EECE 576L: Advanced Topics in Deep Learning

Lecture 2: Invariance, Equivariance, and Deep Learning Models for Sets/Sequences

Renjie Liao

University of British Columbia

2026/27 Winter Session, Term 1 (Fall 2026)

Outline

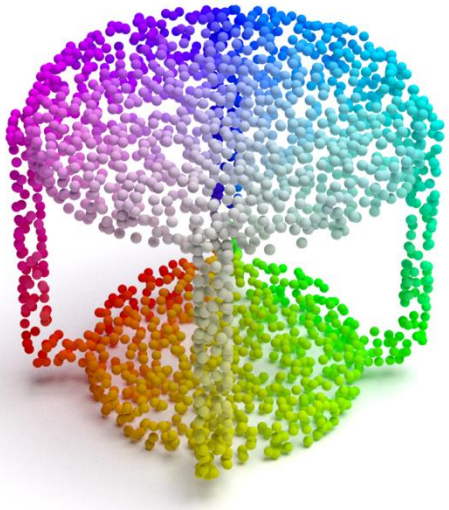
- Invariance & Equivariance Principle
  - Translation equivariance in convolutions
  - Permutation equivariance and invariance

- Models for Sets
  - DeepSets: representation theorem of permutation-invariant set functions & architecture
  - DeepSets: permutation-equivariant linear mapping & architecture

- Models for Sequences
  - Transformers
  - Positional encoding vs. Rotary Positional Embeddings (RoPE)
  - Attention & Flash Attention
  - Pre-norm vs. post-norm
  - Vision Transformers (ViT) & Swin Transformers

Motivating Applications for Sets

- Population Statistics
- Point Cloud Classification



Table

Airplane

Earphone

Image Credit: https://github.com/AnTao97/PointCloudDatasets

Invariance & Equivariance

- Invariance:

  A mathematical object (or a class of mathematical objects) remains unchanged after operations or transformations of a certain type are applied to the objects

$$f(X) = f(g(X))$$

Invariance & Equivariance

- Invariance:

  A mathematical object (or a class of mathematical objects) remains unchanged after operations or transformations of a certain type are applied to the objects

$$f(X) = f(g(X))$$

Invariance & Equivariance

- Invariance:

  A mathematical object (or a class of mathematical objects) remains unchanged after operations or transformations of a certain type are applied to the objects

$$f(X) = f(g(X))$$

- Equivariance:

  Applying a transformation and then computing the function produces the same result as computing the function and then applying the transformation

$$g(f(X)) = f(g(X))$$

Outline

- Invariance & Equivariance Principle
  - **Translation equivariance in convolutions**
  - Permutation equivariance and invariance

- Models for Sets
  - DeepSets: representation theorem of permutation-invariant set functions & architecture
  - DeepSets: permutation-equivariant linear mapping & architecture

- Models for Sequences
  - Transformers
  - Positional encoding vs. Rotary Positional Embeddings (RoPE)
  - Attention & Flash Attention
  - Pre-norm vs. post-norm
  - Vision Transformers (ViT) & Swin Transformers

Revisit Convolution

Matrix multiplication views of (discrete) convolution:

- Filter => Toeplitz matrix
- Data => Toeplitz matrix

Revisit Convolution

Matrix multiplication views of (discrete) convolution:

- Filter => Toeplitz matrix
- Data => Toeplitz matrix

Consider a special Toeplitz matrix: circulant matrix (must be square!)

Convolution with padding

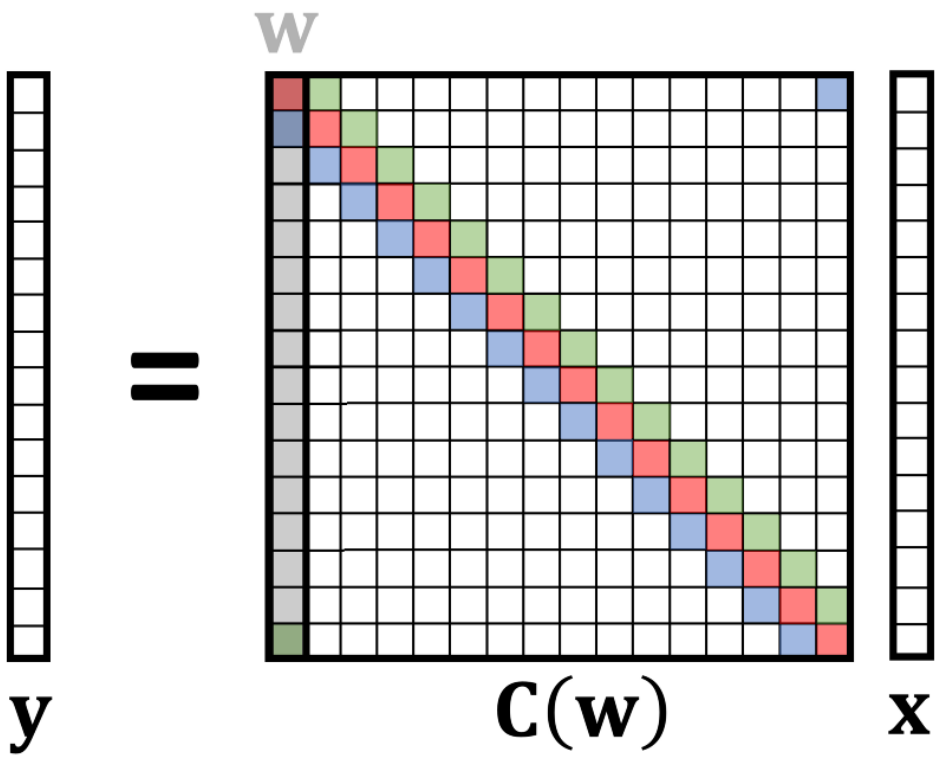


Image Credit: https://towardsdatascience.com/deriving-convolution-from-first-principles-4ff124888028

Translation/Shift Operator

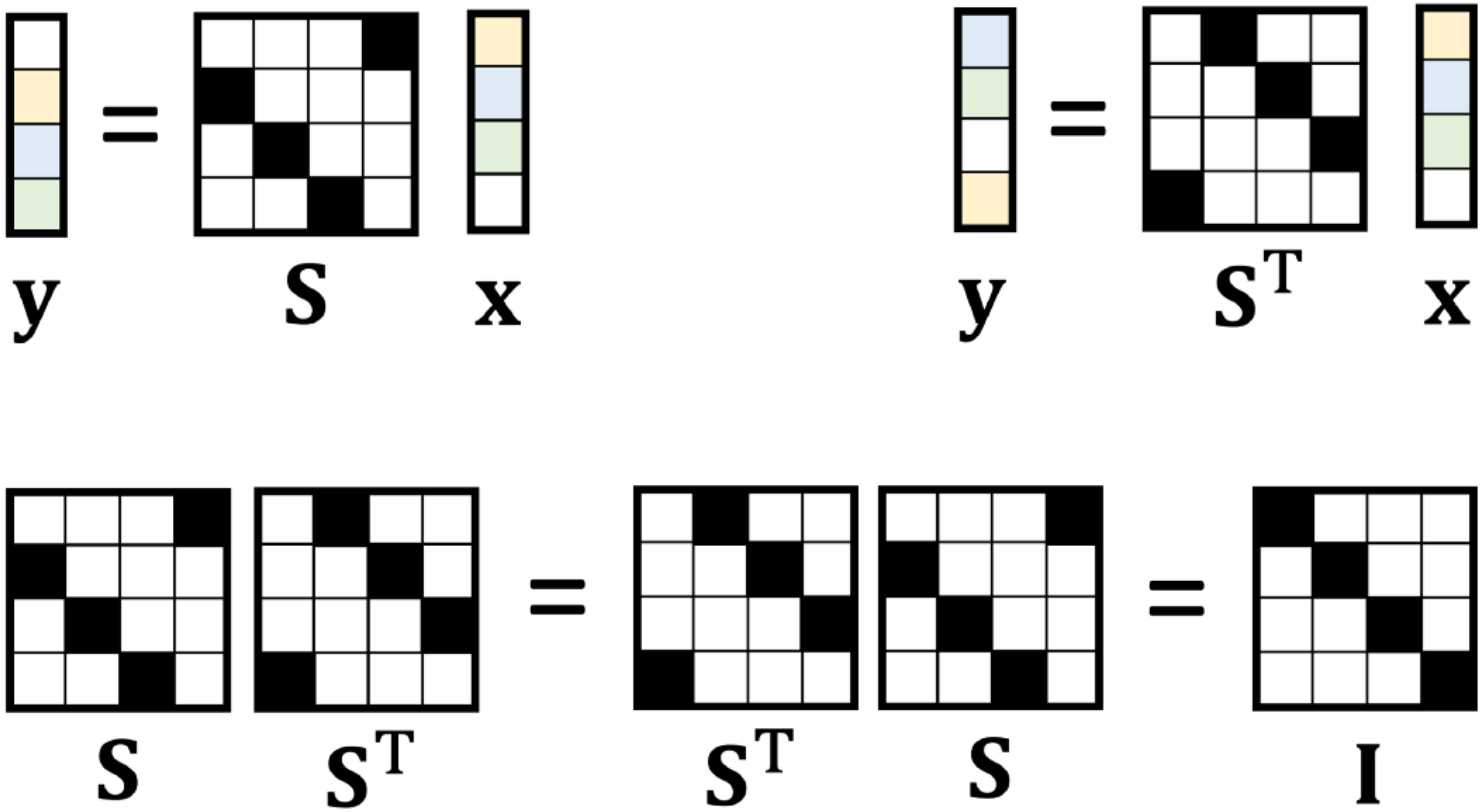
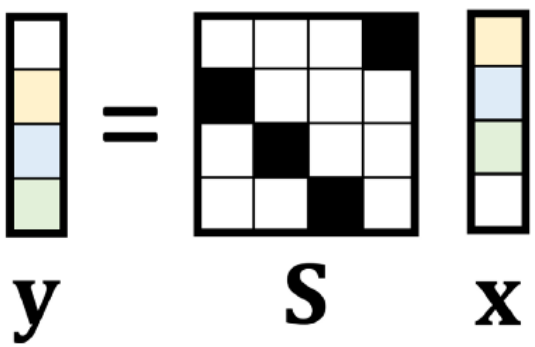
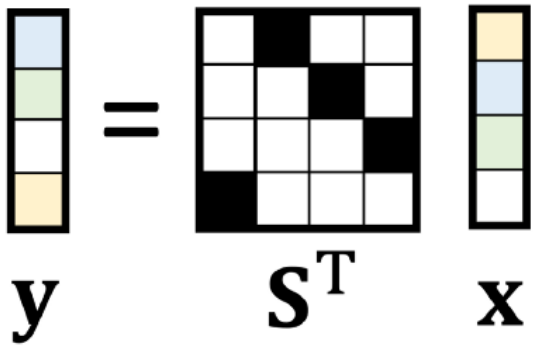


Image Credit: https://towardsdatascience.com/deriving-convolution-from-first-principles-4ff124888028

Translation/Shift Operator

Shift operator is also a circulant matrix!





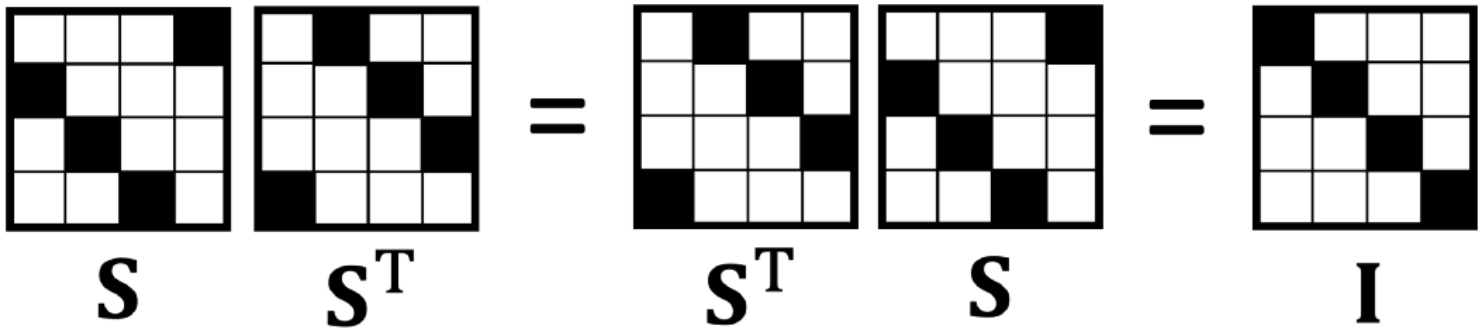


Image Credit: https://towardsdatascience.com/deriving-convolution-from-first-principles-4ff124888028

Translation/Shift Equivariance

Matrix multiplication is non-commutative. But not for circulant matrices!

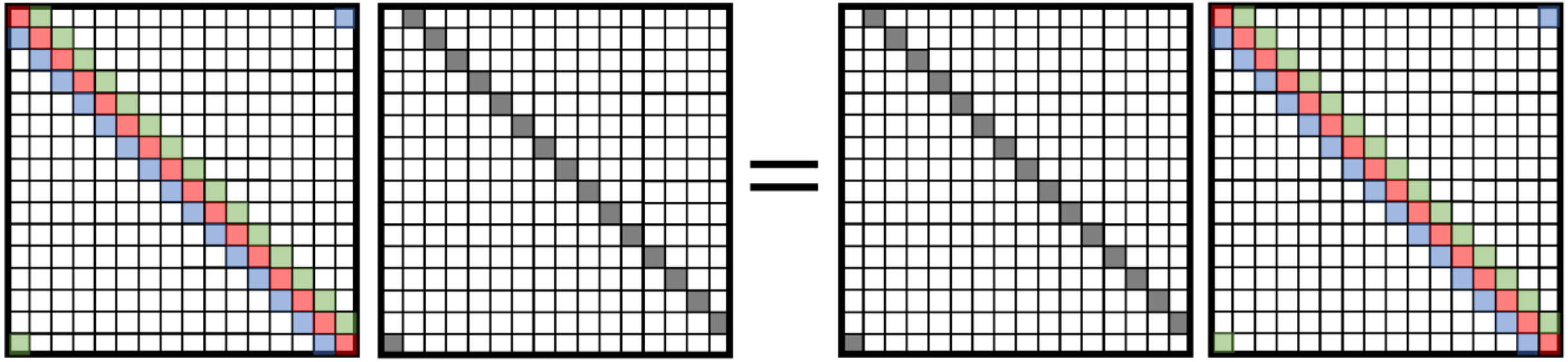
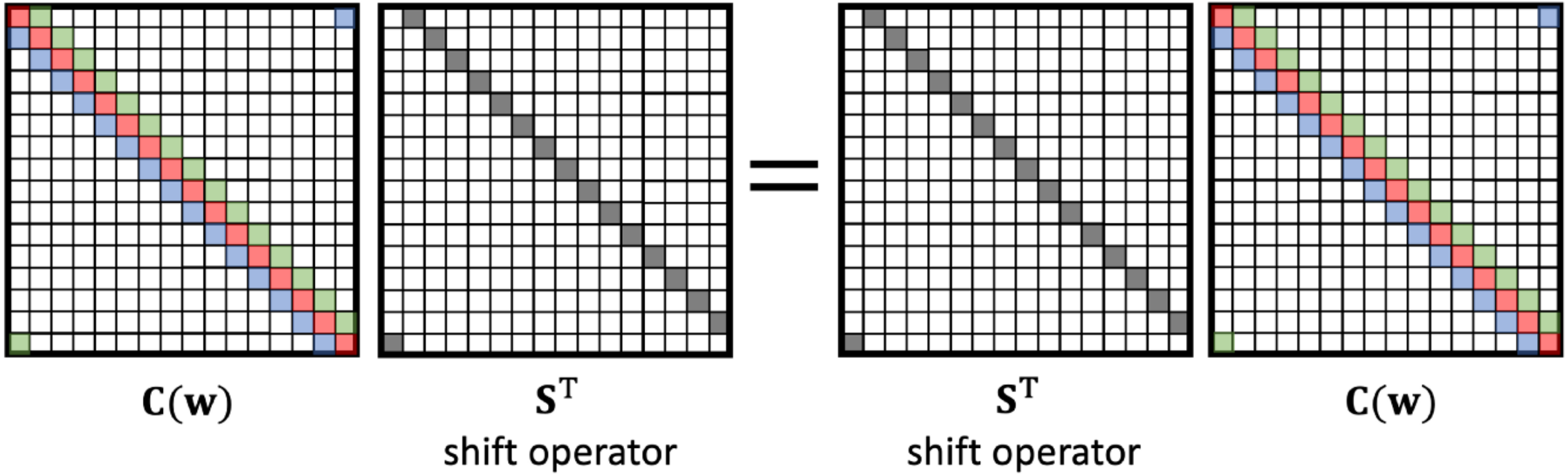


Image Credit: https://towardsdatascience.com/deriving-convolution-from-first-principles-4ff124888028

Translation/Shift Equivariance

Matrix multiplication is non-commutative. But not for circulant matrices!



Convolution is translation equivariant, i.e., Conv(Shift(X)) = Shift(Conv(X))!

Image Credit: https://towardsdatascience.com/deriving-convolution-from-first-principles-4ff124888028

Translation/Shift Invariance

Global pooling gives you shift-invariance!

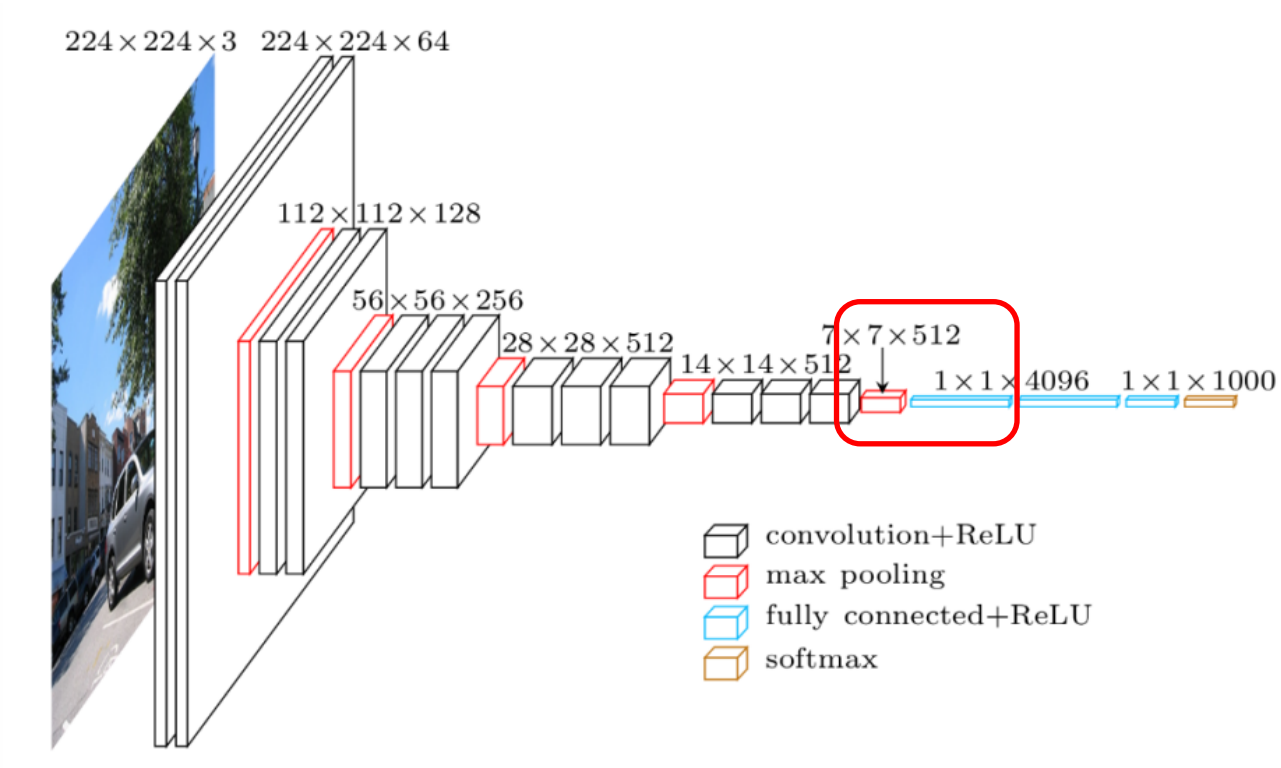
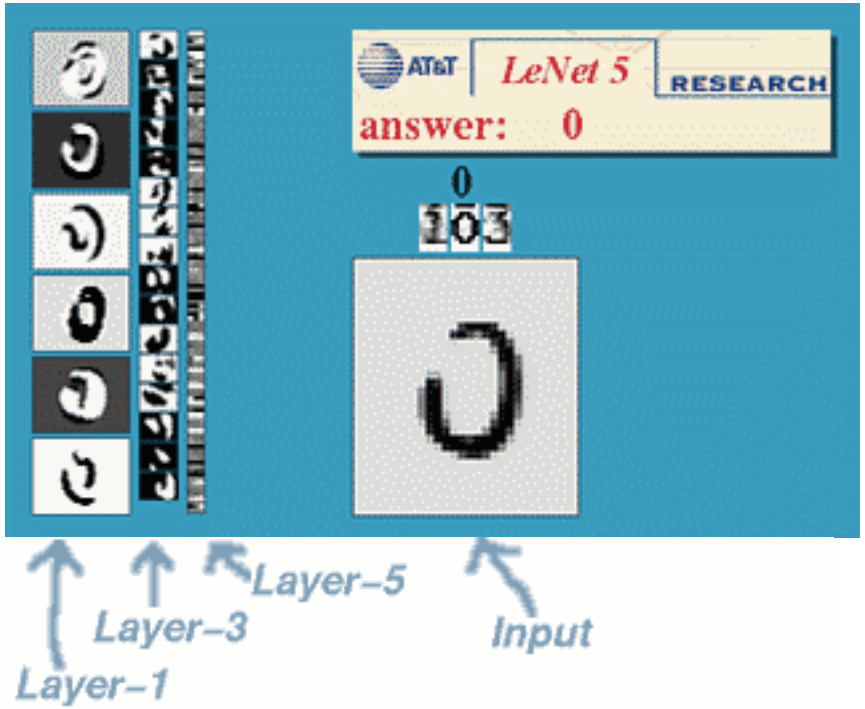


Image Credit: https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529

Translation/Shift Equivariance Invariance

Yann LeCun's LeNet Demo:



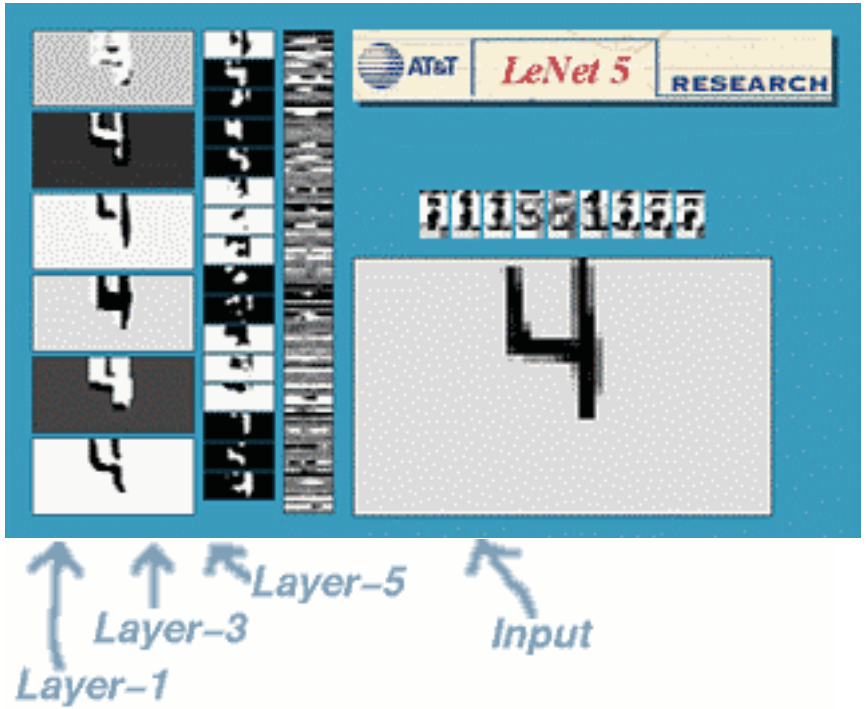


Image Credit: http://yann.lecun.com/exdb/lenet/translation.html

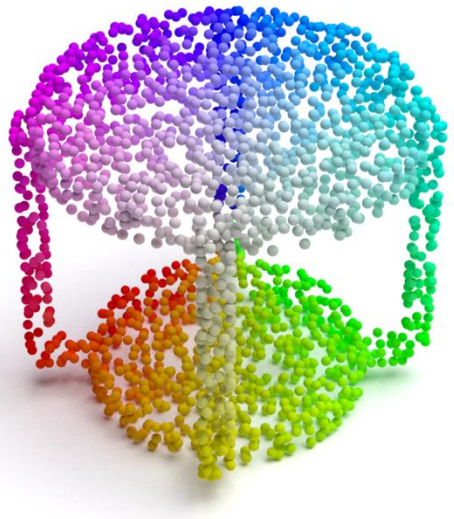
Outline

- Invariance & Equivariance Principle
  - Translation equivariance in convolutions
  - **Permutation equivariance and invariance**

- Models for Sets
  - DeepSets: representation theorem of permutation-invariant set functions & architecture
  - DeepSets: permutation-equivariant linear mapping & architecture

- Models for Sequences
  - Transformers
  - Positional encoding vs. Rotary Positional Embeddings (RoPE)
  - Attention & Flash Attention
  - Pre-norm vs. post-norm
  - Vision Transformers (ViT) & Swin Transformers

Permutation Invariance



Table

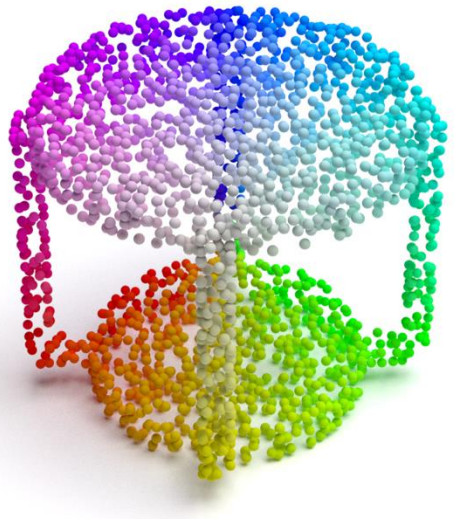
Point Clouds $X \in \mathbb{R}^{n \times 3}$

Probability of Classes $Y \in \mathbb{R}^{1 \times K}$

Permutation / Shuffle $P \in \mathbb{R}^{n \times n}$

Image Credit: https://github.com/AnTao97/PointCloudDatasets

Permutation Invariance



Table

Point Clouds $X \in \mathbb{R}^{n \times 3}$

Probability of Classes $Y \in \mathbb{R}^{1 \times K}$

Permutation / Shuffle $P \in \mathbb{R}^{n \times n}$

$$\begin{bmatrix} 2 \\ 5 \\ 3 \\ 1 \\ 4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix}$$

