EECE 571F: Deep Learning with Structures

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

Renjie Liao

University of British Columbia

Winter, Term 1, 2023

Course Scope

- Brief Intro to Deep Learning

- Geometric Deep Learning

  - Deep Learning Models for Sets and Sequences: Deep Sets & Transformers
  - Deep Learning Models for Graphs: Message Passing & Graph Convolution GNNs
  - Group Equivariant Deep Learning

- Probabilistic Deep Learning

  - Auto-regressive models, Large Language Models (LLMs)
  - Variational Auto-Encoders (VAEs) and Generative Adversarial Networks (GANs)
  - Energy based models (EBMs)
  - Diffusion/Score based models

Course Scope

- Brief Intro to Deep Learning

- Geometric Deep Learning

  - Deep Learning Models for Sets and Sequences: **Deep Sets & Transformers**
  - Deep Learning Models for Graphs: Message Passing & Graph Convolution GNNs
  - Group Equivariant Deep Learning

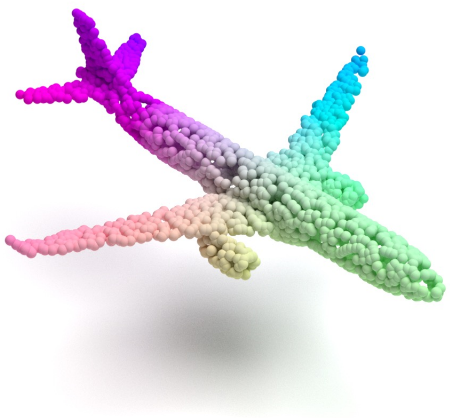
- Probabilistic Deep Learning

  - Auto-regressive models, Large Language Models (LLMs)
  - Variational Auto-Encoders (VAEs) and Generative Adversarial Networks (GANs)
  - Energy based models (EBMs)
  - Diffusion/Score based models

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

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

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

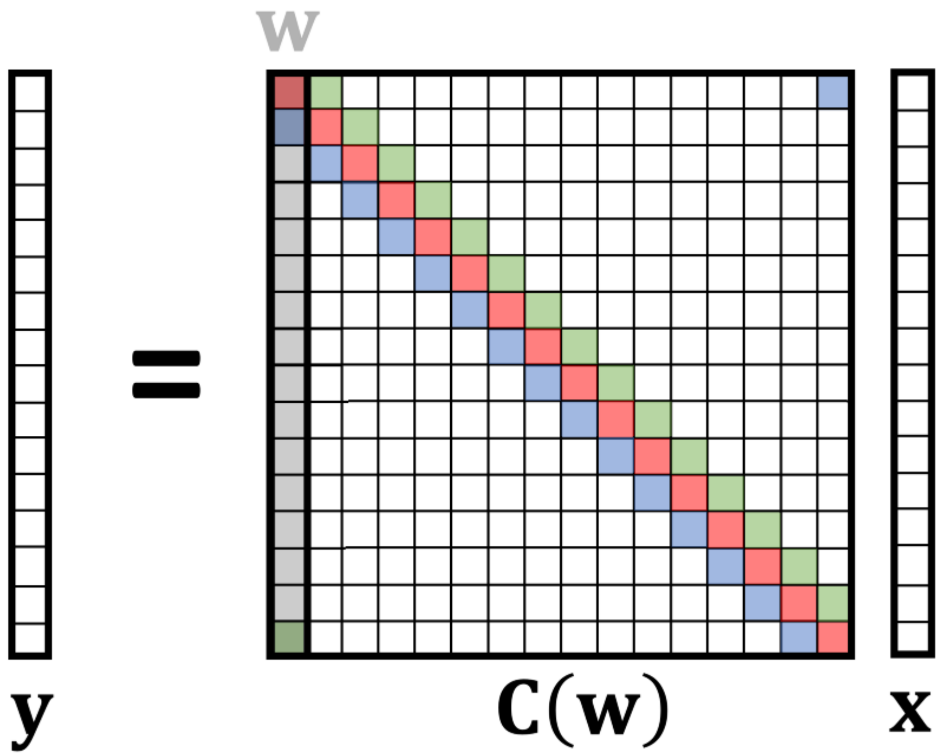


Image Credit: https://towardsdatascience.com/deriving-convolution-from-first-principles-4ff124888028

Translation/Shift Operator

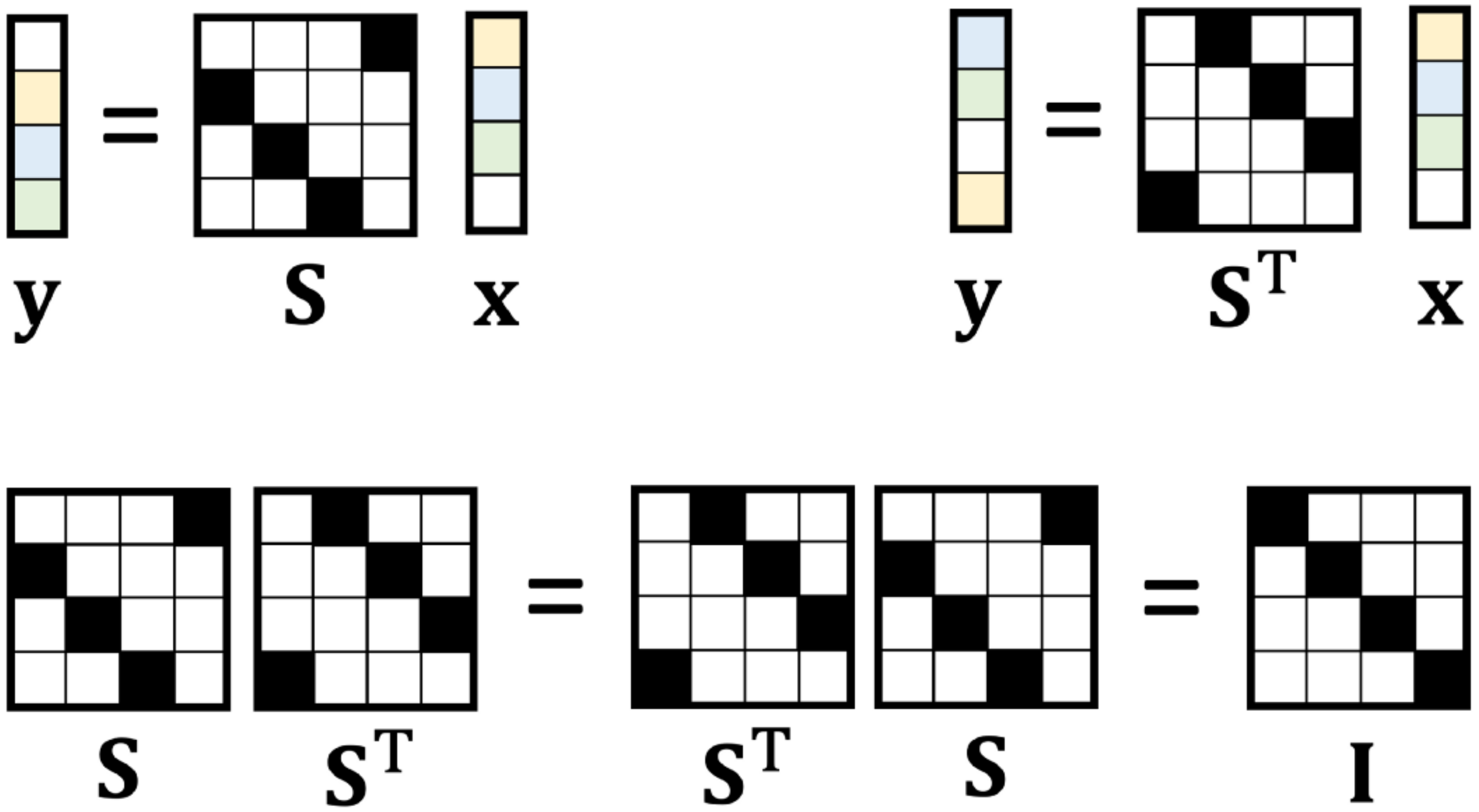


Image Credit: https://towardsdatascience.com/deriving-convolution-from-first-principles-4ff124888028

Translation/Shift Operator

Shift operator is also a circulant matrix!

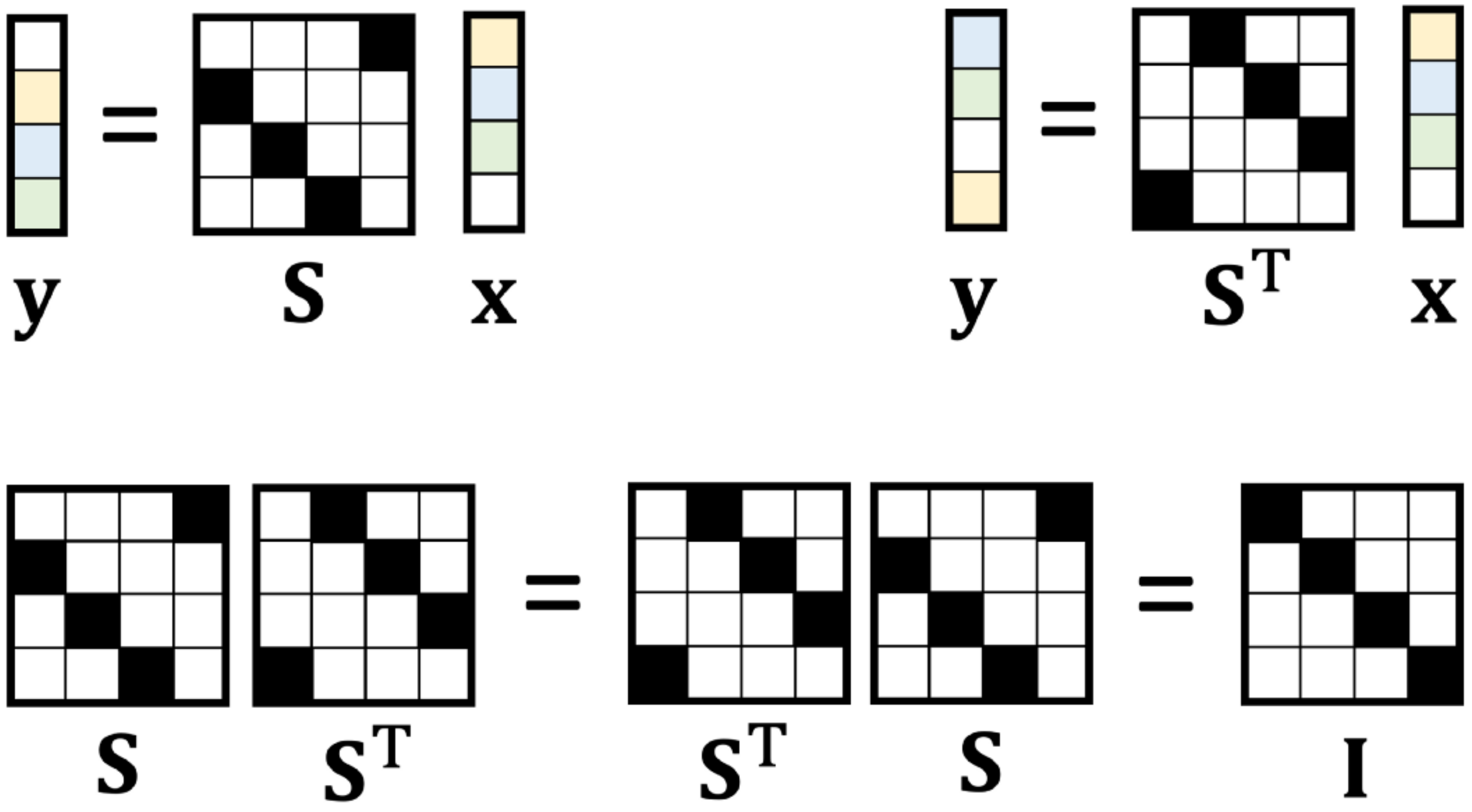


Image Credit: https://towardsdatascience.com/deriving-convolution-from-first-principles-4ff124888028

Translation/Shift Equivariance

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

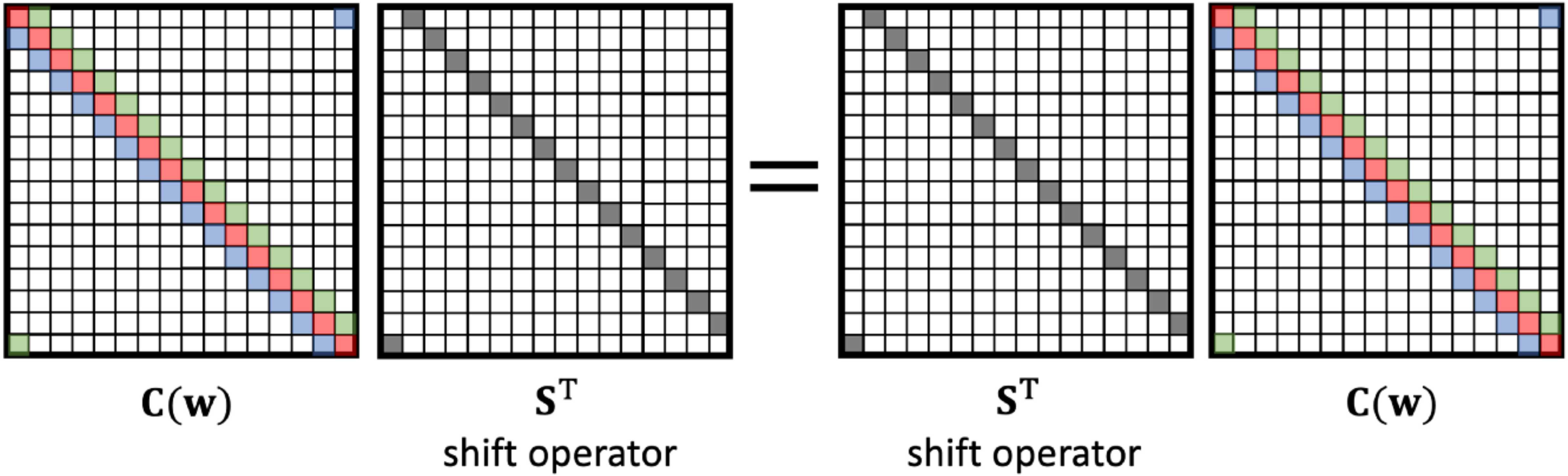
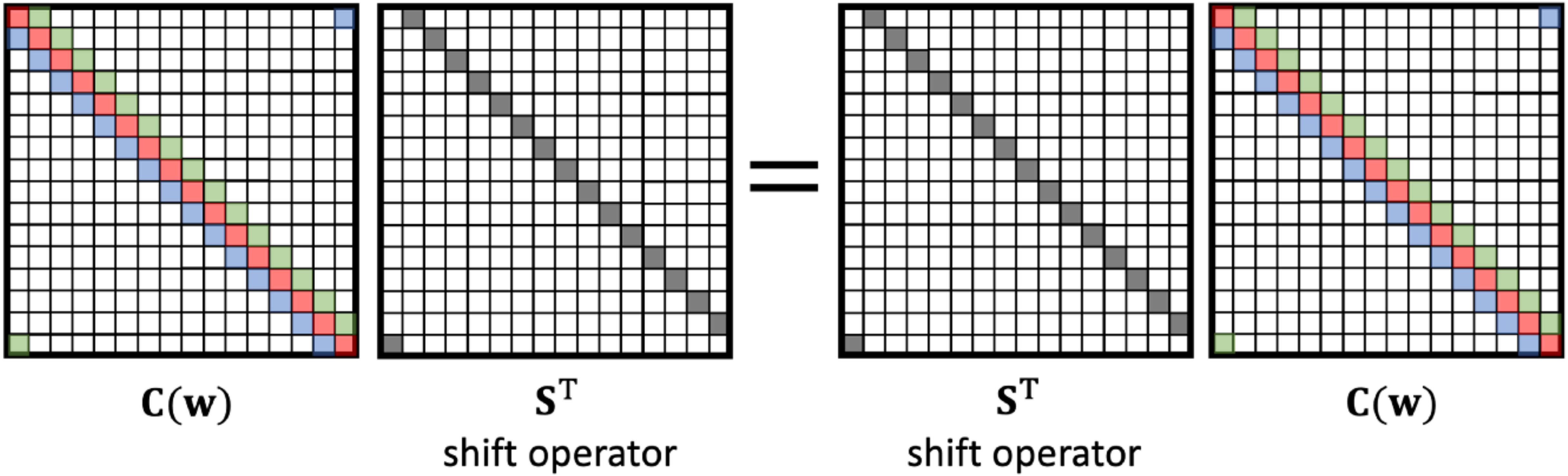


Image Credit: https://towardsdatascience.com/deriving-convolution-from-first-principles-4ff124888028

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!

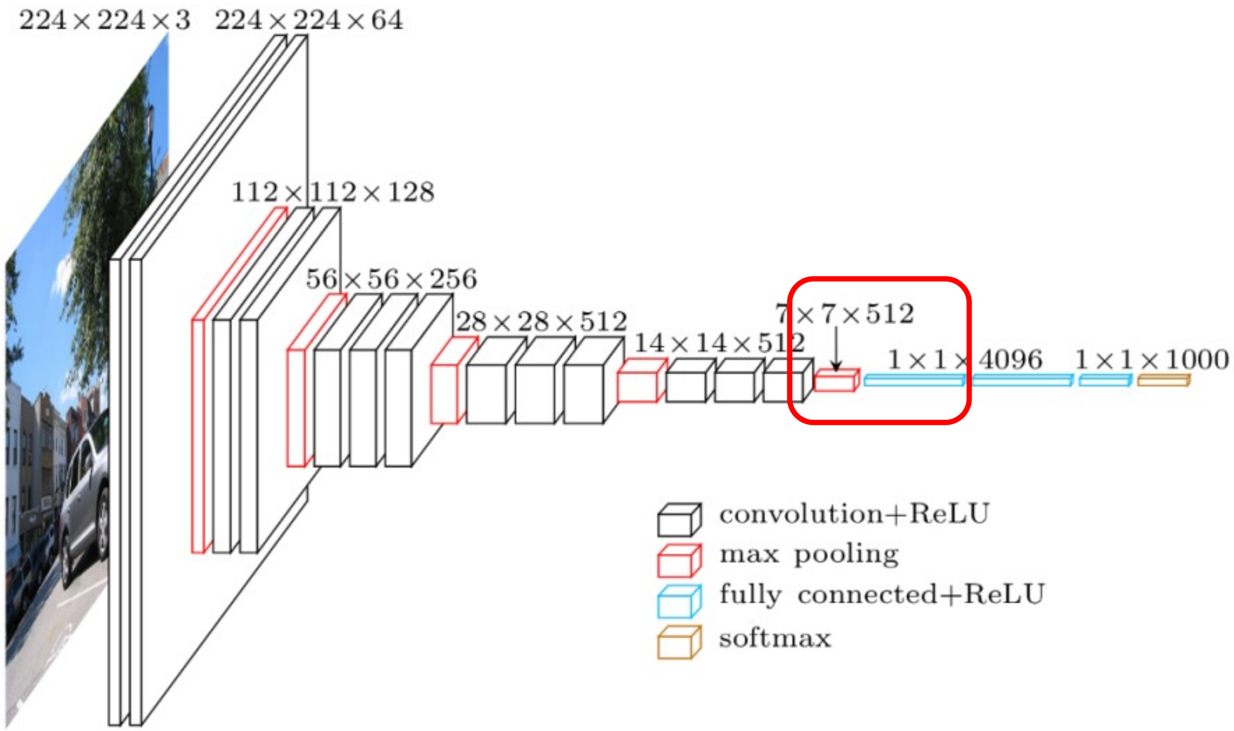
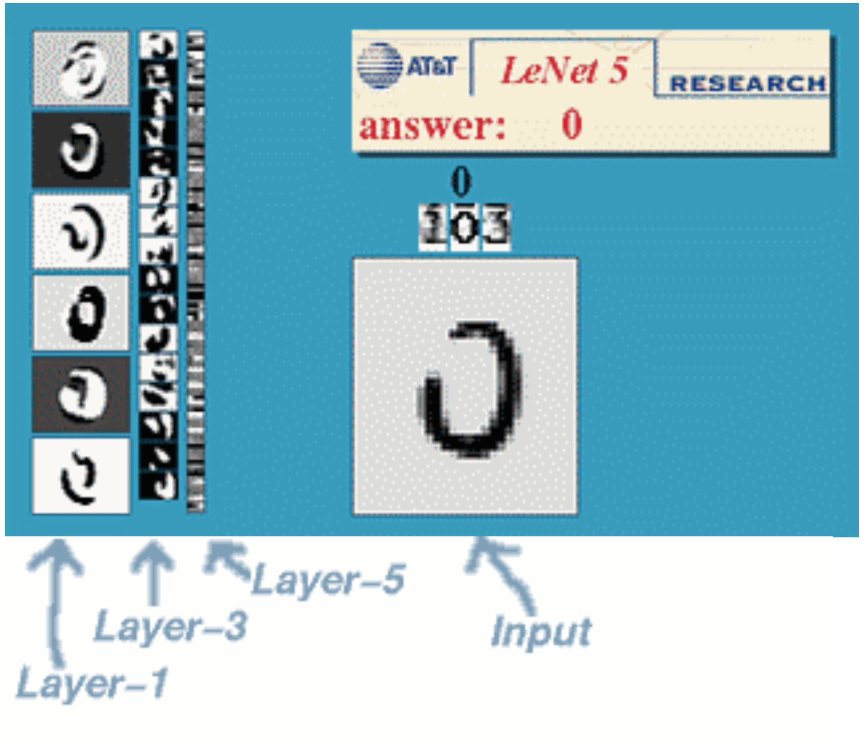


Image Credit: https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529

Translation/Shift Equivariance Invariance

Yann LeCun's LeNet Demo:



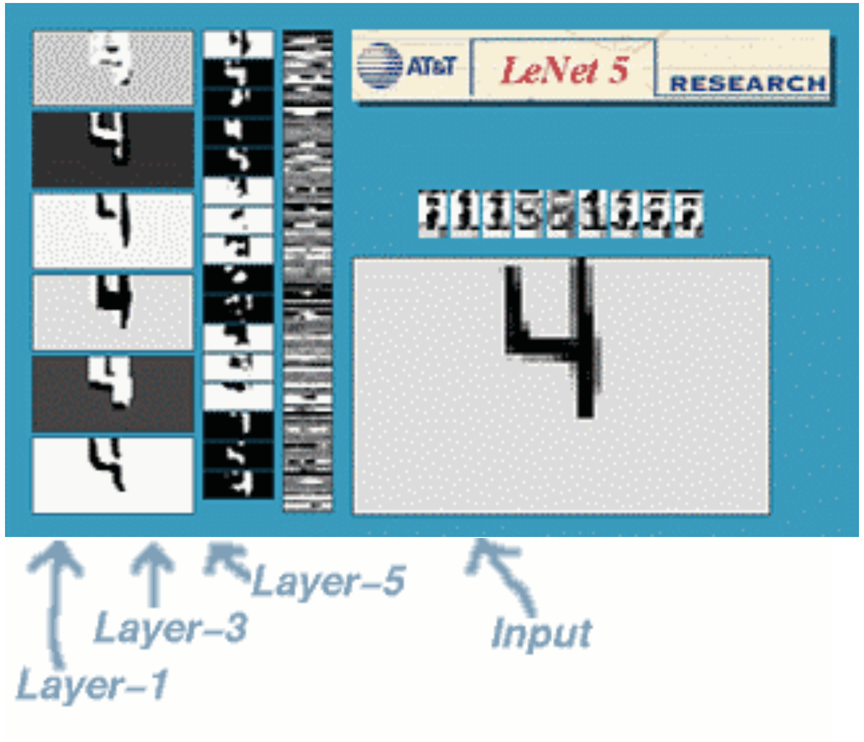
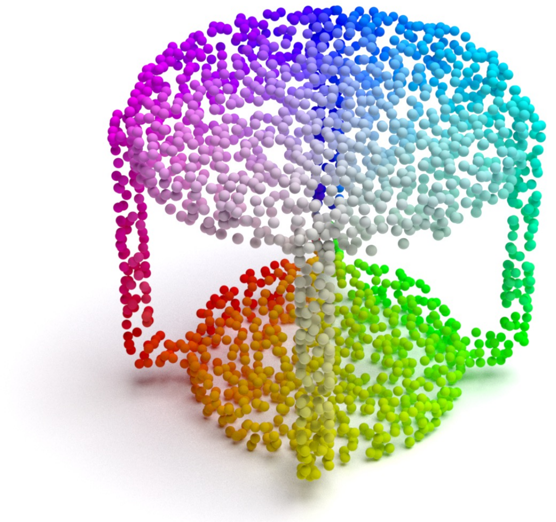


