CPEN 455: Deep Learning

Lecture 8: Transformers

Renjie Liao

University of British Columbia

Winter, Term 1, 2026

Outline

- Applications and Challenges of Sequence Modeling
- Transformers
  - Input Embedding & Sinusoidal Positional Encoding
  - Encoder
    - Multi-Head Self-Attention, Residual Connections & LayerNorm
  - Decoder
    - Masked Self-Attention, Cross-Attention
- Limitations & Variants
  - Efficient Attention (FlashAttention, Sparse & Linear)
  - Rotary Positional Embedding (RoPE)
  - Normalization (LayerNorm/RMSNorm, Pre-/Post-Norm)
  - Gated MLPs (SwiGLU)
  - Vision Transformers (ViT & Swin)

Outline

- **Applications and Challenges of Sequence Modeling**
- Transformers
  - Input Embedding & Sinusoidal Positional Encoding
  - Encoder
    - Multi-Head Self-Attention, Residual Connections & LayerNorm
  - Decoder
    - Masked Self-Attention, Cross-Attention
- Limitations & Variants
  - Efficient Attention (FlashAttention, Sparse & Linear)
  - Rotary Positional Embedding (RoPE)
  - Normalization (LayerNorm/RMSNorm, Pre-/Post-Norm)
  - Gated MLPs (SwiGLU)
  - Vision Transformers (ViT & Swin)

Deep Learning for Sequences

- Language Models

$$P(\boldsymbol{x}^{(t+1)} | \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})$$

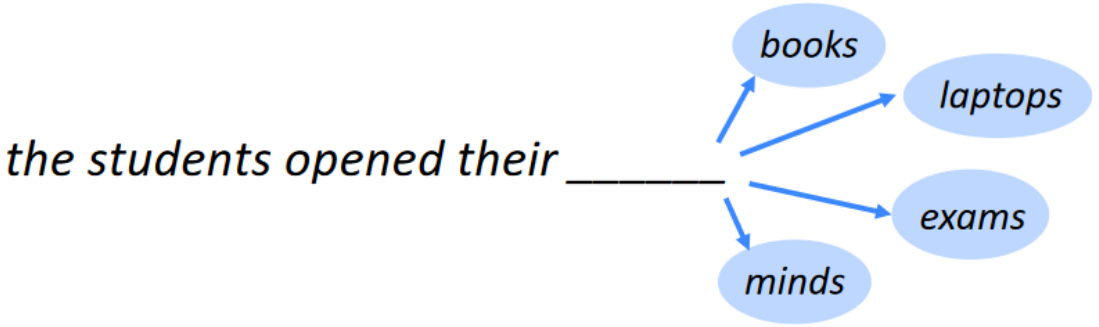
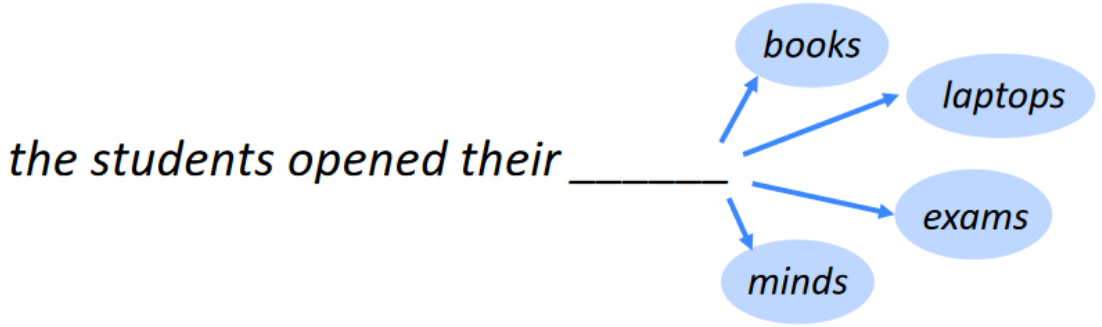


Image Credit: [1]

Deep Learning for Sequences

- Language Models

$$P(\boldsymbol{x}^{(t+1)} | \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})$$



- Machine Translation



Image Credit: [1, 2]

Deep Learning for Sequences

Key Challenges:

- Varying-sized input sequences

Deep Learning for Sequences

Key Challenges:

- Varying-sized input sequences
- Orders “may” be crucial for cognition

Aoccdrnig to a rscheearch at Cmabrigde Uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are, the olny iprmoetnt tihng is taht the frist and lsat ltteer be at the rghit pclae. The rset can be a toatl mses and you can sitll raed it wouthit porbelm. Tihs is bcuseae the huamn mnid deos not raed ervey lteter by istlef, but the wrod as a wlohe.

Image Credit: [3]

Deep Learning for Sequences

Key Challenges:

- Varying-sized input sequences
- Orders “may” be crucial for cognition

  Aoccdrnig to a rscheearch at Cmabrigde Uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are, the olny iprmoetnt tihng is taht the frist and lsat ltteer be at the rghit pclae. The rset can be a toatl mses and you can sitll raed it wouthit porbelm. Tihs is bcuseae the huamn mnid deos not raed ervey lteter by istlef, but the wrod as a wlohe.

- Complex statistical dependencies (e.g. long-range ones)



Image Credit: [3, 4]

Deep Learning for Sequences

Key Challenges:

- Varying-sized input sequences
- Orders “may” be crucial for cognition

  Aoccdrnig to a rscheearch at Cmabrigde Uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are, the olny iprmoetnt tihng is taht the frist and lsat ltteer be at the rghit pclae. The rset can be a toatl mses and you can sitll raed it wouthit porbelm. Tihs is bcuseae the huamn mnid deos not raed ervey lteter by istlef, but the wrod as a wlohe.

- Complex statistical dependencies (e.g. long-range ones)



Transformer [5]

Image Credit: [3, 4]

Outline

- Applications and Challenges of Sequence Modeling
- **Transformers**
  - Input Embedding & Sinusoidal Positional Encoding
  - Encoder
    - Multi-Head Self-Attention, Residual Connections & LayerNorm
  - Decoder
    - Masked Self-Attention, Cross-Attention
- Limitations & Variants
  - Efficient Attention (FlashAttention, Sparse & Linear)
  - Rotary Positional Embedding (RoPE)
  - Normalization (LayerNorm/RMSNorm, Pre-/Post-Norm)
  - Gated MLPs (SwiGLU)
  - Vision Transformers (ViT & Swin)

Transformers

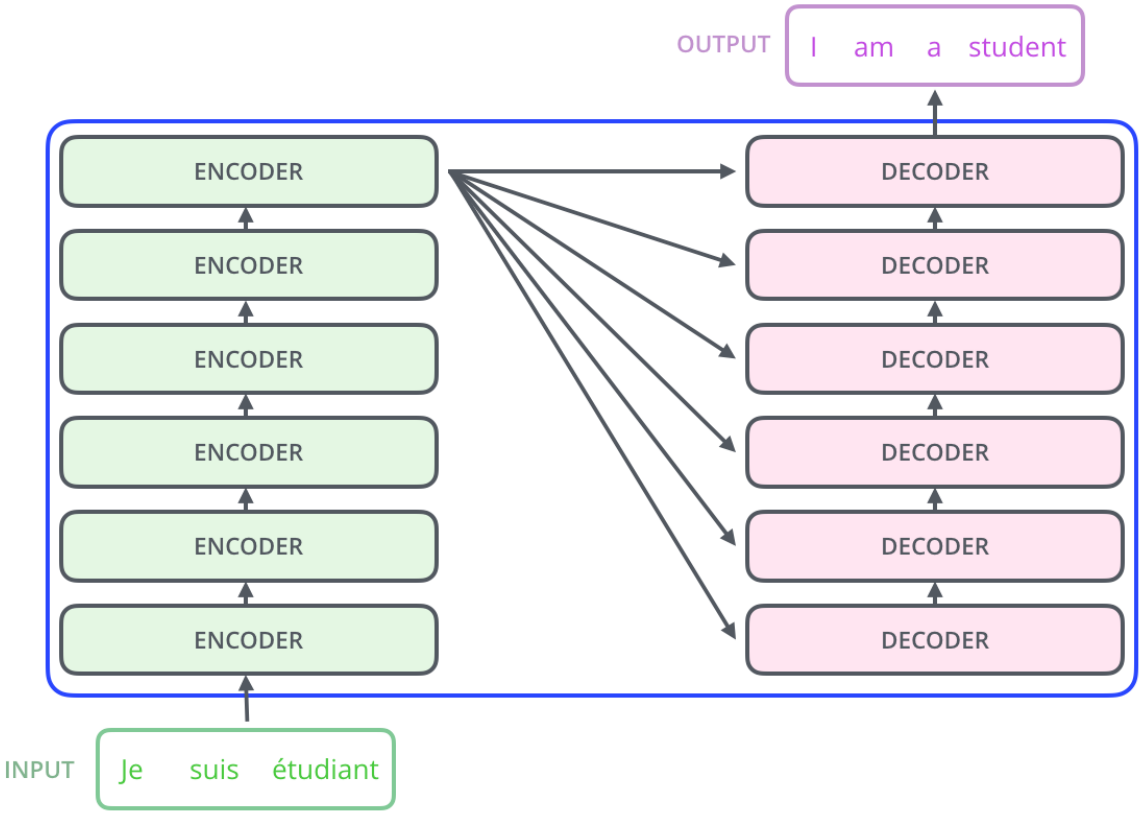
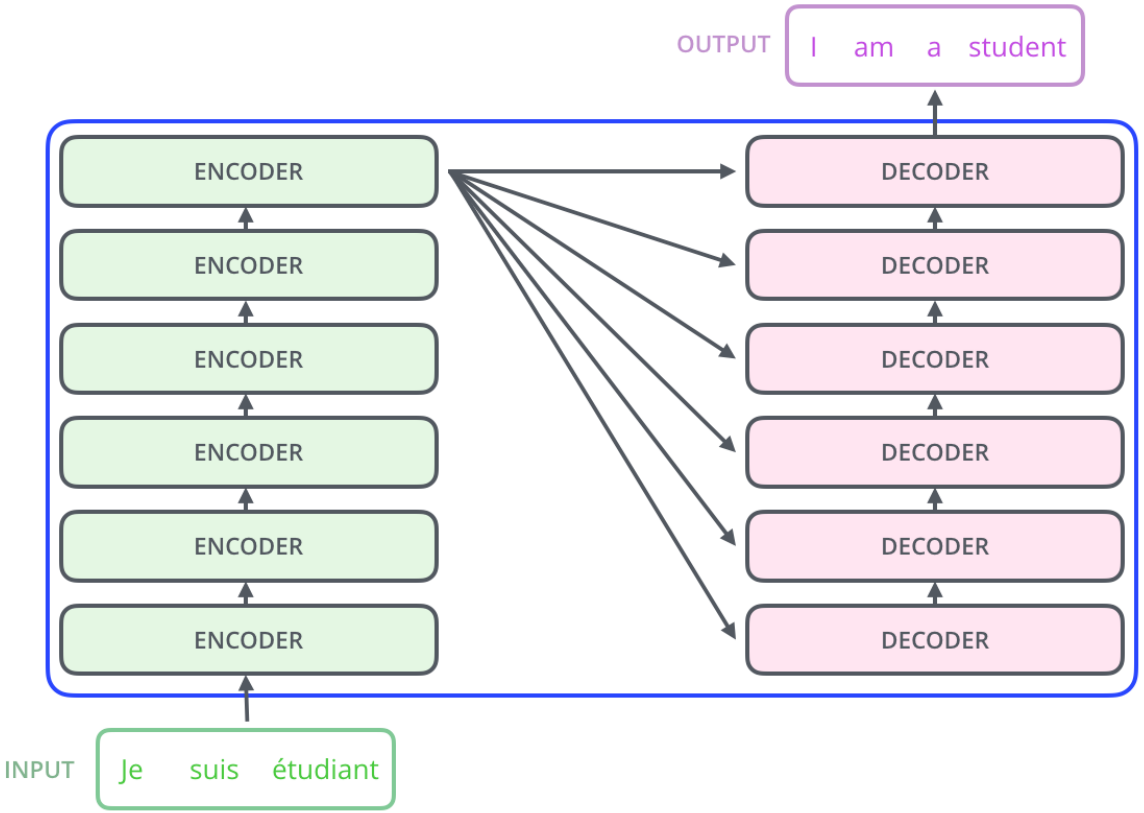


Image Credit: [2]

Transformers



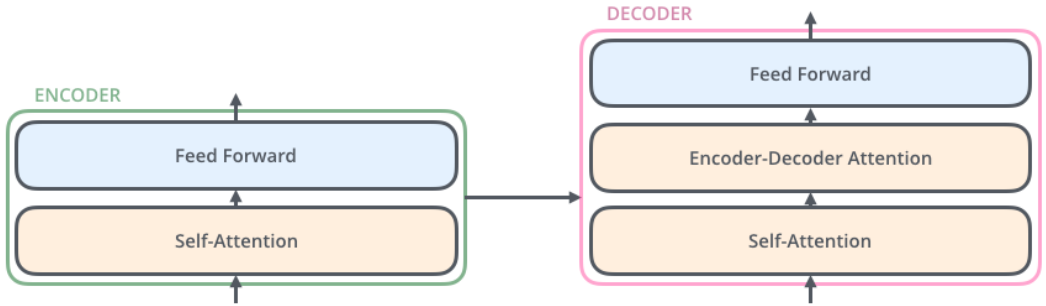
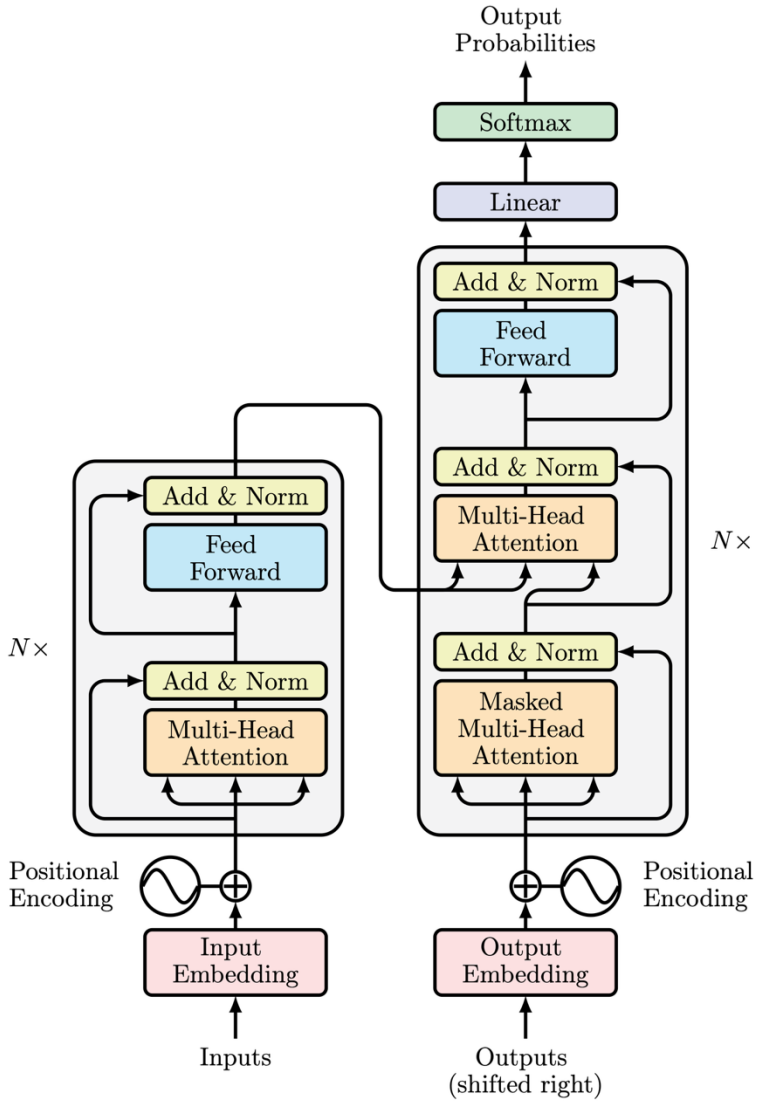


Image Credit: [2]

Transformers



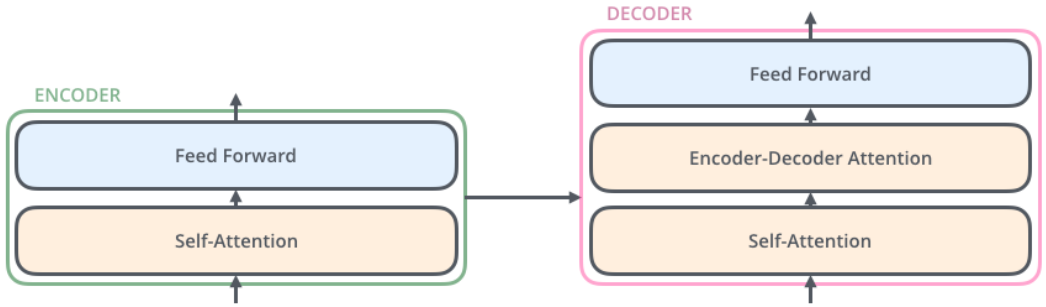
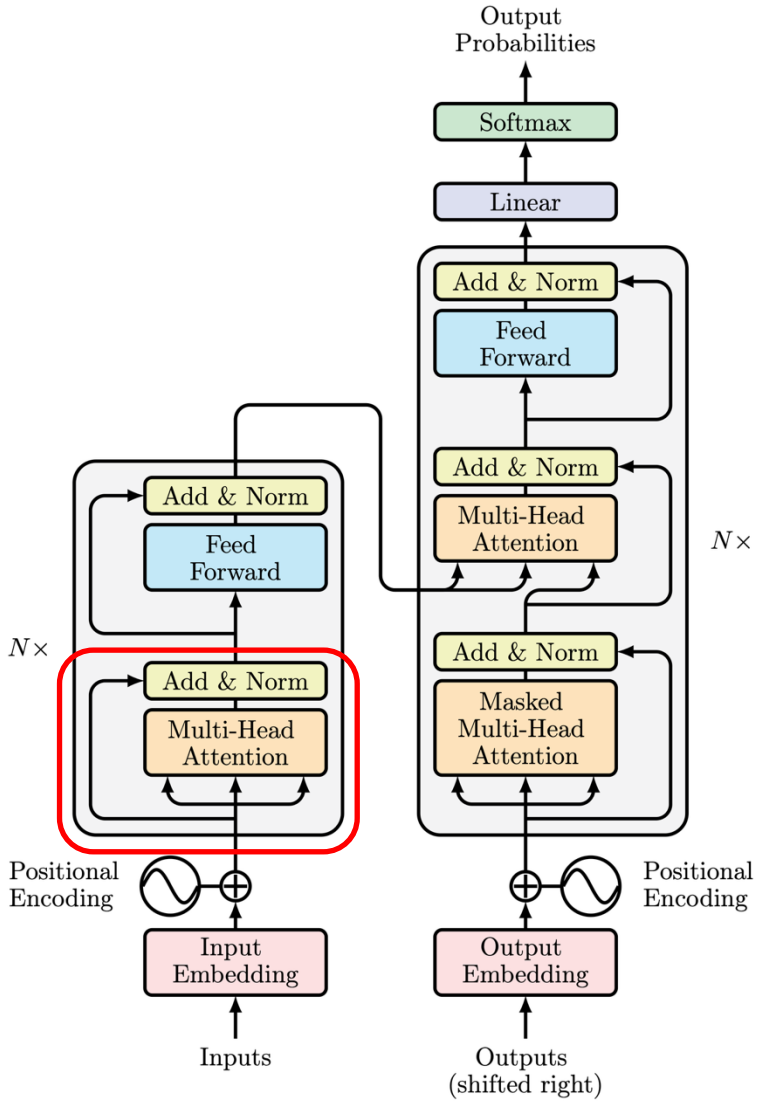


Image Credit: [2, 5]

Transformers



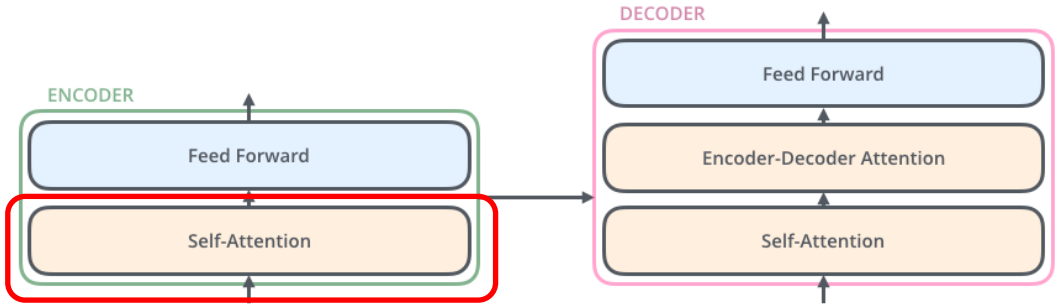
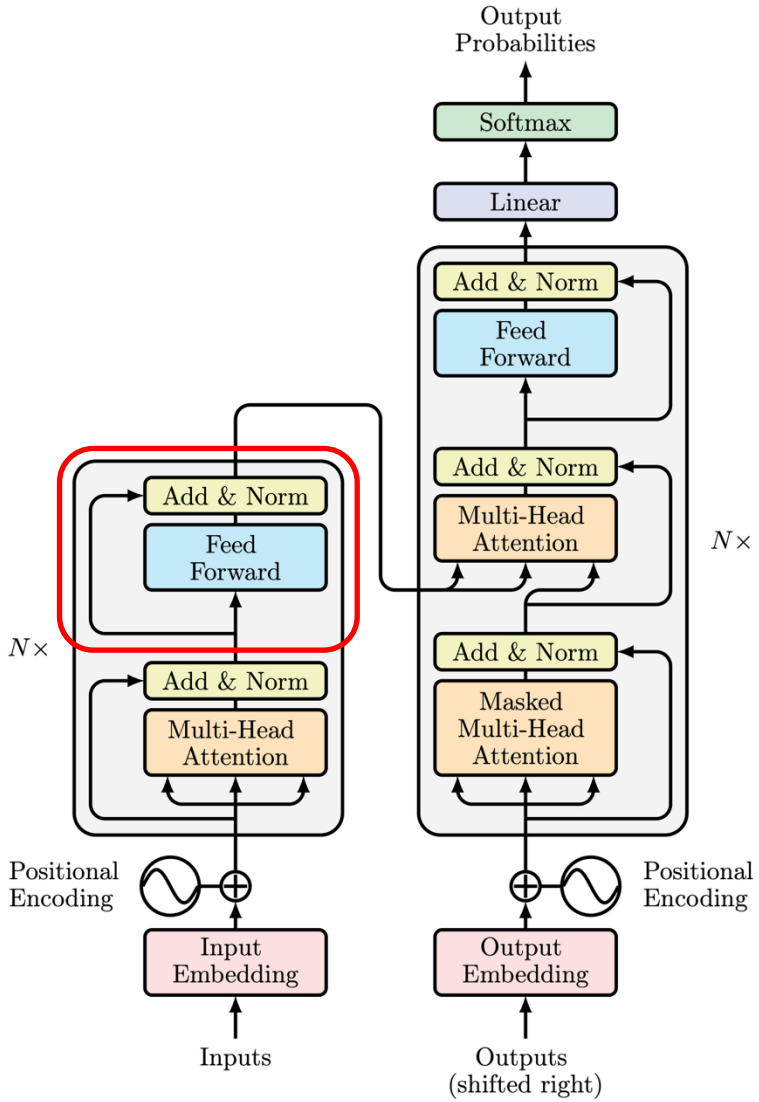


Image Credit: [2, 5]