Image Credit: https://github.com/AnTao97/PointCloudDatasets

Geometric Interpretation of Permutation Matrix

Birkhoff Polytope

$$B_n = \{P \in \mathbb{R}^{n \times n} | \forall i \forall j \ P_{ij} \geq 0, \forall i \ \sum_j P_{ij} = 1, \forall j \ \sum_i P_{ij} = 1\}$$

Doubly Stochastic Matrix

Geometric Interpretation of Permutation Matrix

Birkhoff Polytope

$$B_n = \{P \in \mathbb{R}^{n \times n} | \forall i \forall j \ P_{ij} \geq 0, \forall i \ \sum_j P_{ij} = 1, \forall j \ \sum_i P_{ij} = 1\}$$

Doubly Stochastic Matrix

Birkhoff–von Neumann Theorem:

1. Birkhoff Polytope is the convex hull of permutation matrices
2. Permutation matrices = Vertices of Birkhoff Polytope S_n

Geometric Interpretation of Permutation Matrix

Birkhoff Polytope

$$B_n = \{P \in \mathbb{R}^{n \times n} | \forall i \forall j \ P_{ij} \geq 0, \forall i \ \sum_j P_{ij} = 1, \forall j \ \sum_i P_{ij} = 1\}$$

Doubly Stochastic Matrix

Birkhoff–von Neumann Theorem:

1. Birkhoff Polytope is the convex hull of permutation matrices
2. Permutation matrices = Vertices of Birkhoff Polytope S_n

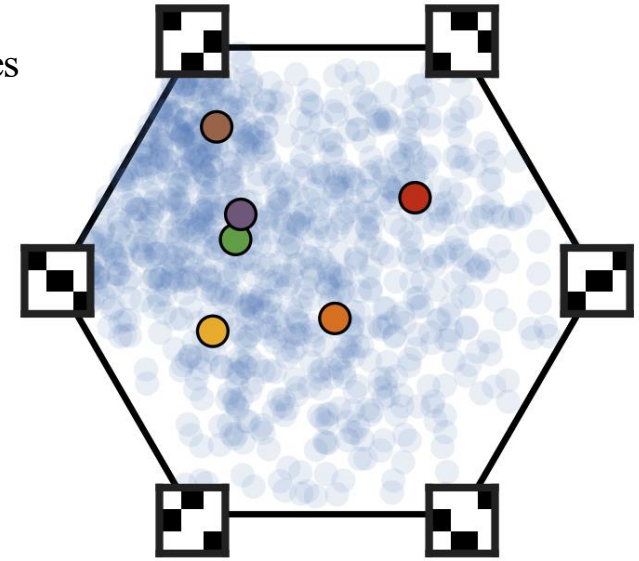
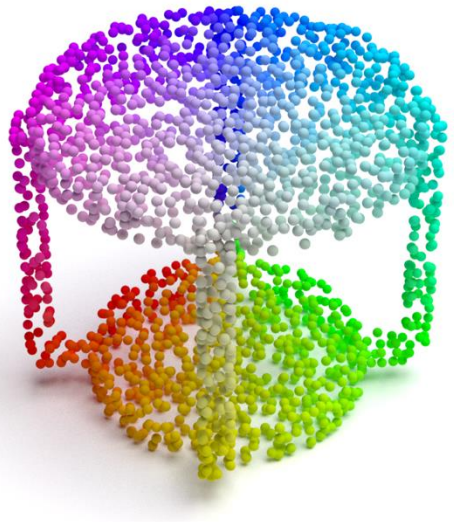


Image Credit: https://arxiv.org/pdf/1710.09508.pdf

Permutation Invariance



Table

Point Clouds $X \in \mathbb{R}^{n \times 3}$

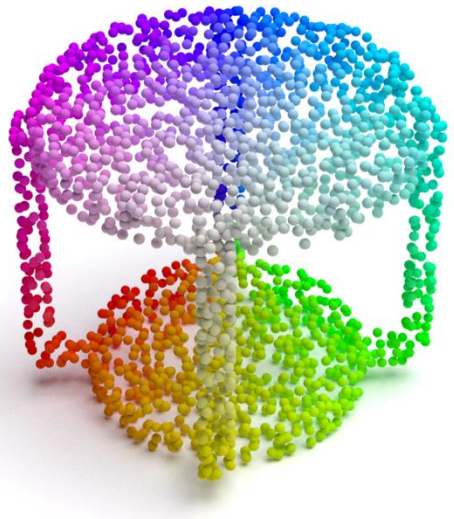
Probability of Classes $Y \in \mathbb{R}^{1 \times K}$

Permutation / Shuffle $P \in \mathbb{R}^{n \times n}$

$$Y = f(PX) \qquad \forall P \in S_n$$

Image Credit: https://github.com/AnTao97/PointCloudDatasets

Permutation Equivariance



Table

Point Clouds $X \in \mathbb{R}^{n \times 3}$

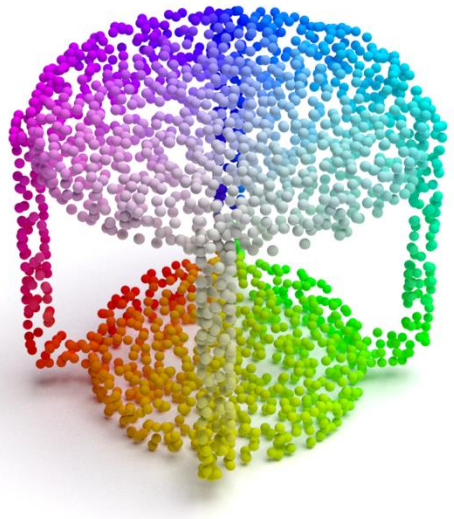
Probability of Classes $Y \in \mathbb{R}^{1 \times K}$

Permutation / Shuffle $P \in \mathbb{R}^{n \times n}$

Point Representations $H \in \mathbb{R}^{n \times d}$

Image Credit: https://github.com/AnTao97/PointCloudDatasets

Permutation Equivariance



Table

Point Clouds $X \in \mathbb{R}^{n \times 3}$

Probability of Classes $Y \in \mathbb{R}^{1 \times K}$

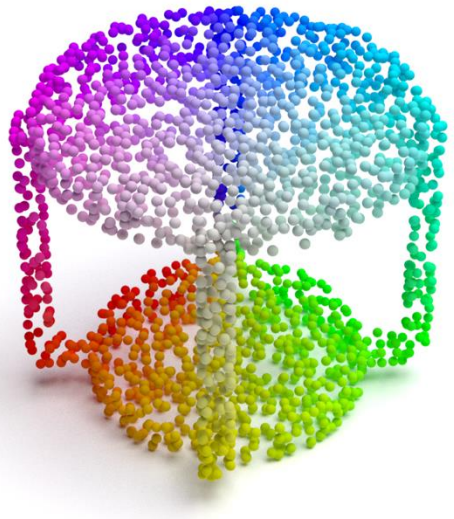
Permutation / Shuffle $P \in \mathbb{R}^{n \times n}$

Point Representations $H \in \mathbb{R}^{n \times d}$

$$H = f(X)$$

Image Credit: https://github.com/AnTao97/PointCloudDatasets

Permutation Equivariance



Table

Point Clouds $X \in \mathbb{R}^{n \times 3}$

Probability of Classes $Y \in \mathbb{R}^{1 \times K}$

Permutation / Shuffle $P \in \mathbb{R}^{n \times n}$

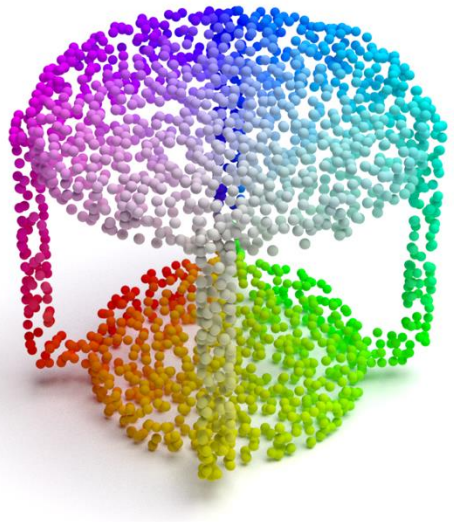
Point Representations $H \in \mathbb{R}^{n \times d}$

$$H = f(X)$$

$$PH = Pf(X) = f(PX)$$

Image Credit: https://github.com/AnTao97/PointCloudDatasets

Permutation Equivariance



Table

Point Clouds $X \in \mathbb{R}^{n \times 3}$

Probability of Classes $Y \in \mathbb{R}^{1 \times K}$

Permutation / Shuffle $P \in \mathbb{R}^{n \times n}$

Point Representations $H \in \mathbb{R}^{n \times d}$

$$H = f(X)$$

$$PH = Pf(X) = f(PX)$$

Image Credit: https://github.com/AnTao97/PointCloudDatasets

More on Invariance & Equivariance

- What about other transformations, e.g., scaling, 2D/3D rotations, Gauge transformation?

Image Credit: http://yann.lecun.com/exdb/lenet/scale.html

More on Invariance & Equivariance

- What about other transformations, e.g., scaling, 2D/3D rotations, Gauge transformation?

- Generalize to Group Invariance & Equivariance

  Recommend Taco Cohen's PhD Thesis: https://pure.uva.nl/ws/files/60770359/Thesis.pdf

Image Credit: http://yann.lecun.com/exdb/lenet/scale.html

Outline

- Invariance & Equivariance Principle
  - Translation equivariance in convolutions
  - Permutation equivariance and invariance

- Models for Sets
  - **DeepSets: representation theorem of permutation-invariant set functions & architecture**
  - DeepSets: permutation-equivariant linear mapping & architecture

- Models for Sequences
  - Transformers
  - Positional encoding vs. Rotary Positional Embeddings (RoPE)
  - Attention & Flash Attention
  - Pre-norm vs. post-norm
  - Vision Transformers (ViT) & Swin Transformers

Deep Learning for Sets

- Point-level Tasks

  Input: a vector per point

  Output: a label/vector per point

  Predictions of individual points are independent, e.g., image classification

Deep Learning for Sets

- Point-level Tasks

  Input: a vector per point

  Output: a label/vector per point

  Predictions of individual points are independent, e.g., image classification

- Set-level Tasks

  Input: a set of vectors, each corresponds to a point

  Output: a label/vector per set

  Prediction of a set depends on all points, e.g., point cloud classification

Deep Learning for Sets

Key Challenges:

- Varying-sized input sets
- Permutation equivariant and invariant models
- Expressive models

Deep Learning for Sets

Deep Sets: invariant representation theorem [1]

Theorem 2 (countable universe)

For ordinary sets drawn from a countable universe,

$f : 2^{\mathfrak{X}} \to \mathbb{R}$ is permutation invariant if and only if

suitable element and output maps give the representation

$$f(X) = \rho\left(\sum_{x \in X} \phi(x)\right)$$

Assumptions: no repeated elements. No continuity requirement.

[1] Zaheer et al., Deep Sets, NeurIPS 2017.

Deep Learning for Sets

Proof sketch: a unique sum code

1. Encode the set
2. Prove uniqueness
3. Recover the output

Choose a unique index for each element.

$$c : \mathfrak{X} \hookrightarrow \mathbb{N}_{>0}$$

$$\phi(x) = 4^{-c(x)}$$

$$E(X) = \sum_{x \in X} 4^{-c(x)}$$

Each digit records membership

j	1	2	3	4	5	$\cdots$
X	1	0	1	0	0	$\cdots$

$$X = \{x_1, x_3\}$$

$$E(X) = (0.10100\ldots)_4$$

[1] Zaheer et al., Deep Sets, NeurIPS 2017.

Deep Learning for Sets

Proof sketch: why the code is unique

1. Encode the set
2. Prove uniqueness
3. Recover the output

For distinct sets, choose their first differing index k.

Earlier contributions cancel.
The entire later tail is too small to cancel this first difference.

First different membership

j	1	2	3	4	5	$\cdots$
X	1	0	1	0	1	$\cdots$
Y	1	0	0	1	0	$\cdots$

$$|E(X) - E(Y)| \geq 4^{-k} - \sum_{j>k} 4^{-j} = \frac{2}{3} 4^{-k} > 0$$

[1] Zaheer et al., Deep Sets, NeurIPS 2017.

Deep Learning for Sets

Proof sketch: recovering the output

1. Encode the set
2. Prove uniqueness
3. Recover the output

A unique code lets us recover the set and then apply its target function.

$$X \xrightarrow{E} E(X) \xrightarrow{E^{-1}} X \xrightarrow{f} f(X)$$

$$\rho = f \circ E^{-1}, \qquad f(X) = \rho(E(X))$$

The inverse is defined on the encoder's image.

Converse: summation is independent of element order.

Continuous neural approximation requires additional assumptions.

[1] Zaheer et al., Deep Sets, NeurIPS 2017.

Deep Learning for Sets

- Deep Sets [1]

Invariant Architecture

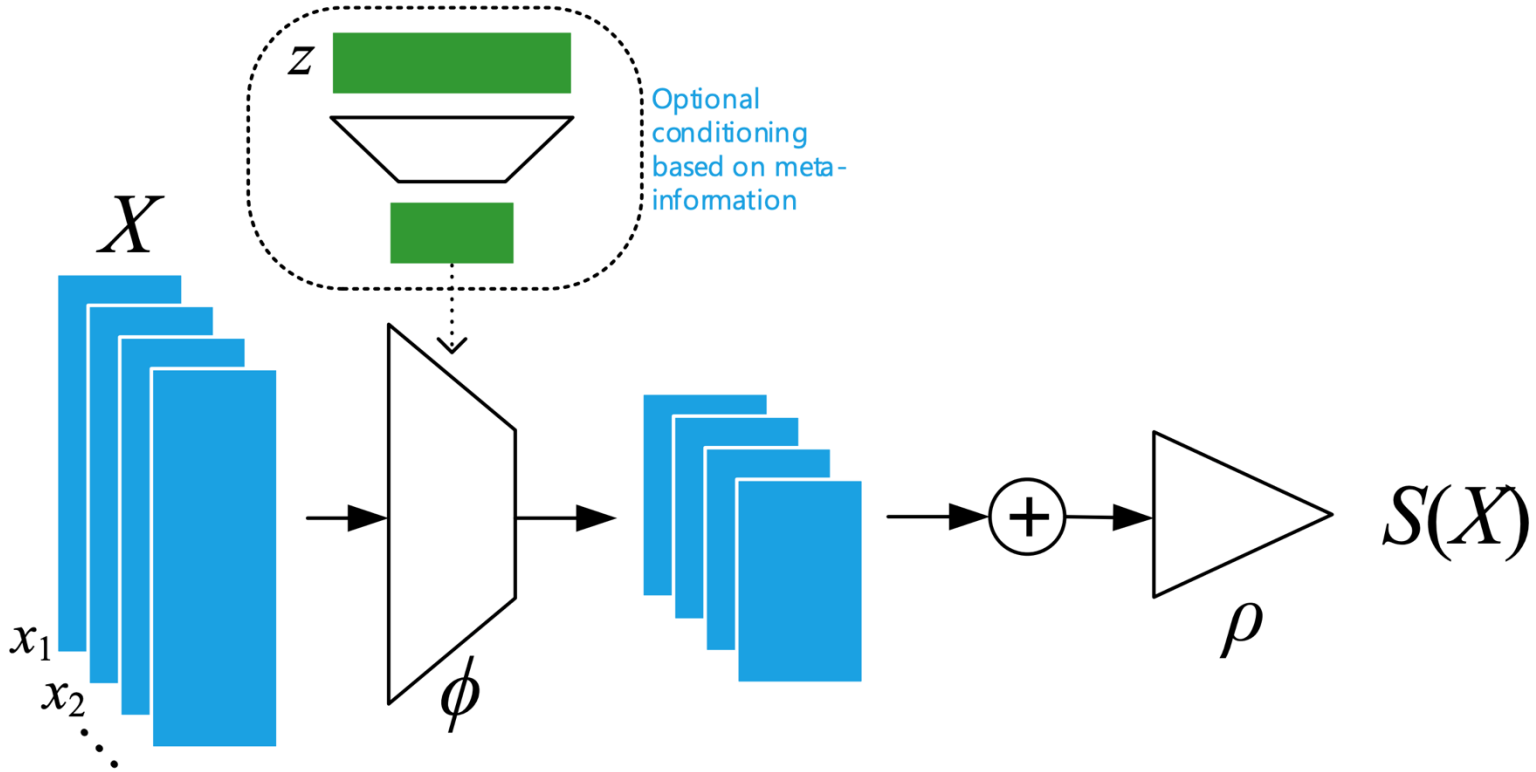


Image Credit: [1] Zaheer, Manzil, et al. "Deep sets." Advances in neural information processing systems 30 (2017).

Outline

- Invariance & Equivariance Principle
  - Translation equivariance in convolutions
  - Permutation equivariance and invariance

- Models for Sets
  - DeepSets: representation theorem of permutation-invariant set functions & architecture
  - **DeepSets: permutation-equivariant linear mapping & architecture**

- Models for Sequences
  - Transformers
  - Positional encoding vs. Rotary Positional Embeddings (RoPE)
  - Attention & Flash Attention
  - Pre-norm vs. post-norm
  - Vision Transformers (ViT) & Swin Transformers

Deep Learning for Sets

Deep Sets: equivariant layer theorem [1]

$$f_\Theta(x) = \sigma(\Theta x), \qquad x \in \mathbb{R}^M, \quad \Theta \in \mathbb{R}^{M \times M}$$

Lemma 3: the layer is permutation equivariant if and only if

$$\Theta = \lambda I + \gamma \mathbf{1}\mathbf{1}^\top, \qquad \lambda, \gamma \in \mathbb{R}$$

Assumption: shared, elementwise, injective activation (e.g., sigmoid or tanh).

Inputs range over all real vectors. The result covers the linear layer too.

Proof roadmap: necessity and sufficiency

Necessity (1–2): equivariance forces the stated matrix form.
Sufficiency (3): this matrix form guarantees equivariance.

[1] Zaheer et al., Deep Sets, NeurIPS 2017.

Deep Learning for Sets

Necessity: equivariance constrains the weight matrix

1. Equivariance constrains the weight matrix

2. The constraint forces two shared weight values

3. This form guarantees equivariance

Equivariance and injectivity give

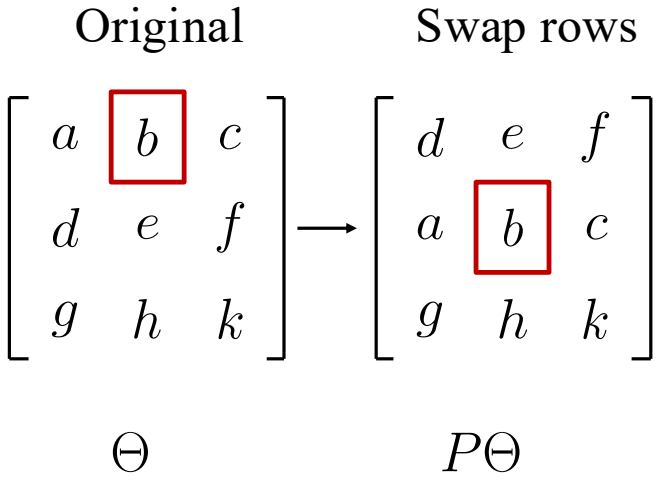
$$\Theta P x = P \Theta x \quad \forall x,\ P$$

$$P \Theta P^{\top} = \Theta$$

P is any permutation matrix.

It relabels rows and columns.

Example: swap indices 1 and 2



Left multiplication swaps rows. Next, swap the matching columns.

[1] Zaheer et al., Deep Sets, NeurIPS 2017.

Deep Learning for Sets

Necessity: equivariance constrains the weight matrix

1. Equivariance constrains the weight matrix

2. The constraint forces two shared weight values

3. This form guarantees equivariance

Equivariance and injectivity give

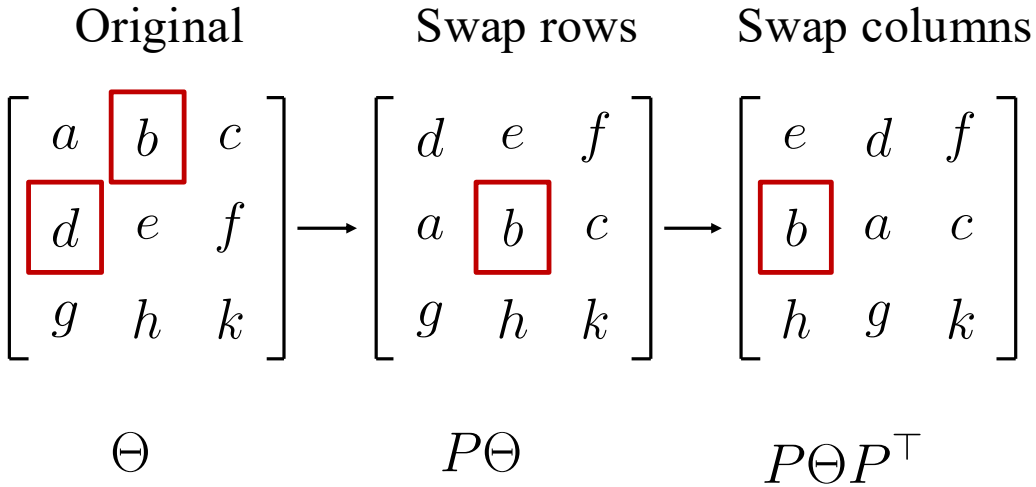
$$\Theta P x = P \Theta x \quad \forall x,\ P$$

$$P \Theta P^\top = \Theta$$

P is any permutation matrix.

It relabels rows and columns.

Example: swap indices 1 and 2



Matching the original and final entries gives $b = d$.

[1] Zaheer et al., Deep Sets, NeurIPS 2017.

Deep Learning for Sets

Necessity: one diagonal and one off-diagonal value

1. Equivariance constrains the weight matrix

2. The constraint forces two shared weight values

3. This form guarantees equivariance

Any diagonal position can reach any other.
All diagonal entries share one value.

Any ordered pair of distinct indices can reach any other. All off-diagonal entries share one value.

$$\Theta_{\pi(i),\pi(j)} = \Theta_{ij}$$

The resulting pattern

$$\begin{bmatrix} d & c & c \\ c & d & c \\ c & c & d \end{bmatrix}$$

$$\Theta = (d-c)I + c\mathbf{1}\mathbf{1}^\top$$

[1] Zaheer et al., Deep Sets, NeurIPS 2017.

Deep Learning for Sets

Sufficiency: the stated form guarantees equivariance

1. Equivariance constrains the weight matrix

2. The constraint forces two shared weight values

3. This form guarantees equivariance

Simultaneous relabeling leaves both matrix components unchanged.

$$PIP^\top = I, \qquad P(\mathbf{1}\mathbf{1}^\top)P^\top = \mathbf{1}\mathbf{1}^\top$$

A shared pointwise activation preserves equivariance.

$$f_\Theta(x)_i = \sigma\left(\lambda x_i + \gamma \sum_{j=1}^{M} x_j\right)$$

Each output combines its own input with the same set summary.

[1] Zaheer et al., Deep Sets, NeurIPS 2017.

Deep Learning for Sets

- Deep Sets [1]

Equivariant Architecture

$$f(\mathbf{x}) = \sigma\left(\mathbf{x}\Lambda - \mathbf{1}\mathbf{1}^{\mathsf{T}}\mathbf{x}\Gamma\right)$$

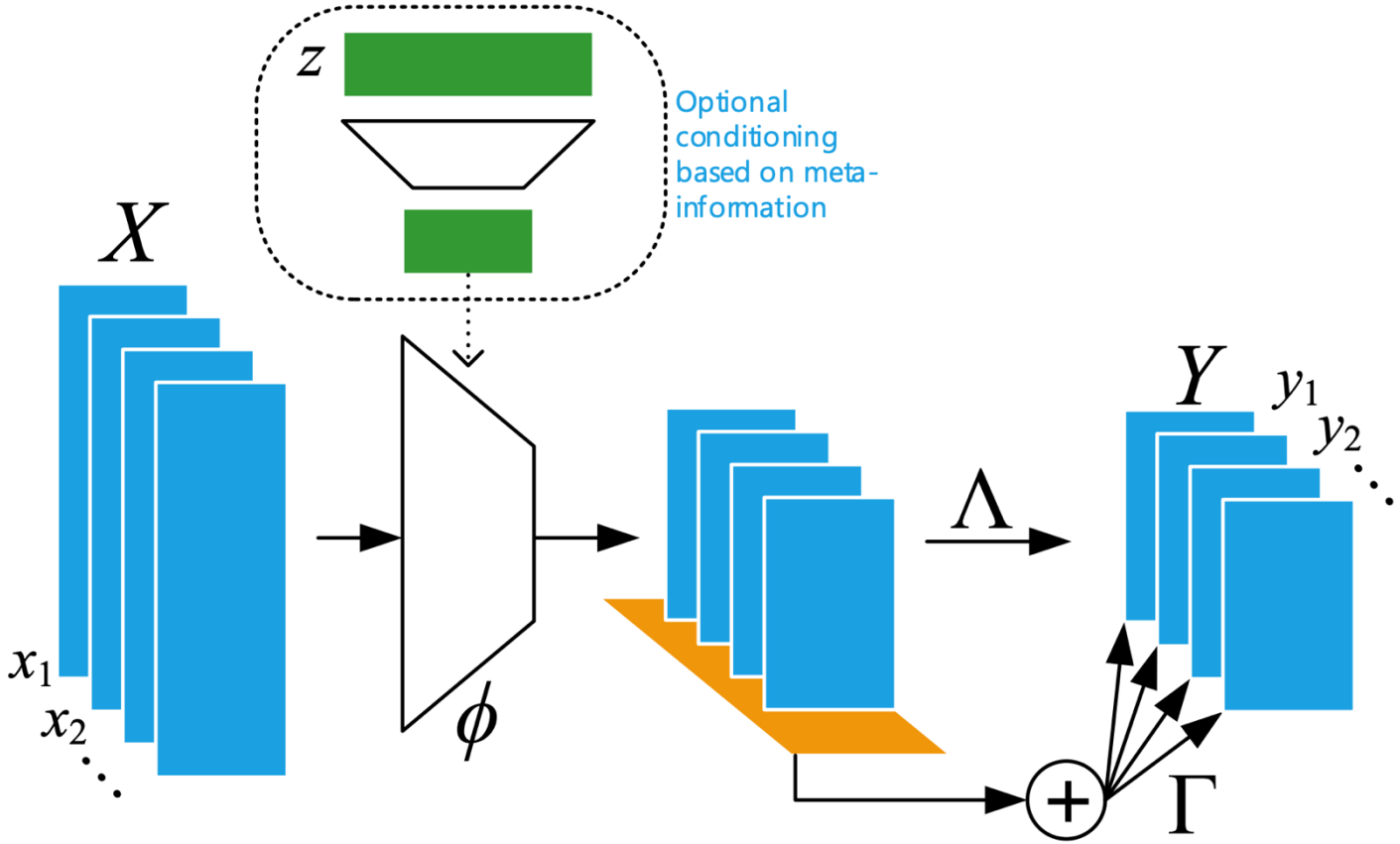


Image Credit: [1] Zaheer, Manzil, et al. "Deep sets." Advances in neural information processing systems 30 (2017).

