Deep Learning

15 Self-Attention & Transformers - II

Dr. Konda Reddy Mopuri
Dept. of AI, IIT Hyderabad
Jan-May 2025

Computational Complexity

1. The attention layer has N i/p tokens and N o/p tokens, each of D dimensions

Computational Complexity

1. The attention layer has N i/p tokens and N o/p tokens, each of D dimensions
2. The overall dimension of i/p (or, o/p) is $N.D$

Computational Complexity

1. The attention layer has N i/p tokens and N o/p tokens, each of D dimensions
2. The overall dimension of i/p (or, o/p) is $N.D$
3. If we used a single fully-connected layer

Computational Complexity

1. The attention layer has N i/p tokens and N o/p tokens, each of D dimensions
2. The overall dimension of i/p (or, o/p) is $N.D$
3. If we used a single fully-connected layer
   - Will have $\mathcal{O}(N^2.D^2)$ independent parameters

Computational Complexity

1. The attention layer has N i/p tokens and N o/p tokens, each of D dimensions
2. The overall dimension of i/p (or, o/p) is $N.D$
3. If we used a single fully-connected layer
   - Will have $\mathcal{O}(N^2.D^2)$ independent parameters
   - Computational cost for one forward pass: $\mathcal{O}(N^2.D^2)$

Computational Complexity

1. In an attention layer, $W^{(Q)}, W^{(K)}, W^{(V)}$ are shared across all the i/p tokens

Computational Complexity

1. In an attention layer, $W^{(Q)}, W^{(K)}, W^{(V)}$ are shared across all the i/p tokens
2. $\rightarrow$ No. of in independent parameters is $\mathcal{O}(D^2)$ $(D_v \approx D_k \approx D)$

Computational Complexity

1. In an attention layer, $W^{(Q)}, W^{(K)}, W^{(V)}$ are shared across all the i/p tokens
2. $\rightarrow$ No. of in independent parameters is $\mathcal{O}(D^2)$ $(D_v \approx D_k \approx D)$
3. For the N i/p tokens

Computational Complexity

1. In an attention layer, $W^{(Q)}, W^{(K)}, W^{(V)}$ are shared across all the i/p tokens
2. $\rightarrow$ No. of in independent parameters is $\mathcal{O}(D^2)$ $(D_v \approx D_k \approx D)$
3. For the N i/p tokens
   - No. of computations required for computing the dot products in self-attention layer $\mathcal{O}(N^2.D)$

Computational Complexity

1. In an attention layer, $W^{(Q)}, W^{(K)}, W^{(V)}$ are shared across all the i/p tokens
2. $\rightarrow$ No. of in independent parameters is $\mathcal{O}(D^2)$ $(D_v \approx D_k \approx D)$
3. For the N i/p tokens
   - No. of computations required for computing the dot products in self-attention layer $\mathcal{O}(N^2.D)$
4. Subsequent Neural Network layer has D inputs and D outputs $\rightarrow$ parameter $= \mathcal{O}(D^2)$ and computational cost of $\mathcal{O}(N.D^2)$

Positional Encoding

1. The weights $(W^{(Q)}, W^{(K)}, W^{(V)})$ are shared across the i/p tokens $\rightarrow$ Transformer is permutation invariant

Positional Encoding

1. The weights $(W^{(Q)}, W^{(K)}, W^{(V)})$ are shared across the i/p tokens $\rightarrow$ Transformer is permutation invariant
2. Strong limitation to processing the sequential data

Positional Encoding

1. The weights $(W^{(Q)}, W^{(K)}, W^{(V)})$ are shared across the i/p tokens $\rightarrow$ Transformer is permutation invariant
2. Strong limitation to processing the sequential data
3. CSK plays better, not MI vs. MI plays better, not CSK

Positional Encoding

1. The weights $(W^{(Q)}, W^{(K)}, W^{(V)})$ are shared across the i/p tokens $\rightarrow$ Transformer is permutation invariant
2. Strong limitation to processing the sequential data
3. CSK plays better, not MI vs. MI plays better, not CSK
4. We need a way to inject the order information

Positional Encoding

1. Without disturbing the design of the transformer, we may encode the position information (r_n) along with the data (tokens) (x_n)

Positional Encoding

1. Without disturbing the design of the transformer, we may encode the position information (r_n) along with the data (tokens) (x_n)
2. Obvious way is to concatenate

Positional Encoding

1. Without disturbing the design of the transformer, we may encode the position information (r_n) along with the data (tokens) (x_n)
2. Obvious way is to concatenate
   - Increased dimensions and computations

Positional Encoding

1. Without disturbing the design of the transformer, we may encode the position information (r_n) along with the data (tokens) (x_n)
2. Obvious way is to concatenate
   - Increased dimensions and computations
3. Instead, add them $\tilde{x_n} = x_n + r_n$

Positional Encoding

1. Would it not corrupt the data?

Positional Encoding

1. Would it not corrupt the data?
   - High dimensionality keeps them separate

Positional Encoding

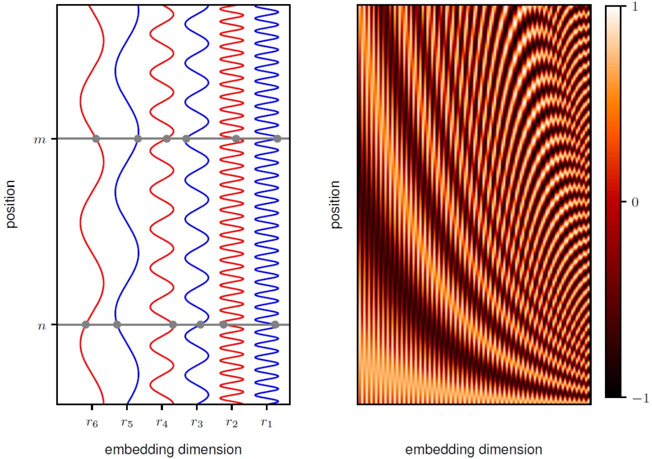
1. Would it not corrupt the data?
    - High dimensionality keeps them separate
    - Skip connections retain the r_n across the layers

