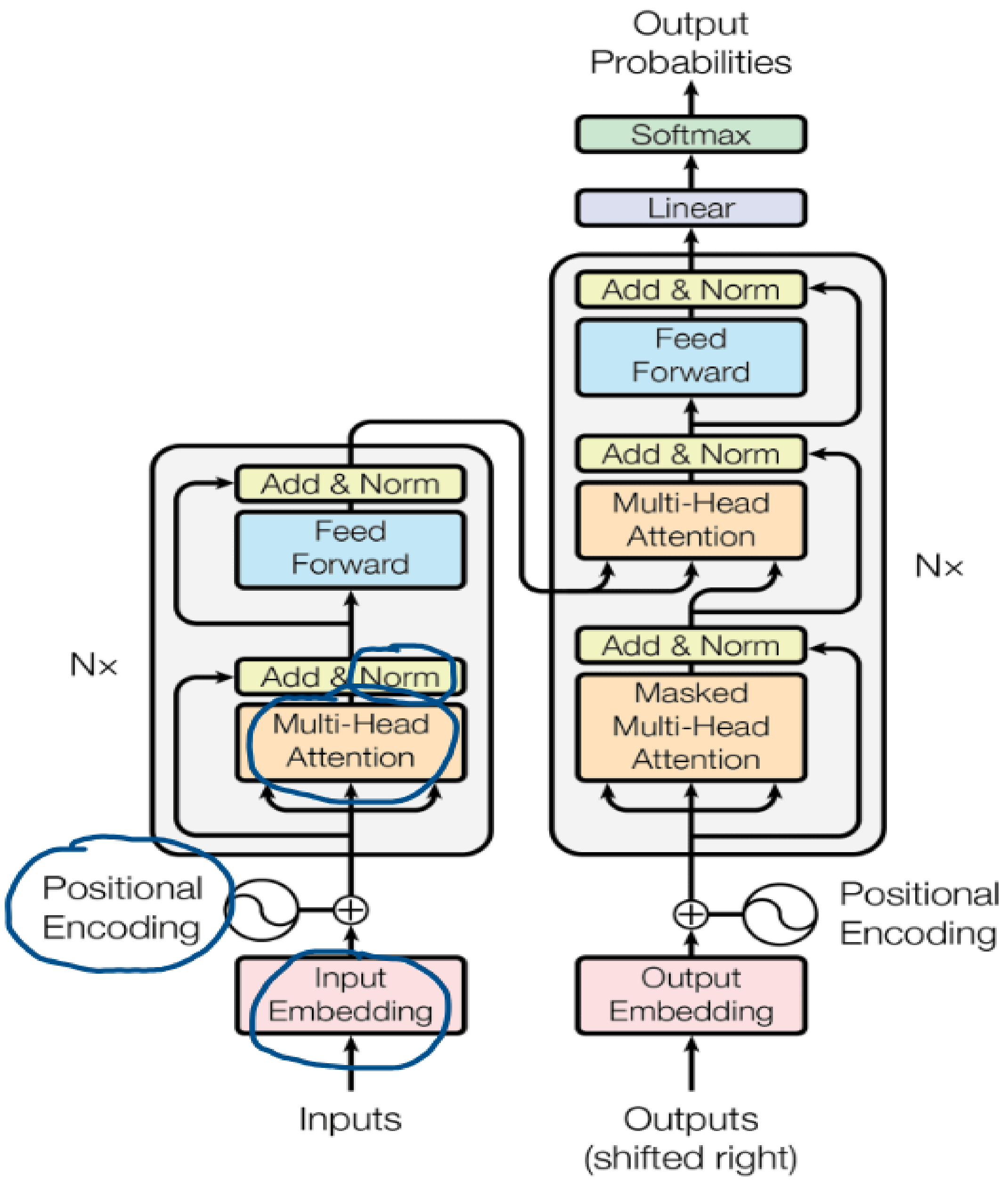
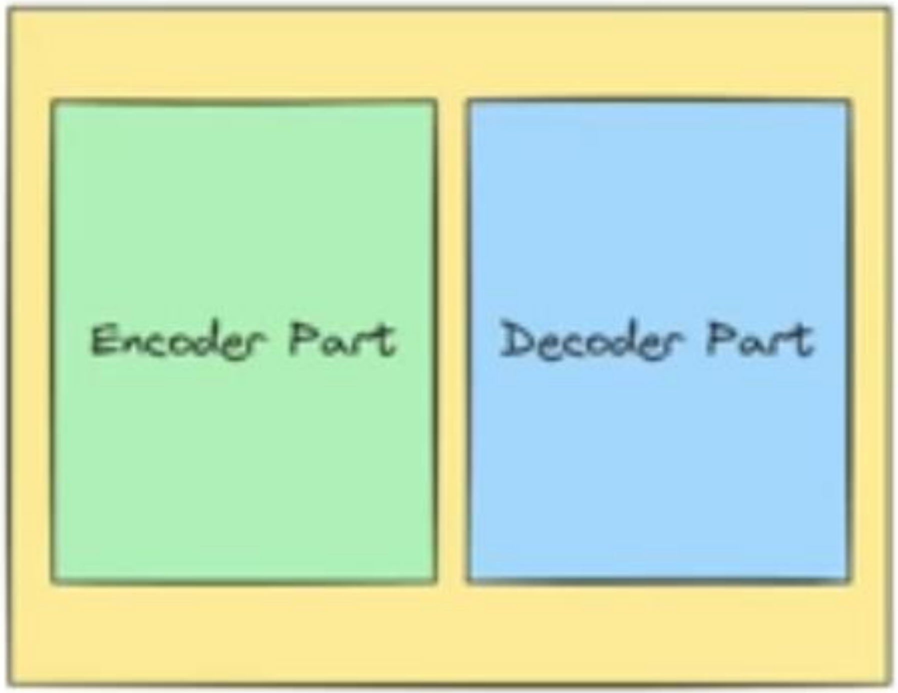
Transformer-Encoder

