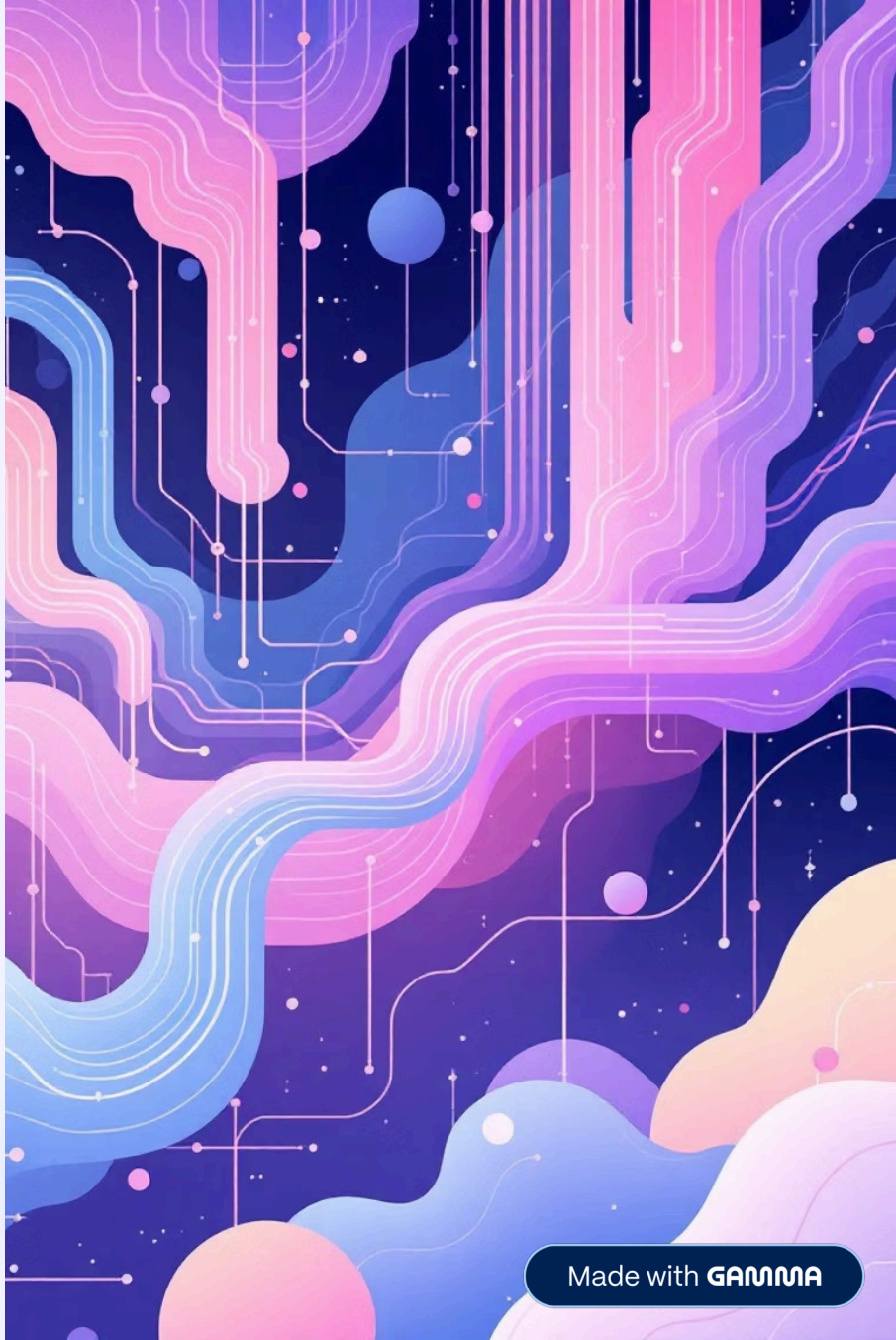
Decoder — Step 4 & Step 5

Completing the Decoder: From Processing to Prediction

Final part of Transformer's Decoder Block



Decoder Recap: Before Step 4

So far, the decoder has accomplished three critical tasks in processing output sequences:

01

Token Embeddings

Converted output tokens into embeddings with positional encodings to capture sequence order

02

Masked Self-Attention

Used Masked Multi-Head Self-Attention to focus on past outputs whilst preventing future token leakage

03

Cross-Attention

Applied Encoder–Decoder Attention to identify and focus on relevant input words from the encoder

Now we advance to the final stages: **Step 4** covers Feed Forward processing with Add & Norm, whilst **Step 5** handles Linear projection and Softmax for word prediction.