Deep Learning for Sets

- Deep Sets [1]

  Recipe for making the model deep:

  Stack multiple equivariant layers (+ invariant layer at the end), e.g., PointNet [2]

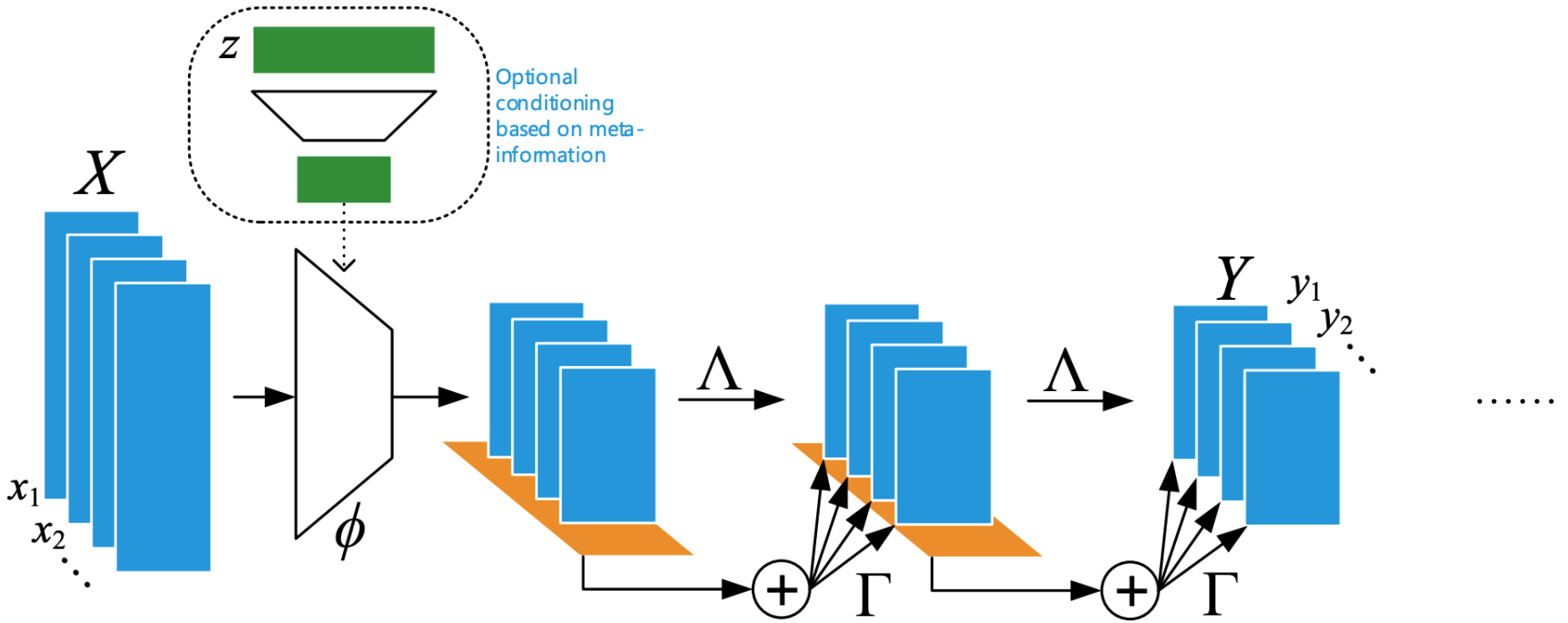


Image Credit: [1] Zaheer, Manzil, et al. "Deep sets." Advances in neural information processing systems 30 (2017).

Outline

- Invariance & Equivariance Principle
  - Translation equivariance in convolutions
  - Permutation equivariance and invariance

- Models for Sets
  - DeepSets: representation theorem of permutation-invariant set functions & architecture
  - DeepSets: permutation-equivariant linear mapping & architecture

- Models for Sequences
  - **Transformers**
  - Positional encoding vs. Rotary Positional Embeddings (RoPE)
  - Attention & Flash Attention
  - Pre-norm vs. post-norm
  - Vision Transformers (ViT) & Swin Transformers

Deep Learning for Sequences

- Language Models

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

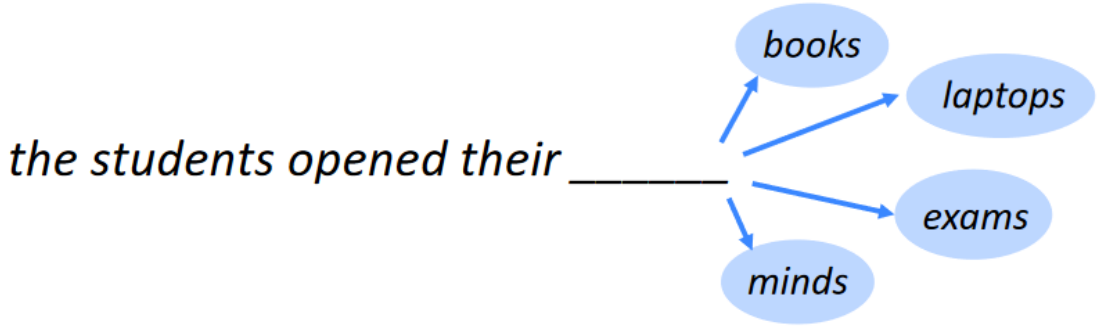
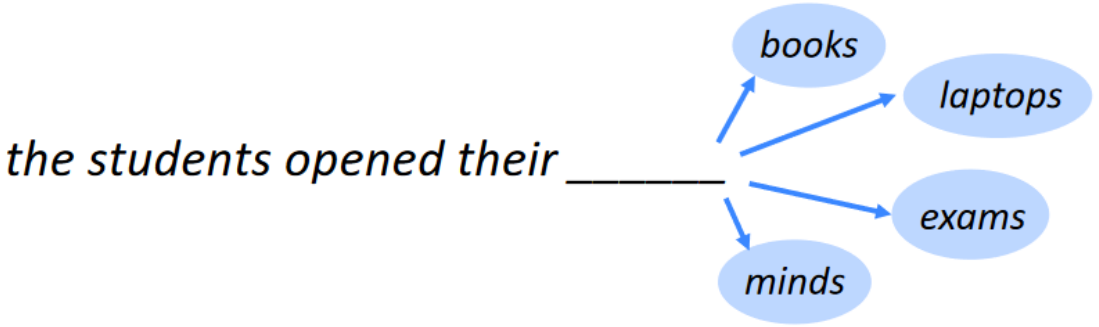


Image Credit: http://web.stanford.edu/class/cs224n/

Deep Learning for Sequences

- Language Models

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



- Machine Translation

Image Credit: http://web.stanford.edu/class/cs224n/ https://jalammar.github.io/illustrated-transformer/

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

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: https://www.mrc-cbu.cam.ac.uk/people/matt.davis/cmabridge/ https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

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)

LSTM [1]
GRU [2]
Seq2Seq [3]
Transformer [4]

Image Credit: https://www.mrc-cbu.cam.ac.uk/people/matt.davis/cmabridge/ https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

[1] Hochreiter, S. "Long Short-term Memory." Neural Computation MIT-Press (1997). [2] Cho, Kyunghyun. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." arXiv preprint arXiv:1406.1078 (2014). [3] Sutskever, I. "Sequence to Sequence Learning with Neural Networks." arXiv preprint arXiv:1409.3215 (2014). [4] Vaswani, A. "Attention is all you need." Advances in Neural Information Processing Systems (2017).

Transformers

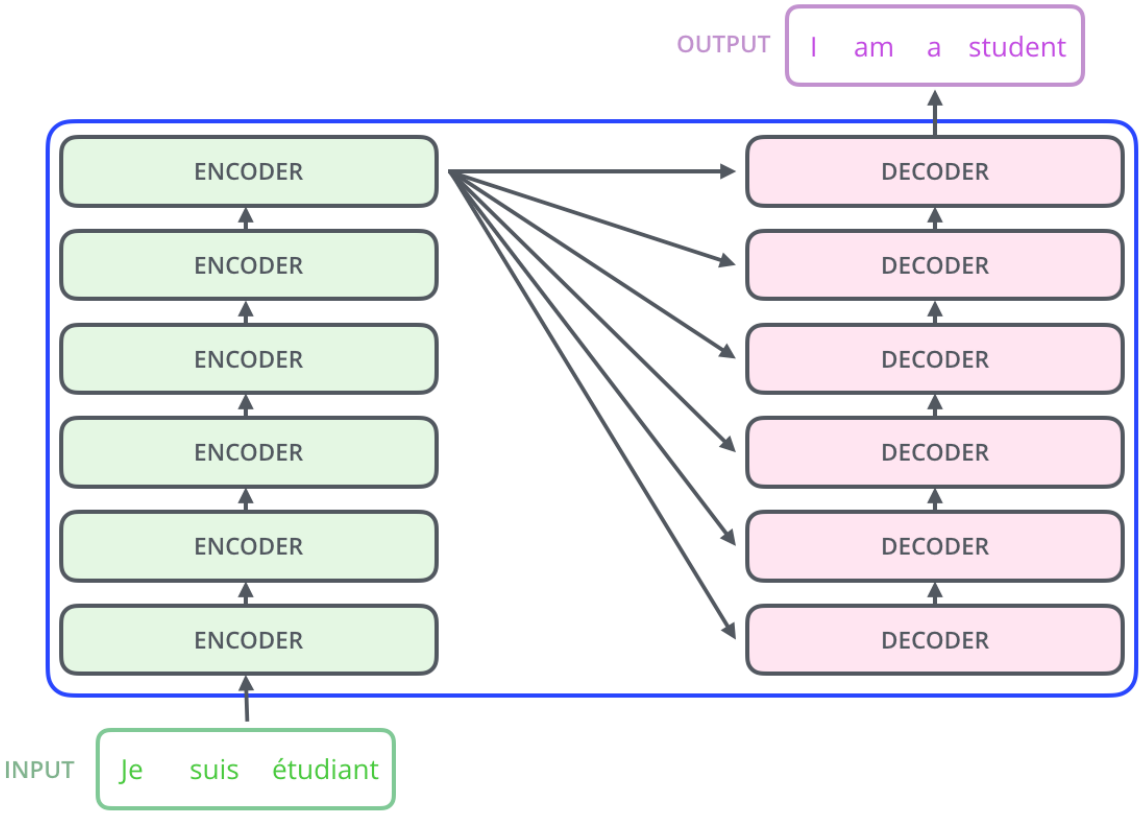
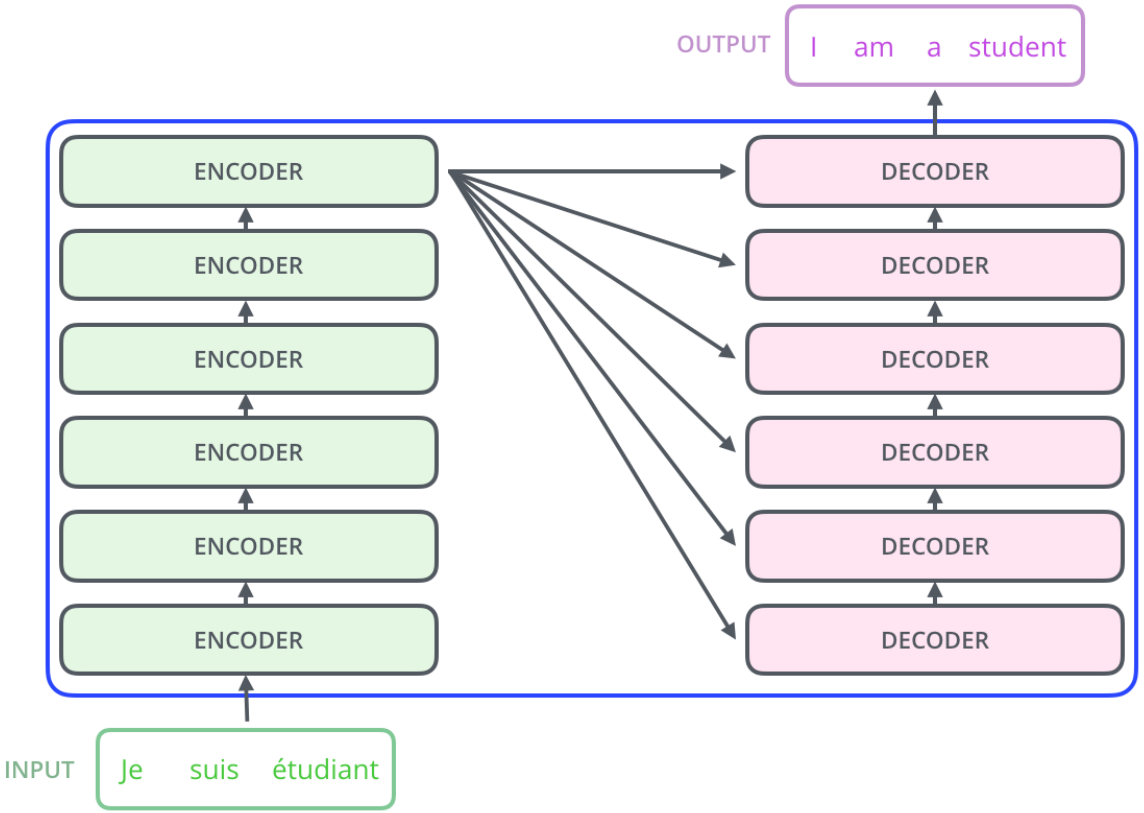


Image Credit: https://jalammar.github.io/illustrated-transformer/

Transformers



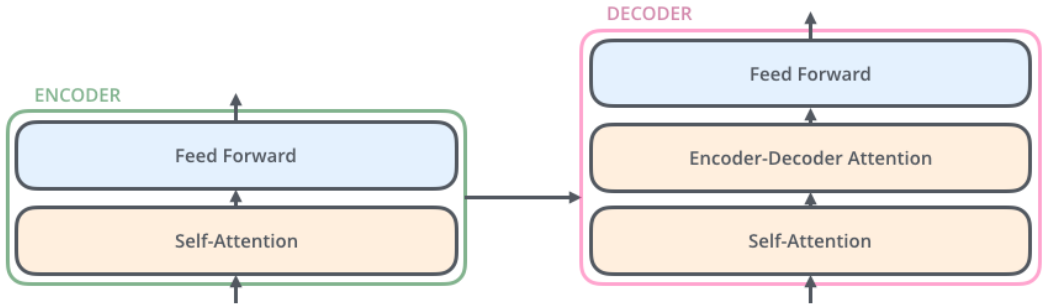
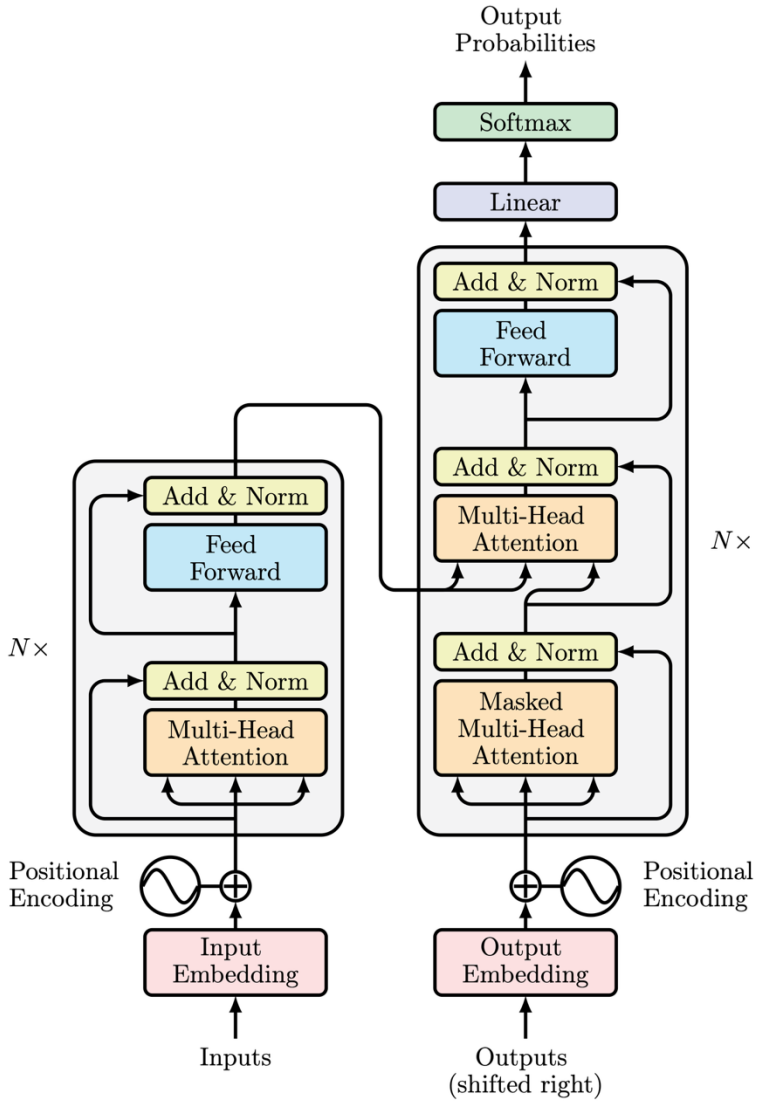


Image Credit: https://jalammar.github.io/illustrated-transformer/

Transformers



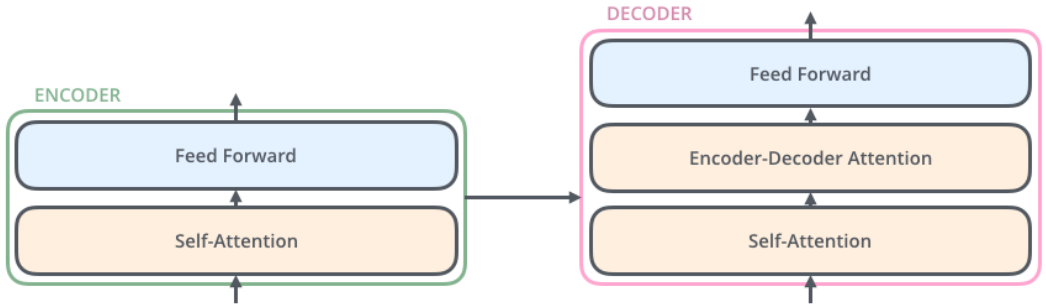
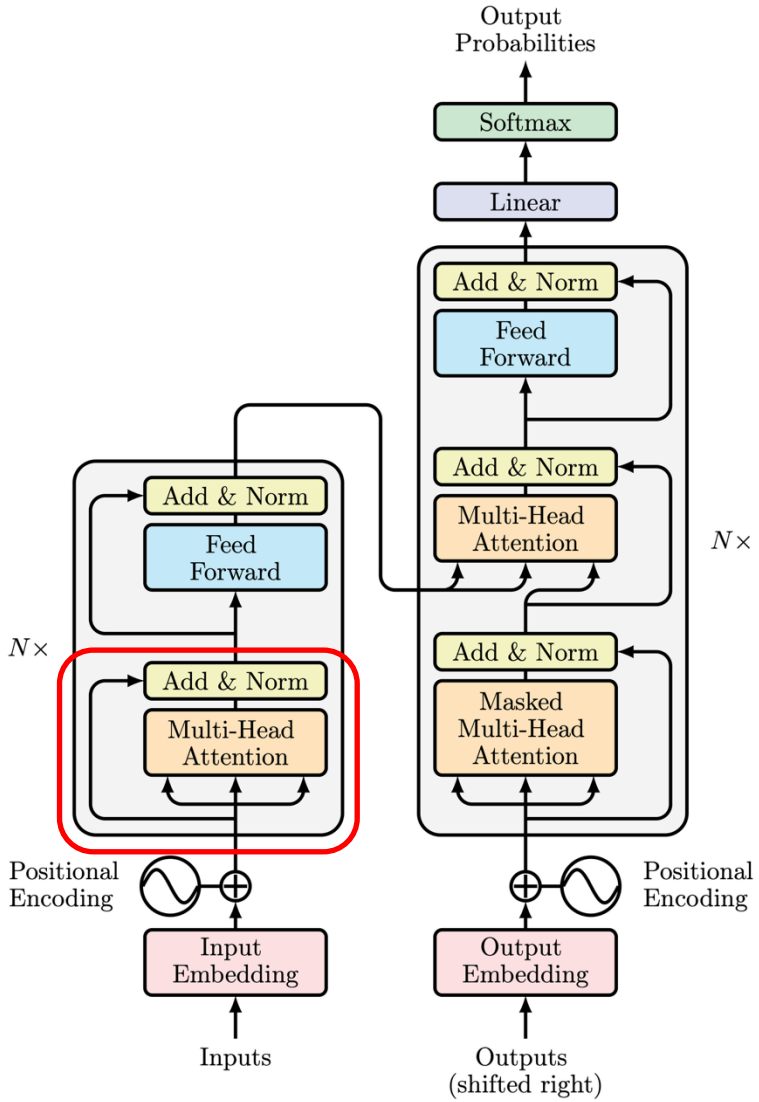


Image Credit: Vaswani, A. "Attention is all you need." Advances in Neural Information Processing Systems (2017). https://jalammar.github.io/illustrated-transformer/

Transformers



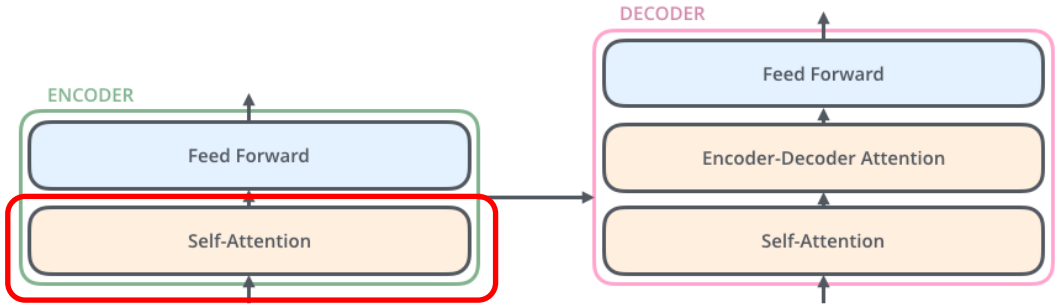
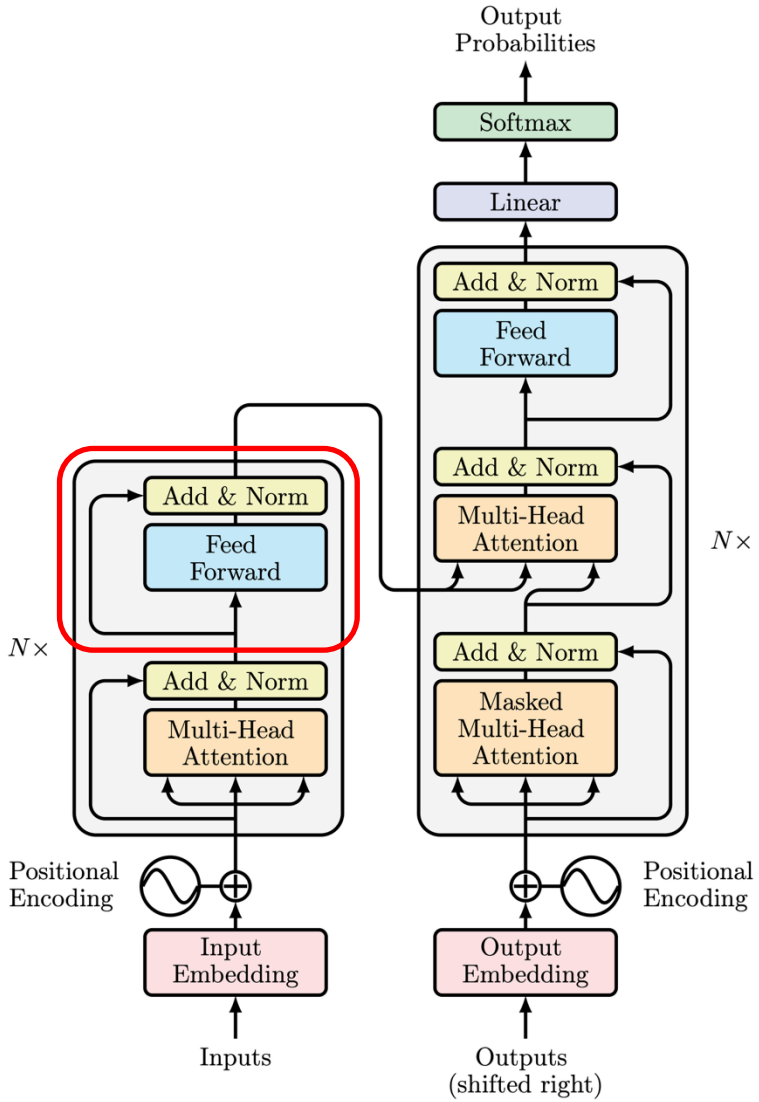


Image Credit: Vaswani, A. "Attention is all you need." Advances in Neural Information Processing Systems (2017). https://jalammar.github.io/illustrated-transformer/

