The Transformer: A Mathematical Deep Dive for Language Translation

1. ✅ Tokenization + Embedding

Let's assume these embeddings for simplicity (real models learn these):

Token.	Embedding
"I"	[1.0, 0.0]
"love"	[0.0, 1.0]
"you"	[1.0, 1.0]

$$X = \begin{bmatrix}1.0 & 0.0\\ 0.0 & 1.0\\ 1.0 & 1.0\end{bmatrix}$$

2. ✅ Positional Encoding

For simplicity, skip real sin/cos positional encodings and assume:

Position	Encoding
0	[0.1, 0.0]
1	[0.0, 0.1]
2	[0.1, 0.1]

So after addition: --> X+PE =

$$\begin{bmatrix}1.1 & 0.0\\ 0.0 & 1.1\\ 1.1 & 1.1\end{bmatrix}$$

✅ Encoder: Self-Attention Calculation

Step 1: Initialization of Wq, Wk & Wv

WQ matrix

WQ is defined as [1 0] [0 1].

WV matrix

WV is defined as [1 1] [0 1].

WK matrix

WK is defined as [1 0] [1 1].

Step 2: Compute Q, K, V

$$Q = X*Wq \quad \begin{bmatrix}1.1 & 0.0\\ 0.0 & 1.1\\ 1.1 & 1.1\end{bmatrix} * \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} = \begin{bmatrix}1.1 & 0.0\\ 0.0 & 1.1\\ 1.1 & 1.1\end{bmatrix}$$

$$K = X*Wk = \begin{bmatrix}1.1 & 0.0\\ 0.0 & 1.1\\ 1.1 & 1.1\end{bmatrix} * \begin{bmatrix}1 & 0\\ 1 & 1\end{bmatrix} = \begin{bmatrix}1.1 & 0.0\\ 1.1 & 1.1\\ 2.2 & 1.1\end{bmatrix}$$

$$V = X*Wv = \begin{bmatrix}1.1 & 0.0\\ 0.0 & 1.1\\ 1.1 & 1.1\end{bmatrix} * \begin{bmatrix}1 & 1\\ 0 & 1\end{bmatrix} = \begin{bmatrix}1.1 & 1.1\\ 0.0 & 1.1\\ 1.1 & 2.2\end{bmatrix}$$

Step 3: Calculation of Attention Scores

Compute scores = Q.KT/√2

$$Q.KT = \begin{bmatrix}1.1*1.1+0.0*1.1, & 1.1*0.0+0.0*1.1, & 1.1*1.1+0.0*2.2\\ 0.0*1.1+1.1*1.1, & 0.0*0.0+1.1*1.1, & 0.0*1.1+1.1*2.2\\ 1.1*1.1+1.1*1.1, & 1.1*0.0+1.1*1.1, & 1.1*1.1+1.1*2.2\end{bmatrix}$$

$$= [[1.21,\ 0.00,\ 1.21],\ [1.21,\ 1.21,\ 2.42],\ [2.42,\ 1.21,\ 3.63]]$$

Scaled scores Q.KT/√2 = $[[0.86,\ 0.0,\ 0.86],\ [0.86,\ 0.86,\ 1.71],\ [1.71,\ 0.86,\ 2.57]]$

Step 4: Apply Softmax (row-wise)

Softmax for row 1: softmax([0.86, 0.00, 0.86]) ≈ [0.42, 0.16, 0.42]
Softmax for row 2: softmax([0.86, 0.86, 1.71]) ≈ [0.27, 0.27, 0.46]
Softmax for row 3: softmax([1.71, 0.86, 2.57]) ≈ [0.25, 0.11, 0.64]

Step 5: Multiply with V

Attention output = softmax scores × V

0.42 * 1.1 + 0.16 * 1.1 + 0.42 * 2.2, 0.42 * 0.0 + 0.16 * 1.1 + 0.42 * 1.1
0.27 * 1.1 + 0.27 * 1.1 + 0.46 * 2.2, 0.27 * 0.0 + 0.27 * 1.1 + 0.46 * 1.1
0.25 * 1.1 + 0.11 * 1.1 + 0.64 * 2.2, 0.25 * 0.0 + 0.11 * 1.1 + 0.64 * 1.1

$$= \begin{bmatrix}1.563, & 0.635\\ 1.606, & 0.803\\ 1.804, & 0.825\end{bmatrix}$$

3. ✅ Encoder Feedforward Network (FFN)

attention outputs from the encoder are: $\begin{bmatrix}1.563, & 0.635\\ 1.606, & 0.803\\ 1.804, & 0.825\end{bmatrix}$