Image Credit: http://yann.lecun.com/exdb/lenet/translation.html

Permutation Invariance



Table

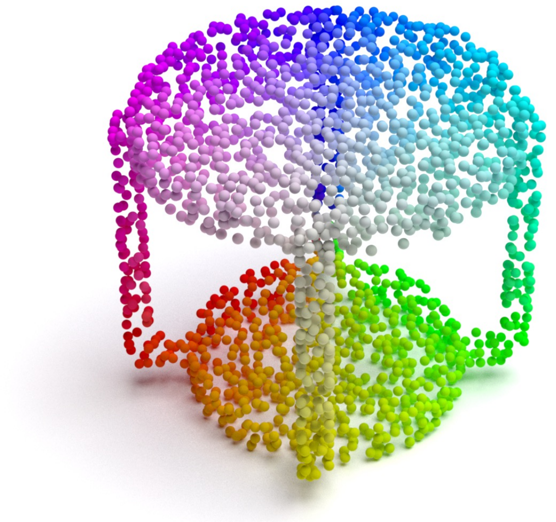
Point Clouds $X \in \mathbb{R}^{n \times 3}$

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

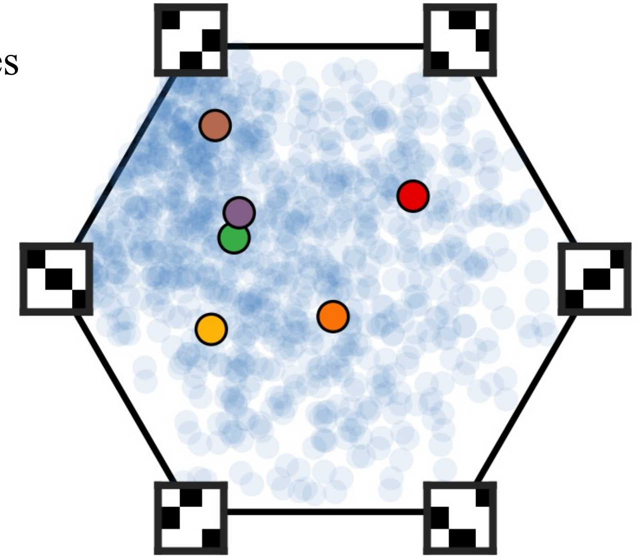
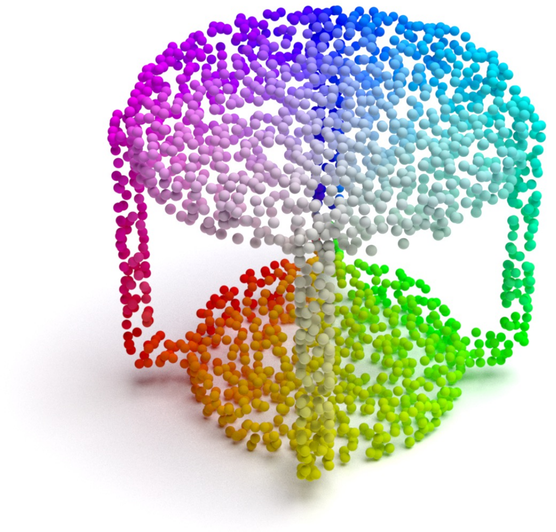


Image Credit: https://arxiv.org/pdf/1710.09508.pdf

Permutation Invariance



Table

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

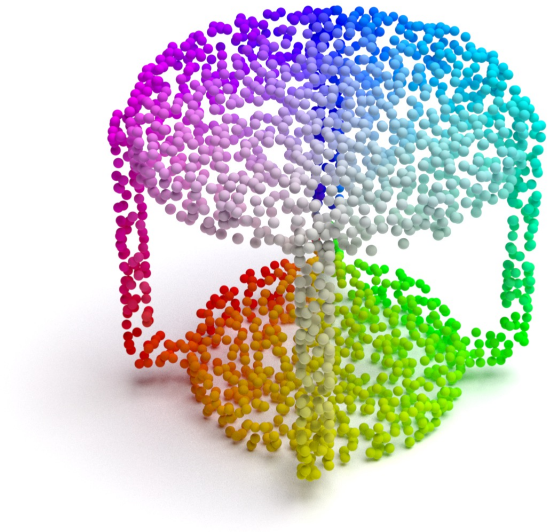
Probability of Classes $Y \in \mathbb{R}^{1 \times K}$

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

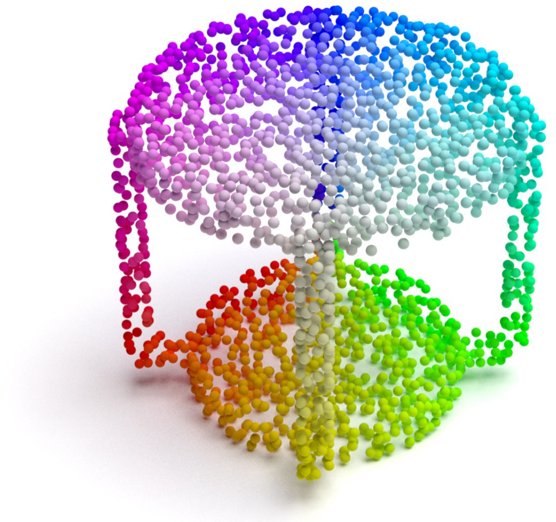
Probability of Classes $Y \in \mathbb{R}^{1 \times K}$

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 $P \in \mathbb{R}^{n \times n}$

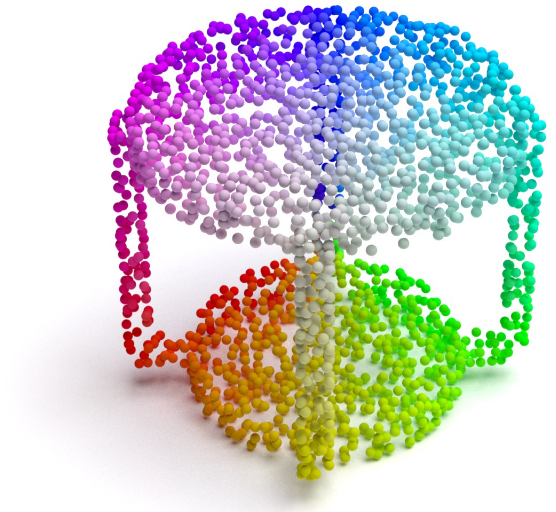
Point Representations $H \in \mathbb{R}^{n \times d}$

$$H = f(X)$$

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

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

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 [1]

Theorem 2 *A function* $f(X)$ *operating on a set* X *having elements from a countable universe, is a valid set function,* i.e., ***invariant*** *to the permutation of instances in* X*, iff it can be decomposed in the form* $\rho\left(\sum_{x\in X}\phi(x)\right)$*, for suitable transformations* ϕ *and* ρ*.*

Deep Learning for Sets

- Deep Sets [1]

Theorem 2 *A function $f(X)$ operating on a set X having elements from a countable universe, is a valid set function,* i.e., ***invariant*** *to the permutation of instances in X, iff it can be decomposed in the form $\rho\left(\sum_{x \in X} \phi(x)\right)$, for suitable transformations ϕ and ρ.*

Sketch of Proof

Sufficiency: summation is permutation invariant!

Deep Learning for Sets

- Deep Sets [1]

Theorem 2 *A function* $f(X)$ *operating on a set* X *having elements from a countable universe, is a valid set function,* i.e., ***invariant*** *to the permutation of instances in* X, *iff it can be decomposed in the form* $\rho\left(\sum_{x\in X}\phi(x)\right)$, *for suitable transformations* ϕ *and* ρ.

Sketch of Proof

Sufficiency: summation is permutation invariant!

Necessity: find an unique representation of any set and then map it!

1. Construct a mapping $c : \mathfrak{X} \to \mathbb{N}$

Deep Learning for Sets

- Deep Sets [1]

Theorem 2 *A function $f(X)$ operating on a set X having elements from a countable universe, is a valid set function,* i.e., ***invariant*** *to the permutation of instances in X, iff it can be decomposed in the form $\rho\left(\sum_{x\in X}\phi(x)\right)$, for suitable transformations ϕ and ρ.*

Sketch of Proof

Sufficiency: summation is permutation invariant!

Necessity: find an unique representation of any set and then map it!

1. Construct a mapping $c : \mathfrak{X} \rightarrow \mathbb{N}$
2. Let $\phi(x) = 4^{-c(x)}$

Deep Learning for Sets

- Deep Sets [1]

Theorem 2 *A function $f(X)$ operating on a set X having elements from a countable universe, is a valid set function,* i.e., ***invariant*** *to the permutation of instances in X, iff it can be decomposed in the form $\rho\left(\sum_{x \in X} \phi(x)\right)$, for suitable transformations ϕ and ρ.*

Sketch of Proof

Sufficiency: summation is permutation invariant!

Necessity: find an unique representation of any set and then map it!

1. Construct a mapping $c : \mathfrak{X} \to \mathbb{N}$
2. Let $\phi(x) = 4^{-c(x)}$
3. Injection $X \in 2^{\mathfrak{X}} \to \sum_{x \in X} \phi(x)$

Deep Learning for Sets

- Deep Sets [1]

Theorem 2 *A function $f(X)$ operating on a set X having elements from a countable universe, is a valid set function,* i.e., ***invariant*** *to the permutation of instances in X, iff it can be decomposed in the form $\rho\left(\sum_{x\in X}\phi(x)\right)$, for suitable transformations ϕ and ρ.*

Sketch of Proof

Sufficiency: summation is permutation invariant!

Necessity: find an unique representation of any set and then map it!

1. Construct a mapping $c : \mathfrak{X} \to \mathbb{N}$
2. Let $\phi(x) = 4^{-c(x)}$ Base 2 does not work! Why?
3. Injection $X \in 2^{\mathfrak{X}} \to \sum_{x\in X} \phi(x)$

Deep Learning for Sets

- Deep Sets [1]

Invariant Architecture

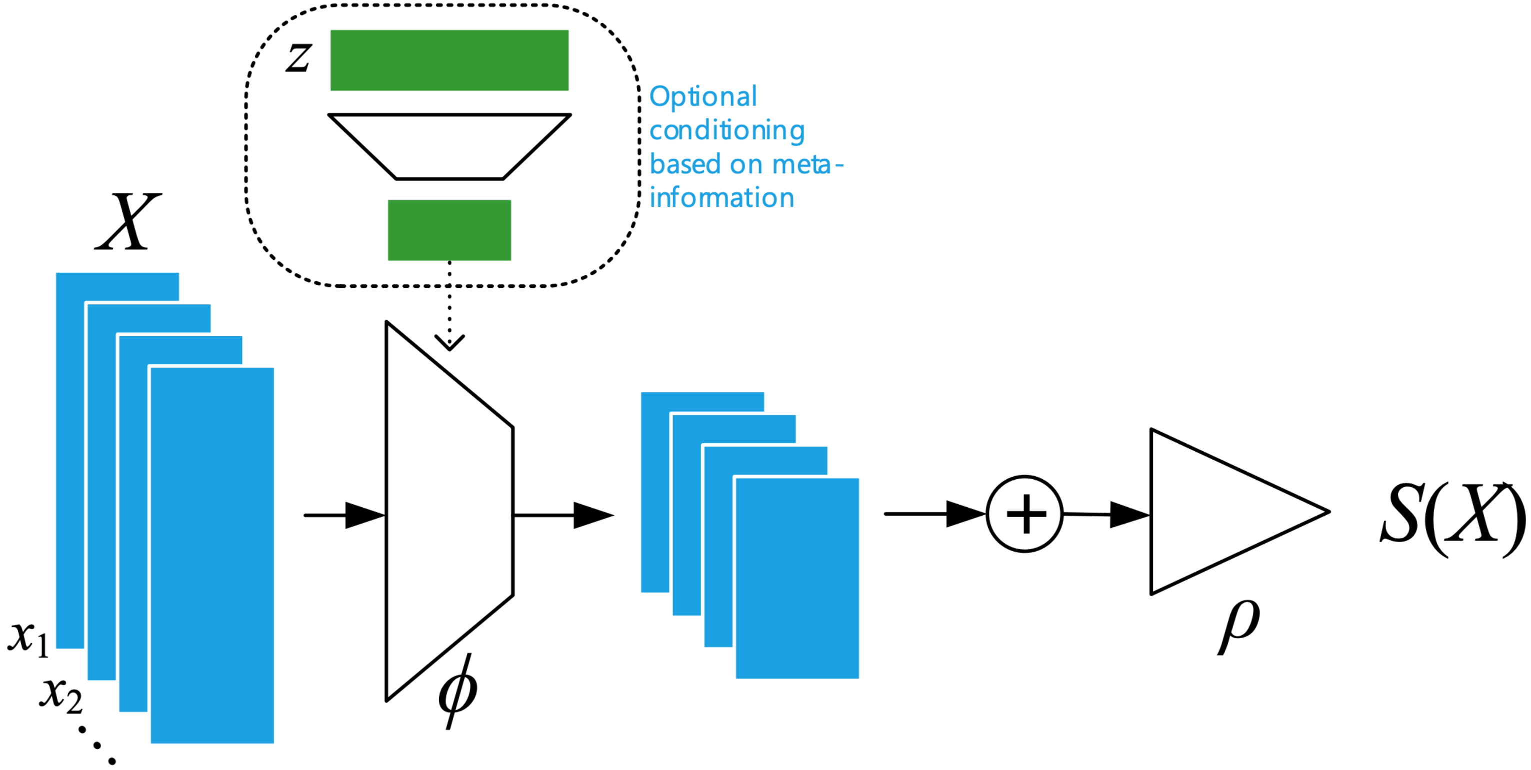


Image Credit: [1]

Deep Learning for Sets

- Deep Sets [1] $\mathbf{f}_\Theta(\mathbf{x}) \doteq \boldsymbol{\sigma}(\Theta\mathbf{x}) \quad \Theta \in \mathbb{R}^{M\times M}$

Lemma 3 *The function* $\mathbf{f}_\Theta : \mathbb{R}^M \to \mathbb{R}^M$ *defined above is permutation* ***equivariant*** *iff all the off-diagonal elements of* Θ *are tied together and all the diagonal elements are equal as well. That is,*

$\Theta = \lambda\mathbf{I} + \gamma\left(\mathbf{1}\mathbf{1}^\mathsf{T}\right) \qquad \lambda, \gamma \in \mathbb{R} \quad \mathbf{1} = [1, \ldots, 1]^\mathsf{T} \in \mathbb{R}^M \qquad \mathbf{I} \in \mathbb{R}^{M\times M}$ *is the identity matrix*

Deep Learning for Sets

- Deep Sets [1] $\mathbf{f}_{\Theta}(\mathbf{x}) \doteq \boldsymbol{\sigma}(\Theta\mathbf{x}) \quad \Theta \in \mathbb{R}^{M \times M}$

Lemma 3 *The function* $\mathbf{f}_{\Theta} : \mathbb{R}^M \to \mathbb{R}^M$ *defined above is permutation* ***equivariant*** *iff all the off-diagonal elements of* Θ *are tied together and all the diagonal elements are equal as well. That is,*

$$\Theta = \lambda \mathbf{I} + \gamma\left(\mathbf{1}\mathbf{1}^{\mathsf{T}}\right) \qquad \lambda, \gamma \in \mathbb{R} \quad \mathbf{1} = [1, \ldots, 1]^{\mathsf{T}} \in \mathbb{R}^M \qquad \mathbf{I} \in \mathbb{R}^{M \times M}$$ *is the identity matrix*

Sketch of Proof

Permutation Equivariance $\sigma(\Theta\pi\mathbf{x}) = \pi\sigma(\Theta\mathbf{x})$ (w. element-wise bijective nonlinearity) reduces to $\pi\Theta\mathbf{x} = \Theta\pi\mathbf{x}$

Deep Learning for Sets

- Deep Sets [1] $\mathbf{f}_\Theta(\mathbf{x}) \doteq \boldsymbol{\sigma}(\Theta\mathbf{x}) \quad \Theta \in \mathbb{R}^{M\times M}$

Lemma 3 *The function* $\mathbf{f}_\Theta : \mathbb{R}^M \to \mathbb{R}^M$ *defined above is permutation* ***equivariant*** *iff all the off-diagonal elements of* Θ *are tied together and all the diagonal elements are equal as well. That is,*

$$\Theta = \lambda\mathbf{I} + \gamma\,(\mathbf{1}\mathbf{1}^\mathsf{T}) \qquad \lambda, \gamma \in \mathbb{R} \quad \mathbf{1} = [1, \ldots, 1]^\mathsf{T} \in \mathbb{R}^M \qquad \mathbf{I} \in \mathbb{R}^{M\times M} \textit{ is the identity matrix}$$

Sketch of Proof

Permutation Equivariance $\sigma(\Theta\pi\mathbf{x}) = \pi\sigma(\Theta\mathbf{x})$ (w. element-wise bijective nonlinearity) reduces to $\pi\Theta\mathbf{x} = \Theta\pi\mathbf{x}$

Sufficiency: Θ is commutable with permutation matrix

Deep Learning for Sets

- Deep Sets [1] $\mathbf{f}_\Theta(\mathbf{x}) \doteq \boldsymbol{\sigma}(\Theta\mathbf{x}) \quad \Theta \in \mathbb{R}^{M\times M}$

Lemma 3 *The function $\mathbf{f}_\Theta : \mathbb{R}^M \to \mathbb{R}^M$ defined above is permutation* ***equivariant*** *iff all the off-diagonal elements of Θ are tied together and all the diagonal elements are equal as well. That is,*

$$\Theta = \lambda\mathbf{I} + \gamma\left(\mathbf{1}\mathbf{1}^\mathsf{T}\right) \qquad \lambda, \gamma \in \mathbb{R} \quad \mathbf{1} = [1, \ldots, 1]^\mathsf{T} \in \mathbb{R}^M \qquad \mathbf{I} \in \mathbb{R}^{M\times M} \textit{is the identity matrix}$$

Sketch of Proof

Permutation Equivariance $\sigma(\Theta\pi\mathbf{x}) = \pi\sigma(\Theta\mathbf{x})$ (w. element-wise bijective nonlinearity) reduces to $\pi\Theta\mathbf{x} = \Theta\pi\mathbf{x}$

Sufficiency: Θ is commutable with permutation matrix

Necessity: consider a special permutation (i.e., transposition / swap) $\pi_{i,j}^\top = \pi_{i,j}^{-1} = \pi_{j,i}$

1. All diagonal elements are identical

$$\pi_{k,l}\Theta = \Theta\pi_{k,l} \;\Rightarrow\; \pi_{k,l}\Theta\pi_{l,k} = \Theta \;\Rightarrow\; (\pi_{k,l}\Theta\pi_{l,k})_{l,l} = \Theta_{l,l} \;\Rightarrow\; \Theta_{k,k} = \Theta_{l,l}$$

Deep Learning for Sets

- Deep Sets [1] $\mathbf{f}_\Theta(\mathbf{x}) \doteq \boldsymbol{\sigma}(\Theta\mathbf{x}) \quad \Theta \in \mathbb{R}^{M\times M}$

Lemma 3 *The function* $\mathbf{f}_\Theta : \mathbb{R}^M \to \mathbb{R}^M$ *defined above is permutation* ***equivariant*** *iff all the off-diagonal elements of* Θ *are tied together and all the diagonal elements are equal as well. That is,*

$$\Theta = \lambda\mathbf{I} + \gamma\left(\mathbf{1}\mathbf{1}^\mathsf{T}\right) \qquad \lambda,\gamma \in \mathbb{R} \quad \mathbf{1} = [1,\ldots,1]^\mathsf{T} \in \mathbb{R}^M \qquad \mathbf{I} \in \mathbb{R}^{M\times M} \textit{ is the identity matrix}$$

Sketch of Proof

Permutation Equivariance $\sigma(\Theta\pi\mathbf{x}) = \pi\sigma(\Theta\mathbf{x})$ (w. element-wise bijective nonlinearity) reduces to $\pi\Theta\mathbf{x} = \Theta\pi\mathbf{x}$

Sufficiency: Θ is commutable with permutation matrix

Necessity: consider a special permutation (i.e., transposition / swap) $\pi_{i,j}^\top = \pi_{i,j}^{-1} = \pi_{j,i}$

1. All diagonal elements are identical

   $$\pi_{k,l}\Theta = \Theta\pi_{k,l} \Rightarrow \pi_{k,l}\Theta\pi_{l,k} = \Theta \Rightarrow (\pi_{k,l}\Theta\pi_{l,k})_{l,l} = \Theta_{l,l} \Rightarrow \Theta_{k,k} = \Theta_{l,l}$$

2. All off-diagonal elements are identical

   $$\pi_{j',j}\pi_{i,i'}\Theta = \Theta\pi_{j',j}\pi_{i,i'} \Rightarrow \pi_{j',j}\pi_{i,i'}\Theta(\pi_{j',j}\pi_{i,i'})^{-1} = \Theta \Rightarrow$$
   $$\pi_{j',j}\pi_{i,i'}\Theta\pi_{i',i}\pi_{j,j'} = \Theta \Rightarrow (\pi_{j',j}\pi_{i,i'}\Theta\pi_{i',i}\pi_{j,j'})_{i,j} = \Theta_{i,j} \Rightarrow \Theta_{i',j'} = \Theta_{i,j}$$

Deep Learning for Sets

- Deep Sets [1]

Equivariant Architecture

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

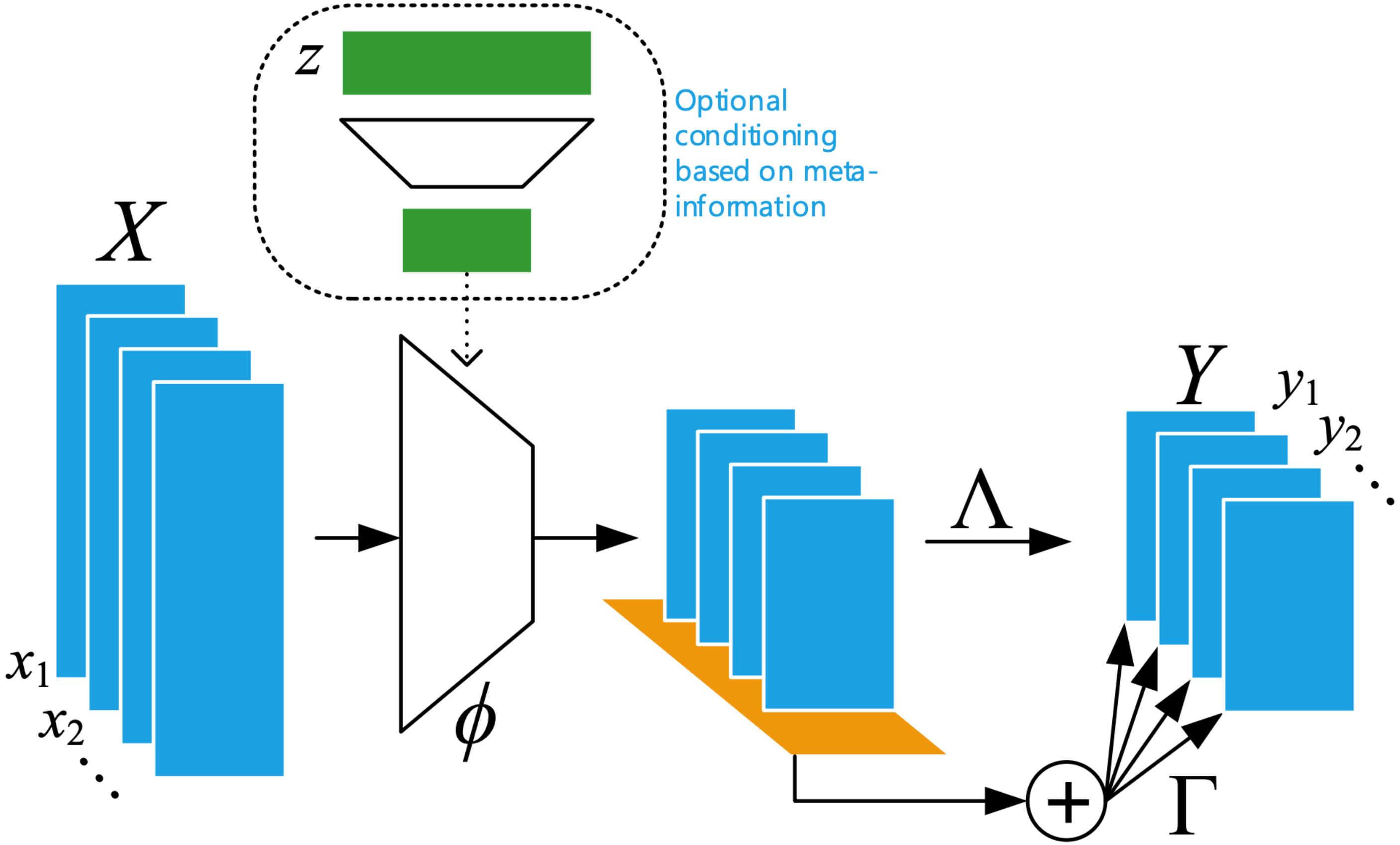


Image Credit: [1]