Positional Encoding

$$r_{ni} = \begin{cases} \sin\left(\frac{n}{L^{i/D}}\right), & \text{if } i \text{ is even,} \\ \cos\left(\frac{n}{L^{(i-1)/D}}\right), & \text{if } i \text{ is odd.} \end{cases}$$

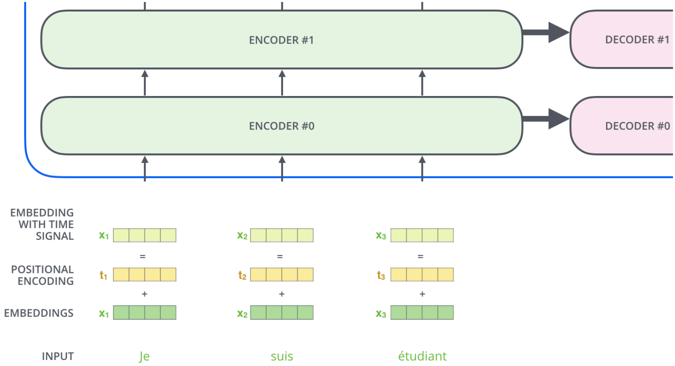
The Bishop's book

Positional Encoding



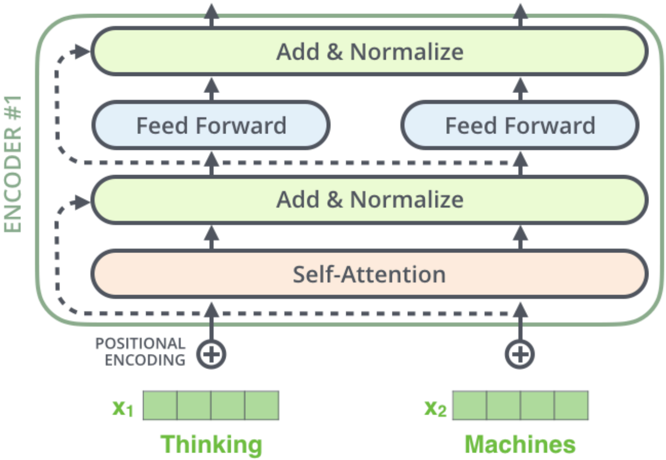
The Bishop's book

Positional Encoding



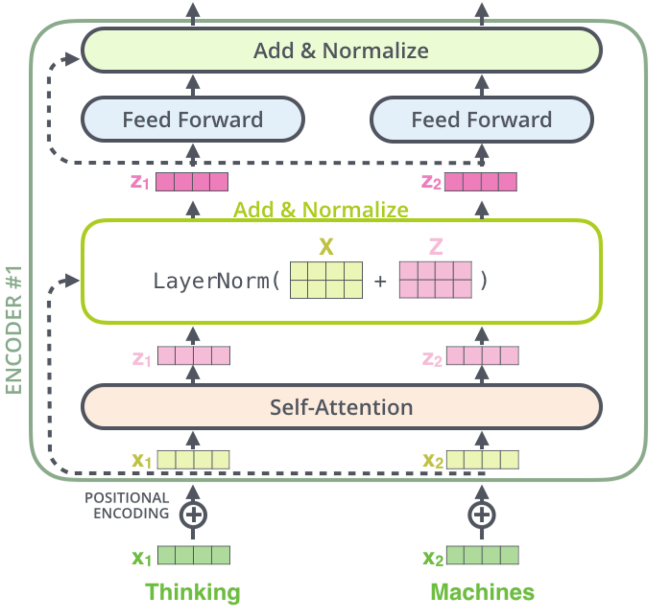
Credits: Jay Alammar

Residuals in the Encoder

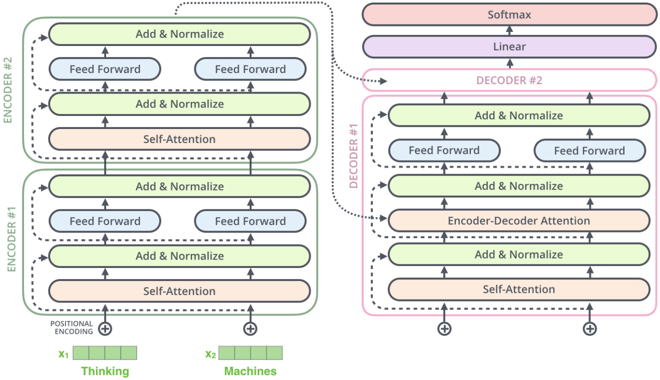


Credits: Jay Alammar

Residuals in the Encoder



Tranformer-Decoder



Credits: Jay Alammar

Transformer-Decoder

1. Self-attention here works in a slightly different way $\rightarrow$ masks the future positions

Transformer-Decoder

1. Self-attention here works in a slightly different way $\rightarrow$ masks the future positions
2. Uses the top encoder's K and V vectors for its' encoder-decoder (cross) attention

Transformer-Decoder

1. Self-attention here works in a slightly different way $\rightarrow$ masks the future positions
2. Uses the top encoder's K and V vectors for its' encoder-decoder (cross) attention
3. Encoder-decoder attention layer borrows the queries from the layer below it