CSCI 544: Applied Natural Language Processing

Transformer-II

Xuezhe Ma (Max)

USC University of Southern California

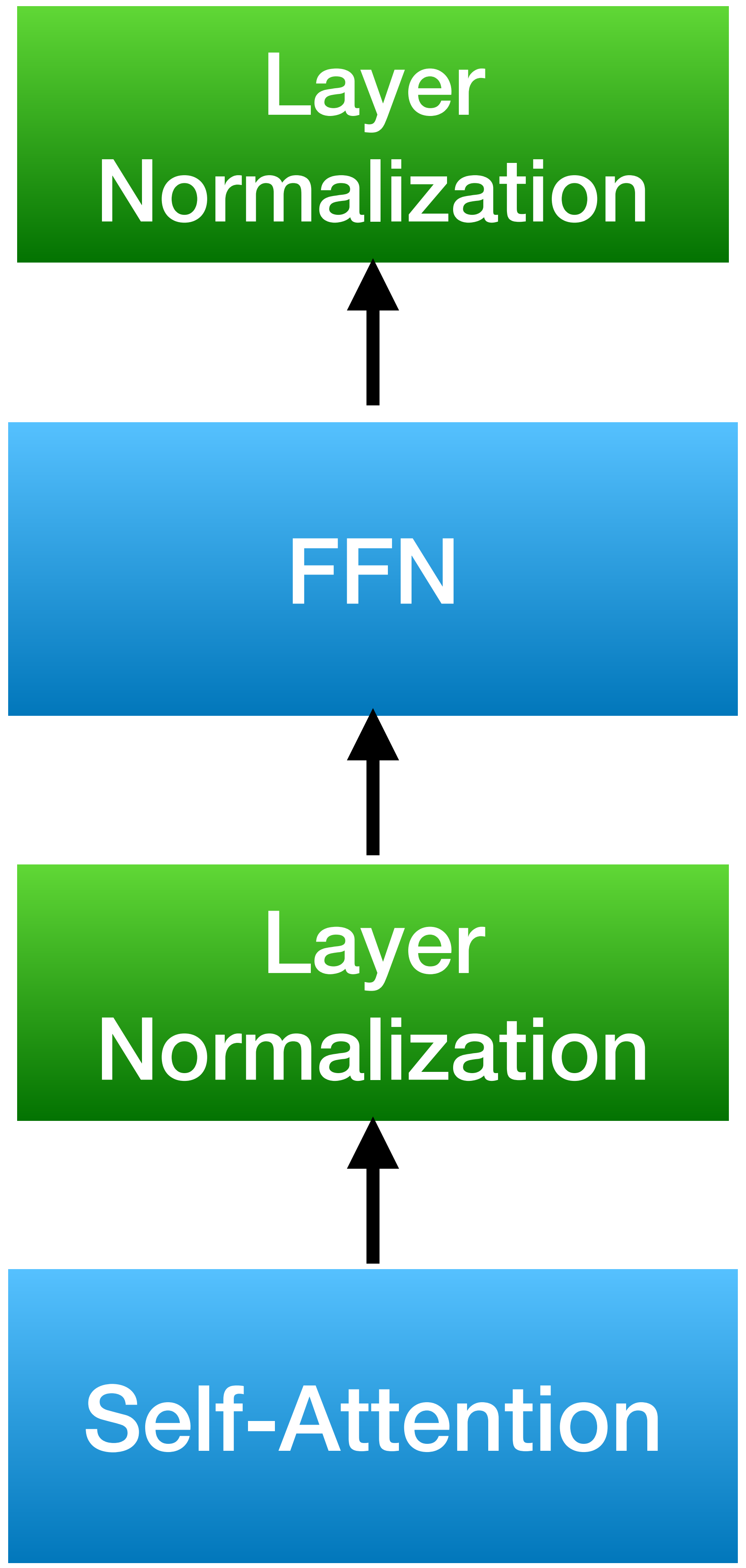
Course Organization

NLP Tasks

Models	Text Classification	Sequence Labeling	Syntactic Parsing	Seq2seq & MT	Language Modeling & Generation
	Naive Bayes Logistic Regression SVM …	HMMs MEMMs CRFs	Tree-structured Models/Algorithms	……	……
	Word Embedding & Neural Networks	RNN LSTM	Neural Structured Networks	$\boxed{\text{Attention \& Transformers}}$	Generative Models & LLMs