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]

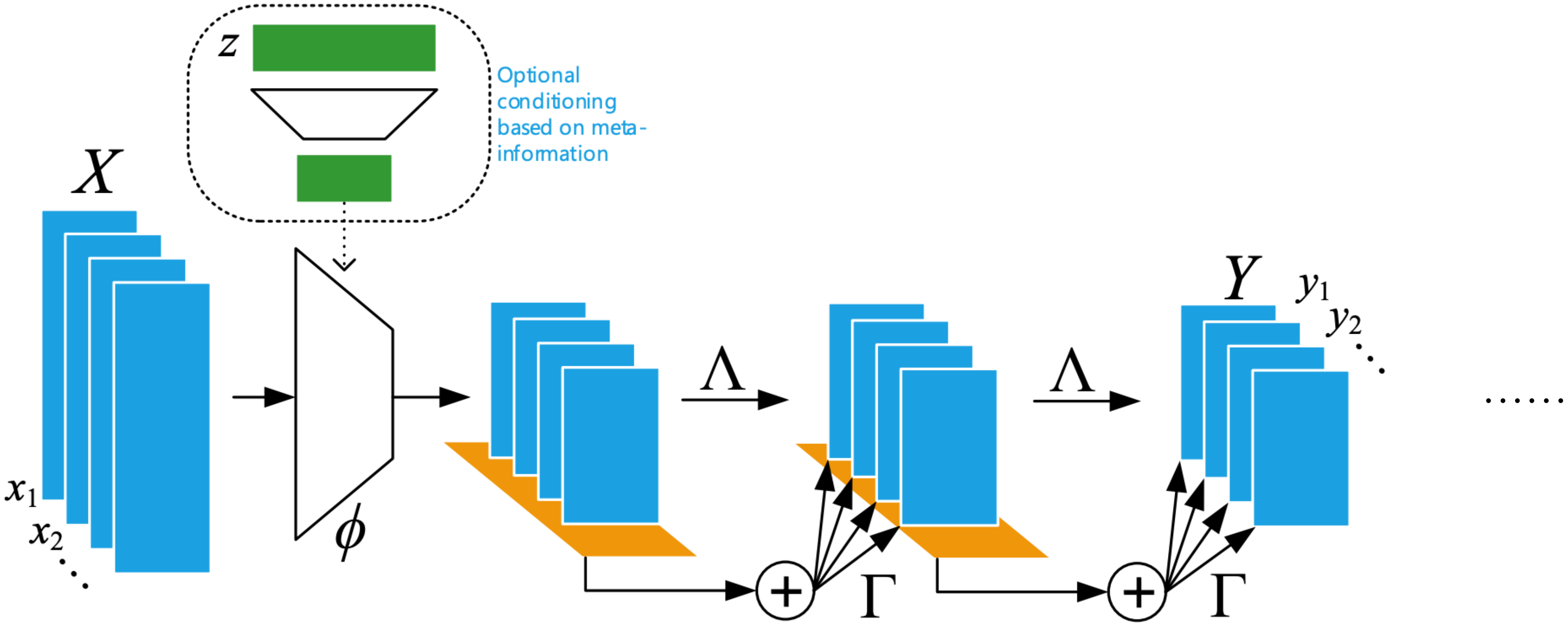


Image Credit: [1]

Deep Learning for Sequences

- Language Models

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

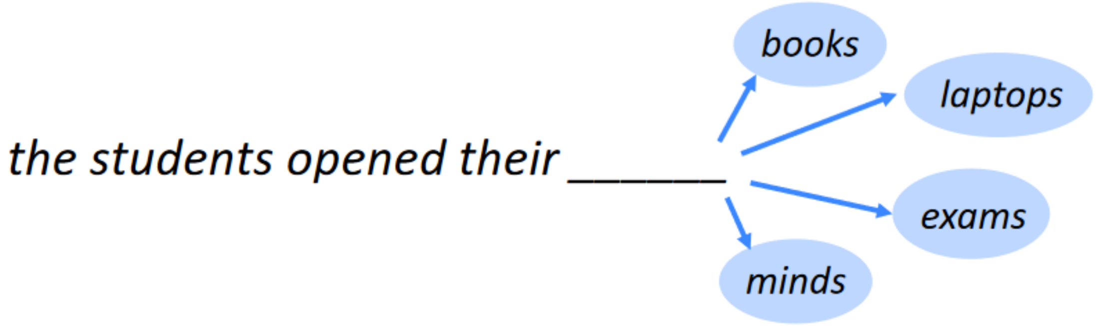
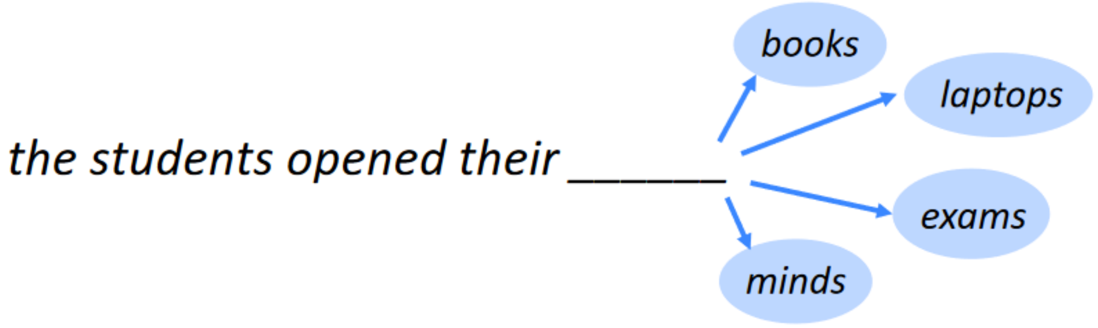


Image Credit: http://web.stanford.edu/class/cs224n/

Deep Learning for Sequences

- Language Models

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



- 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 [3]
GRU [4]
Seq2Seq [5]
Transformer [6]

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

Transformers

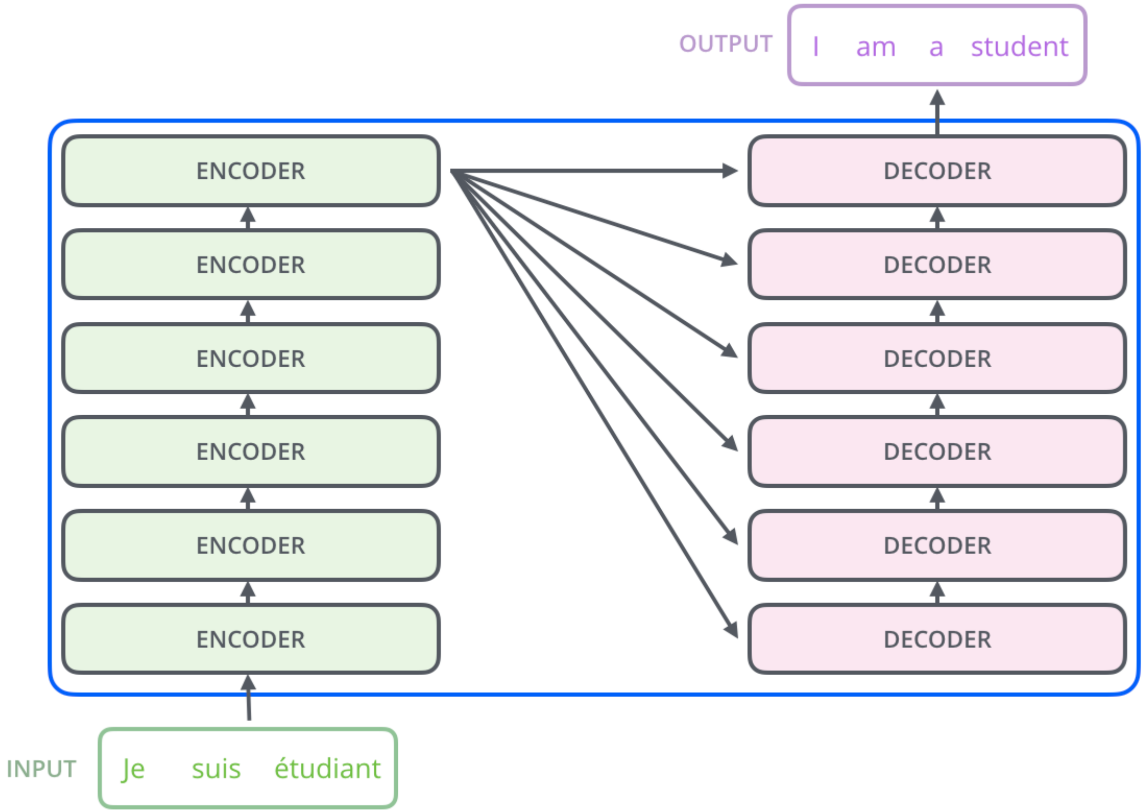
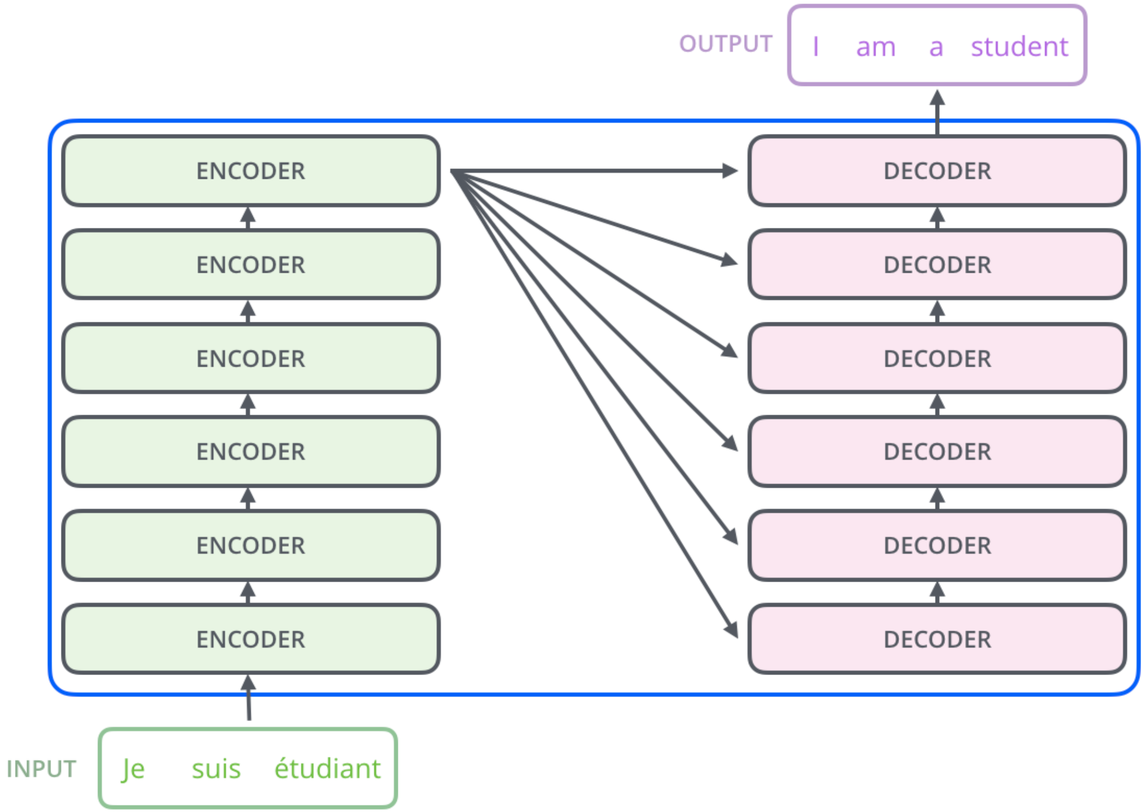


Image Credit: https://jalammar.github.io/illustrated-transformer/

Transformers



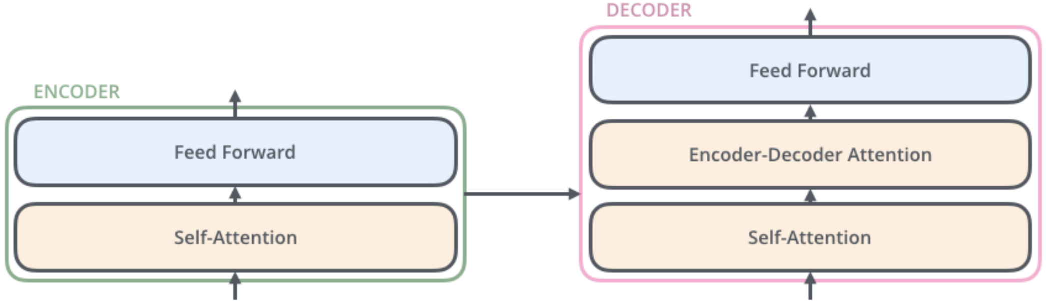
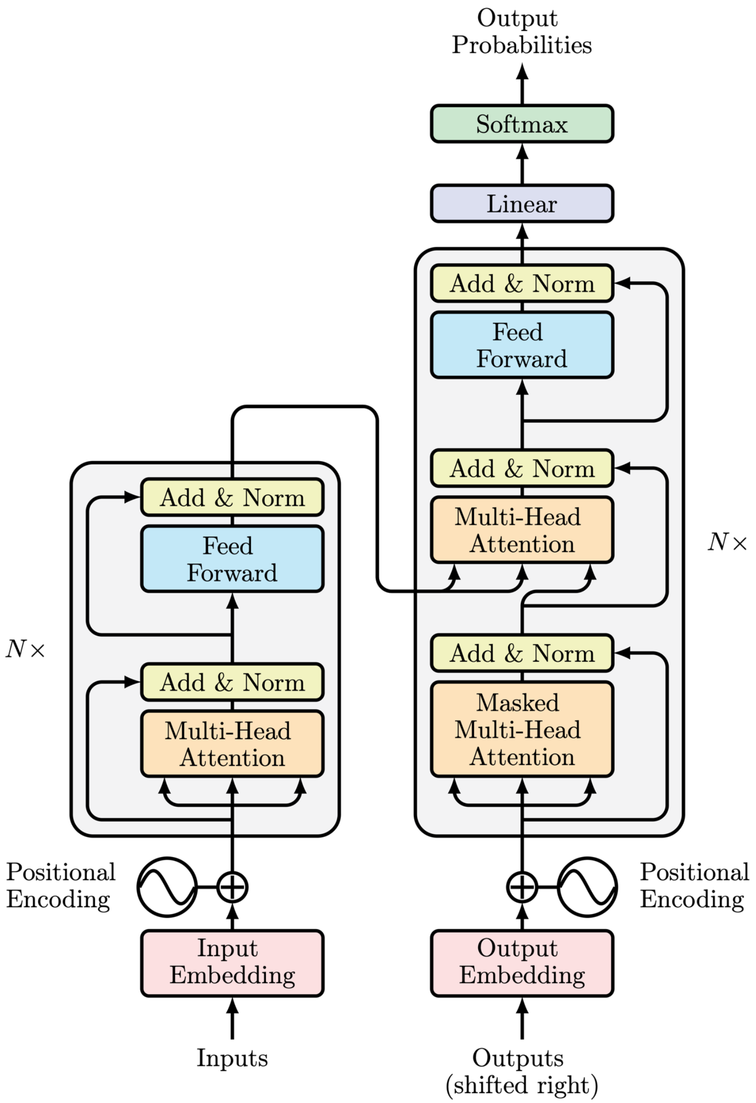


Image Credit: https://jalammar.github.io/illustrated-transformer/

Transformers



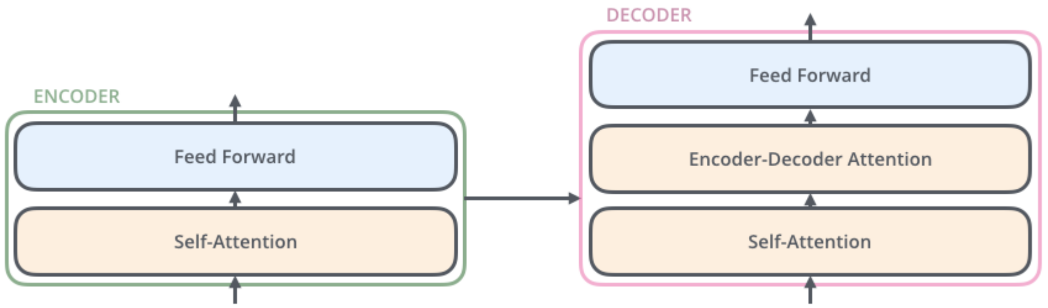
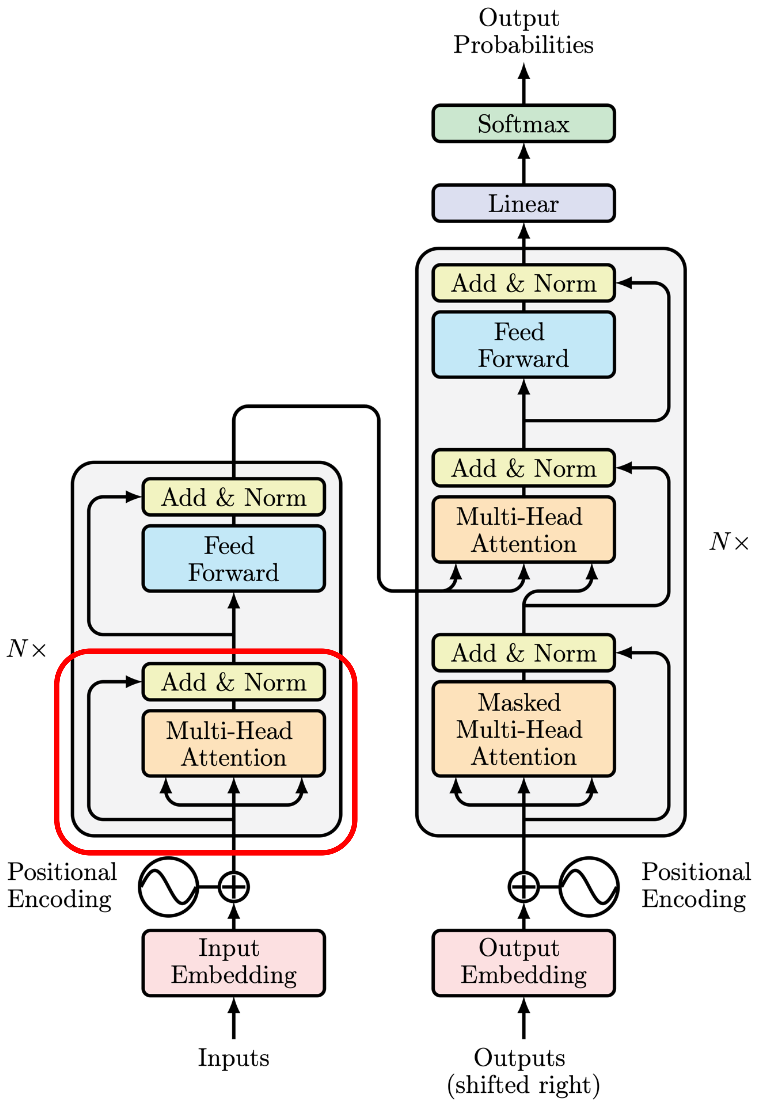


Image Credit: [6] & https://jalammar.github.io/illustrated-transformer/

Transformers



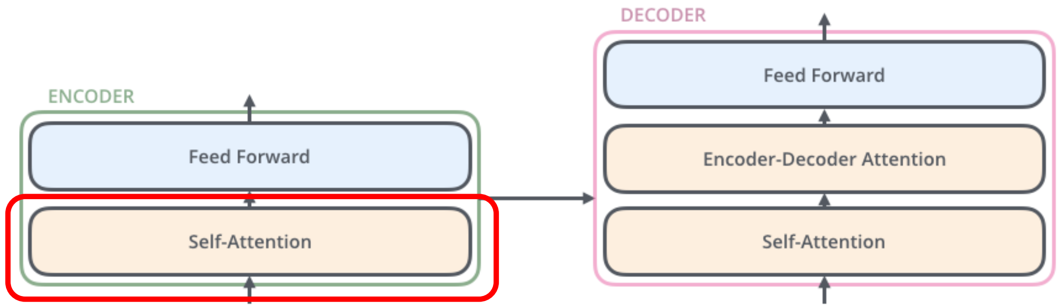
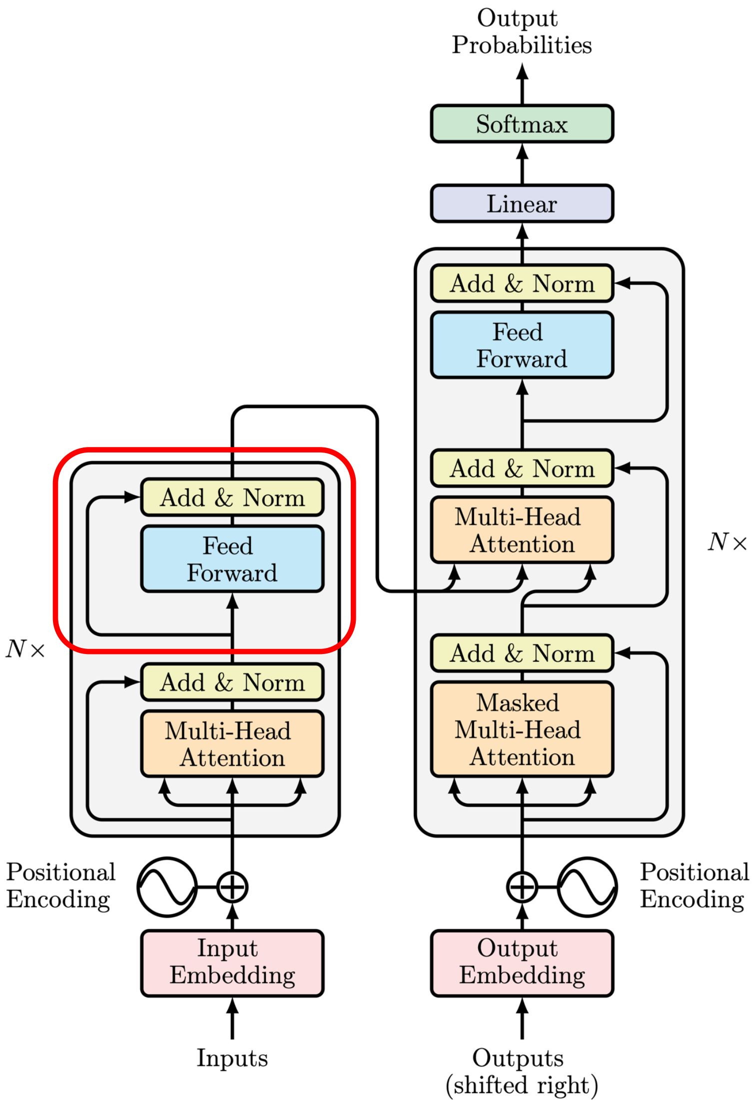