Transformers



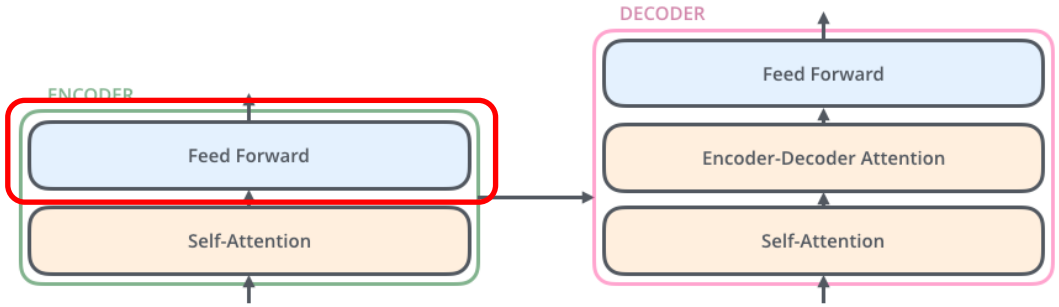
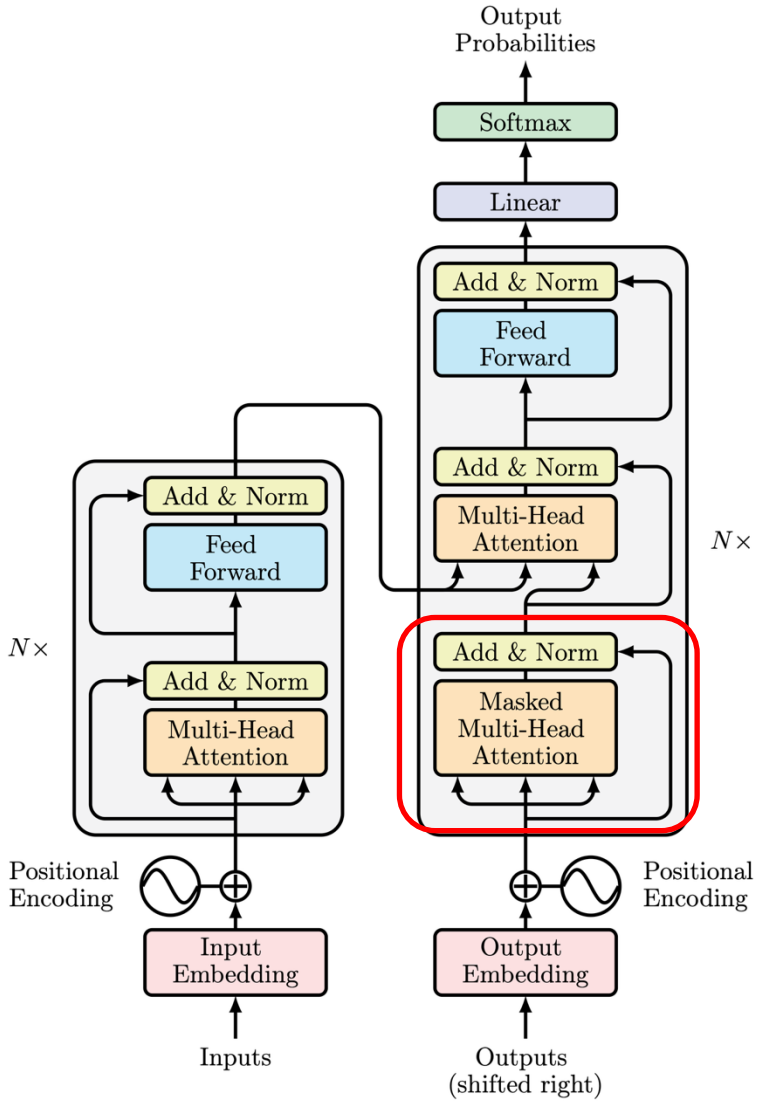


Image Credit: [2, 5]

Transformers



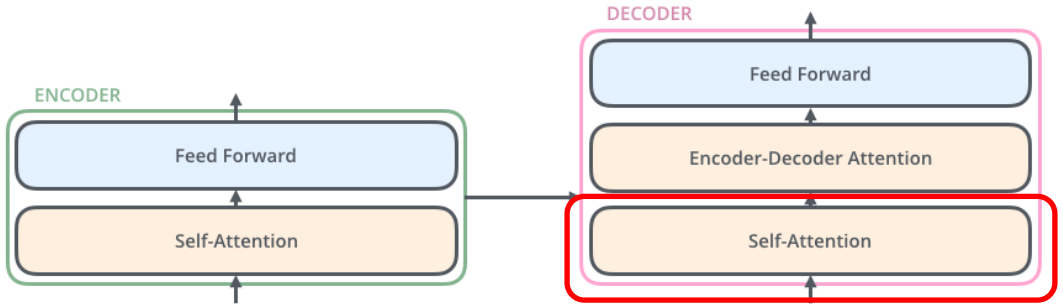
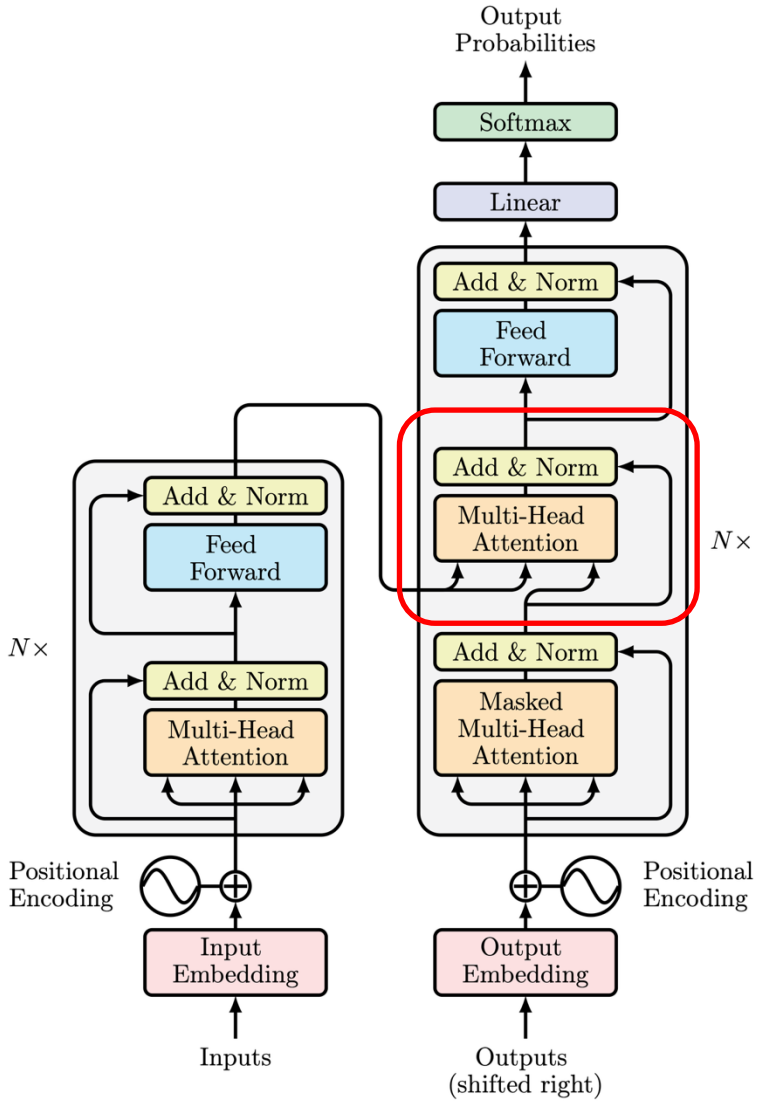


Image Credit: [2, 5]

Transformers



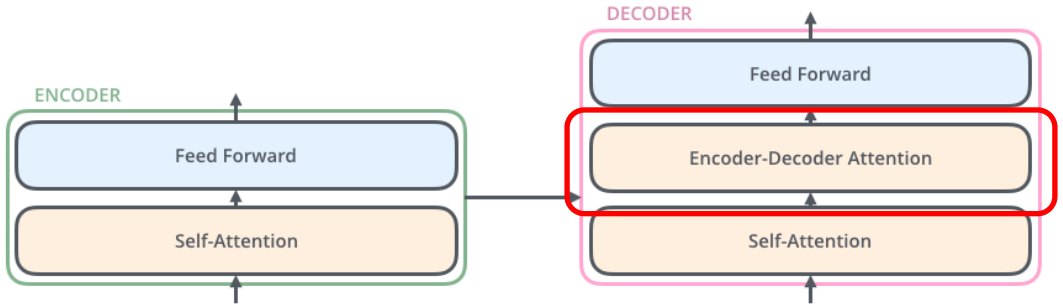
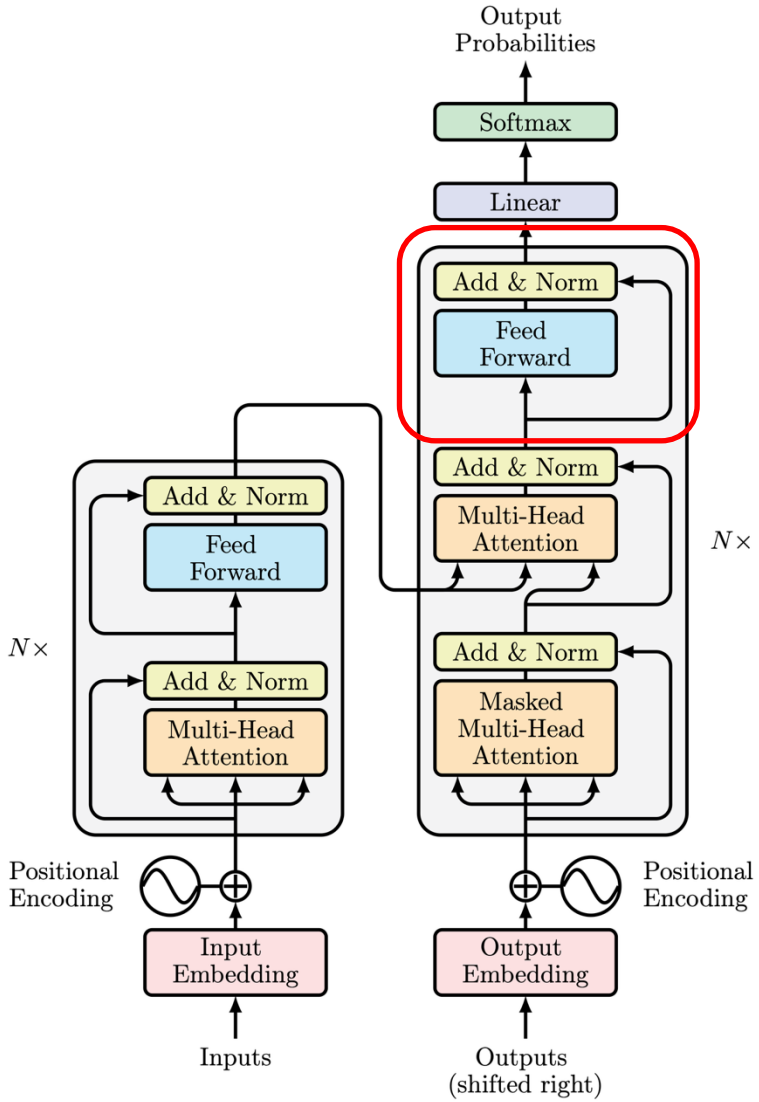


Image Credit: [2, 5]

Transformers



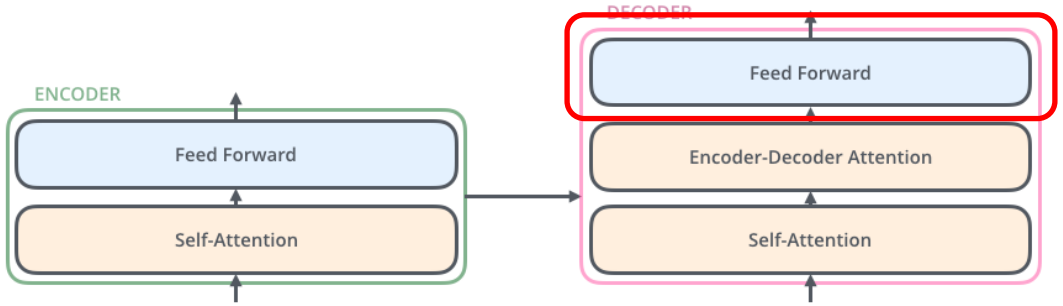


Image Credit: [2, 5]

Outline

- Applications and Challenges of Sequence Modeling
- Transformers
  - **Input Embedding & Sinusoidal Positional Encoding**
  - Encoder
    - Multi-Head Self-Attention, Residual Connections & LayerNorm
  - Decoder
    - Masked Self-Attention, Cross-Attention
- Limitations & Variants
  - Efficient Attention (FlashAttention, Sparse & Linear)
  - Rotary Positional Embedding (RoPE)
  - Normalization (LayerNorm/RMSNorm, Pre-/Post-Norm)
  - Gated MLPs (SwiGLU)
  - Vision Transformers (ViT & Swin)

Input Encoding

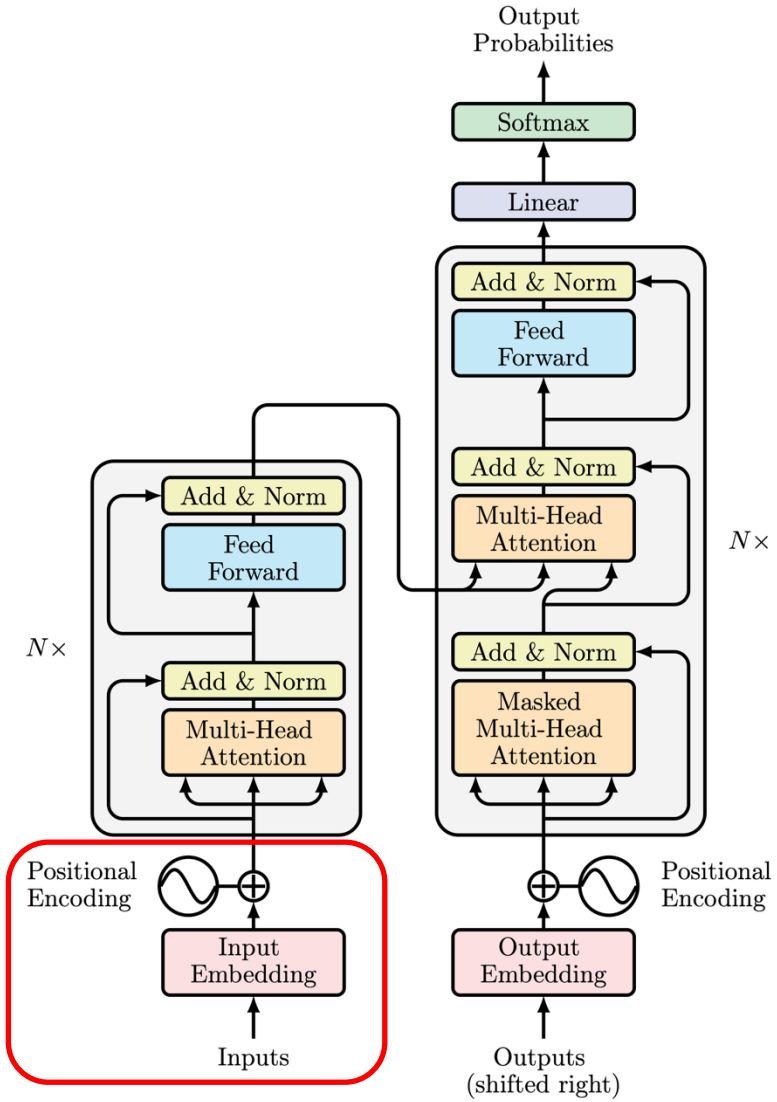
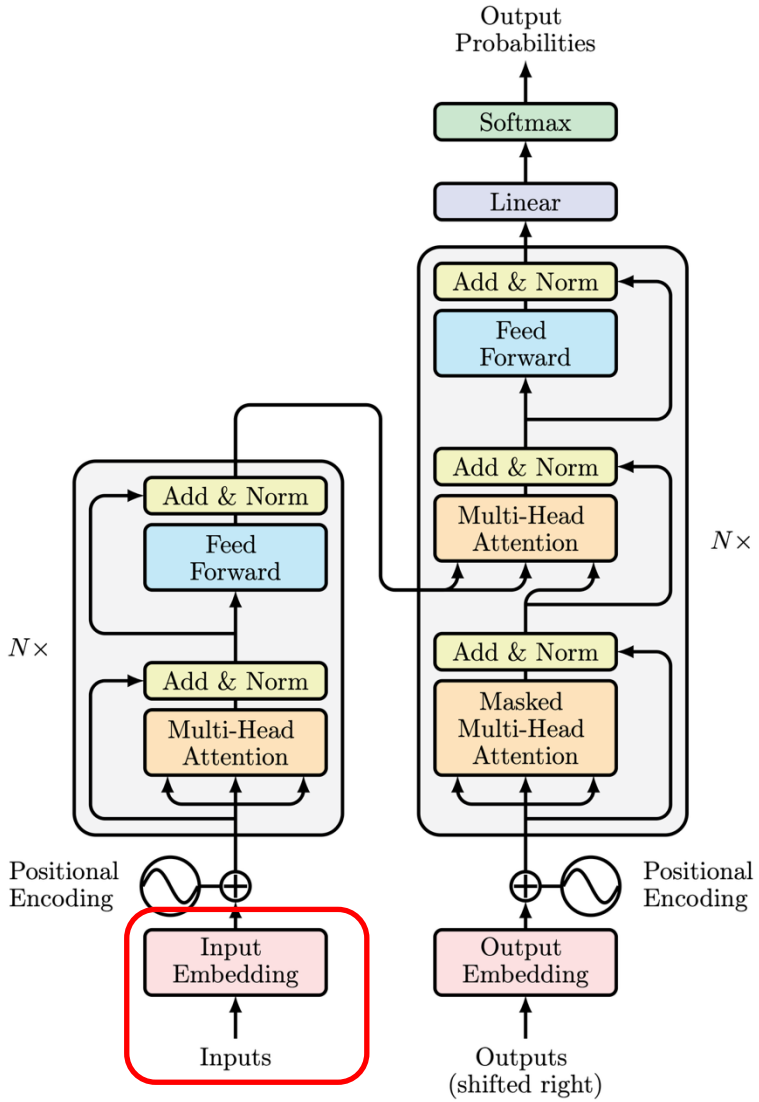


Image Credit: [5]

Input Embedding



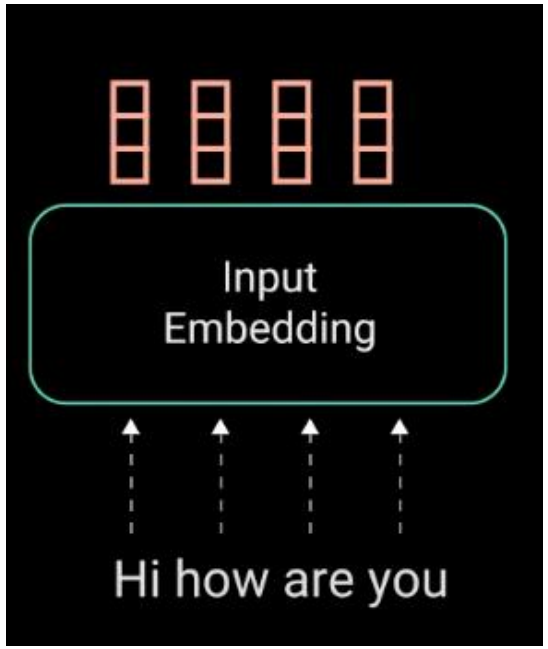
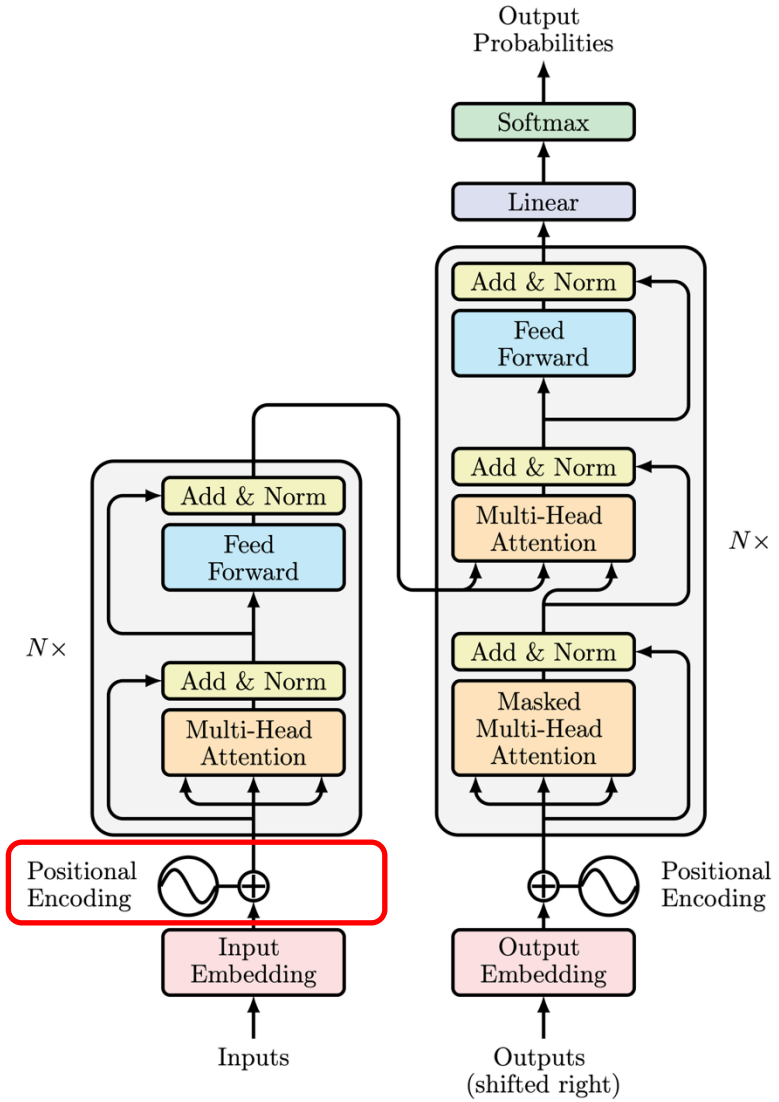
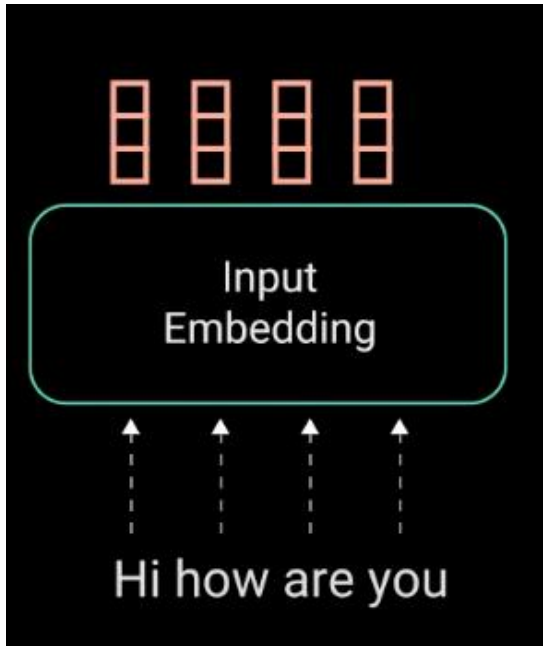


Image Credit: [4, 5]