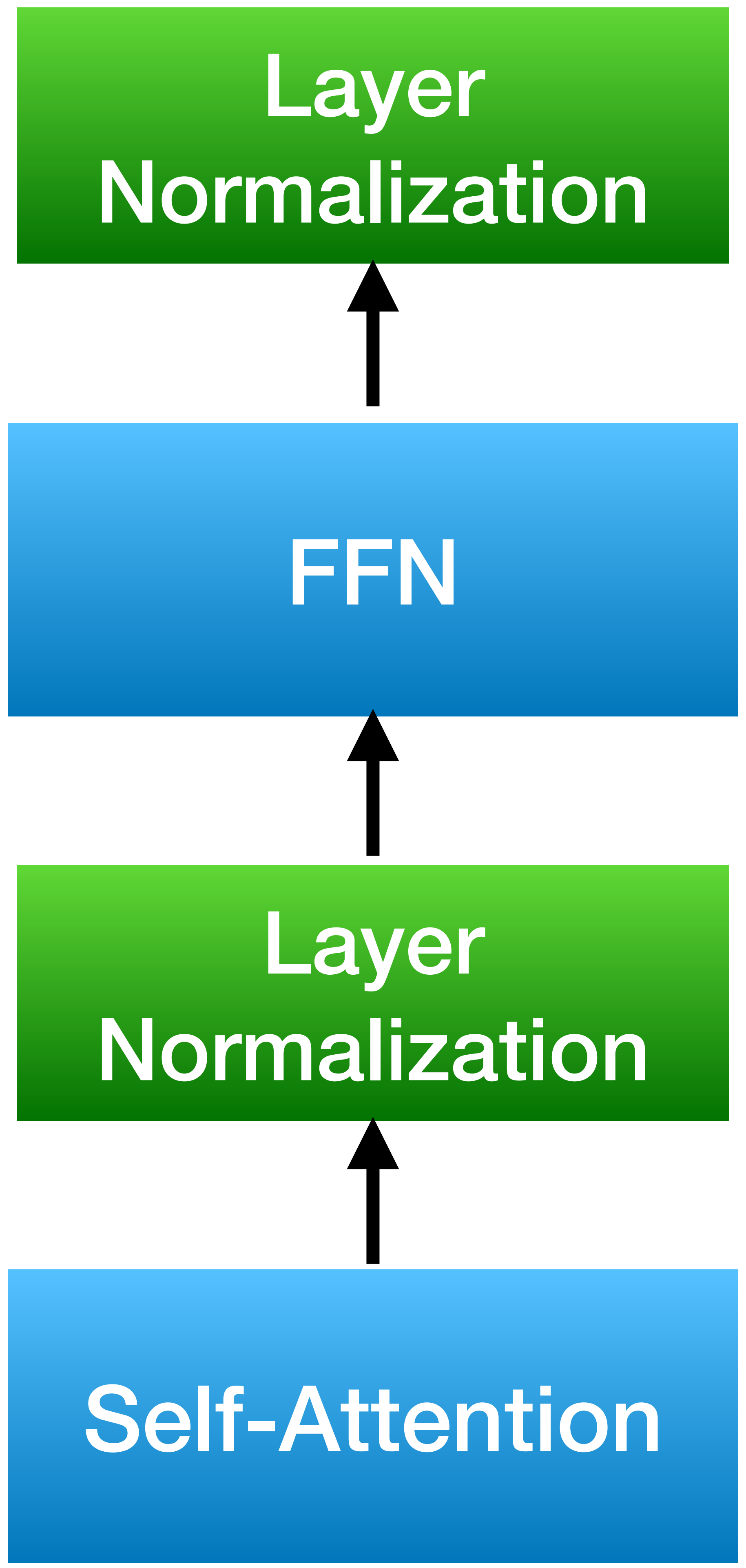
Transformers

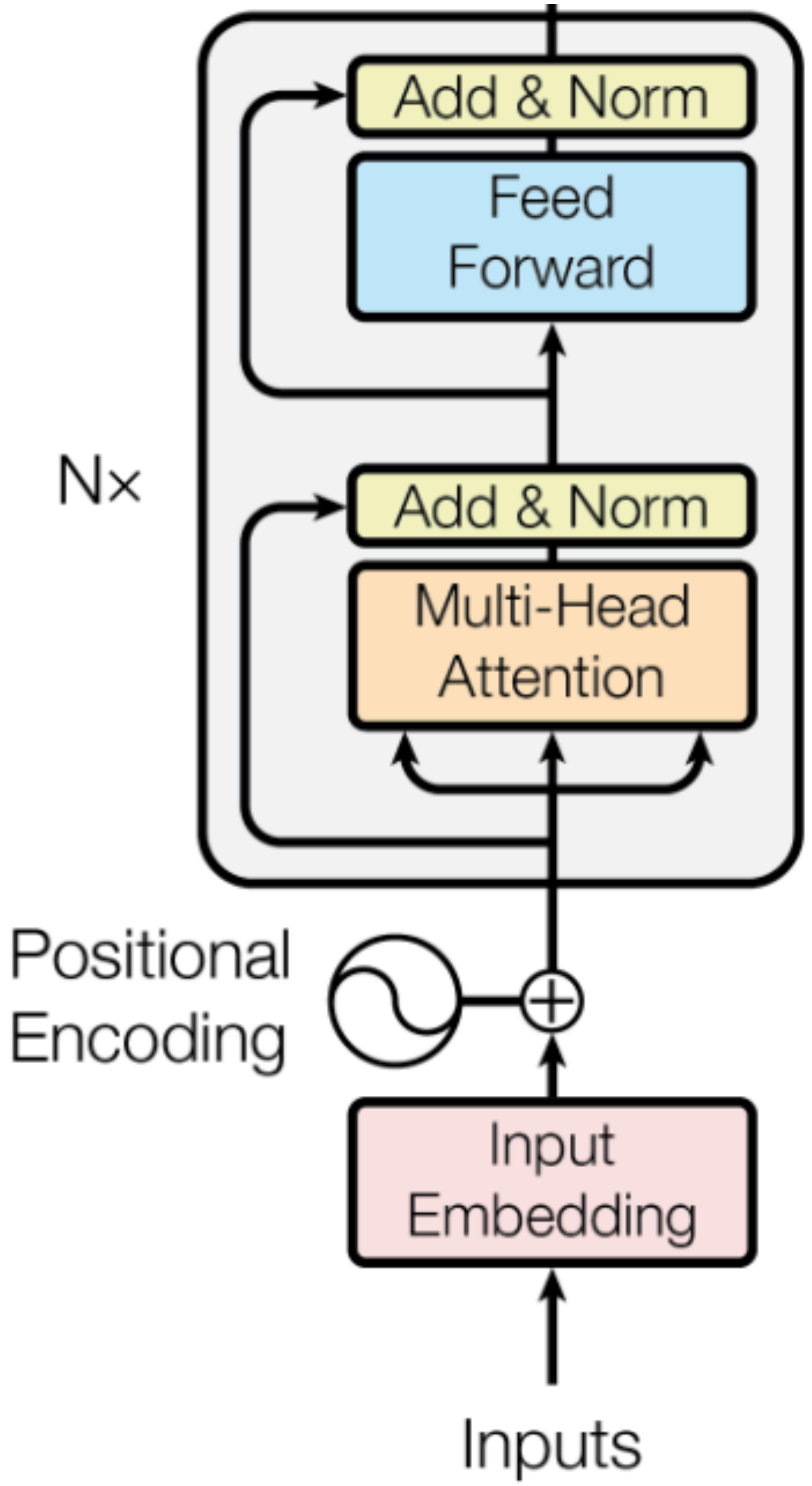
Transformer Encoder Block



- **Three Key Components**
  - (Masked) Multi-head Self-Attention
  - Layer Normalization
  - Position-wise Feed-Forward Network

Transformers





Outline

- **Residual Connections**
- **Normalization Layers**
  - Variants of Normalization Layers
  - Pre-Norm vs. Post-Norm
- **Positional Embeddings**
  - Absolute Positional Embeddings
  - Rotary Positional Embeddings

Residual Connections

Residual Connections

Deep Residual Learning for Image Recognition

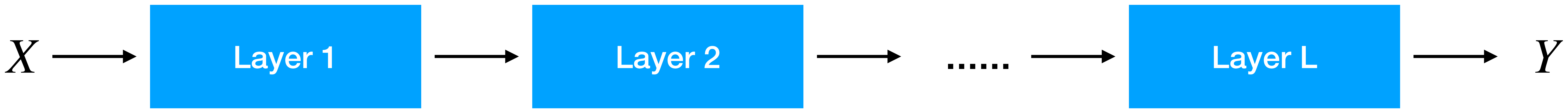
Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

Deep residual learning for **image recognition**
K He, X Zhang, S Ren, J Sun - … and **pattern recognition**, 2016 - openaccess.thecvf.com
… **Deeper** neural **networks** are more difficult to train. We present a **residual learning** framework to ease the training of **networks** that are substantially **deeper** than those used previously. …

Residual Connections

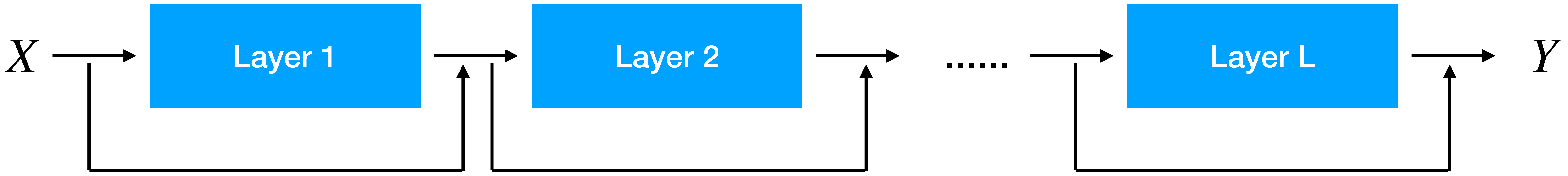
- **Deep Neural Networks**

$$Y_1 = f_1(X) \qquad Y_2 = f_2(Y_1)$$

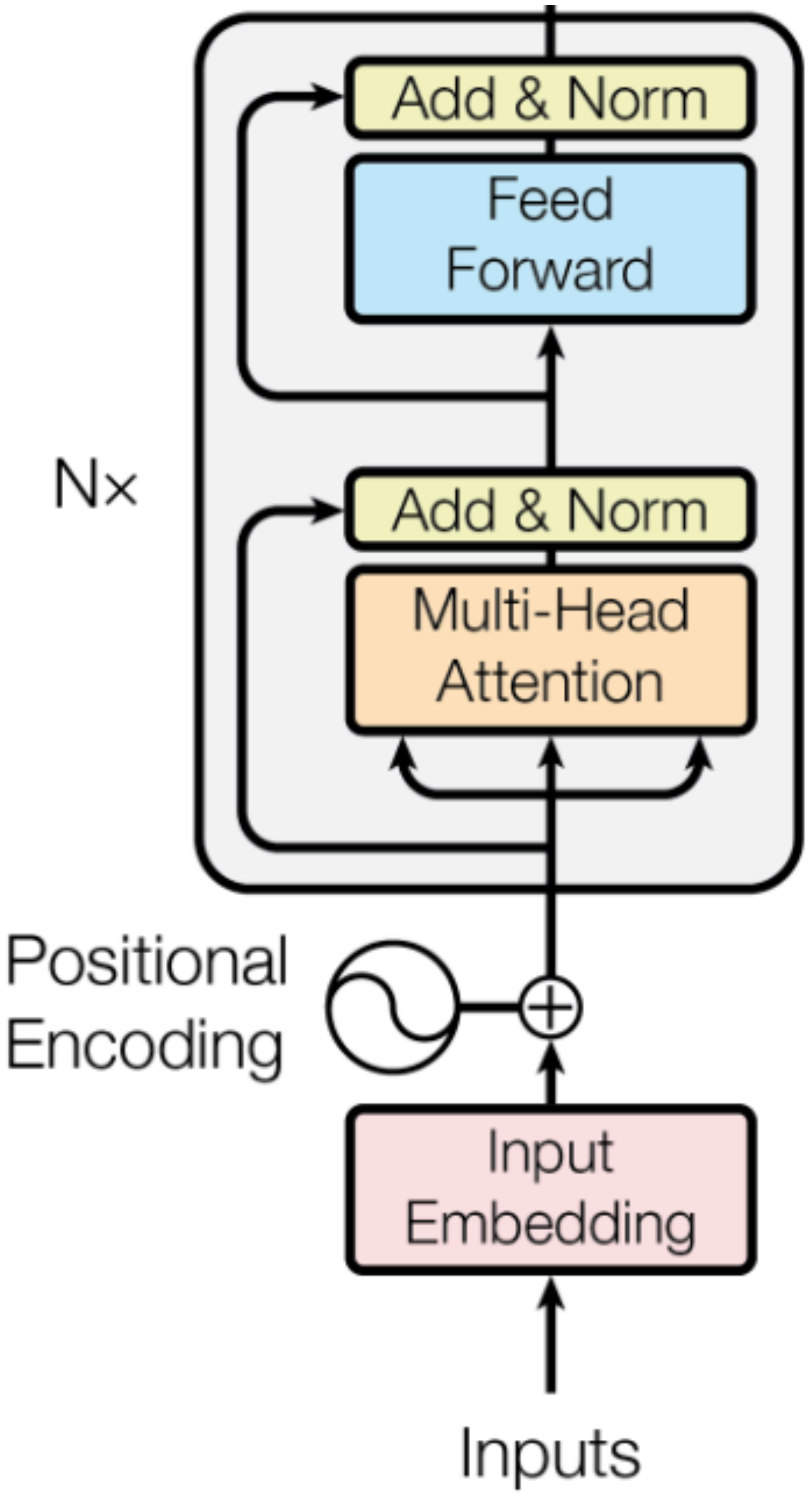


- **Residual Connections**

$$Y = f_1(X) + X \qquad Y_2 = f_2(Y_1) + Y_1$$



Residual Connection in Transformers



$$Y_1 = \textbf{LayerNorm}\left(\textbf{SelfAttn}(X) \boxed{+ X}\right)$$

$$Y_2 = \textbf{LayerNorm}\left(\textbf{FFN}(Y_1) \boxed{+ Y_1}\right)$$

Outline

- Residual Connections
- **Normalization Layers**
  - Variants of Normalization Layers
  - Pre-Norm vs. Post-Norm
- **Positional Embeddings**
  - Absolute Positional Embeddings
  - Rotary Positional Embeddings