Image Credit: [6] & https://jalammar.github.io/illustrated-transformer/

Transformers



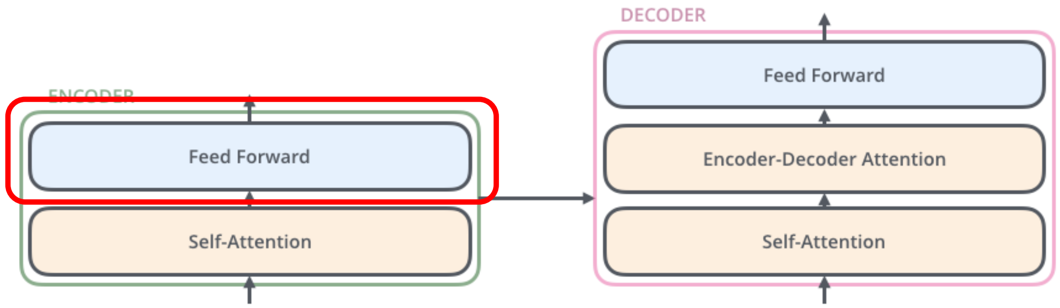
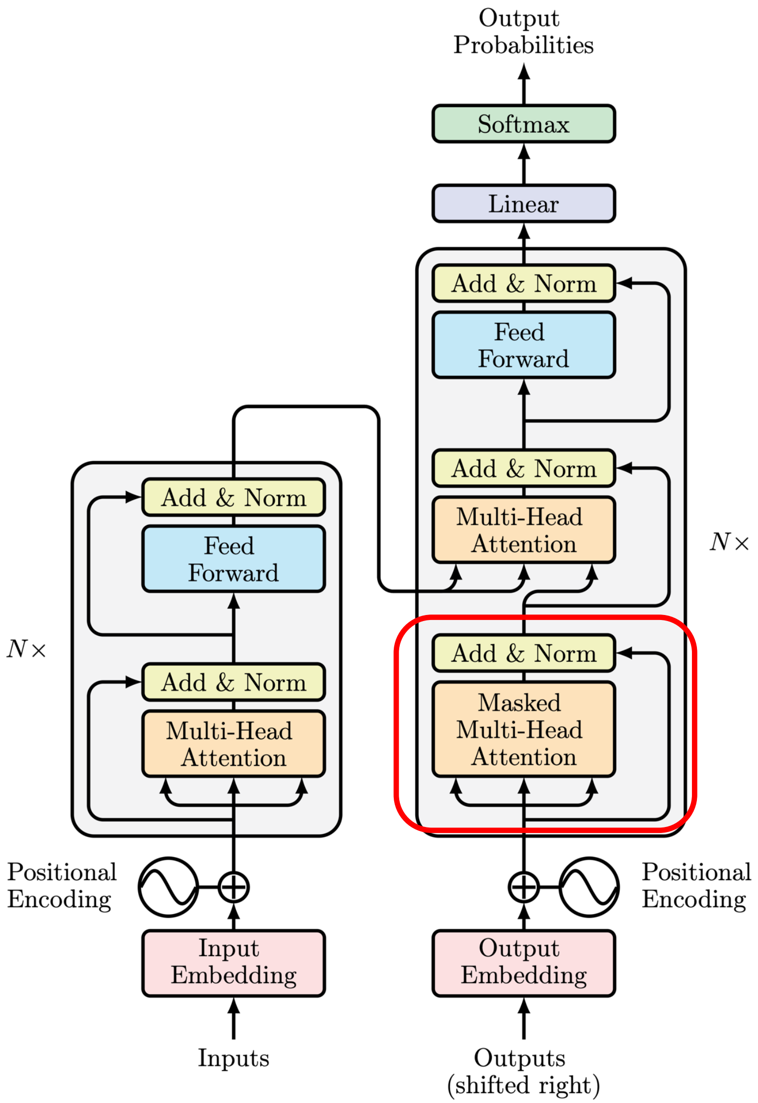


Image Credit: [6] & https://jalammar.github.io/illustrated-transformer/

Transformers



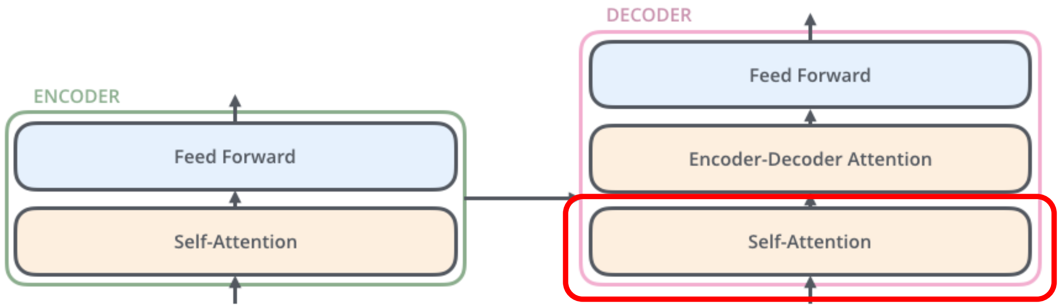
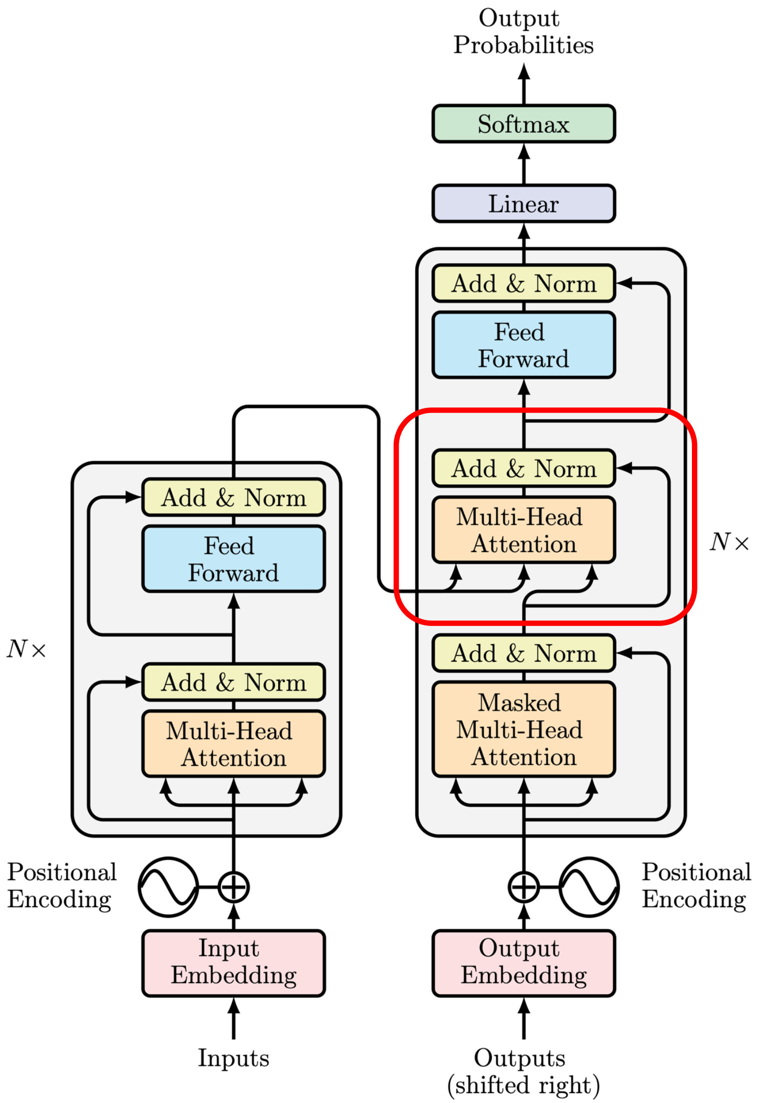


Image Credit: [6] & https://jalammar.github.io/illustrated-transformer/

Transformers



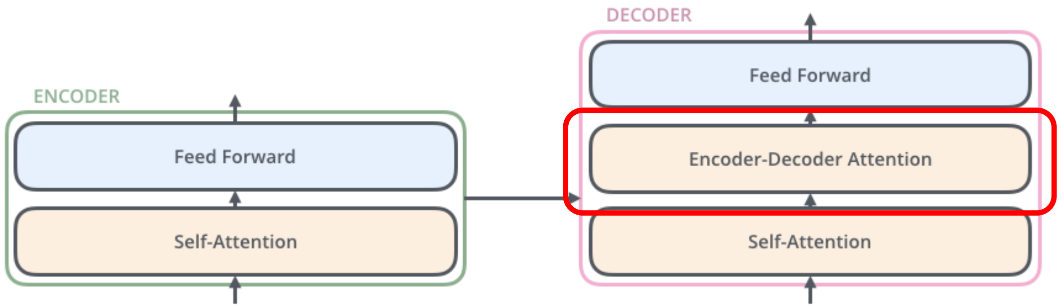
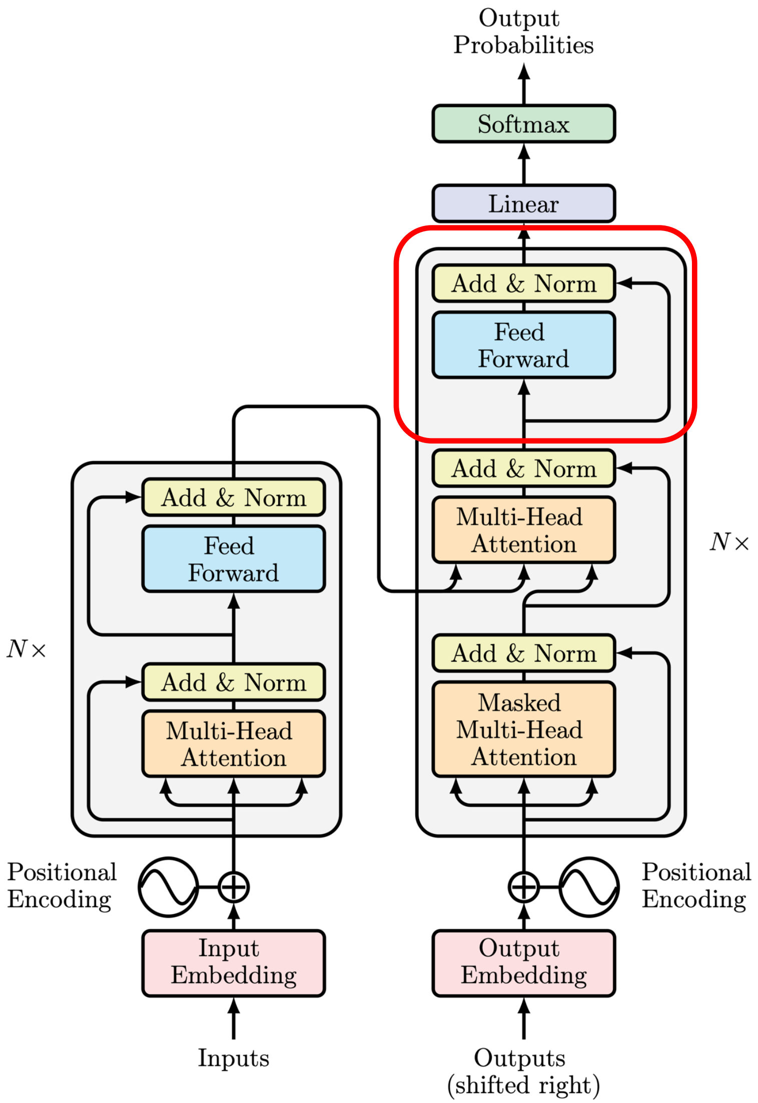


Image Credit: [6] & https://jalammar.github.io/illustrated-transformer/

Transformers



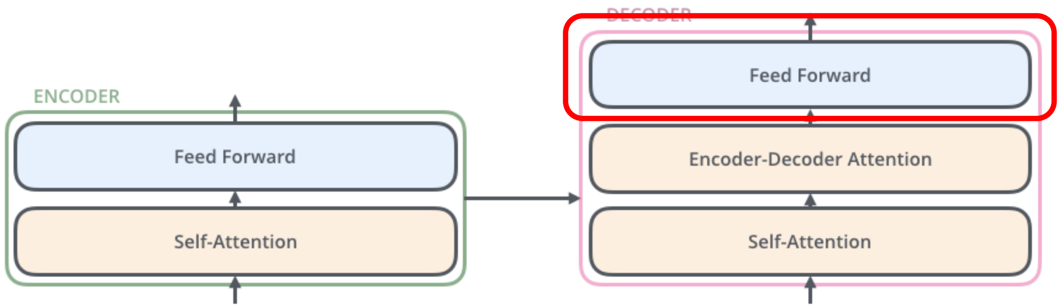


Image Credit: [6] & https://jalammar.github.io/illustrated-transformer/

Input Encoding

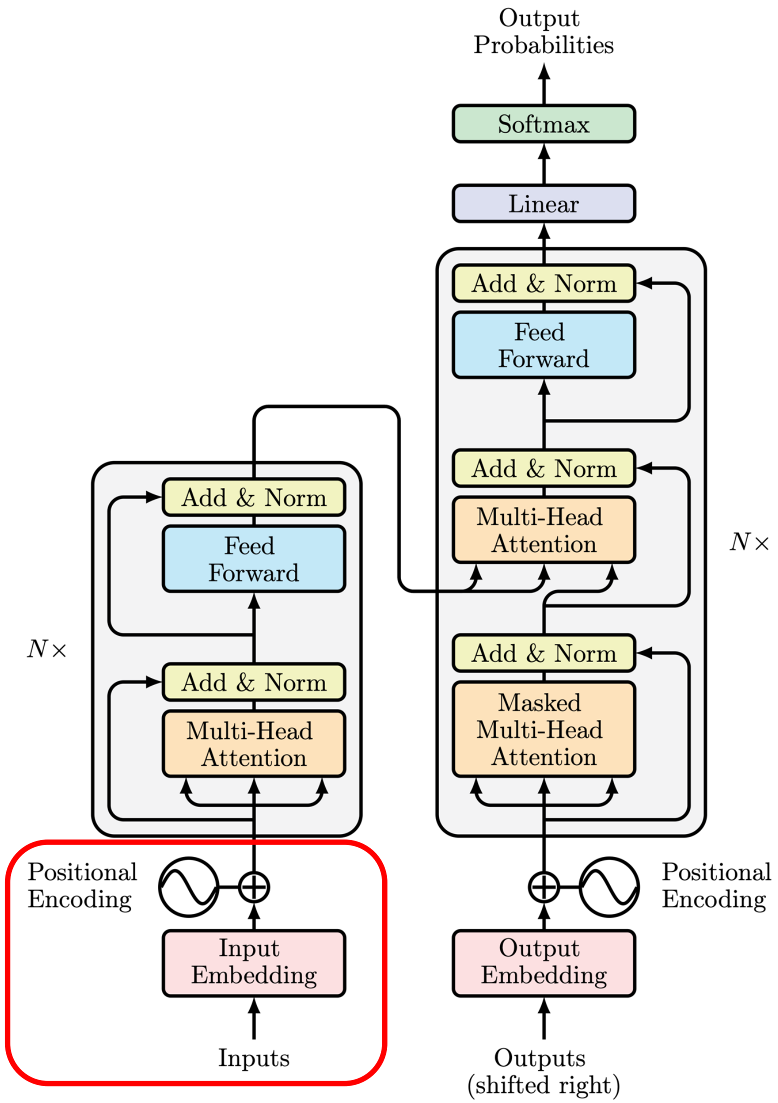
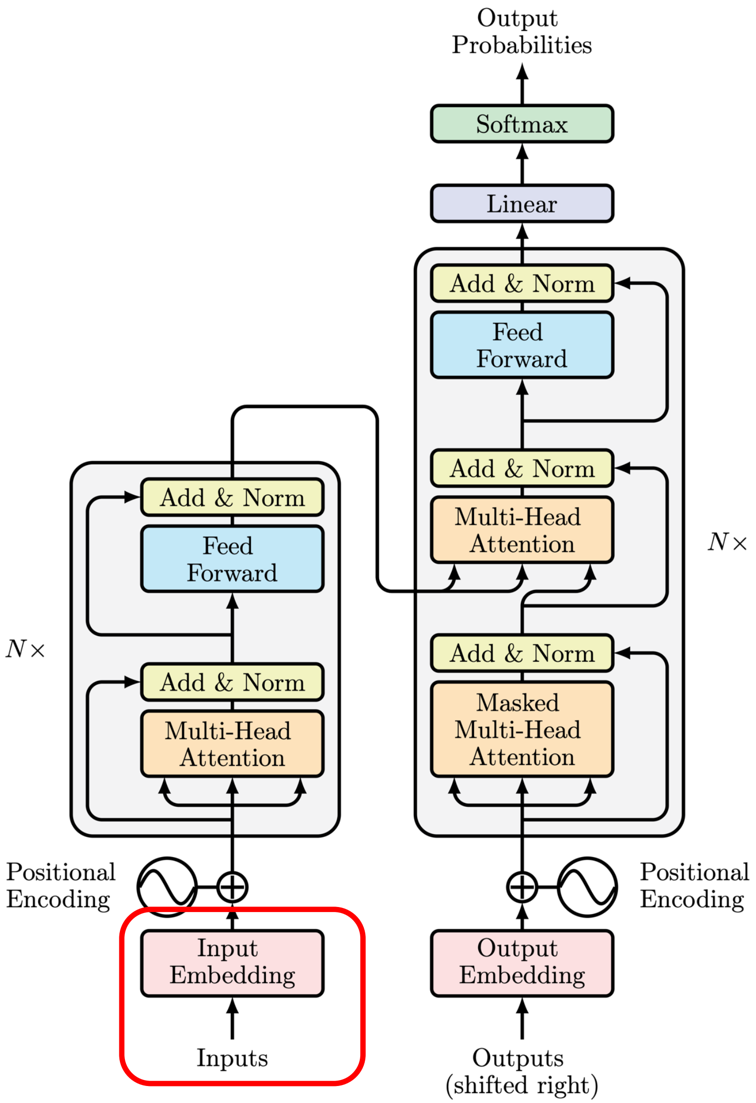


Image Credit: [6]

Input Embedding



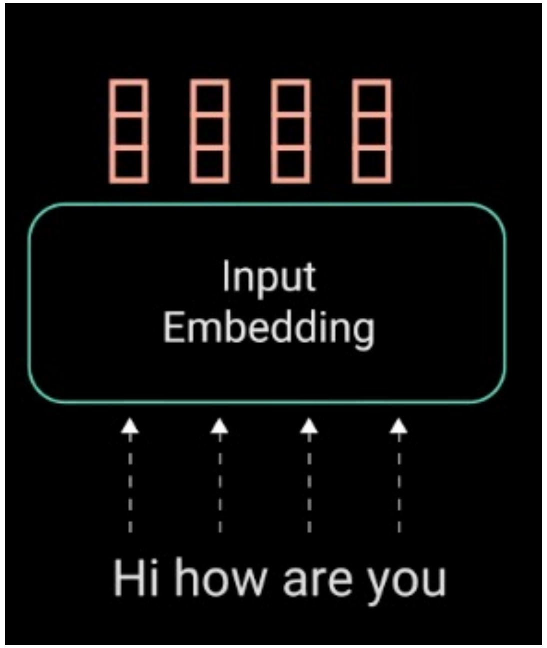
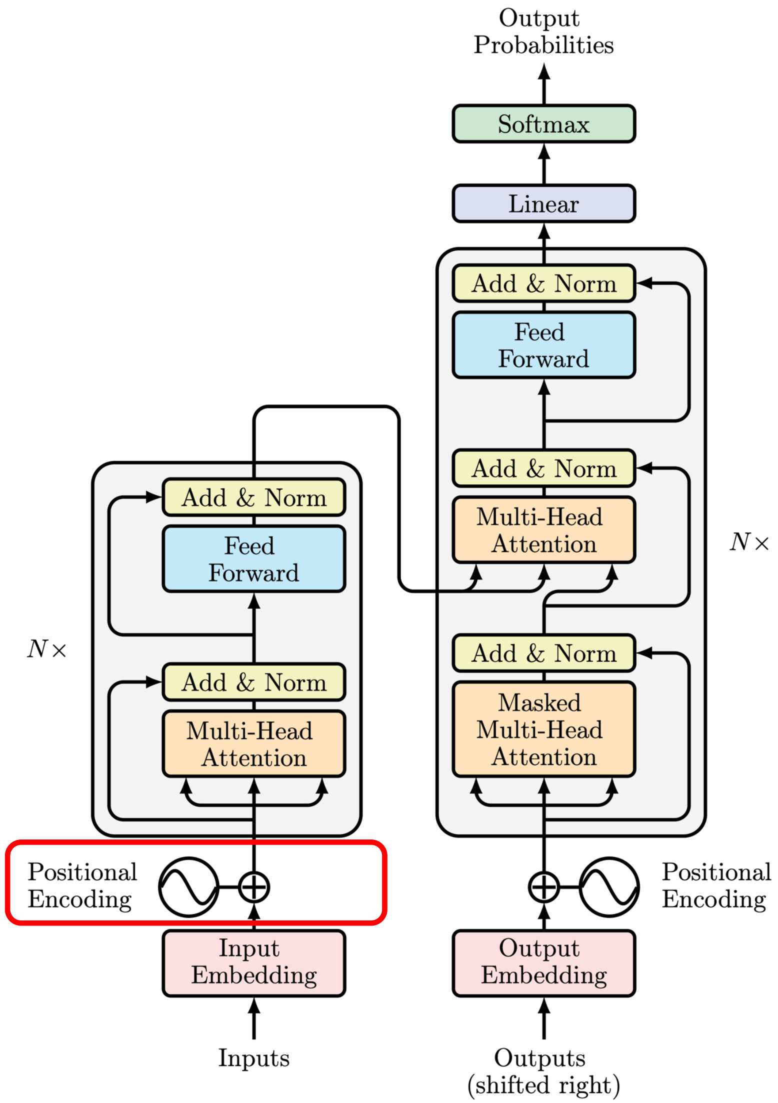
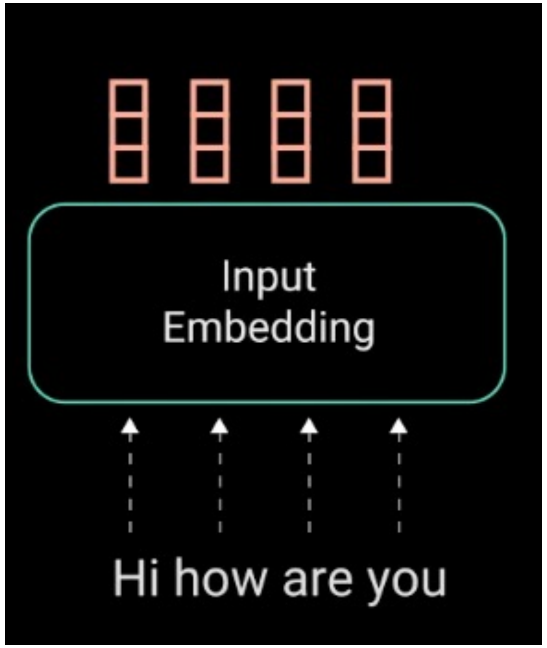