Positional Encoding



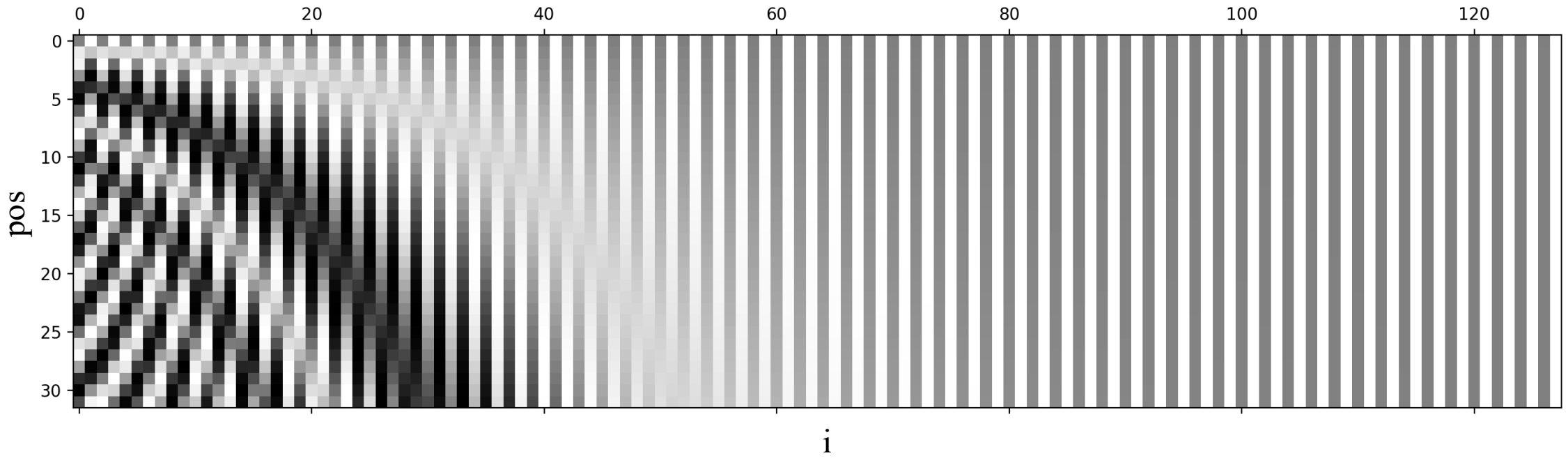


$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}})$$

Image Credit: [4, 5]

Positional Encoding



$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$

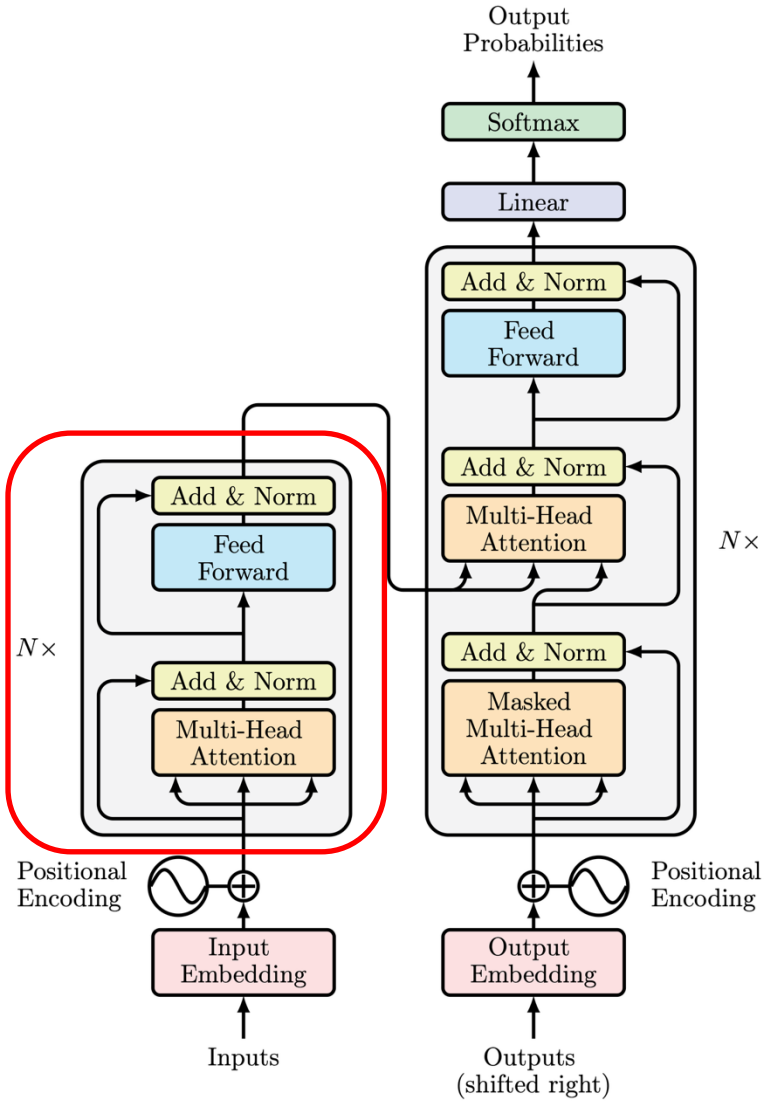
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}})$$

Image Credit: [6]

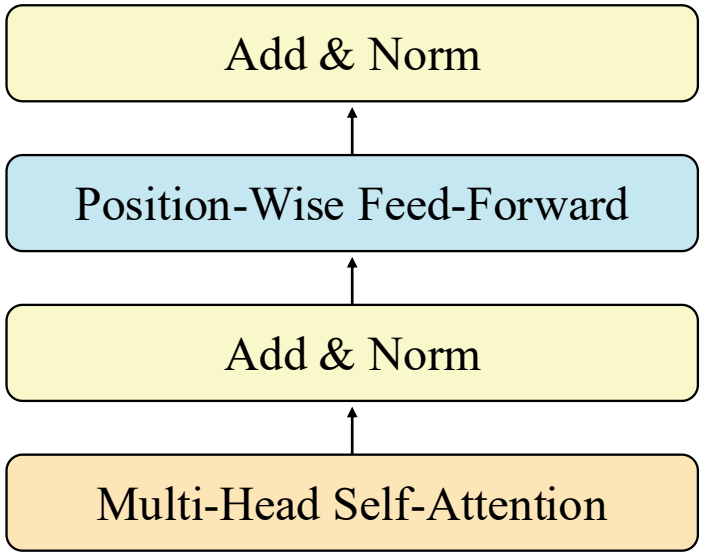
Outline

- Applications and Challenges of Sequence Modeling
- Transformers
  - Input Embedding & Sinusoidal Positional Encoding
  - **Encoder**
    - Multi-Head Self-Attention, Residual Connections & LayerNorm
  - Decoder
    - Masked Self-Attention, Cross-Attention
- Limitations & Variants
  - Efficient Attention (FlashAttention, Sparse & Linear)
  - Rotary Positional Embedding (RoPE)
  - Normalization (LayerNorm/RMSNorm, Pre-/Post-Norm)
  - Gated MLPs (SwiGLU)
  - Vision Transformers (ViT & Swin)

Encoder



Each encoder block [5]



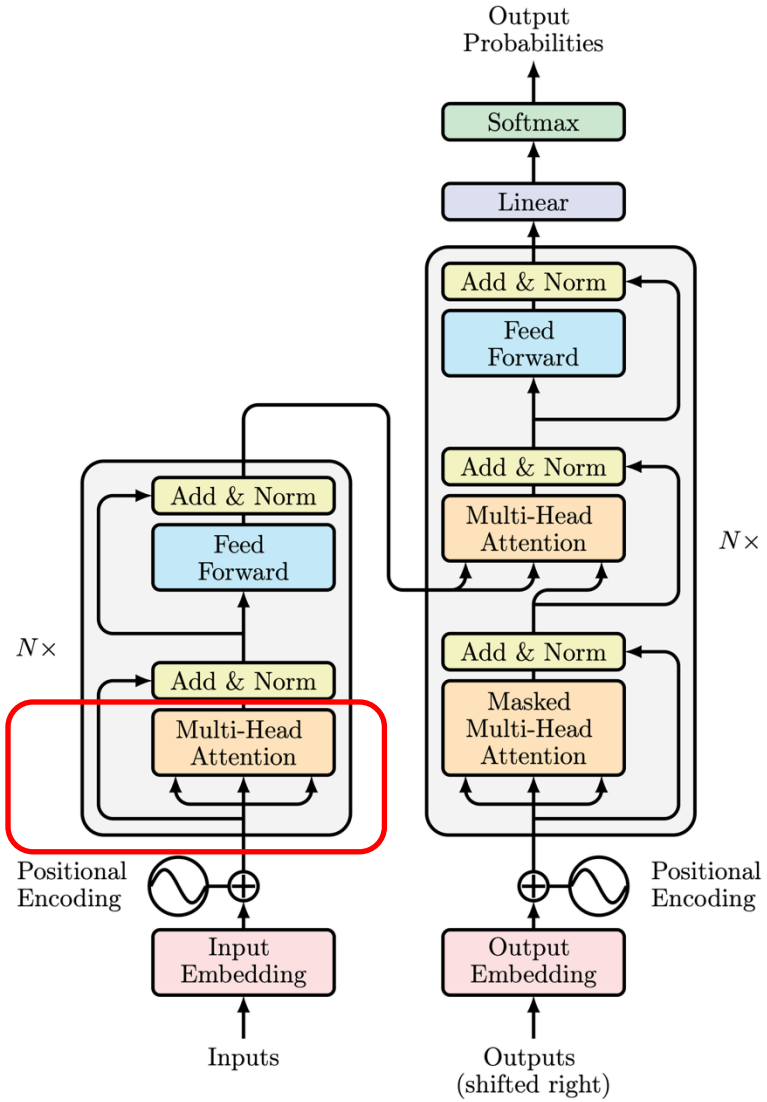
Repeated N times to form contextual representations of the source tokens.

Image Credit: [4, 5]

Outline

- Applications and Challenges of Sequence Modeling
- Transformers
  - Input Embedding & Sinusoidal Positional Encoding
  - Encoder
    - **Multi-Head Self-Attention**, Residual Connections & LayerNorm
  - Decoder
    - Masked Self-Attention, Cross-Attention
- Limitations & Variants
  - Efficient Attention (FlashAttention, Sparse & Linear)
  - Rotary Positional Embedding (RoPE)
  - Normalization (LayerNorm/RMSNorm, Pre-/Post-Norm)
  - Gated MLPs (SwiGLU)
  - Vision Transformers (ViT & Swin)

Multi-Head Attention



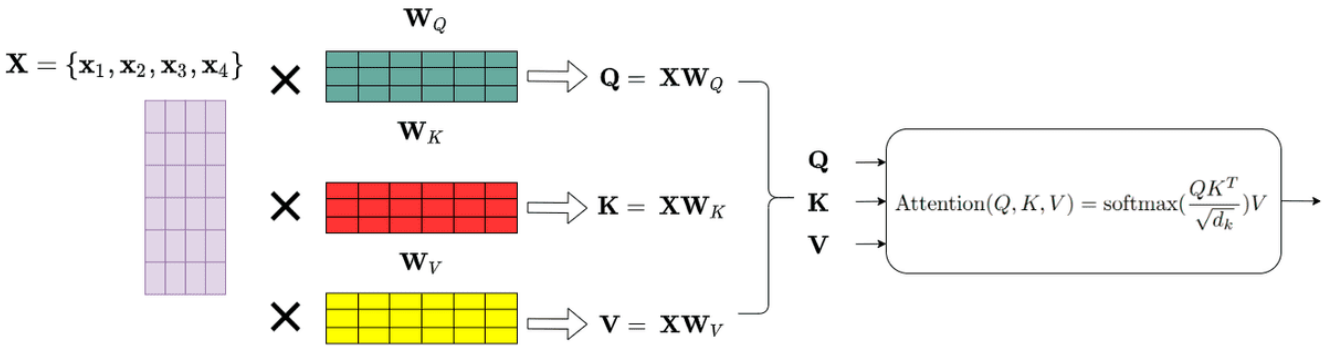
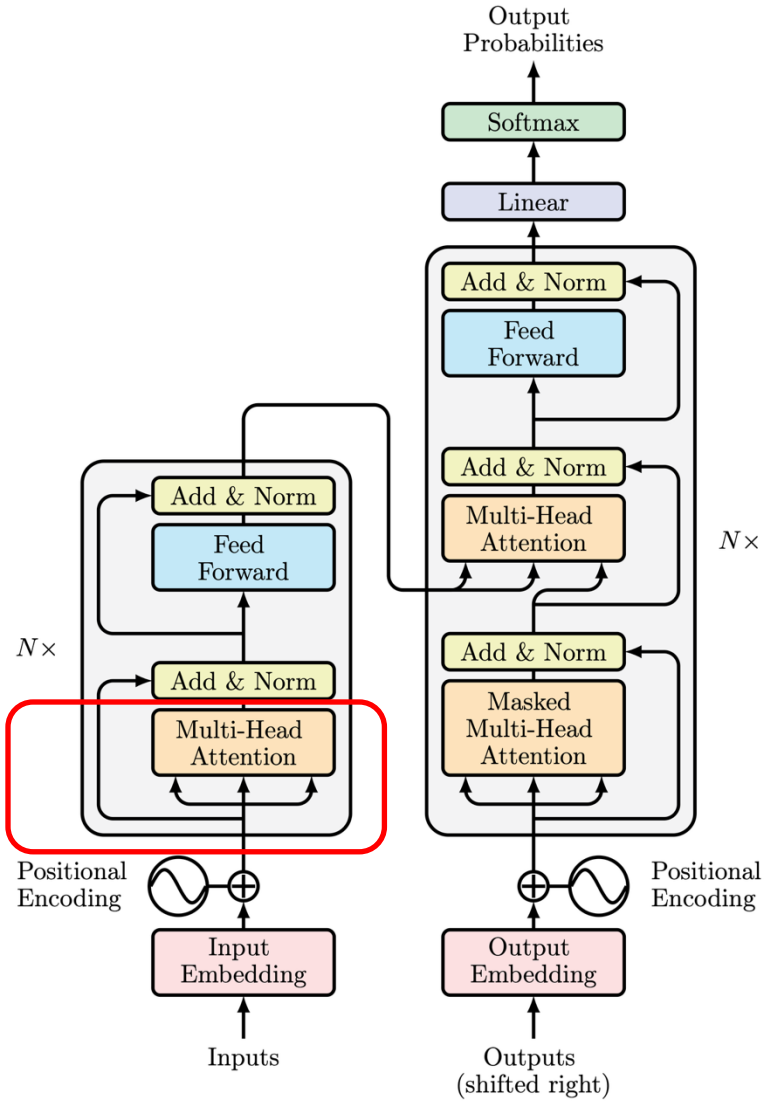


Image Credit: [5, 7]

Multi-Head Attention



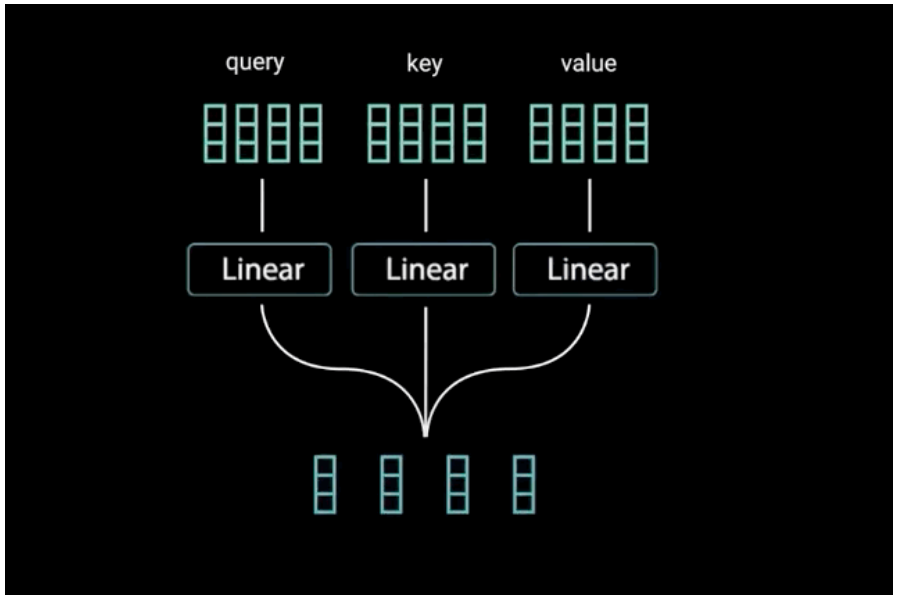
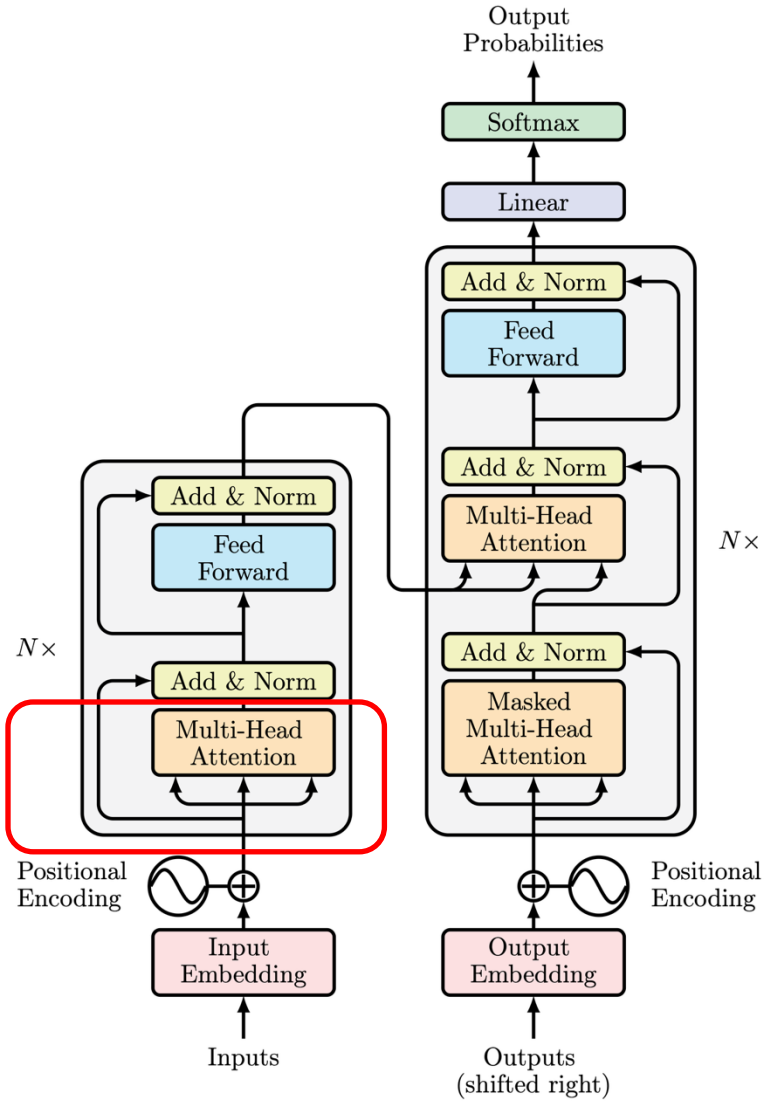
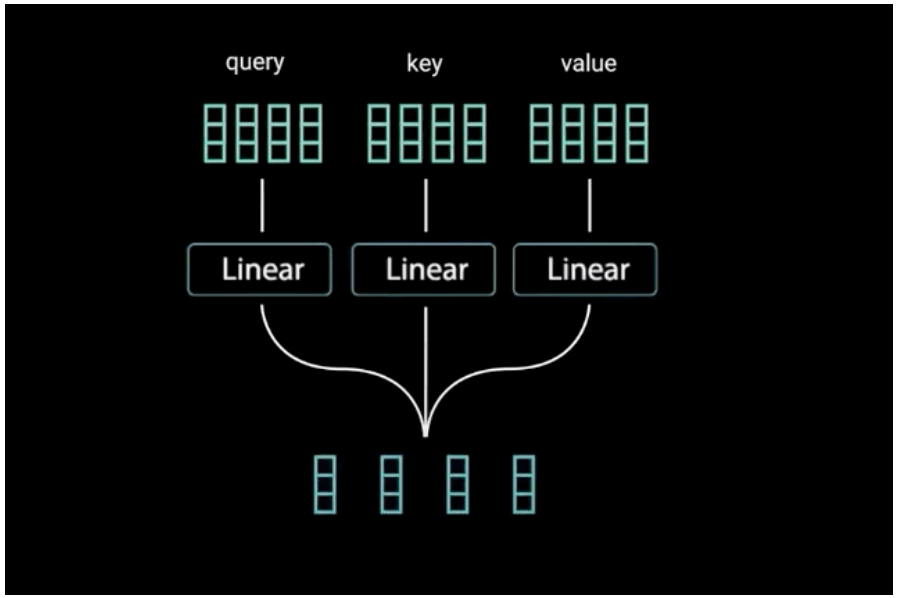


Image Credit: [4, 5]

Multi-Head Attention



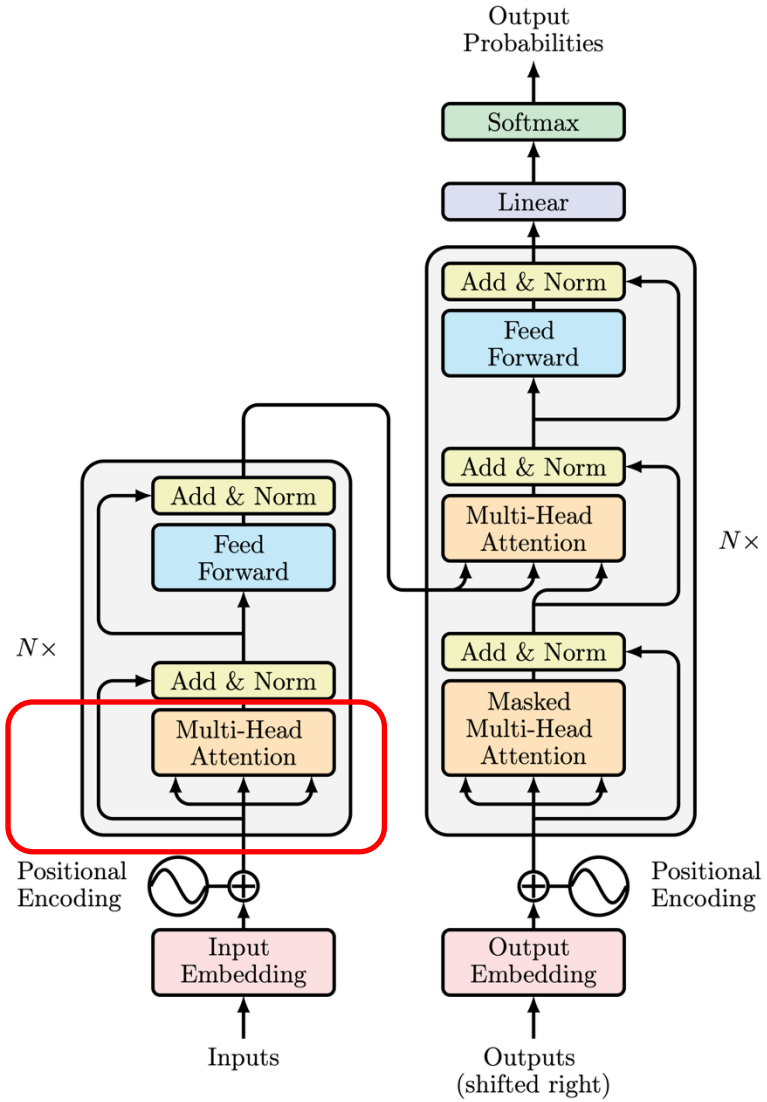


Dot products

	Hi	how	are	you
Hi	4	-2	0	2
how	0	2	4	-2
are	-2	2	4	0
you	2	0	-2	4

Visuals adapted from [4, 5]. Numerical example is illustrative.

