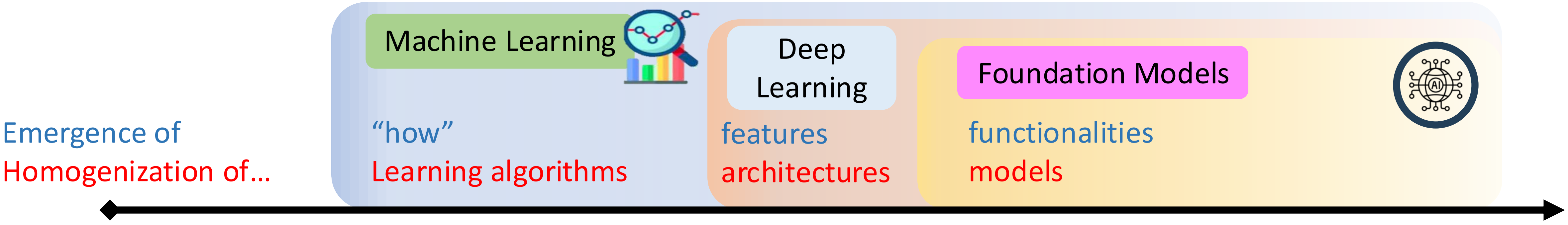
Foundational Models

Chetan Arora

Department of Computer Science and Engineering
Joint Faculty: School of Artificial Intelligence
Indian Institute of Technology Delhi

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Foundational Models

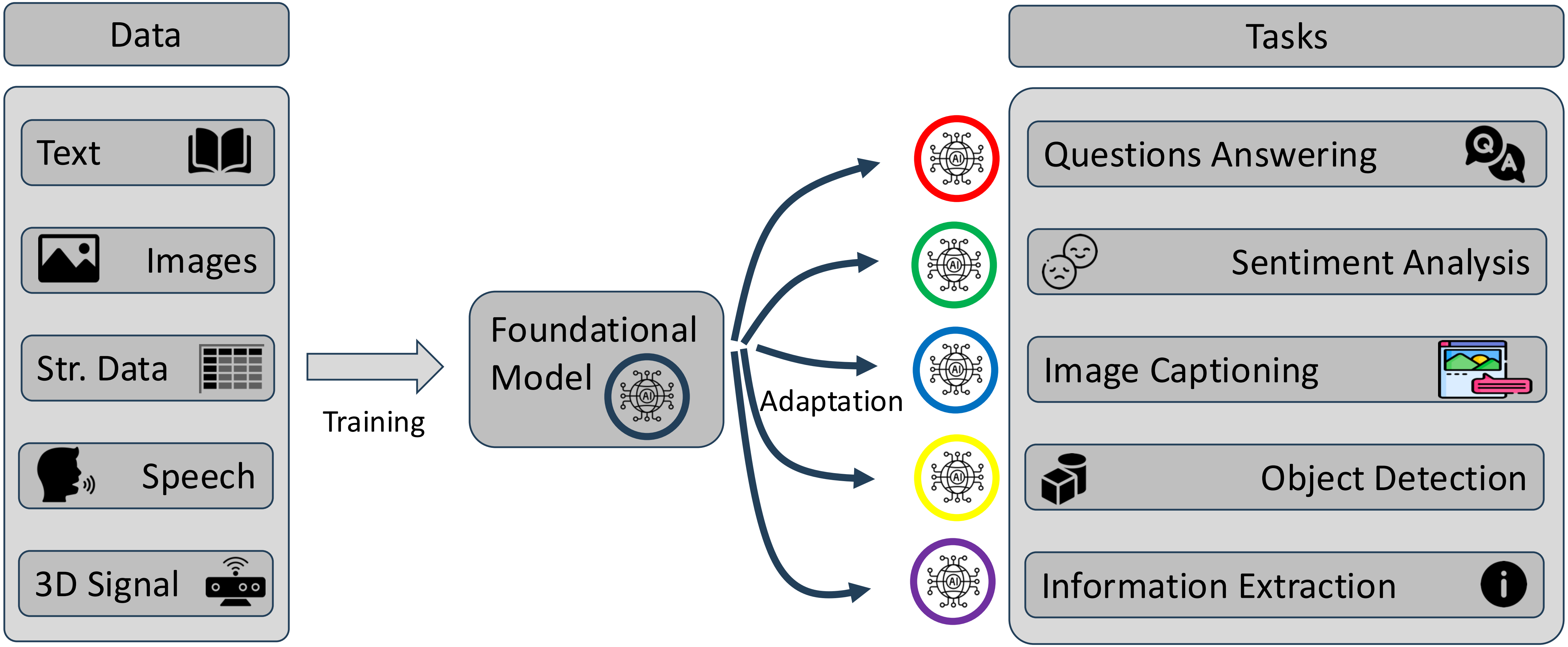


- Coined in 2021, it references the recent paradigm shift to develop a single model that can implicitly support many downstream tasks.

Bommasani et al. On the Opportunities and Risks of Foundation Models. arXiv 2021

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

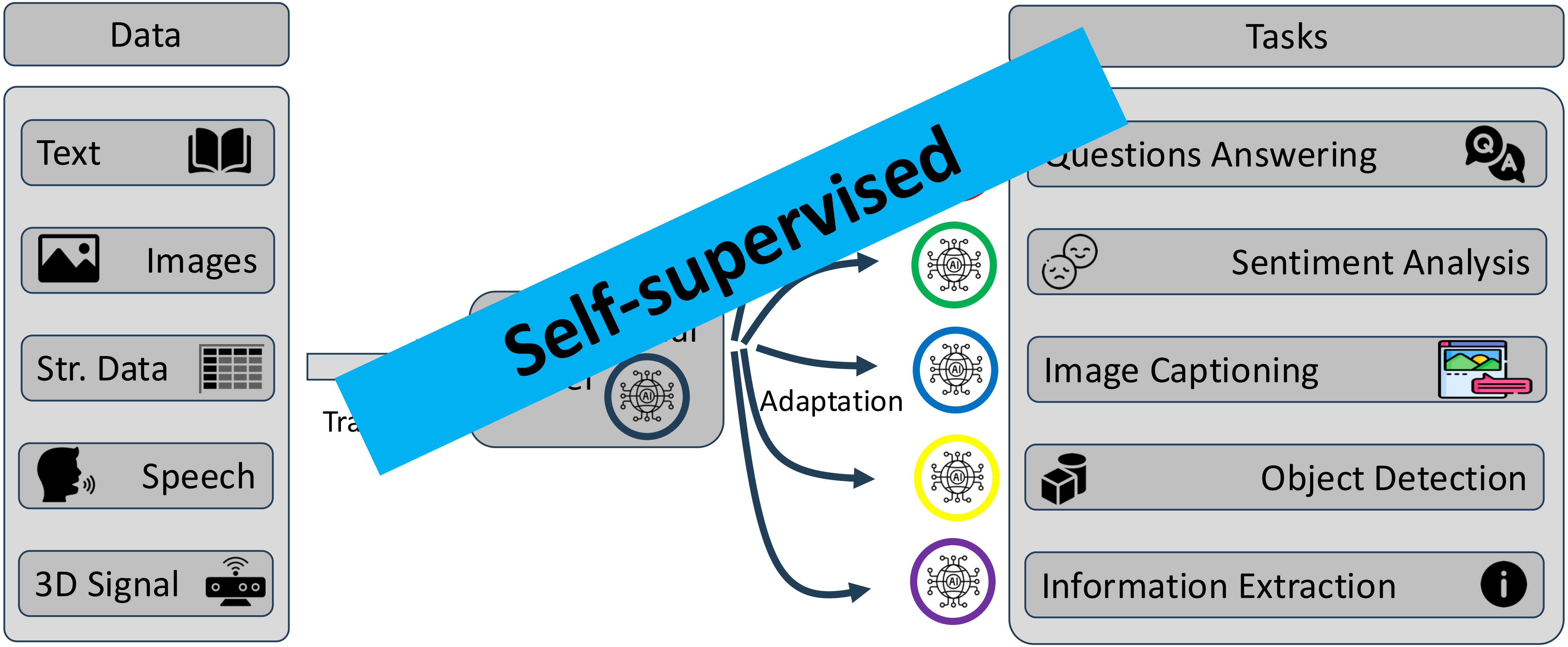
Foundational Models



Bommasani et al. On the Opportunities and Risks of Foundation Models. arXiv 2021

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Foundational Models



Bommasani et al. On the Opportunities and Risks of Foundation Models. arXiv 2021

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Beyond Pretraining and Fine-Tuning Paradigm