Each attention output (per token) is passed through a position-wise feedforward network:

FFN(x)=max(0, xW1 + b1) W2+b 2

Let's define:

d_model = 2
d_ff = 4

$$W1 = \begin{bmatrix}1, & 0, & 1, & 0\\ 0, & 1, & 0, & 1\end{bmatrix} \qquad W2 = \begin{bmatrix}1, & 0\\ 0, & 1\\ 1, & 0\\ 0, & 1\end{bmatrix}$$

For Simplicity lets consider bias =0:

b1 = [0, 0, 0, 0]
b2 = [0, 0]

For Token 1: ?? = [1.563, 0.635]

Step 1: Linear Layer 1 xW1 =[1.563*1 + 0.635*0, 1.563*0+0.635*1, 1.563*1+0.635*0, 1.563*0+0.635*1] = [1.563, 0.635, 1.563, 0.635]

Step 2: ReLU Activation ReLU(xW1) = [1.563, 0.635, 1.563, 0.635]

Step 3: Linear Layer 2 xW2 =[1.563 + 1.563, 0.635 + 0.635] =[3.126, 1.270]

For Token 2: ?? = [1.606, 0.803]

Step 1: Linear Layer 1 xW1 = [1.606, 0.803, 1.606, 0.803]

Step 2: ReLU Activation ReLU(xW1) = [1.606, 0.803, 1.606, 0.803]

Step 3: Linear Layer 2 xW2 = [1.606+ 1.606, 0.803+ 0.803] =[3.212, 1.606]

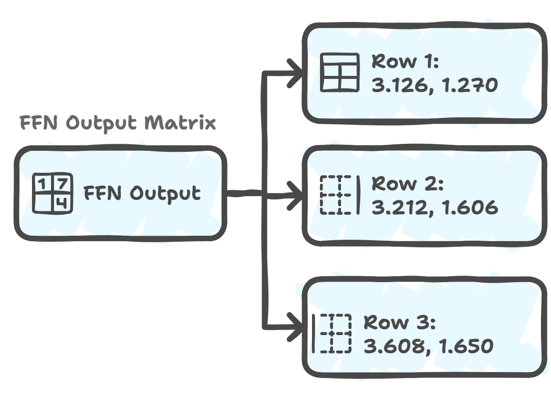
For Token 3: ?? = [1.804, 0.825]

Step 1: Linear Layer 1 xW1 = [1.804, 0.825, 1.804, 0.825]

Step 2: ReLU Activation ReLU(xW1) = [1.804, 0.825, 1.804, 0.825]

Step 3: Linear Layer 2 xW2 = [1.804 + 1.804, 0.825 + 0.825] =[3.608, 1.650]

FFN Output Matrix



4. ✅ Add & Norm (Residual Connection)

We need to add FFN output to the attention output and apply layer normalization.

Residual Sum = Attention Output + FFN Output

$$\begin{bmatrix}1.563, & 0.635\\ 1.606, & 0.803\\ 1.804, & 0.825\end{bmatrix} + \begin{bmatrix}3.126, & 1.270\\ 3.212, & 1.606\\ 3.608, & 1.650\end{bmatrix} = \begin{bmatrix}4.689, & 1.905\\ 4.818, & 2.409\\ 5.412, & 2.475\end{bmatrix}$$

5. ✅ Batch Normalisation

LayerNorm is applied per token (per row), across the features (axis=-1).
For token vector ?? = [??1, ??2]