Dr. Muhammad Safyan

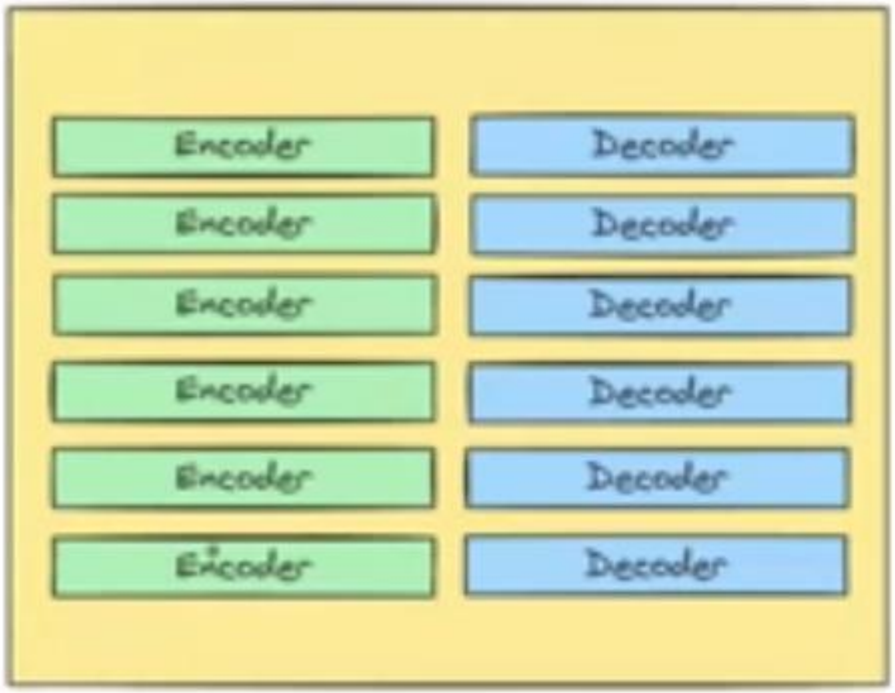
- Encoder=6
- Decoder=6



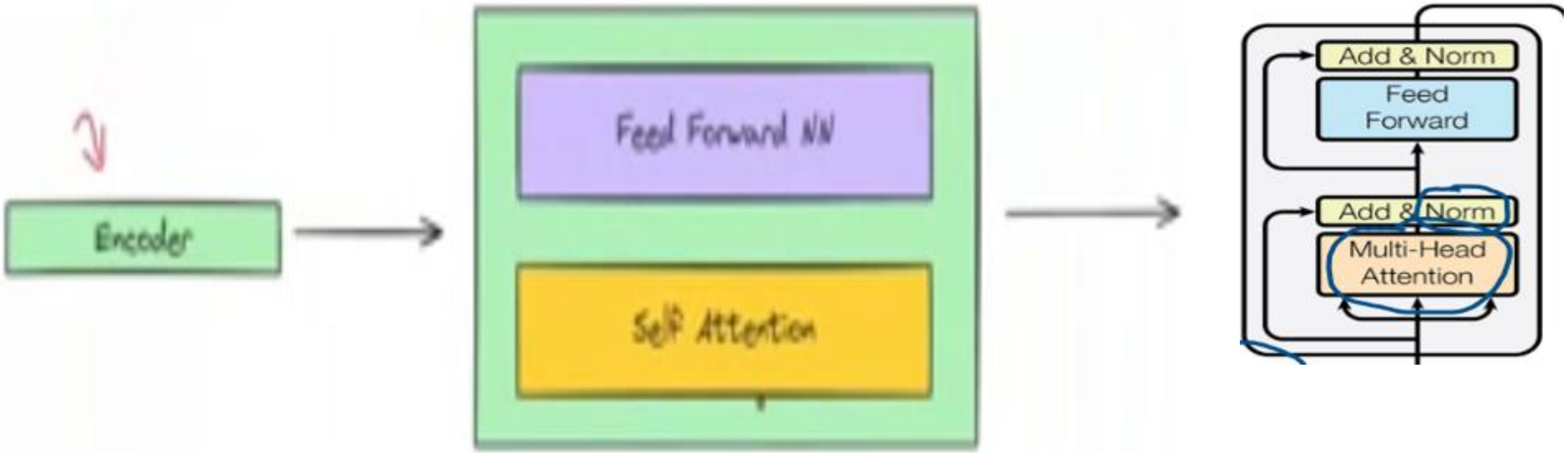
- All the blocks are identical