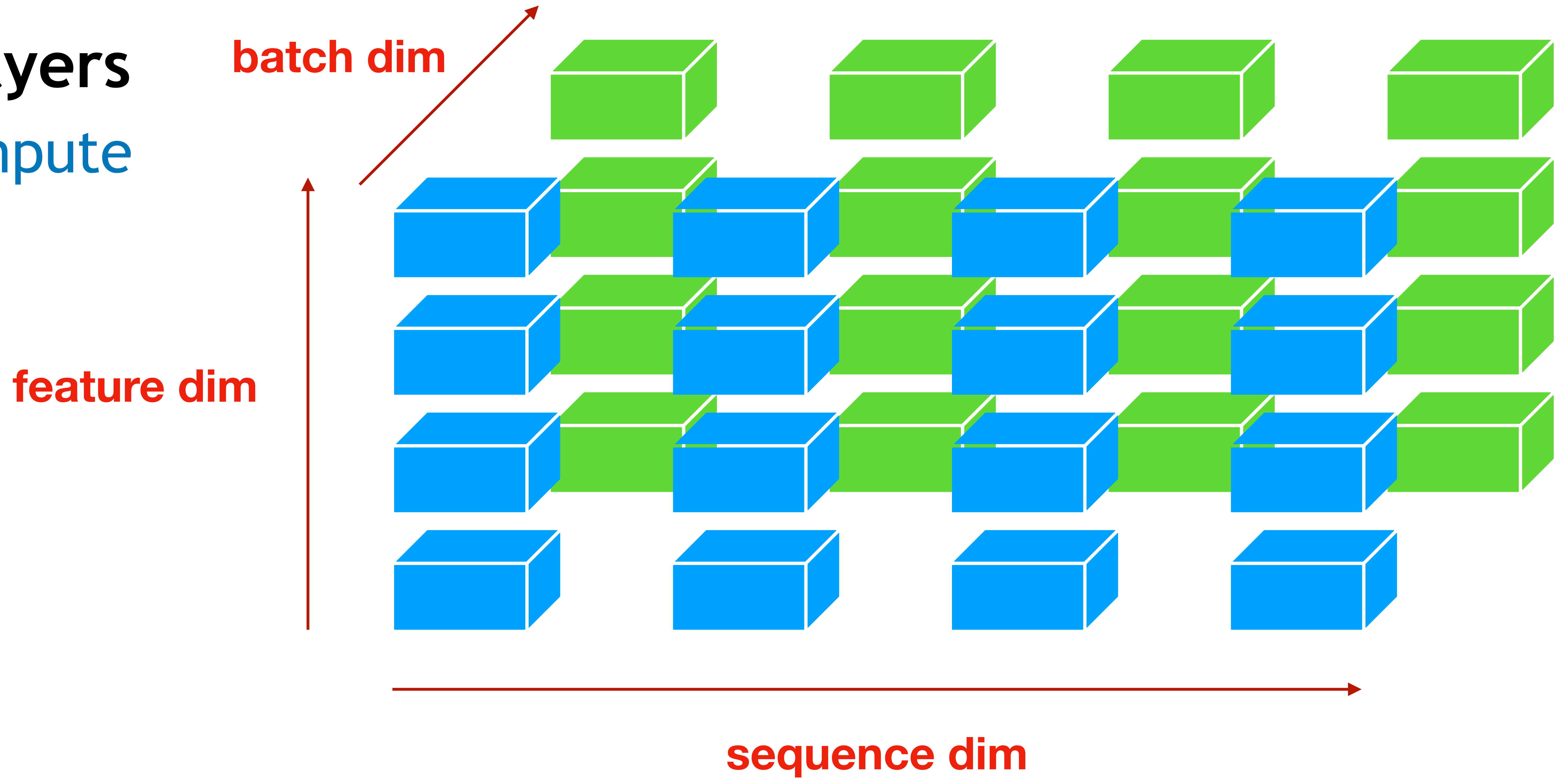
Normalization Layers

Normalization Layers

- **General Form:**

$$Y = \frac{X - \mathrm{E}[X]}{\sqrt{\mathrm{Var}[X] + \epsilon}} * \gamma + \beta$$

- **Variants of Normalization Layers**
  - Along which dimensions to compute expectation and variance
  - Batch Normalization
  - Layer Normalization
  - ......



Batch Normalization

- **First Proposed for CV models**
  - Loffe and Szegedy, 2014

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

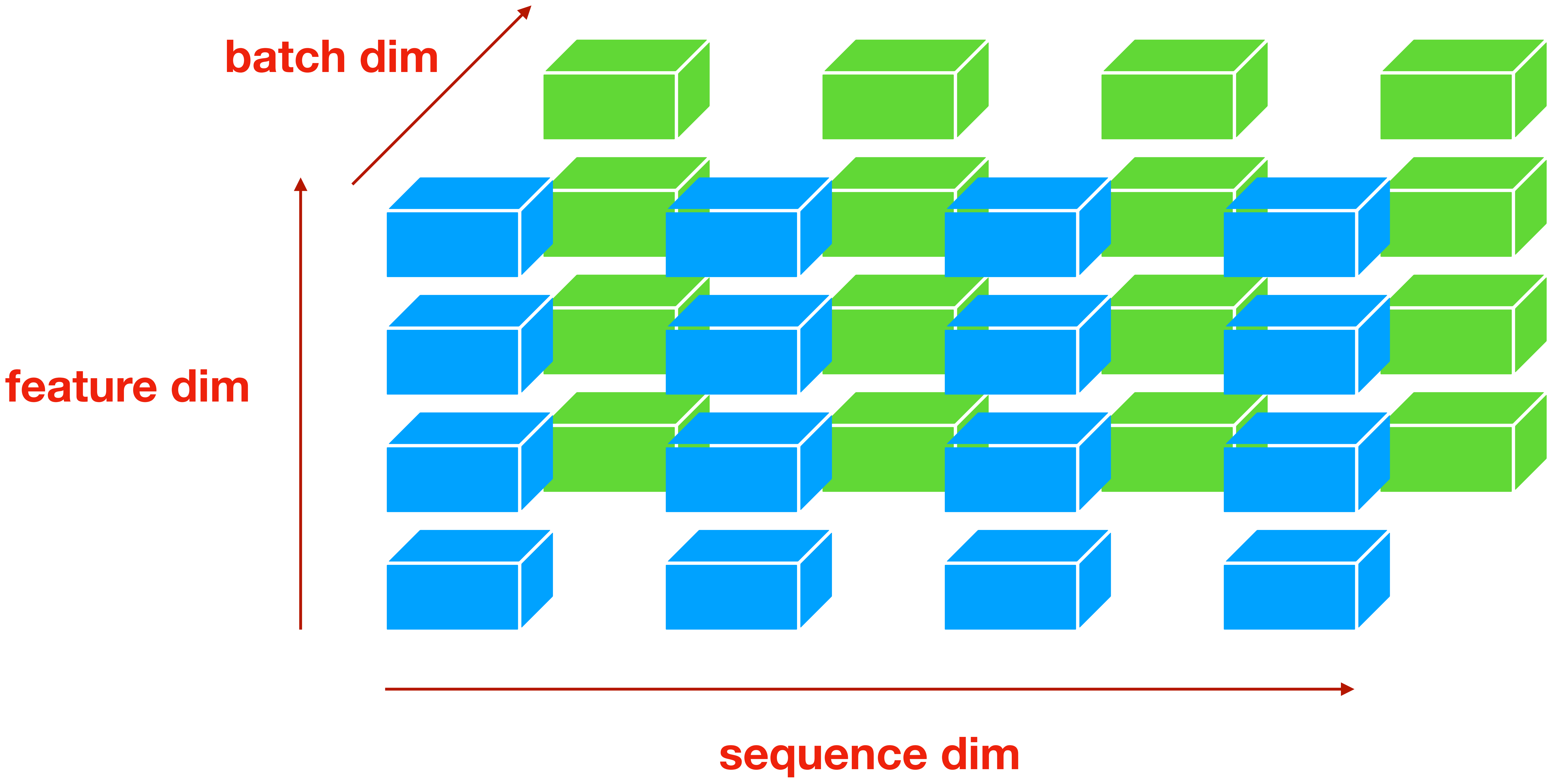
Sergey Ioffe SIOFFE@GOOGLE.COM
Christian Szegedy SZEGEDY@GOOGLE.COM
Google, 1600 Amphitheatre Pkwy, Mountain View, CA 94043