Step 4: Feed Forward Network

Each token's representation from the previous attention layers is passed through a **Position-wise Feed Forward Network (FFN)**. Crucially, it processes *each word vector independently* — no information mixing occurs between different word positions.

Formula

$$FFN(x) = ReLU(xW_1 + b_1)W_2 + b_2$$

Where:

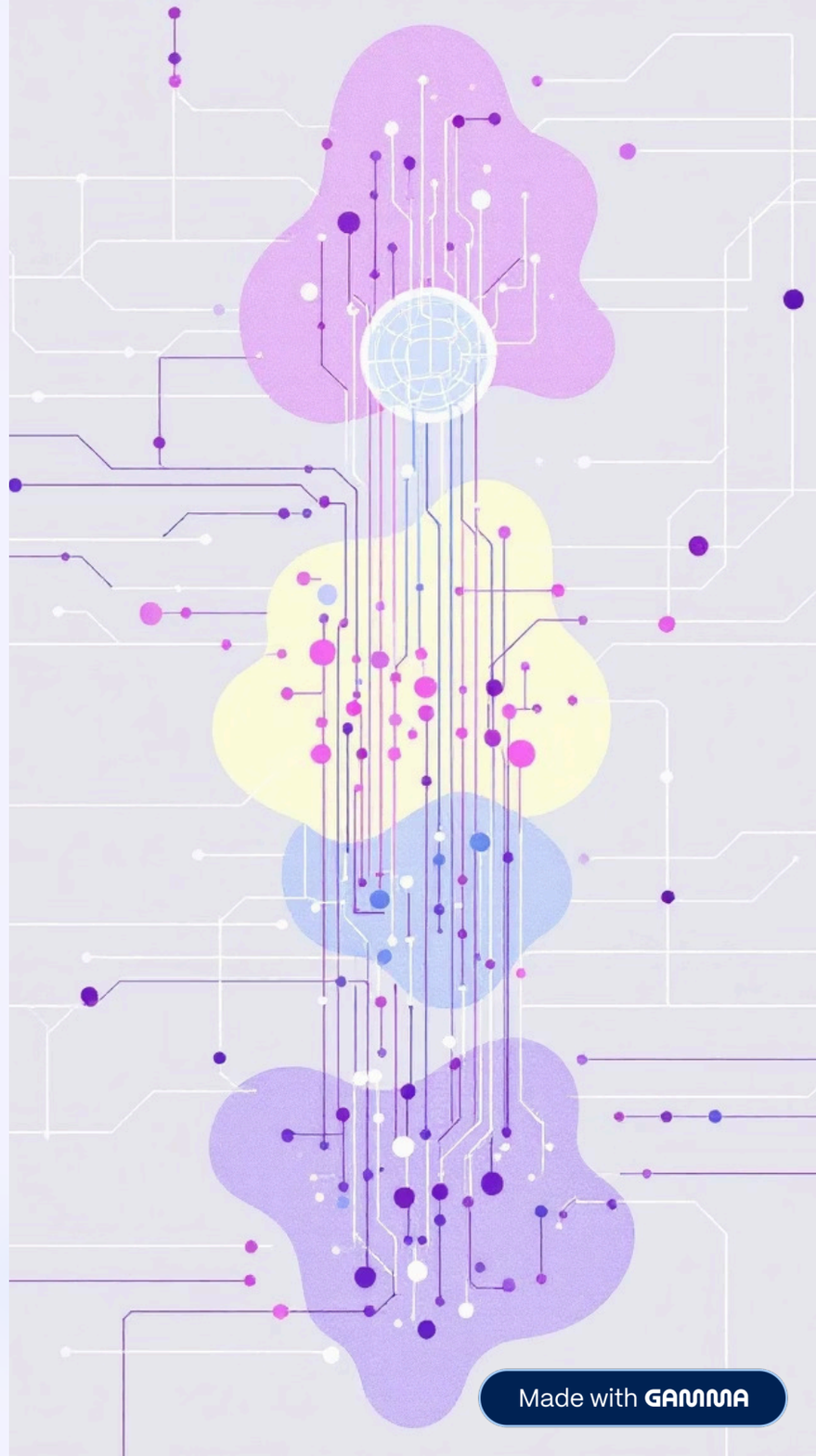
- x: input vector from attention layers
- W_1, W_2: learned weight matrices
- b_1, b_2: bias terms
- ReLU: adds non-linearity