Multi-Head Attention



Dot products

	Hi	how	are	you
Hi	4	-2	0	2
how	0	2	4	-2
are	-2	2	4	0
you	2	0	-2	4

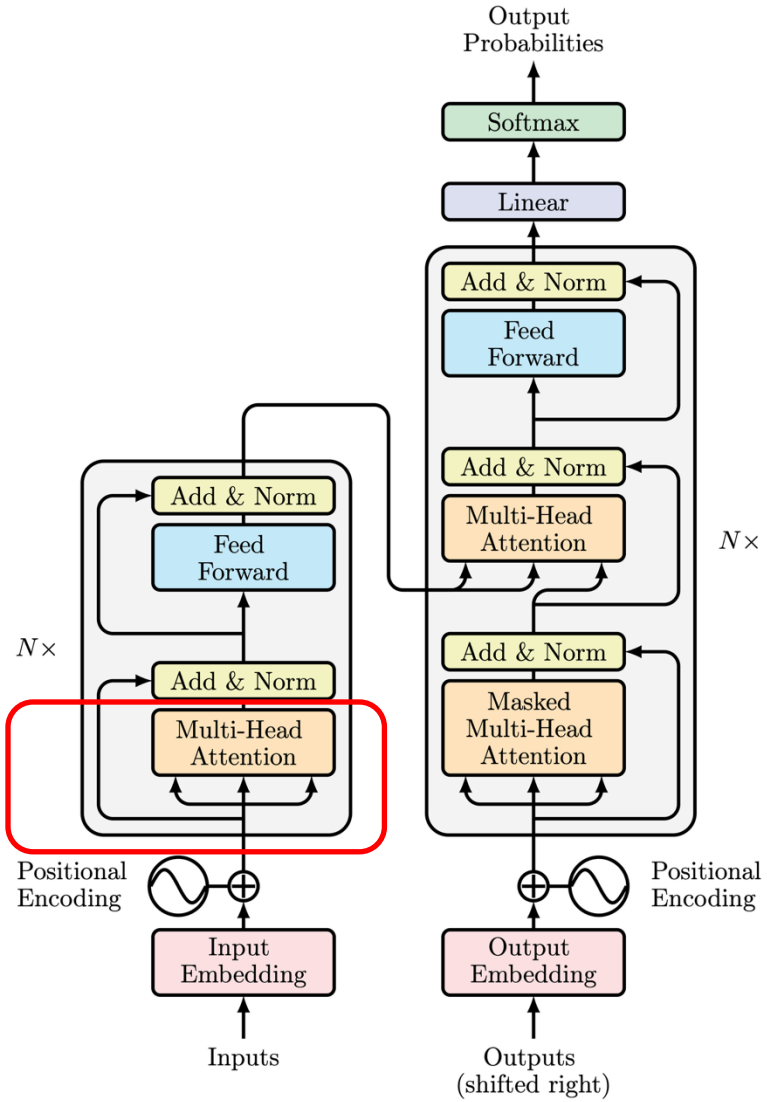
Scaled scores (before softmax)

$$S = \frac{QK^\top}{\sqrt{d_k}}, \qquad d_k = 4$$

	Hi	how	are	you
Hi	2	-1	0	1
how	0	1	2	-1
are	-1	1	2	0
you	1	0	-1	2

Visuals adapted from [4, 5]. Numerical example is illustrative.

Multi-Head Attention



Scaled scores (before softmax)

$$S = \frac{QK^\top}{\sqrt{d_k}}, \qquad d_k = 4$$

	Hi	how	are	you
Hi	2	-1	0	1
how	0	1	2	-1
are	-1	1	2	0
you	1	0	-1	2

Row-wise softmax

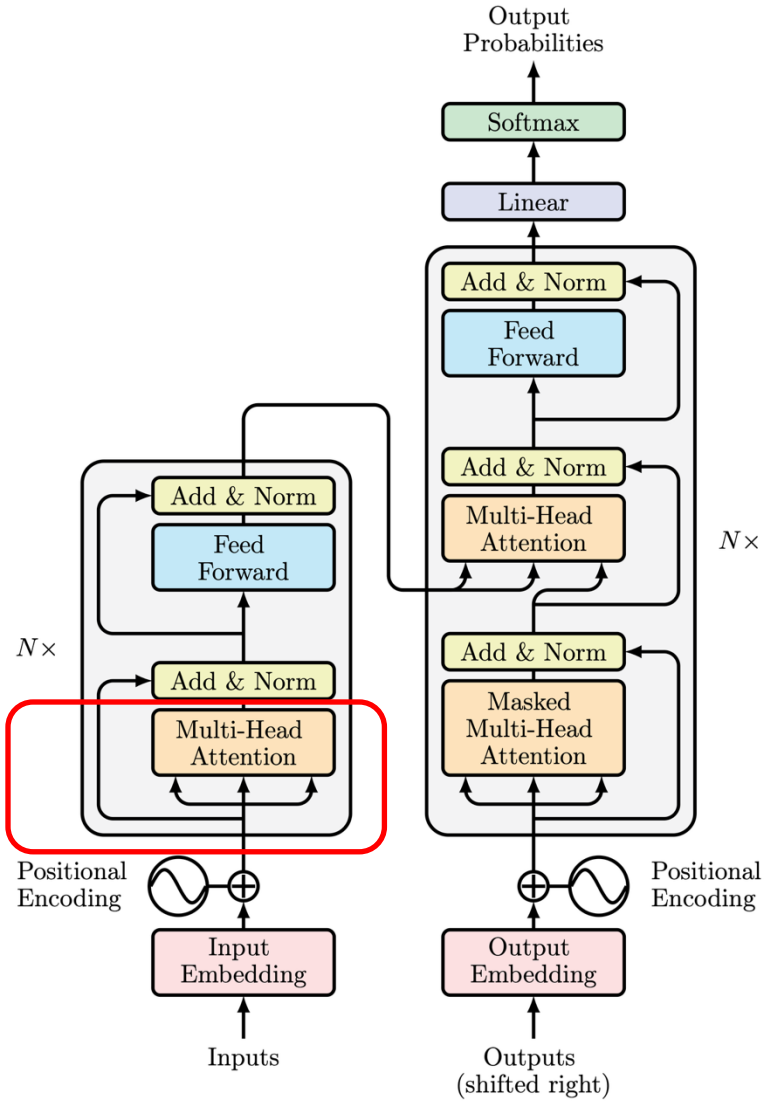
	Hi	how	are	you
Hi	.644	.032	.087	.237
how	.087	.237	.644	.032
are	.032	.237	.644	.087
you	.237	.087	.032	.644

$$P_{ij} = \frac{\exp(S_{ij})}{\sum_\ell \exp(S_{i\ell})}$$

Rows sum to 1 (up to rounding)

Visuals adapted from [4, 5]. Numerical example is illustrative.

Multi-Head Attention



Scaled scores (before softmax)

$$S = \frac{QK^\top}{\sqrt{d_k}}, \qquad d_k = 4$$

	Hi	how	are	you
Hi	2	-1	0	1
how	0	1	2	-1
are	-1	1	2	0
you	1	0	-1	2

Why square root?

Row-wise softmax

	Hi	how	are	you
Hi	.644	.032	.087	.237
how	.087	.237	.644	.032
are	.032	.237	.644	.087
you	.237	.087	.032	.644

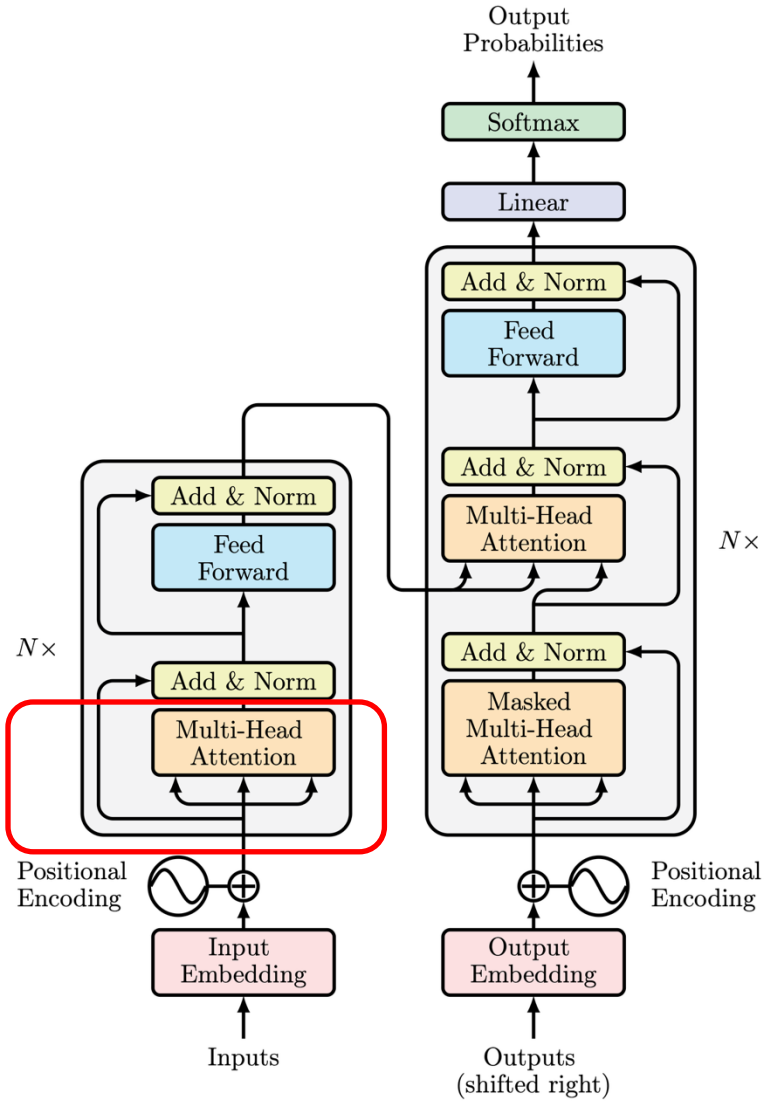
$$P_{ij} = \frac{\exp(S_{ij})}{\sum_\ell \exp(S_{i\ell})}$$

Rows sum to 1 (up to rounding)

Hint: recall the variance analysis for weight initialization.

Visuals adapted from [4, 5]. Numerical example is illustrative.

Multi-Head Attention

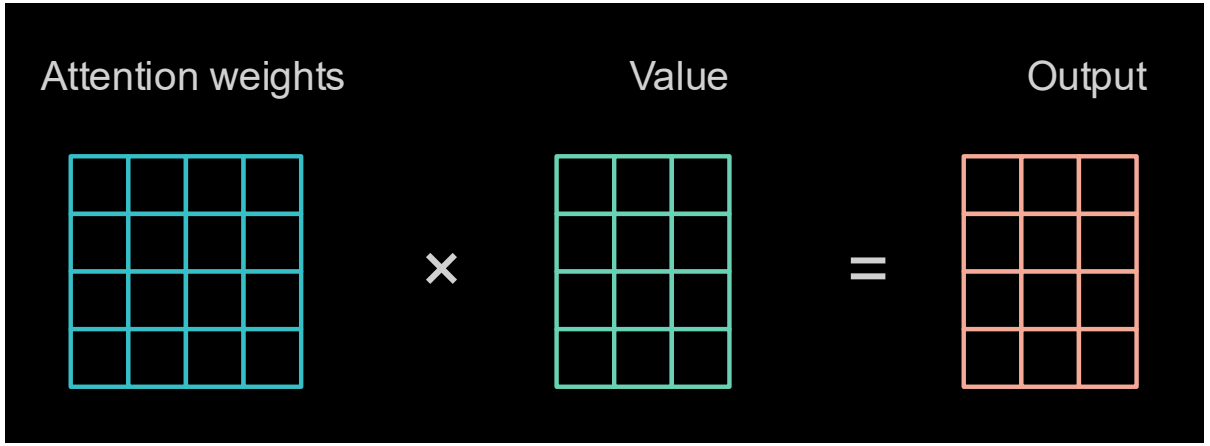


Row-wise softmax

	Hi	how	are	you
Hi	.644	.032	.087	.237
how	.087	.237	.644	.032
are	.032	.237	.644	.087
you	.237	.087	.032	.644

$$P_{ij} = \frac{\exp(S_{ij})}{\sum_{\ell} \exp(S_{i\ell})}$$

Rows sum to 1 (up to rounding)

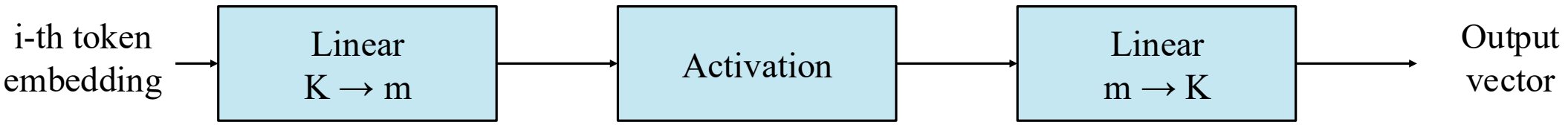


Visuals adapted from [4, 5]. Numerical example is illustrative.

Position-Wise Feed-Forward Networks

Row i of X is the i-th token embedding. Apply the same MLP to every row [5].

$$\mathrm{FFN}(X_i) = \phi(X_i W_1) W_2, \qquad W_1 \in \mathbb{R}^{K \times m}, \quad W_2 \in \mathbb{R}^{m \times K}$$



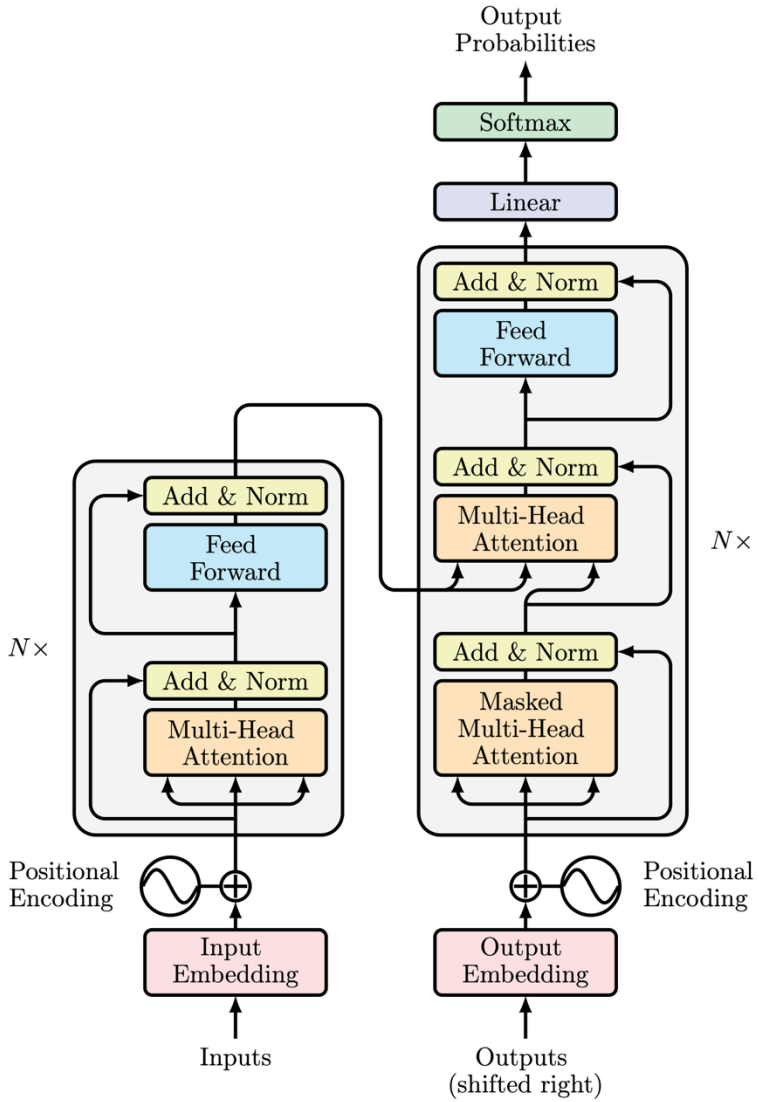
Same weights at every position; this sublayer does not mix tokens.

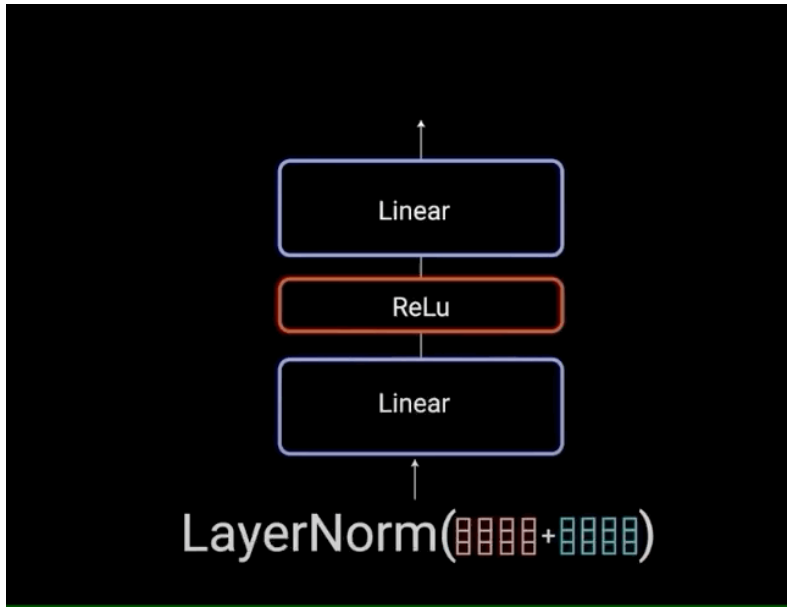
The activation can be ReLU or GELU. Biases are omitted here.

Outline

- Applications and Challenges of Sequence Modeling
- Transformers
  - Input Embedding & Sinusoidal Positional Encoding
  - Encoder
    - Multi-Head Self-Attention, **Residual Connections & LayerNorm**
  - Decoder
    - Masked Self-Attention, Cross-Attention
- Limitations & Variants
  - Efficient Attention (FlashAttention, Sparse & Linear)
  - Rotary Positional Embedding (RoPE)
  - Normalization (LayerNorm/RMSNorm, Pre-/Post-Norm)
  - Gated MLPs (SwiGLU)
  - Vision Transformers (ViT & Swin)

Residual Connections





$$Y = X + \text{Sublayer}(X)$$

Attention and feed-forward blocks each add their output to the input.

Skip paths help propagate information and gradients.

Image Credit: [4, 5]

Layer Normalization

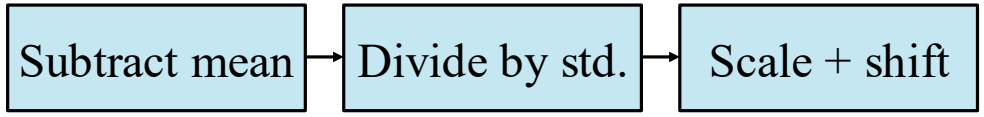
Normalize token i across its K features. The original Transformer uses LayerNorm [5].

For one token

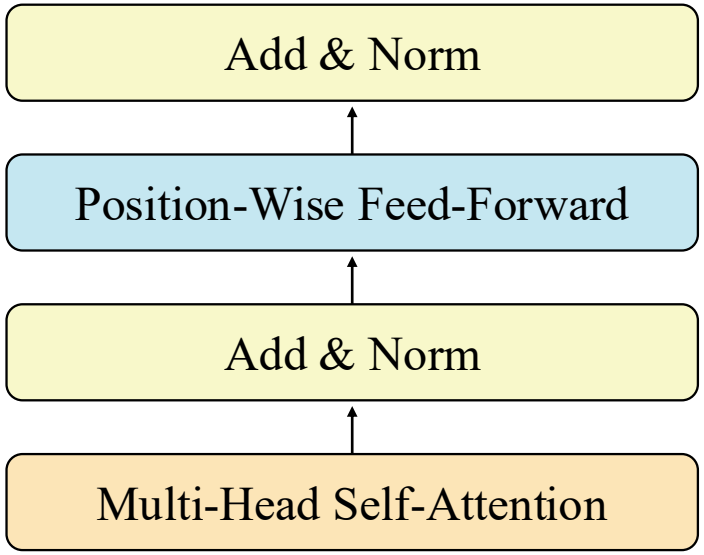
$$\mu_i = \frac{1}{K}\sum_{k=1}^{K} X_{i,k}$$

$$\sigma_i^2 = \frac{1}{K}\sum_{k=1}^{K}(X_{i,k} - \mu_i)^2$$

$$\mathrm{LN}(X_i)_k = \gamma_k \frac{X_{i,k} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} + \beta_k$$



Within the encoder block

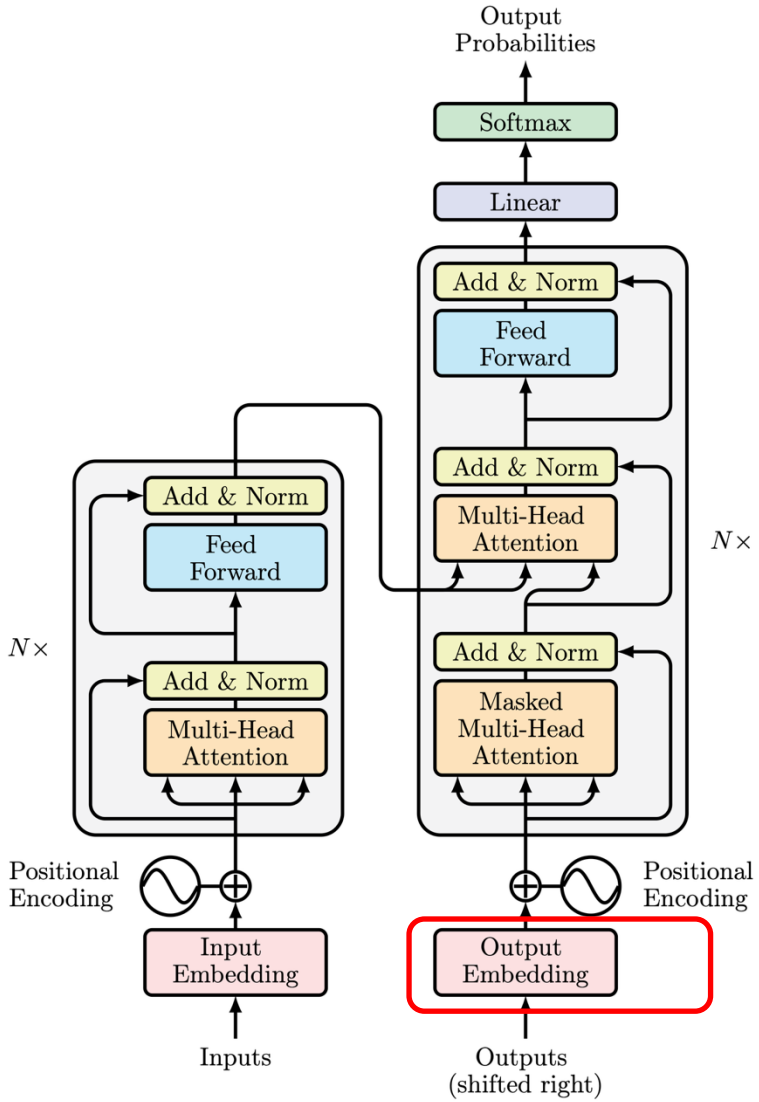


LayerNorm uses each token's feature statistics independently.
The original Add & Norm applies it after residual addition.