$$\text{Mean}(\mu) = (x1 + x1) / 2$$

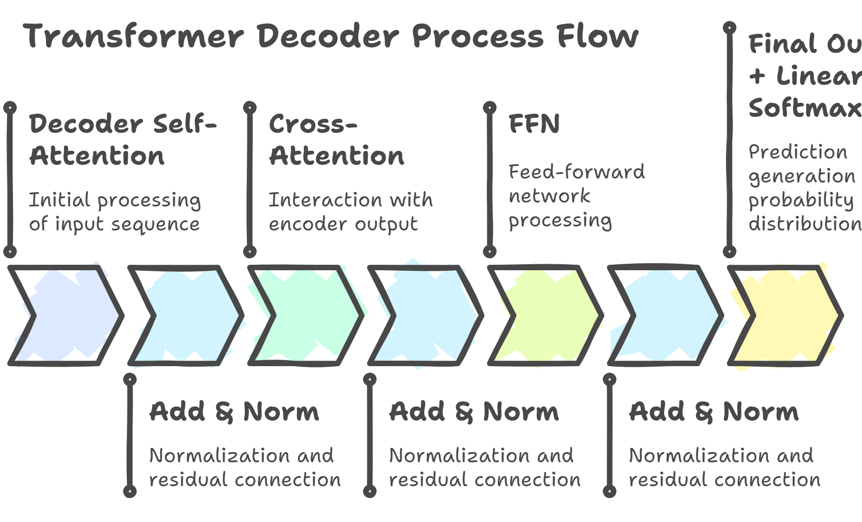
x1

$$\text{Variance}(\sigma) = \sqrt{\frac{(x1 - \mu)2 + (x1 - \mu)2}{2} + \varepsilon}$$

$$\text{LayerNorm}(x) = (x - \mu) / \sigma$$

✅ Decoder: Step-by-step cal

Transformer Decoder Process Flow



✅ 1. Decoder Self-Attention (Masked)

Step 1: Initialization of Wq, Wk & Wv

WQ matrix

WQ is defined as [1 0] [0 1].

WV matrix

WV is defined as [1.1 0] [0 1.1].

WK matrix

WK is defined as [1 0] [1 1].

Let's say the decoder has so far generated:

<s> (start token)
We'll embed this as: x_decoder = [0.5,0.5]

$$Q = X * Wq = [0.5,\ 0.5] * \begin{bmatrix}1, & 0\\ 0, & 1\end{bmatrix} = [0.5,\ 0.5]$$

$$K = X * Wk = [0.5,\ 0.5] * \begin{bmatrix}1, & 0\\ 1, & 1\end{bmatrix} = [1.0,\ 0.5]$$

$$V = X * Wv = [0.5,\ 0.5] * \begin{bmatrix}1.1, & 0\\ 0, & 1.1\end{bmatrix} = [0.55,\ 0.55]$$

Attention scores

$$Q.KT = [0.5,\ 0.5] * \begin{bmatrix}1.0\\ 0.5\end{bmatrix} = 0.75$$

Scaled scores

$$Q.KT/\sqrt{2} = 0.75/1.414 = 0.53$$

Applying Softmax :

only 1 token, so it becomes 1.0

Attention Output :

1.0 * [0.55, 0.55] = [0.55, 0.55]

✅ 2. Add & Norm

Residual connection:

x + attention output = [0.5 + 0.55, 0.5 + 0.55]

Output of Decoder Self-Attention= [1.05, 1.05]

✅ 3. Cross-Attention with Encoder Output

We now attend to encoder outputs:

Decoder Q(??_??) = [1.05, 1.05]

Step 1: Attention scores

Score 1: = 1.05 * 4.689 + 1.05 * 1.905 = 4.92345 + 2.00025 = 6.9237
Score 2: = 1.05 * 4.818 + 1.05 * 2.409 = 5.0589 + 2.52945 = 7.58835
Score 3: = 1.05 * 5.412 + 1.05 * 2.475 = 5.6826 + 2.59875 = 8.28135

Step 2: Scale

[6.9237/1.414, 7.58835/1.414, 8.28135/1.414]
≈ [4.895, 5.363, 5.854]

Step 3: Softmax

[0.1923, 0.3064, 0.5013]

Step 4: Weighted Sum of V_enc

$$[0.1923,\ 0.3064,\ 0.5013] * \begin{bmatrix}4.689, & 1.905\\ 4.818, & 2.409\\ 5.412, & 2.475\end{bmatrix}$$

= [0.9018 + 1.4765 + 2.7135, 0.3667 +0.7383 + 1.2408] = [5.0918, 2.3458]

4. ✅ Decoder Feedforward Network (FFN)

The FFN is applied independently per token: FFN(x)=max(0, xW1 + b1) W2+b2

Let's define:

Input dim = 2
Hidden dim = 4
Output dim = 2

$$W1 = \begin{bmatrix}0.2, & 0.3, & 0.5, & -0.4\\ 0.6, & -0.1, & 0.2, & 0.3\end{bmatrix} \qquad W2 = \begin{bmatrix}0.4, & -0.2\\ 0.3, & 0.5\\ -0.1, & 0.2\\ 0.6, & 0.3\end{bmatrix}$$

b1 = [0.1, 0.1, 0.1, 0.1]
b2 = [0.05, -0.05]