Image Credit: [6] & https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

Positional Encoding



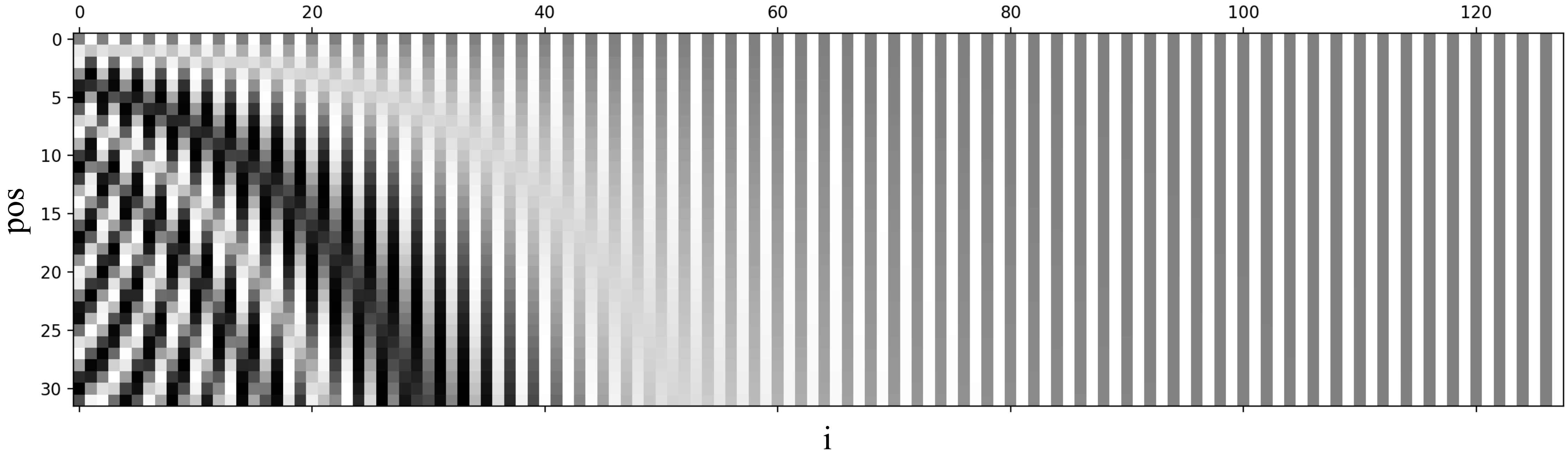


$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$

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

Image Credit: [6] & https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

Positional Encoding

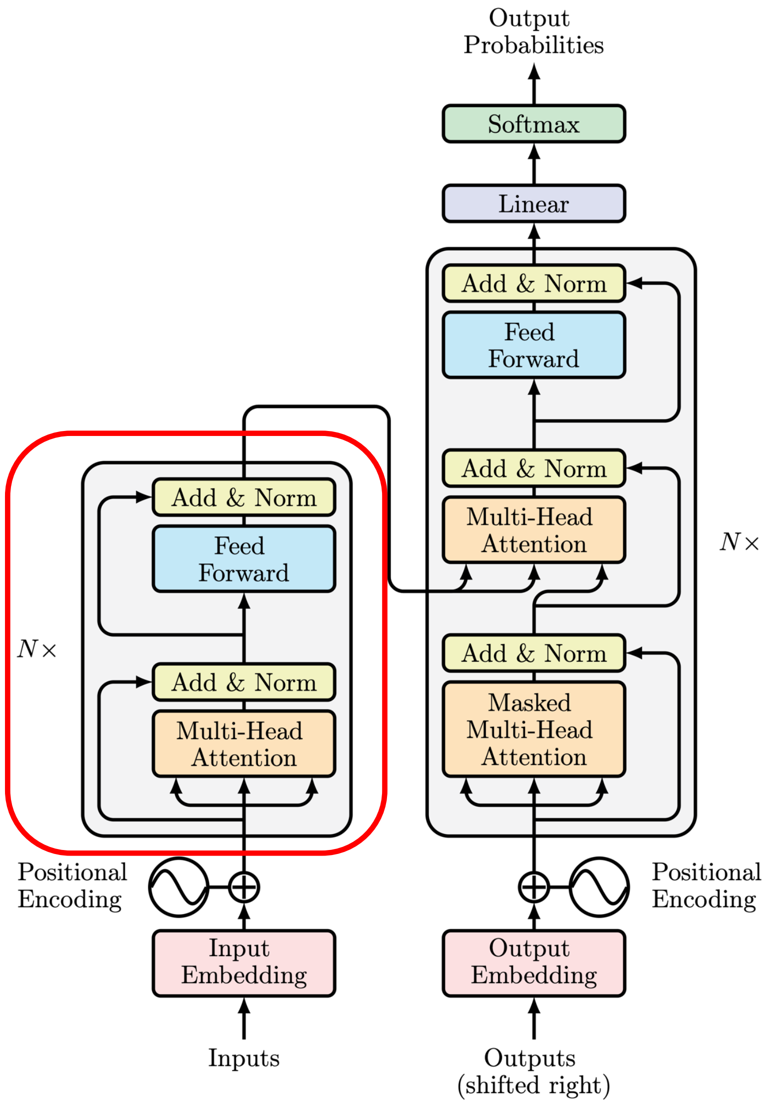


$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$

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

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

Encoder



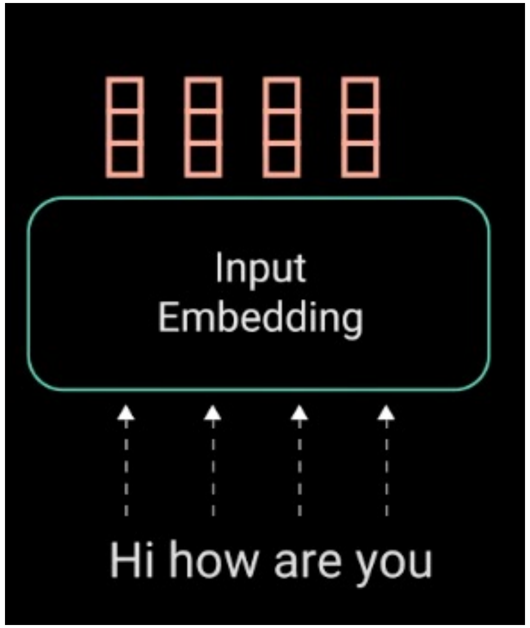
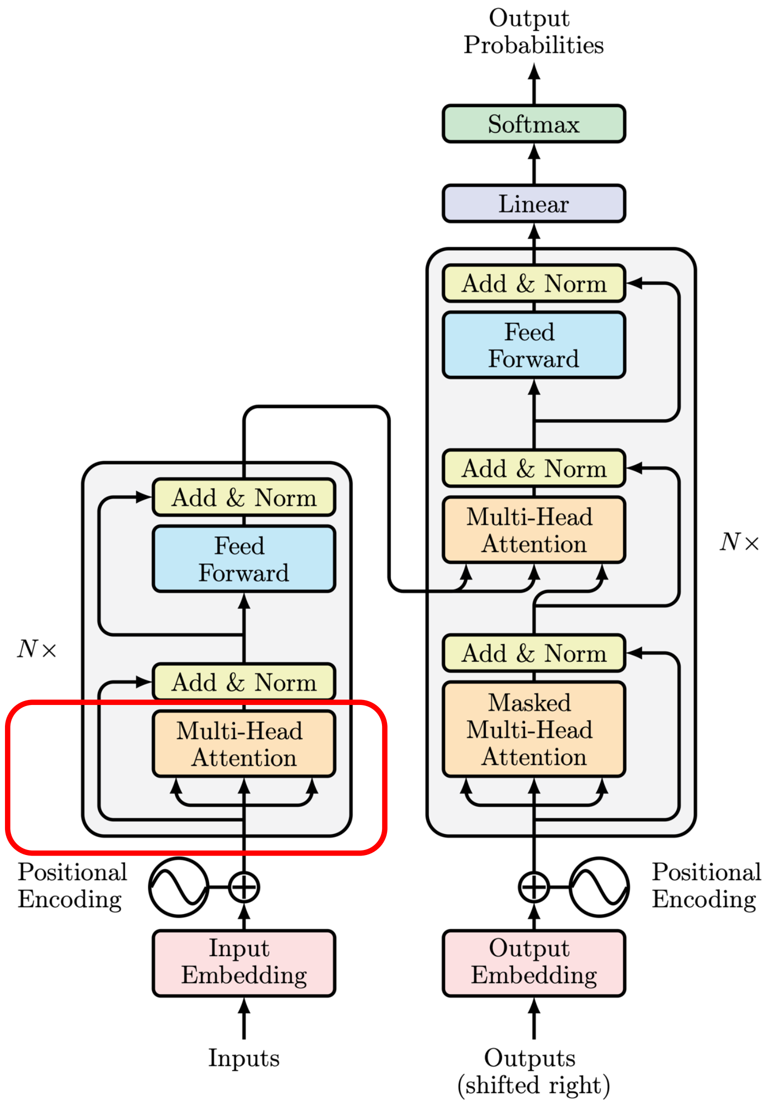


Image Credit: [6] & https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

Multi-Head Attention



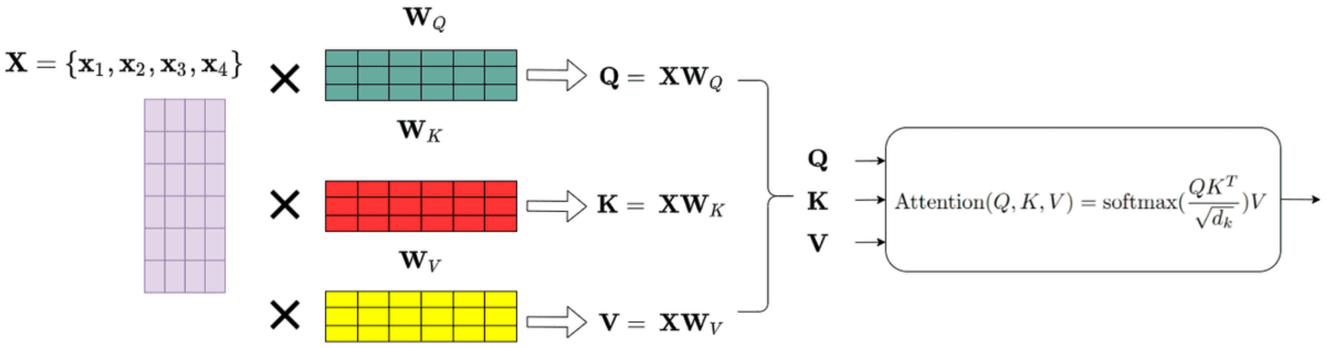
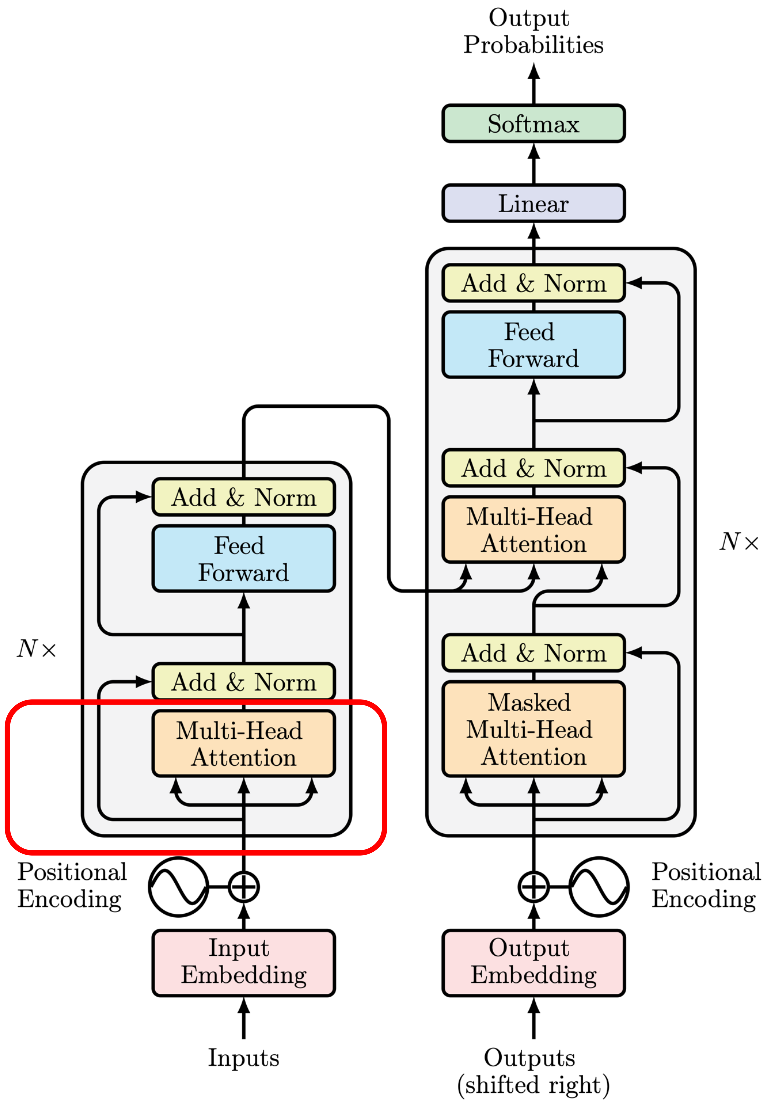


Image Credit: [6] & https://theaisummer.com/transformer/

Multi-Head Attention



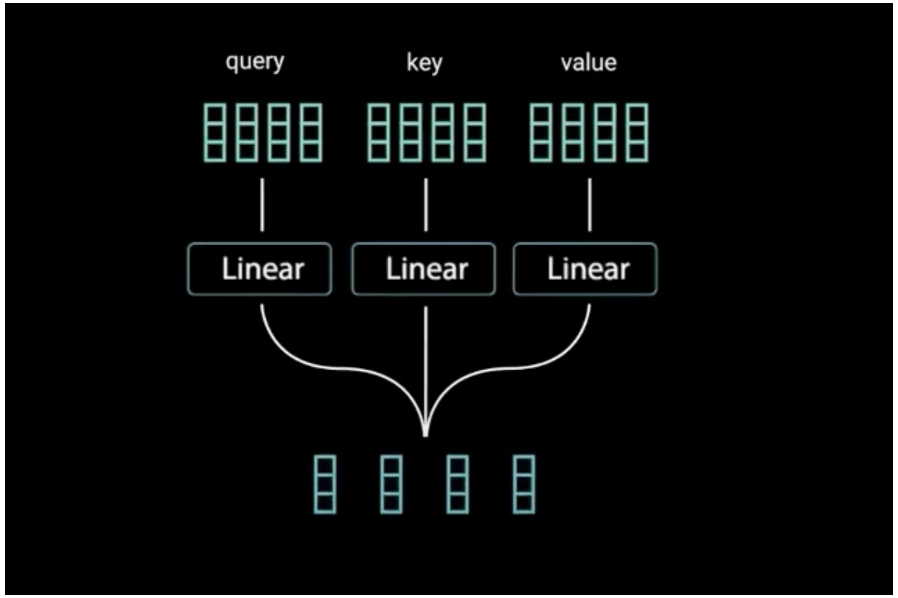
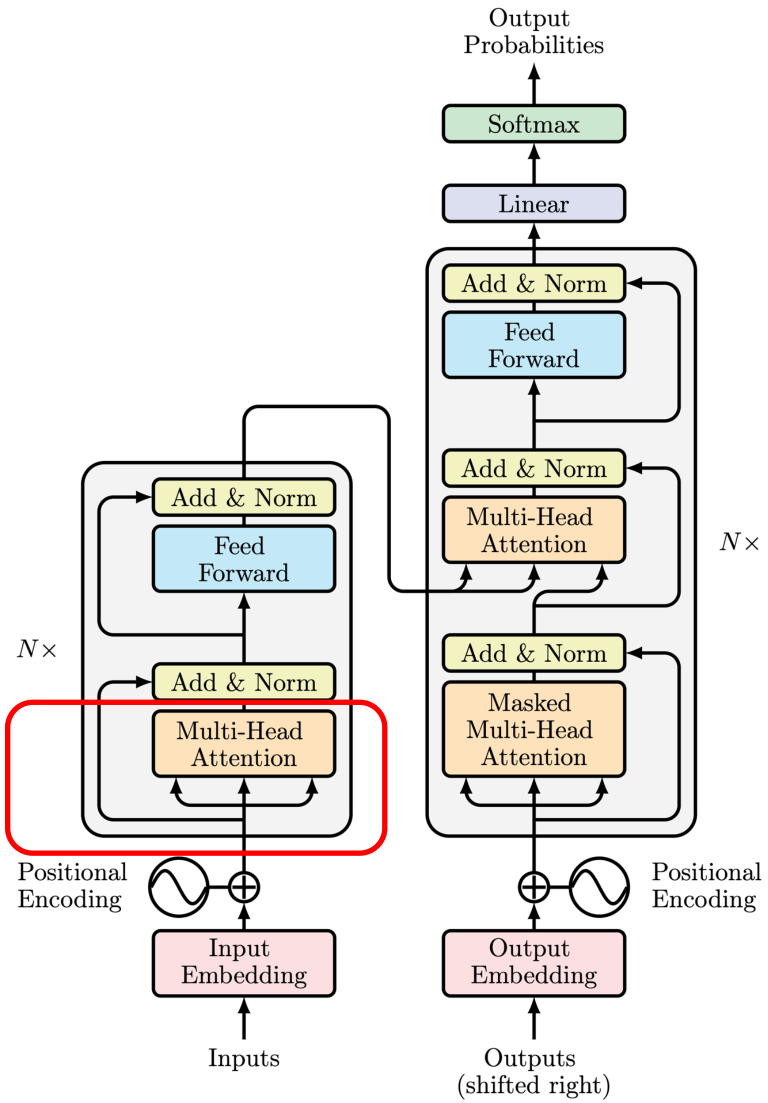
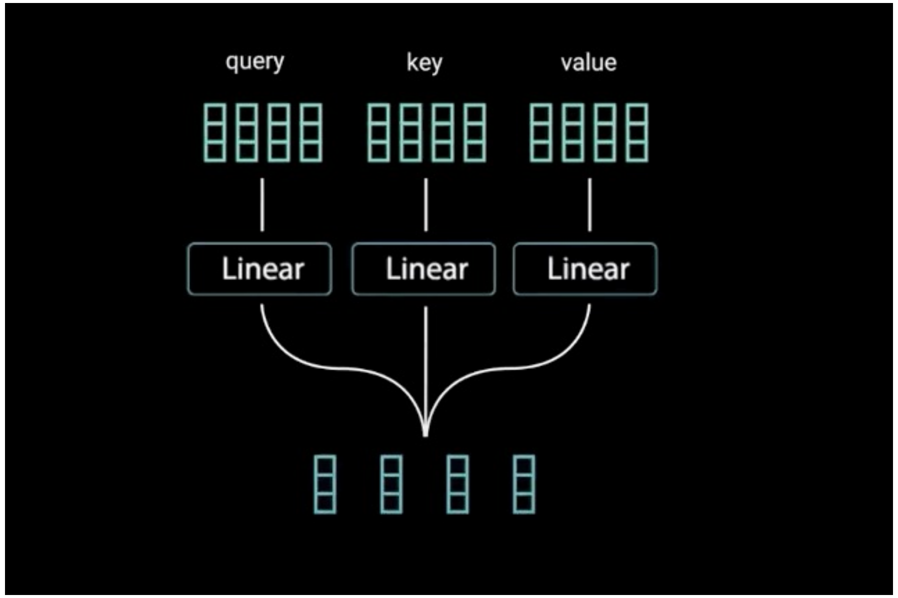


Image Credit: [6] & https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

Multi-Head Attention





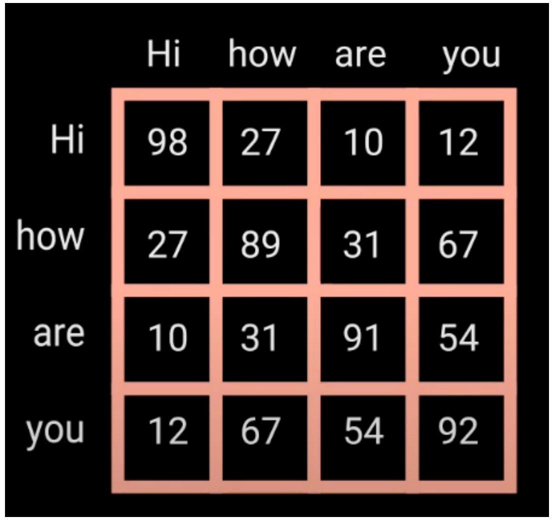
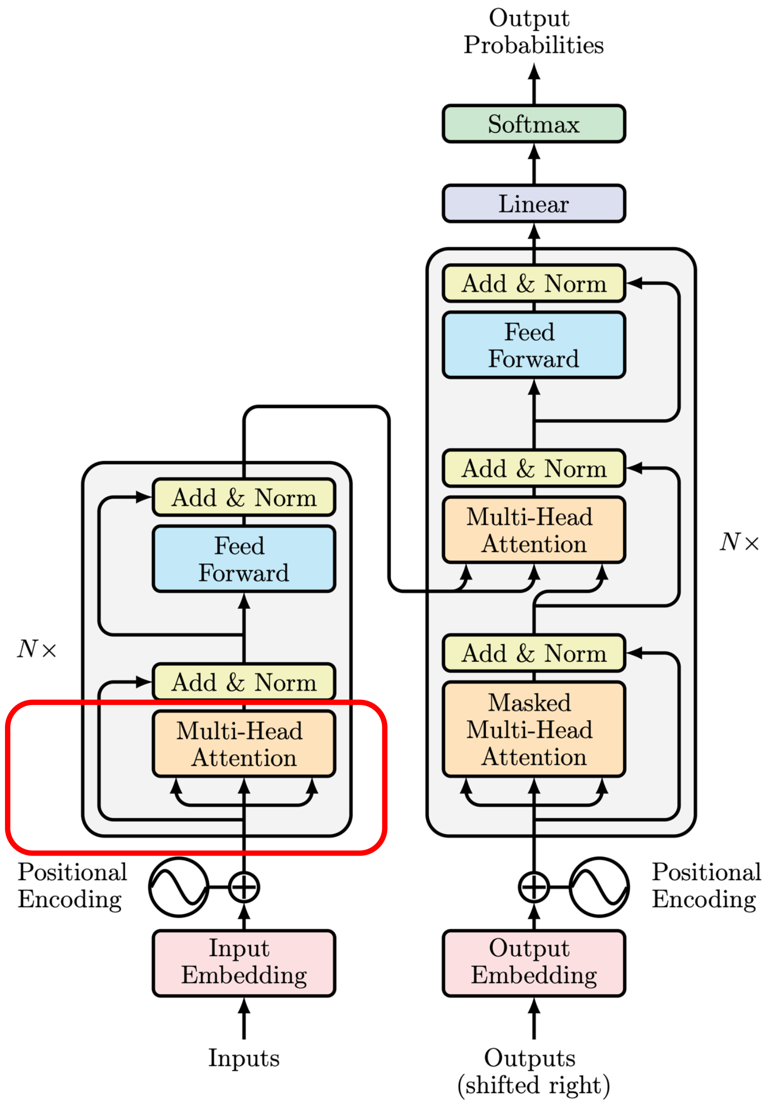
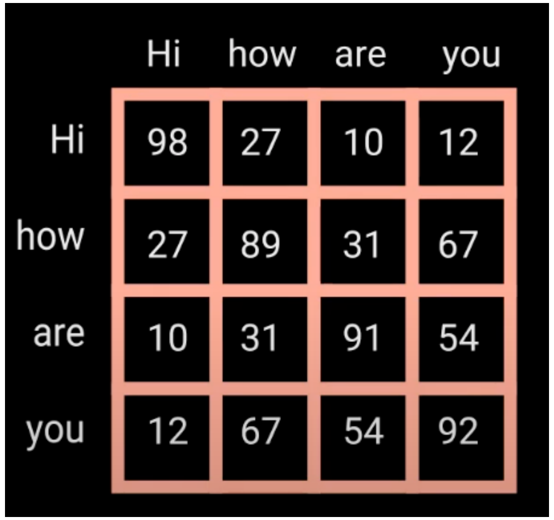


Image Credit: [6] & https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

Multi-Head Attention





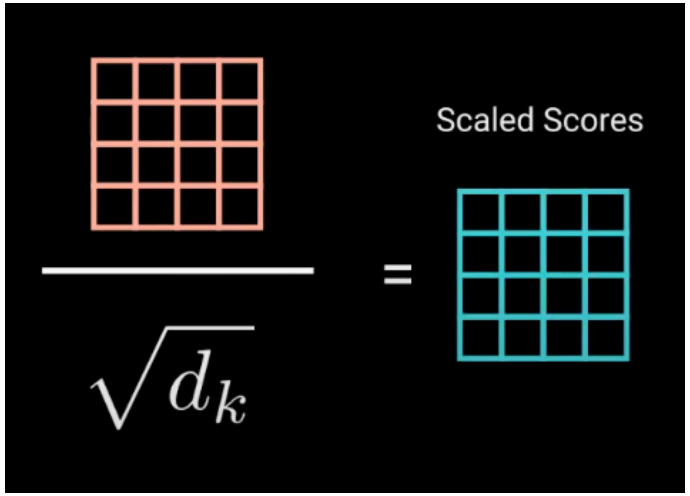
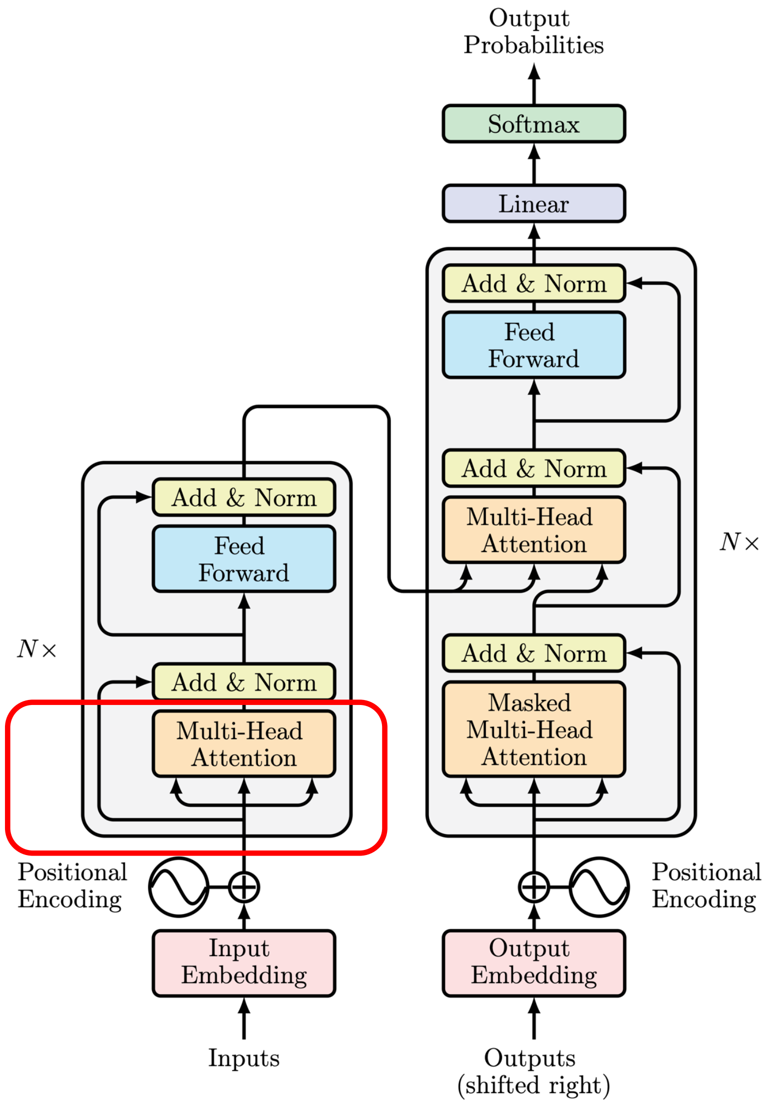
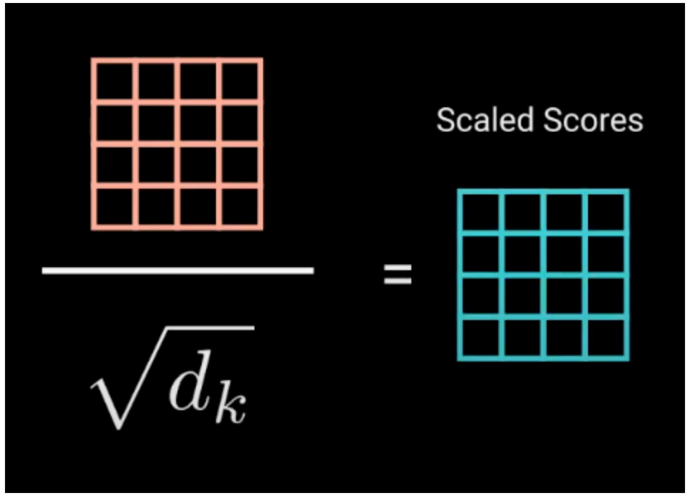