Outline

- Applications and Challenges of Sequence Modeling
- Transformers
  - Input Embedding & Sinusoidal Positional Encoding
  - Encoder
    - Multi-Head Self-Attention, Residual Connections & LayerNorm
  - **Decoder**
    - Masked Self-Attention, Cross-Attention
- Limitations & Variants
  - Efficient Attention (FlashAttention, Sparse & Linear)
  - Rotary Positional Embedding (RoPE)
  - Normalization (LayerNorm/RMSNorm, Pre-/Post-Norm)
  - Gated MLPs (SwiGLU)
  - Vision Transformers (ViT & Swin)

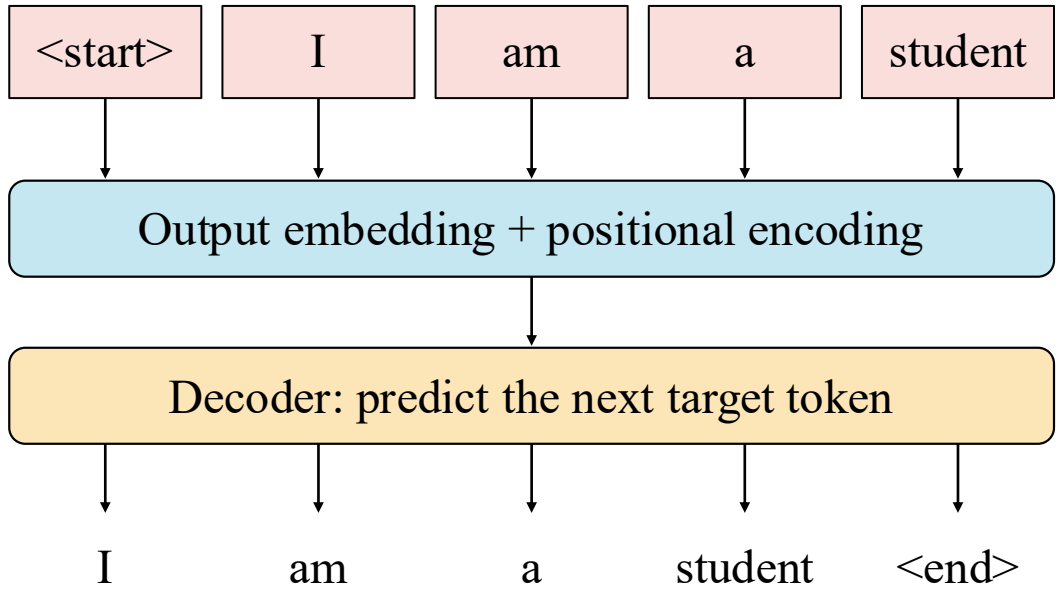
Decoder: Target Embeddings



Embed previous target tokens, not encoder outputs [5].

Translation: Je suis étudiant → I am a student

Decoder inputs (target shifted right)



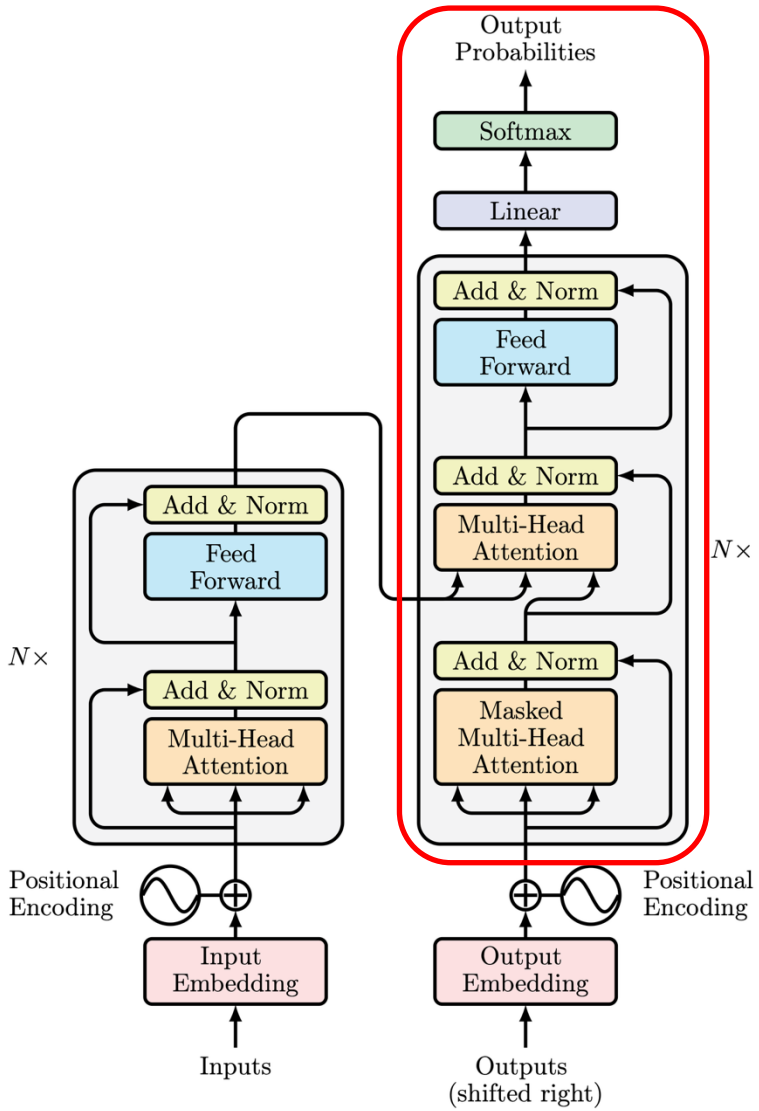
Training: use the shifted target sentence.
Inference: feed generated tokens back as inputs.

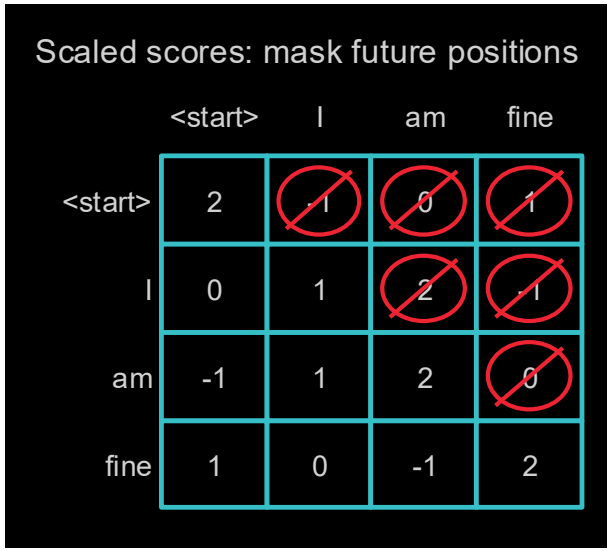
Image Credit: [4, 5]

Outline

- Applications and Challenges of Sequence Modeling
- Transformers
  - Input Embedding & Sinusoidal Positional Encoding
  - Encoder
    - Multi-Head Self-Attention, Residual Connections & LayerNorm
  - Decoder
    - **Masked Self-Attention**, Cross-Attention
- Limitations & Variants
  - Efficient Attention (FlashAttention, Sparse & Linear)
  - Rotary Positional Embedding (RoPE)
  - Normalization (LayerNorm/RMSNorm, Pre-/Post-Norm)
  - Gated MLPs (SwiGLU)
  - Vision Transformers (ViT & Swin)

Masked Multi-Head Attention

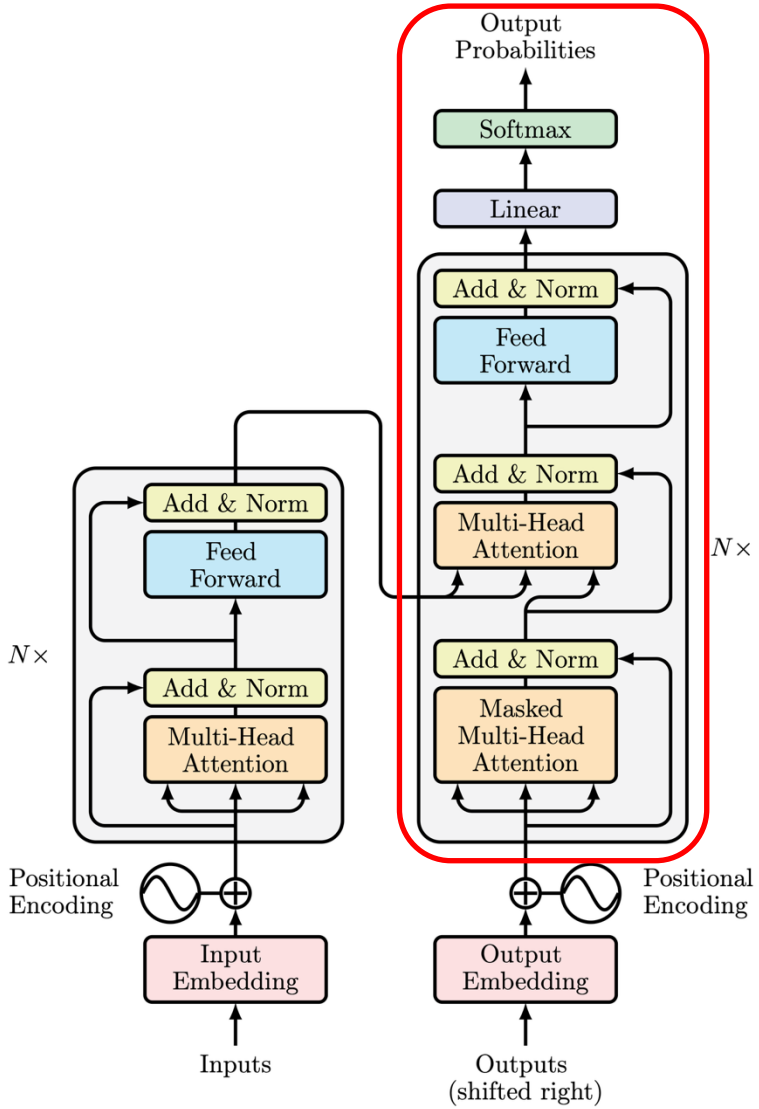


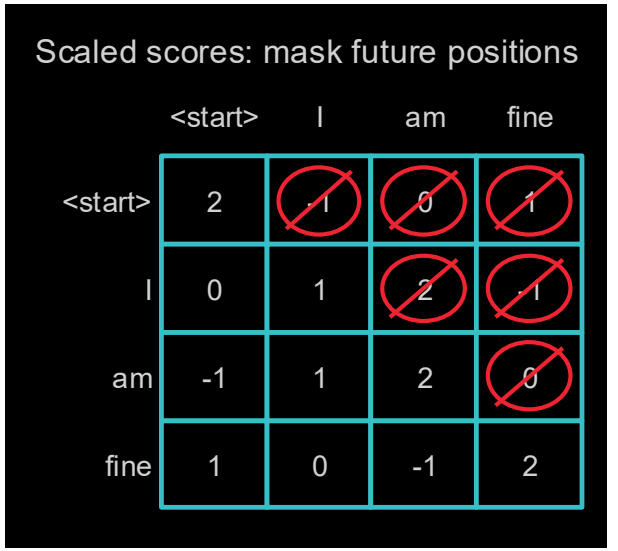


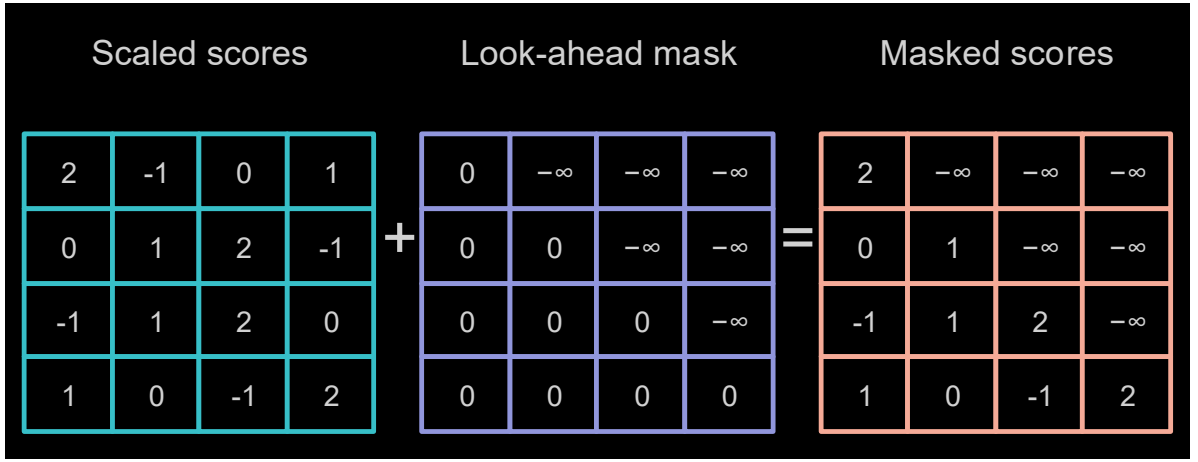
Prevent attending to future target tokens!

Visuals adapted from [4, 5]. Numerical example is illustrative.

Masked Multi-Head Attention

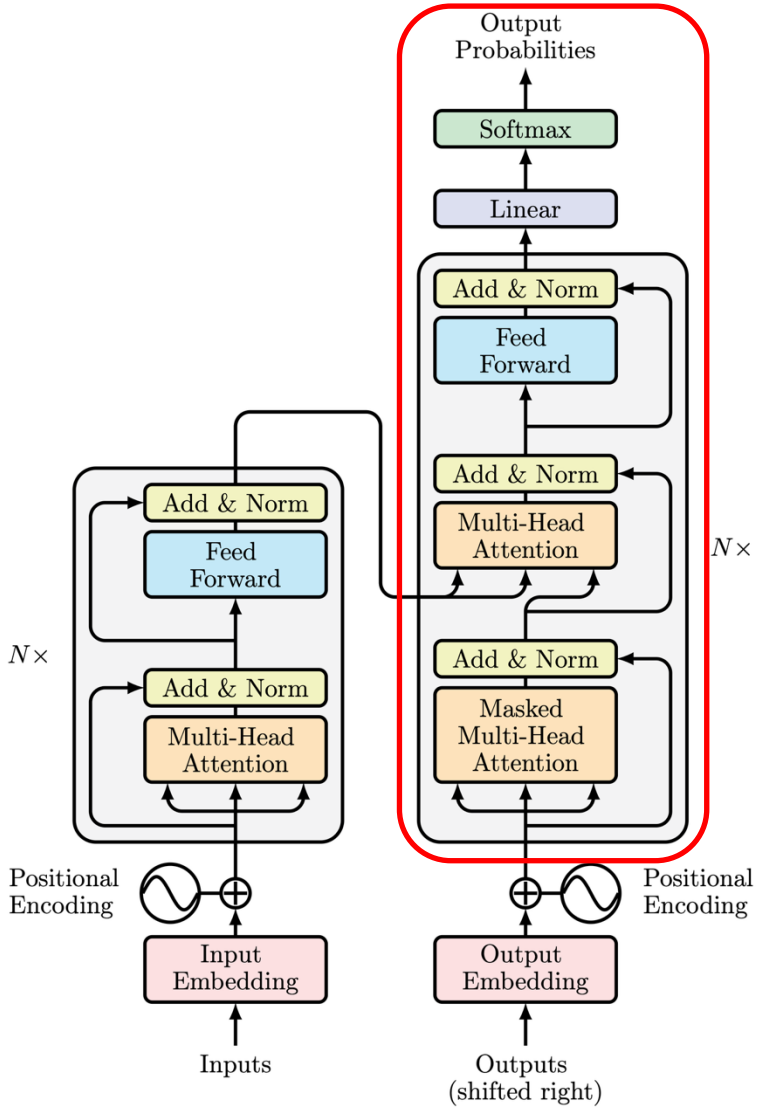






Visuals adapted from [4, 5]. Numerical example is illustrative.

Masked Multi-Head Attention



Scaled scores				+	Look-ahead mask				=	Masked scores			
2	-1	0	1		0	−∞	−∞	−∞		2	−∞	−∞	−∞
0	1	2	-1		0	0	−∞	−∞		0	1	−∞	−∞
-1	1	2	0		0	0	0	−∞		-1	1	2	−∞
1	0	-1	2		0	0	0	0		1	0	-1	2

Row-wise softmax of the masked scores

softmax				=		<start>	I	am	fine
2	−∞	−∞	−∞		<start>	1	0	0	0
0	1	−∞	−∞		I	.269	.731	0	0
-1	1	2	−∞		am	.035	.259	.705	0
1	0	-1	2		fine	.237	.087	.032	.644

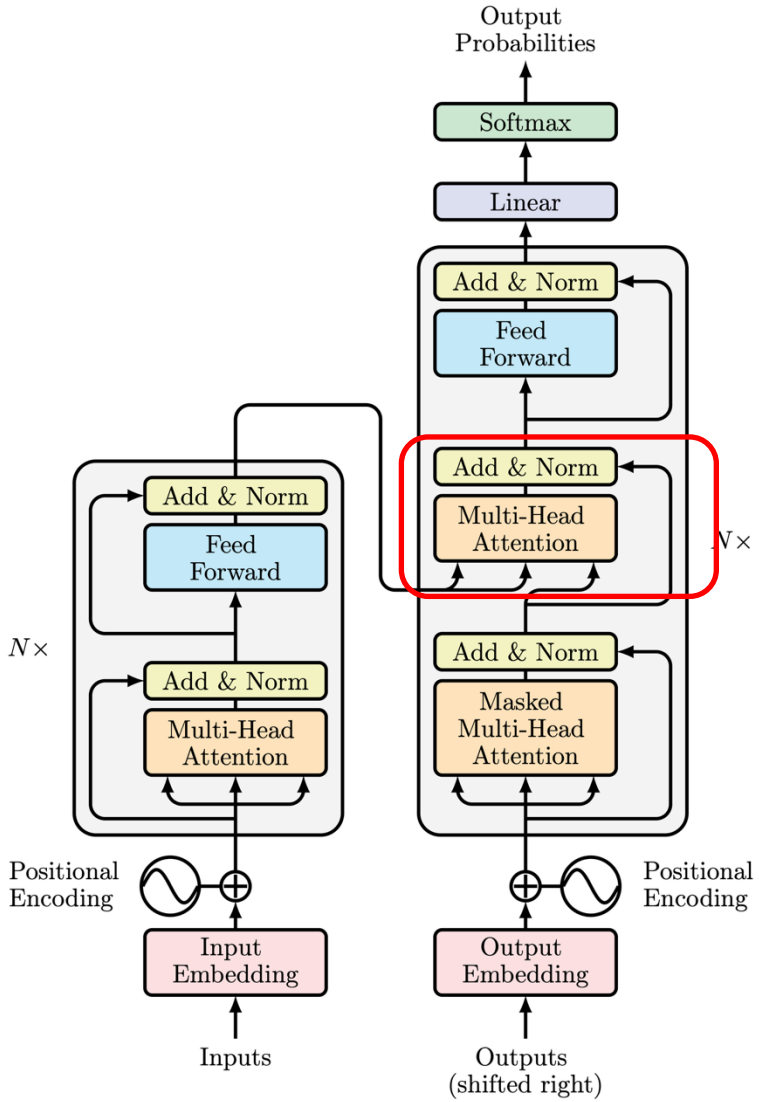
Masked positions receive exactly zero weight.

Visuals adapted from [4, 5]. Numerical example is illustrative.

Outline

- Applications and Challenges of Sequence Modeling
- Transformers
  - Input Embedding & Sinusoidal Positional Encoding
  - Encoder
    - Multi-Head Self-Attention, Residual Connections & LayerNorm
  - Decoder
    - Masked Self-Attention, **Cross-Attention**
- Limitations & Variants
  - Efficient Attention (FlashAttention, Sparse & Linear)
  - Rotary Positional Embedding (RoPE)
  - Normalization (LayerNorm/RMSNorm, Pre-/Post-Norm)
  - Gated MLPs (SwiGLU)
  - Vision Transformers (ViT & Swin)

Cross-Attention



Queries come from the decoder; keys and values come from the encoder [5].

Decoder states after masked attention

Encoder outputs for the full source

$$Q = H_{\text{dec}} W^Q \qquad K = H_{\text{enc}} W^K, \quad V = H_{\text{enc}} W^V$$

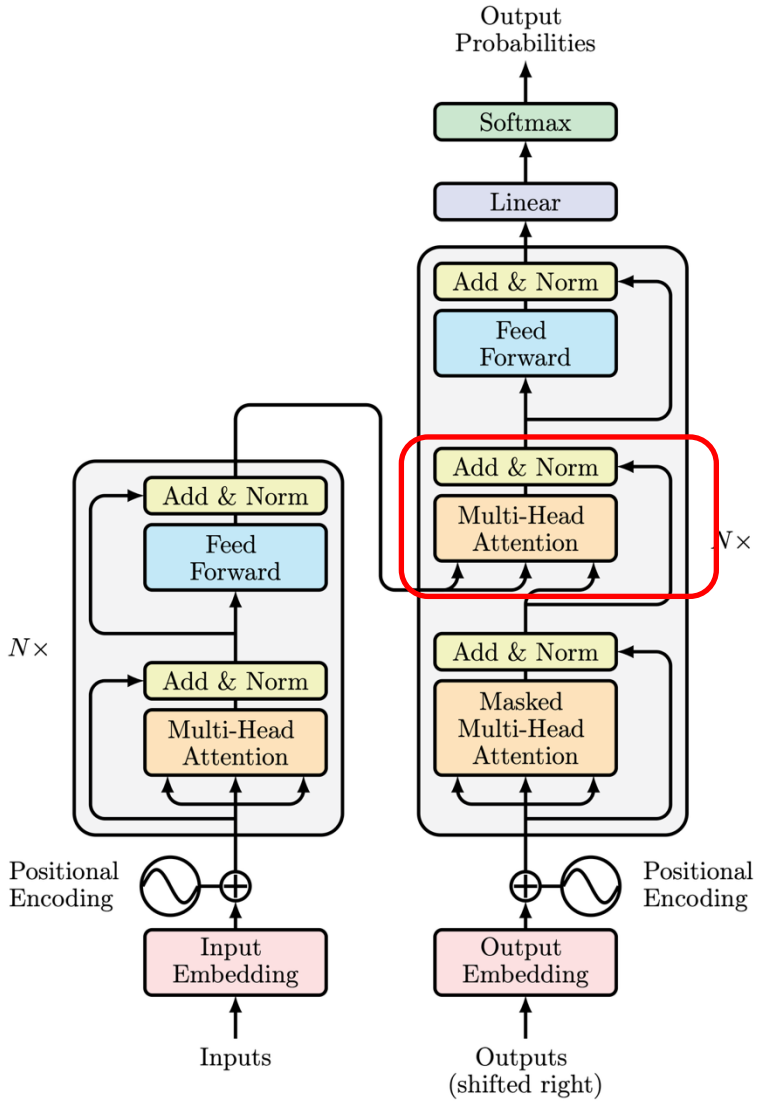
Cross-Attention

$$C = \underbrace{\text{softmax}_{\text{row}}\left(\frac{QK^\top}{\sqrt{d_k}}\right)}_{T \times S} V$$

T target queries attend to S source positions.
Softmax is applied across the source positions.

Image Credit: [4, 5]

Cross-Attention



An illustrative translation alignment [5]

French source keys / values (S = 3)

English query → prediction	Je	suis	étudiant
<start> → I	.800	.100	.100
I → am	.100	.800	.100
am → a	.150	.150	.700
a → student	.050	.100	.850

All non-padding source positions are available;
the causal mask applies to target self-attention.