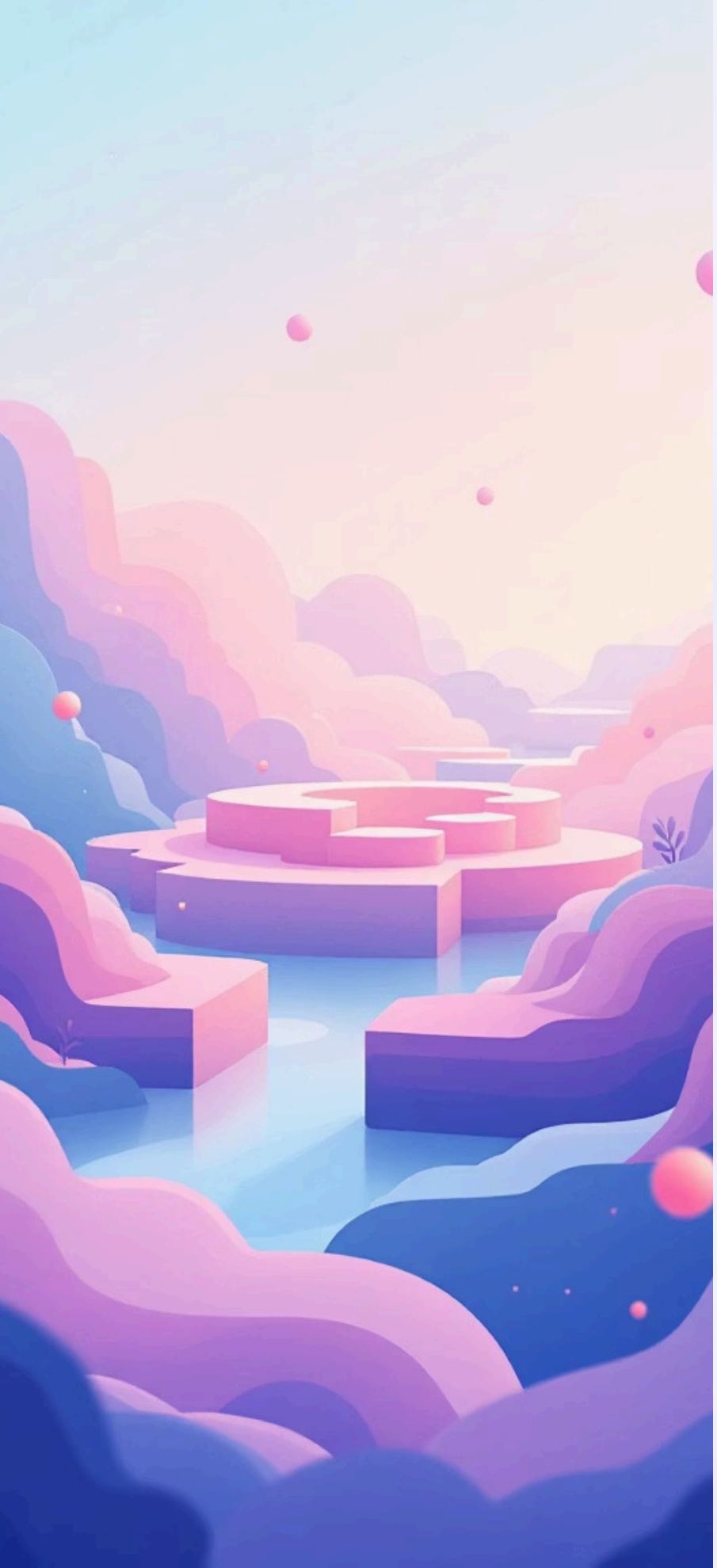
Dimension Transformation

The FFN expands and then compresses the word vector through two linear transformations:

512 → 2048 → 512

This expansion allows the model to learn richer representations before projecting back to the original dimensionality.





Why Feed Forward Networks?

Non-Linear Transformations

Introduces essential non-linearity through activation functions, enabling the model to capture complex patterns beyond linear relationships

Semantic Pattern Learning

Learns deeper semantic patterns and relationships between contextual information gathered from attention layers

Representation Refinement