Transformers



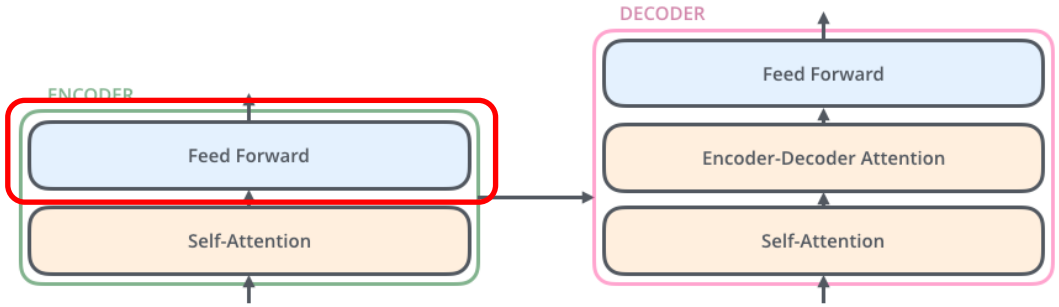
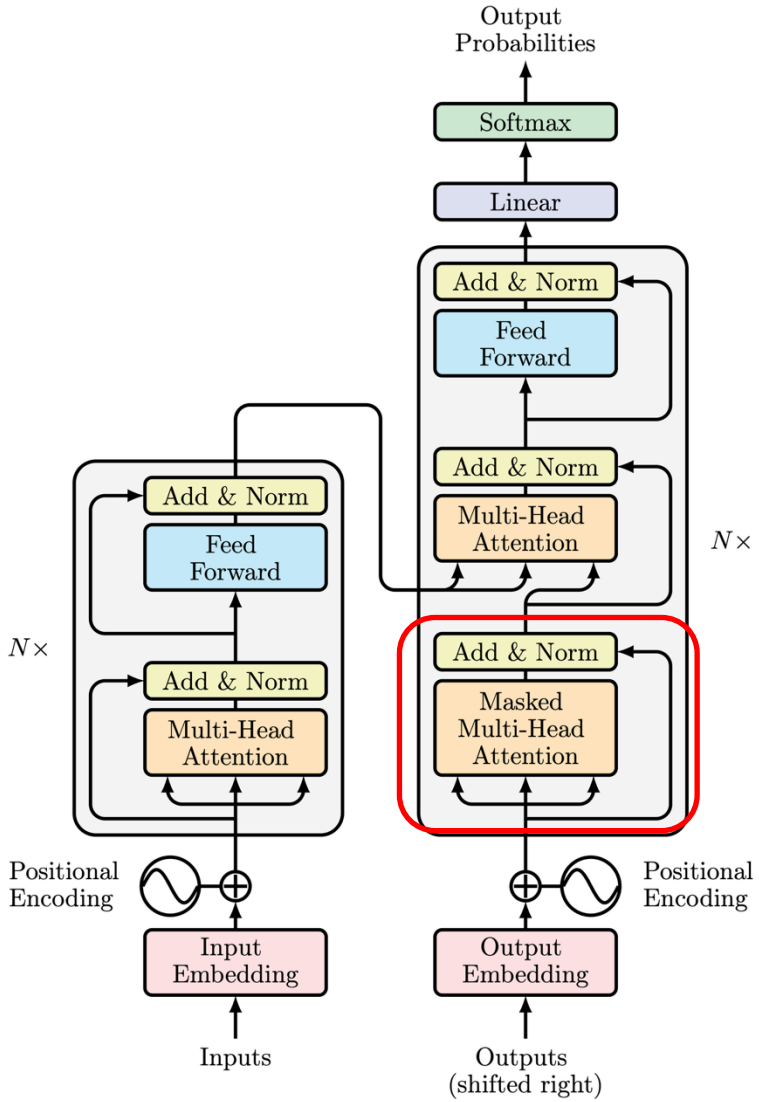


Image Credit: Vaswani, A. "Attention is all you need." Advances in Neural Information Processing Systems (2017). https://jalammar.github.io/illustrated-transformer/

Transformers



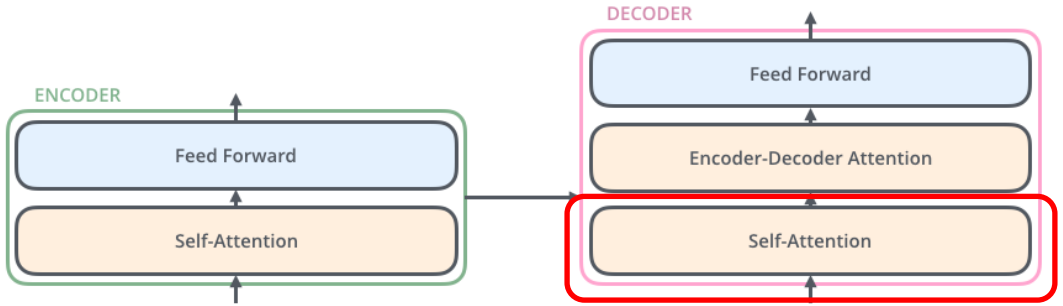
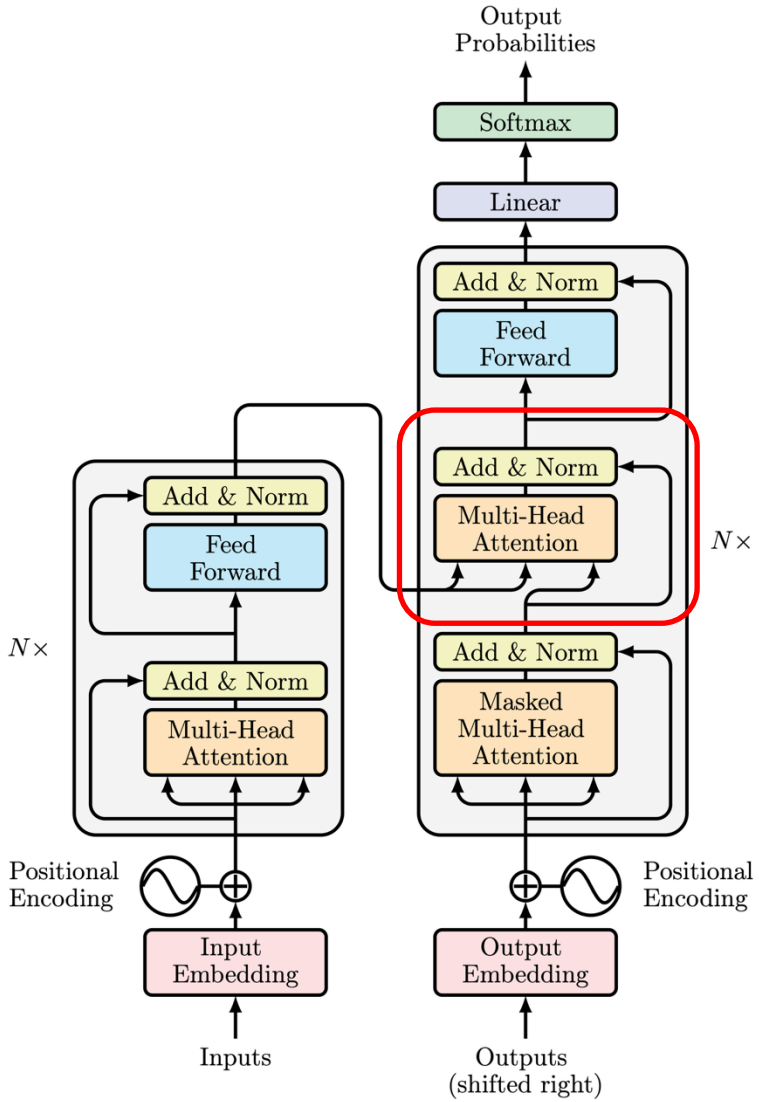


Image Credit: Vaswani, A. "Attention is all you need." Advances in Neural Information Processing Systems (2017). https://jalammar.github.io/illustrated-transformer/

Transformers



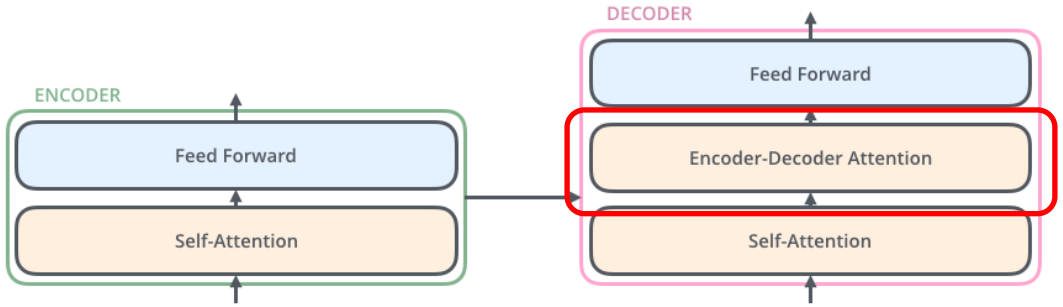
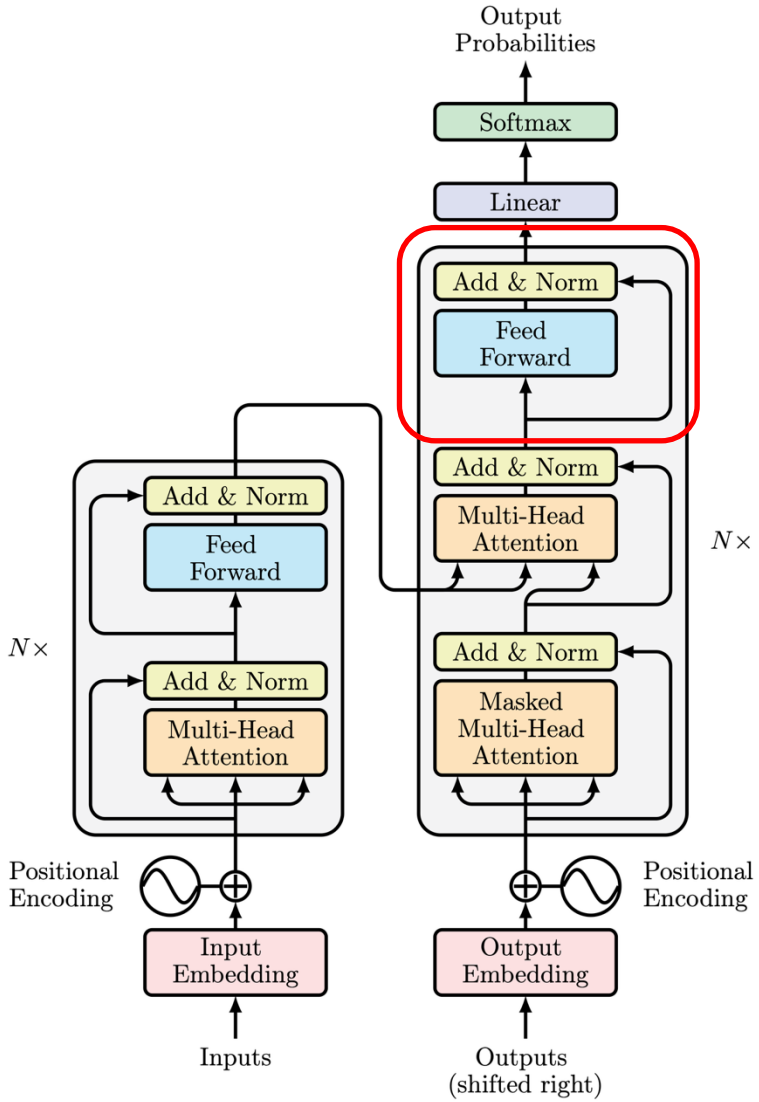


Image Credit: Vaswani, A. "Attention is all you need." Advances in Neural Information Processing Systems (2017). https://jalammar.github.io/illustrated-transformer/

Transformers



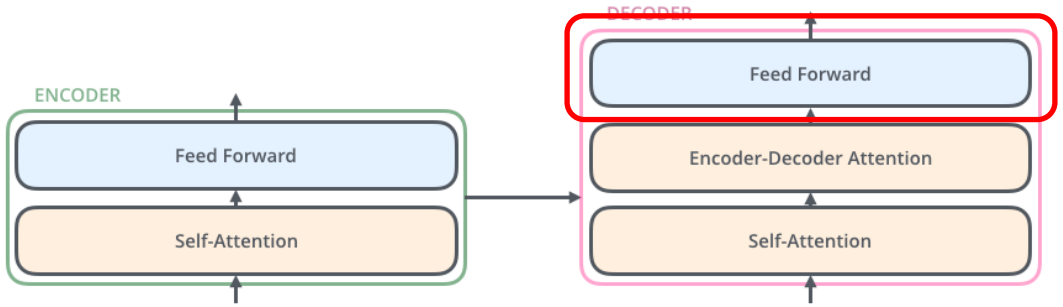


Image Credit: Vaswani, A. "Attention is all you need." Advances in Neural Information Processing Systems (2017). https://jalammar.github.io/illustrated-transformer/

Input Encoding

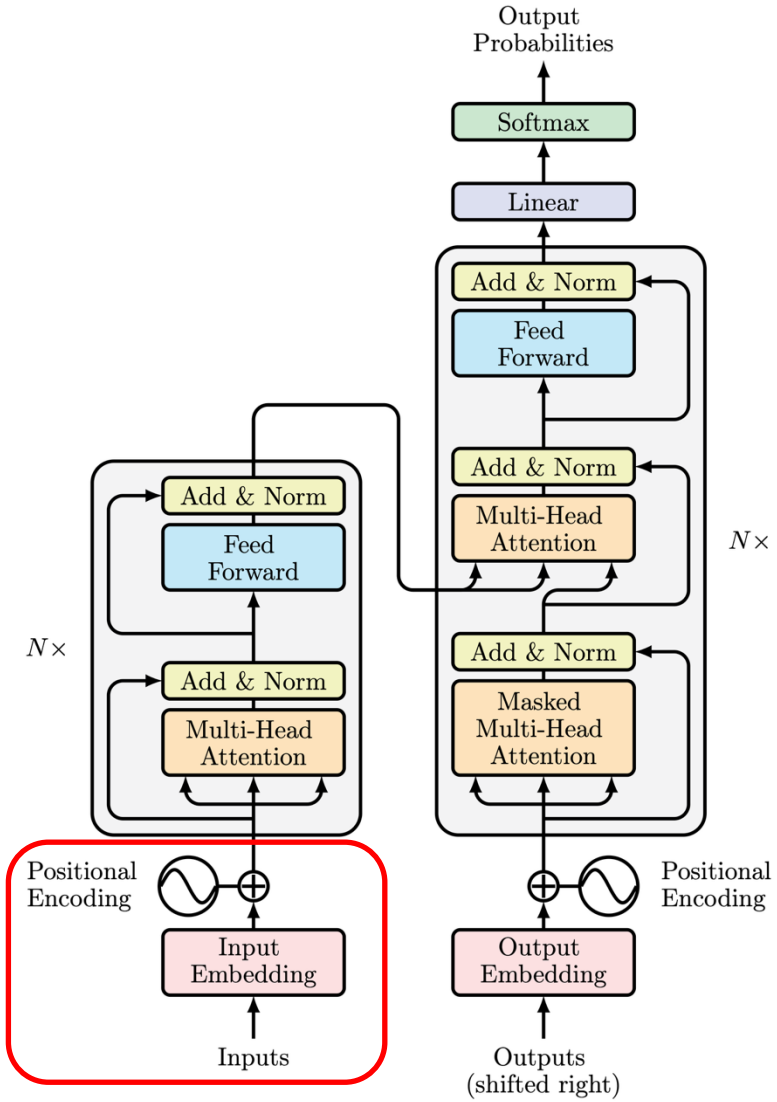
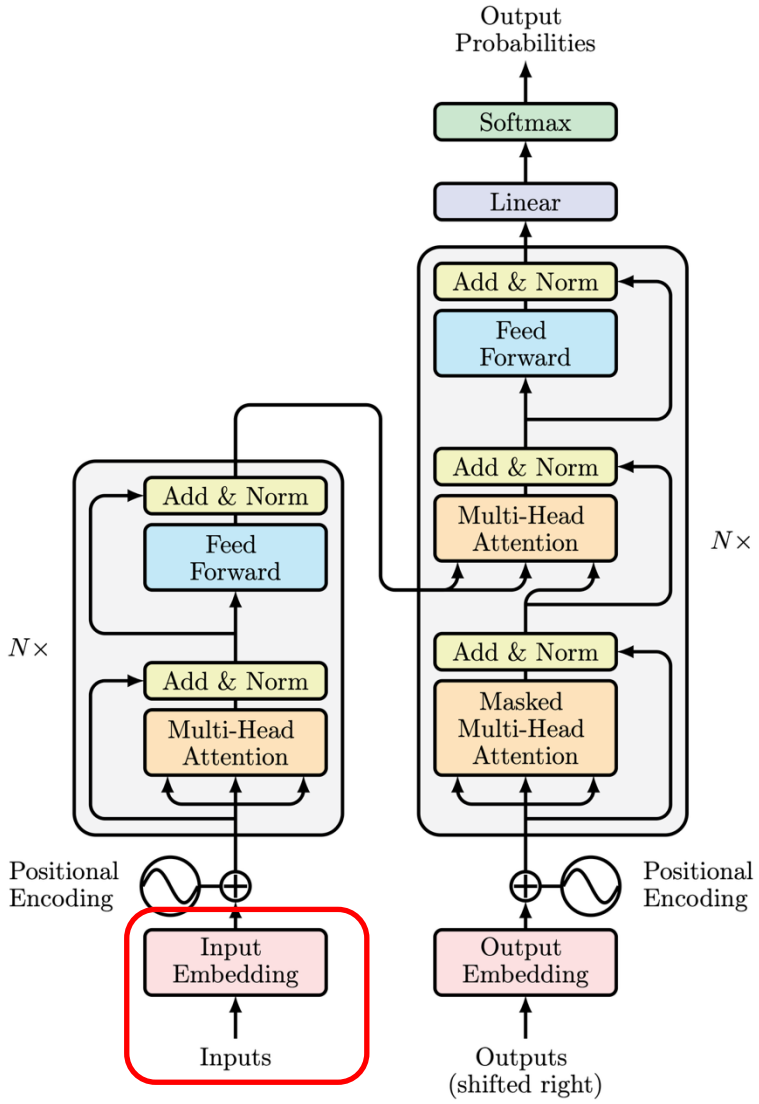


Image Credit: Vaswani, A. "Attention is all you need." Advances in Neural Information Processing Systems (2017). https://jalammar.github.io/illustrated-transformer/

Input Embedding



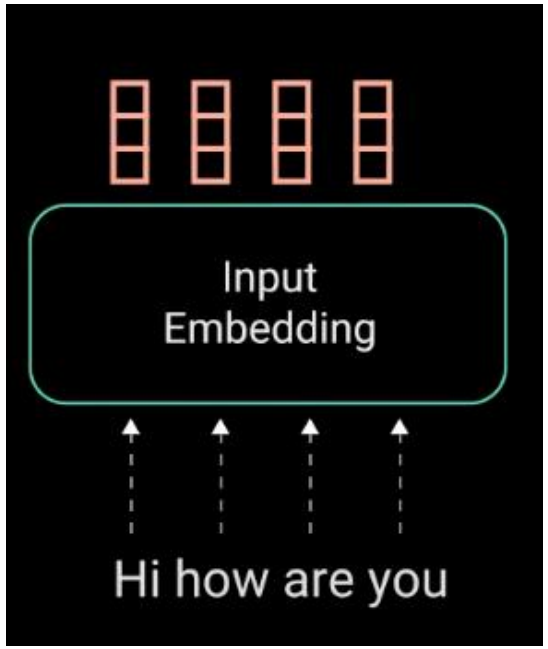


Image Credit: Vaswani, A. "Attention is all you need." Advances in Neural Information Processing Systems (2017). https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

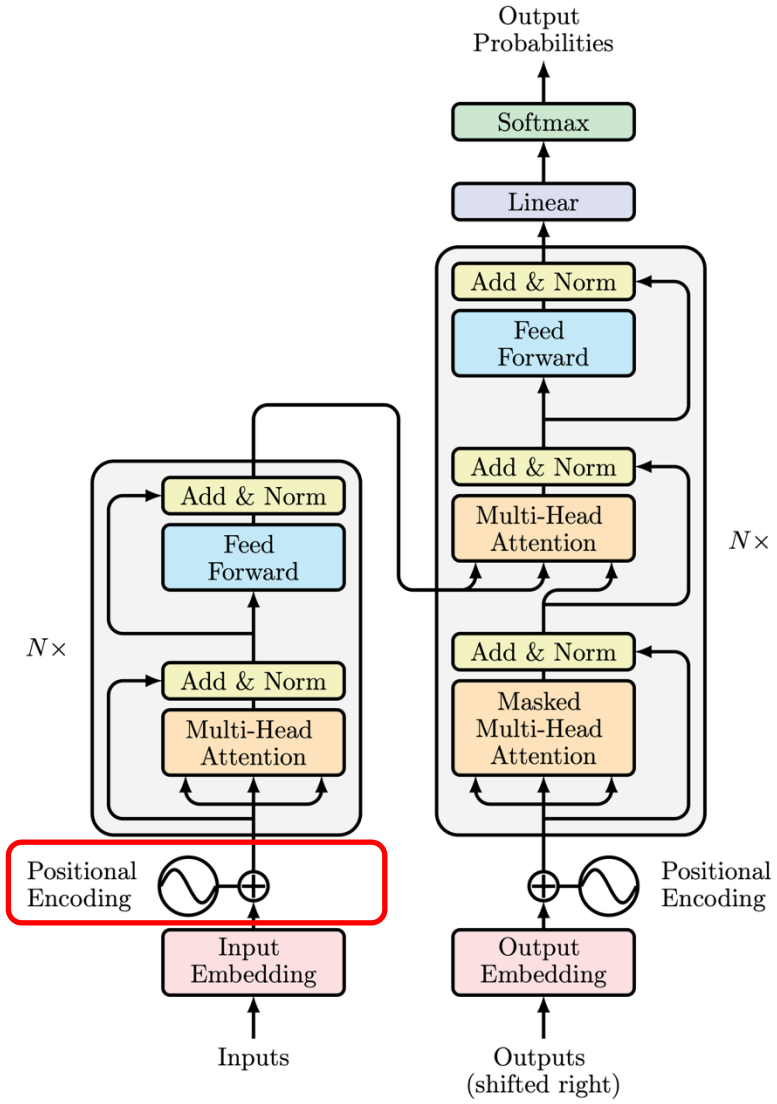
Outline

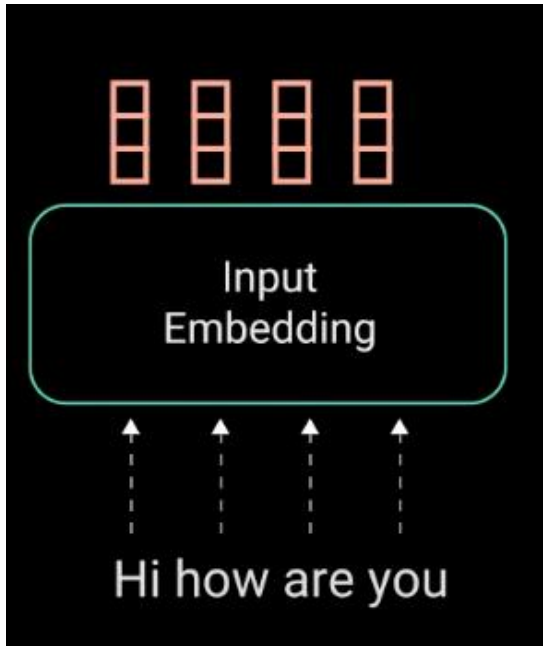
- Invariance & Equivariance Principle
  - Translation equivariance in convolutions
  - Permutation equivariance and invariance

- Models for Sets
  - DeepSets: representation theorem of permutation-invariant set functions & architecture
  - DeepSets: permutation-equivariant linear mapping & architecture

- Models for Sequences
  - Transformers
  - **Positional encoding vs. Rotary Positional Embeddings (RoPE)**
  - Attention & Flash Attention
  - Pre-norm vs. post-norm
  - Vision Transformers (ViT) & Swin Transformers

Positional Encoding



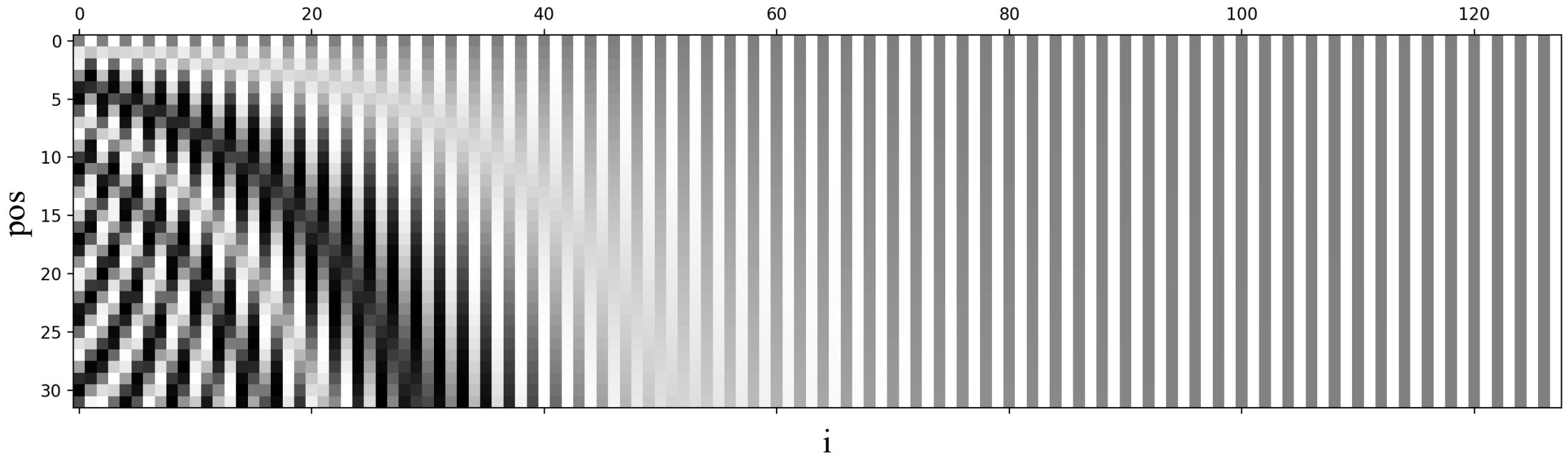


$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$

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

Image Credit: Vaswani, A. "Attention is all you need." Advances in Neural Information Processing Systems (2017). https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

Positional Encoding



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

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

Image Credit: https://lilianweng.github.io/lil-log/2020/04/07/the-transformer-family.html

Absolute vs. Relative Positional Encoding

Encode relative position information could help better model the dependency among tokens.

How to encode relative positions?

- We can inject the relative position into the bias of attention.
- We can use Rotary Position Embedding (RoPE) [1], which is more effective empirically.

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!

[1] Su, Jianlin, et al. "RoFormer: Enhanced Transformer with Rotary Position Embedding." arXiv preprint arXiv:2104.09864 (2021).