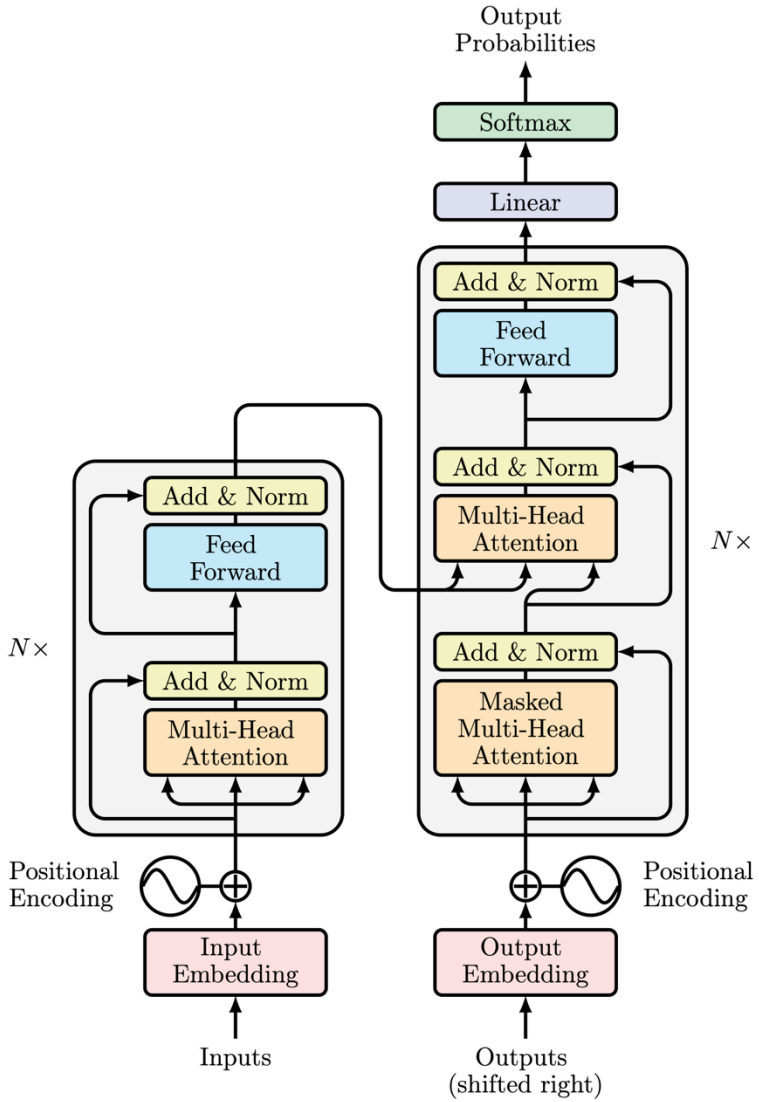
$$c_4 = 0.05v_{\text{Je}} + 0.10v_{\text{suis}} + 0.85v_{\text{étudiant}}$$

Image Credit: [4, 5]

Outline

- Applications and Challenges of Sequence Modeling
- Transformers
  - Input Embedding & Sinusoidal Positional Encoding
  - Encoder
    - Multi-Head Self-Attention, Residual Connections & LayerNorm
  - Decoder
    - Masked Self-Attention, Cross-Attention
- **Limitations & Variants**
  - Efficient Attention (FlashAttention, Sparse & Linear)
  - Rotary Positional Embedding (RoPE)
  - Normalization (LayerNorm/RMSNorm, Pre-/Post-Norm)
  - Gated MLPs (SwiGLU)
  - Vision Transformers (ViT & Swin)

Limitations



Quadratic self-attention cost

For N tokens, self-attention computes N × N scores. Time and score storage grow quadratically: $O(N^2)$.

Improve the implementation [8]

FlashAttention keeps exact attention and avoids storing the full score matrix. Arithmetic remains quadratic.

Change the attention [9, 10, 11]

Sparse patterns inject structure. Kernel methods use a compact summary instead of all pairwise weights.

Image Credit: [5]

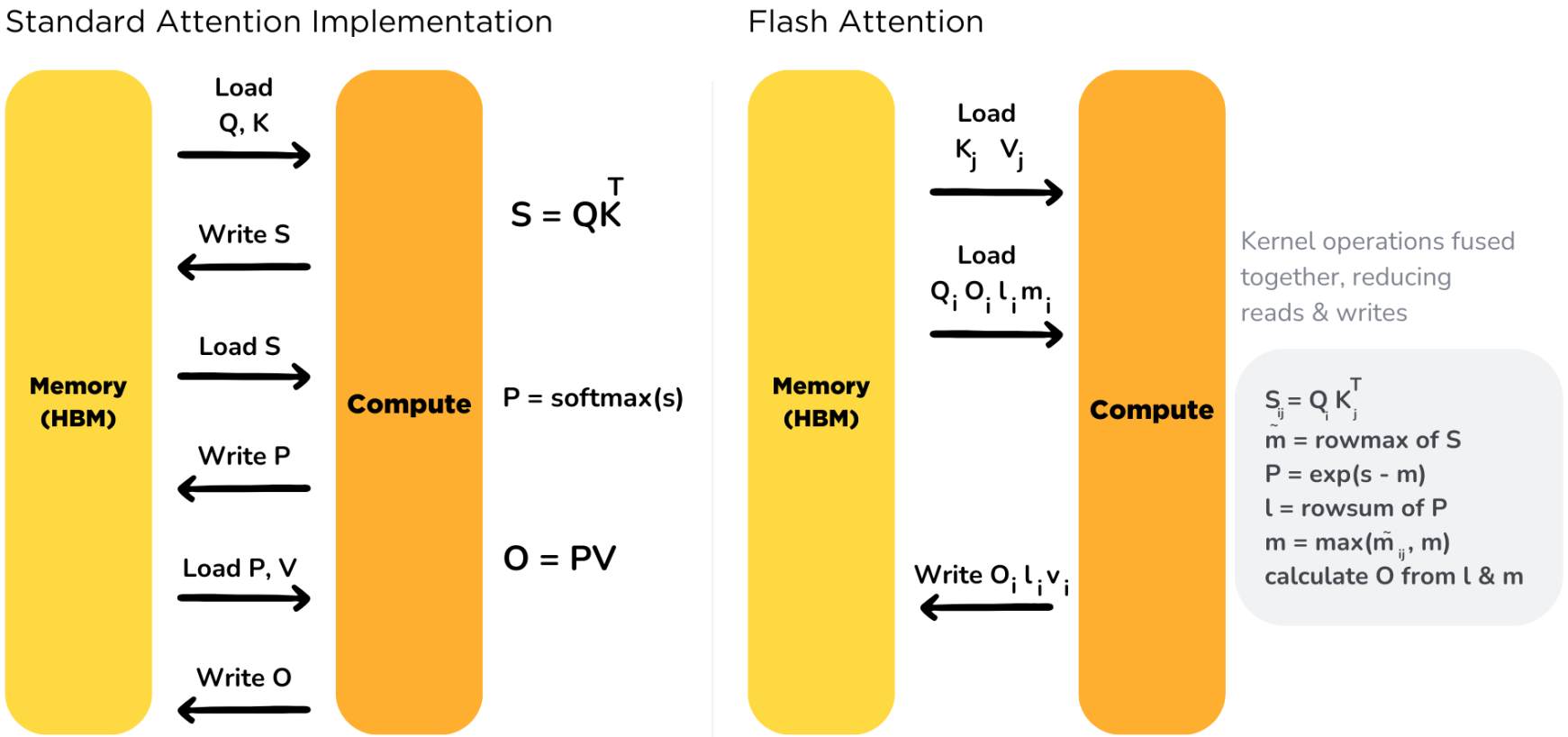
Outline

- Applications and Challenges of Sequence Modeling
- Transformers
  - Input Embedding & Sinusoidal Positional Encoding
  - Encoder
    - Multi-Head Self-Attention, Residual Connections & LayerNorm
  - Decoder
    - Masked Self-Attention, Cross-Attention
- Limitations & Variants
  - **Efficient Attention (FlashAttention, Sparse & Linear)**
  - Rotary Positional Embedding (RoPE)
  - Normalization (LayerNorm/RMSNorm, Pre-/Post-Norm)
  - Gated MLPs (SwiGLU)
  - Vision Transformers (ViT & Swin)

FlashAttention [8]

Compute attention in tiles in fast on-chip SRAM.

Avoid storing the full score matrix in GPU high-bandwidth memory (HBM).



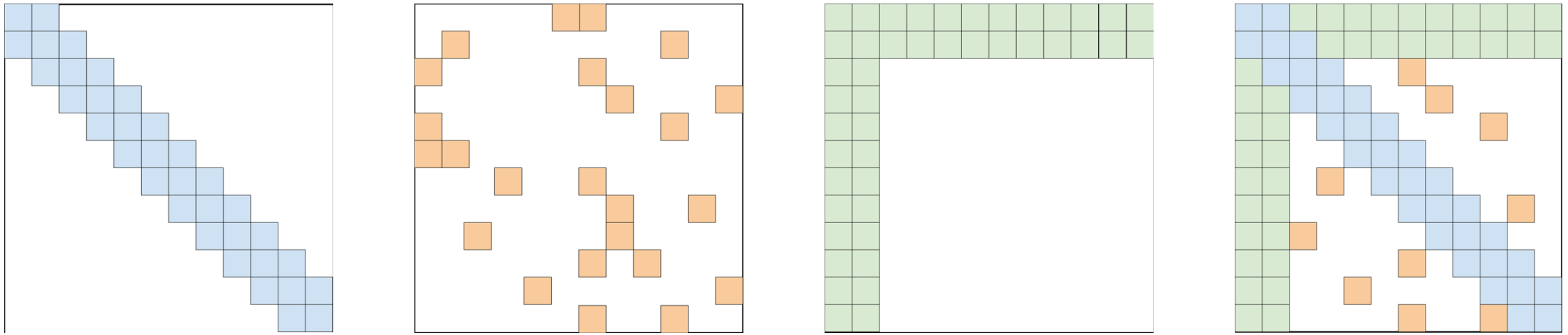
Same dense softmax attention (up to rounding), with quadratic arithmetic.

Image Credit: [12]

Sparse Attention

BigBird combines local, random, and global connections [9].

Local **Random** **Global** **Combined**



Filled entries indicate which token pairs can interact directly.

With fixed local, random, and global budgets, the number of pairs grows linearly with sequence length.

Image Credit: [9], Figure 1

Linear Attention

Original attention for one head [5, 11]

$$S = \frac{QK^\top}{\sqrt{d_k}}, \qquad A = \exp(S) \quad \text{(elementwise)}$$

A contains positive, unnormalized attention weights.

O is the head output: one row per query,
before concatenating heads.

$$O_{i,:} = \frac{\sum_{j=1}^{L} A_{ij} V_{j,:}}{\sum_{j=1}^{L} A_{ij}}, \qquad O \in \mathbb{R}^{L \times d_v}$$

A small example

One query, three scalar values.

$$A_{i,:} = \begin{bmatrix} 2 & 1 & 1 \end{bmatrix},$$

$$V = \begin{bmatrix} 1 & 3 & 5 \end{bmatrix}^\top$$

Total weight: 4

$$O_{i,:} = \frac{2 \cdot 1 + 1 \cdot 3 + 1 \cdot 5}{2 + 1 + 1} = 2.5$$

Linear Attention

Original attention for one head [5, 11]

$$S = \frac{QK^\top}{\sqrt{d_k}}, \qquad A = \exp(S) \quad \text{(elementwise)}$$

A contains positive, unnormalized attention weights.

O is the head output: one row per query,
before concatenating heads.

$$O_{i,:} = \frac{\sum_{j=1}^{L} A_{ij} V_{j,:}}{\sum_{j=1}^{L} A_{ij}}, \qquad O \in \mathbb{R}^{L \times d_v}$$

The diagonal normalizer

$$D = \operatorname{diag}(A\mathbf{1}), \qquad D_{ii} = \sum_{j=1}^{L} A_{ij}$$

$$O = D^{-1}AV = \operatorname{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V$$

A small example

One query, three scalar values.

$$A_{i,:} = \begin{bmatrix} 2 & 1 & 1 \end{bmatrix},$$

$$V = \begin{bmatrix} 1 & 3 & 5 \end{bmatrix}^\top$$

Total weight: 4

$$O_{i,:} = \frac{2 \cdot 1 + 1 \cdot 3 + 1 \cdot 5}{2 + 1 + 1} = 2.5$$

D contains row sums, not learned parameters.
Its inverse divides each row by its total weight.
Here 1 is the all-ones vector.

Linear Attention

Feature weights with the same row normalization [10, 11]

$\widehat{D} = \mathrm{diag}(\widehat{A}\mathbf{1}), \qquad \widehat{O} = \widehat{D}^{-1}\widehat{A}V$

Weighted values divided by their row sums.

$Q' = \phi(Q), \quad K' = \phi(K), \qquad Q', K' \in \mathbb{R}^{L \times r}$

$\widehat{A} = Q'K'^{\top}, \qquad \mathrm{rank}(\widehat{A}) \leq r$

r features per token. The exact exponential matrix can still have full rank.

Linear Transformer [10]

$\phi(x) = \mathrm{ELU}(x) + 1$

Defines a different similarity kernel.

Performer [11]

$$\exp\left(\frac{q_i^{\top} k_j}{\sqrt{d_k}}\right) \approx \phi(q_i)^{\top}\phi(k_j)$$

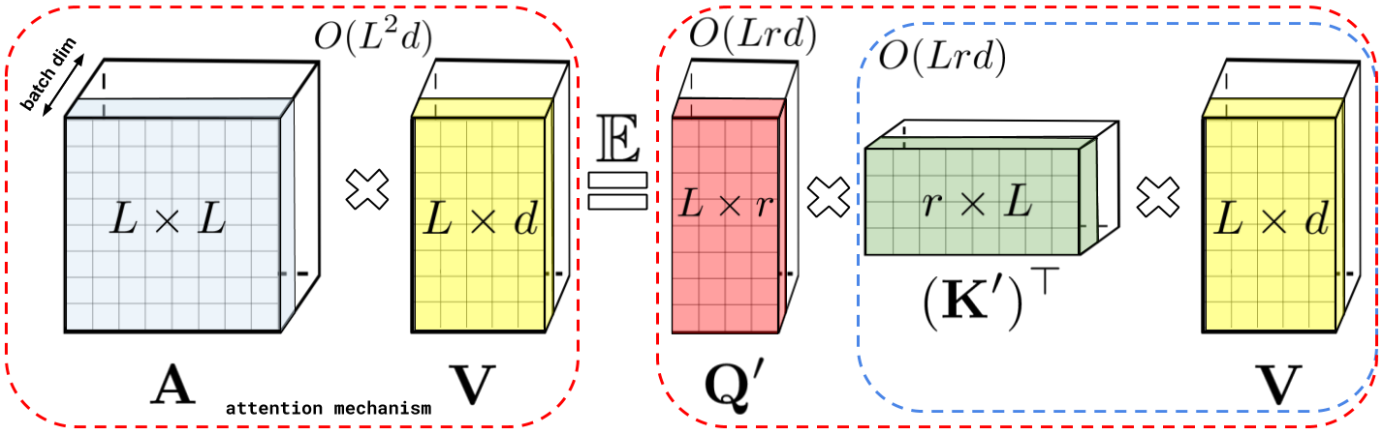
Positive random features approximate the exponential kernel.

The weights and outputs generally change.

Linear Attention

Efficient evaluation of the same attention [10, 11]

$\widehat{D} = \mathrm{diag}(\widehat{A}\mathbf{1}), \qquad \widehat{O} = \widehat{D}^{-1}\widehat{A}V$ Weighted values divided by their row sums.



Weighted values

$$B = K'^{\top}V \in \mathbb{R}^{r \times d_v},$$

$$\widehat{A}V = Q'(K'^{\top}V) = Q'B$$

Row sums

$$z = K'^{\top}\mathbf{1} \in \mathbb{R}^{r},$$

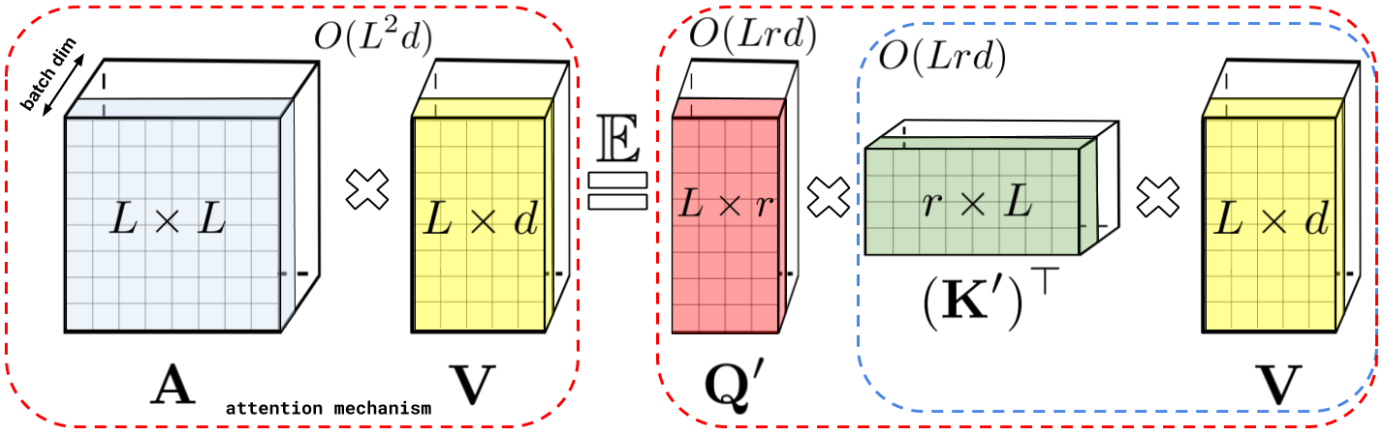
$$\widehat{A}\mathbf{1} = Q'(K'^{\top}\mathbf{1}) = Q'z$$

Image Credit: [11], Fig. 1. Numerator only. E: random-feature expectation; d in figure is value width d_v.

Linear Attention

Efficient evaluation of the same attention [10, 11]

$\widehat{D} = \mathrm{diag}(\widehat{A}\mathbf{1}), \qquad \widehat{O} = \widehat{D}^{-1}\widehat{A}V$ Weighted values divided by their row sums.



Weighted values

$$B = K'^{\top}V \in \mathbb{R}^{r \times d_v},$$

$$\widehat{A}V = Q'(K'^{\top}V) = Q'B$$

Row sums

$$z = K'^{\top}\mathbf{1} \in \mathbb{R}^{r},$$

$$\widehat{A}\mathbf{1} = Q'(K'^{\top}\mathbf{1}) = Q'z$$

$\widehat{D} = \mathrm{diag}(Q'z), \qquad \widehat{O} = \widehat{D}^{-1}Q'B$ Linear work in L at fixed feature/head widths.

Image Credit: [11], Fig. 1. Numerator only. E: random-feature expectation; d in figure is value width d_v.

Outline

- Applications and Challenges of Sequence Modeling
- Transformers
  - Input Embedding & Sinusoidal Positional Encoding
  - Encoder
    - Multi-Head Self-Attention, Residual Connections & LayerNorm
  - Decoder
    - Masked Self-Attention, Cross-Attention
- Limitations & Variants
  - Efficient Attention (FlashAttention, Sparse & Linear)
  - **Rotary Positional Embedding (RoPE)**
  - Normalization (LayerNorm/RMSNorm, Pre-/Post-Norm)
  - Gated MLPs (SwiGLU)
  - Vision Transformers (ViT & Swin)

Absolute vs. Relative Positional Encoding

Relative positions help describe dependencies between tokens.

How to encode relative positions?

- We can inject the relative position into the bias of attention.
- RoPE [13] rotates query and key vectors according to their positions.

To understand RoPE, let us recap how to rotate a 2D vector:

$$\begin{bmatrix} x_1' \\ x_2' \end{bmatrix} = \underbrace{\begin{bmatrix} \cos m\theta & -\sin m\theta \\ \sin m\theta & \cos m\theta \end{bmatrix}}_{\mathbf{R}_{\theta,m}} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Rotation matrix is orthogonal and preserves the norm!

Reference: [13]

Rotary Positional Embedding

Split an even-dimensional vector into 2D pairs; rotate each pair according to its position:

$$\begin{bmatrix} x'_1 \\ \vdots \\ x'_d \end{bmatrix} = \underbrace{\begin{bmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos m\theta_{d/2} & -\sin m\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin m\theta_{d/2} & \cos m\theta_{d/2} \end{bmatrix}}_{\mathbf{R}^d_{\Theta,m}} \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix}$$

Here $\Theta = \{\theta_i = 10000^{-2(i-1)/d}, i \in [1, 2, ..., d/2]\}$

Apply rotations to adjacent pairs: $(x_1, x_2),\ (x_3, x_4),\ \ldots,\ (x_{d-1}, x_d)$

Rotary Positional Embedding

Split an even-dimensional vector into 2D pairs; rotate each pair according to its position:

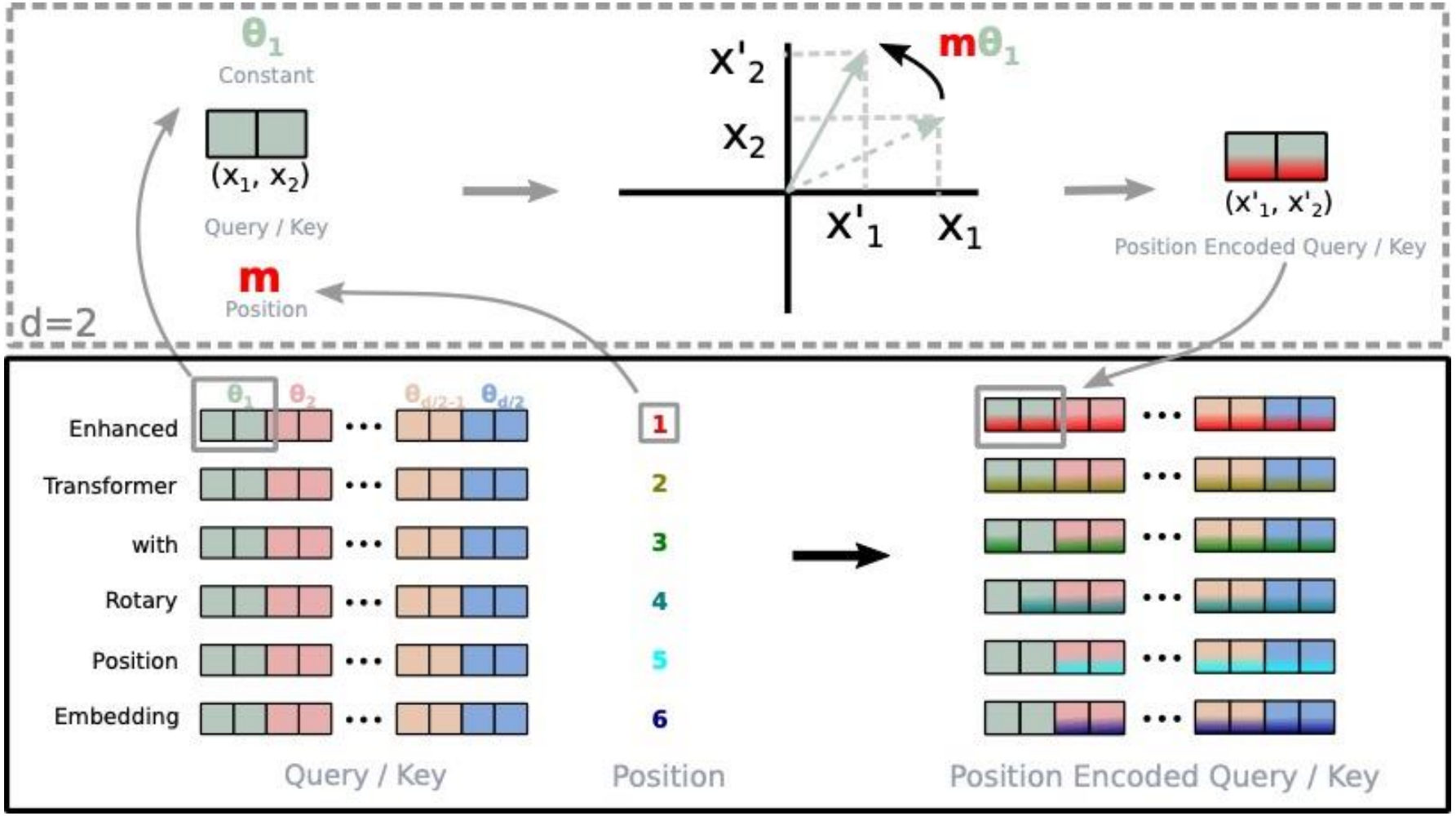


Image Credit: [13]

Rotary Positional Embedding

What do we gain in RoPE?