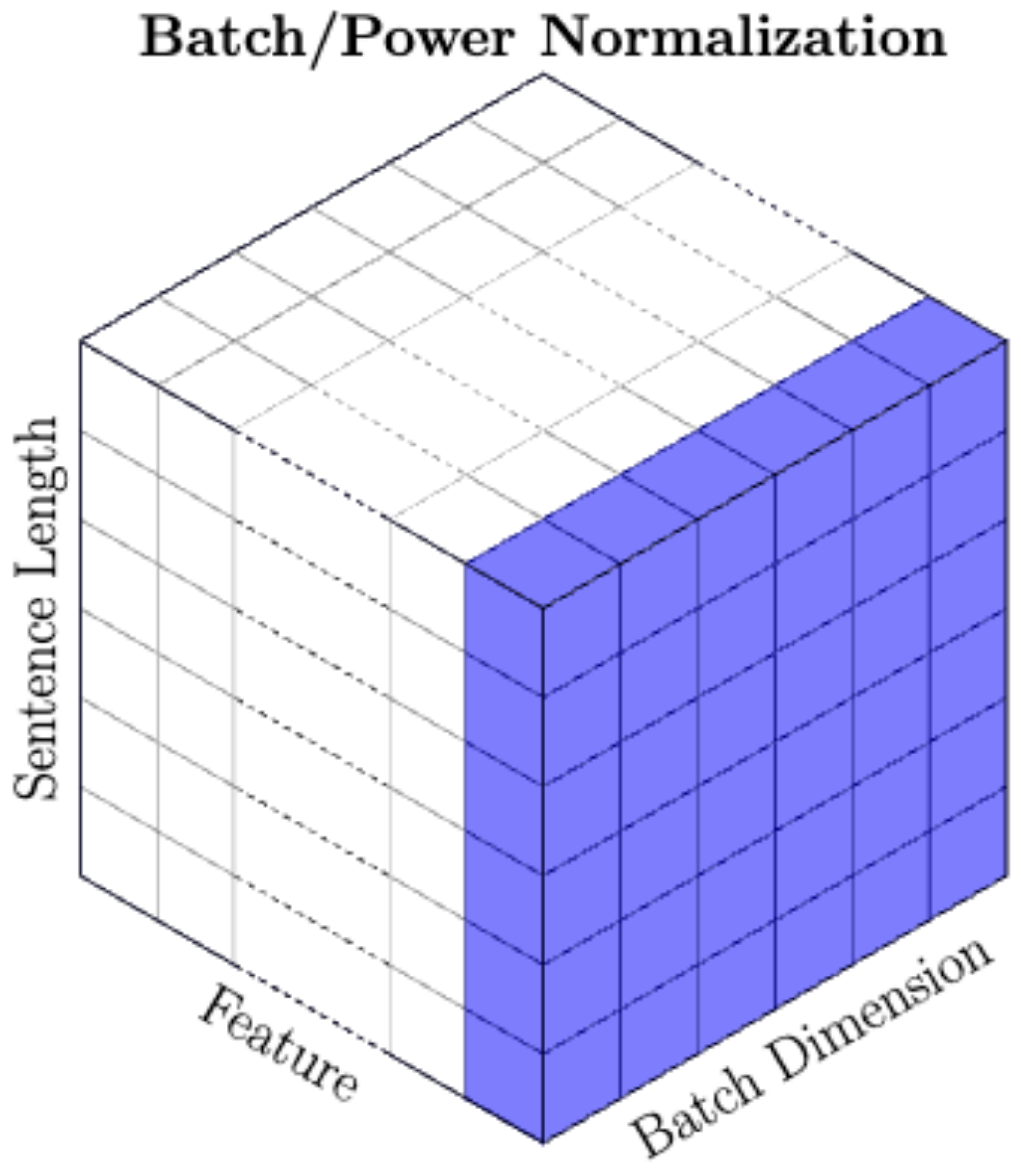
- **Two Basic Motivations**
  - Reducing internal covariate shift (bias) from data
  - Controlling output range/scale of each layer

Batch Normalization

- Normalize along both batch & spatial/sequential dimensions

$$Y = \frac{X - \mathrm{E}[X]}{\sqrt{\mathrm{Var}[X] + \epsilon}} * \gamma + \beta$$





Issues of Batch Normalization

- **Normalization involves batch dimension**
  - Batch size cannot be too small
  - When batch size < 8, batch normalization works poorly in practice

Hard to scale up

- **What are the expectation/variance in test time?**
  - Running stats for expectation/variance
  - At each mini-batch in training:
    - Compute mean and variance in this mini-batch

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

    - Update running mean and variance

Brittle for data from different domains

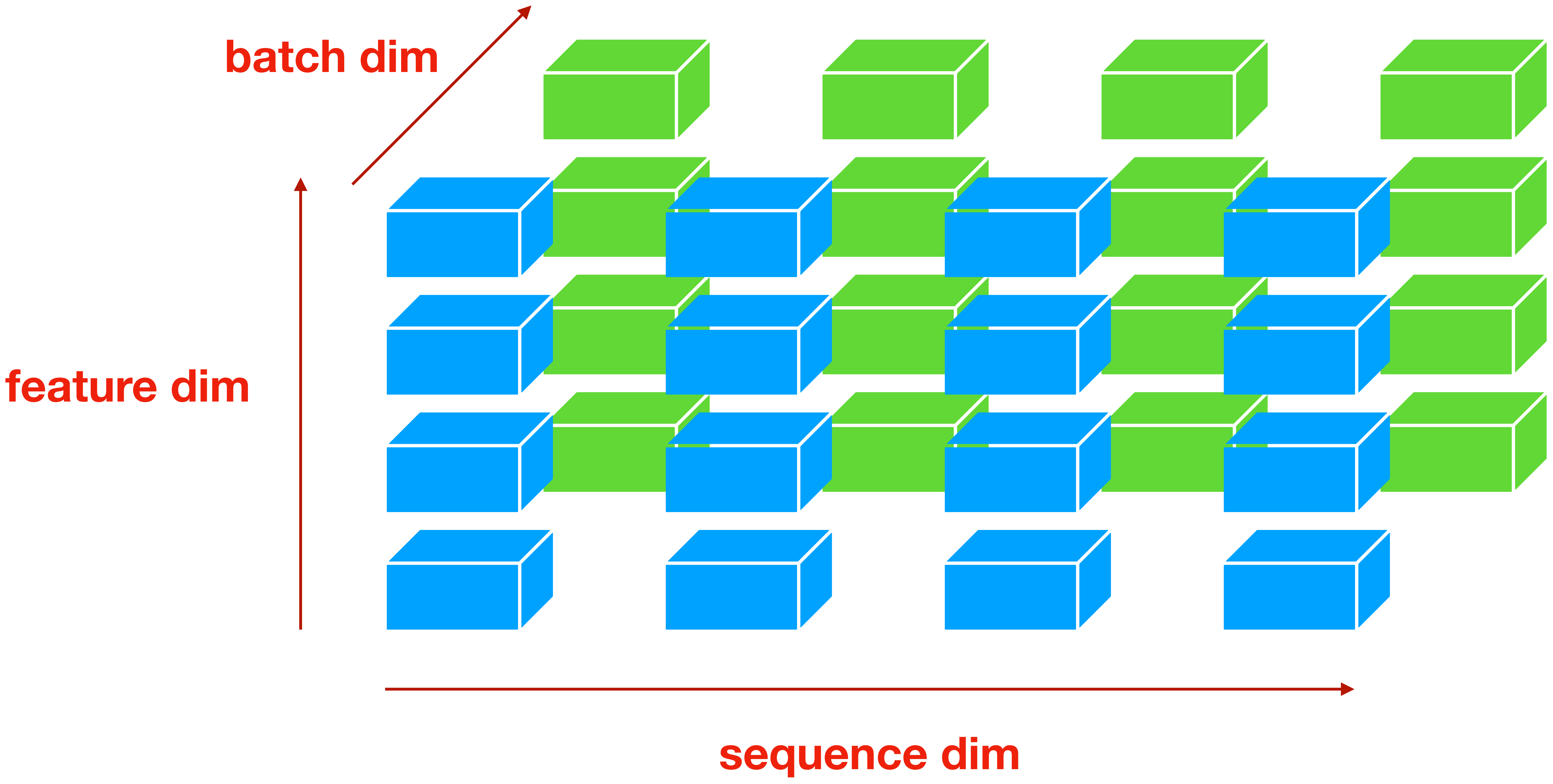
$$\mu \leftarrow \alpha\mu + (1-\alpha)\mu_{\mathcal{B}}$$