Rotary Positional Embedding

RoPE first divide d-dimension vector space in d/2 subspaces and then rotate them based on the 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 $\Theta = \{\theta_i = 10000^{-2(i-1)/d}, i \in [1, 2, ..., d/2]\}$

In practice, we can apply 2D rotations to pairs $(x_1, x_{1+d/2}), (x_2, x_{2+d/2}), \ldots, (x_{d/2}, x_d)$

Rotary Positional Embedding

RoPE first divide d-dimension vector space in d/2 subspaces and then rotate them based on the position:

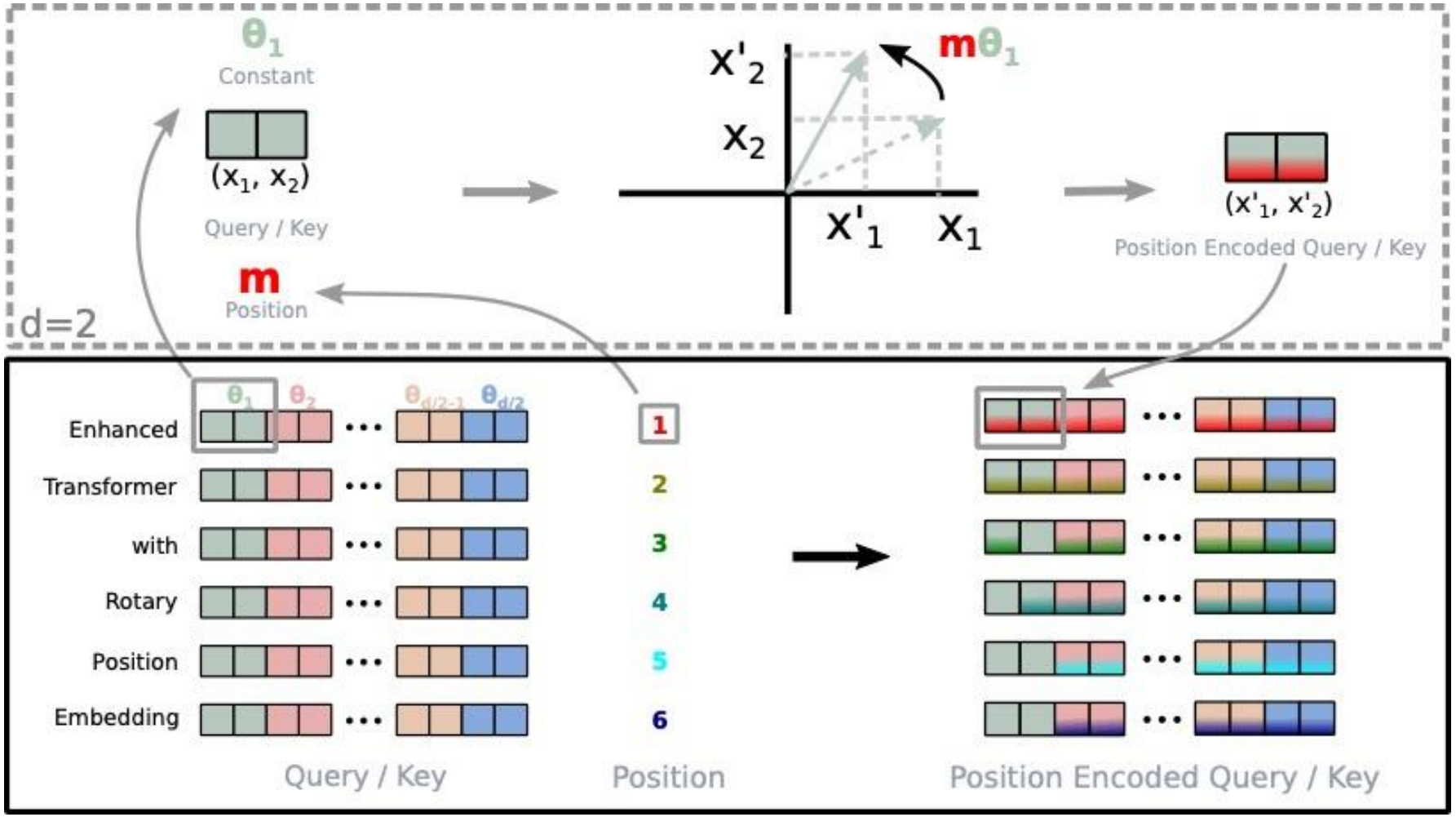


Image Credit: Su, Jianlin, et al. "RoFormer: Enhanced Transformer with Rotary Position Embedding." arXiv preprint arXiv:2104.09864 (2021).

Rotary Positional Embedding

What do we gain in RoPE?

- Inner product depends on the relative position

Let us look at the case of 2D:

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

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

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

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

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

Rotary Positional Embedding

What do we gain in RoPE?

- Inner product depends on the relative position

Let us look at the case of 2D:

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

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

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

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

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

This holds for d-dimension as we construct a block-diagonal matrix with 2D rotation matrices!

Rotary Positional Embedding

What do we gain in RoPE?

- Long-term decay of inner product w.r.t. relative positions

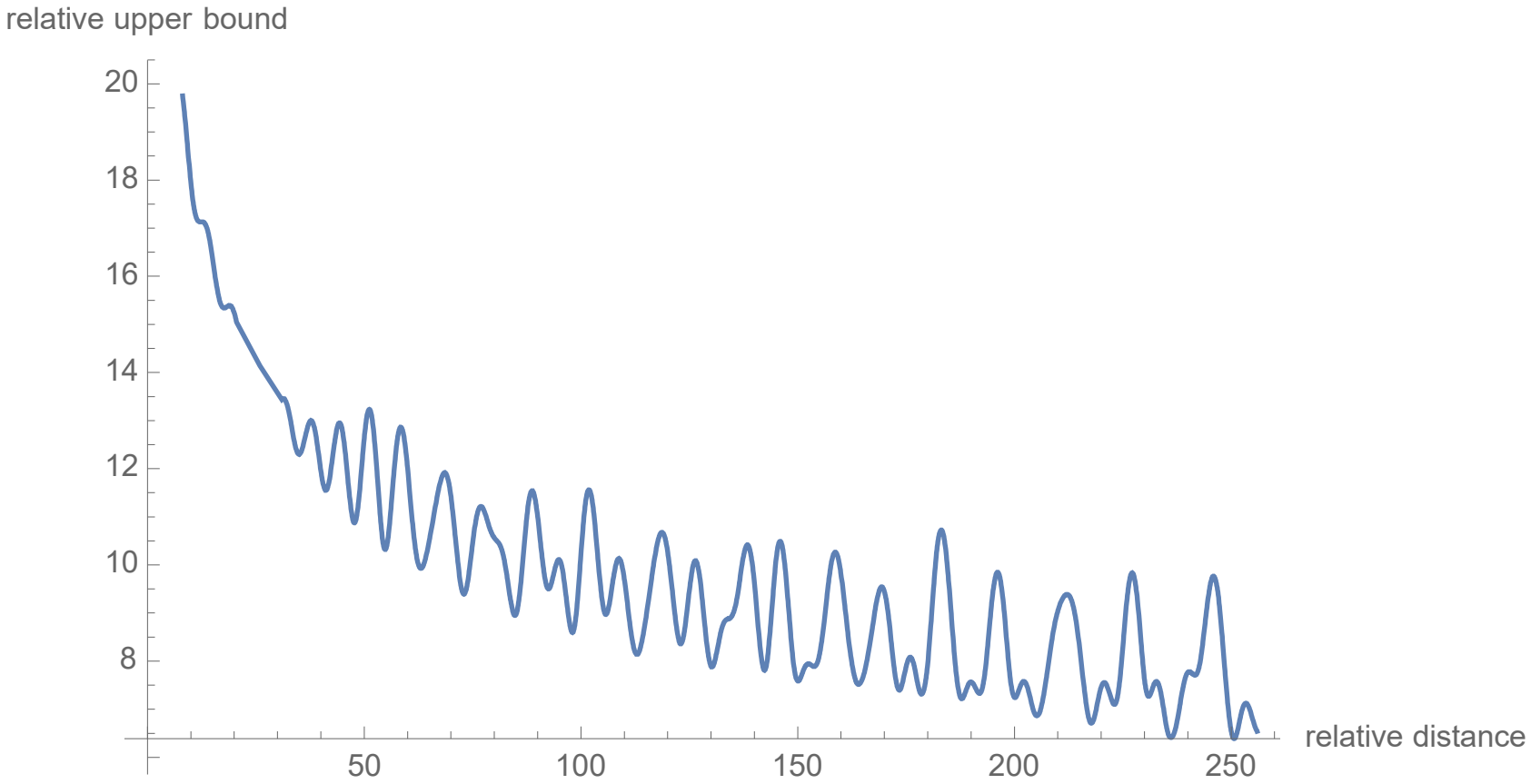


Image Credit: Su, Jianlin, et al. "RoFormer: Enhanced Transformer with Rotary Position Embedding." arXiv preprint arXiv:2104.09864 (2021).

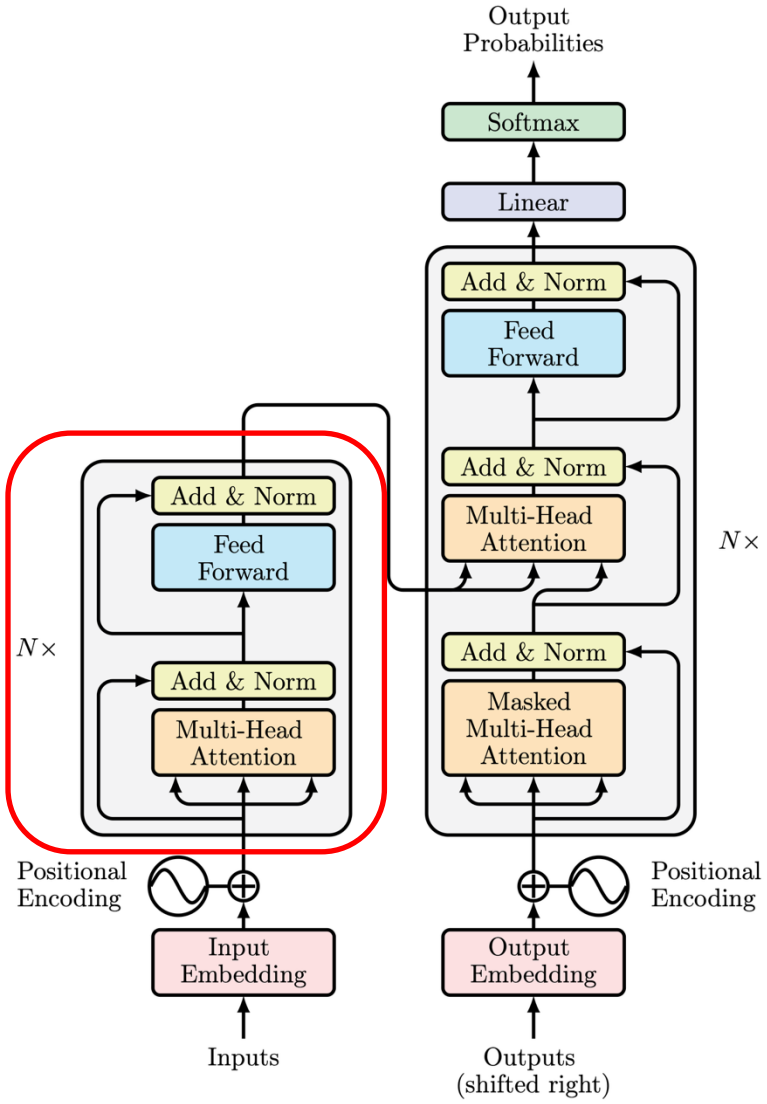
Outline

- Invariance & Equivariance Principle
  - Translation equivariance in convolutions
  - Permutation equivariance and invariance

- Models for Sets
  - DeepSets: representation theorem of permutation-invariant set functions & architecture
  - DeepSets: permutation-equivariant linear mapping & architecture

- Models for Sequences
  - Transformers
  - Positional encoding vs. Rotary Positional Embeddings (RoPE)
  - **Attention & Flash Attention**
  - Pre-norm vs. post-norm
  - Vision Transformers (ViT) & Swin Transformers

Encoder



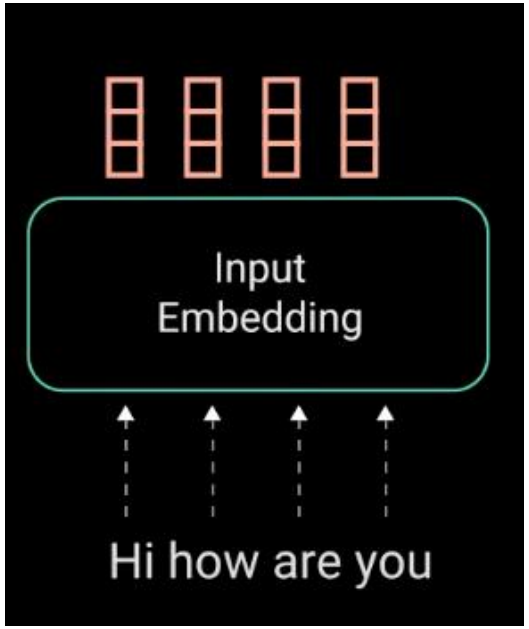
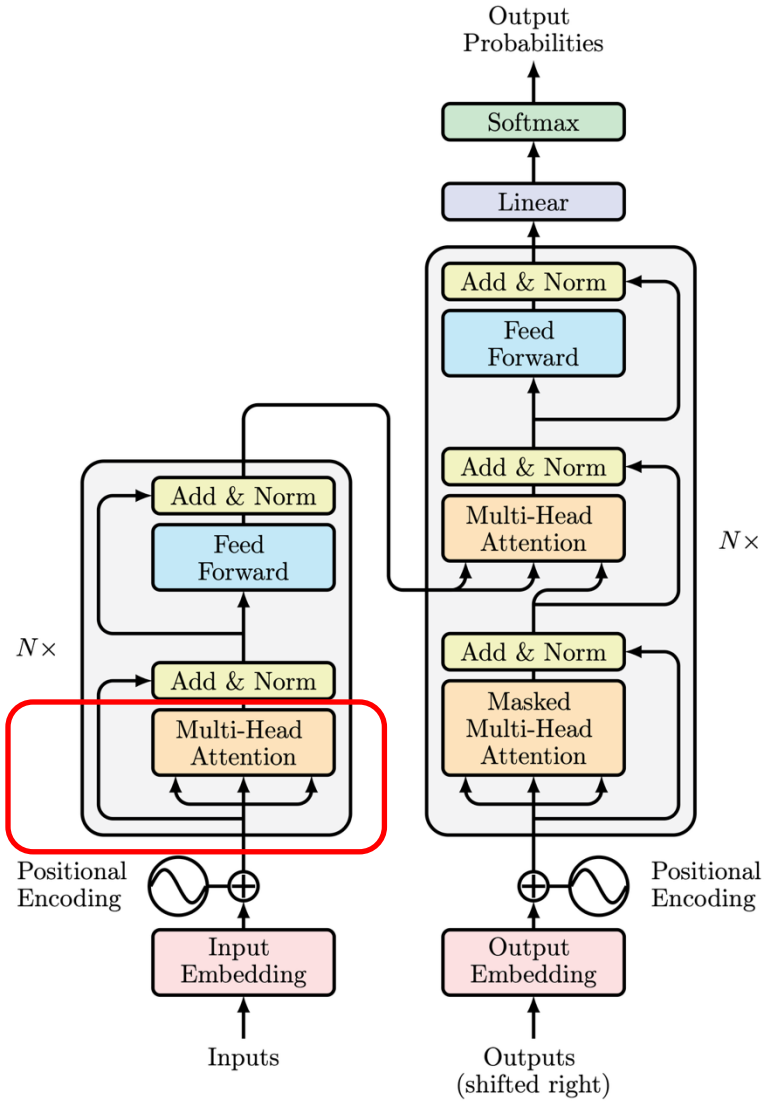


Image Credit: Vaswani, A. "Attention is all you need." Advances in Neural Information Processing Systems (2017). https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

Multi-Head Attention



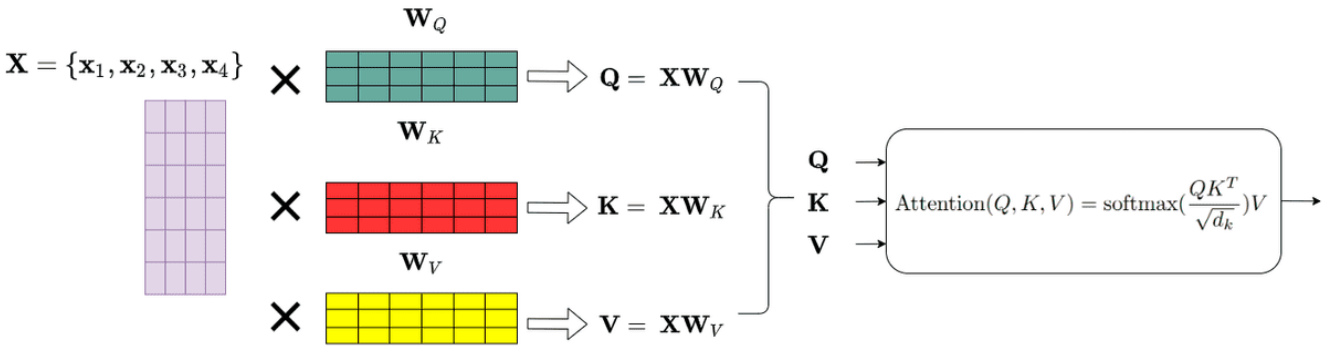
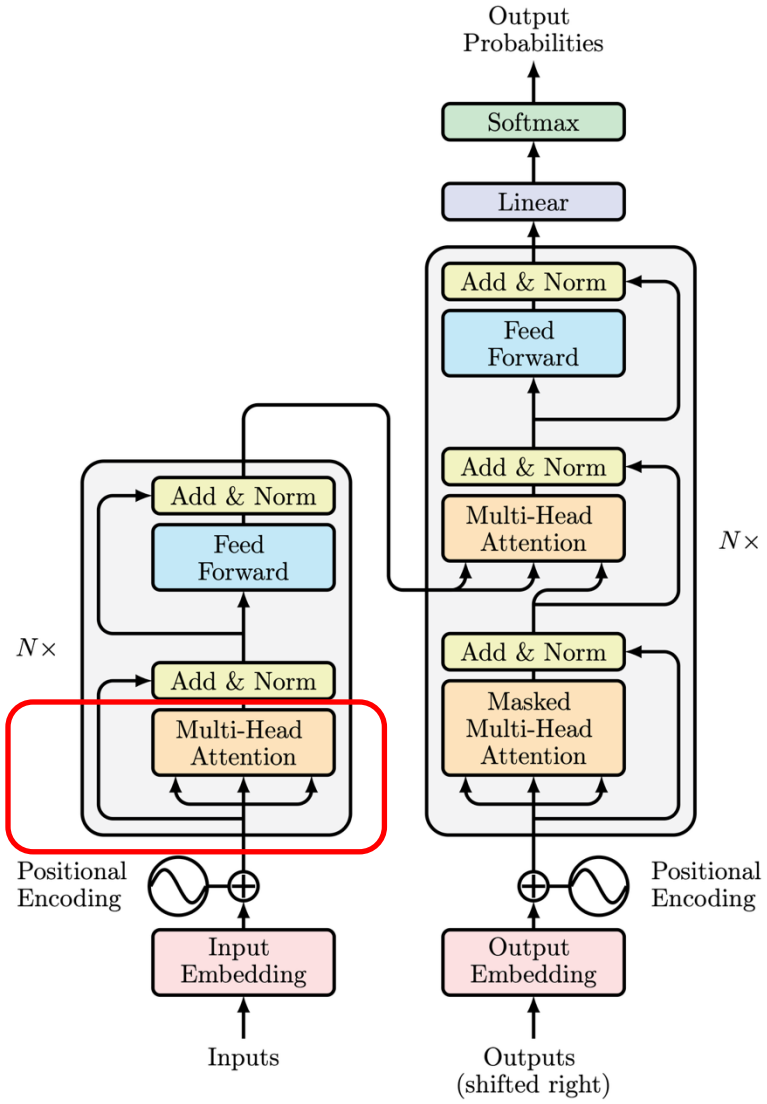


Image Credit: Vaswani, A. "Attention is all you need." Advances in Neural Information Processing Systems (2017). https://theaisummer.com/transformer/

Multi-Head Attention



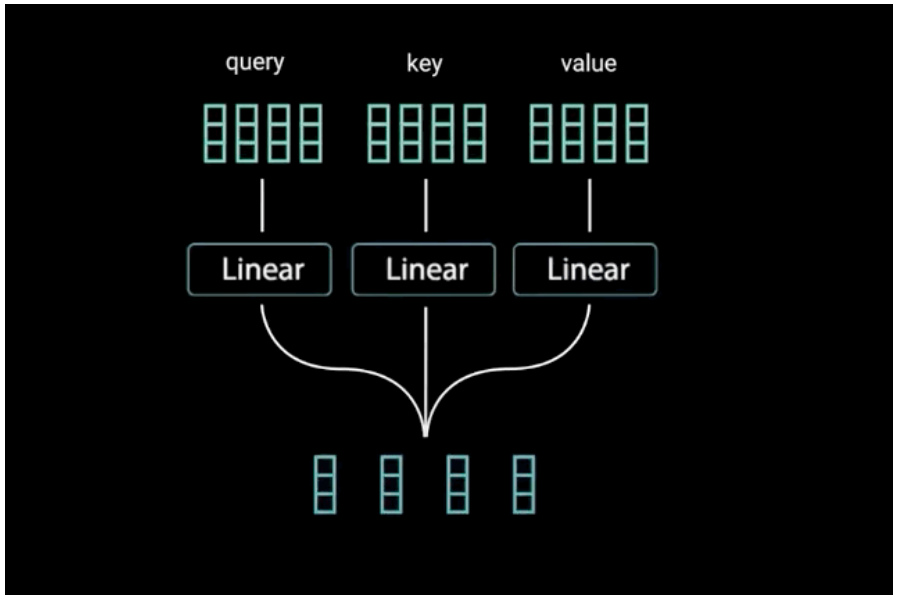
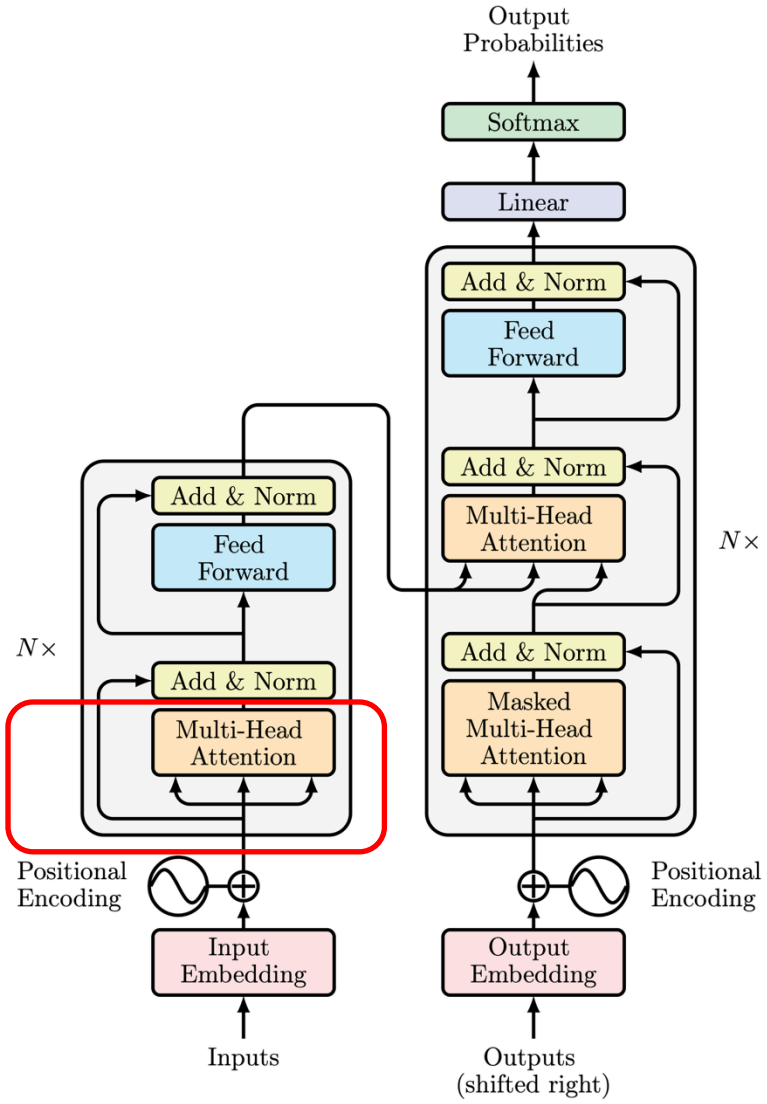
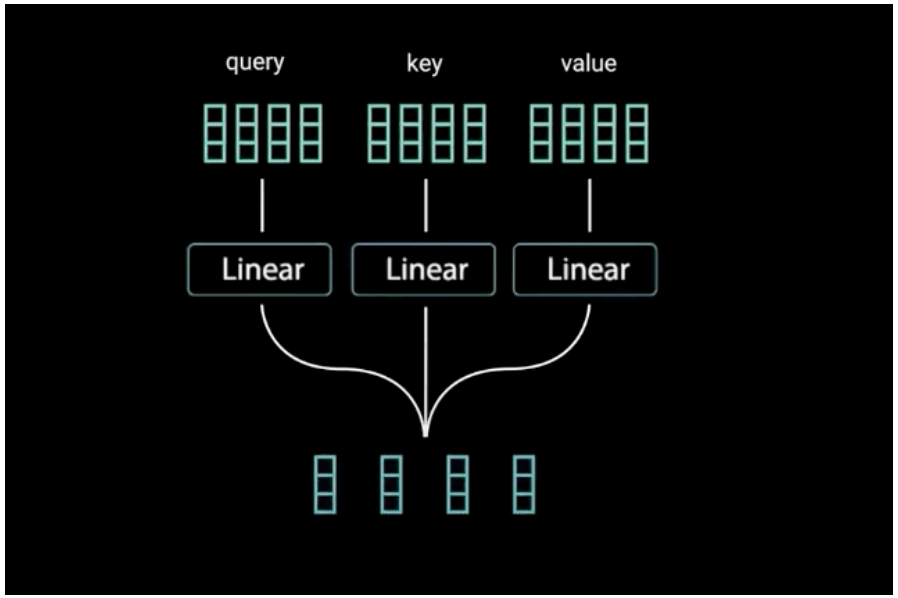


Image Credit: Vaswani, A. "Attention is all you need." Advances in Neural Information Processing Systems (2017). https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