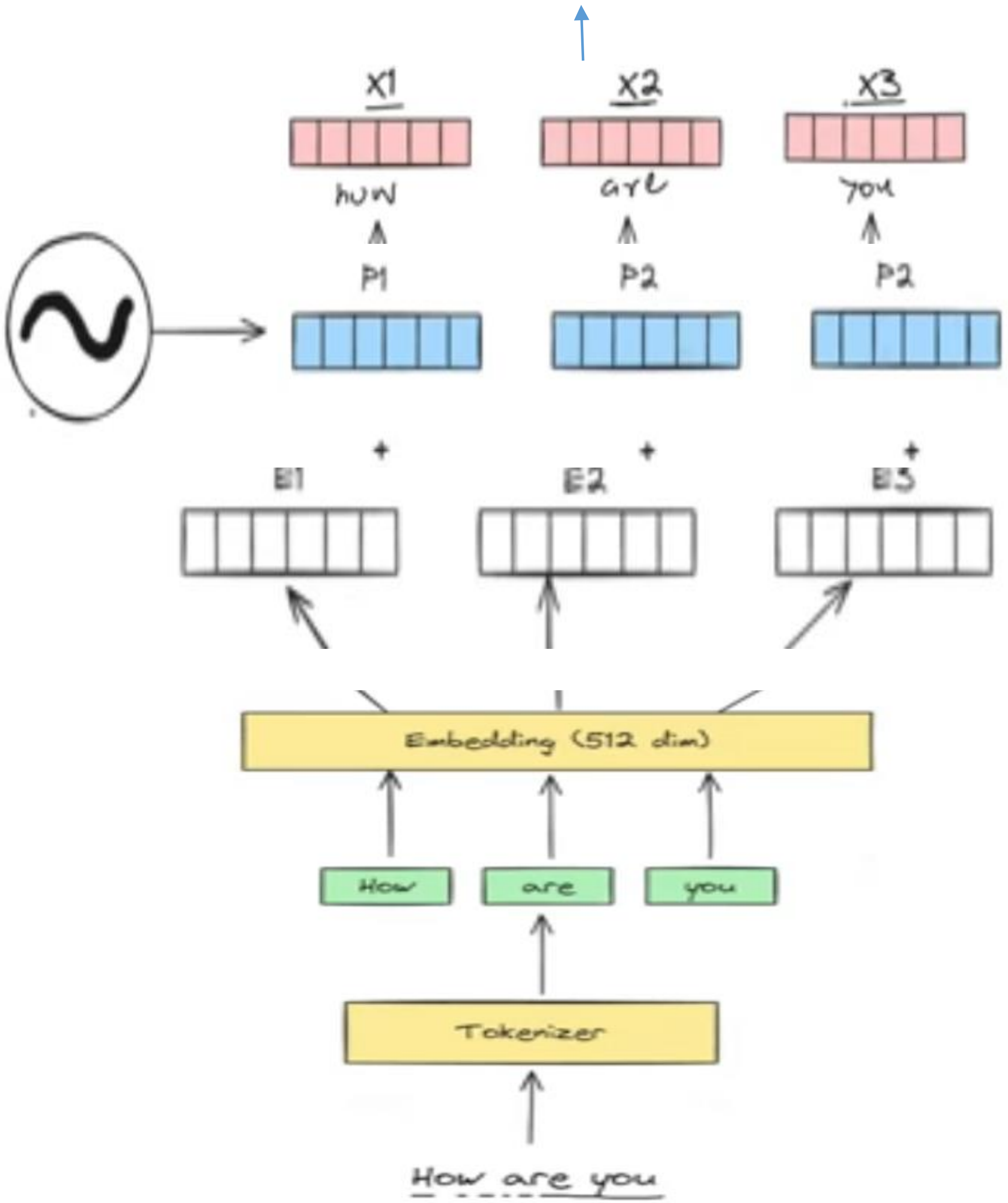
Transformer

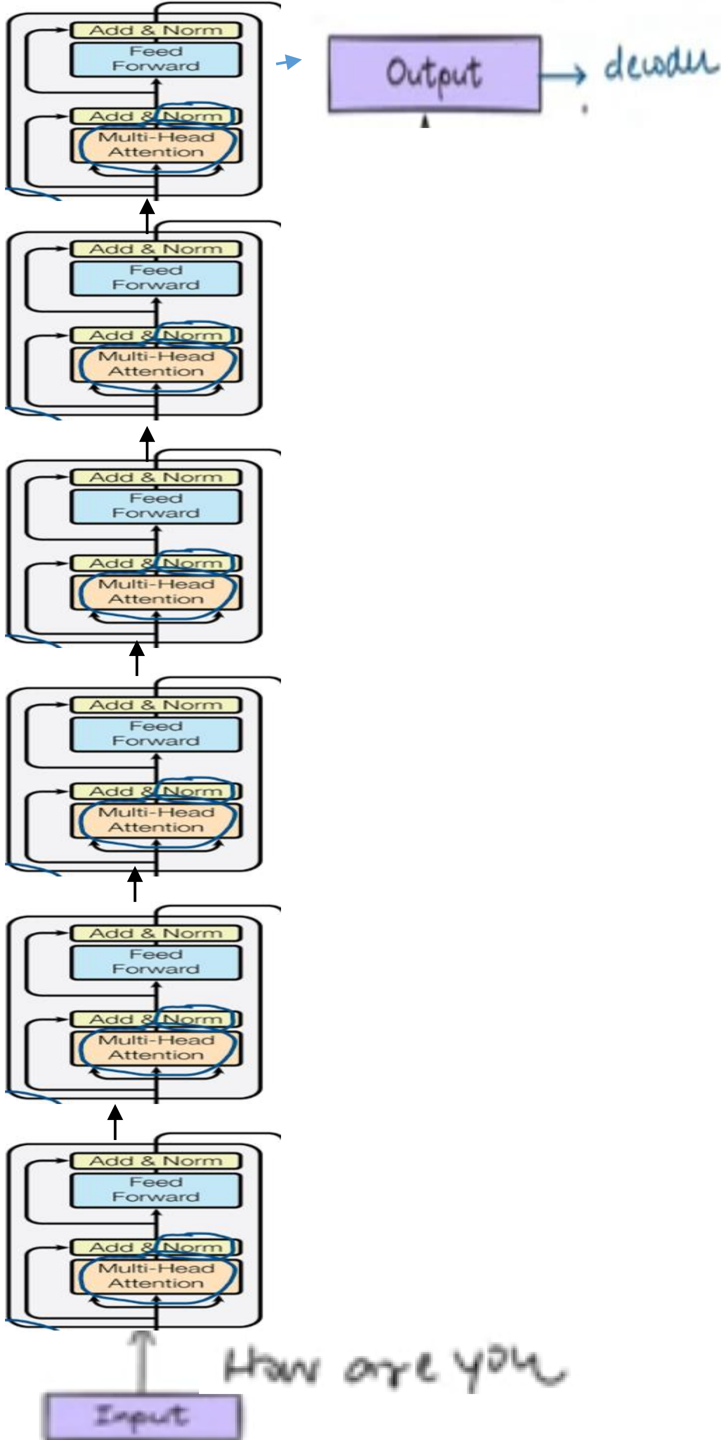


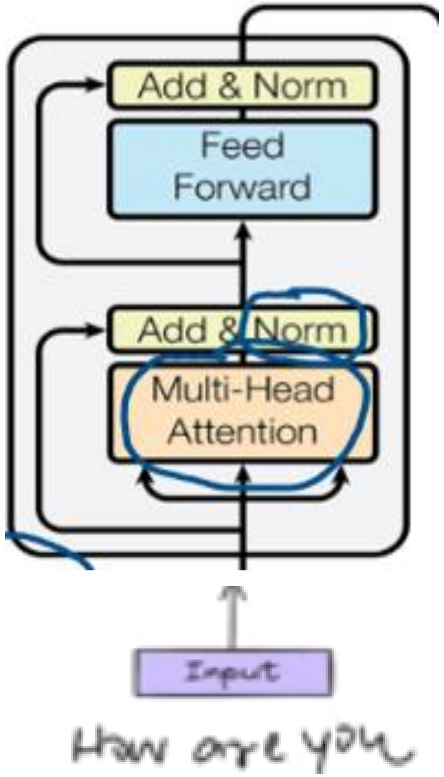