Pre-Training

Large unlabelled datasets
(e.g. Wikipedia, BookCorpus)

Self-supervised training
(hours to days)

→ Pre-Trained Weights →

Fine-Tuning

Smaller labelled datasets
(SQuAD, MNLI Similarity)

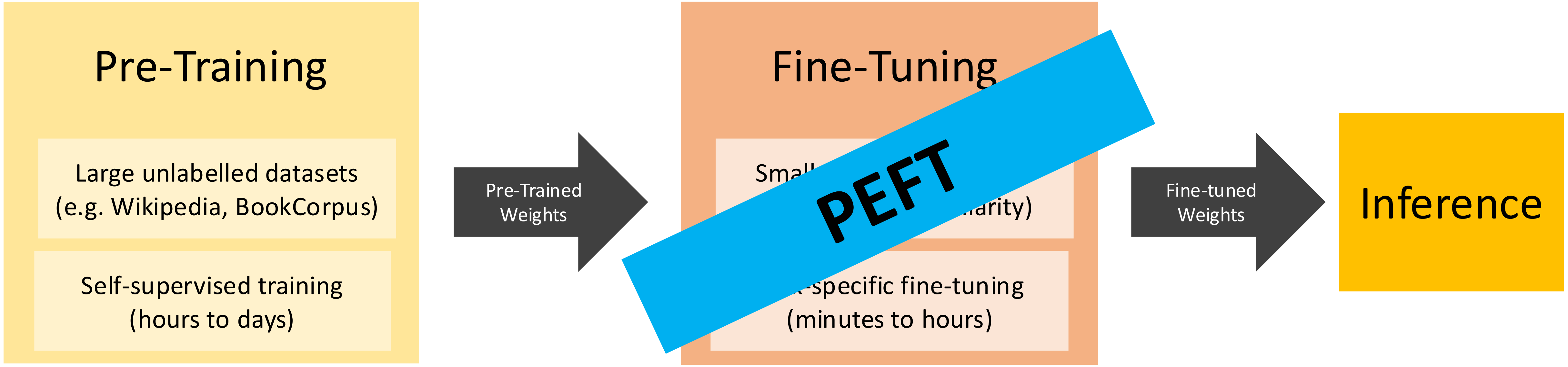
Task-specific fine-tuning
(minutes to hours)

→ Fine-tuned Weights →

Inference

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Beyond Pretraining and Fine-Tuning Paradigm

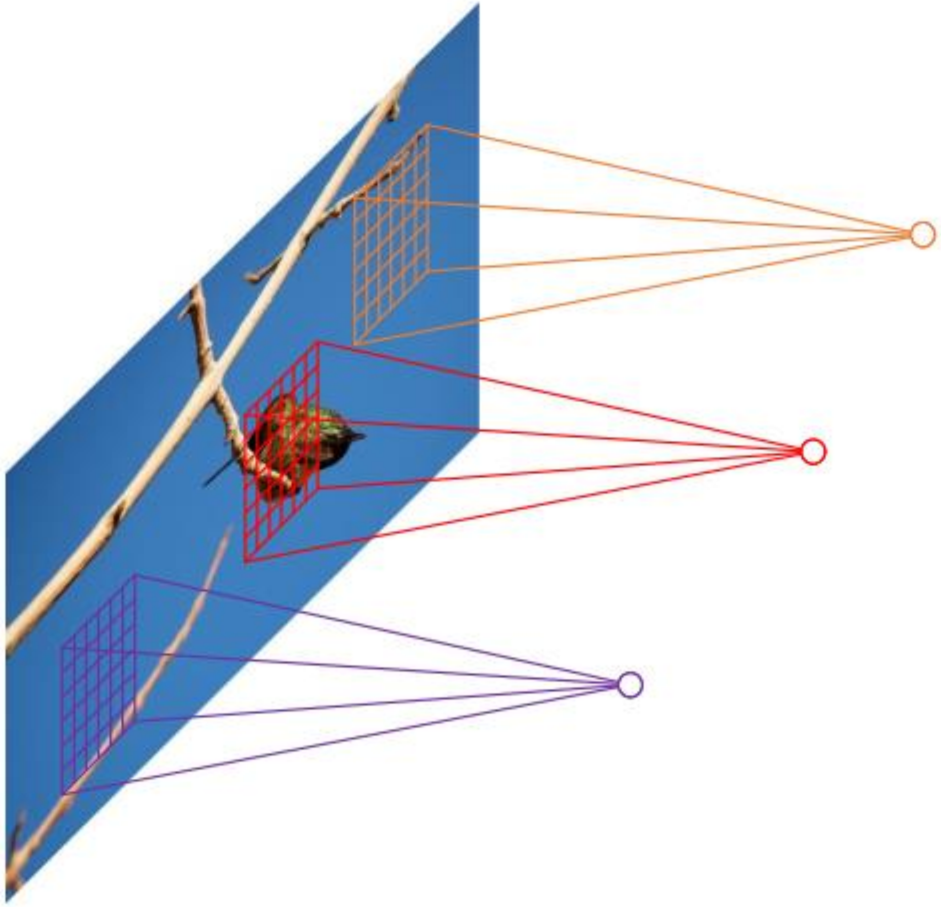


Neural Architecture for Foundational Models

Transformers

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

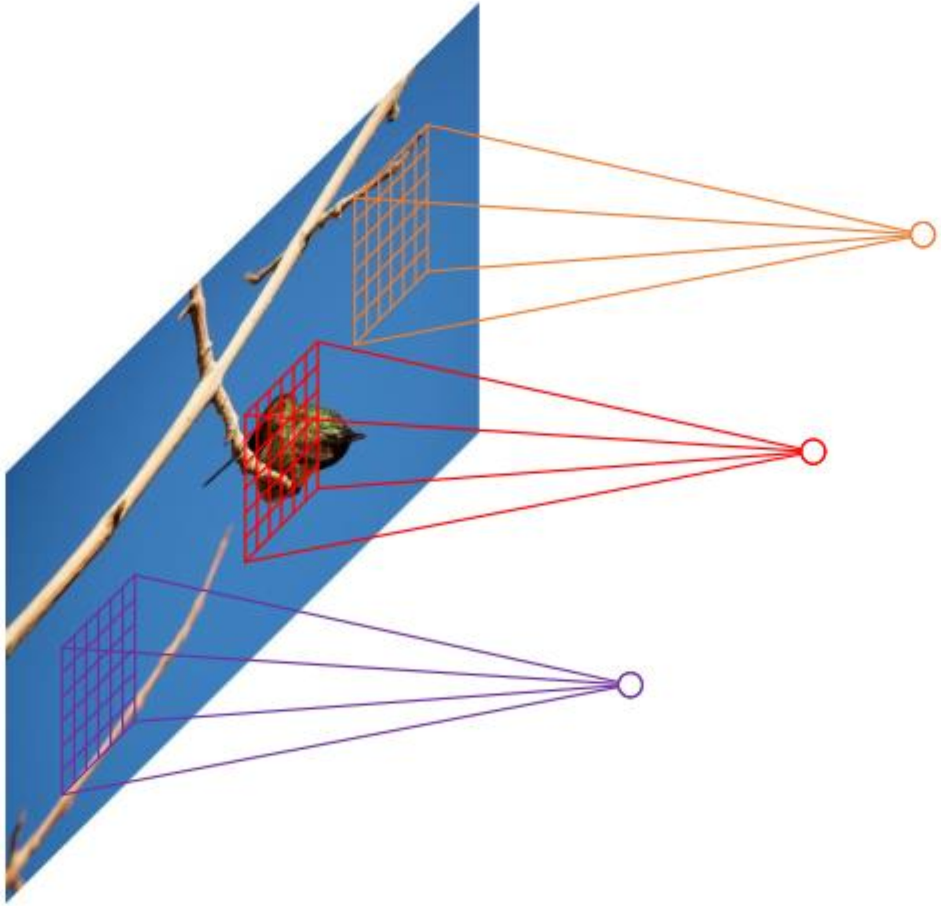
CNNs as Pattern Detector



- Convolutional layers are locally connected
- a filter/kernel/window slides on the image or the previous map
- the position of the filter explicitly provides information for localizing
- local spatial information w.r.t. the window is encoded in the channels