Multi-Head Attention





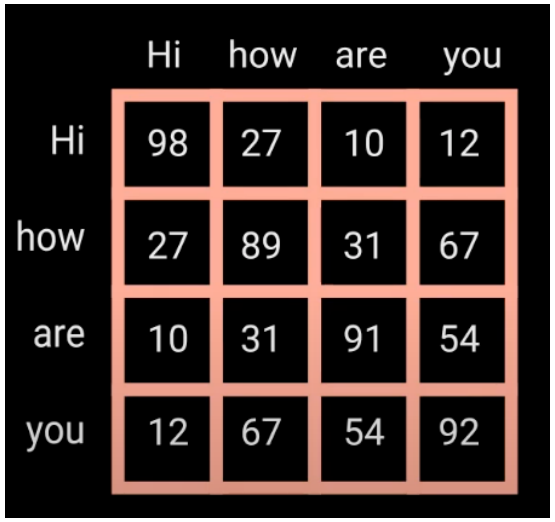
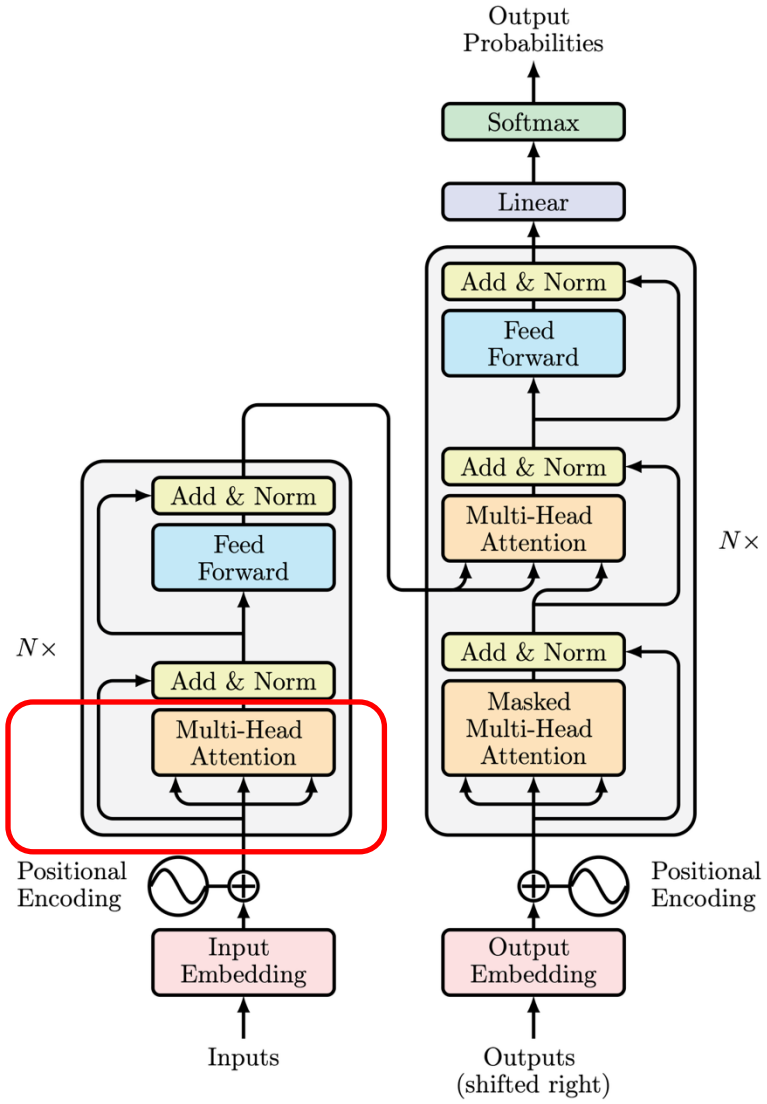
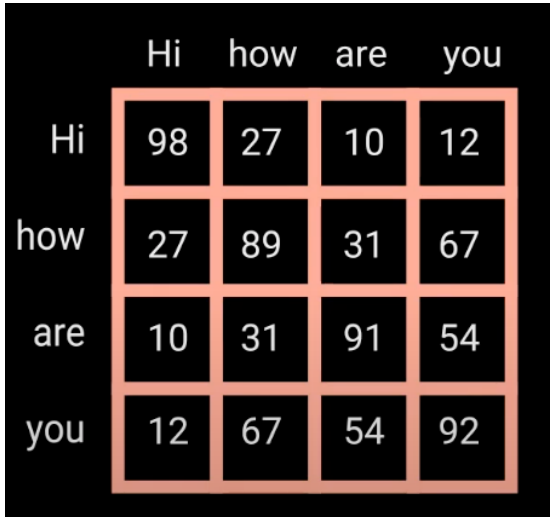


Image Credit: Vaswani, A. "Attention is all you need." Advances in Neural Information Processing Systems (2017). https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

Multi-Head Attention





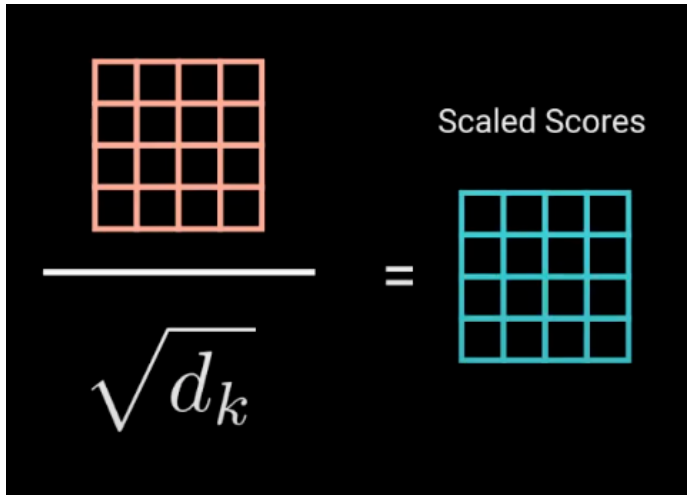
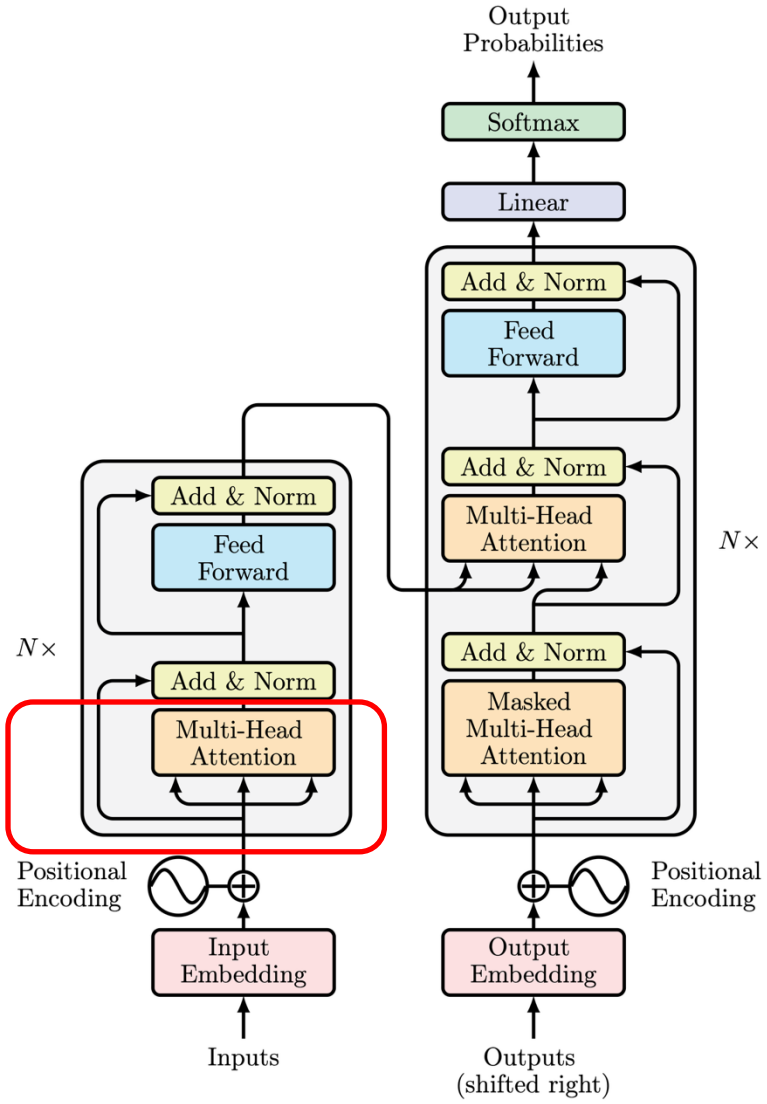
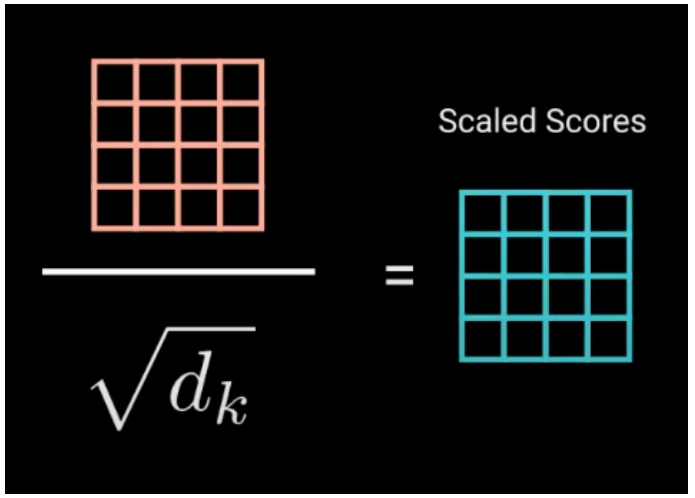


Image Credit: Vaswani, A. "Attention is all you need." Advances in Neural Information Processing Systems (2017). https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

Multi-Head Attention





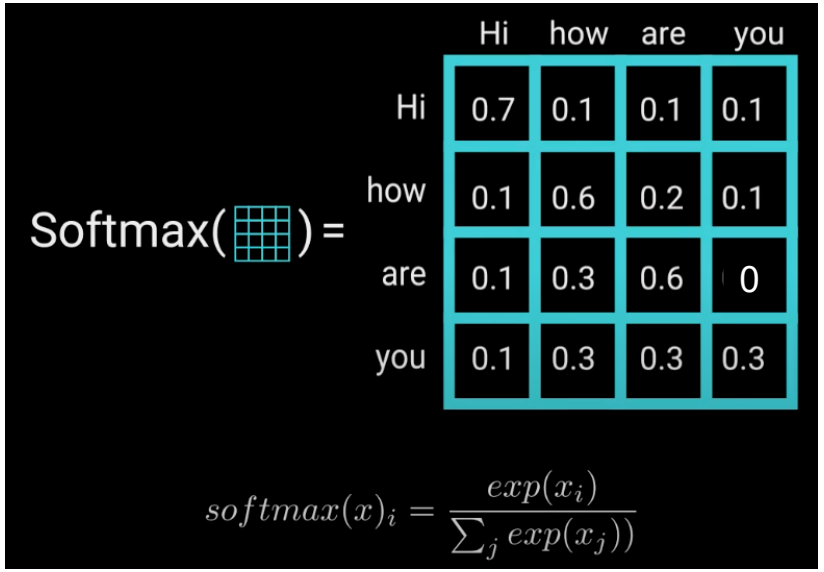
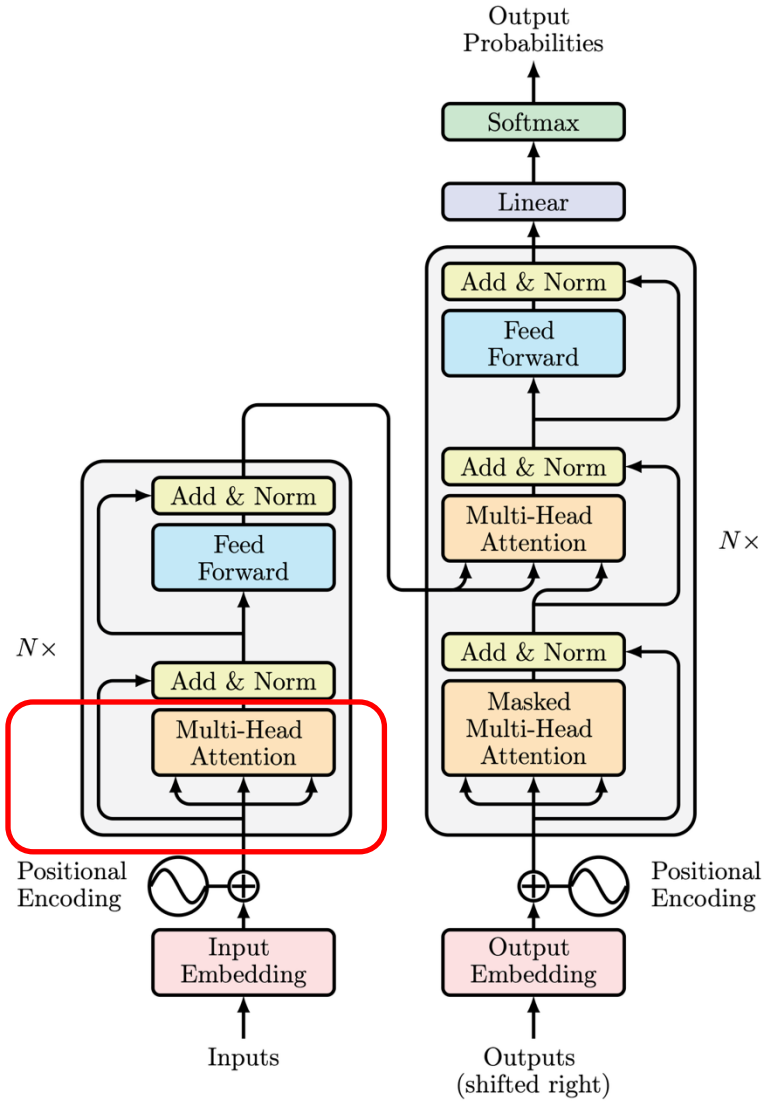
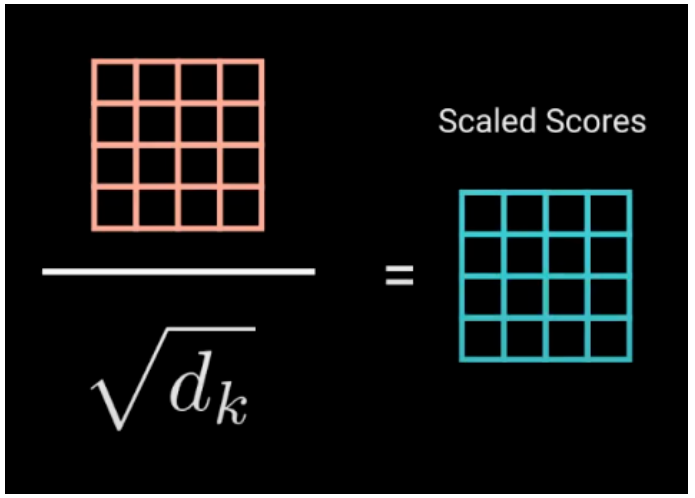


Image Credit: Vaswani, A. "Attention is all you need." Advances in Neural Information Processing Systems (2017). https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

Multi-Head Attention





Why square root?

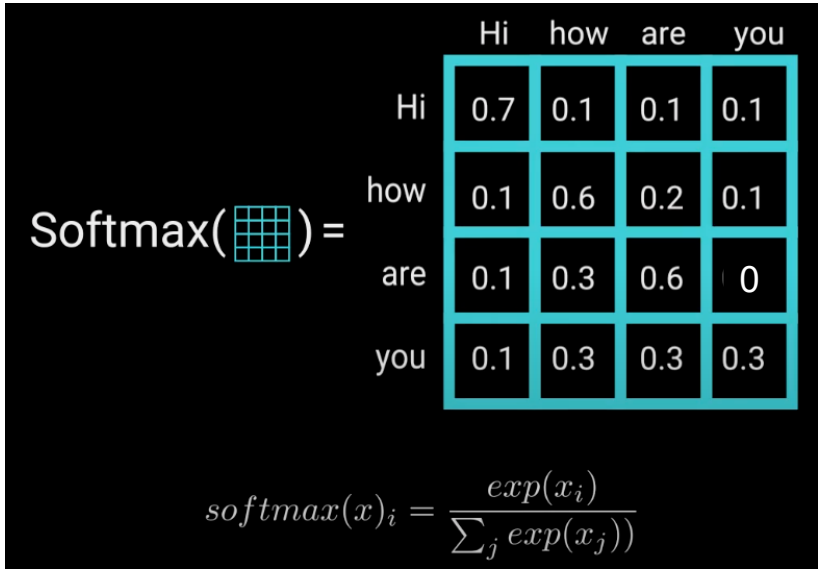
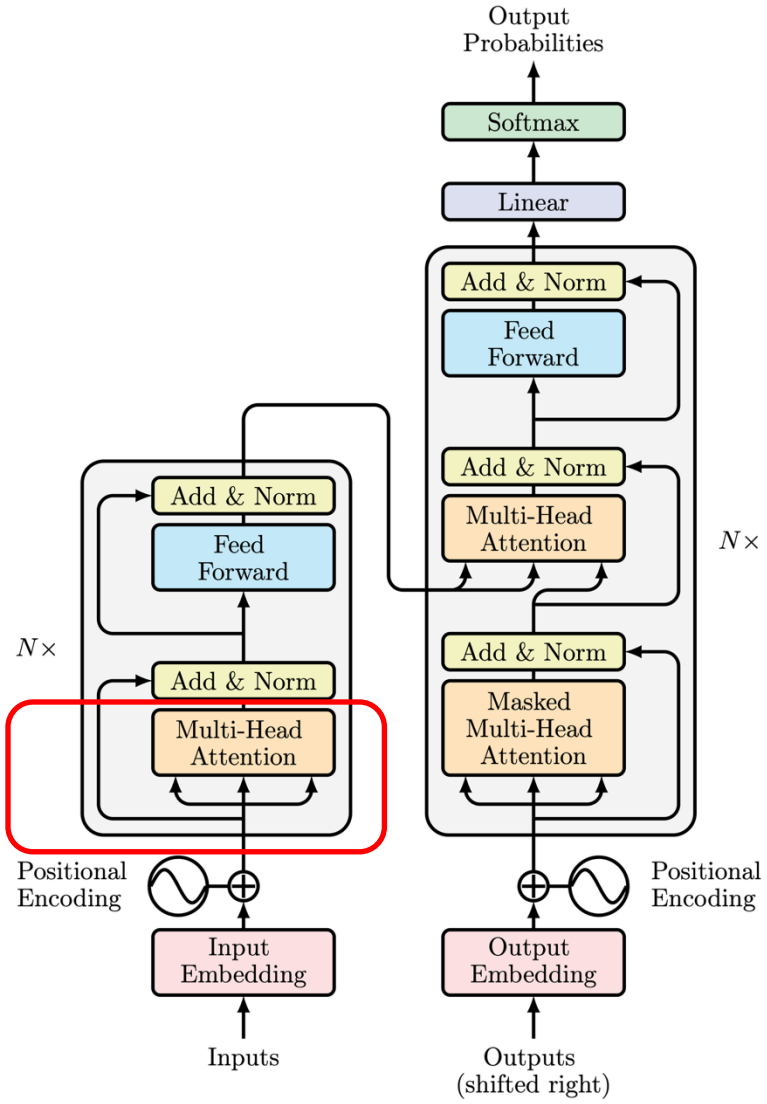
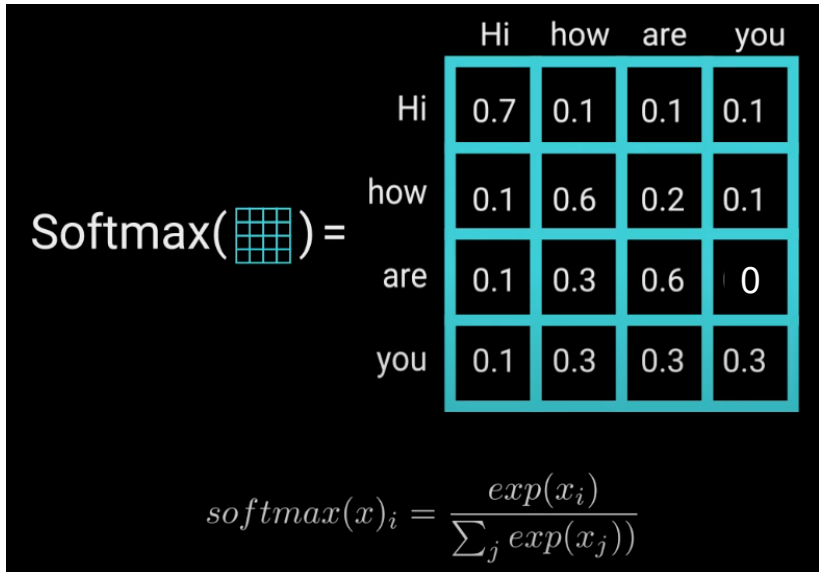


Image Credit: Vaswani, A. "Attention is all you need." Advances in Neural Information Processing Systems (2017). https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

Multi-Head Attention





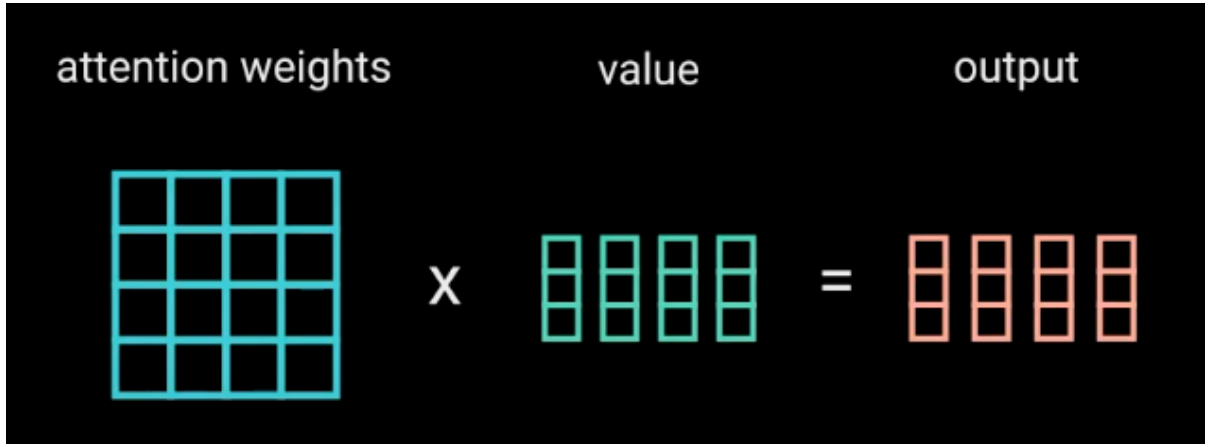
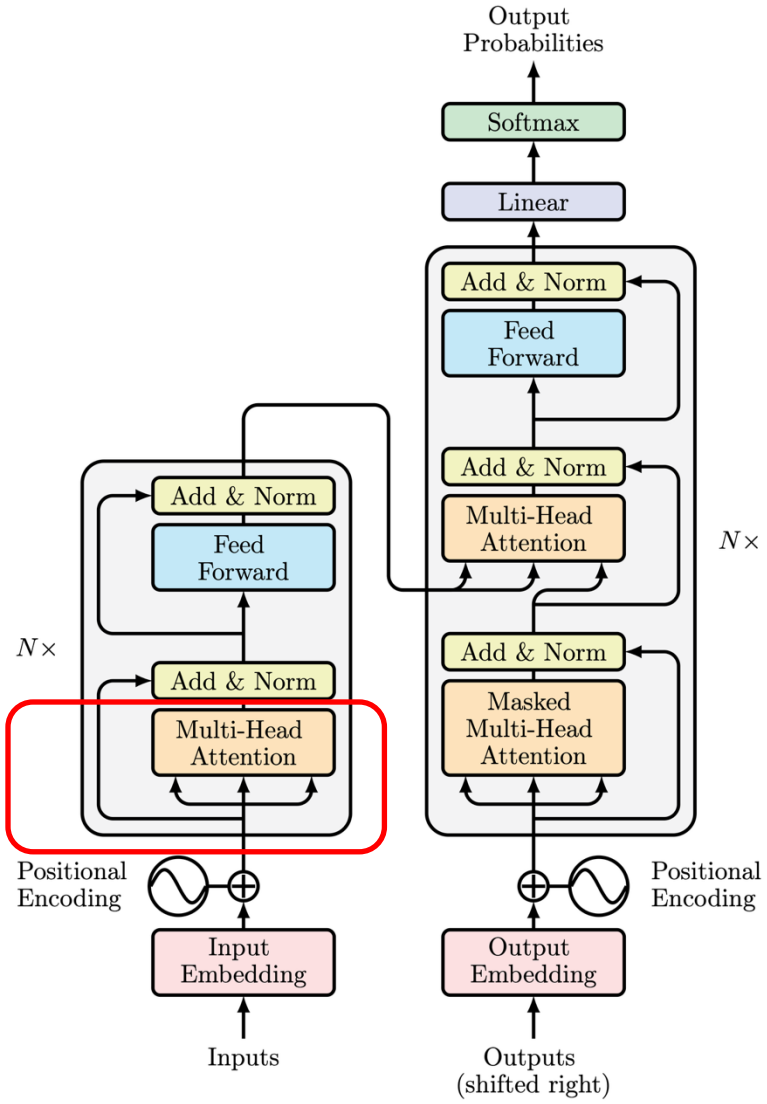


Image Credit: Vaswani, A. "Attention is all you need." Advances in Neural Information Processing Systems (2017). https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