Image Credit: [6] & https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

Multi-Head Attention





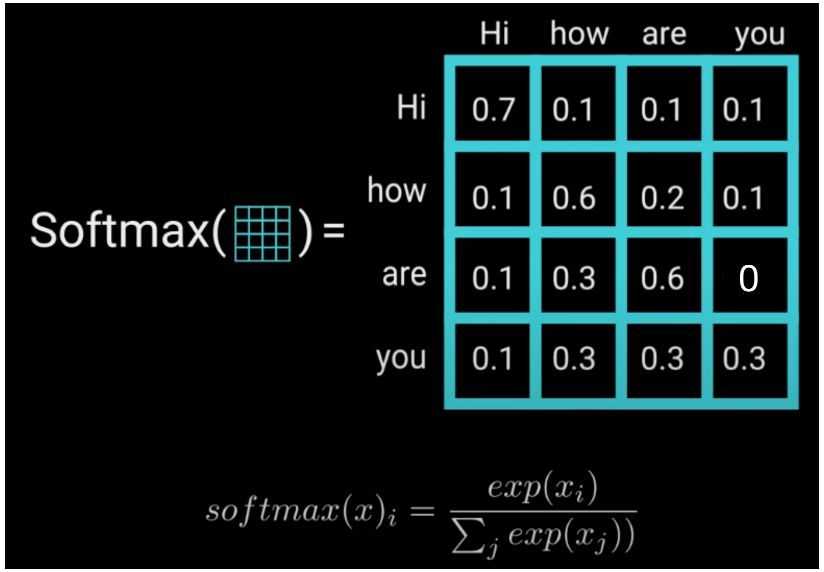
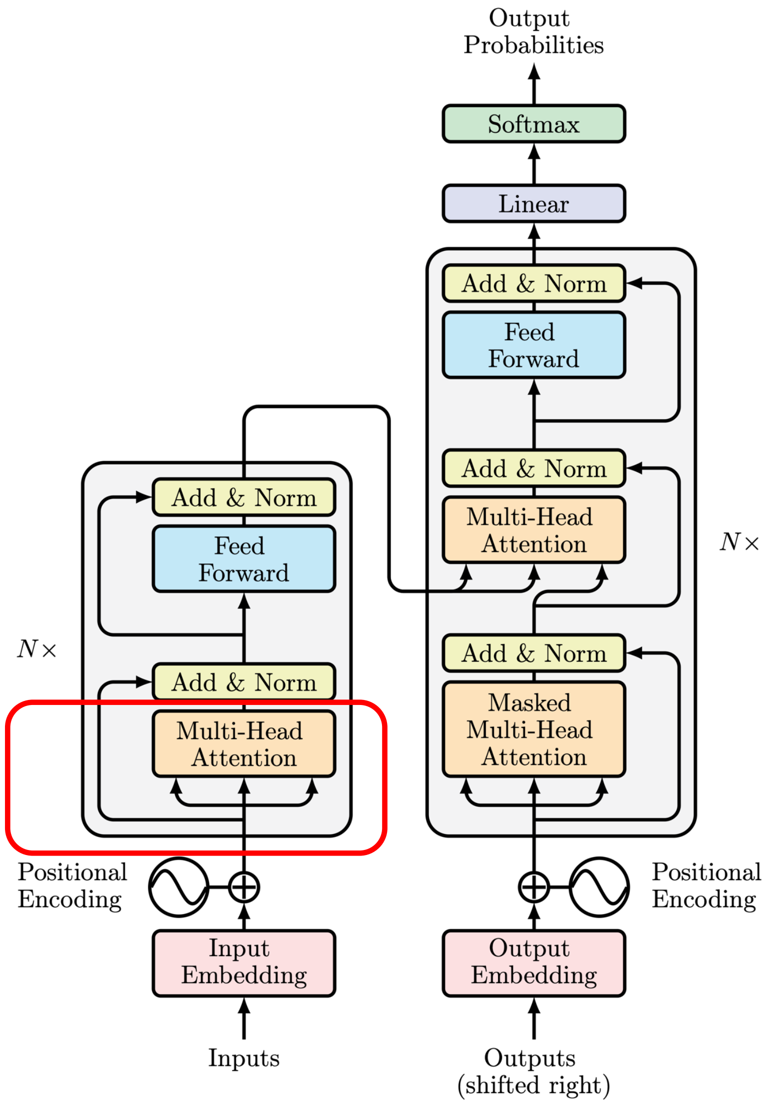
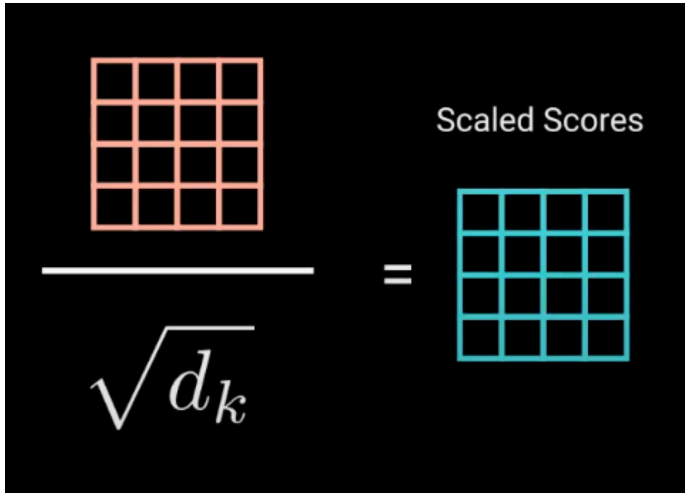


Image Credit: [6] & https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

Multi-Head Attention





Why square root?

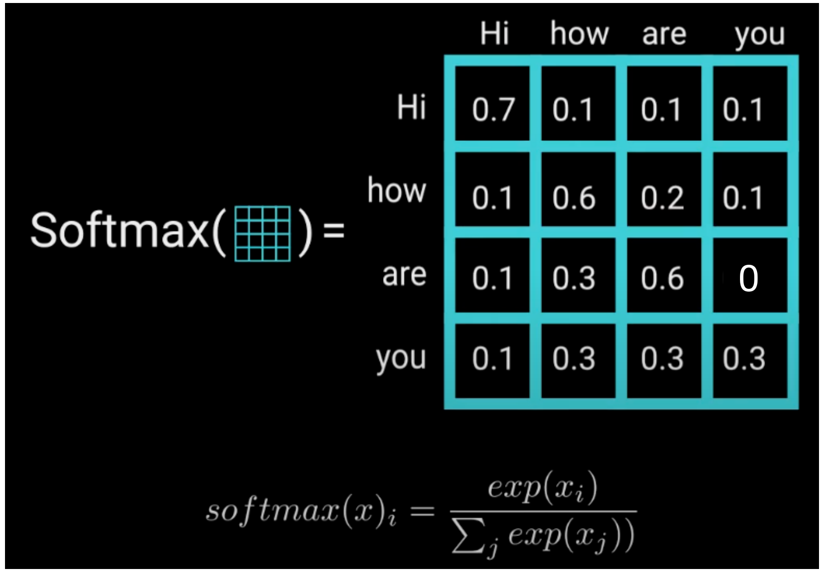
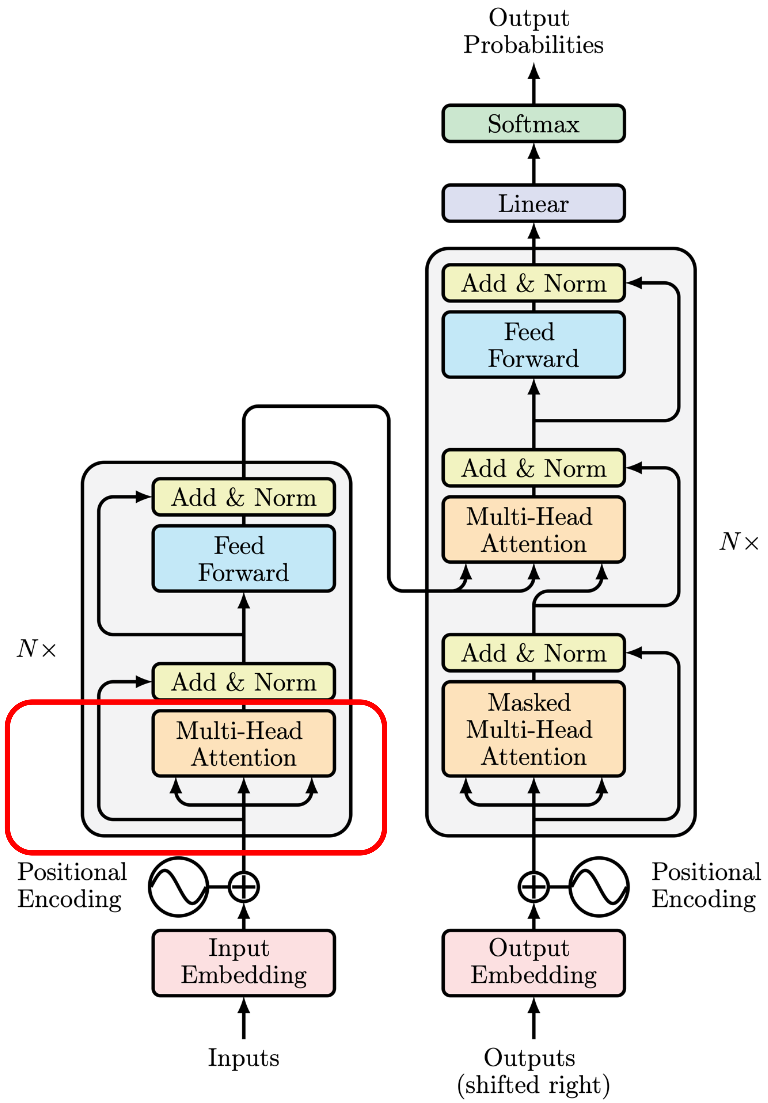
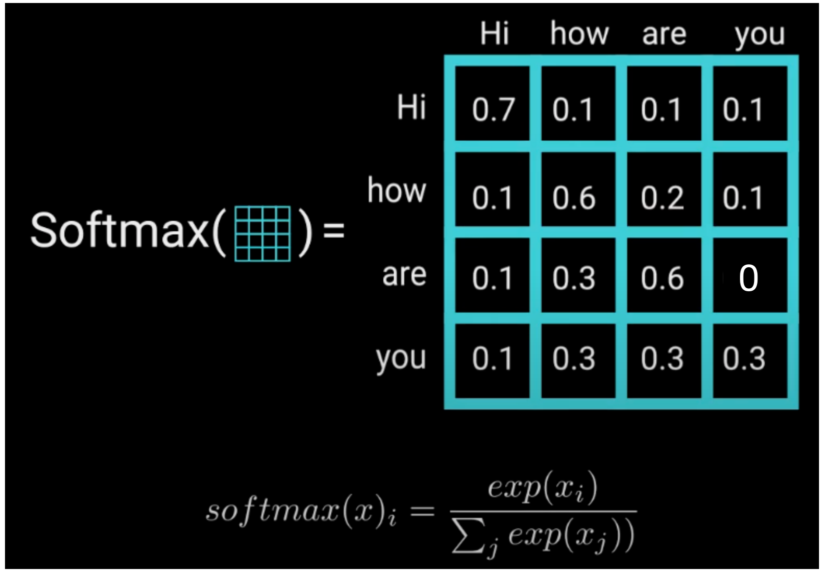


Image Credit: [6] & https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

Multi-Head Attention





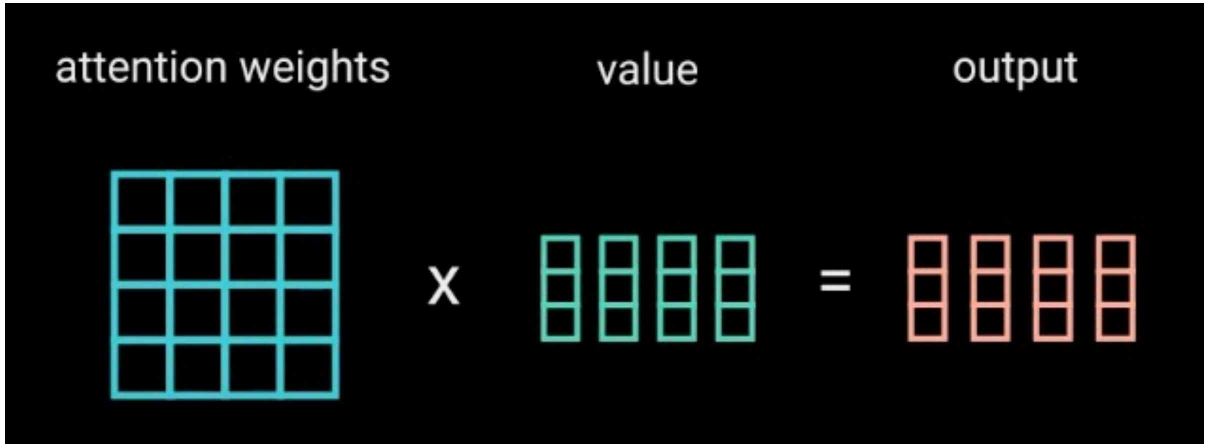
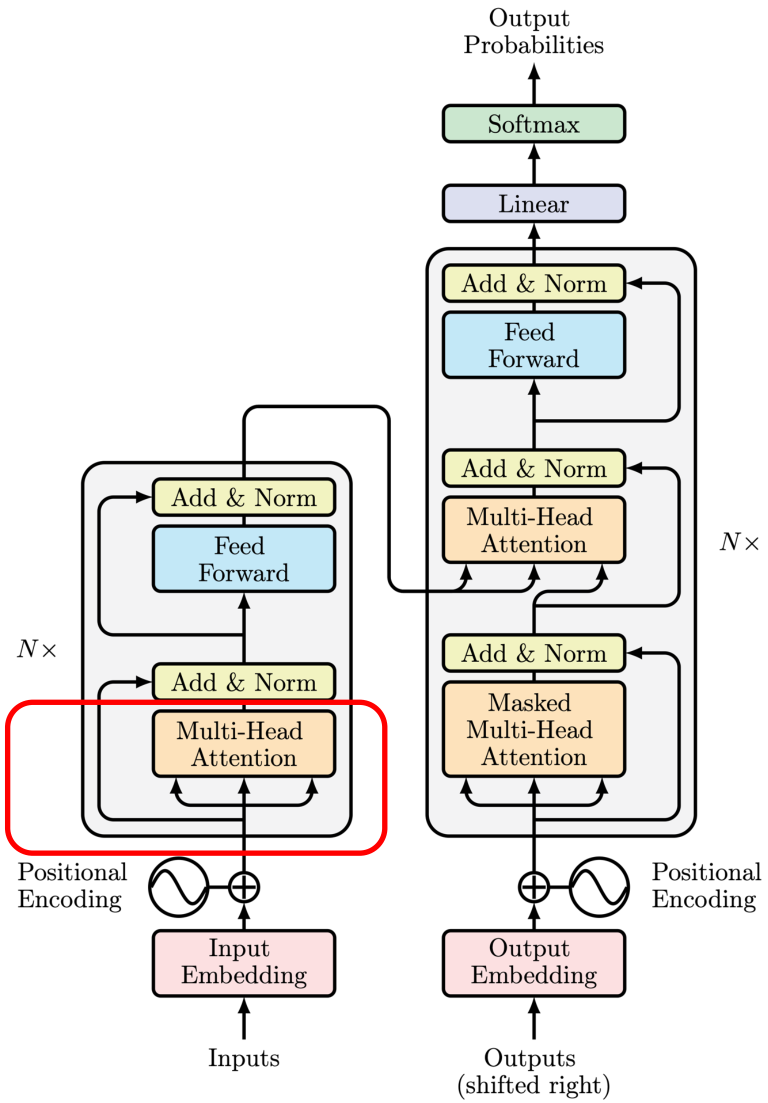


Image Credit: [6] & https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

Multi-Head Attention



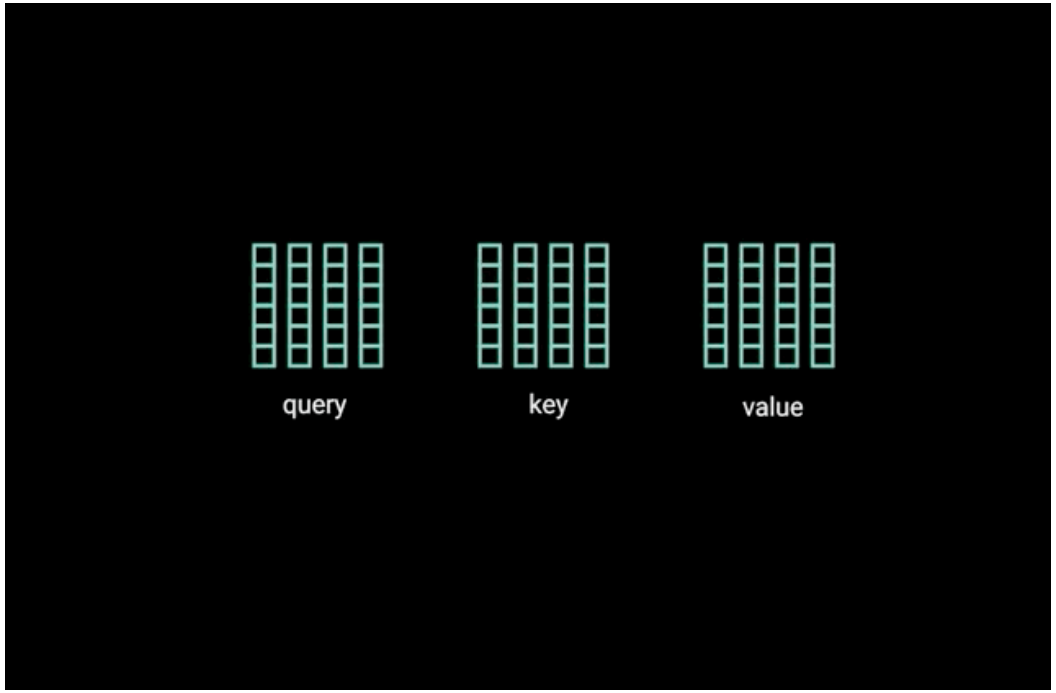
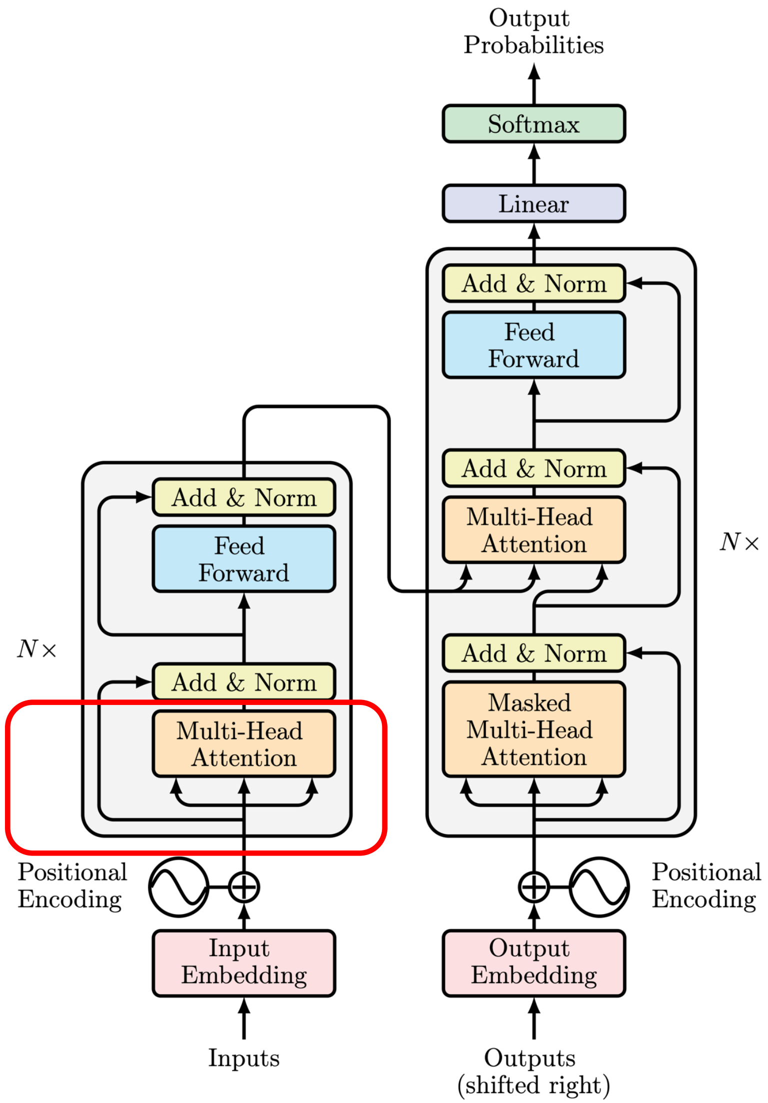
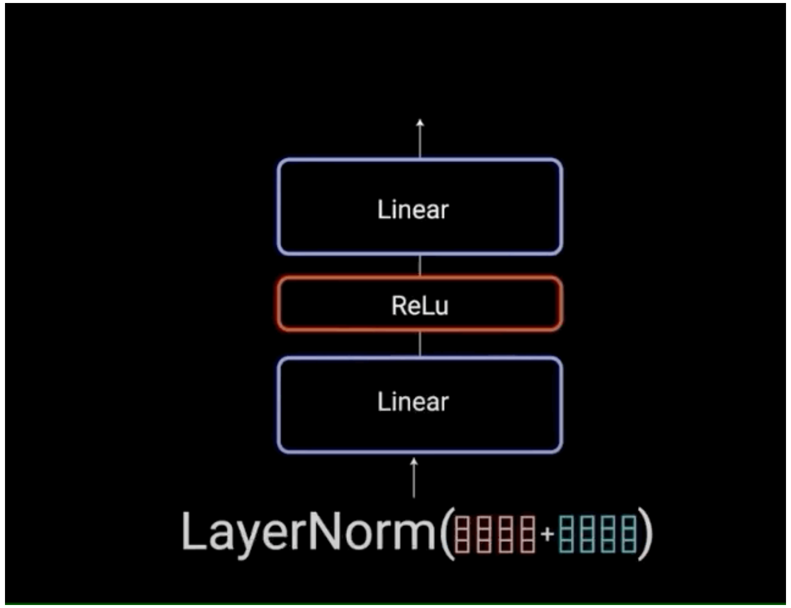