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

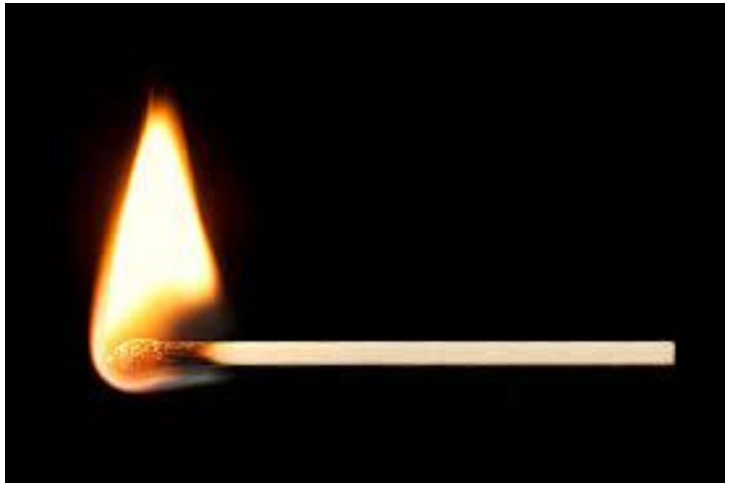
CNNs for Translation Invariance Features



- Convolutional layers share weights spatially leading to translation-invariant features
- Translation-invariance: a translated region will produce the same response at the correspondingly translated position
- A local pattern's convolutional response can be re-used by different candidate regions

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Limitations of CNN's Inductive Bias

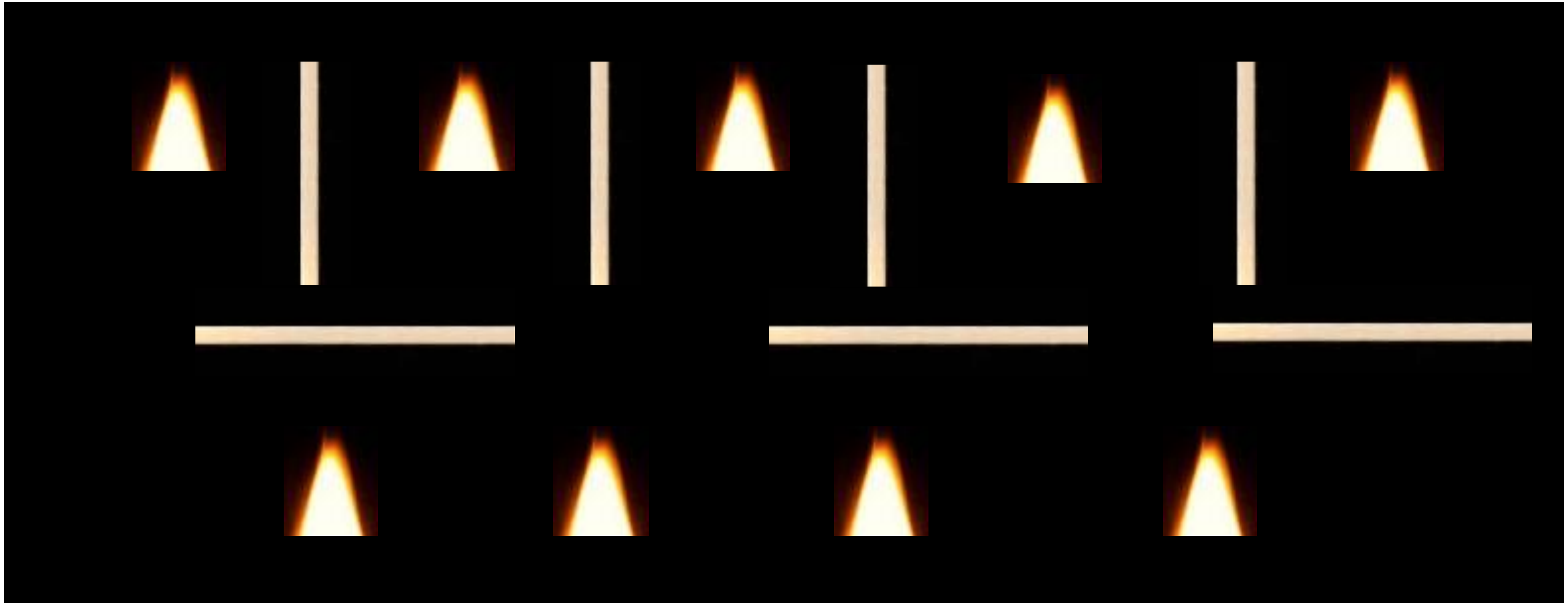


Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Limitations of CNN’s Inductive Bias

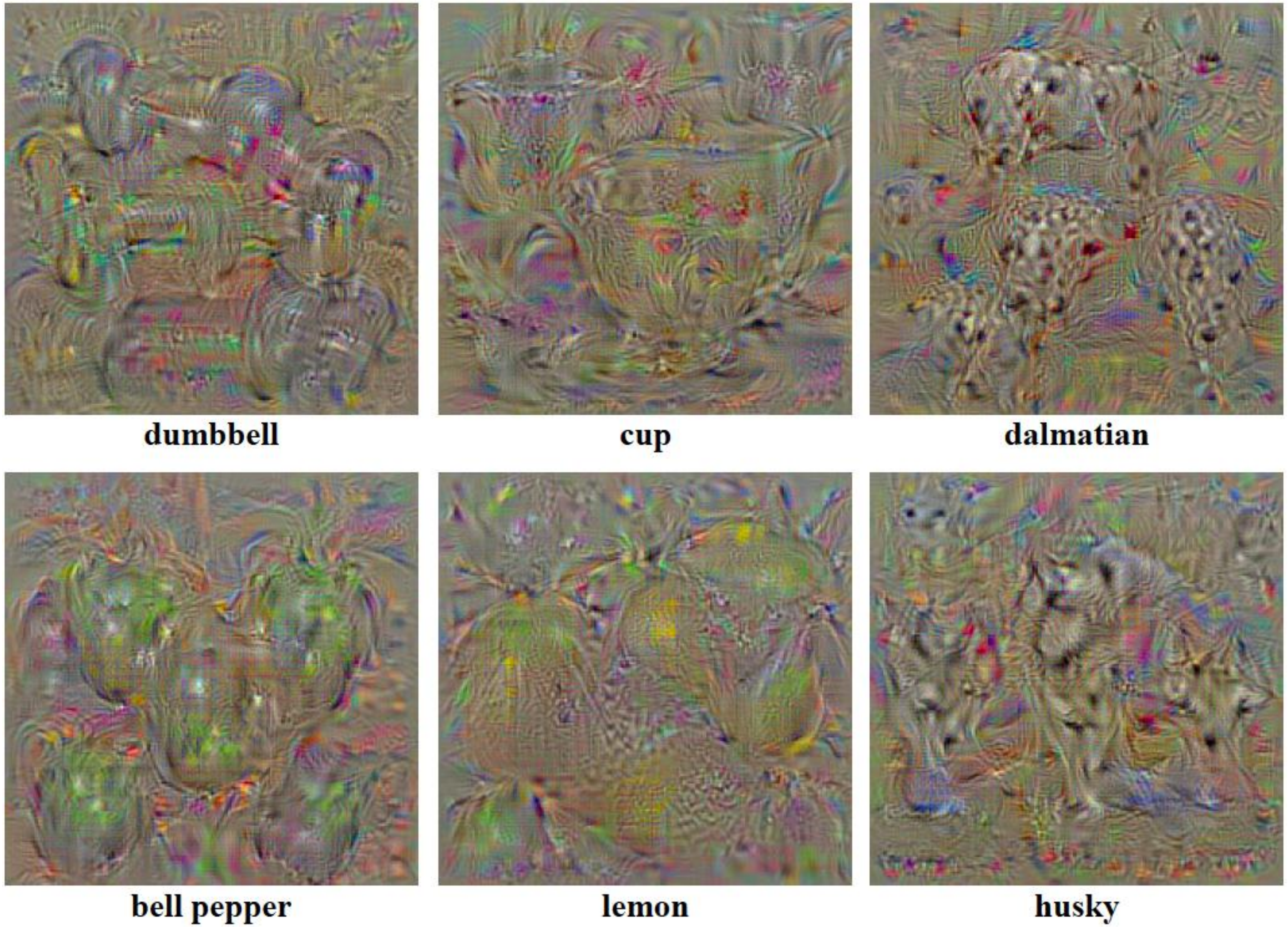
Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Limitations of CNN's Inductive Bias



Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

What is a Class to a CNN



Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman
Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Alternate Paradigm: Attention

What is attention

- In psychology attention is defined as the cognitive ability of humans to focus on the relevant things while processing a lot of information.

Attention mechanism in neural networks tries to do the same, by focusing on the few important things/regions among many.