Refines and enriches the representations already learned by attention mechanisms, adding another layer of understanding

💡 **Think of it like:** "Each word already knows its context from attention — now let's process what that context actually *means* for prediction."

Add & Norm: Stabilising Training

After the Feed Forward Network, we apply two crucial operations to stabilise training and preserve information flow:

Formula

$$Output = LayerNorm(x + SubLayer(x))$$

Process

1. **Add residual connection:** x + FFN(x)
2. **Apply Layer Normalization** to stabilise

Residual Connection

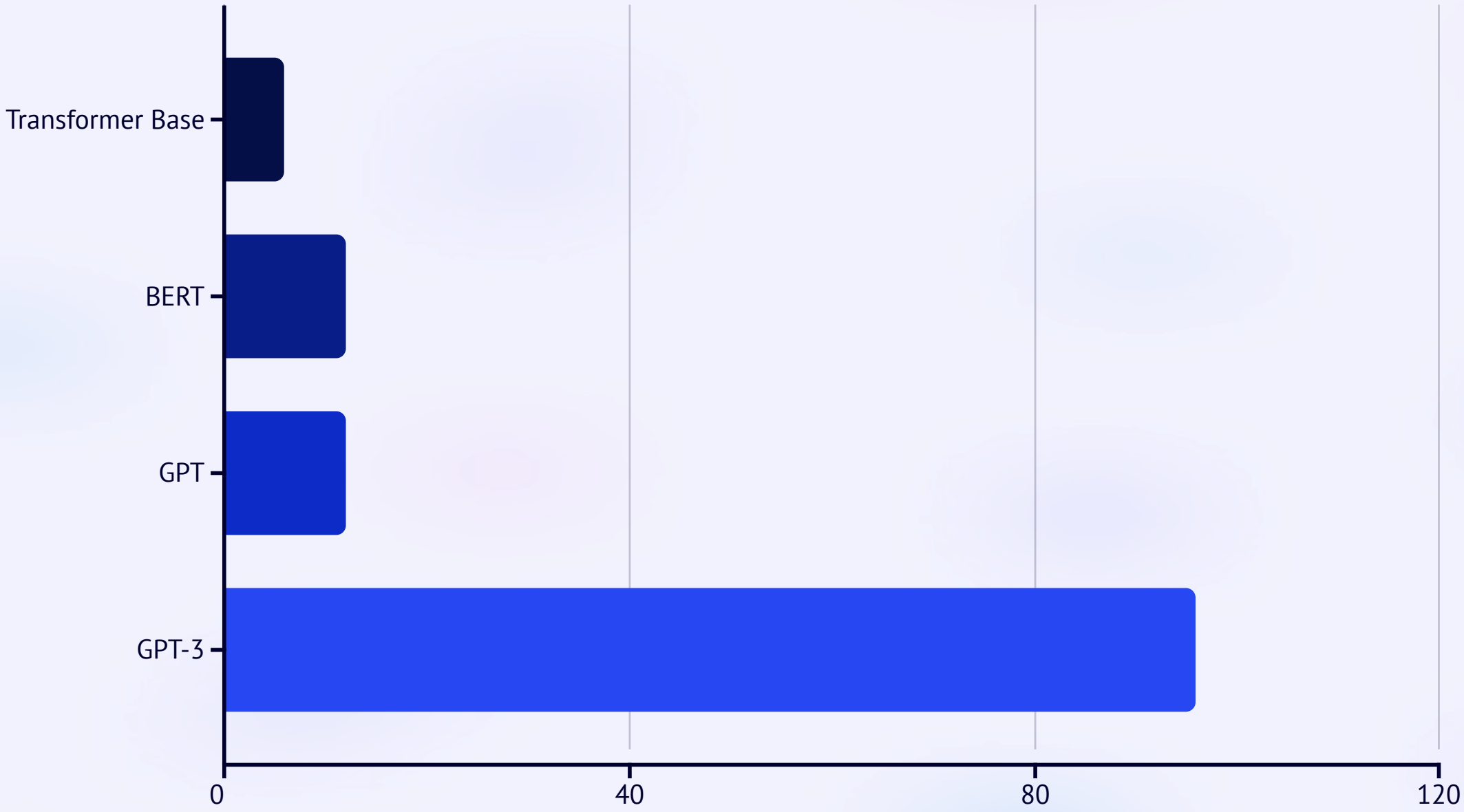
Preserves original information and facilitates gradient flow during backpropagation, preventing vanishing gradients