Image Credit: [6] & https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

Layer Norm & Residual Connection





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

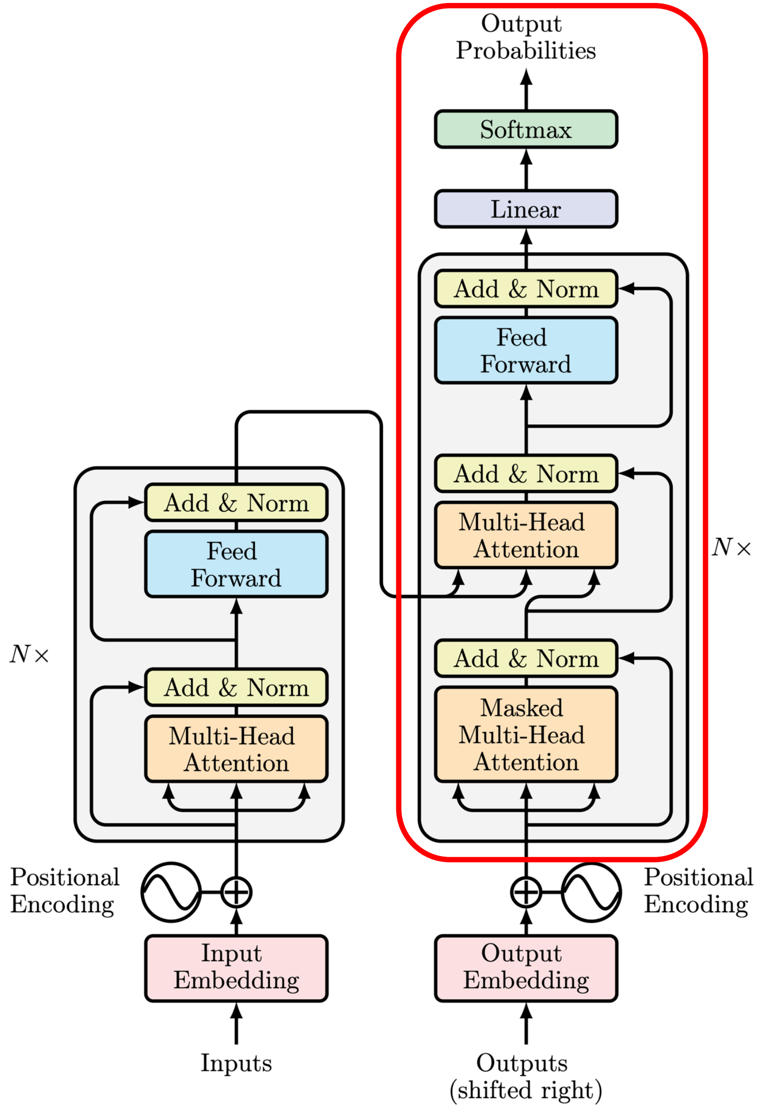
$$\sigma_i^2 = \frac{1}{K}\sum_{k=1}^{K} (x_{i,k} - \mu_i)^2$$

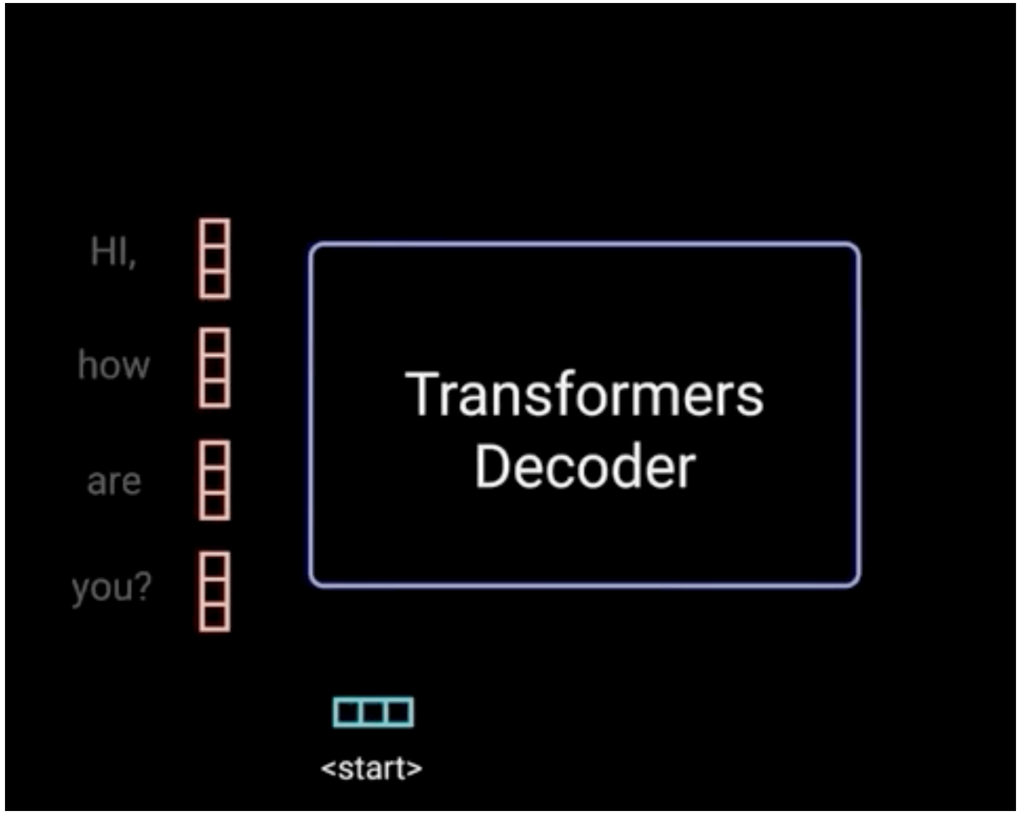
$$\hat{x}_{i,k} = \frac{x_{i,k} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}}$$

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

Image Credit: [6] & [7] & https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

Decoder

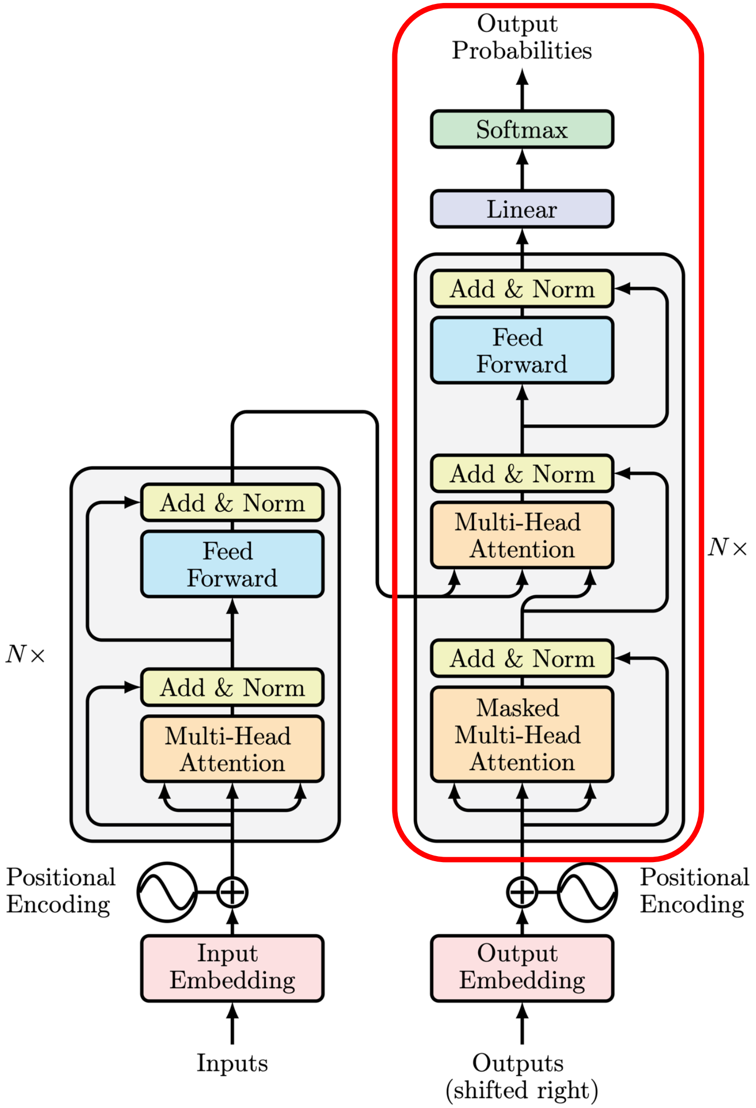


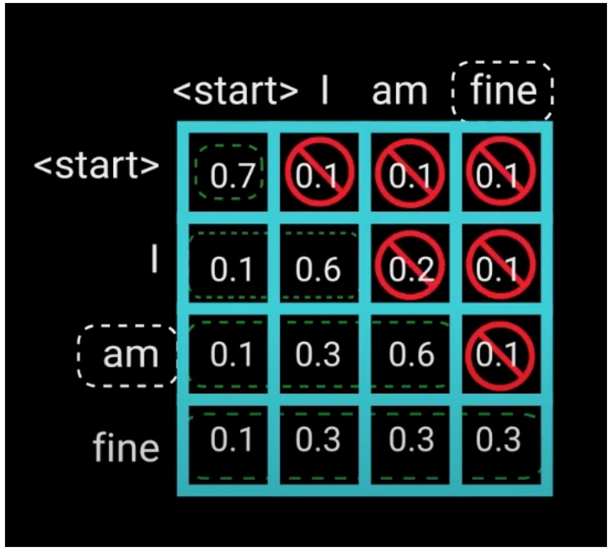


For certain applications like language models, decoder should be autoregressive!

Image Credit: [6] & https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

Masked Multi-Head Attention

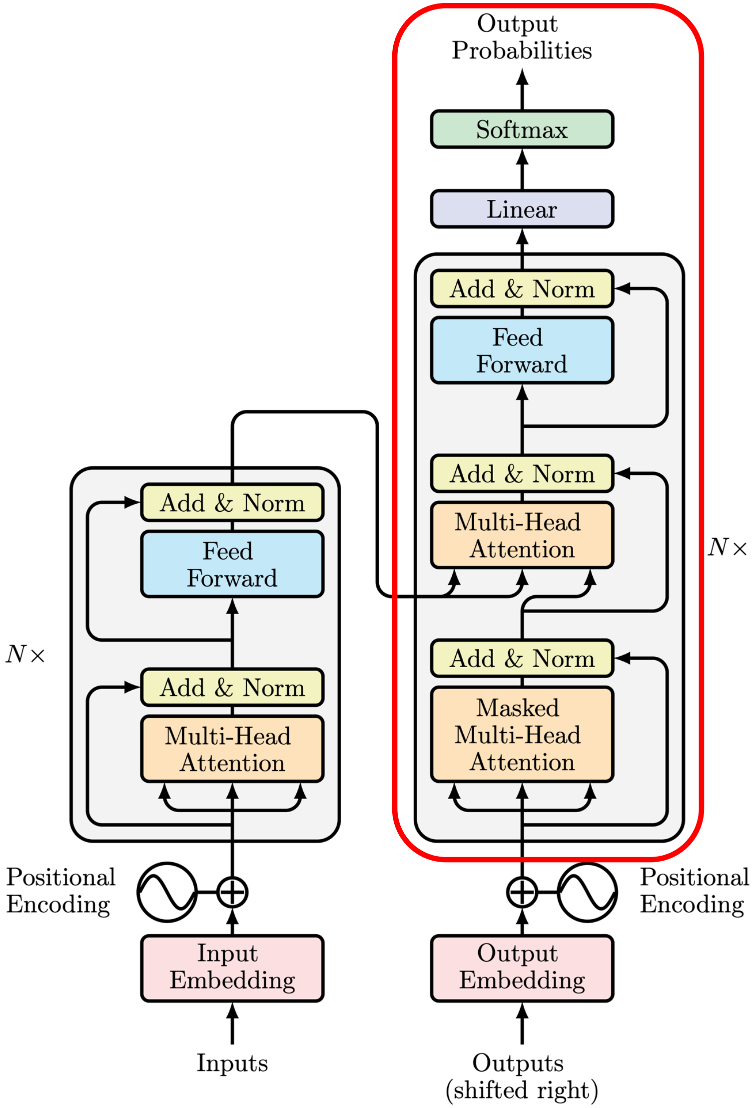


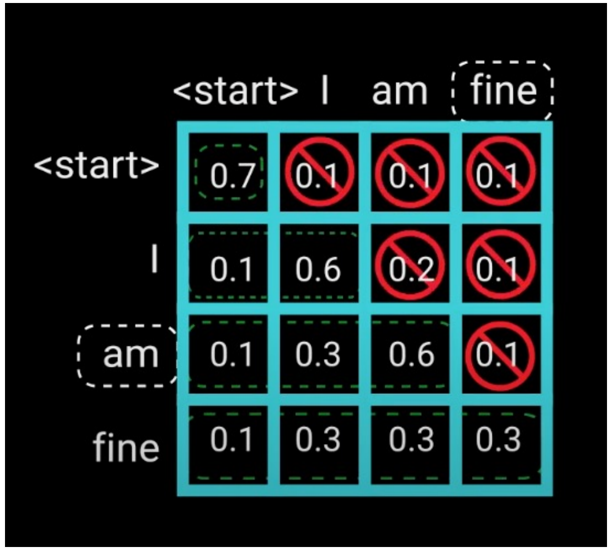


Prevent attending from future!

Image Credit: [6] & https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

Masked Multi-Head Attention





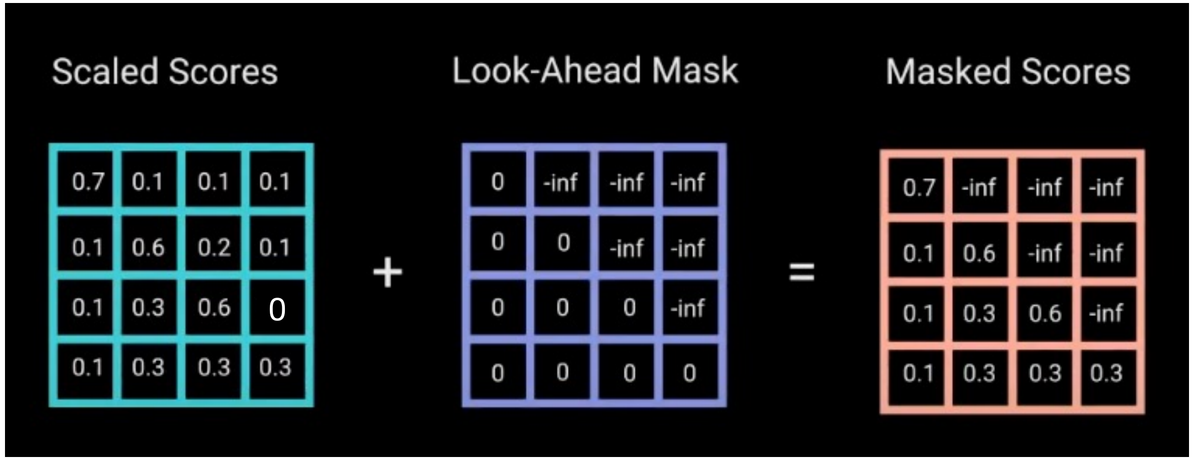
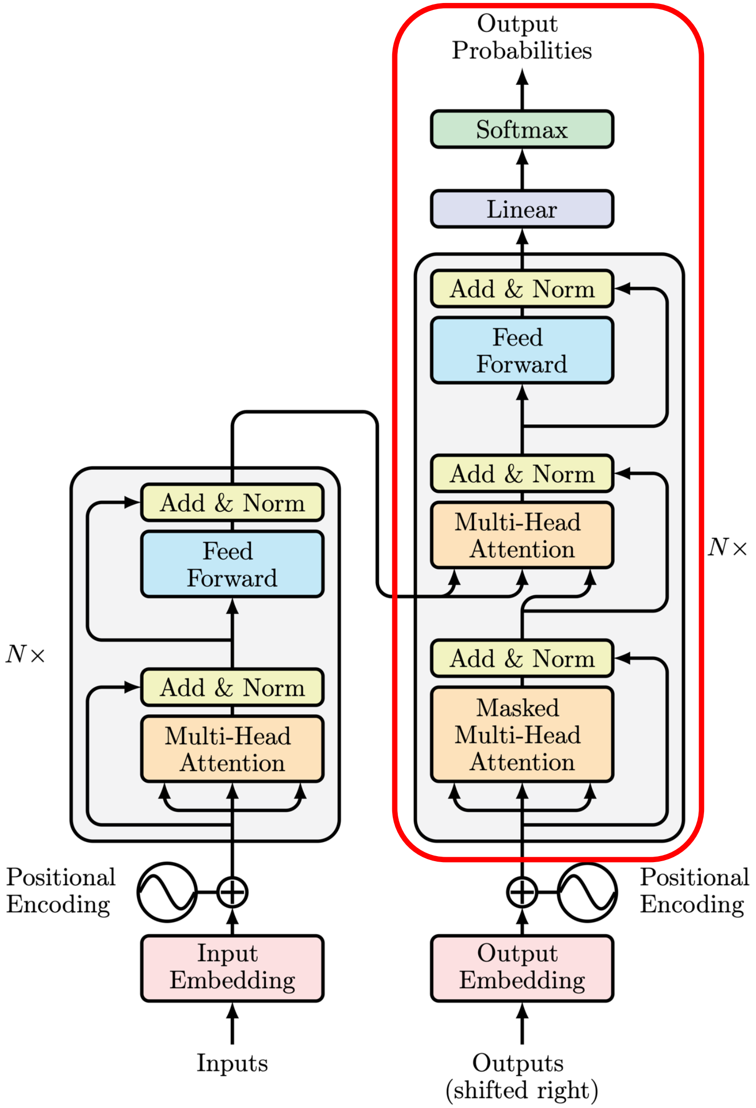
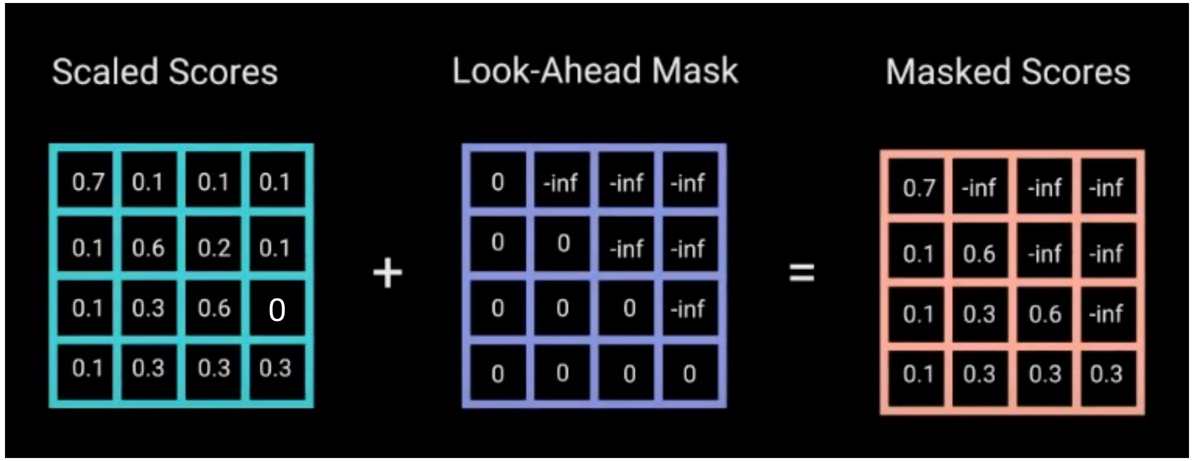


Image Credit: [6] & https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

Masked Multi-Head Attention





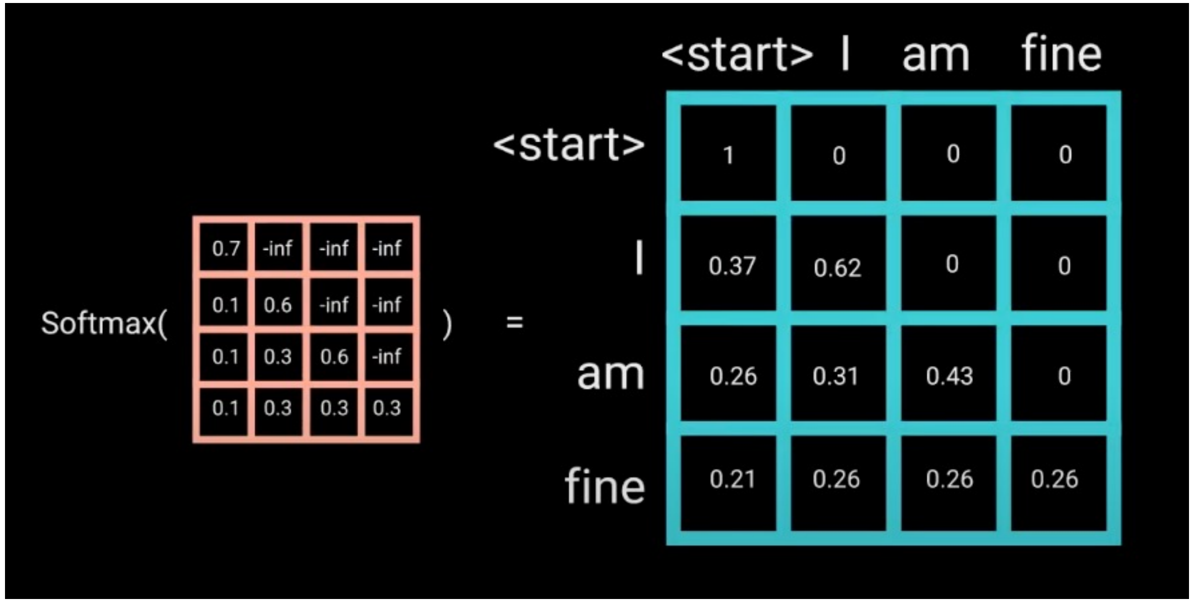


Image Credit: [6] & https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

Hugging Face Demos

https://transformer.huggingface.co/

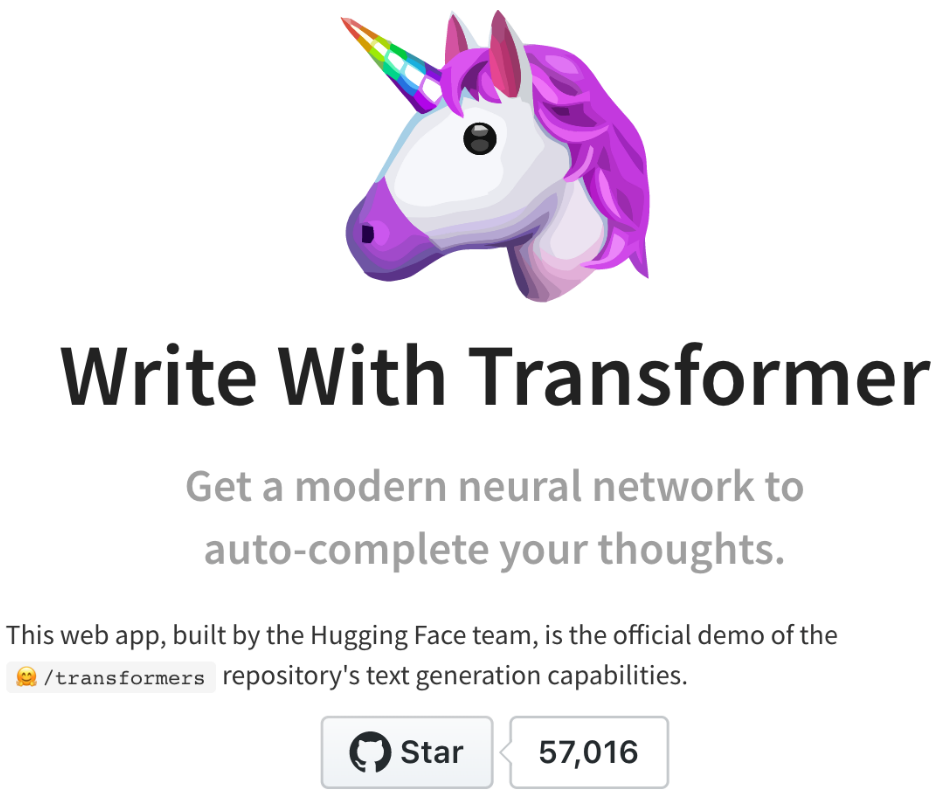
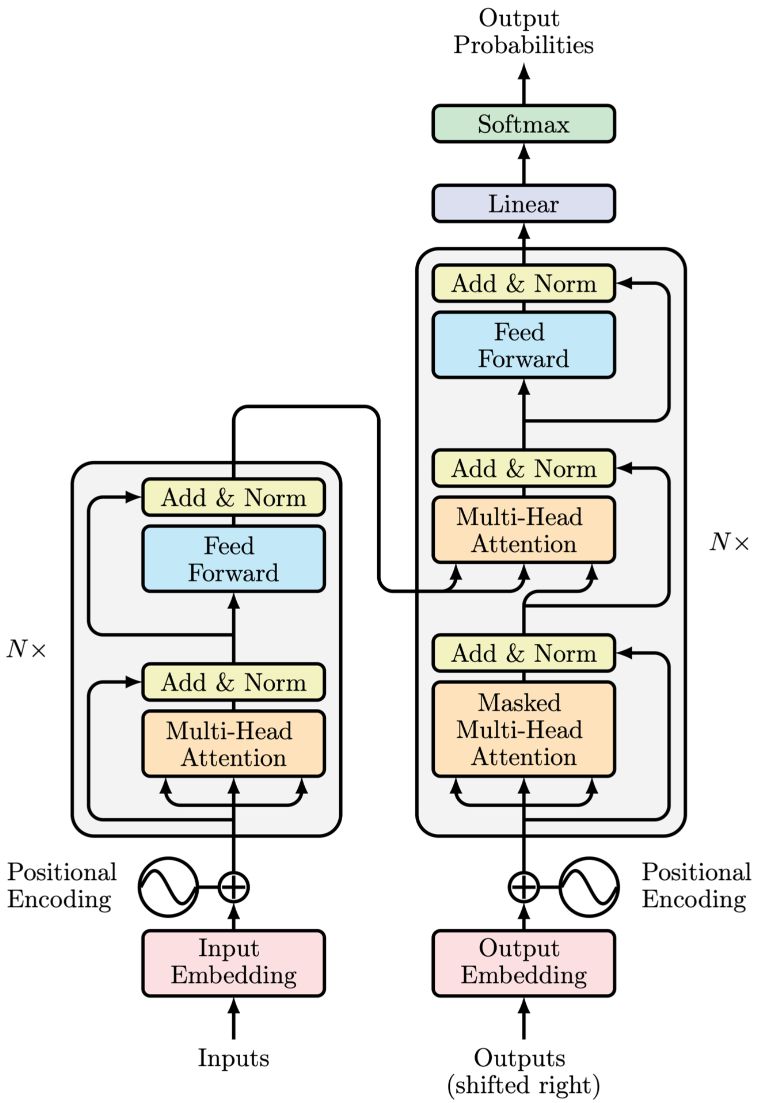