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Attention Layer

Inputs:

Query vector: Q (Shape: $\mathrm{N_Q} \times D_Q$)

Input vectors: X (Shape: $N_X \times D_Q$)

Key matrix: $W_K (Shape: D_X \times D_Q)$

Value matrix: $W_V (Shape: D_X \times D_V)$

Computation:

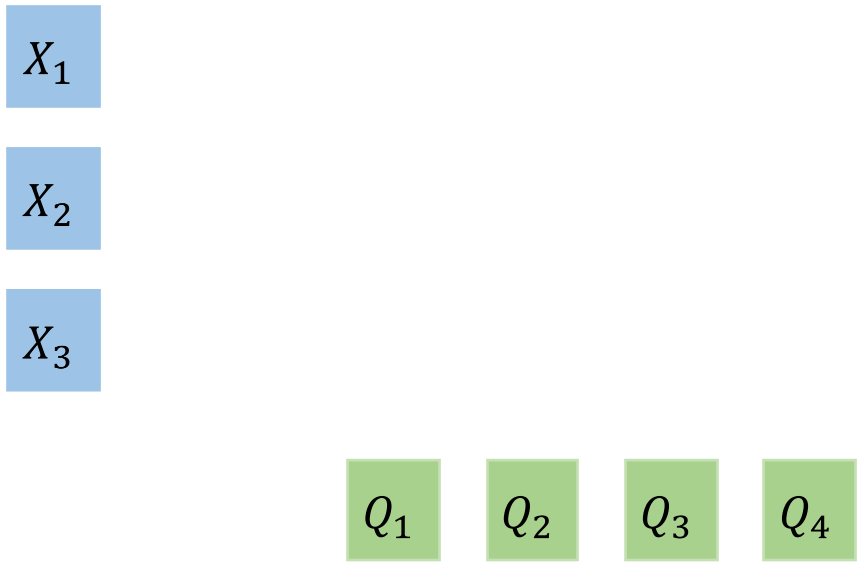
Key vectors: $K = XW_K$ $(Shape: N_X \times D_Q)$

Value Vectors: $V = XW_V$ $(Shape: N_X \times D_V)$

Similarities: $E = \frac{QK^T}{\sqrt{D_Q}}$ $(Shape: N_Q \times N_X)$, $E_{i,j} = (Q_i \cdot K_j)/\sqrt{D_q}$

Attention weights: $A = \mathrm{softmax}(E, \mathrm{dim} = 1)$ (Shape: $N_Q \times N_X$)

Output vectors: $Y = AV$ $(Shape: N_Q \times D_X)$ $Y_i = \Sigma_j A_{i,j} V_j$



Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Attention Layer

Inputs:

Query vector: Q (Shape: $\mathrm{N_Q} \times D_Q$)

Input vectors: X (Shape: $N_X \times D_Q$)

Key matrix: $W_K (Shape: D_X \times D_Q)$

Value matrix: $W_V (Shape:\ D_X \times D_V)$

Computation:

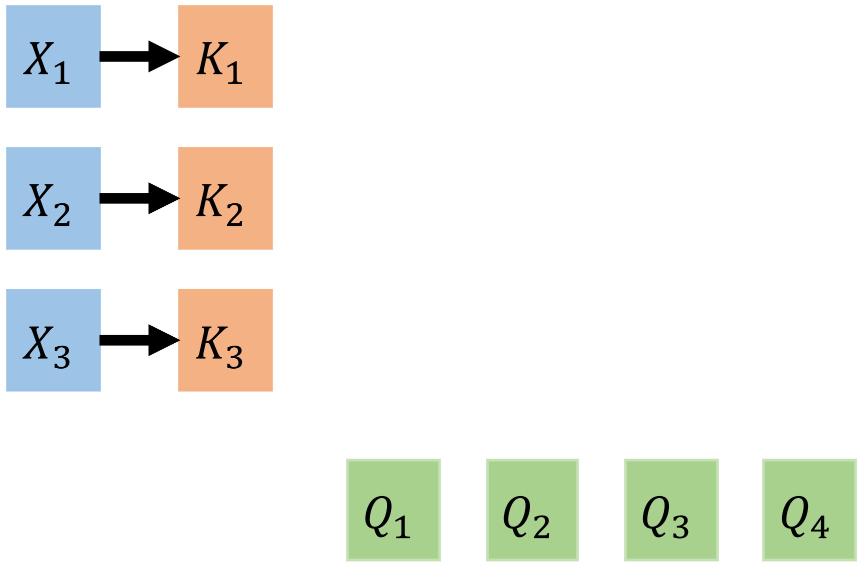
Key vectors: $K = XW_K\ (Shape:\ N_X \times D_Q)$

Value Vectors: $V = XW_V\ (Shape:\ N_X \times D_V)$

Similarities: $E = \frac{QK^T}{\sqrt{D_Q}}\ (Shape:\ N_Q \times N_X), E_{i,j} = (Q_i \cdot K_j)/\sqrt{D_q}$

Attention weights: $A = \mathrm{softmax}(E, \mathrm{dim} = 1)$ (Shape: $N_Q \times N_X$)

Output vectors: $Y = AV\ (Shape:\ N_Q \times D_X) Y_i = \Sigma_j A_{i,j} V_j$



Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Attention Layer

Inputs:

Query vector: Q (Shape: $N_Q \times D_Q$)

Input vectors: X (Shape: $N_X \times D_Q$)

Key matrix: $W_K (Shape: D_X \times D_Q)$

Value matrix: $W_V (Shape: D_X \times D_V)$

Computation:

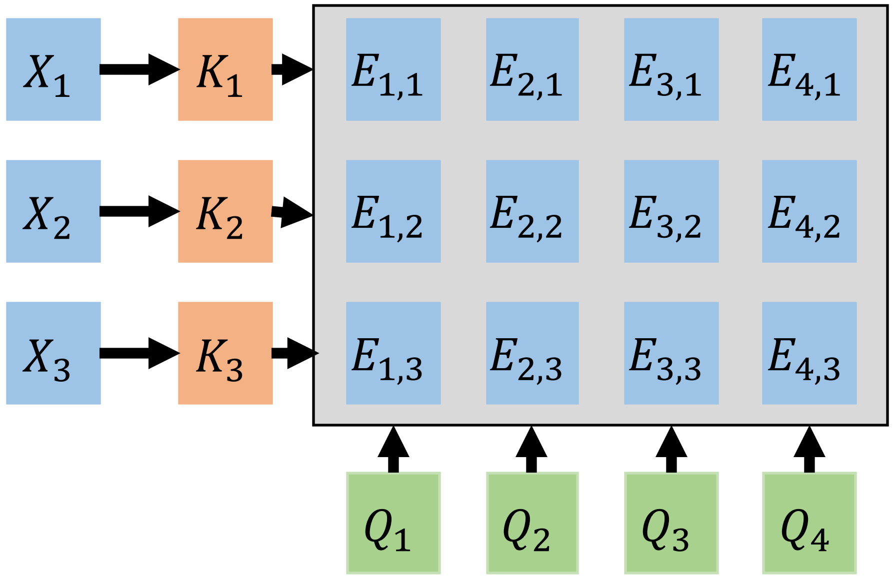
Key vectors: $K = XW_K$ $(Shape: N_X \times D_Q)$

Value Vectors: $V = XW_V$ $(Shape: N_X \times D_V)$

Similarities: $E = \frac{QK^T}{\sqrt{D_Q}}$ $(Shape: N_Q \times N_X), E_{i,j} = (Q_i \cdot K_j)/\sqrt{D_q}$

Attention weights: $A = \text{softmax}(E, \text{dim} = 1)$ (Shape: $N_Q \times N_X$)

Output vectors: $Y = AV$ $(Shape: N_Q \times D_X) Y_i = \Sigma_j A_{i,j} V_j$



Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Attention Layer

Inputs:

Query vector: Q (Shape: $N_Q \times D_Q$)

Input vectors: X (Shape: $N_X \times D_Q$)

Key matrix: $W_K (Shape: D_X \times D_Q)$

Value matrix: $W_V (Shape: D_X \times D_V)$

Computation:

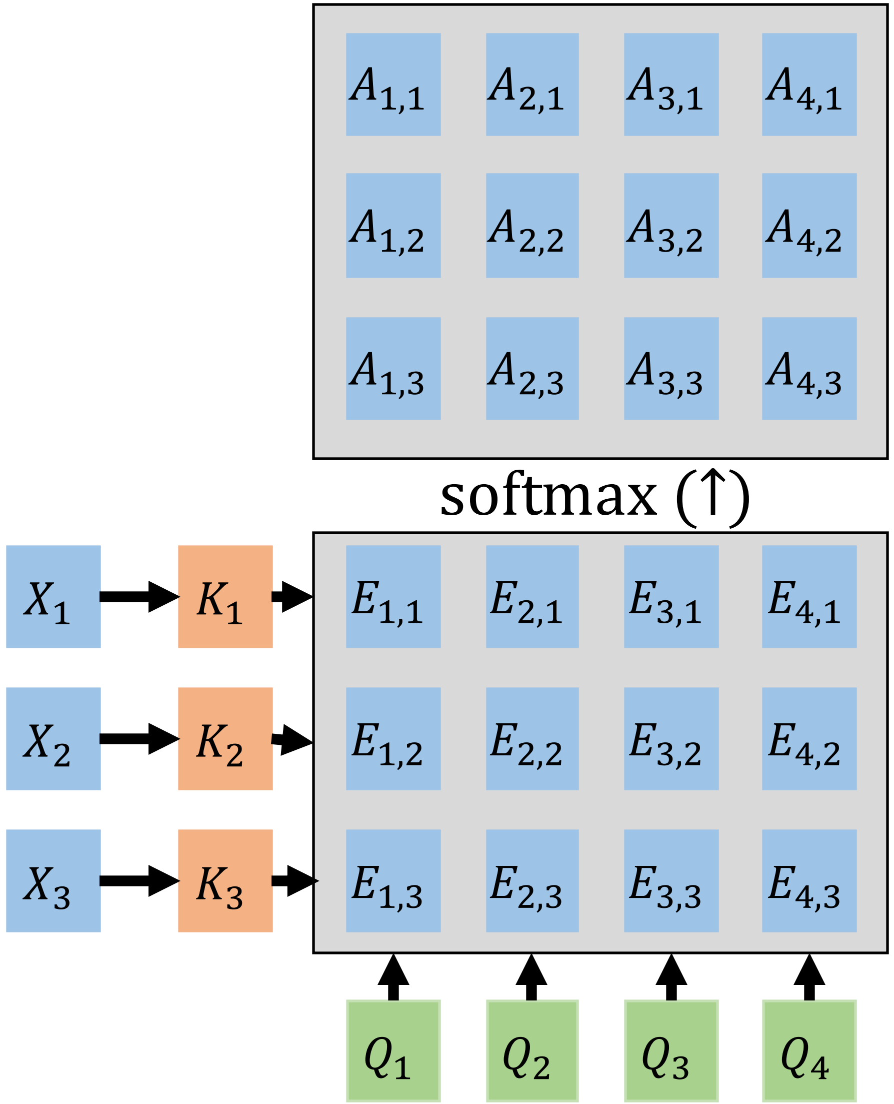
Key vectors: $K = XW_K$ $(Shape: N_X \times D_Q)$

Value Vectors: $V = XW_V$ $(Shape: N_X \times D_V)$

Similarities: $E = \frac{QK^T}{\sqrt{D_Q}}$ $\left(Shape: N_Q \times N_X\right), E_{i,j} = (Q_i \cdot K_j)/\sqrt{D_q}$

Attention weights: $A = \text{softmax}(E, \text{dim} = 1)$ (Shape: $N_Q \times N_X$)

Output vectors: $Y = AV$ $\left(Shape: N_Q \times D_X\right) Y_i = \Sigma_j A_{i,j} V_j$



Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Attention Layer

Inputs:

Query vector: Q (Shape: $N_Q \times D_Q$)

Input vectors: X (Shape: $N_X \times D_Q$)

Key matrix: $W_K (Shape: D_X \times D_Q)$

Value matrix: $W_V (Shape: D_X \times D_V)$

Computation:

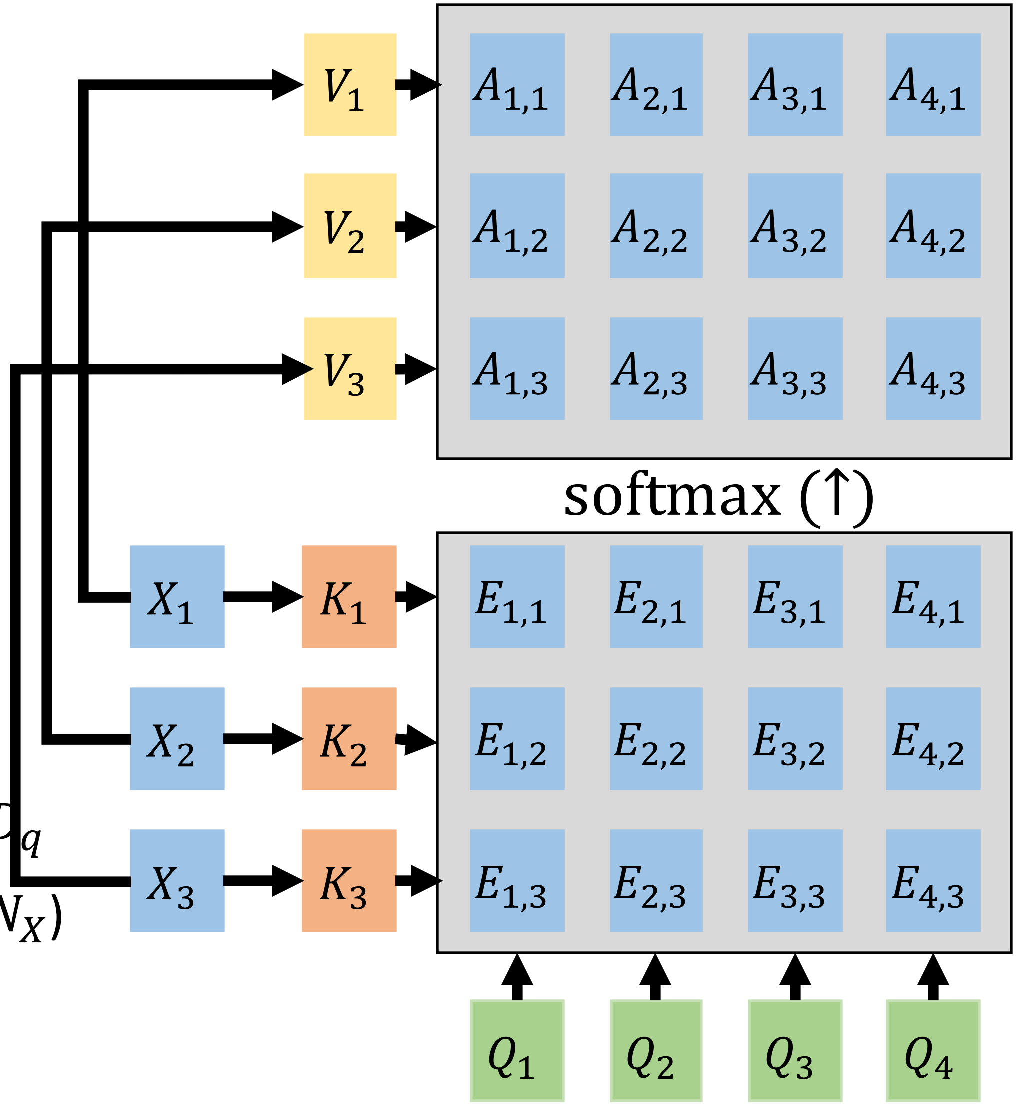
Key vectors: $K = XW_K \ (Shape: N_X \times D_Q)$

Value Vectors: $V = XW_V \ (Shape: N_X \times D_V)$

Similarities: $E = \frac{QK^T}{\sqrt{D_Q}} \ (Shape: N_Q \times N_X), E_{i,j} = (Q_i \cdot K_j)/\sqrt{D_q}$

Attention weights: $A = \text{softmax}(E, \text{dim} = 1)$ (Shape: $N_Q \times N_X$)

Output vectors: $Y = AV \ (Shape: N_Q \times D_X) Y_i = \Sigma_j A_{i,j} V_j$



Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Attention Layer

Inputs:

Query vector: Q (Shape: $N_Q \times D_Q$)

Input vectors: X (Shape: $N_X \times D_Q$)

Key matrix: $W_K (Shape: D_X \times D_Q)$

Value matrix: $W_V (Shape: D_X \times D_V)$

Computation:

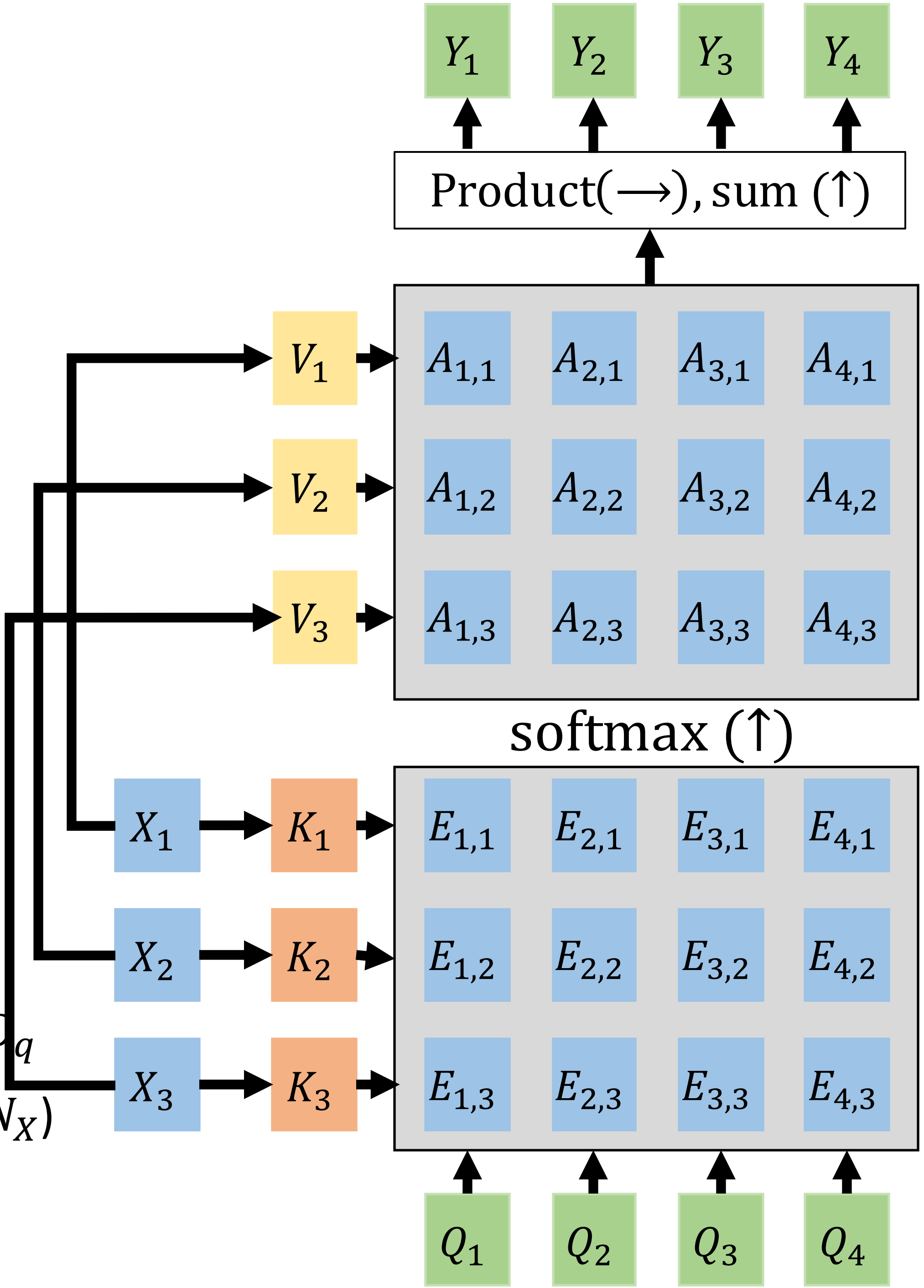
Key vectors: $K = XW_K$ $(Shape: N_X \times D_Q)$

Value Vectors: $V = XW_V$ $(Shape: N_X \times D_V)$

Similarities: $E = \frac{QK^T}{\sqrt{D_Q}}$ $(Shape: N_Q \times N_X), E_{i,j} = (Q_i \cdot K_j)/\sqrt{D_q}$

Attention weights: $A = \text{softmax}(E, \text{dim} = 1)$ (Shape: $N_Q \times N_X$)

Output vectors: $Y = AV$ $(Shape: N_Q \times D_X) Y_i = \Sigma_j A_{i,j} V_j$



Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Self-Attention Layer

One query per input vector

Inputs:

Query vector: Q (Shape: $N_Q \times D_Q$)

Input vectors: X (Shape: $N_X \times D_Q$)

Key matrix: $W_K (Shape: D_X \times D_Q)$

Value matrix: $W_V (Shape: D_X \times D_V)$

Computation:

Key vectors: $K = XW_K$ $(Shape: N_X \times D_Q)$

Value Vectors: $V = XW_V$ $(Shape: N_X \times D_V)$

Similarities: $E = \frac{QK^T}{\sqrt{D_Q}}$ $(Shape: N_Q \times N_X), E_{i,j} = (Q_i \cdot K_j)/\sqrt{D_q}$

Attention weights: $A = \text{softmax}(E, \text{dim} = 1)$ (Shape: $N_Q \times N_X$)

Output vectors: $Y = AV$ $\left(Shape: N_Q \times D_X\right) Y_i = \Sigma_j A_{i,j} V_j$

X_1 X_2 X_3

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Self-Attention Layer

One query per input vector

Inputs:

Input vectors: X (Shape: $N_X \times D_Q$)

Key matrix: $W_K (Shape: D_X \times D_Q)$

Value matrix: $W_V (Shape: D_X \times D_V)$

Query matrix: $\mathbf{W}_\mathrm{Q}$ (Shape: $\mathrm{D}_\mathrm{Q} \times D_Q$)

Computation:

Query Vectors $Q = XW_Q$

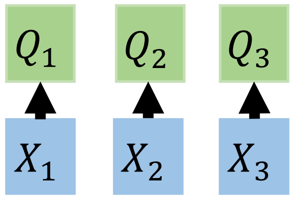
Key vectors: $K = XW_K \ (Shape: N_X \times D_Q)$

Value Vectors: $V = XW_V \ (Shape: N_X \times D_V)$

Similarities: $E = \frac{QX^T}{\sqrt{D_Q}} \ (Shape: N_X \times N_X) E_{i,j} = (Q_i \cdot K_j)/\sqrt{D_q}$

Attention weights: $A = softmax(E, \dim = 1) \ (Shape: N_X \times N_X)$

Output vectors: $Y = AV \ (Shape: N_X \times D_V) Y_i = \Sigma_j A_{i,j} V_j$



Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Self-Attention Layer

One query per input vector

Inputs:

Input vectors: X (Shape: $N_X \times D_Q$)

Key matrix: $W_K (Shape: D_X \times D_Q)$

Value matrix: $W_V (Shape: D_X \times D_V)$

Query matrix: $\mathbf{W}_{\mathrm{Q}}$ (Shape: $\mathrm{D_Q} \times D_Q$)

Computation:

Query Vectors $Q = XW_Q$

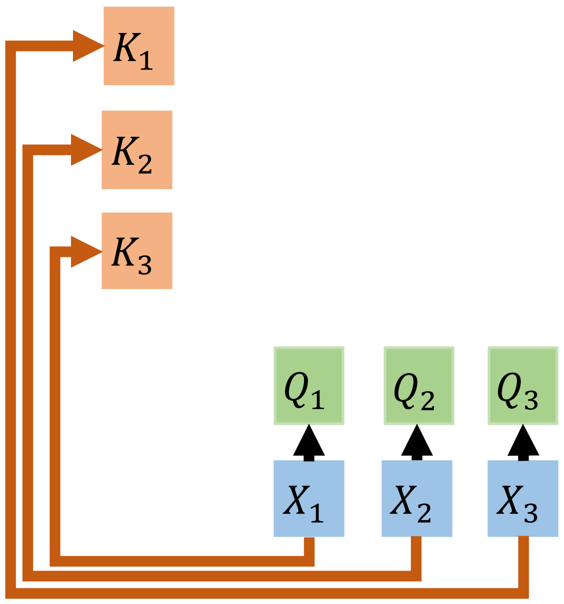
Key vectors: $K = XW_K$ $(Shape: N_X \times D_Q)$

Value Vectors: $V = XW_V$ $(Shape: N_X \times D_V)$

Similarities: $E = \frac{QX^T}{\sqrt{D_Q}}$ $(Shape: N_X \times N_X) E_{i,j} = (Q_i \cdot K_j)/\sqrt{D_q}$