Multi-Head Attention



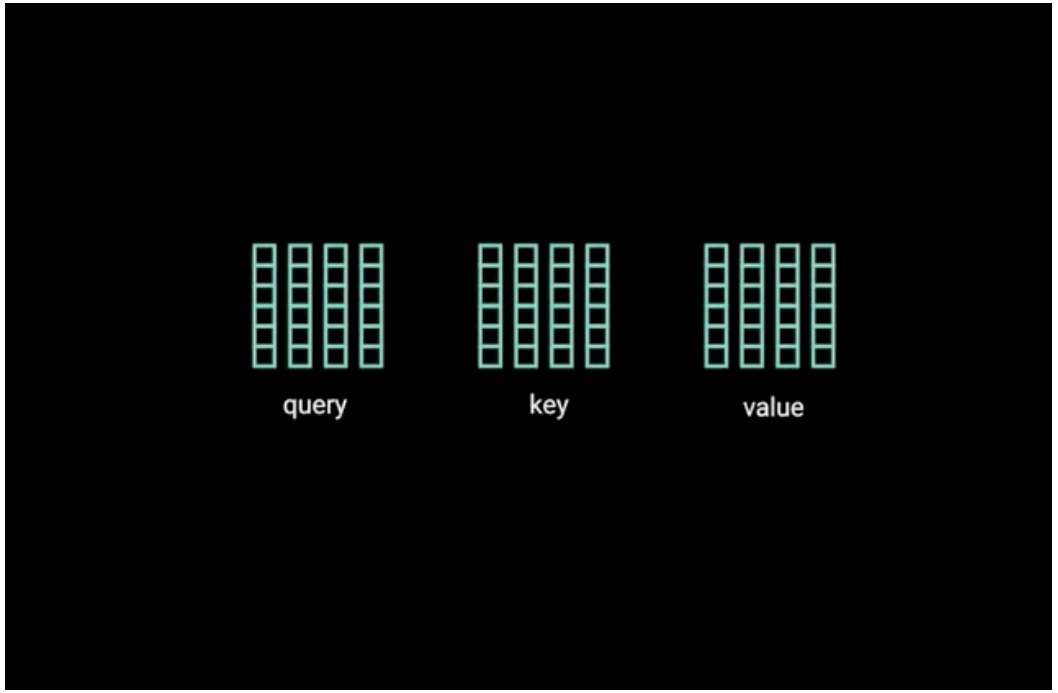
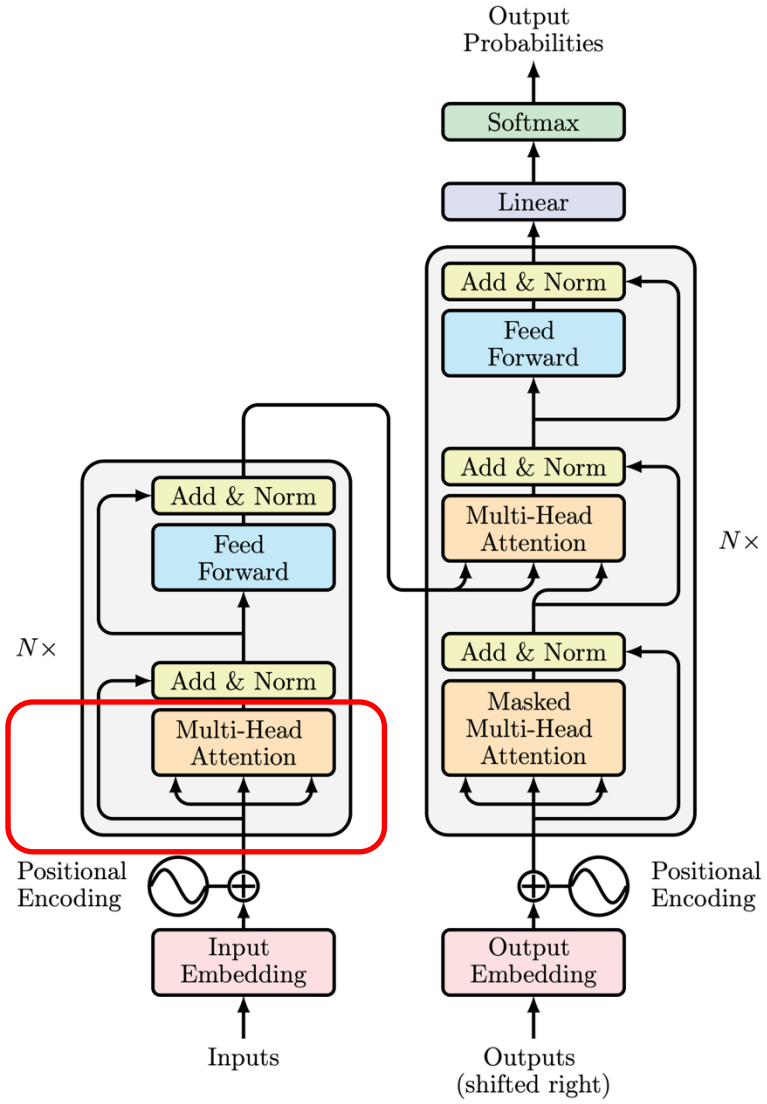
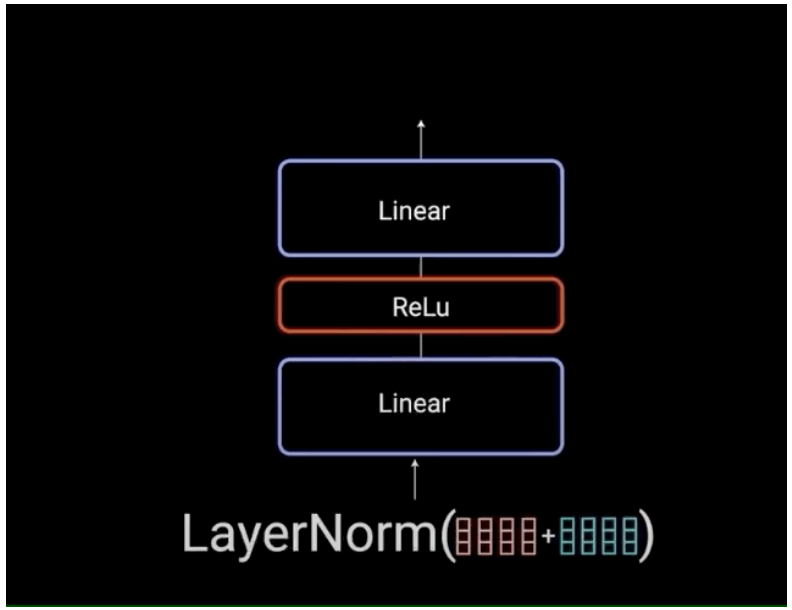


Image Credit: Vaswani, A. "Attention is all you need." Advances in Neural Information Processing Systems (2017). https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

Layer Norm [1] & Residual Connection





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

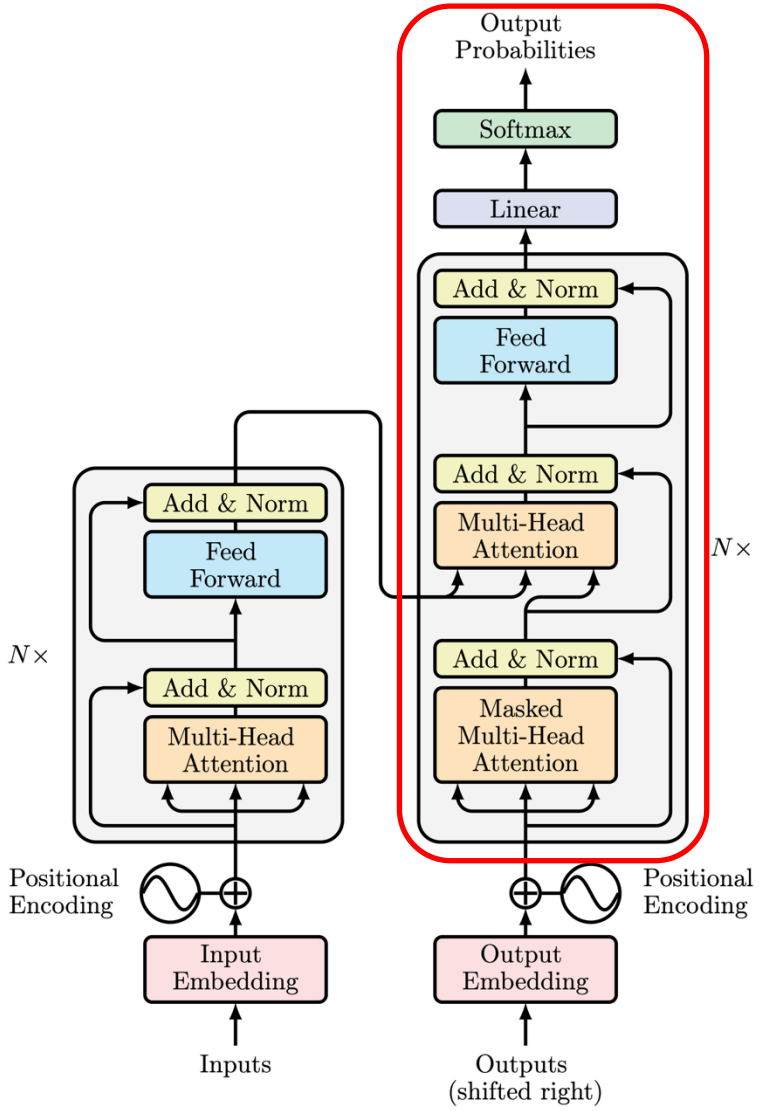
$$\sigma_i^2 = \frac{1}{K}\sum_{k=1}^{K}(x_{i,k} - \mu_i)^2$$

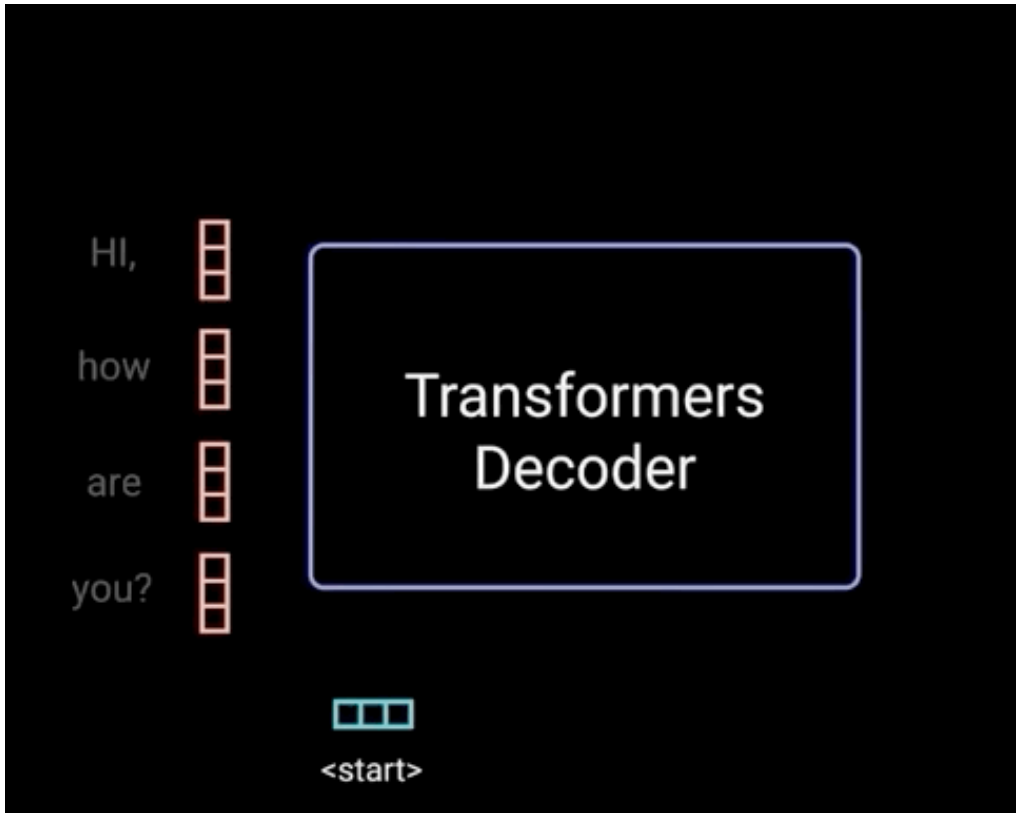
$$\hat{x}_{i,k} = \frac{x_{i,k} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}}$$

$$y_i = \gamma\hat{x}_i + \beta \equiv \mathrm{LN}_{\gamma,\beta}(x_i)$$

[1] Ba, Jimmy Lei. "Layer normalization." arXiv preprint arXiv:1607.06450 (2016). Image Credit: Vaswani, A. "Attention is all you need." Advances in Neural Information Processing Systems (2017).
https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

Decoder

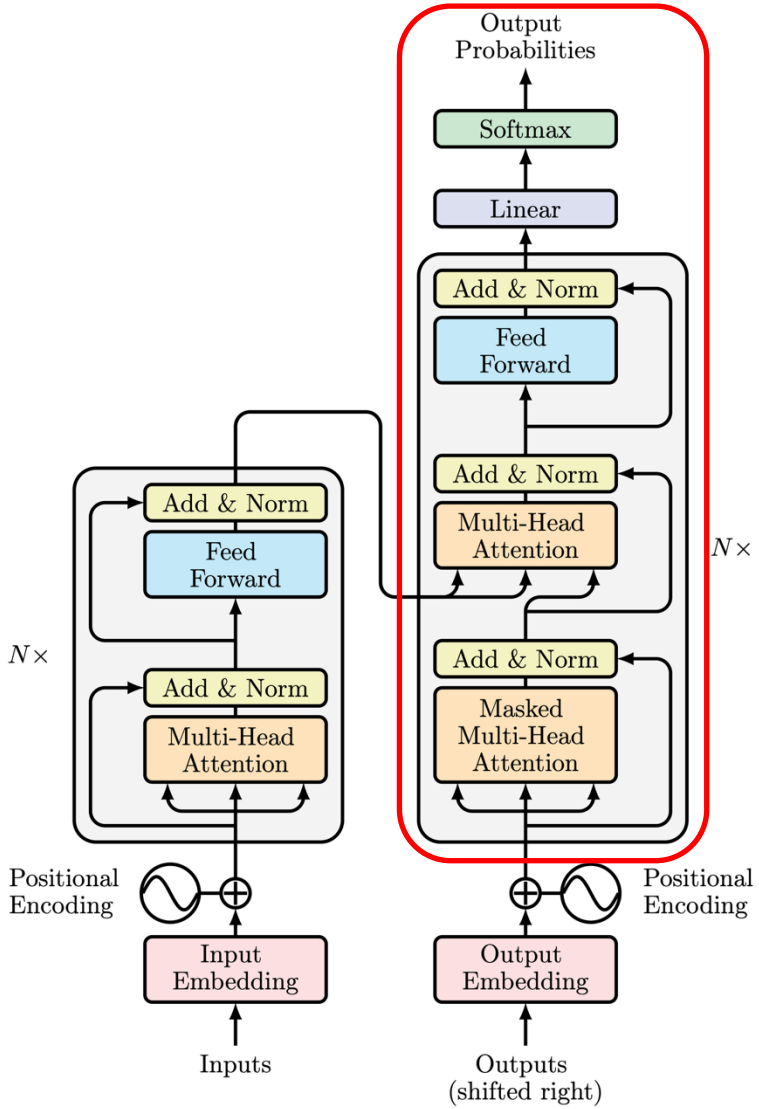


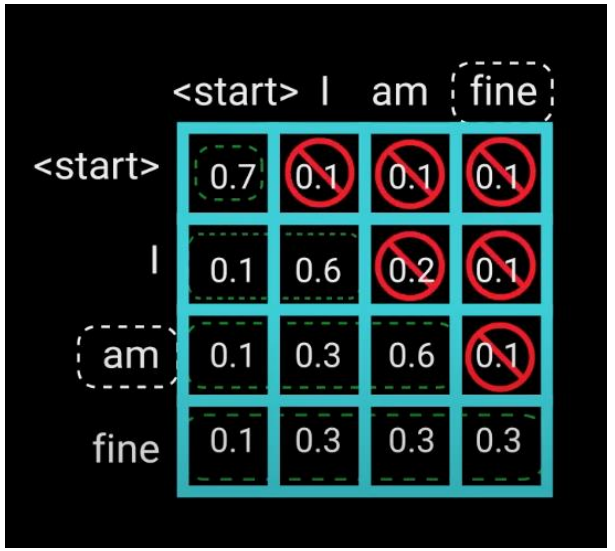


For certain applications like language models, decoder should be autoregressive!

Image Credit: Vaswani, A. "Attention is all you need." Advances in Neural Information Processing Systems (2017). https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

Masked Multi-Head Attention

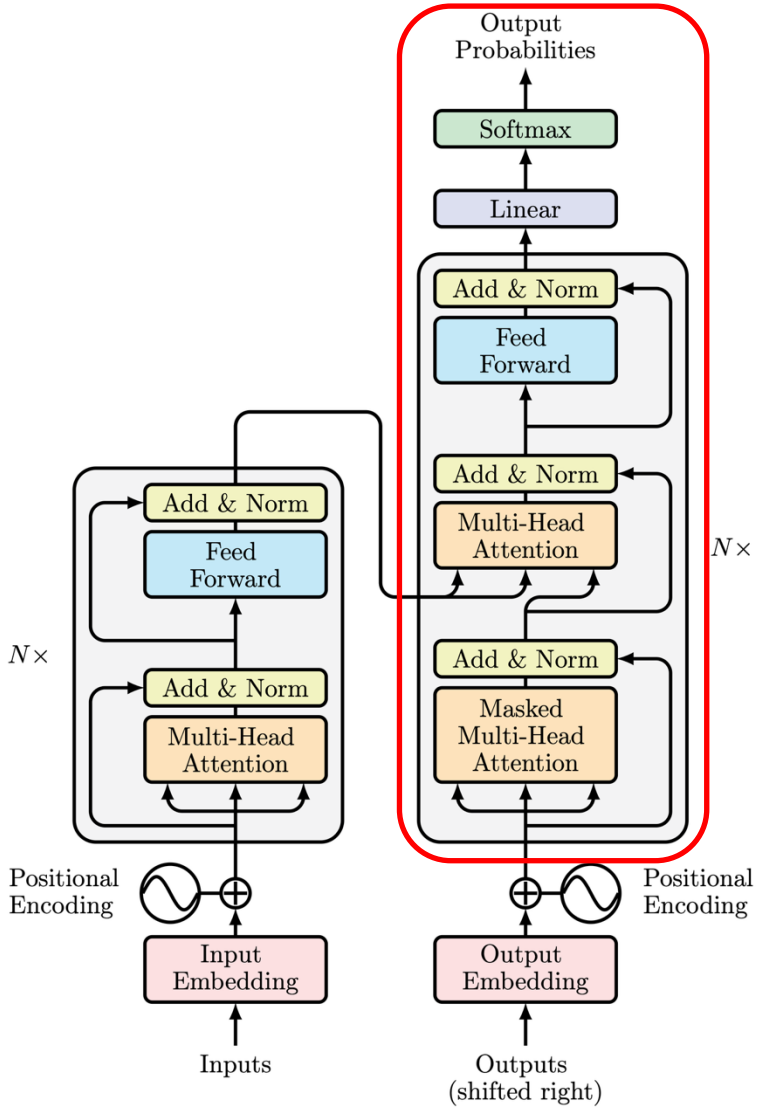


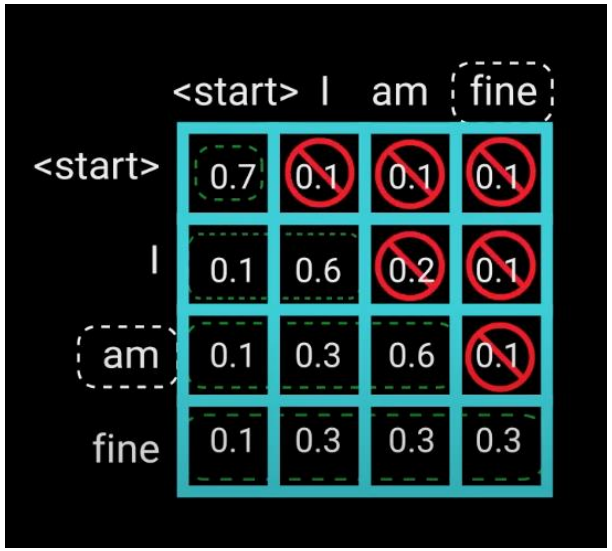


Prevent attending from future!

Image Credit: Vaswani, A. "Attention is all you need." Advances in Neural Information Processing Systems (2017). https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

Masked Multi-Head Attention





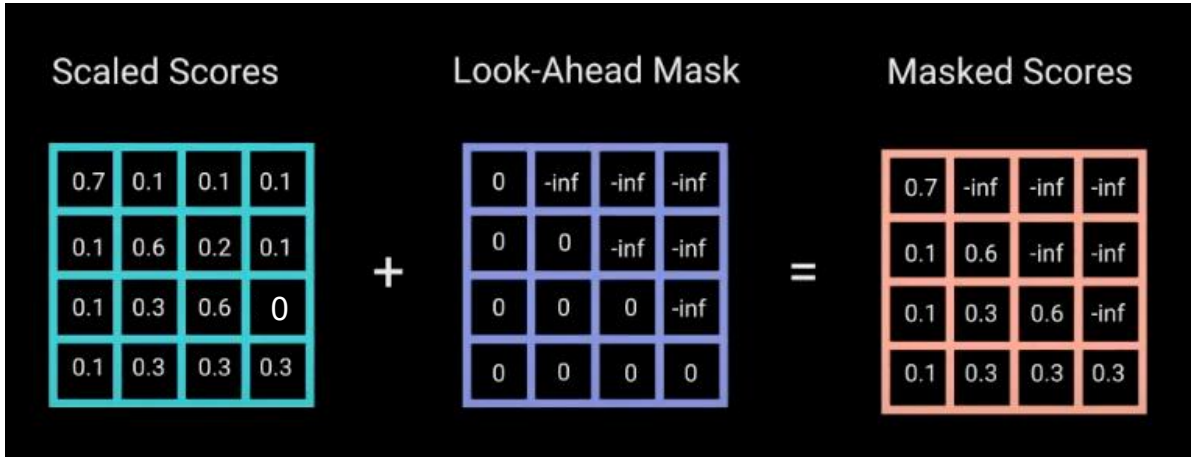
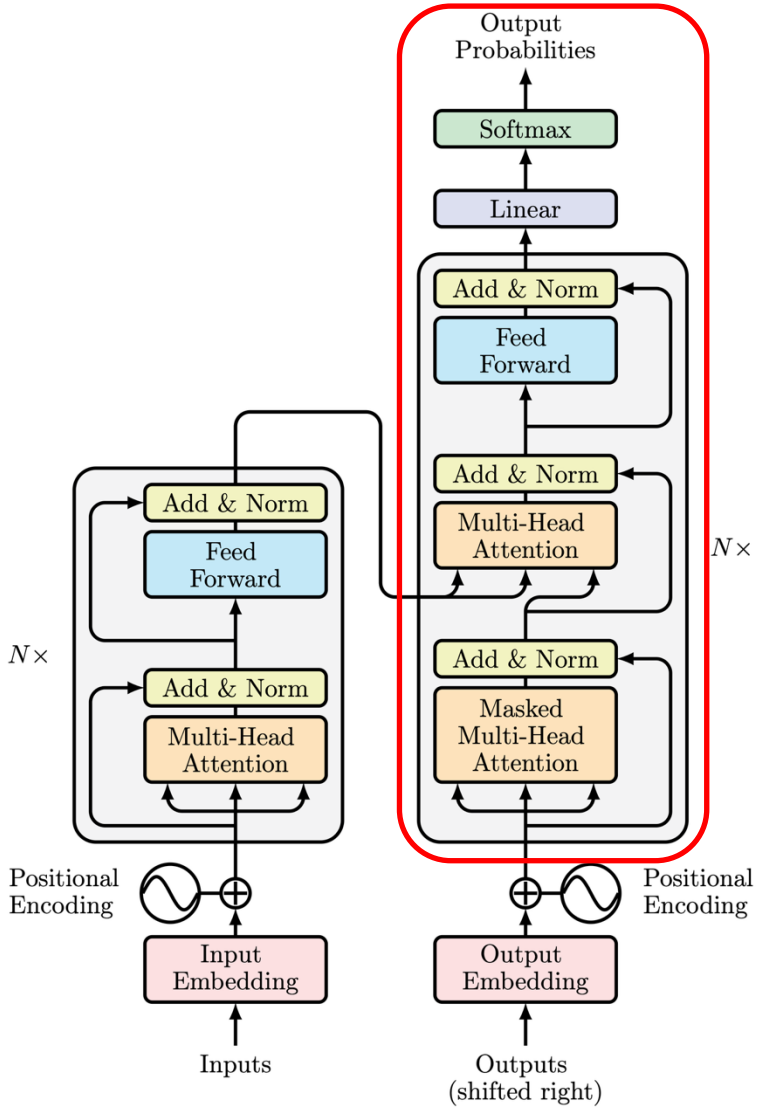
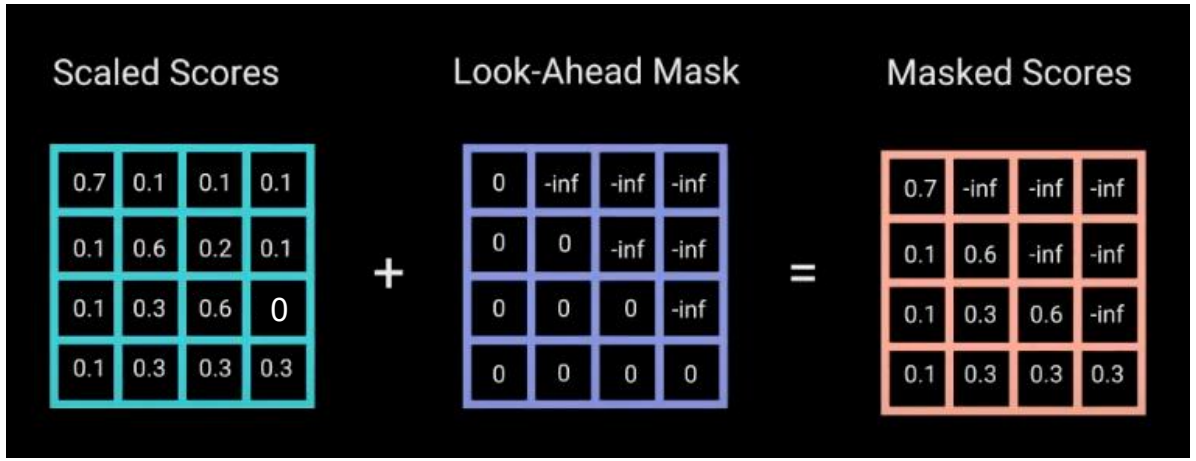


Image Credit: Vaswani, A. "Attention is all you need." Advances in Neural Information Processing Systems (2017). https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

Masked Multi-Head Attention





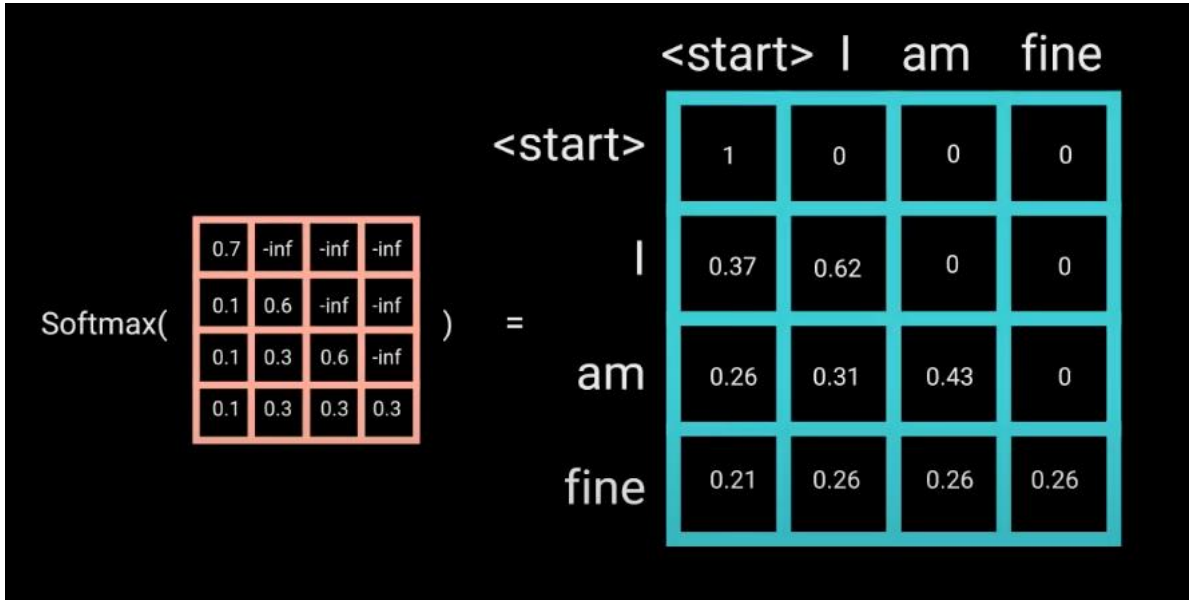
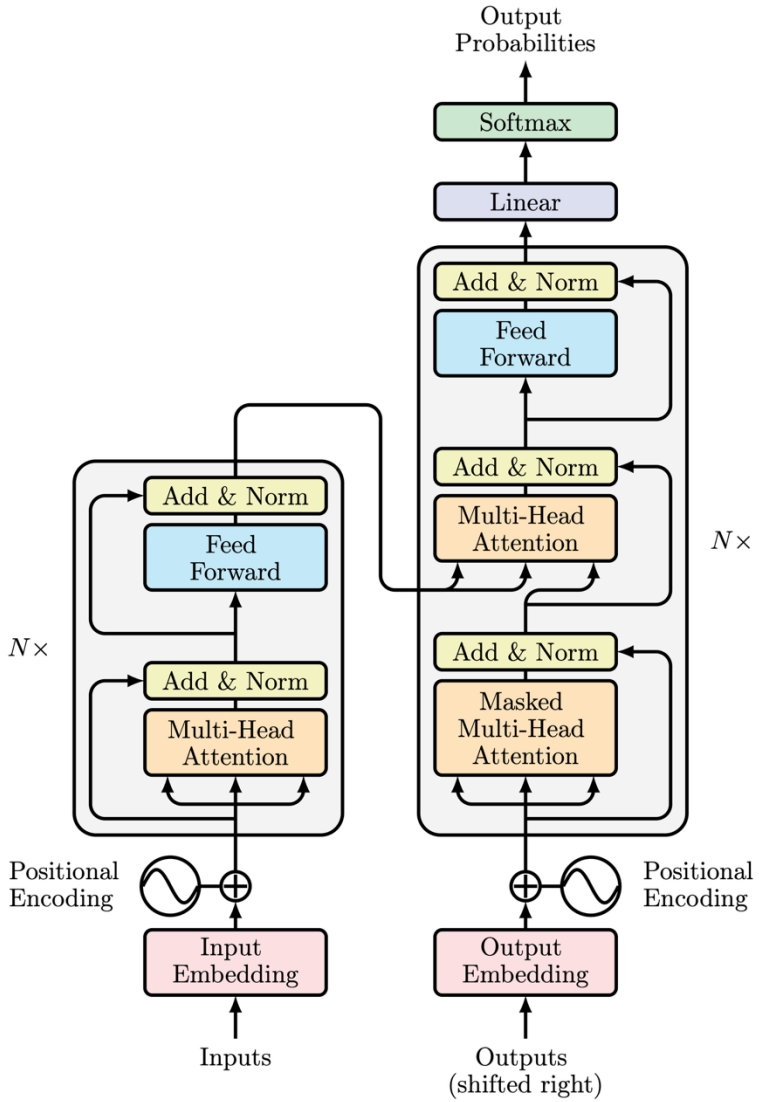