Image Credit: https://transformer.huggingface.co/

Limitations

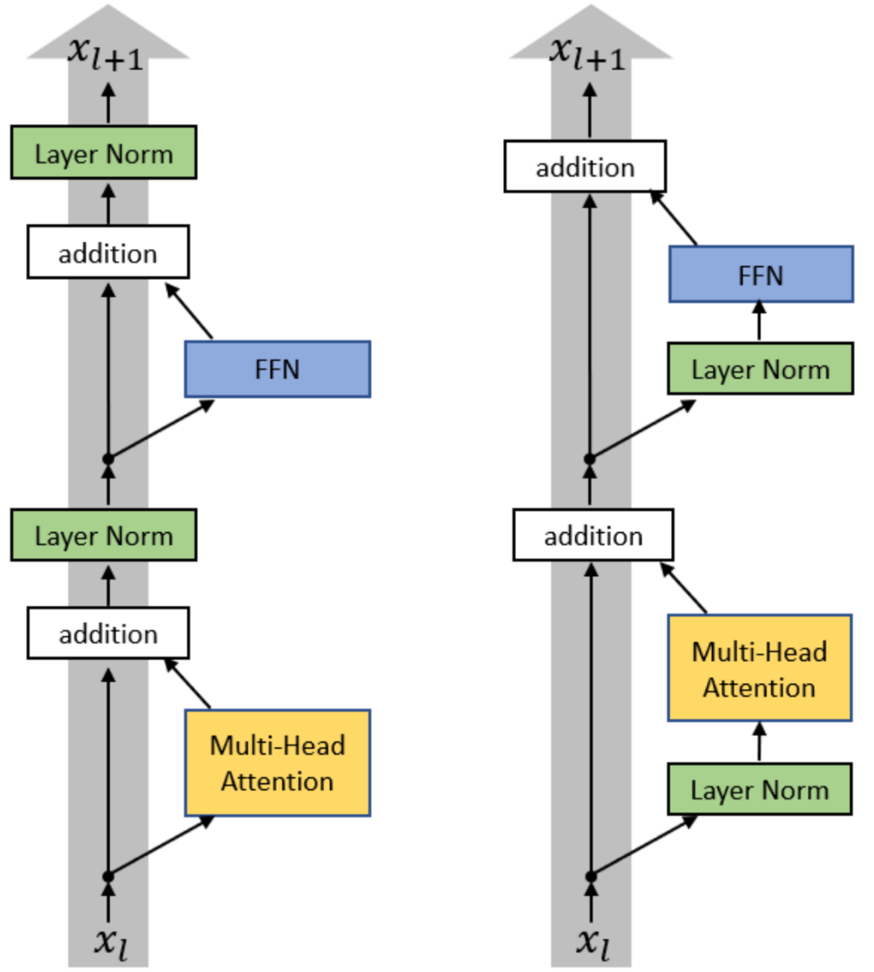


- O(L^2) time/memory cost for self-attention
- How can we incorporate prior knowledge into attention rather than having a fully connected attention?
  - Encourage sparse attention
  - Inject known graph structures
  - ……

Image Credit: [6]

Pre-Norm vs. Post-Norm

Where to place the Layer Normalization?



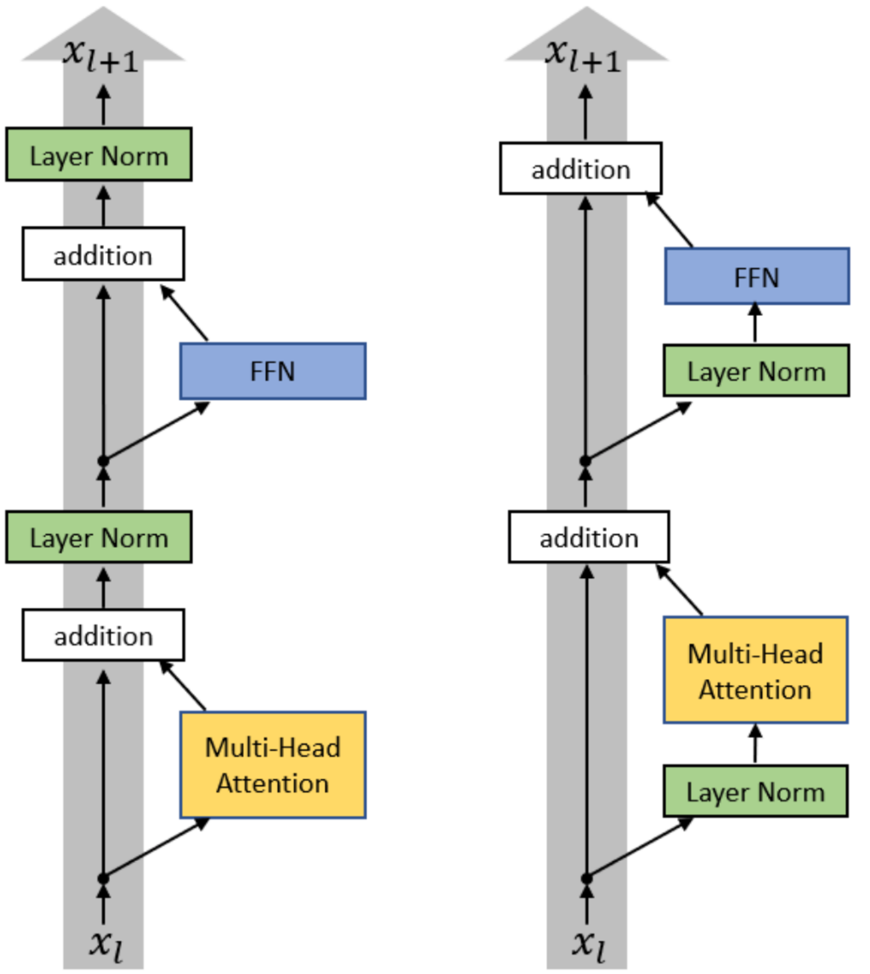
Post-Norm Pre-Norm

Image Credit: [8]

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: [8]

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

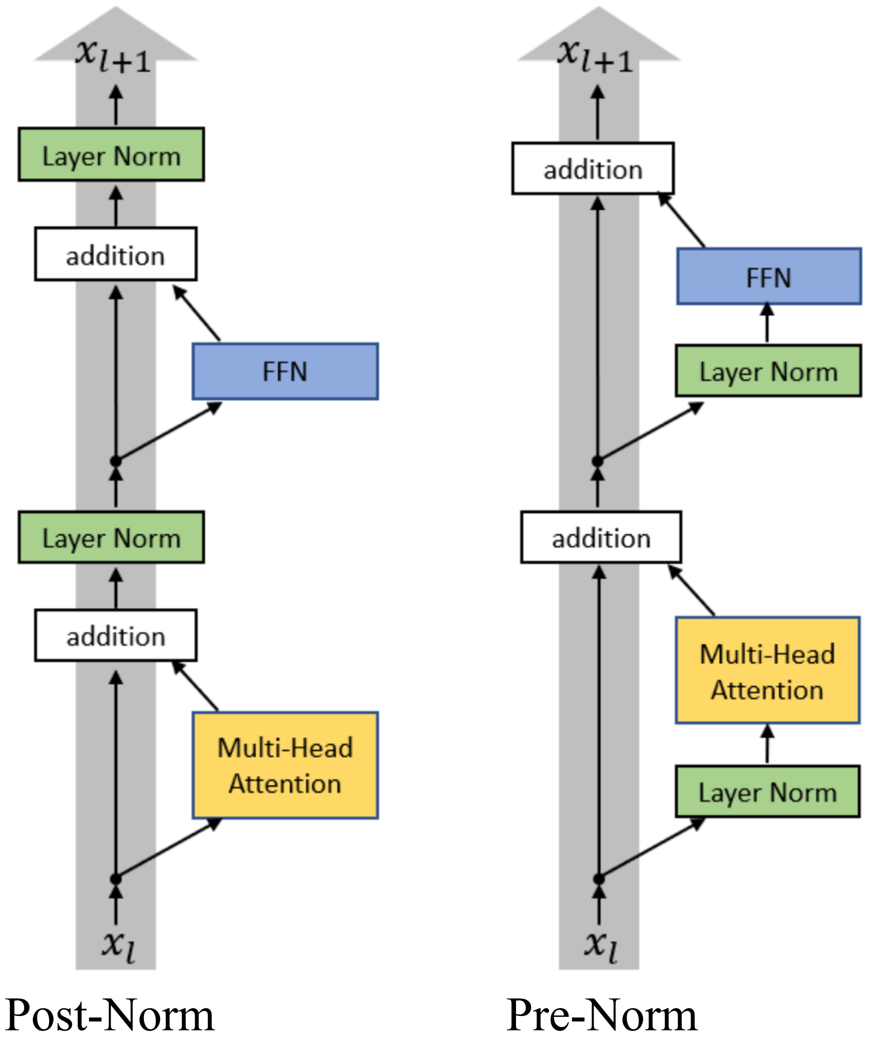


Image Credit: [8]

Extensions: Vision Transformer

Image Credit: [9] & https://github.com/lucidrains/vit-pytorch

Extensions: Swin Transformer

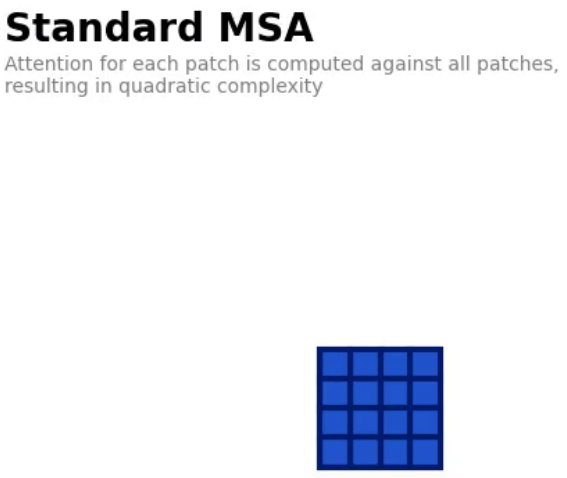
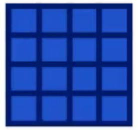


Image Credit: https://towardsdatascience.com/a-comprehensive-guide-to-swin-transformer-64965f89d14c

Extensions: Swin Transformer

Standard MSA

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

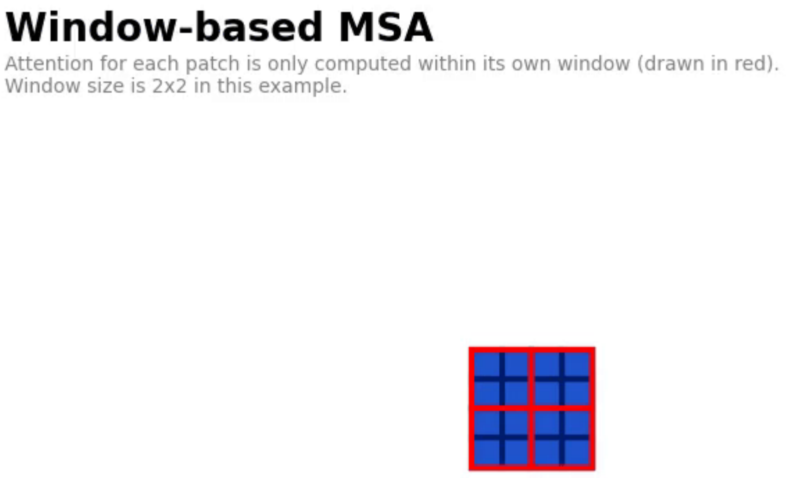


Window-based MSA

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

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

Extensions: Swin Transformer



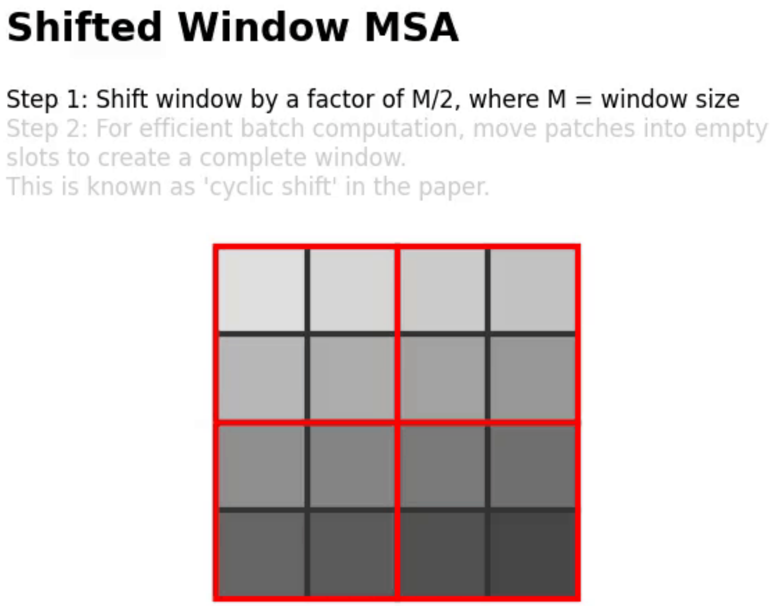
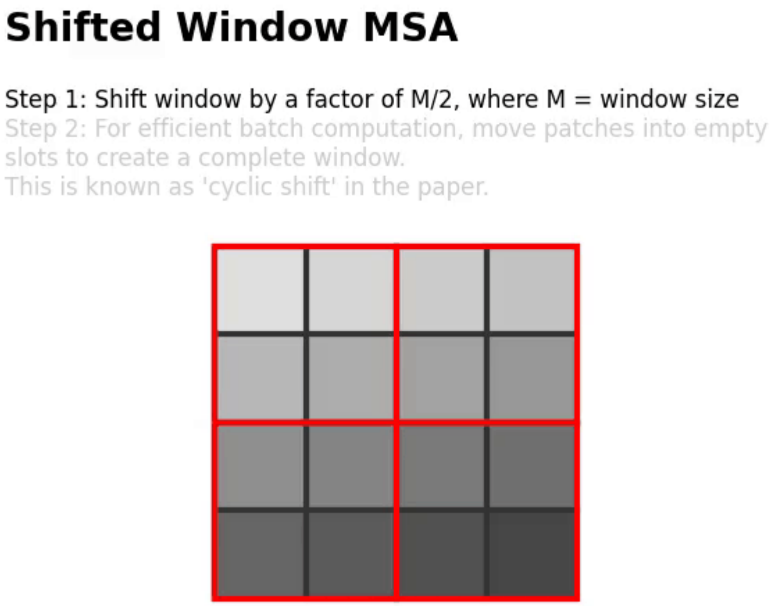


Image Credit: https://towardsdatascience.com/a-comprehensive-guide-to-swin-transformer-64965f89d14c

Extensions: Swin Transformer



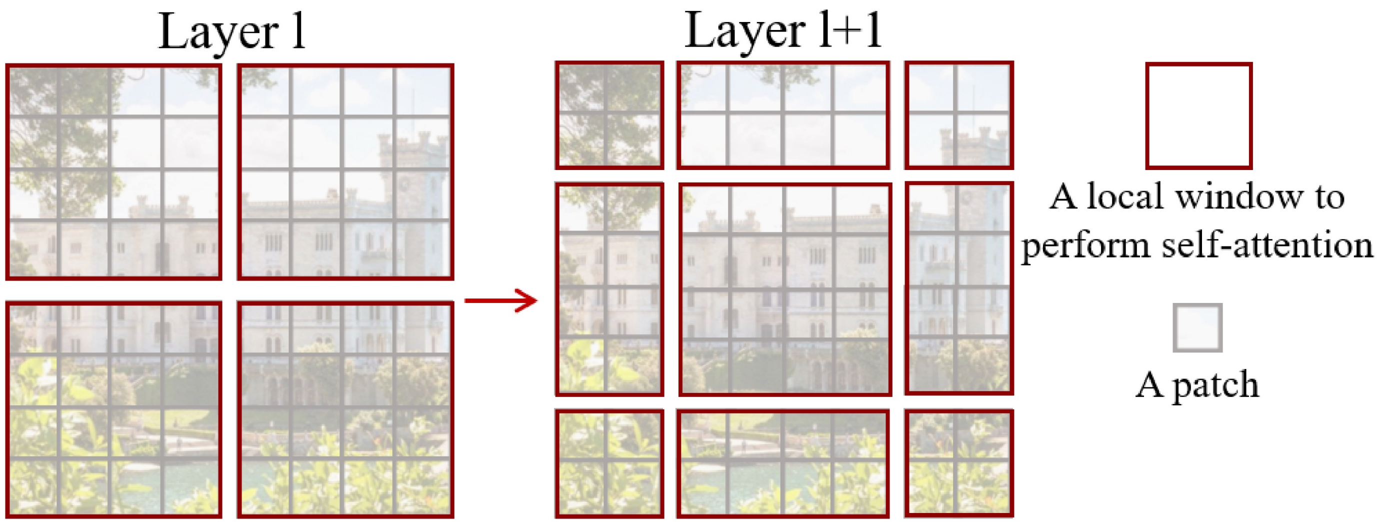


Image Credit: https://towardsdatascience.com/a-comprehensive-guide-to-swin-transformer-64965f89d14c, [10]

Extensions: Swin Transformer

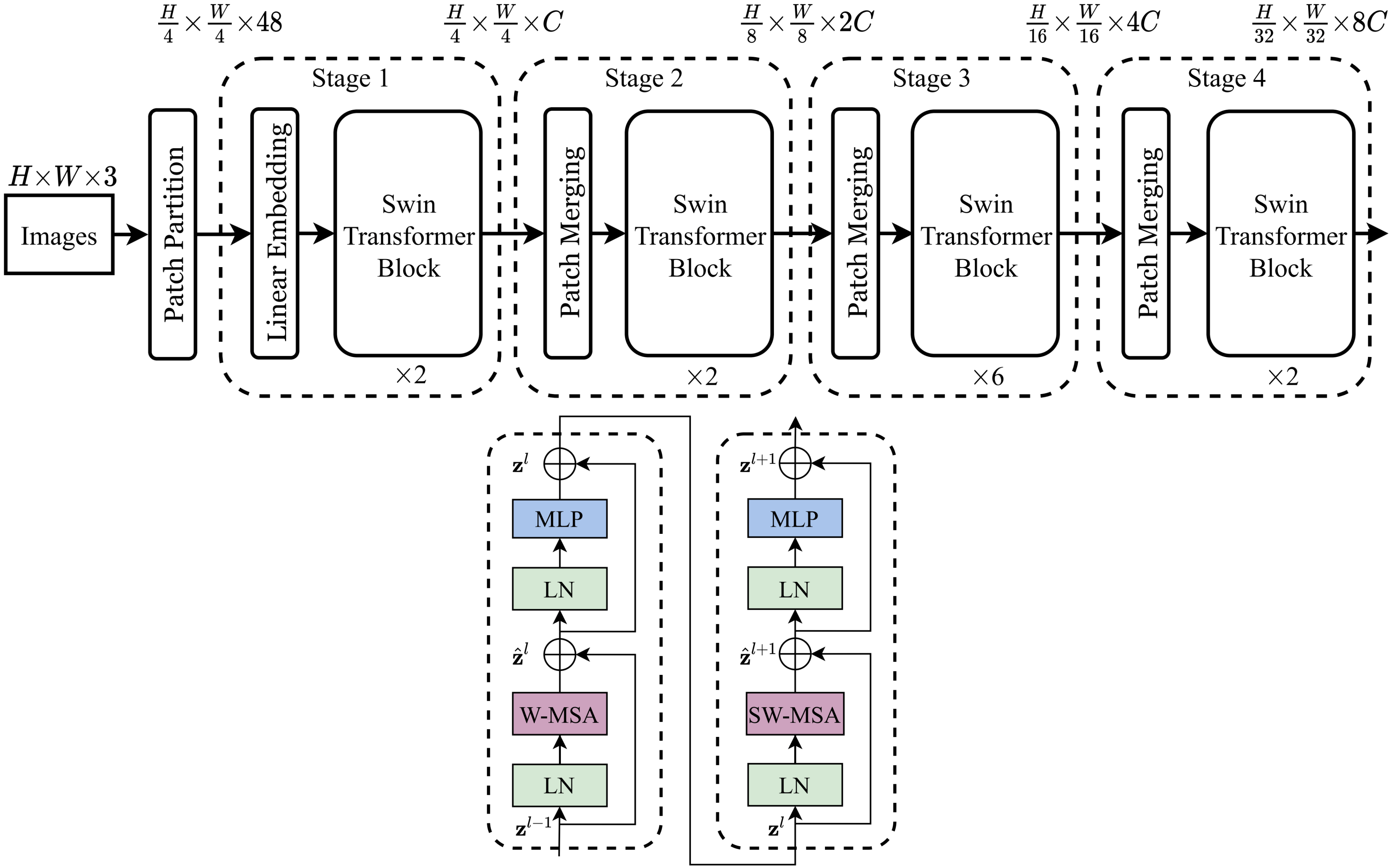


Image Credit: [10]

References

[1] Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. and Smola, A., 2017. Deep sets. arXiv preprint arXiv:1703.06114.

[2] Qi, C.R., Su, H., Mo, K. and Guibas, L.J., 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 652-660).

[3] Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. Neural computation, 9(8), pp.1735-1780.

[4] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y., 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

[5] Sutskever, I., Vinyals, O. and Le, Q.V., 2014. Sequence to sequence learning with neural networks. In Advances in neural information processing systems (pp. 3104-3112).

[6] 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).

[7] Ba, J.L., Kiros, J.R. and Hinton, G.E., 2016. Layer normalization. arXiv preprint arXiv:1607.06450.

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

[9] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S. and Uszkoreit, J., 2020. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.

[10] 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).

Questions?