Attention weights: $A = softmax(E, \mathrm{dim} = 1)$ $(Shape: N_X \times N_X)$

Output vectors: $Y = AV$ $(Shape: N_X \times D_V) Y_i = \Sigma_j A_{i,j} V_j$



Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Self-Attention Layer

One query per input vector

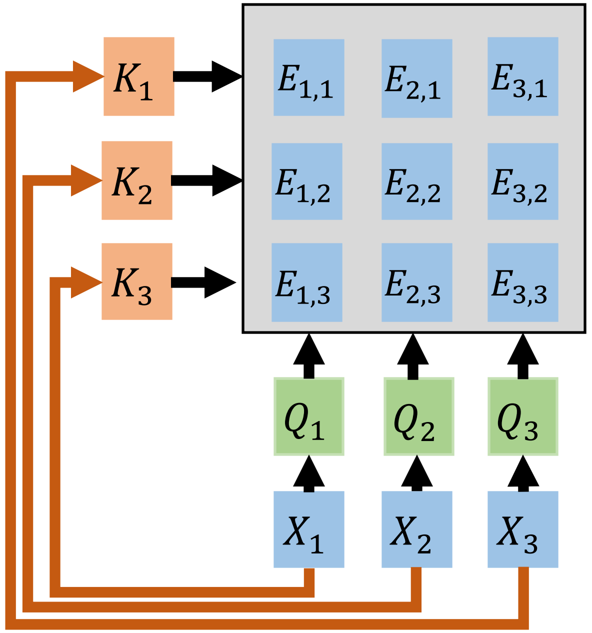
Inputs:

Input vectors: X (Shape: $N_X \times D_Q$)

Key matrix: $W_K (Shape: D_X \times D_Q)$

Value matrix: $W_V (Shape: \; D_X \times D_V)$

Query matrix: $\mathbf{W}_\mathrm{Q}$ (Shape: $\mathrm{D_Q} \times D_Q$)



Computation:

Query Vectors $Q = XW_Q$

Key vectors: $K = XW_K \; (Shape: \; N_X \times D_Q)$

Value Vectors: $V = XW_V \; (Shape: \; N_X \times D_V)$

Similarities: $E = \frac{QX^T}{\sqrt{D_Q}} \; (Shape: \; N_X \times N_X) E_{i,j} = (Q_i \cdot K_j)/\sqrt{D_q}$

Attention weights: $A = softmax(E, \mathrm{dim} = 1) \; (Shape: \; N_X \times N_X)$

Output vectors: $Y = AV \; (Shape: \; N_X \times D_V) Y_i = \Sigma_j A_{i,j} V_j$

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Self-Attention Layer

One query per input vector

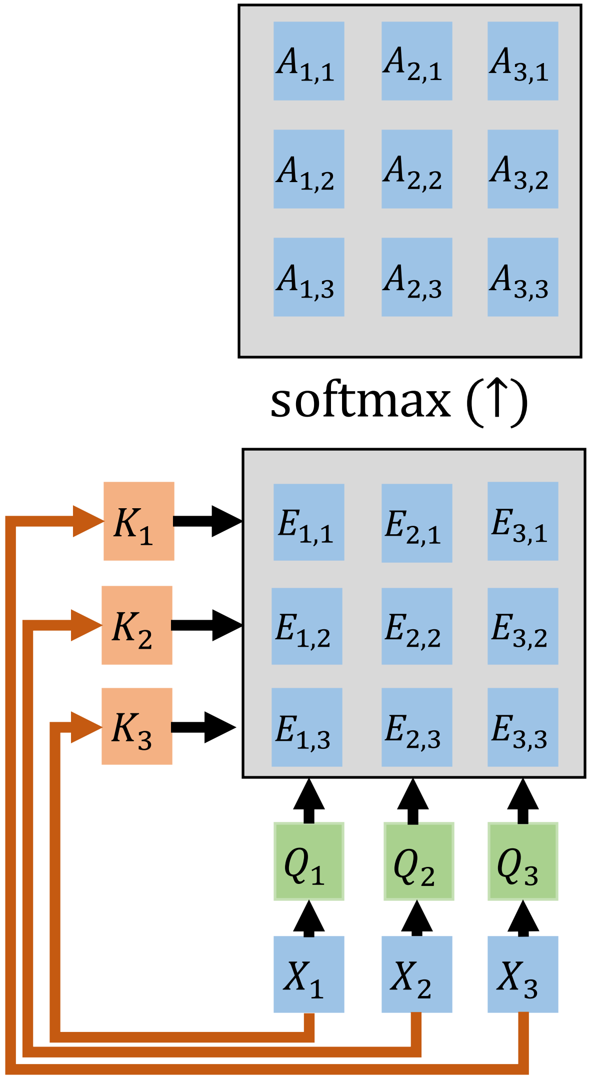
Inputs:

Input vectors: X (Shape: $N_X \times D_Q$)

Key matrix: $W_K(Shape: D_X \times D_Q)$

Value matrix: $W_V(Shape:\ D_X \times D_V)$

Query matrix: $\mathbf{W}_\mathrm{Q}$ (Shape: $\mathrm{D}_\mathrm{Q} \times D_Q$)



Computation:

Query Vectors $Q = XW_Q$

Key vectors: $K = XW_K$ $(Shape:\ N_X \times D_Q)$

Value Vectors: $V = XW_V$ $(Shape:\ N_X \times D_V)$

Similarities: $E = \frac{QX^T}{\sqrt{D_Q}}$ $(Shape:\ N_X \times N_X)$ $E_{i,j} = (Q_i \cdot K_j)/\sqrt{D_q}$

Attention weights: $A = softmax(E, \mathrm{dim} = 1)$ $(Shape:\ N_X \times N_X)$

Output vectors: $Y = AV$ $(Shape:\ N_X \times D_V)$ $Y_i = \Sigma_j A_{i,j} V_j$

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Self-Attention Layer

One query per input vector

Inputs:

Input vectors: X (Shape: $N_X \times D_Q$)

Key matrix: $W_K (Shape: D_X \times D_Q)$

Value matrix: $W_V (Shape: D_X \times D_V)$

Query matrix: $\mathrm{W_Q}$ (Shape: $\mathrm{D_Q} \times D_Q$)

Computation:

Query Vectors $Q = XW_Q$

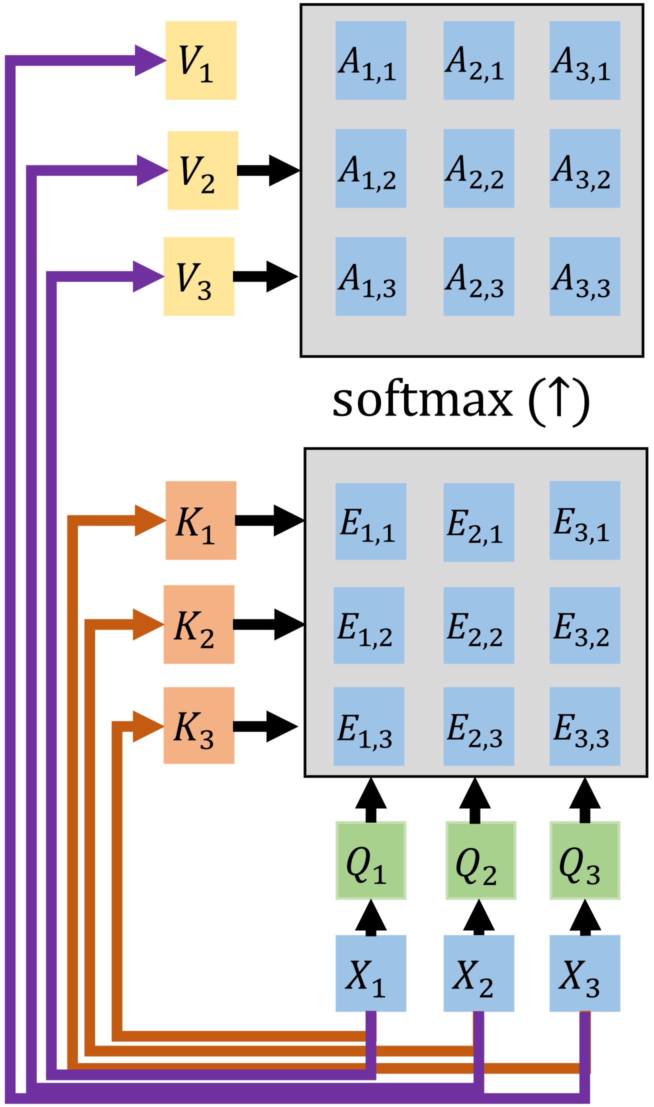
Key vectors: $K = XW_K$ $(Shape: N_X \times D_Q)$