Step 1: First Linear Layer (x * W1 + b1)

$$z1 = [5.0918,\ 2.3458] * \begin{bmatrix}0.2, & 0.3, & 0.5, & -0.4\\ 0.6, & -0.1, & 0.2, & 0.3\end{bmatrix} = \begin{bmatrix}5.0918*0.2, & 2.3458*0.6\\ 5.0918*0.3, & 2.3458*(-0.1)\\ 5.0918*0.5, & 2.3458*0.2\\ 5.0918*(-0.4), & 2.3458*0.3\end{bmatrix}$$

= [2.42584, 1.29296, 3.01506, -1.33298]

Now z1 + b1 = [2.42584, 1.29296, 3.01506, -1.33298] + [0.1, 0.1, 0.1, 0.1]
= [2.52584, 1.39296, 3.11506, -1.23298]

Step 2: ReLU Activation

Relu(z1 + b1)= [2.52584, 1.39296, 3.11506, 0.0]

Step 3: Second Linear Layer (a1 * W2 + b2)

$$z2 = [2.52584,\ 1.39296,\ 3.11506,\ 0.0] * \begin{bmatrix}0.4, & -0.2\\ 0.3, & 0.5\\ -0.1, & 0.2\\ 0.6, & 0.3\end{bmatrix}$$

= [1.11672, 0.81432]

Now z2 + b2 = [1.11672, 0.81432] + [0.05, -0.05] = [1.16672, 0.76432]

5. ✅ Final Output + Linear → Softmax

Assume vocab size is 3 (just for demo), and projection matrix:

$$W\_vocab = \begin{bmatrix}0.8, & 0.2\\ 0.1, & 0.5\\ -0.2, & 0.4\end{bmatrix} \begin{matrix}\rightarrow \text{Token "Le"}\\ \rightarrow \text{Token "chat"}\\ \rightarrow \text{Token "mange"}\end{matrix}$$

$$\text{logits} = W\_vocab * [1.16672,\ 0.76432]^T = \begin{bmatrix}1.16672*0.8 + 0.76432*0.2\\ 1.16672*0.1 + 0.76432*0.5\\ 1.16672*(-0.2) + 0.76432*0.4\end{bmatrix} = \begin{bmatrix}1.08623\\ 0.49883\\ 0.07238\end{bmatrix}$$

Now apply softmax:

exp(1.08623) ≈ 2.962
exp(0.49883) ≈ 1.647
exp(0.07238) ≈ 1.075

Probabilities:

"Le" → 2.962 / 5.684 ≈ 0.521
"chat" → 1.647 / 5.684 ≈ 0.290
"mange" → 1.075 / 5.684 ≈ 0.189

?? Predicted Token: "Le" with 52.1% confidence ✅

✅ 1. Decoding Next Token

Step 1: Initialization of Wq, Wk & Wv

WQ is defined as [1 0] [0 1].

WK is defined as [1 0] [1 1].

WV is defined as [1.1 0] [0 1.1].

Let's say the decoder has so far generated: ["<s>", "Le"]

$$\text{We'll embed this as: } X\_decoder = \begin{bmatrix}0.5, & 0.5\\ 0.6, & 0.4\end{bmatrix}$$

$$Q = X * Wq = \begin{bmatrix}0.5, & 0.5\\ 0.6, & 0.4\end{bmatrix} * \begin{bmatrix}1, & 0\\ 0, & 1\end{bmatrix} = \begin{bmatrix}0.5, & 0.5\\ 0.6, & 0.4\end{bmatrix}$$

$$V = X * Wv = \begin{bmatrix}0.5, & 0.5\\ 0.6, & 0.4\end{bmatrix} * \begin{bmatrix}1.1, & 0\\ 0, & 1.1\end{bmatrix} = \begin{bmatrix}0.55, & 0.55\\ 0.66, & 0.44\end{bmatrix}$$

$$K = X * Wk = \begin{bmatrix}0.5, & 0.5\\ 0.6, & 0.4\end{bmatrix} * \begin{bmatrix}1, & 0\\ 1, & 1\end{bmatrix} = \begin{bmatrix}1.0, & 0.5\\ 1.0, & 0.4\end{bmatrix}$$

Attention scores (Q.KT)

We'll compute attention for token2 attending over K0 and K1

Dot products of Q2 with each Ki:

$Q_2 \cdot K_0 = 0.6*1.0 + 0.4*0.5 = 0.6 + 0.2 = 0.8$

$Q_2 \cdot K_1 = 0.6*1.0 + 0.4*0.4 = 0.6 + 0.16 = 0.76$

Scaled scores (Q.KT/√2)

Scaled scores = [0.8 / 1.414, 0.76 / 1.414]
≈ [0.5656, 0.5373]