Image Credit: Vaswani, A. "Attention is all you need." Advances in Neural Information Processing Systems (2017). https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

Limitations



- O(L^2) time/memory cost for self-attention

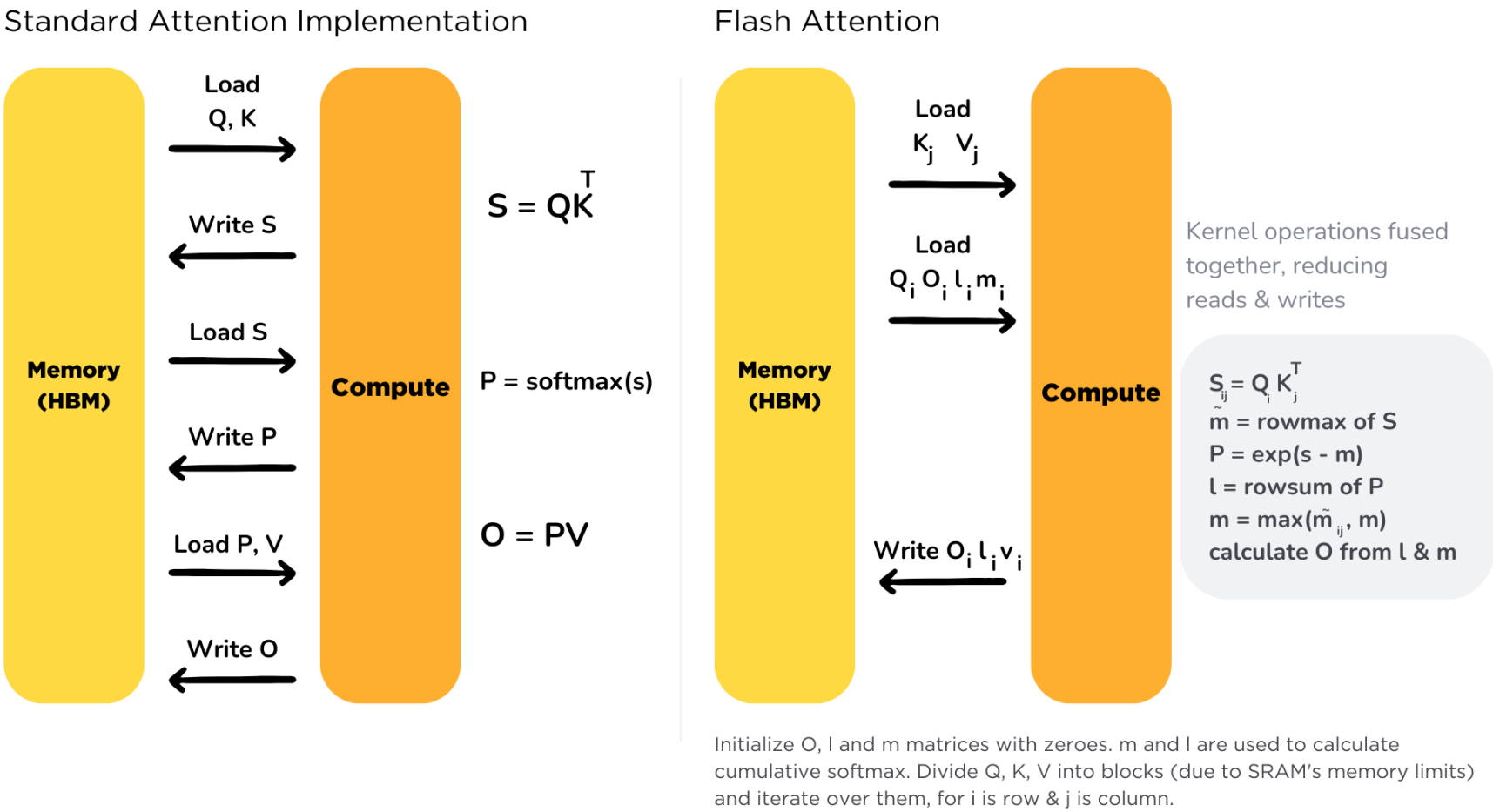
  Methods like Reformer [1] speed up attention to O(L log L) using locality-sensitive hashing techniques

- How can we incorporate prior knowledge into attention rather than having a fully connected attention?
  - Encourage sparse attention
  - Inject known graph structures
  - ……

[1] Kitaev, Nikita, Łukasz Kaiser, and Anselm Levskaya. "Reformer: The efficient transformer." arXiv preprint arXiv:2001.04451 (2020). Image Credit: Vaswani, A. "Attention is all you need." Advances in Neural Information Processing Systems (2017).

Flash Attention [1]

Flash attention accelerates attention by using on-chip static random-access memory (SRAM, small memory but fast) to reduce the IO with high bandwidth memory (HBM, large memory but slow).



[1] Dao, Tri, et al. "Flashattention: Fast and memory-efficient exact attention with io-awareness." Advances in Neural Information Processing Systems 35 (2022): 16344-16359. Image Credit: https://huggingface.co/docs/text-generation-inference/en/conceptual/flash_attention

Outline

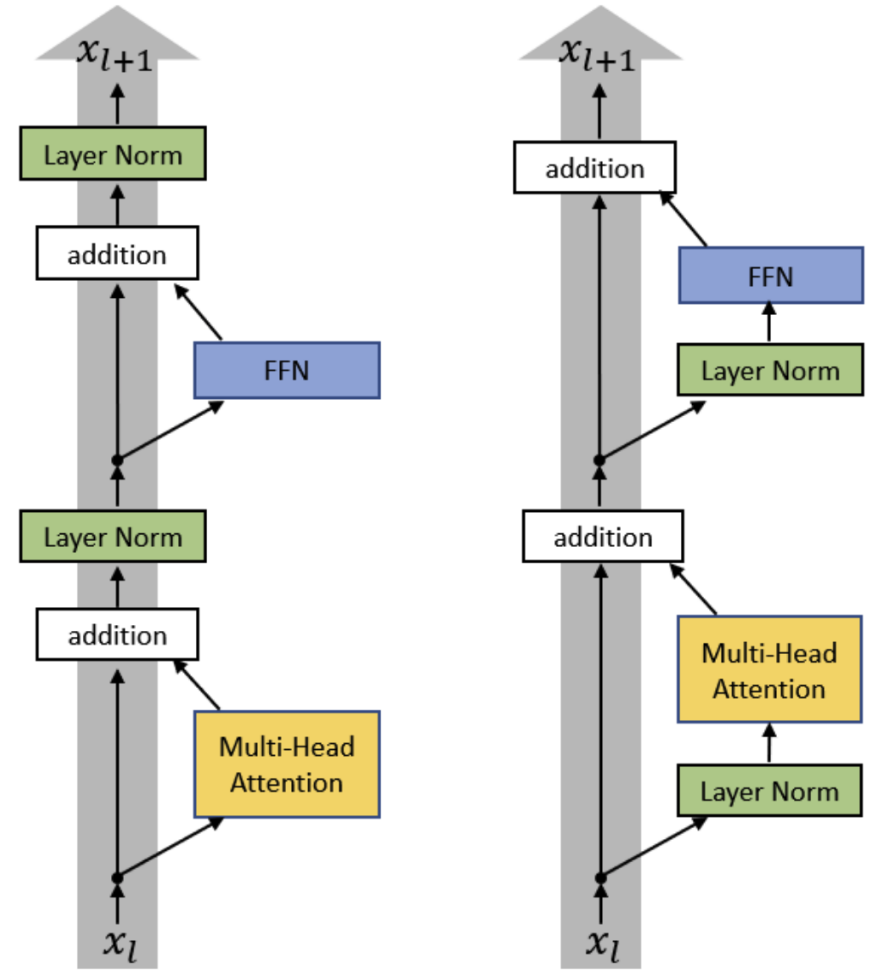
- Invariance & Equivariance Principle
  - Translation equivariance in convolutions
  - Permutation equivariance and invariance

- Models for Sets
  - DeepSets: representation theorem of permutation-invariant set functions & architecture
  - DeepSets: permutation-equivariant linear mapping & architecture

- Models for Sequences
  - Transformers
  - Positional encoding vs. Rotary Positional Embeddings (RoPE)
  - Attention & Flash Attention
  - **Pre-norm vs. post-norm**
  - Vision Transformers (ViT) & Swin Transformers

Pre-Norm vs. Post-Norm

Where to place the Layer Normalization?



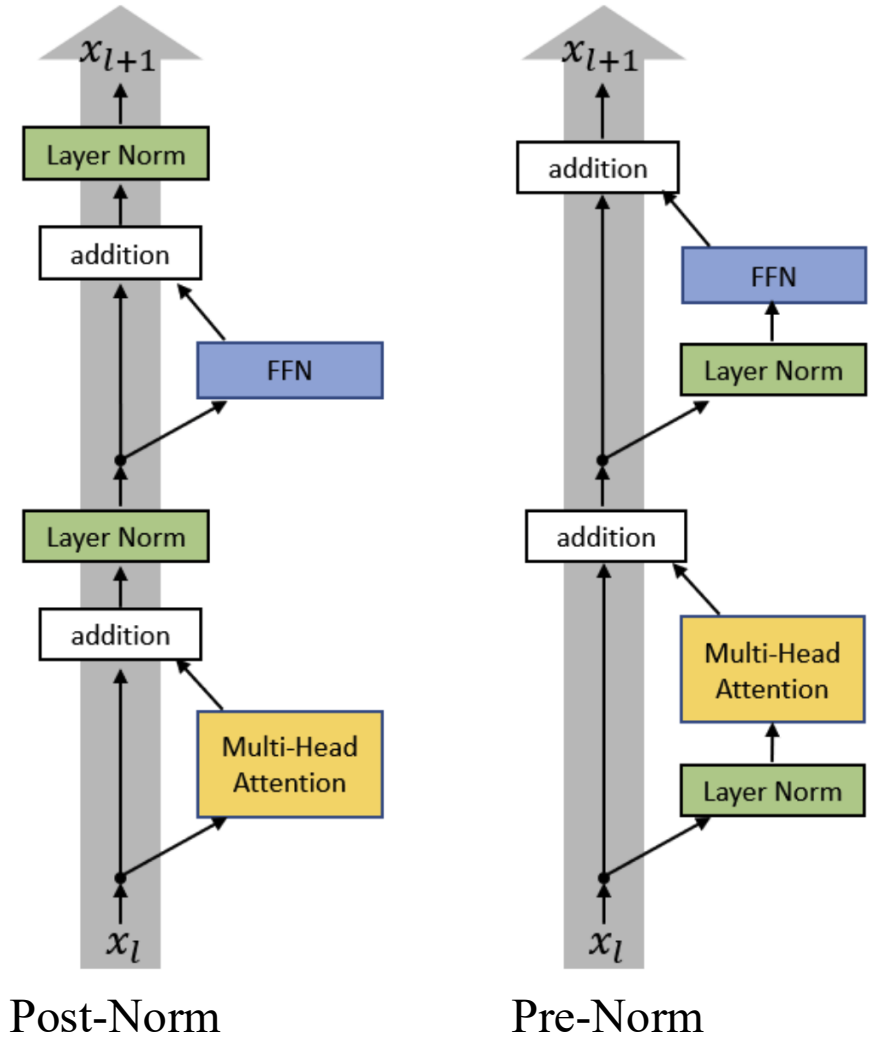
Post-Norm Pre-Norm

Image Credit: Xiong, Ruibin, et al. "On layer normalization in the transformer architecture." International Conference on Machine Learning. PMLR, 2020.

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

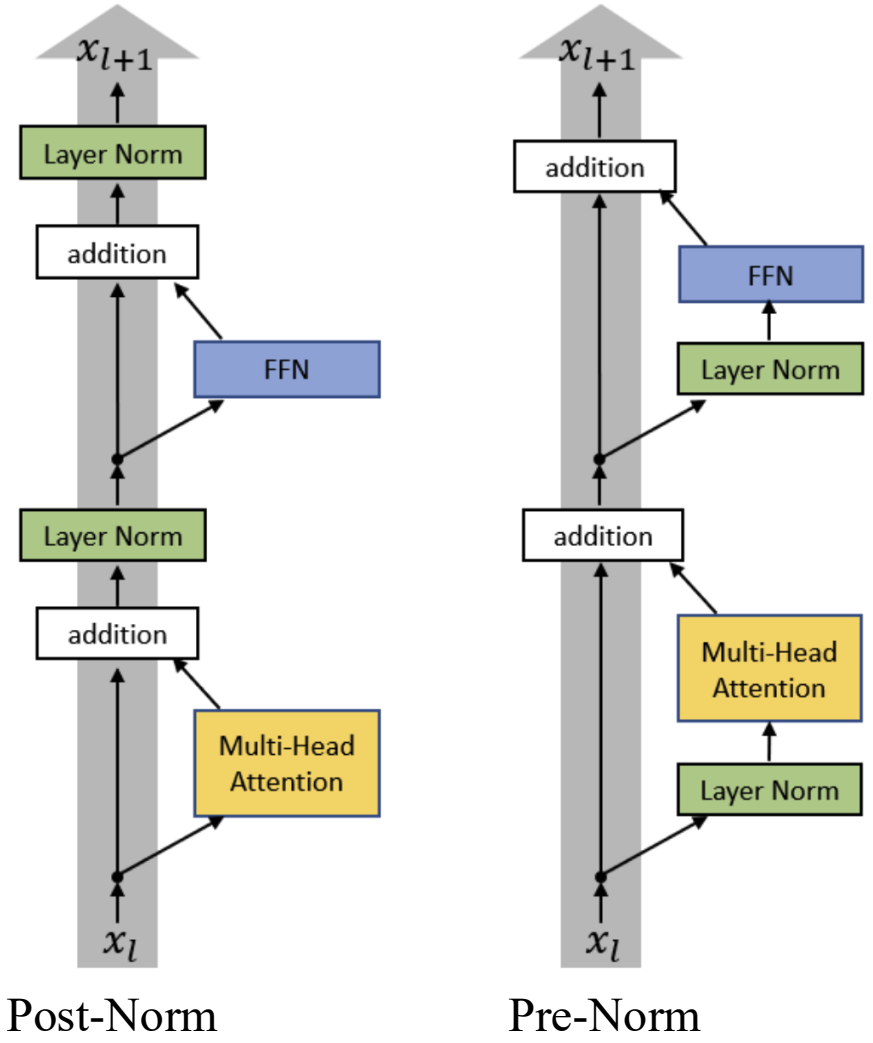
Image Credit: Xiong, Ruibin, et al. "On layer normalization in the transformer architecture." International Conference on Machine Learning. PMLR, 2020.

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

- Training the Pre-Norm Transformer does not rely on the learning rate warm-up stage and can be trained much faster than the Post-Norm



Post-Norm Pre-Norm

Image Credit: Xiong, Ruibin, et al. "On layer normalization in the transformer architecture." International Conference on Machine Learning. PMLR, 2020.

Outline

- Invariance & Equivariance Principle
  - Translation equivariance in convolutions
  - Permutation equivariance and invariance

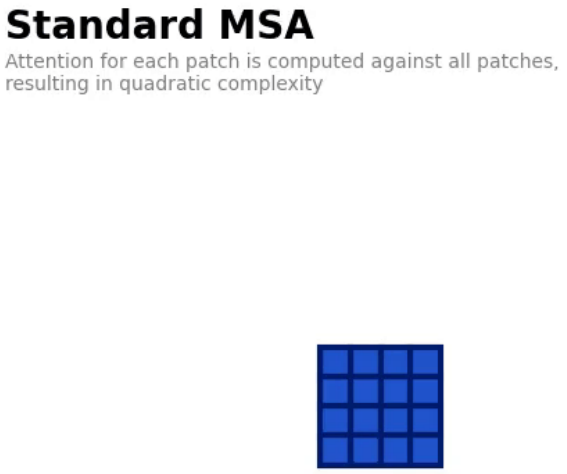
- Models for Sets
  - DeepSets: representation theorem of permutation-invariant set functions & architecture
  - DeepSets: permutation-equivariant linear mapping & architecture

- Models for Sequences
  - Transformers
  - Positional encoding vs. Rotary Positional Embeddings (RoPE)
  - Attention & Flash Attention
  - Pre-norm vs. post-norm
  - **Vision Transformers (ViT) & Swin Transformers**

Extensions: Vision Transformers [1]

[1] Dosovitskiy, Alexey, et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale." International Conference on Learning Representations. 2020. Image Credit: https://github.com/lucidrains/vit-pytorch

Extensions: Swin Transformers [1]

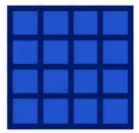


[1] Liu, Ze, et al. "Swin transformer: Hierarchical vision transformer using shifted windows." Proceedings of the IEEE/CVF international conference on computer vision. 2021. Image credit: https://towardsdatascience.com/a-comprehensive-guide-to-swin-transformer-64965f89d14c

Extensions: Swin Transformers

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

Extensions: Swin Transformers

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.

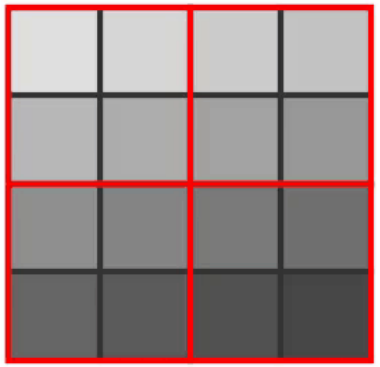
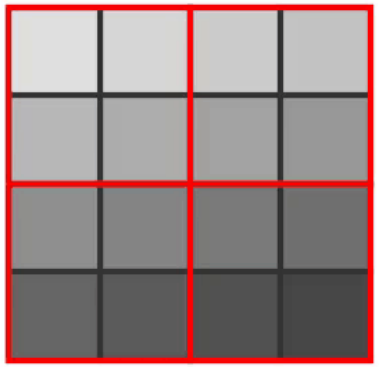


Image Credit: https://towardsdatascience.com/a-comprehensive-guide-to-swin-transformer-64965f89d14c

Extensions: Swin Transformers

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.



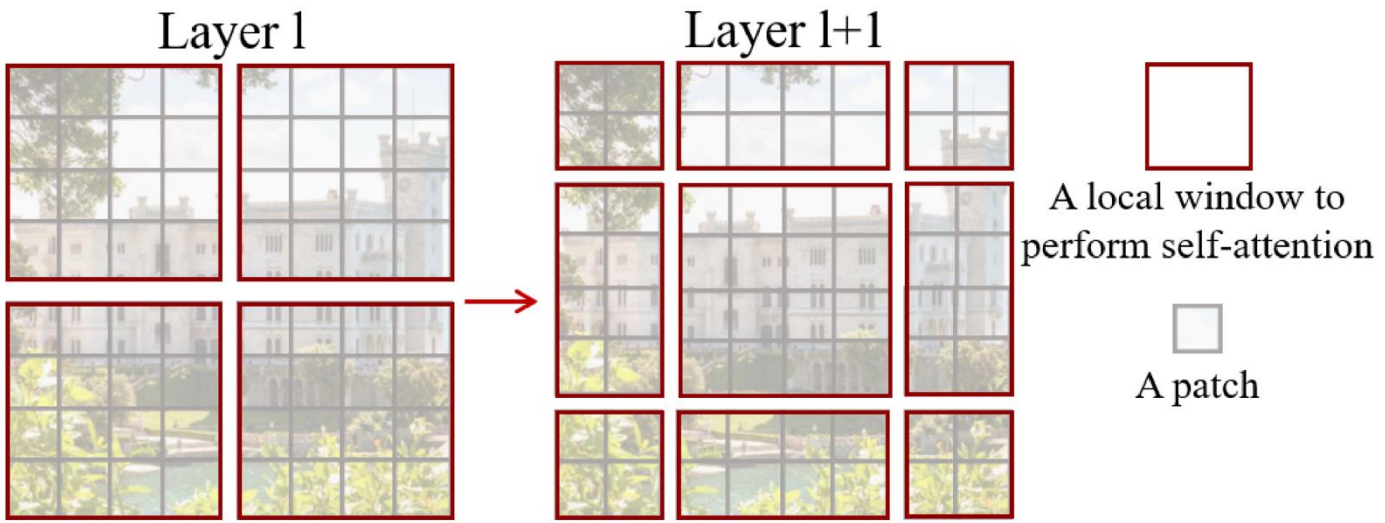


Image Credit: https://towardsdatascience.com/a-comprehensive-guide-to-swin-transformer-64965f89d14c Liu, Ze, et al. "Swin transformer: Hierarchical vision transformer using shifted windows." Proceedings of the IEEE/CVF international conference on computer vision. 2021.

Extensions: Swin Transformers

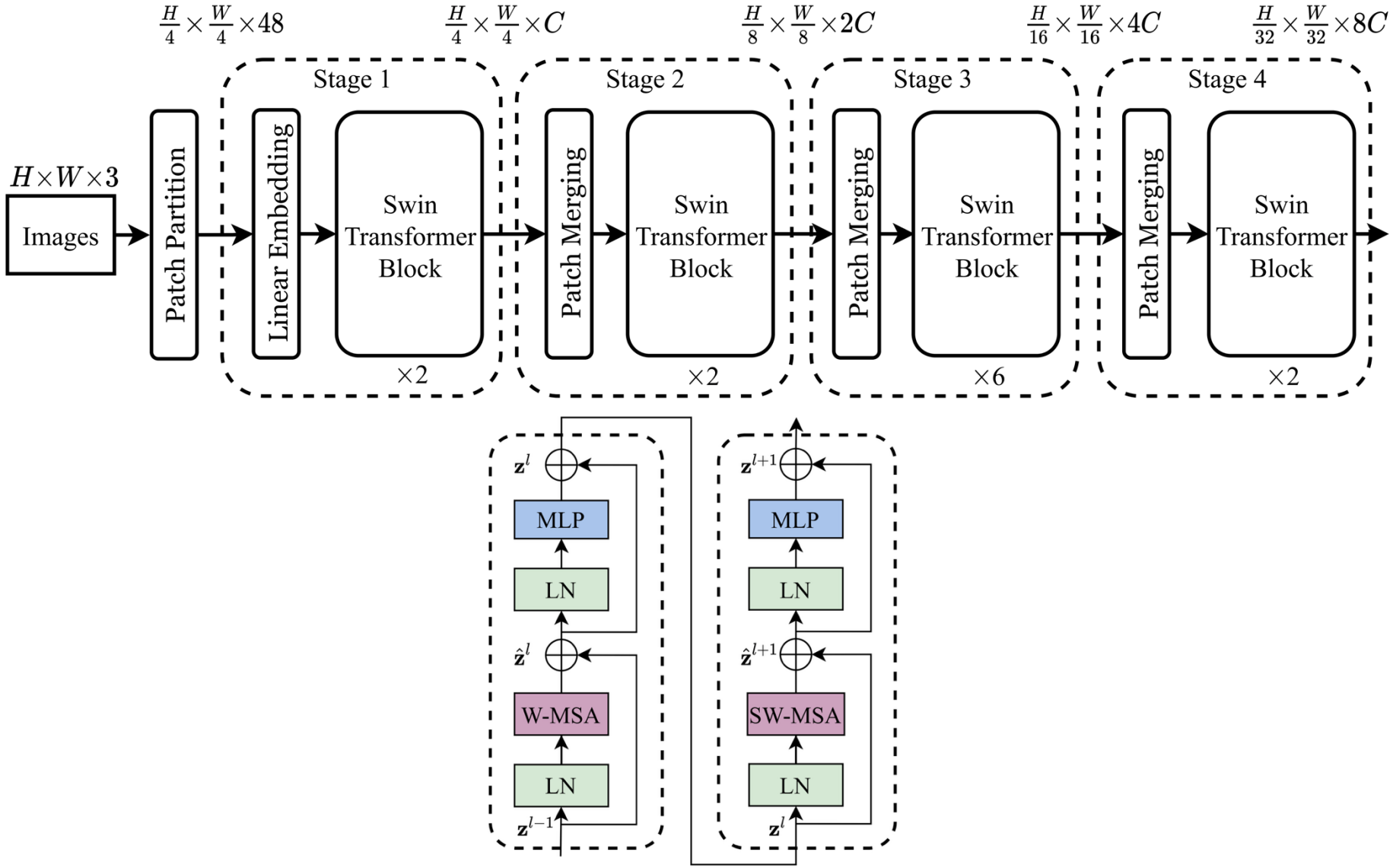


Image Credit: Liu, Ze, et al. "Swin transformer: Hierarchical vision transformer using shifted windows." Proceedings of the IEEE/CVF international conference on computer vision. 2021.

Questions?