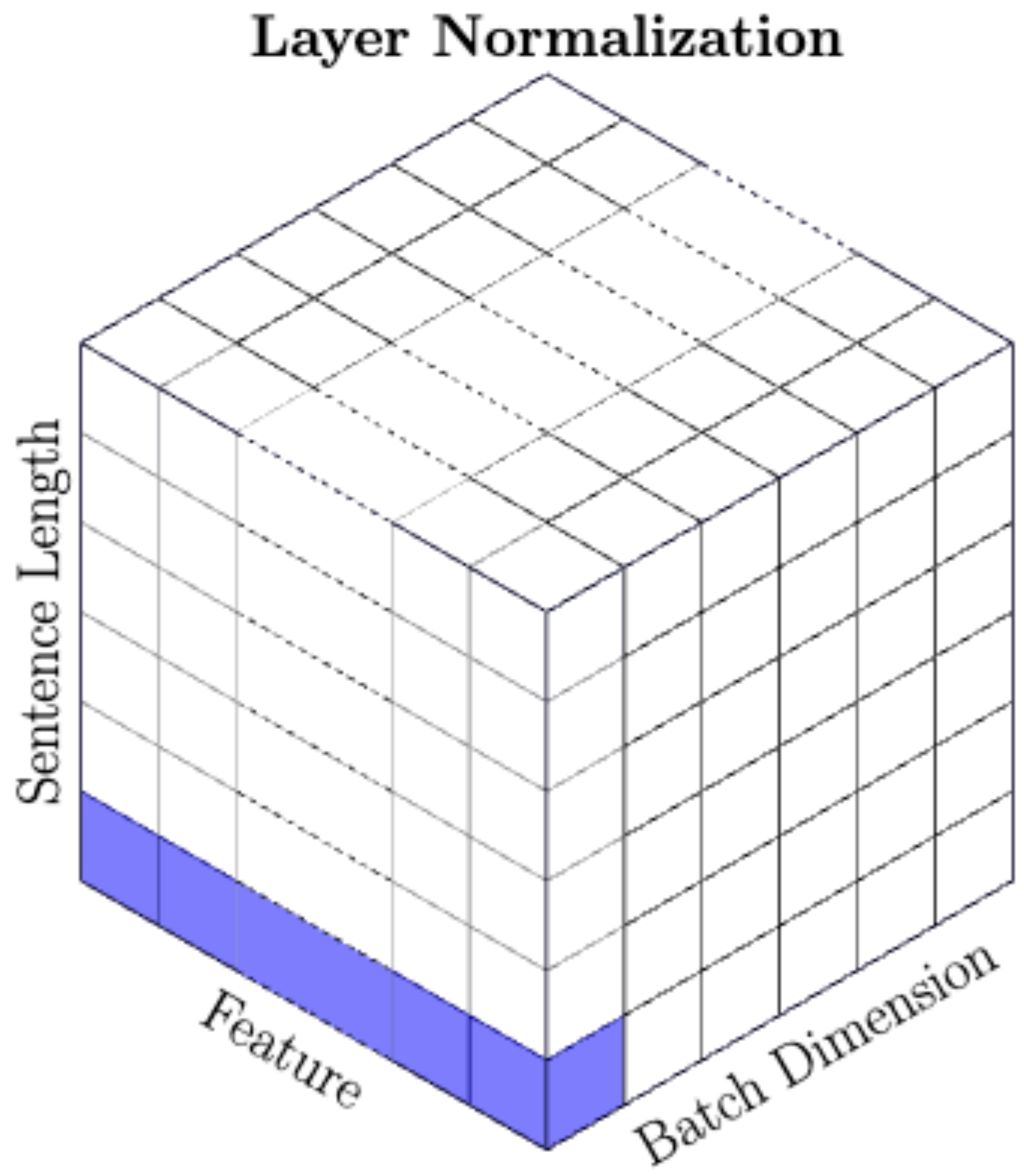
$$\sigma^2 \leftarrow \alpha\sigma^2 + (1-\alpha)\sigma_{\mathcal{B}}^2$$

Layer Normalization

- Normalize along the feature dimension

$$Y = \frac{X - \mathrm{E}[X]}{\sqrt{\mathrm{Var}[X] + \epsilon}} * \gamma + \beta$$





Layer Normalization

- **Pros:**
  - Does not involve the batch dimension (no need for running stats)
  - Does not involve sequential dimension
    - Easy to parallel
    - Working well for data w. various lengths (language)
- **Cons:**
  - Cannot reduce internal covariate shift/bias from data
    - Bias of data usually raises across different instances or steps
  - Root Mean Square (RMS) Normalization
    - Used in LLama

$$Y = \frac{X}{\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2 + \epsilon}} * \gamma$$

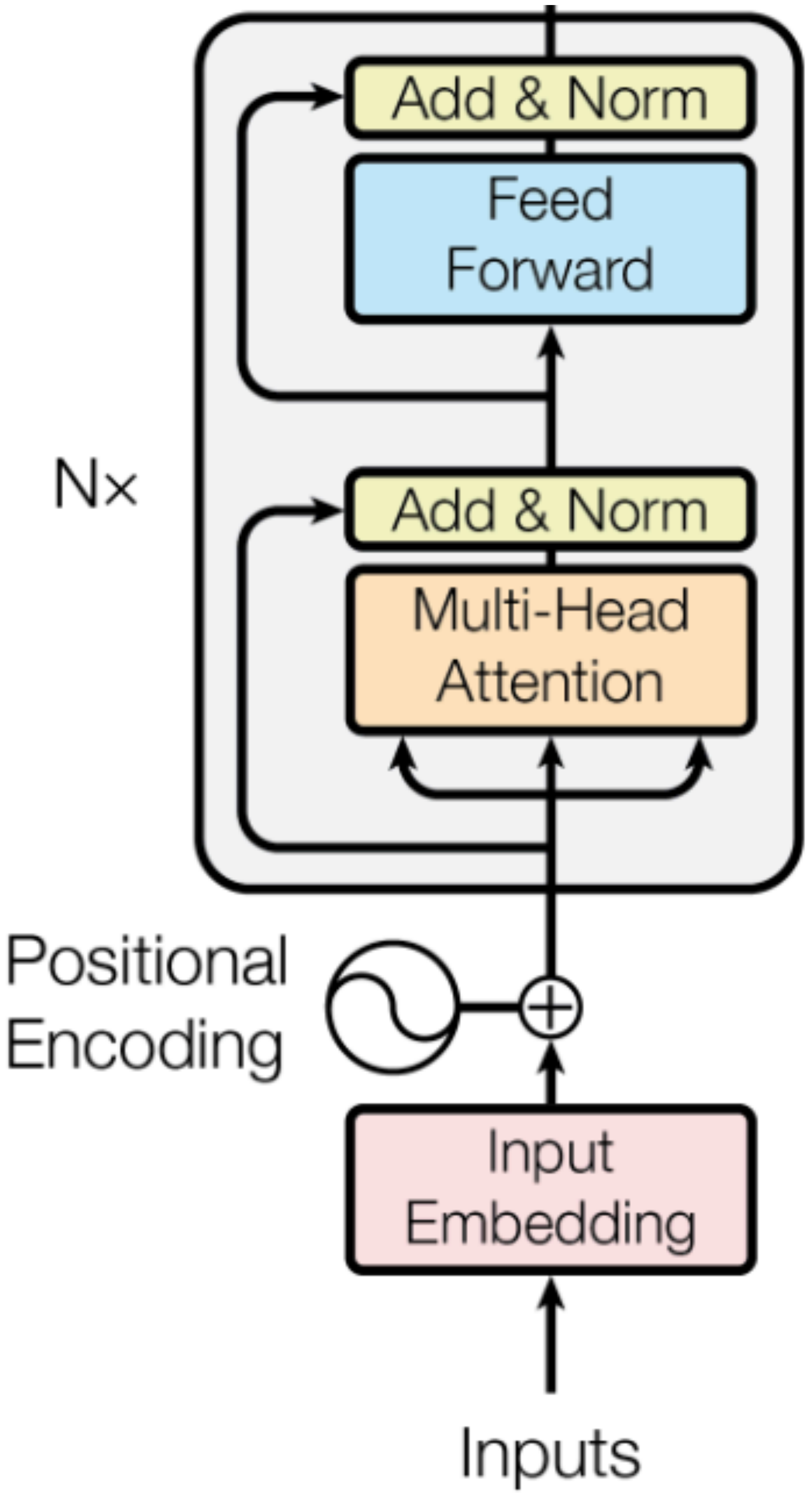
Outline

- Residual Connections
- **Normalization Layers**
  - Variants of Normalization Layers
  - Pre-Norm vs. Post-Norm
- **Positional Embeddings**
  - Absolute Positional Embeddings
  - Rotary Positional Embeddings

Pre-Norm vs. Post-Norm

- **Position of Normalizations**
  - Normalization layers block gradients in practice
  - The original Transformer architecture is hard to train w. more layers, even w. residual connections
  - Post-Norm -> Pre-Norm