- Inner product depends on the relative position

Let us look at the case of 2D:

$$\left\langle \begin{bmatrix} x_1' \\ x_2' \end{bmatrix}, \begin{bmatrix} y_1' \\ y_2' \end{bmatrix} \right\rangle = \left\langle \underbrace{\begin{bmatrix} \cos m\theta & -\sin m\theta \\ \sin m\theta & \cos m\theta \end{bmatrix}}_{\mathbf{R}_{\theta,m}} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \underbrace{\begin{bmatrix} \cos n\theta & -\sin n\theta \\ \sin n\theta & \cos n\theta \end{bmatrix}}_{\mathbf{R}_{\theta,n}} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \right\rangle$$

$$= \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^\top \begin{bmatrix} \cos m\theta & -\sin m\theta \\ \sin m\theta & \cos m\theta \end{bmatrix}^\top \begin{bmatrix} \cos n\theta & -\sin n\theta \\ \sin n\theta & \cos n\theta \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

$$= \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^\top \begin{bmatrix} \cos m\theta \cos n\theta + \sin m\theta \sin n\theta & -\cos m\theta \sin n\theta + \sin m\theta \cos n\theta \\ -\sin m\theta \cos n\theta + \cos m\theta \sin n\theta & \sin m\theta \sin n\theta + \cos m\theta \cos n\theta \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

$$= \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^\top \begin{bmatrix} \cos(m-n)\theta & \sin(m-n)\theta \\ \sin(n-m)\theta & \cos(m-n)\theta \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

$$= \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^\top \underbrace{\begin{bmatrix} \cos(m-n)\theta & -\sin(m-n)\theta \\ \sin(m-n)\theta & \cos(m-n)\theta \end{bmatrix}^\top}_{\mathbf{R}_{\theta,m-n}} \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{\mathbf{R}_{\theta,0}} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \left\langle \mathbf{R}_{\theta,m-n} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \mathbf{R}_{\theta,0} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \right\rangle$$

Reference: [13]

Rotary Positional Embedding

What do we gain in RoPE?

- Inner product depends on the relative position

The same identity holds in d dimensions, block by block!

Let us look at the case of 2D:

$$\left\langle \begin{bmatrix} x_1' \\ x_2' \end{bmatrix}, \begin{bmatrix} y_1' \\ y_2' \end{bmatrix} \right\rangle = \left\langle \underbrace{\begin{bmatrix} \cos m\theta & -\sin m\theta \\ \sin m\theta & \cos m\theta \end{bmatrix}}_{\mathbf{R}_{\theta,m}} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \underbrace{\begin{bmatrix} \cos n\theta & -\sin n\theta \\ \sin n\theta & \cos n\theta \end{bmatrix}}_{\mathbf{R}_{\theta,n}} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \right\rangle$$

$$= \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^\top \begin{bmatrix} \cos m\theta & -\sin m\theta \\ \sin m\theta & \cos m\theta \end{bmatrix}^\top \begin{bmatrix} \cos n\theta & -\sin n\theta \\ \sin n\theta & \cos n\theta \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

$$= \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^\top \begin{bmatrix} \cos m\theta \cos n\theta + \sin m\theta \sin n\theta & -\cos m\theta \sin n\theta + \sin m\theta \cos n\theta \\ -\sin m\theta \cos n\theta + \cos m\theta \sin n\theta & \sin m\theta \sin n\theta + \cos m\theta \cos n\theta \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

$$= \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^\top \begin{bmatrix} \cos(m-n)\theta & \sin(m-n)\theta \\ \sin(n-m)\theta & \cos(m-n)\theta \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

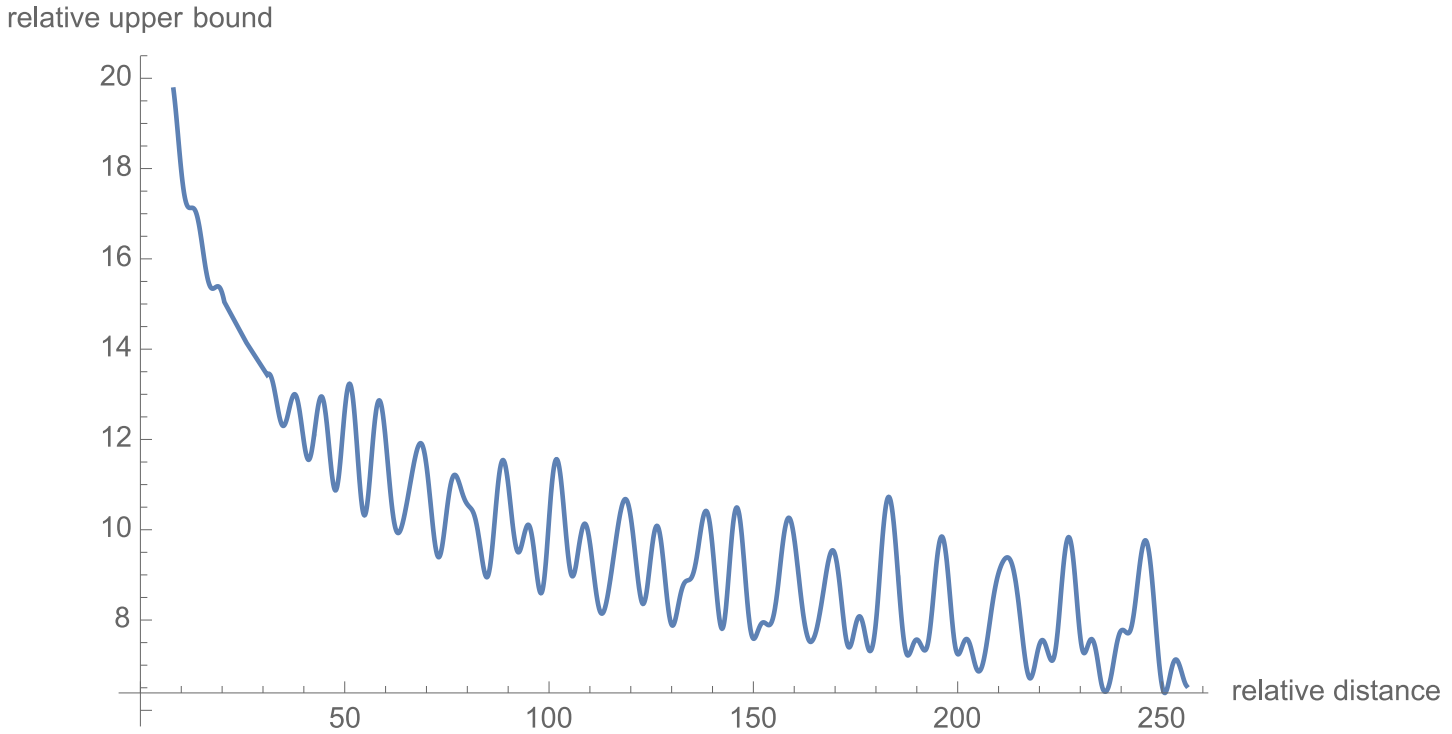
$$= \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^\top \underbrace{\begin{bmatrix} \cos(m-n)\theta & -\sin(m-n)\theta \\ \sin(m-n)\theta & \cos(m-n)\theta \end{bmatrix}^\top}_{\mathbf{R}_{\theta,m-n}} \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{\mathbf{R}_{\theta,0}} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \left\langle \mathbf{R}_{\theta,m-n} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \mathbf{R}_{\theta,0} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \right\rangle$$

Reference: [13]

Rotary Positional Embedding

What does this upper bound describe?

- It caps the magnitude of the query–key dot product after RoPE, before scaling and softmax.



The curve shows the bound's distance factor. Another factor depends on query/key content [13]. Its decay suggests locality, while actual attention weights need not decrease with distance.

Image Credit: [13]

Outline

- Applications and Challenges of Sequence Modeling
- Transformers
  - Input Embedding & Sinusoidal Positional Encoding
  - Encoder
    - Multi-Head Self-Attention, Residual Connections & LayerNorm
  - Decoder
    - Masked Self-Attention, Cross-Attention
- Limitations & Variants
  - Efficient Attention (FlashAttention, Sparse & Linear)
  - Rotary Positional Embedding (RoPE)
  - Normalization (**LayerNorm/RMSNorm**, Pre-/Post-Norm)
  - Gated MLPs (SwiGLU)
  - Vision Transformers (ViT & Swin)

LayerNorm & RMSNorm

Normalization choice: compare the original LayerNorm with RMSNorm [14].

LayerNorm

$$\mu_i = \frac{1}{K}\sum_{k=1}^{K} X_{i,k}$$

$$\sigma_i^2 = \frac{1}{K}\sum_{k=1}^{K}(X_{i,k} - \mu_i)^2$$

$$\mathrm{LN}(X_i)_k = \gamma_k \frac{X_{i,k} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} + \beta_k$$

Subtract mean → Divide by std. → Scale + shift

RMSNorm

$$\mathrm{RMS}_\epsilon(X_i) = \sqrt{\frac{1}{K}\sum_{k=1}^{K} X_{i,k}^2 + \epsilon}$$

$$\mathrm{RMSNorm}(X_i)_k = \gamma_k \frac{X_{i,k}}{\mathrm{RMS}_\epsilon(X_i)}$$

Divide by RMS → Scale

RMSNorm omits mean subtraction and usually the additive bias.
This reduces normalization work while controlling activation scale [14].

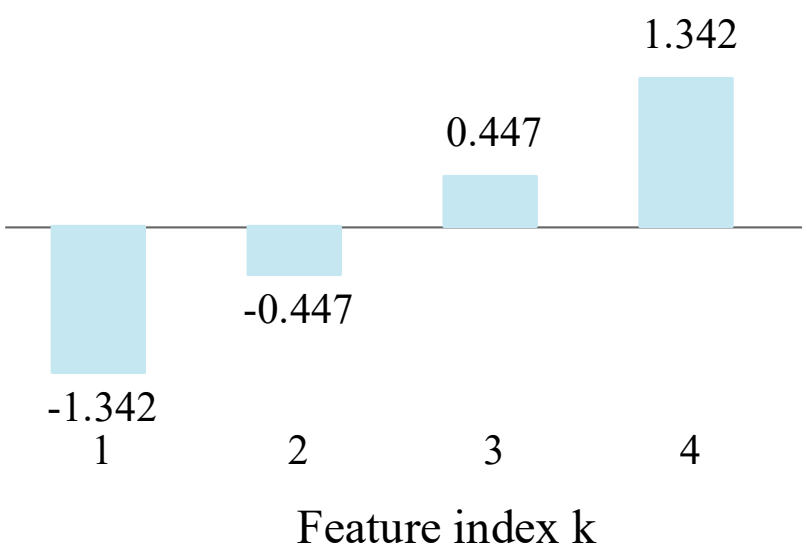
LayerNorm & RMSNorm: An Example

Same token, two different normalizations [14].
In [14], RMSNorm matched translation quality with ~7% less time per training step.

$$X_i = (1, 2, 3, 4), \qquad \gamma_k = 1, \quad \beta_k = 0, \quad \epsilon \approx 0$$

LayerNorm

$$\mu_i = 2.5, \quad \sigma_i^2 = 1.25$$



RMSNorm

$$\mathrm{RMS}(X_i) = \sqrt{7.5} \approx 2.739$$



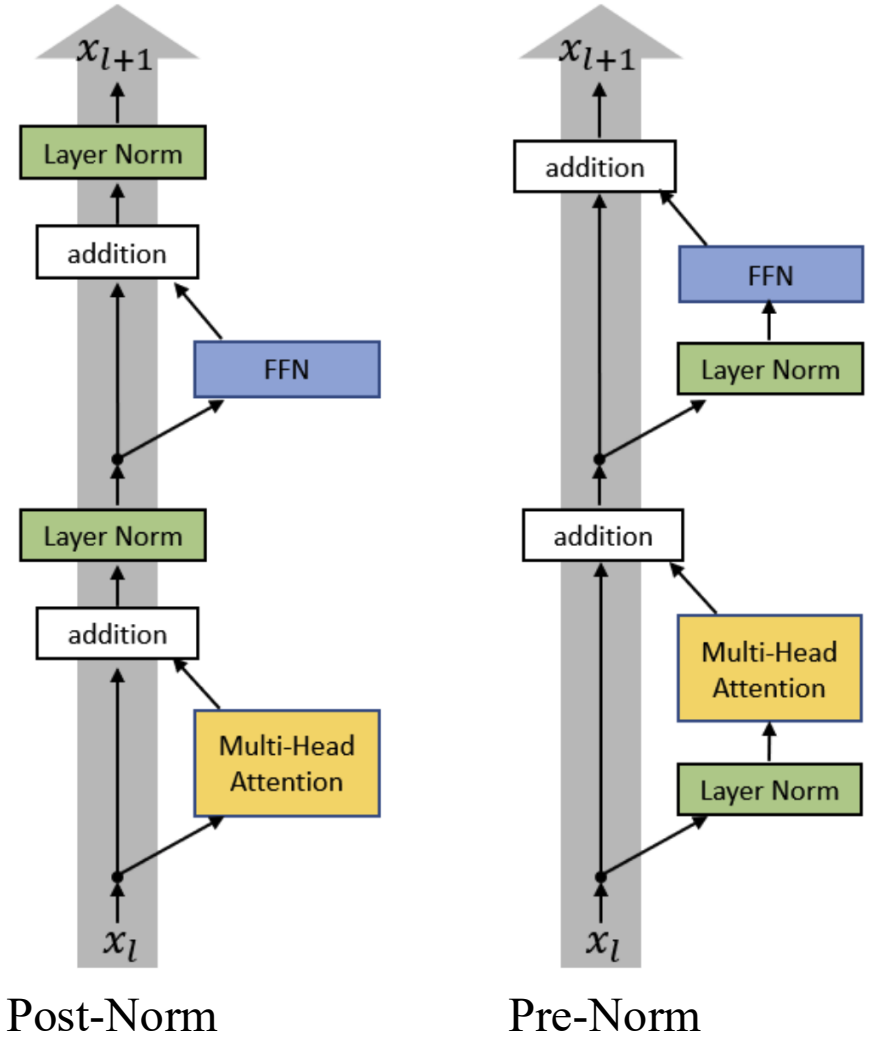
RMSNorm controls magnitude; its output need not have zero mean.

Outline

- Applications and Challenges of Sequence Modeling
- Transformers
  - Input Embedding & Sinusoidal Positional Encoding
  - Encoder
    - Multi-Head Self-Attention, Residual Connections & LayerNorm
  - Decoder
    - Masked Self-Attention, Cross-Attention
- Limitations & Variants
  - Efficient Attention (FlashAttention, Sparse & Linear)
  - Rotary Positional Embedding (RoPE)
  - Normalization (LayerNorm/RMSNorm, **Pre-/Post-Norm**)
  - Gated MLPs (SwiGLU)
  - Vision Transformers (ViT & Swin)

Pre-Norm vs. Post-Norm

Where to place the Layer Normalization?



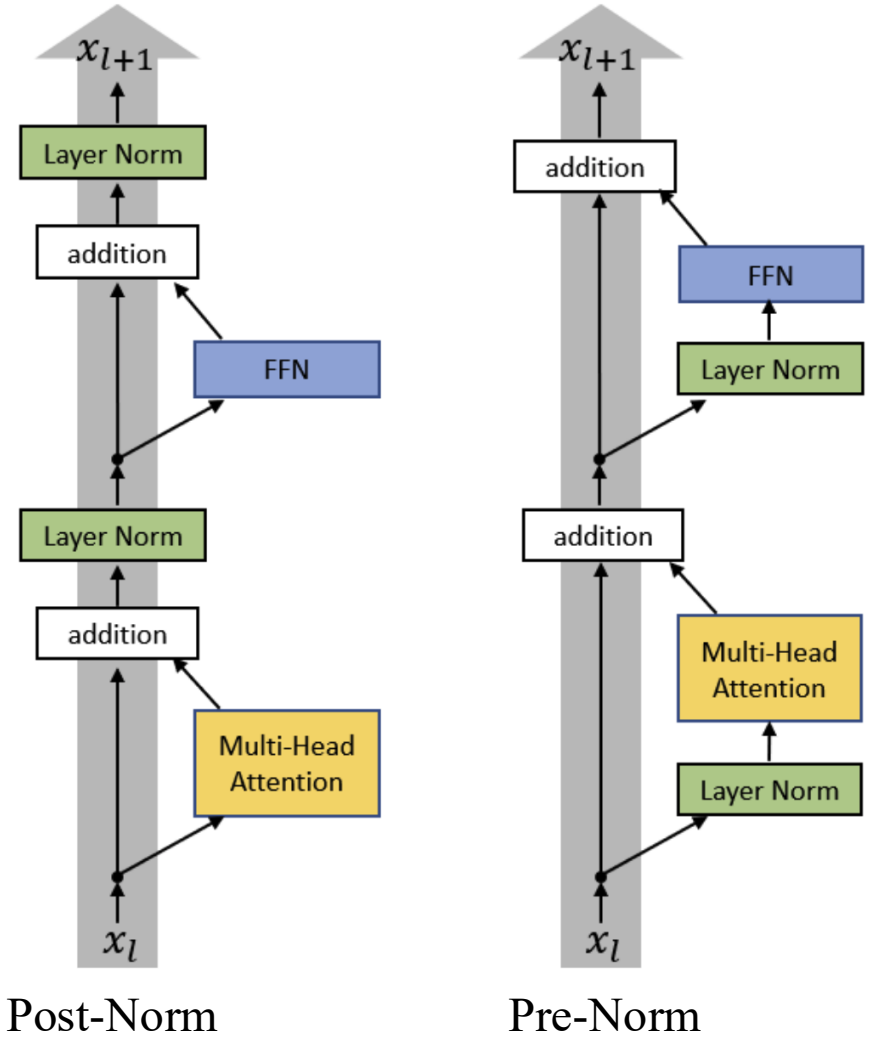
Post-Norm Pre-Norm

Image Credit: [15]

Pre-Norm vs. Post-Norm

Where to place the Layer Normalization?

- Gradient norm in the Post-Norm Transformer is large for parameters near the output and will be likely to decay as the layer gets closer to input



Post-Norm Pre-Norm

Image Credit: [15]

Pre-Norm vs. Post-Norm

Where to place the Layer Normalization?

- Gradient norm in the Post-Norm Transformer is large for parameters near the output and will be likely to decay as the layer gets closer to input

- Pre-norm can stabilize early gradients and reduce the need for warm-up [15].

Pre-norm is a common LLM choice for training stability (e.g., LLaMA [16]).

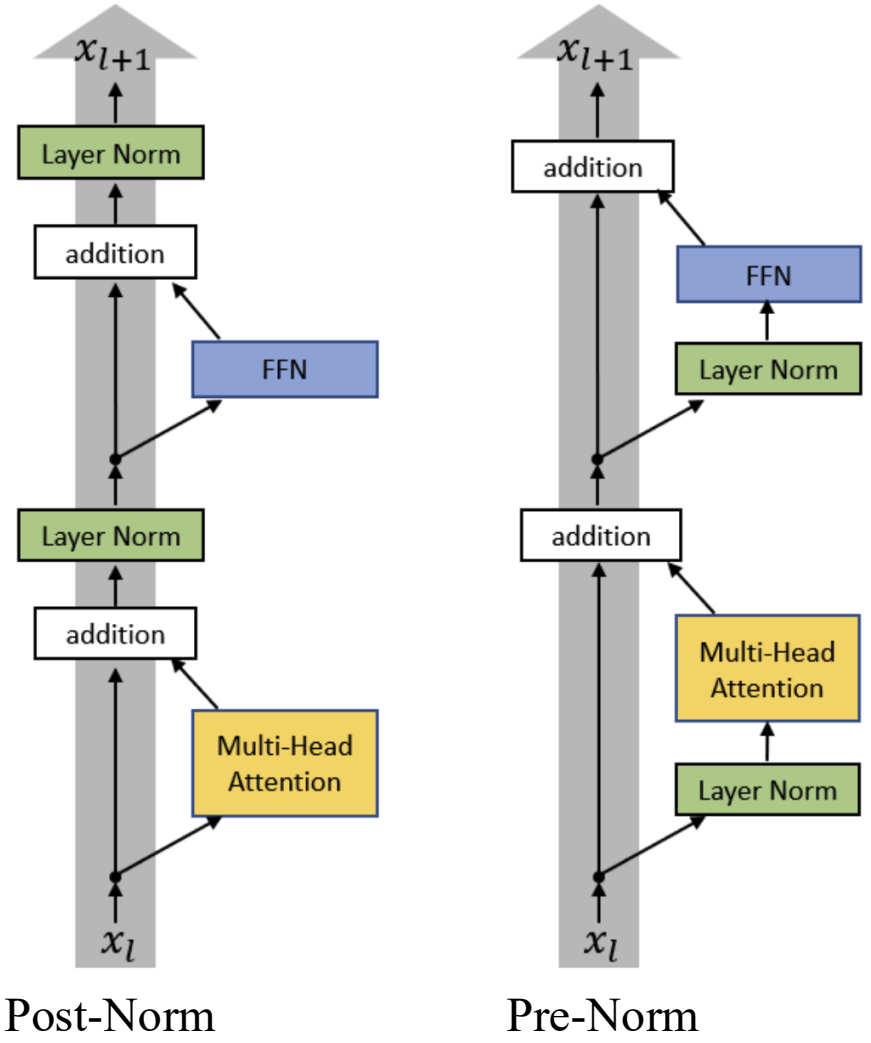


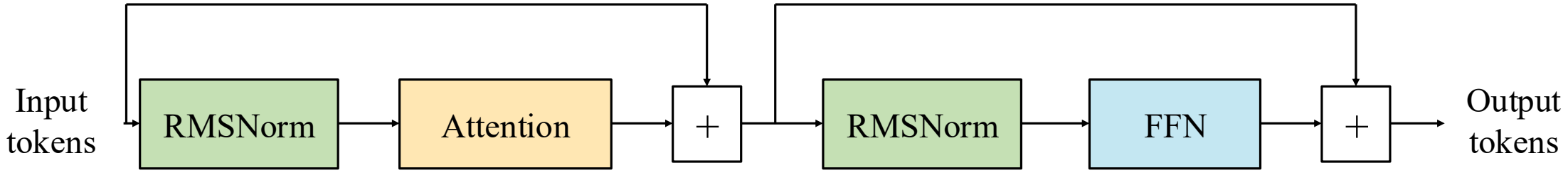
Image Credit: [15]

RMSNorm in a Pre-Norm Block

Norm type and norm placement are separate choices [14, 16].

$$Z = X + \text{Attention}(\text{RMSNorm}(X)),$$

$$Y = Z + \text{FFN}(\text{RMSNorm}(Z)).$$



RMSNorm acts independently on each token; residual paths bypass each sublayer.