Layer Normalization

Stabilises training dynamics and accelerates convergence by normalising inputs across features

Repeat N Times: Deep Layer Stacking

In the Transformer Decoder, the complete block — comprising three attention layers, FFN, and Add&Norm — is **repeated N times** to create a deep architecture.



Each successive layer refines understanding, learns higher-level abstractions, and builds upon the output of the previous layer, creating increasingly sophisticated representations.

Step 5: Linear + Softmax

After N decoder layers, we generate the final prediction through two transformation steps:

Linear Projection

Each token's final representation (512-dimensional vector) is projected into vocabulary size (e.g., 50,000 dimensions)

Softmax Activation

Converts raw scores (logits) into probability distribution over entire vocabulary

Softmax Formula

$$P(word) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

Where `z_i` is the logit (raw score) for word *i* in the vocabulary.

Output Generation: Autoregressive Decoding

At each time step, the model employs an autoregressive approach:

- Selects the **highest probability word** using argmax operation
- That predicted word is **fed back** as the next input to the decoder
- Process continues until the model outputs <END> token

Example: English to French Translation

t=1

Input: <START>

Output: "Le"

1

t=2

Input: <START> Le

Output: "chat"

2

t=3

Input: <START> Le chat

Output: "s'est"

3

t=n

Continue until...

Output: <END>

4

Big Picture: Complete Decoder Flow

Output Embedding + Positional Encoding

Initial representation of output tokens with positional information

Masked Multi-Head Self-Attention

Attends to previous positions in output sequence

Encoder–Decoder Attention

Cross-attention to relevant encoder outputs

Feed Forward → Add & Norm

Position-wise transformation with residual connections

Repeat N Times

Stack multiple decoder layers for depth

Linear + Softmax

Generate probability distribution over vocabulary

This entire pipeline runs **for each token** during both training and inference, with masking ensuring causality is maintained.

Summary: Decoder Steps 4 & 5

1

Feed Forward

Function: Deepens understanding

Key Idea: Non-linear per-token processing through two-layer network with ReLU activation

2

Add & Norm

Function: Stabilises training

Key Idea: Residual connection preserves information whilst layer normalisation stabilises gradients

3

Repeat N Times

Function: Deep stacking

Key Idea: Layer-wise refinement builds increasingly abstract representations

4

Linear + Softmax

Function: Output layer

Key Idea: Generates probability distribution for next word prediction

✅ **End of Decoder** — The final probabilities from softmax are used to select the actual output words, completing the Transformer's sequence-to-sequence transformation.