Transformer

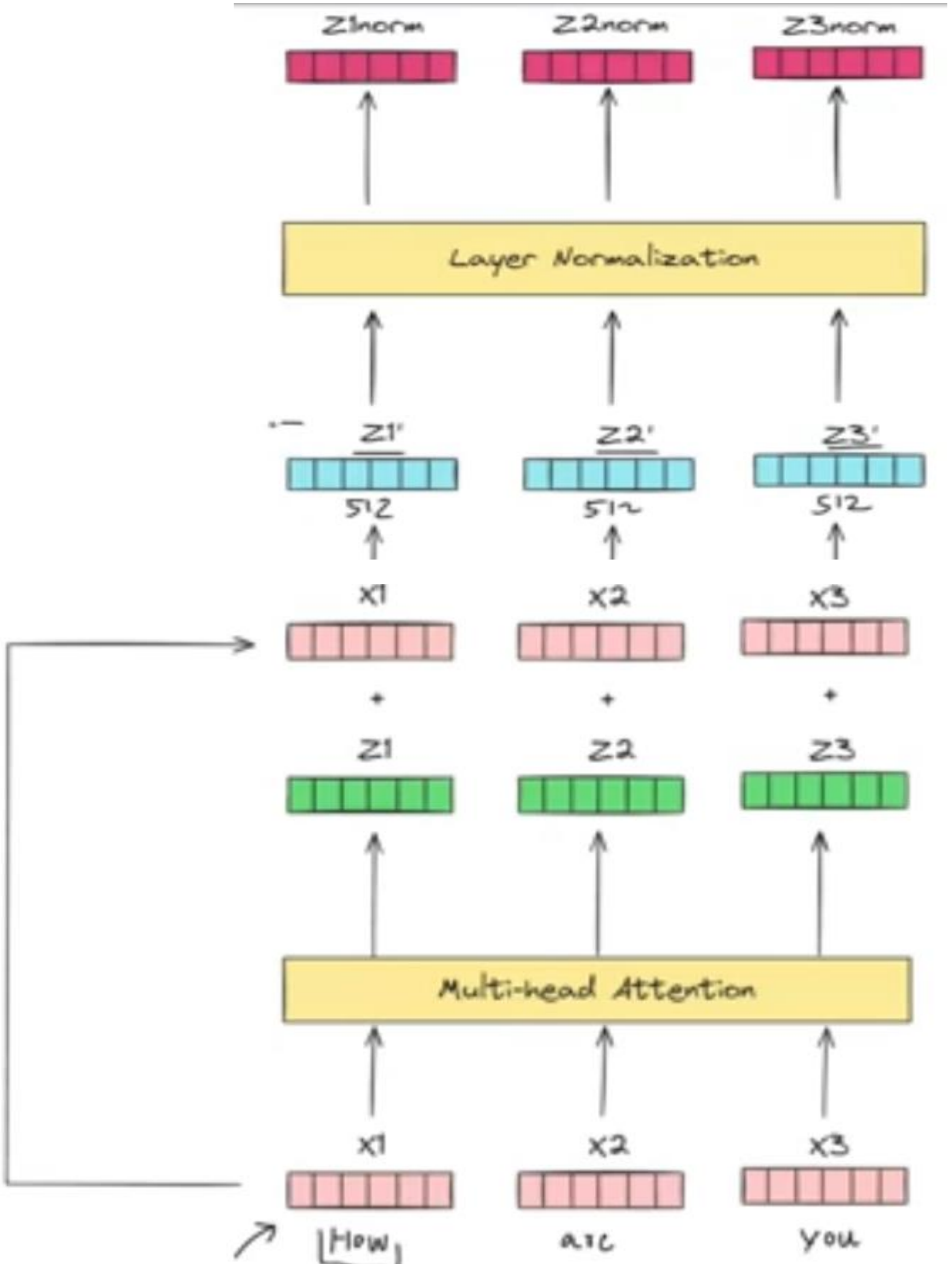

Encoder
Feed Forward NN
Self Attention
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention

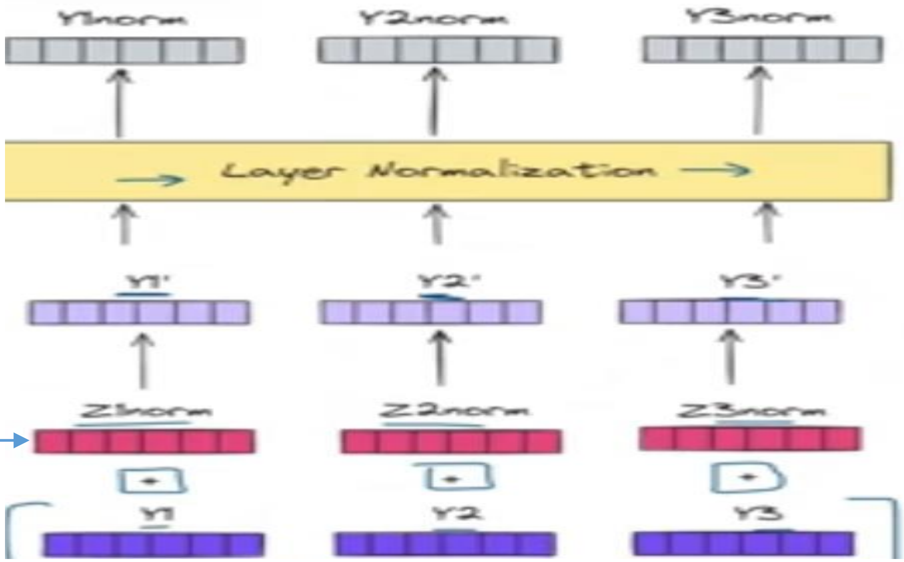
Encoder

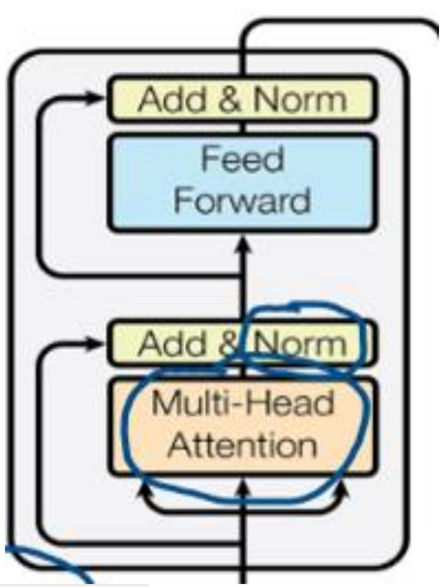

X1
X2
X3
how
are
you
P1
P2
P2
+
E1
+
E2
+
E3
Embedding (512 dim)
How
are
you
Tokenizer
How are you