Applying Softmax :

exp(0.5656) ≈ 1.760 & exp(0.5373) ≈ 1.711
Sum = 1.760 + 1.711 = 3.471
Attention weights = [1.760/3.471, 1.711/3.471]
= [0.507, 0.493]

Attention Output :

$$\text{Softmax}(Q.KT/\sqrt{2}) * V = [0.507,\ 0.493] * \begin{bmatrix}0.55, & 0.55\\ 0.66, & 0.44\end{bmatrix}$$

= [0.60423, 0.49577]

✅ 2. Add & Norm

Residual connection: x + attention output

[0.6 + 0.604, 0.4 + 0.495]

Output of Decoder Self-Attention = [1.204, 0.895]

✅ 3. Cross-Attention with Encoder Output

We now attend to encoder outputs:

Decoder Q(Q_d) =[1.204, 0.895]

Step 1: Attention scores

Score 1: = 1.204 * 4.689 + 0.895 * 1.905 = 5.642 + 1.707 = 7.349
Score 2: = 1.204 * 4.818 + 0.895 * 2.409 = 5.798 + 2.158 = 7.956
Score 3: = 1.204 * 5.412 + 0.895 * 2.475 = 6.513 + 2.217 = 8.73

Step 2: Scale

[7.349/1.414, 7.956/1.414, 8.73/1.414]
≈ [4.895, 5.363, 5.854]

Step 3: Softmax

[0.147, 0.269, 0.584]

Step 4: Weighted Sum of V_enc

$$[0.147,\ 0.269,\ 0.584] * \begin{bmatrix}4.689, & 1.905\\ 4.818, & 2.409\\ 5.412, & 2.475\end{bmatrix}$$

= [2.738 + 1.296 + 3.160, 0.28 +0.6480 + 1.4454]
= [7.194, 2.3734]

4. ✅ Decoder Feedforward Network (FFN)

The FFN is applied independently per token: FFN(x)=max(0, xW1 + b1) W2+b2

$$W1 = \begin{bmatrix}0.2, & 0.3, & 0.5, & -0.4\\ 0.6, & -0.1, & 0.2, & 0.3\end{bmatrix} \qquad W2 = \begin{bmatrix}0.4, & -0.2\\ 0.3, & 0.5\\ -0.1, & 0.2\\ 0.6, & 0.3\end{bmatrix}$$

Step 1: First Linear Layer (x * W1 + b1)

$$z1 = [7.194,\ 2.3734] * \begin{bmatrix}0.2, & 0.3, & 0.5, & -0.4\\ 0.6, & -0.1, & 0.2, & 0.3\end{bmatrix} = [2.42584,\ 1.29296,\ 4.07168,\ -2.16558]$$

Now z1 + b1 = [2.42584, 1.29296, 4.07168, -2.16558] + [0.1, 0.1, 0.1, 0.1]
= [2.52584, 1.39296, 4.08168, -2.06558]

Step 2: ReLU Activation

Relu(z1 + b1)= [2.52584, 1.39296, 4.08168, 0.0]

Step 3: Second Linear Layer (a1 * W2 + b2)

$$z2 = [2.52584,\ 1.39296,\ 4.08168,\ 0.0] * \begin{bmatrix}0.4, & -0.2\\ 0.3, & 0.5\\ -0.1, & 0.2\\ 0.6, & 0.3\end{bmatrix} = [1.02006,\ 1.00765]$$

Now z2 + b2 = [1.02006, 1.00765] + [0.05, -0.05] = [1.07006, 0.95765]

5. ✅ Final Output + Linear → Softmax

Assume vocab size is 3 (just for demo), and projection matrix:

$$W\_vocab = \begin{bmatrix}0.8, & 0.2\\ 0.1, & 0.5\\ -0.2, & 0.4\end{bmatrix} \begin{matrix}\rightarrow \text{Token "Le"}\\ \rightarrow \text{Token "chat"}\\ \rightarrow \text{Token "mange"}\end{matrix}$$

$$\text{logits} = W\_vocab * [1.07006,\ 0.95765]^T = \begin{bmatrix}1.07006*0.8 + 0.95765*0.2\\ 1.07006*0.1 + 0.95765*0.5\\ 1.07006*(-0.2) + 0.95765*0.4\end{bmatrix} = \begin{bmatrix}1.047578,\\ 0.585831,\\ 0.169048\end{bmatrix}$$

Now apply softmax and getting probabilities:

"Le" → ≈ 0.489
"chat" → ≈ 0.308
"mange" → ≈ 0.203

?? Predicted Token: "Le" with 48.9% confidence ✅