Post-Normalization



$$Y_1 = \textbf{LayerNorm}\left(\textbf{SelfAttn}(X) \boxed{+ X}\right)$$

$$Y_2 = \textbf{LayerNorm}\left(\textbf{FFN}(Y_1) \boxed{+ Y_1}\right)$$

Detailed analysis

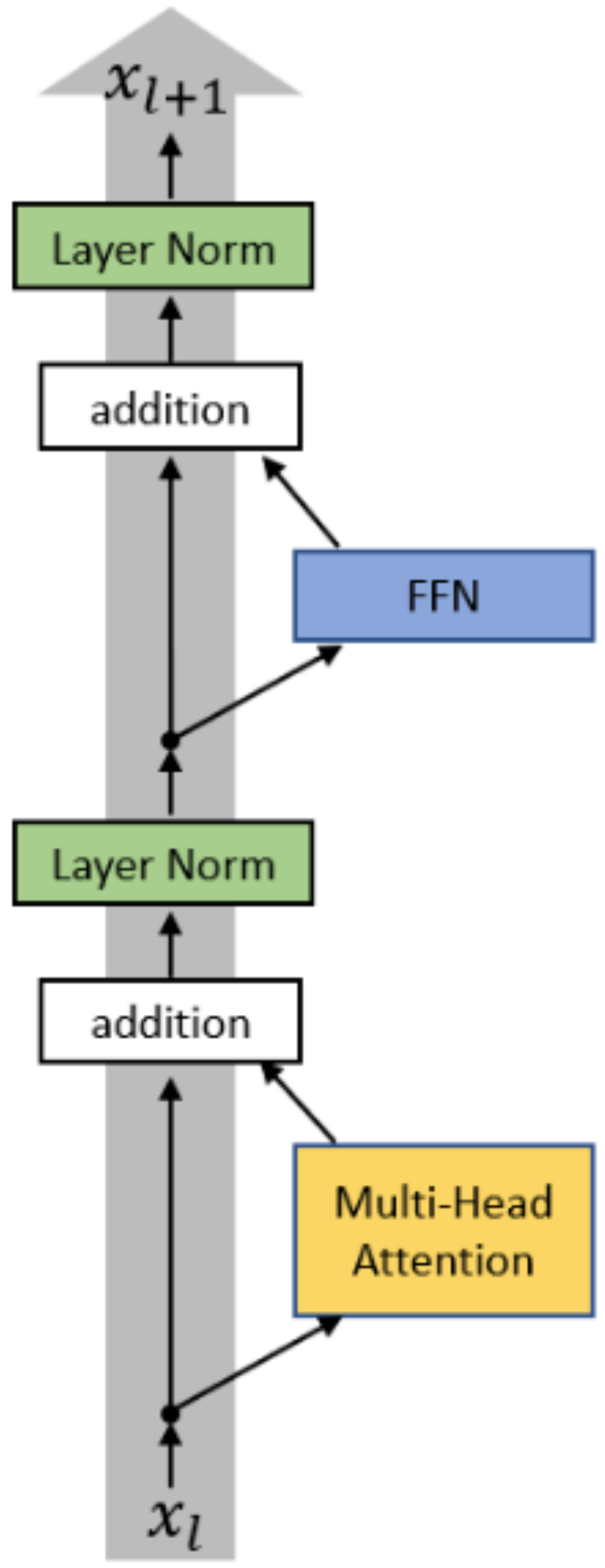
$$Y_1 = \textbf{LayerNorm}\left(f_1(X) + X\right)$$

$$Y_2 = \textbf{LayerNorm}\left(f_2(Y_1) + Y1\right) = \textbf{LayerNorm}\left(f_2(Y_1) + \textbf{LayerNorm}\left(f_1(X) + X\right)\right)$$

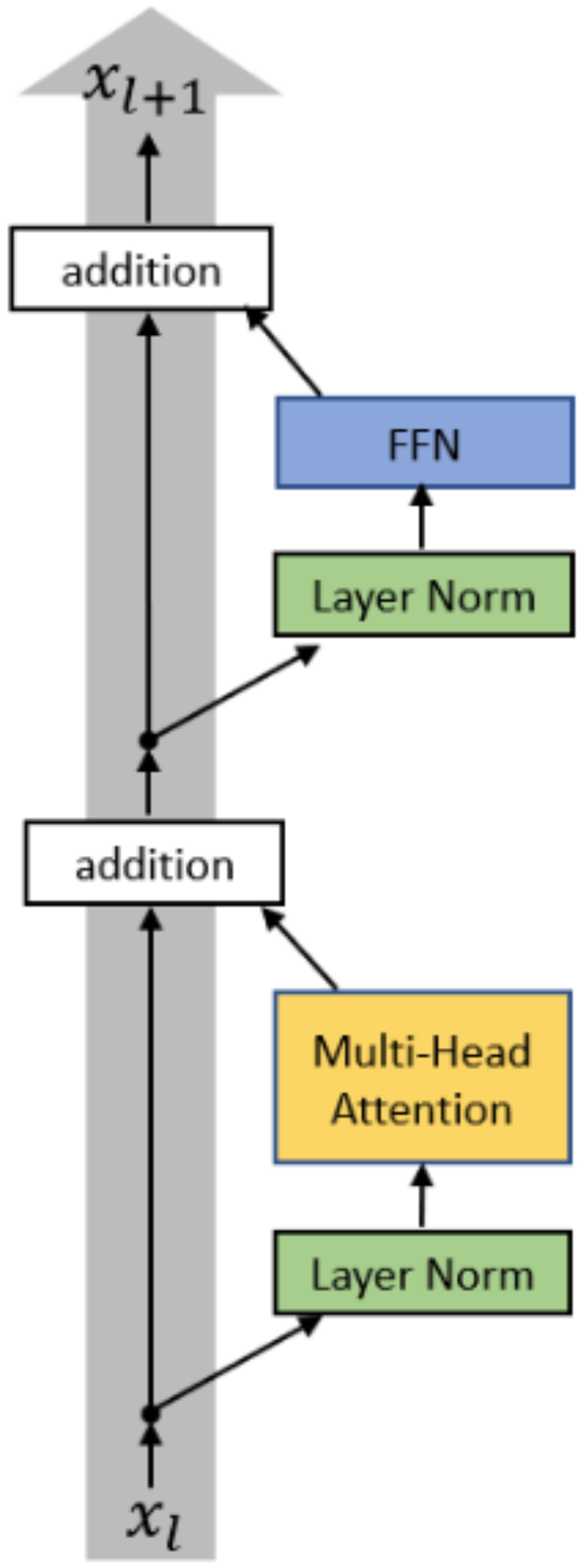
$$Y_l = \textbf{LayerNorm}\left(f_l(Y_{l-1}) + Y_{l-1}\right)$$

l layer-normalization blocks

Pre-Norm vs. Post-Norm

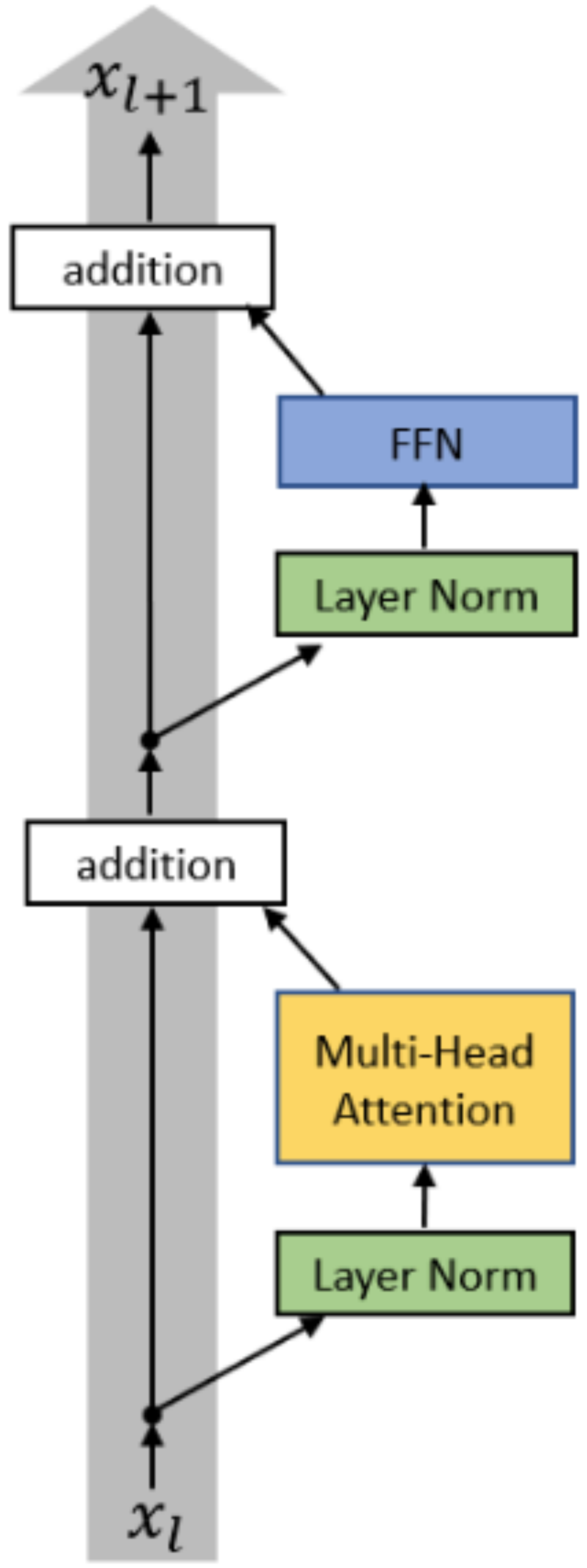


Post-Norm



Pre-Norm

Pre-Normalization



Pre-Norm

$$Y_1 = \textbf{SelfAttn}\left(\textbf{LayerNorm}(X)\right) \boxed{+ X}$$

$$Y_2 = \textbf{FFN}\left(\textbf{LayerNorm}(Y_1)\right) \boxed{+ Y_1}$$

Detailed analysis

$$Y_1 = f_1\left(\textbf{LayerNorm}(X)\right) + X$$

$$Y_2 = f_2\left(\textbf{LayerNorm}(Y_1)\right) + Y1 = f_2\left(\textbf{LayerNorm}(Y_1)\right) + f_1\left(\textbf{LayerNorm}(X)\right) + X$$

1 layer-normalization block for each layer output

Pre-Normalization

- **Pros:**
  - Keeping (almost) all good properties of post-norm
  - Similar performance w. Post-norm
  - Able to train very deep Transformer models
- **Cons:**
  - Numerically unstable (training spikes)

What is the range of Y_2 in post-norm and pre-norm?