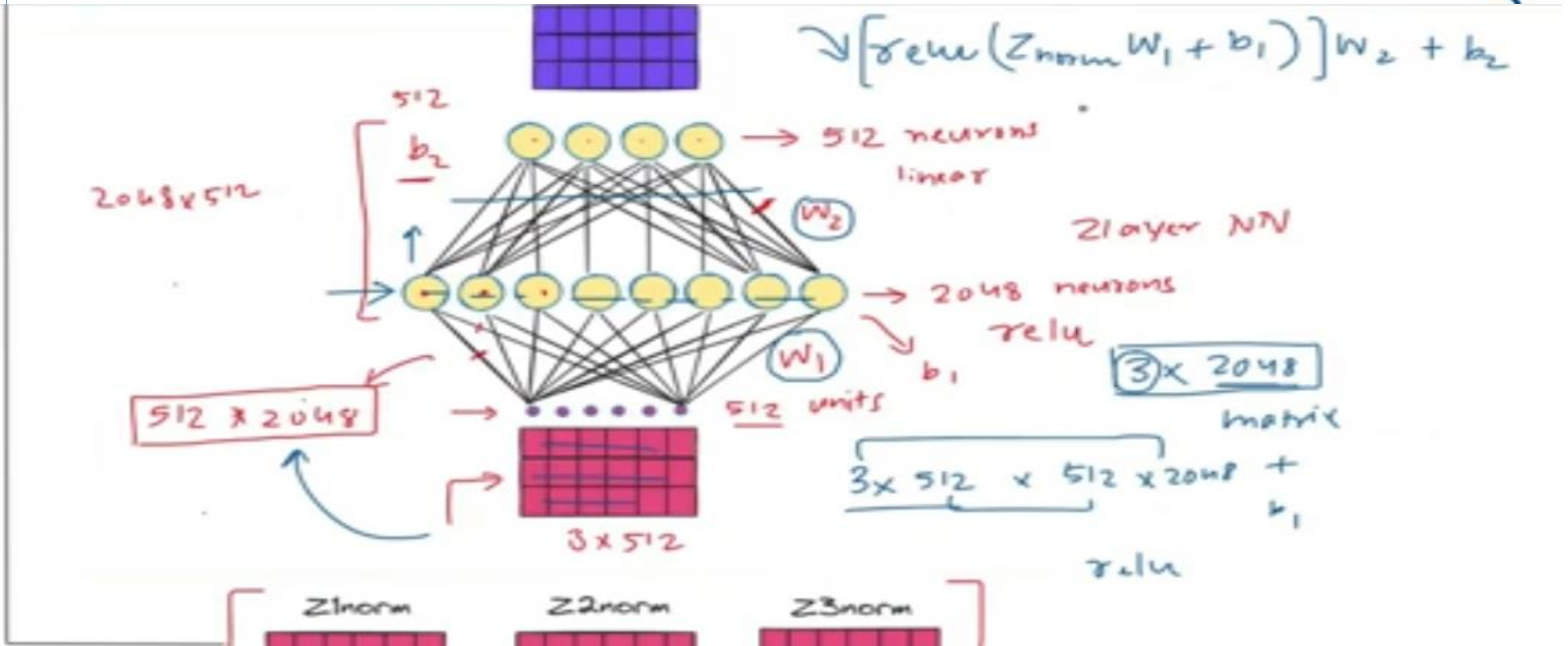

Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Output
decoder
How are you
Input


Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Input
How are you

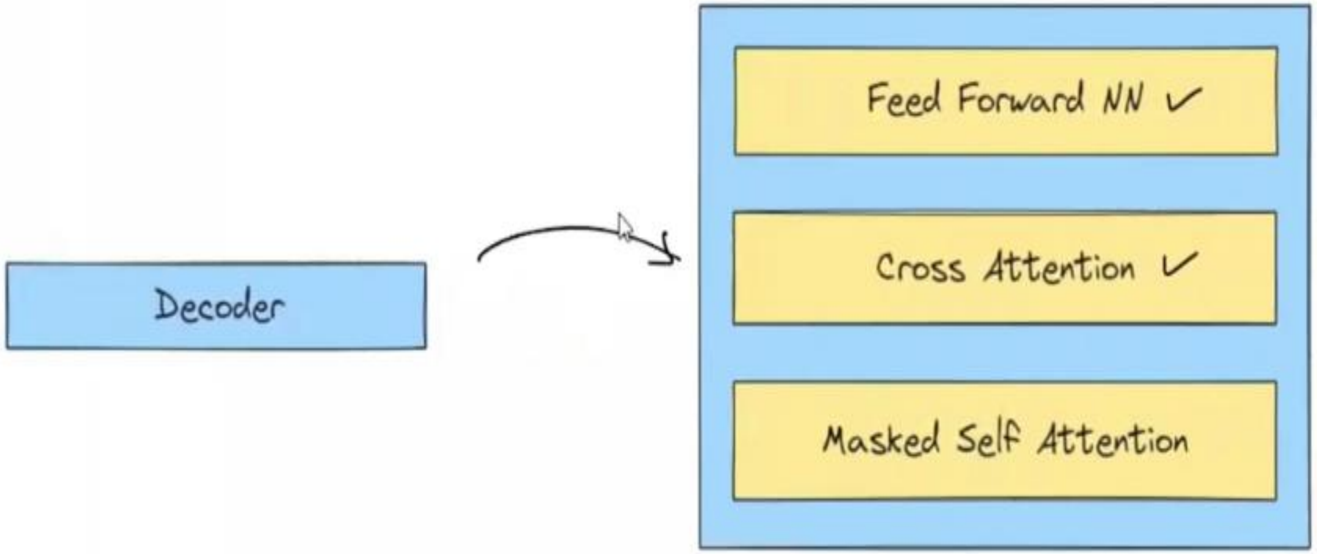

Z1norm
Z2norm
Z3norm
Layer Normalization
Z1'
Z2'
Z3'
512
512
512
X1
X2
X3
+
+
+
Z1
Z2
Z3
Multi-head Attention
X1
X2
X3
How
are
you


Y1norm
Y2norm
Y3norm
Layer Normalization
Y1'
Y2'
Y3'
Z1norm
Z2norm
Z3norm
+
+
+
Y1
Y2
Y3


Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention


[relu(Znorm W1 + b1)] W2 + b2
512
b2
→ 512 neurons
linear
2048x512
W2
2 layer NN
→ 2048 neurons
relu
b1
W1
3x 2048
matrix
512 x 2048
512 units
3x512 x 512 x 2048 +
b1
relu
3x512
Z1norm
Z2norm
Z3norm