LLaMA combines pre-normalization, RMSNorm, and a gated feed-forward sublayer [16].

Outline

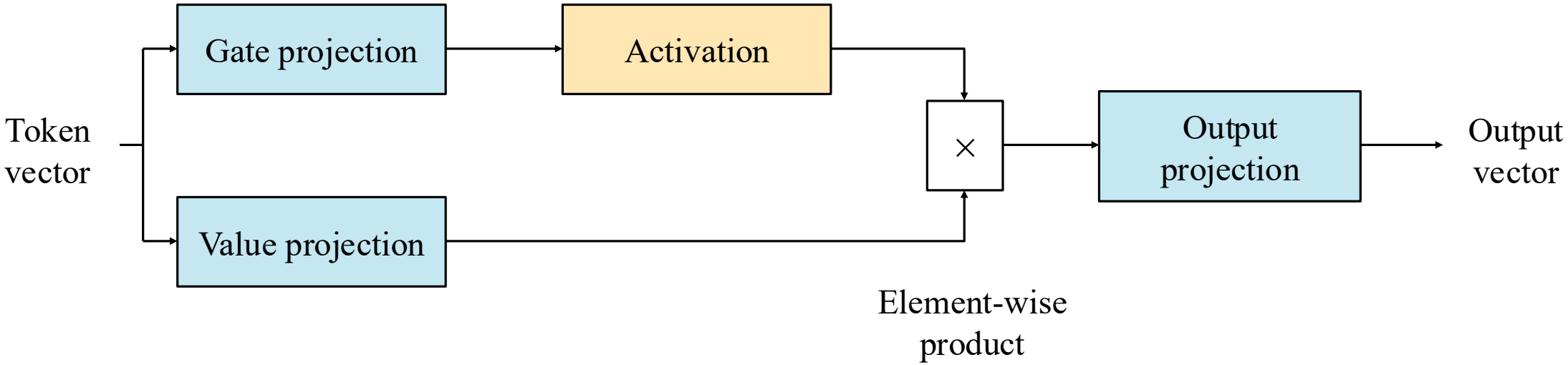
- Applications and Challenges of Sequence Modeling
- Transformers
  - Input Embedding & Sinusoidal Positional Encoding
  - Encoder
    - Multi-Head Self-Attention, Residual Connections & LayerNorm
  - Decoder
    - Masked Self-Attention, Cross-Attention
- Limitations & Variants
  - Efficient Attention (FlashAttention, Sparse & Linear)
  - Rotary Positional Embedding (RoPE)
  - Normalization (LayerNorm/RMSNorm, Pre-/Post-Norm)
  - **Gated MLPs (SwiGLU)**
  - Vision Transformers (ViT & Swin)

Gated MLPs

Two projections form a gate branch and a value branch [17].

$$\mathrm{GatedFFN}(X_i) = \left[\phi(X_i W_g) \odot (X_i W_v)\right] W_o$$

$$W_g, W_v \in \mathbb{R}^{K \times m}, \qquad W_o \in \mathbb{R}^{m \times K}$$



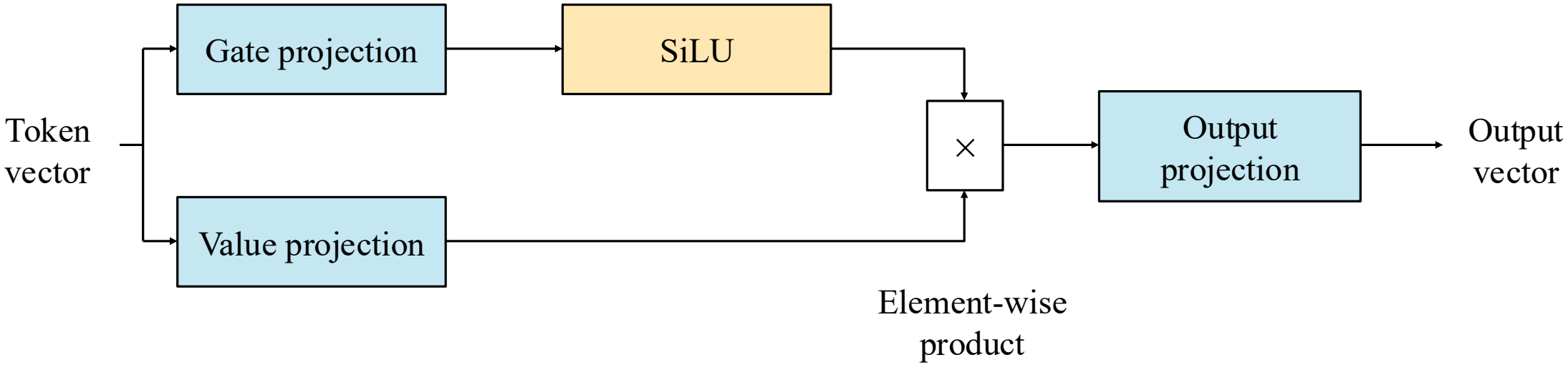
The input controls how each intermediate feature contributes.
This gated MLP replaces the vanilla Transformer's position-wise FFN [17].

Gated MLPs: SwiGLU

SwiGLU uses SiLU on the gate branch [17]; LLaMA adopts this choice [16].

$$\mathrm{SwiGLUFFN}(X_i) = \left[\mathrm{SiLU}(X_i W_g) \odot (X_i W_v)\right] W_o$$

$$W_g, W_v \in \mathbb{R}^{K \times m}, \qquad W_o \in \mathbb{R}^{m \times K} \qquad \mathrm{SiLU}(u) = u \odot \sigma(u)$$



Gates are not probabilities. Other choices: sigmoid → GLU; GELU → GEGLU [17].

SwiGLU/GEGLU gave the best held-out T5 perplexities at matched parameter/compute budgets [17].

Outline

- Applications and Challenges of Sequence Modeling
- Transformers
  - Input Embedding & Sinusoidal Positional Encoding
  - Encoder
    - Multi-Head Self-Attention, Residual Connections & LayerNorm
  - Decoder
    - Masked Self-Attention, Cross-Attention
- Limitations & Variants
  - Efficient Attention (FlashAttention, Sparse & Linear)
  - Rotary Positional Embedding (RoPE)
  - Normalization (LayerNorm/RMSNorm, Pre-/Post-Norm)
  - Gated MLPs (SwiGLU)
  - Vision Transformers (**ViT** & Swin)

Extensions: Vision Transformer

Image Credit: [18, 19]

Vision Transformer: Patches & Computation

Patch size determines the number of image tokens [18].

Patch representation

$$N = \frac{H}{P}\frac{W}{P}, \qquad r = P^2 C$$

Shared linear projection

$$X_p \in \mathbb{R}^{N \times r}, \quad E \in \mathbb{R}^{r \times K},$$

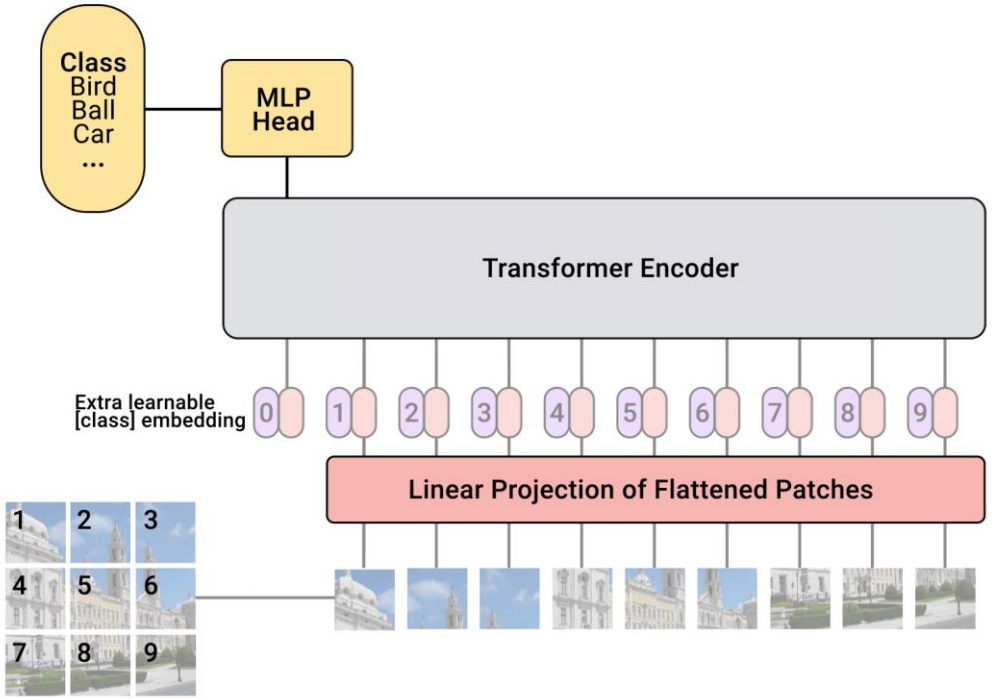
$$X_p E \in \mathbb{R}^{N \times K}$$

Example

$$H = W = 224, \quad P = 16, \quad C = 3,$$

$$N = 14^2 = 196, \quad r = 16^2 \cdot 3 = 768.$$

H, W: image size; P: patch side; C: channels.
K: token features.



Prepend a class token and add learned positional embeddings: 197 tokens.

Image Credit: [18, 19]

Vision Transformer: Patches & Computation

Patch size determines the number of image tokens [18].

Patch representation

$$N = \frac{H}{P}\frac{W}{P}, \qquad r = P^2 C$$

Shared linear projection

$$X_p \in \mathbb{R}^{N \times r}, \quad E \in \mathbb{R}^{r \times K},$$

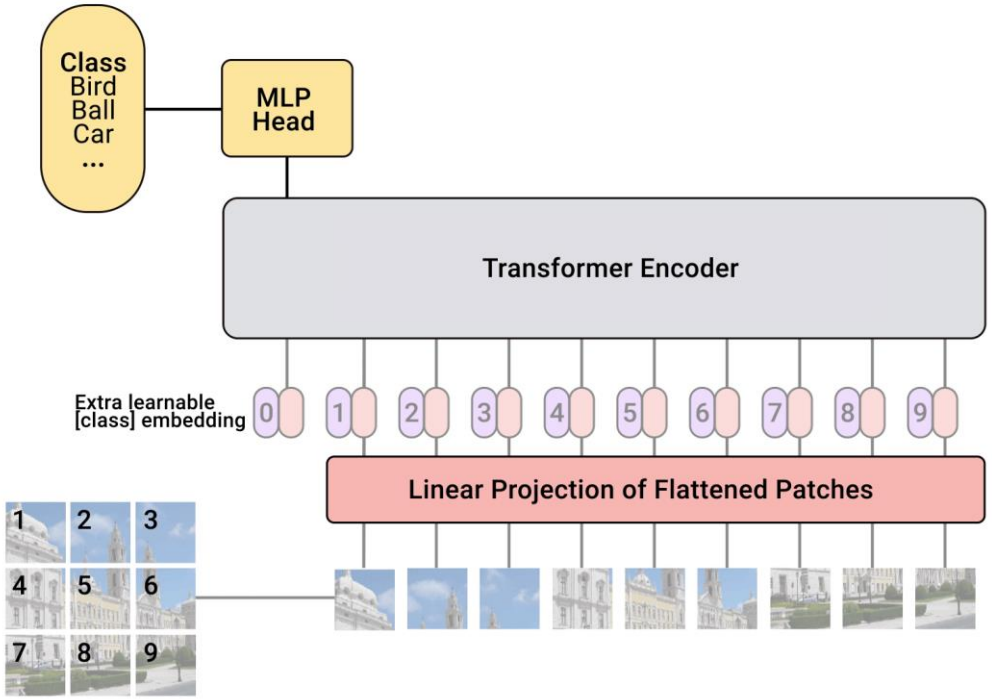
$$X_p E \in \mathbb{R}^{N \times K}$$

Example

$$H = W = 224, \quad P = 16, \quad C = 3,$$

$$N = 14^2 = 196, \quad r = 16^2 \cdot 3 = 768.$$

H, W: image size; P: patch side; C: channels.
K: token features.



Prepend a class token and add learned positional embeddings: 197 tokens.

Each head uses 197 × 197 attention scores. At fixed P, doubling H and W gives about 16× more scores, motivating Swin's local windows [20].

Image Credit: [18, 19]

Outline

- Applications and Challenges of Sequence Modeling
- Transformers
  - Input Embedding & Sinusoidal Positional Encoding
  - Encoder
    - Multi-Head Self-Attention, Residual Connections & LayerNorm
  - Decoder
    - Masked Self-Attention, Cross-Attention
- Limitations & Variants
  - Efficient Attention (FlashAttention, Sparse & Linear)
  - Rotary Positional Embedding (RoPE)
  - Normalization (LayerNorm/RMSNorm, Pre-/Post-Norm)
  - Gated MLPs (SwiGLU)
  - Vision Transformers (ViT & **Swin**)

Extensions: Swin Transformer

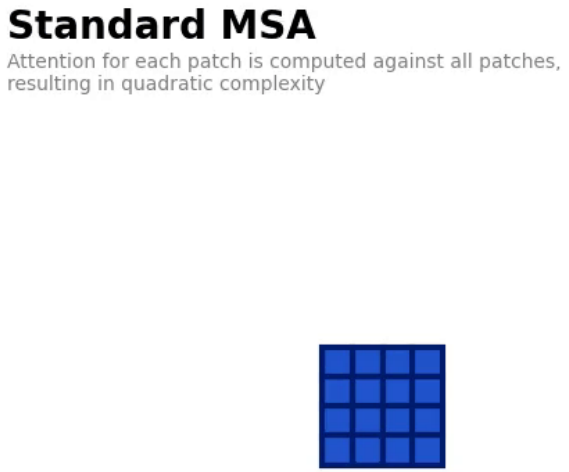
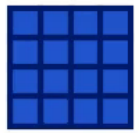


Image Credit: [21]

Extensions: Swin Transformer

Standard MSA

Attention for each patch is computed against all patches, resulting in quadratic complexity



Window-based MSA

Attention for each patch is only computed within its own window (drawn in red). Window size is 2x2 in this example.

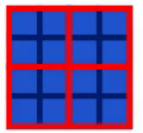


Image Credit: [21]

Extensions: Swin Transformer

Window-based MSA

Attention for each patch is only computed within its own window (drawn in red).
Window size is 2x2 in this example.



Shifted Window MSA

Step 1: Shift window by a factor of M/2, where M = window size
Step 2: For efficient batch computation, move patches into empty slots to create a complete window.
This is known as 'cyclic shift' in the paper.

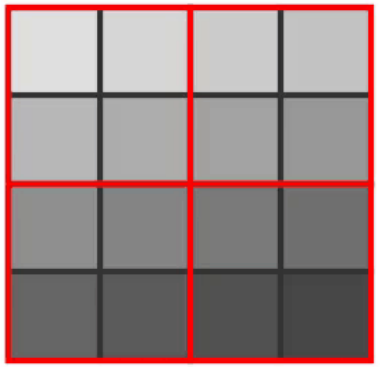
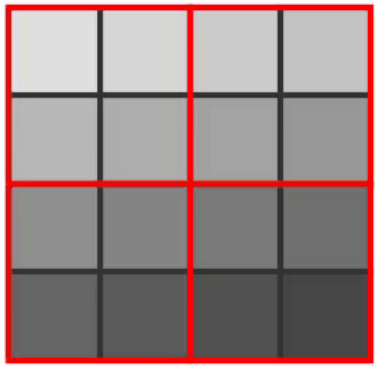


Image Credit: [21]

Extensions: Swin Transformer

Shifted Window MSA

Step 1: Shift window by a factor of M/2, where M = window size
Step 2: For efficient batch computation, move patches into empty slots to create a complete window.
This is known as 'cyclic shift' in the paper.



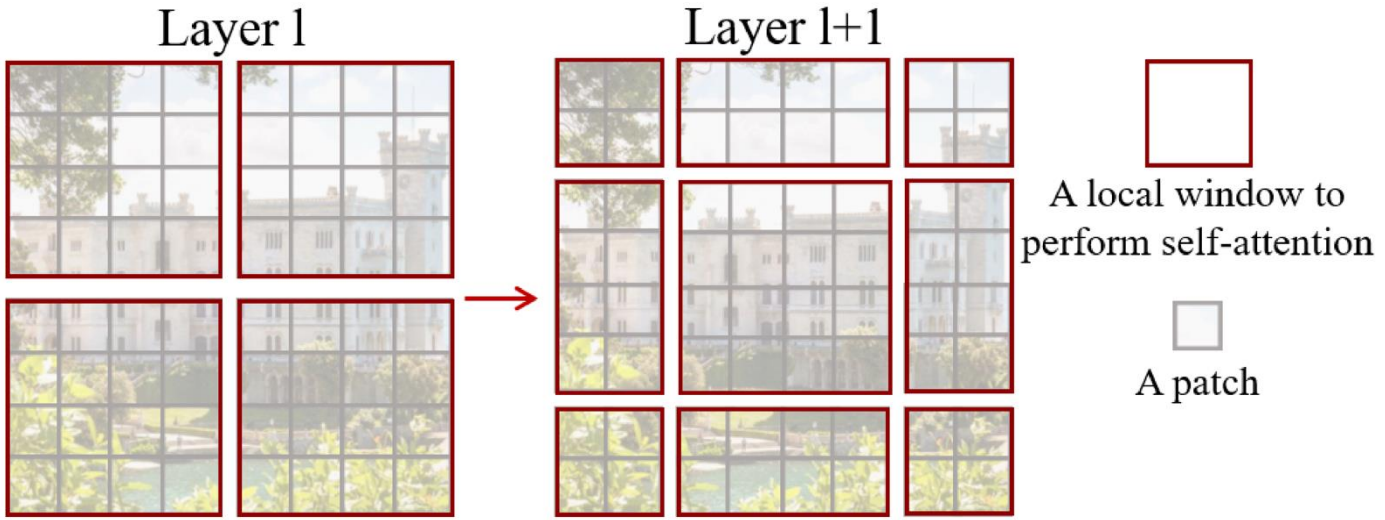
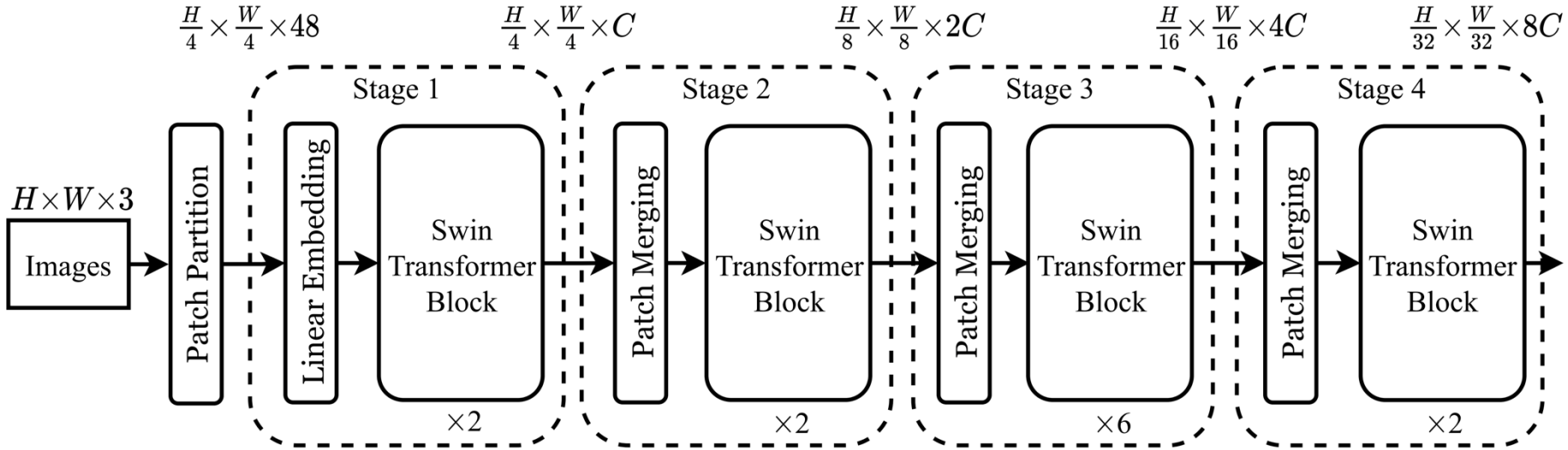


Image Credit: [20, 21]

Extensions: Swin Transformer



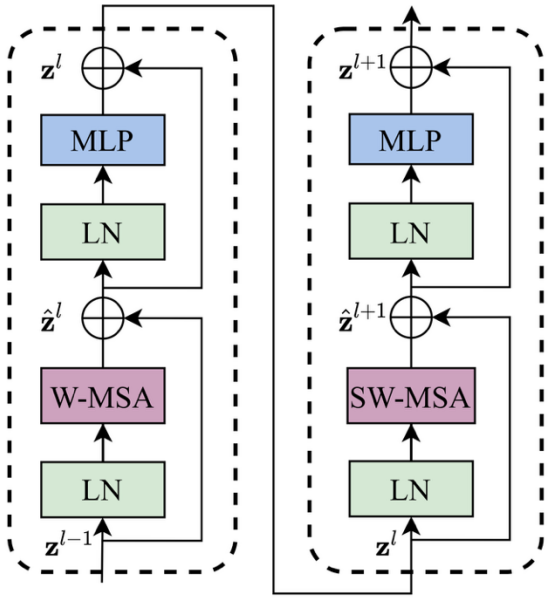


Image Credit: [20]

References

[1] http://web.stanford.edu/class/cs224n/

[2] https://jalammar.github.io/illustrated-transformer/

[3] https://www.mrc-cbu.cam.ac.uk/people/matt.davis/cmabridge

[4] https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

[5] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).

[6] https://lilianweng.github.io/lil-log/2020/04/07/the-transformer-family.html

[7] https://theaisummer.com/transformer/

[8] Dao, T., et al., 2022. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. NeurIPS, 35. [paper]

[9] Zaheer, M., et al., 2020. Big Bird: Transformers for longer sequences. In Advances in Neural Information Processing Systems, 33. [paper]

[10] Katharopoulos, A., Vyas, A., Pappas, N. and Fleuret, F., 2020. Transformers are RNNs: Fast autoregressive Transformers with linear attention. In International Conference on Machine Learning (pp. 5156–5165). PMLR. [paper]

[11] Choromanski, K., et al., 2021. Rethinking attention with Performers. In International Conference on Learning Representations. [paper]

References

[12] https://huggingface.co/docs/text-generation-inference/en/conceptual/flash_attention

[13] Su, J., et al., 2024. RoFormer: Enhanced transformer with rotary position embedding. Neurocomputing, 568, 127063. [paper]

[14] Zhang, B. and Sennrich, R., 2019. Root mean square layer normalization. In Advances in Neural Information Processing Systems, 32. [paper]

[15] Xiong, R., Yang, Y., He, D., Zheng, K., Zheng, S., Xing, C., Zhang, H., Lan, Y., Wang, L. and Liu, T., 2020. On layer normalization in the transformer architecture. In International Conference on Machine Learning (pp. 10524-10533). PMLR.

[16] Touvron, H., et al., 2023. LLaMA: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971. [paper]

[17] Shazeer, N., 2020. GLU variants improve transformer. arXiv preprint arXiv:2002.05202. [paper]

[18] Dosovitskiy, A., et al., 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In International Conference on Learning Representations. [paper]

[19] https://github.com/lucidrains/vit-pytorch

[20] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S. and Guo, B., 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 10012-10022).

[21] https://towardsdatascience.com/a-comprehensive-guide-to-swin-transformer-64965f89d14c

Questions?