Post-norm $$Y_2 = \textbf{LayerNorm}\left(f_2(Y_1) + Y1\right) = \textbf{LayerNorm}\left(f_2(Y_1) + \textbf{LayerNorm}\left(f_1(X) + X\right)\right)$$

Pre-norm $$Y_2 = f_2\left(\textbf{LayerNorm}(Y_1)\right) + Y1 = f_2\left(\textbf{LayerNorm}(Y_1)\right) + f_1\left(\textbf{LayerNorm}(X)\right) + X$$

Outline

- **Residual Connections**
- **Normalization Layers**
  - Variants of Normalization Layers
  - Pre-Norm vs. Post-Norm
- **Positional Embeddings**
  - Absolute Positional Embeddings
  - Rotary Positional Embeddings

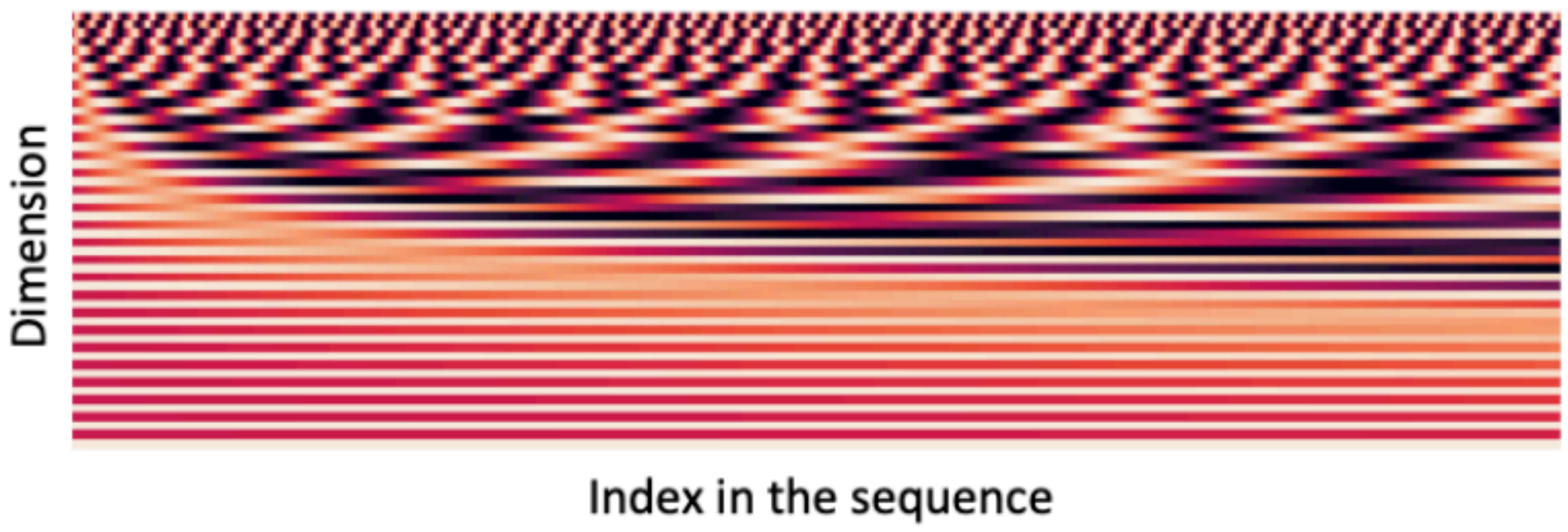
Positional Information

- **Unlike RNNs, self-attention does not build in order information**
  - Encode the order of the sentence into the input $x_1, \ldots, x_n$
- **Solution: add positional encoding to the input embeddings**

$$x_i \leftarrow x_i + p_i$$

- **Use sine and cosine functions of different frequencies (not learnable)**

$$p_i = \begin{pmatrix} \sin(i/10000^{2*1/d}) \\ \cos(i/10000^{2*1/d}) \\ \vdots \\ \sin(i/10000^{2*\frac{d}{2}/d}) \\ \cos(i/10000^{2*\frac{d}{2}/d}) \end{pmatrix}$$



Sinusoidal Positional Embedding

$$p_{t,j} = \sin\left(t \cdot \theta^{\frac{j}{2d}}\right) \qquad j \,\%\, 2 = 0$$

$$p_{t,j} = \cos\left(t \cdot \theta^{\frac{j-1}{2d}}\right) \qquad j \,\%\, 2 = 1$$

$$\theta = \frac{1}{10000}$$

Position $t = 0,1,\ldots,n$

dim $j = 0,\ldots,d$

	t=0	t=1	t=2	…	t=n
j=0	$\sin(0 \cdot \theta^0)$	$\sin(1 \cdot \theta^0)$	$\sin(2 \cdot \theta^0)$		$\sin(n \cdot \theta^0)$
j=1	$\cos(0 \cdot \theta^0)$	$\cos(1 \cdot \theta^0)$	$\cos(2 \cdot \theta^0)$		$\cos(n \cdot \theta^0)$
j=2	$\sin(0 \cdot \theta^{\frac{1}{2d}})$	$\sin(1 \cdot \theta^{\frac{1}{2d}})$	$\sin(2 \cdot \theta^{\frac{1}{2d}})$		$\sin(n \cdot \theta^{\frac{1}{2d}})$
j=3	$\cos(0 \cdot \theta^{\frac{1}{2d}})$	$\cos(1 \cdot \theta^{\frac{1}{2d}})$	$\cos(2 \cdot \theta^{\frac{1}{2d}})$		$\cos(n \cdot \theta^{\frac{1}{2d}})$

Sinusoidal Positional Embedding

$$p_{t,j} = \sin\left(t \cdot \theta^{\frac{j}{2d}}\right) \qquad j \,\%\, 2 = 0$$

$$p_{t,j} = \cos\left(t \cdot \theta^{\frac{j-1}{2d}}\right) \qquad j \,\%\, 2 = 1$$

$$\theta = \frac{1}{10000}$$

Position $t = 0,1,\ldots,n$

dim $j = 0,\ldots,d$

	t=0	t=1	t=2	…	t=n
j=0	$\sin(0 \cdot \theta^0)$	$\sin(1 \cdot \theta^0)$	$\sin(2 \cdot \theta^0)$		$\sin(n \cdot \theta^0)$
j=1	$\cos(0 \cdot \theta^0)$	$\cos(1 \cdot \theta^0)$	$\cos(2 \cdot \theta^0)$		$\cos(n \cdot \theta^0)$
j=2	$\sin(0 \cdot \theta^{\frac{1}{2d}})$	$\sin(1 \cdot \theta^{\frac{1}{2d}})$	$\sin(2 \cdot \theta^{\frac{1}{2d}})$		$\sin(n \cdot \theta^{\frac{1}{2d}})$
j=3	$\cos(0 \cdot \theta^{\frac{1}{2d}})$	$\cos(1 \cdot \theta^{\frac{1}{2d}})$	$\cos(2 \cdot \theta^{\frac{1}{2d}})$		$\cos(n \cdot \theta^{\frac{1}{2d}})$

Sinusoidal Positional Embedding

$$p_{t,j} = \sin\left(t \cdot \theta^{\frac{j}{2d}}\right) \quad j \,\%\, 2 = 0$$

$$p_{t,j} = \cos\left(t \cdot \theta^{\frac{j-1}{2d}}\right) \quad j \,\%\, 2 = 1$$

$$\theta = \frac{1}{10000}$$

Position $t = 0,1,\ldots,n$

dim $j = 0,\ldots,d$

	t=0	t=1	t=2	…	t=n
j=0	$\sin(0 \cdot \theta^0)$	$\sin(1 \cdot \theta^0)$	$\sin(2 \cdot \theta^0)$		$\sin(n \cdot \theta^0)$
j=1	$\cos(0 \cdot \theta^0)$	$\cos(1 \cdot \theta^0)$	$\cos(2 \cdot \theta^0)$		$\cos(n \cdot \theta^0)$
j=2	$\sin(0 \cdot \theta^{\frac{1}{2d}})$	$\sin(1 \cdot \theta^{\frac{1}{2d}})$	$\sin(2 \cdot \theta^{\frac{1}{2d}})$		$\sin(n \cdot \theta^{\frac{1}{2d}})$
j=3	$\cos(0 \cdot \theta^{\frac{1}{2d}})$	$\cos(1 \cdot \theta^{\frac{1}{2d}})$	$\cos(2 \cdot \theta^{\frac{1}{2d}})$		$\cos(n \cdot \theta^{\frac{1}{2d}})$