1. Why use residual connections?
2. Why use a FFNN?
3. Why use 6 encoder blocks?

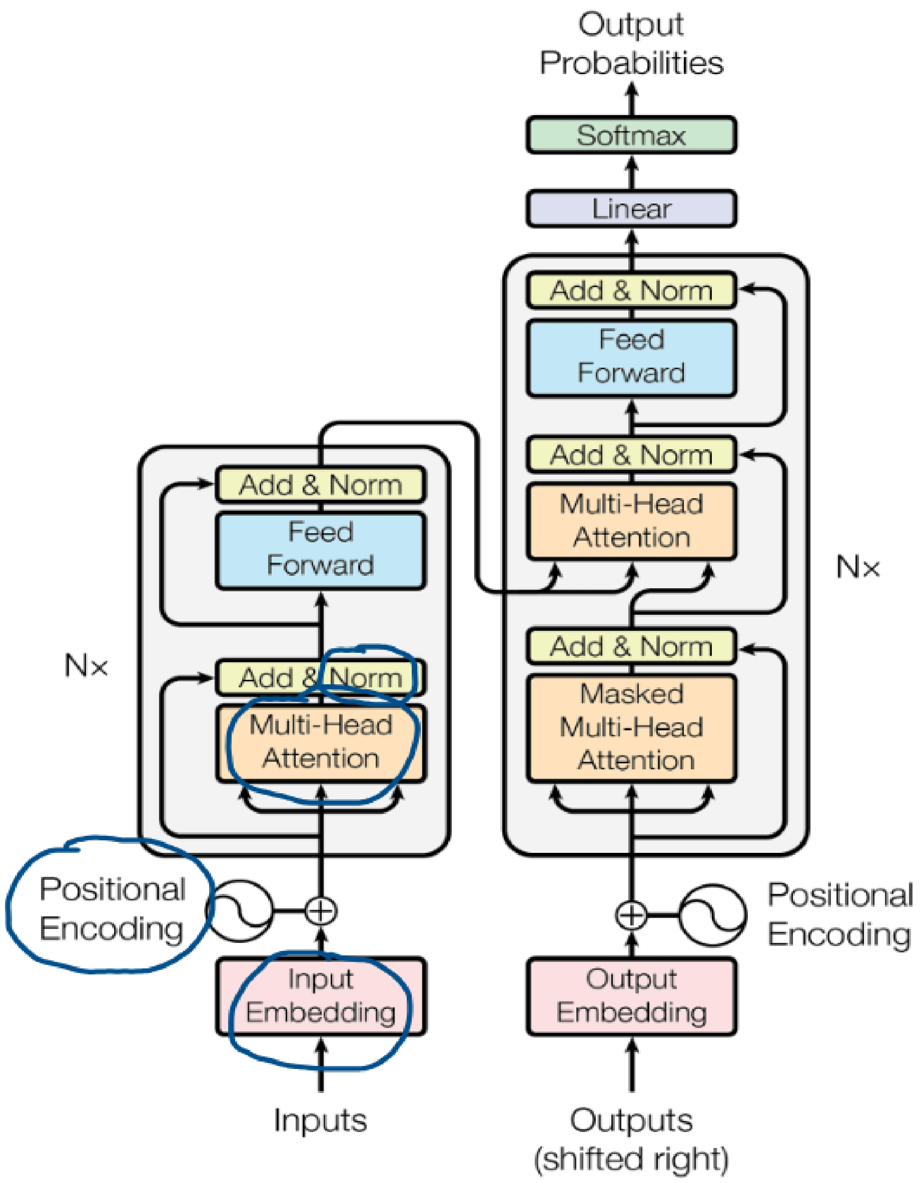


Decoder

Feed Forward NN ✓

Cross Attention ✓

Masked Self Attention



Output Probabilities

Softmax

Linear

Add & Norm

Feed Forward

Add & Norm

Multi-Head Attention

Add & Norm

Masked Multi-Head Attention

Nx

Add & Norm

Feed Forward

Add & Norm

Multi-Head Attention

Nx

Positional Encoding

Input Embedding

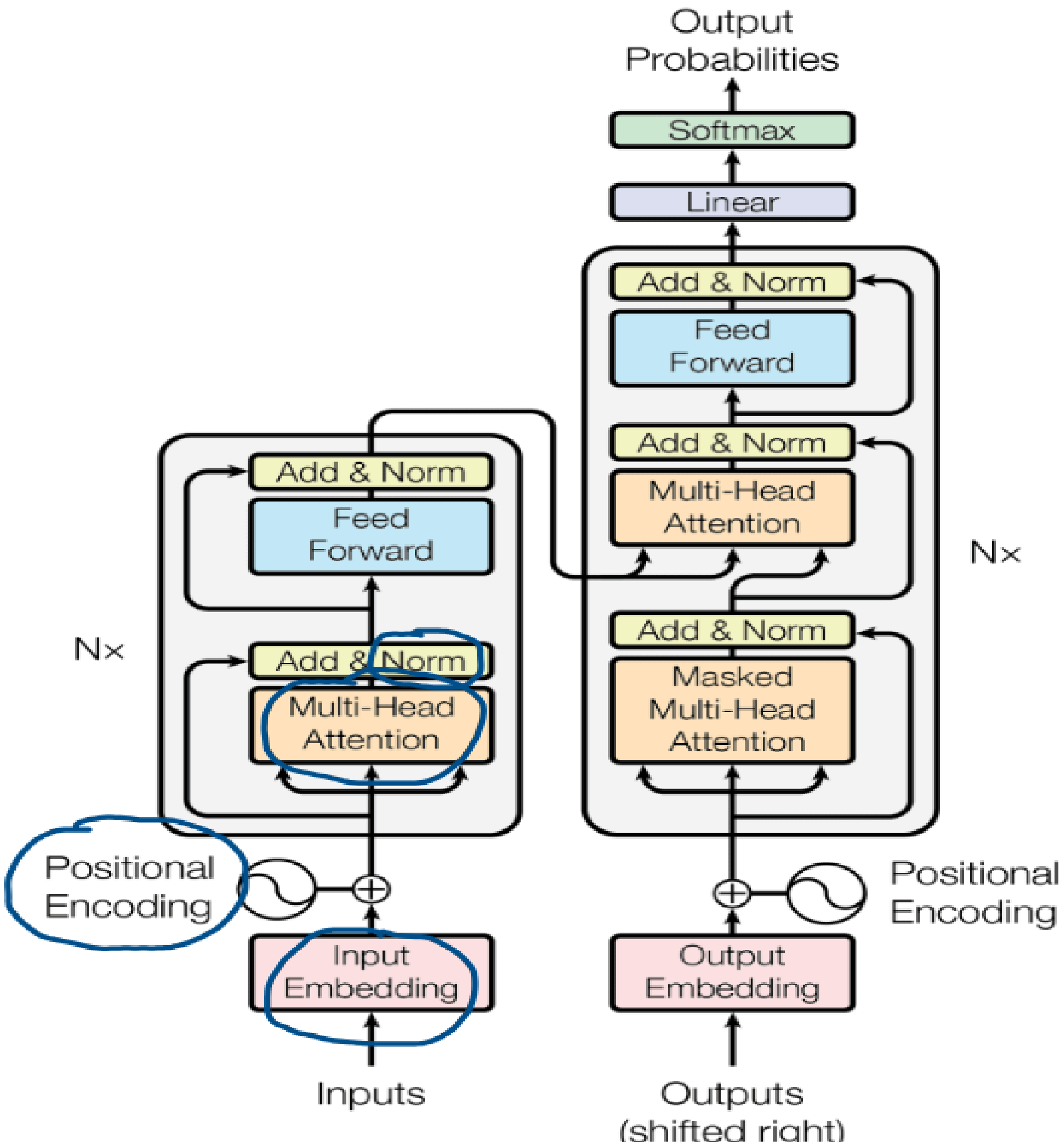
Inputs

Positional Encoding

Output Embedding

Outputs (shifted right)

- Encoder=6
- Decoder=6



If the Transformer encoder is already powerful, why do we need the decoder?

- Because:
-
- 👉 The encoder alone CANNOT generate output text.
- It can only understand the input sequence.
- The decoder is responsible for GENERATING the translated sentence (token by token) using the encoder’s representations.
-
- 🔥 Short Answer
- Encoder = Understands the source sentence.
-
- Decoder = Generates the target sentence based on this understanding.

- Encoder’s Job → Understanding
- The encoder takes an input sentence, for example:
-
- "How are you?"
-
- It produces deep contextual embeddings like:
-
- meaning of each word
-
- grammar structure
- relationships between words
-
- long-range dependencies

- BUT the encoder does NOT generate

- It only produces hidden representations.
- 2. Decoder’s Job → Generating Output Language
- The decoder takes two things:
-
- (a) Encoder output
- This is the meaning of the source sentence.
-
- (b) The previously generated tokens

-
- ⍰ Analogy
- Encoder = Brain that understands English
-
- Decoder = Mouth that speaks Urdu
- (using the understanding created by the brain)

Is chat gpt has both parts?

- No — ChatGPT does not use both encoder and decoder parts.
- ChatGPT (GPT-3, GPT-3.5, GPT-4, GPT-5, etc.) is a decoder-only Transformer.
- ✅ ChatGPT Architecture = Decoder-Only Transformer
- All understanding and generation are done inside a stack of decoder layers.

Why Decoder-Only Works for ChatGPT?

- Because ChatGPT is trained to do next-token prediction:
- It reads all previous tokens (the prompt)
- It predicts the next token.
- It repeats this process → generating text