Sinusoidal Positional Embedding

$$p_{t,j} = \sin\left(t \cdot \theta^{\frac{j}{2d}}\right) \quad j \,\%\, 2 = 0$$

$$p_{t,j} = \cos\left(t \cdot \theta^{\frac{j-1}{2d}}\right) \quad j \,\%\, 2 = 1$$

$$\theta = \frac{1}{10000}$$

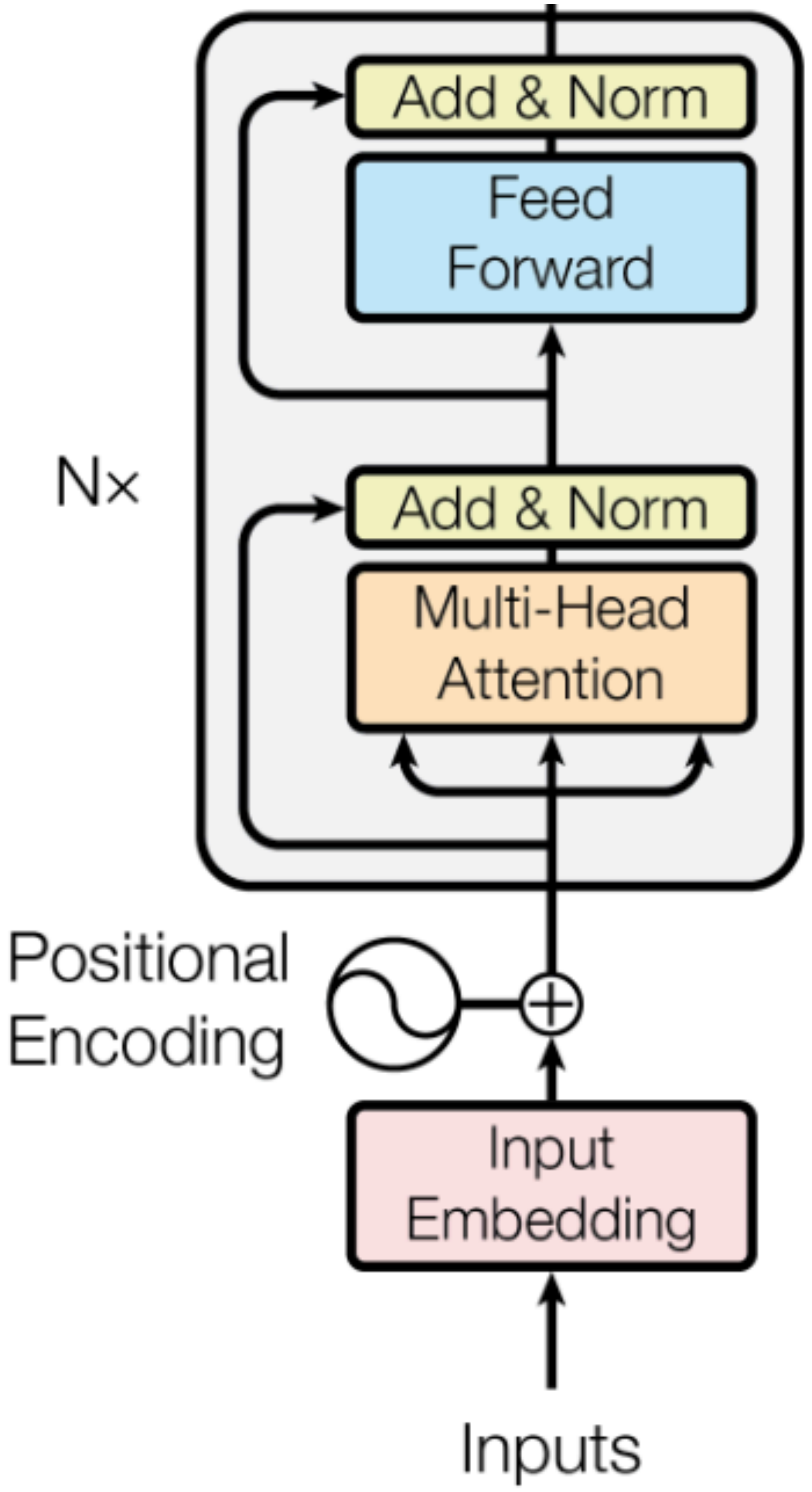
Position $t = 0,1,\ldots,n$

dim $j = 0,\ldots,d$

	t=0	t=1	t=2	…	t=n
j=0	$\sin(0 \cdot \theta^{0})$	$\sin(1 \cdot \theta^{0})$	$\sin(2 \cdot \theta^{0})$		$\sin(n \cdot \theta^{0})$
j=1	$\cos(0 \cdot \theta^{0})$	$\cos(1 \cdot \theta^{0})$	$\cos(2 \cdot \theta^{0})$		$\cos(n \cdot \theta^{0})$
j=2	$\sin(0 \cdot \theta^{\frac{1}{2d}})$	$\sin(1 \cdot \theta^{\frac{1}{2d}})$	$\sin(2 \cdot \theta^{\frac{1}{2d}})$		$\sin(n \cdot \theta^{\frac{1}{2d}})$
j=3	$\cos(0 \cdot \theta^{\frac{1}{2d}})$	$\cos(1 \cdot \theta^{\frac{1}{2d}})$	$\cos(2 \cdot \theta^{\frac{1}{2d}})$		$\cos(n \cdot \theta^{\frac{1}{2d}})$

Sinusoidal Positional Embedding

- **Sinusoidal positional embedding depends on absolute positional information**
- **Only added to the input word embeddings**
- **Ability for length extension**
  - Training on up to k words
  - Test on > k words



Rotary Positional Embedding

- **Absolute positional index is not meaningful**
  - I don't know
  - In fact, I don't know

			I	**don't**	**know**
			p_1	p_2	p_3
In	**fact**	**,**	**I**	**don't**	**know**
p_1	p_2	p_3	p_4	p_5	p_6

Rotary Positional Embedding

- **Absolute positional index is not meaningful**
  - I don't know
  - In fact, I don't know
- **How about relative positional embedding?**
  - Positional information should be invariant w.r.t relative distance of two positions
  - Rotary Position Embedding
    - Used in Llama models

Rotary Positional Embedding

Rotary Operation

$$X \in \mathbb{R}^d \qquad \theta_j = \left(\frac{1}{10000}\right)^{\frac{j}{2d}}$$

$$\textbf{Rotary}(X, m) = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \\ x_{d-1} \\ x_d \end{pmatrix} \otimes \begin{pmatrix} \cos m\theta_1 \\ \cos m\theta_1 \\ \cos m\theta_2 \\ \cos m\theta_2 \\ \vdots \\ \cos m\theta_{d/2} \\ \cos m\theta_{d/2} \end{pmatrix} + \begin{pmatrix} -x_2 \\ x_1 \\ -x_4 \\ x_3 \\ \vdots \\ -x_d \\ x_{d-1} \end{pmatrix} \otimes \begin{pmatrix} \sin m\theta_1 \\ \sin m\theta_1 \\ \sin m\theta_2 \\ \sin m\theta_2 \\ \vdots \\ \sin m\theta_{d/2} \\ \sin m\theta_{d/2} \end{pmatrix}$$

Rotary Positional Embedding

Attention

$$\boxed{\mathrm{attn}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d}})V}$$

Applying rotary operation to each query and key vector

$$q'_i = \textbf{Rotary}(q_i, i)$$

$$k'_j = \textbf{Rotary}(k_j, j)$$

Replacing Q, K with Q', K' to compute attention

Rotary Positional Embedding

- **A simple case:** $d = 2$

$$\textbf{Rotary}(X, m) = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \\ x_{d-1} \\ x_d \end{pmatrix} \otimes \begin{pmatrix} \cos m\theta_1 \\ \cos m\theta_1 \\ \cos m\theta_2 \\ \cos m\theta_2 \\ \vdots \\ \cos m\theta_{d/2} \\ \cos m\theta_{d/2} \end{pmatrix} + \begin{pmatrix} -x_2 \\ x_1 \\ -x_4 \\ x_3 \\ \vdots \\ -x_d \\ x_{d-1} \end{pmatrix} \otimes \begin{pmatrix} \sin m\theta_1 \\ \sin m\theta_1 \\ \sin m\theta_2 \\ \sin m\theta_2 \\ \vdots \\ \sin m\theta_{d/2} \\ \sin m\theta_{d/2} \end{pmatrix}$$

$$q_i' = \mathrm{Rotary}(q_i, i) = \begin{pmatrix} q_{i,1}\cos(i\theta) - q_{i,2}\sin(i\theta) \\ q_{i,1}\sin(i\theta) + q_{i,2}\cos(i\theta) \end{pmatrix}$$

$$k_j' = \mathrm{Rotary}(k_j, j) = \begin{pmatrix} k_{j,1}\cos(j\theta) - k_{j,2}\sin(j\theta) \\ k_{j,1}\sin(j\theta) + k_{j,2}\cos(j\theta) \end{pmatrix}$$

Rotary Positional Embedding

- **A simple case:** $d = 2$

$$q'_i = \mathrm{Rotary}(q_i, i) = \begin{pmatrix} q_{i,1}\cos(i\theta) - q_{i,2}\sin(i\theta) \\ q_{i,1}\sin(i\theta) + q_{i,2}\cos(i\theta) \end{pmatrix}$$

$$k'_j = \mathrm{Rotary}(k_j, j) = \begin{pmatrix} k_{j,1}\cos(j\theta) - k_{j,2}\sin(j\theta) \\ k_{j,1}\sin(j\theta) + k_{j,2}\cos(j\theta) \end{pmatrix}$$

$$< q'_i, k'_j > = (q_{i,1}\cos(i\theta) - q_{i,2}\sin(i\theta))(k_{j,1}\cos(j\theta) - k_{j,2}\sin(j\theta))$$

$$+(q_{i,1}\sin(i\theta) + q_{i,2}\cos(i\theta))(k_{j,1}\sin(j\theta) + k_{j,2}\cos(j\theta))$$

$$= \left(q_{i,1}k_{j,1} + q_{i,2}k_{j,2}\right)\cos\boxed{\left((i-j)\theta\right)} + \left(q_{i,1}k_{j,2} - q_{i,2}k_{j,1}\right)\sin\boxed{\left((i-j)\theta\right)}$$

$< q'_i, k'_j >$ only depends on $(i-j)\theta$

Q&A