- This is exactly what a decoder does in the original architecture.

How does ChatGPT “understand” without an encoder?

- Self-attention (no mask during training input phase)
- Masked self-attention (for generation)
- Feed-forward networks
- Positional embeddings
- Even though it’s called a decoder, it learns representation of the input as well.
- So ChatGPT’s decoder effectively acts like:
- Understanding module (encoder-like behavior)
- Generation module (decoder behavior)
- All inside one unified stack.

Model	Architecture
BERT	Encoder-only (understands but cannot generate)
GPT / ChatGPT	Decoder-only (understands + generates)
T5 / BART	Encoder + Decoder (full translation-style architecture)

And gemini?

- Gemini is not decoder-only like ChatGPT.
- Google’s Gemini uses an Encoder–Decoder style architecture, more similar to T5 and original Transformers, but heavily upgraded for multimodality.
- Here is the clear breakdown:
- ✅ Gemini Architecture = Encoder + Decoder (Unified Multimodal Transformer)
- Gemini models (1.0, 1.5, 2.0) use a multi-modal encoder and a text decoder.
- 1. Encoder part
- Can process text
- Can process images
- Can process audio
- Can process video
- Can process documents

- Why Gemini Needs an Encoder?
- Because Gemini is multimodal by design.
- If you give it:
- an image,
- a diagram,
- a PDF,
- a video frame,
- or audio,
- the encoder processes and converts these into embeddings.
- A decoder-only model (like GPT) cannot natively encode all these efficiently.
-

- Gemini is better at:
- Understanding videos
- Reading large PDFs (hundreds of pages)
- Processing long context (1M+ tokens)

- Handling images + text together
- Detailed extraction from tables, diagrams, charts
- Because its encoder is designed to fuse different modalities.
- 🔥 2. **ChatGPT (GPT models) are decoder-only**
- This design gives ChatGPT special strengths:

ChatGPT is better at:

- Creative writing
- Mathematical reasoning
- Code generation
- Long chain-of-thought reasoning
- Dialogue quality
- Maintaining conversational context
- Fast generation
- GPT models “think while generating,” which makes them extremely good at deep logical reasoning and detailed step-by-step solution generation.