Value Vectors: $V = XW_V$ $(Shape: N_X \times D_V)$

Similarities: $E = \frac{QX^T}{\sqrt{D_Q}}$ $(Shape: N_X \times N_X) E_{i,j} = (Q_i \cdot K_j)/\sqrt{D_q}$

Attention weights: $A = softmax(E, \mathrm{dim} = 1)$ $(Shape: N_X \times N_X)$

Output vectors: $Y = AV$ $(Shape: N_X \times D_V) Y_i = \Sigma_j A_{i,j} V_j$



Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Self-Attention Layer

One query per input vector

Inputs:

Input vectors: X (Shape: $N_X \times D_Q$)

Key matrix: $W_K (Shape: D_X \times D_Q)$

Value matrix: $W_V (Shape: D_X \times D_V)$

Query matrix: $\mathrm{W_Q}$ (Shape: $\mathrm{D_Q} \times D_Q$)

Computation:

Query Vectors $Q = XW_Q$

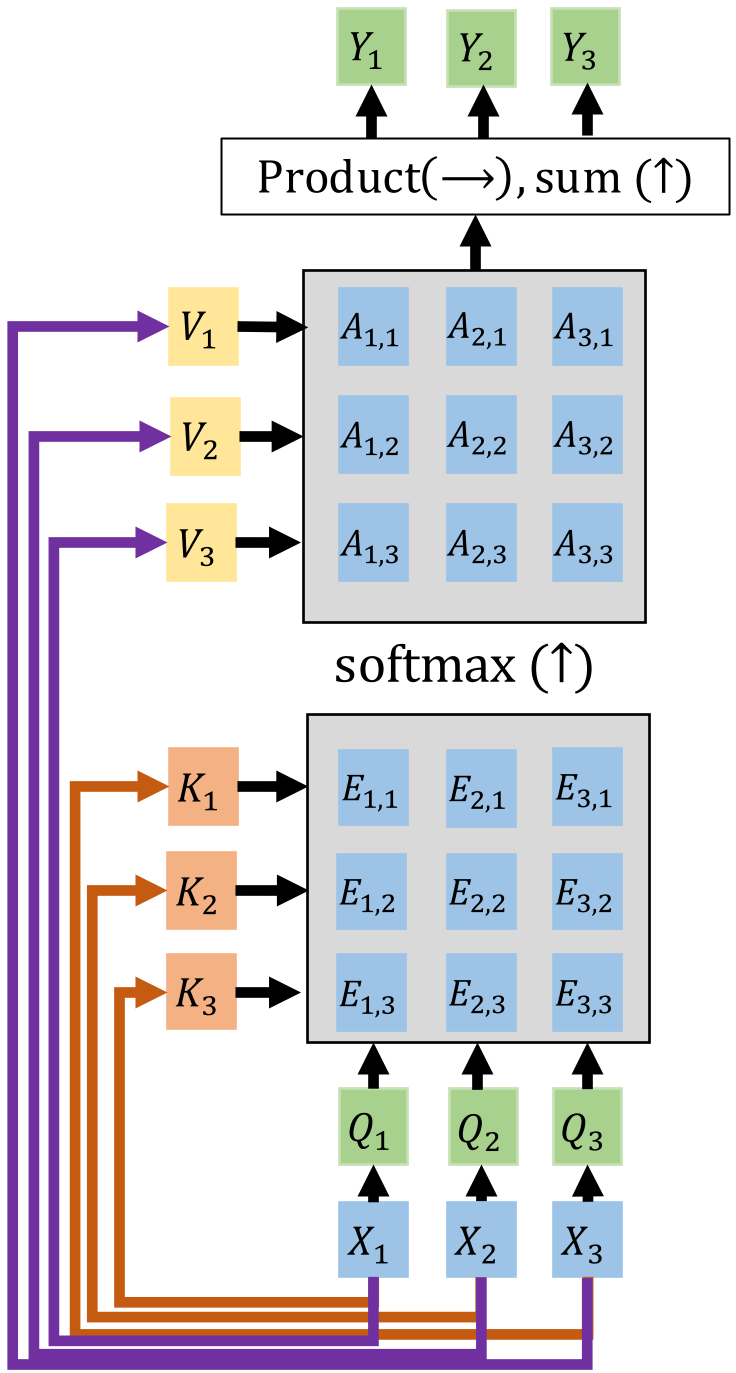
Key vectors: $K = XW_K$ $(Shape: N_X \times D_Q)$

Value Vectors: $V = XW_V$ $(Shape: N_X \times D_V)$

Similarities: $E = \frac{QX^T}{\sqrt{D_Q}}$ $(Shape: N_X \times N_X) E_{i,j} = (Q_i \cdot K_j)/\sqrt{D_q}$

Attention weights: $A = softmax(E, \mathrm{dim} = 1)$ $(Shape: N_X \times N_X)$

Output vectors: $Y = AV$ $(Shape: N_X \times D_V) Y_i = \Sigma_j A_{i,j} V_j$



Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Multi-head Self-Attention Layer

Inputs:

Input vectors: X (Shape: $N_X \times D_Q$)

Key matrix: $W_K (Shape: D_X \times D_Q)$

Value matrix: $W_V (Shape: \; D_X \times D_V)$

Query matrix: $\mathbf{W}_{\mathrm{Q}}$ (Shape: $\mathrm{D_Q} \times D_Q$)

Computation:

Query Vectors $Q = XW_Q$

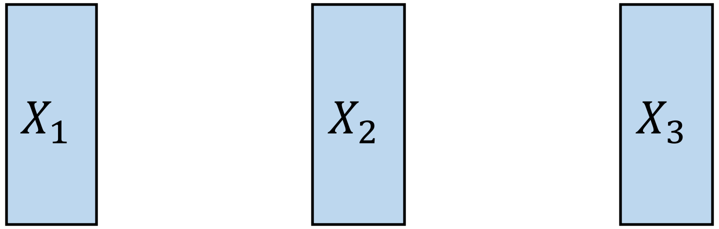
Key vectors: $K = XW_K \; (Shape: \; N_X \times D_Q)$

Value Vectors: $V = XW_V \; (Shape: \; N_X \times D_V)$

Similarities: $E = \frac{QX^T}{\sqrt{D_Q}} \; (Shape: \; N_X \times N_X) E_{i,j} = (Q_i \cdot K_j)/\sqrt{D_q}$

Attention weights: $A = softmax(E, \mathrm{dim} = 1) \; (Shape: \; N_X \times N_X)$

Output vectors: $Y = AV \; (Shape: \; N_X \times D_V) Y_i = \Sigma_j A_{i,j} V_j$



Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Multi-head Self-Attention Layer

Inputs:

Input vectors: X (Shape: $N_X \times D_Q$)

Key matrix: $W_K (Shape: D_X \times D_Q)$

Value matrix: $W_V (Shape: D_X \times D_V)$

Query matrix: $\mathbf{W}_\mathrm{Q}$ (Shape: $\mathrm{D}_\mathrm{Q} \times D_Q$)

Computation:

Query Vectors $Q = XW_Q$

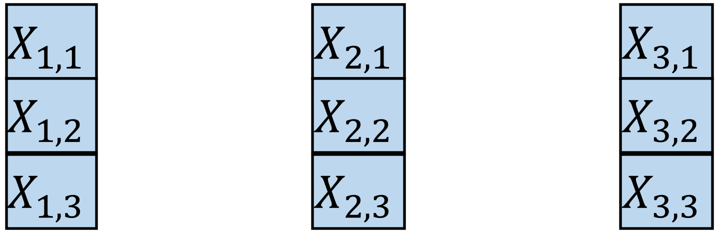
Key vectors: $K = XW_K \ (Shape: N_X \times D_Q)$

Value Vectors: $V = XW_V \ (Shape: N_X \times D_V)$

Similarities: $E = \frac{QX^T}{\sqrt{D_Q}} \ (Shape: N_X \times N_X) E_{i,j} = (Q_i \cdot K_j)/\sqrt{D_q}$

Attention weights: $A = softmax(E, \dim = 1) \ (Shape: N_X \times N_X)$

Output vectors: $Y = AV \ (Shape: N_X \times D_V) Y_i = \Sigma_j A_{i,j} V_j$



Multi-head Self-Attention Layer

Inputs:

Input vectors: X (Shape: $N_X \times D_Q$)

Key matrix: $W_K (Shape: D_X \times D_Q)$

Value matrix: $W_V (Shape: \ D_X \times D_V)$

Query matrix: $\mathbf{W}_\mathrm{Q}$ (Shape: $\mathrm{D_Q} \times D_Q$)

Computation:

Query Vectors $Q = XW_Q$

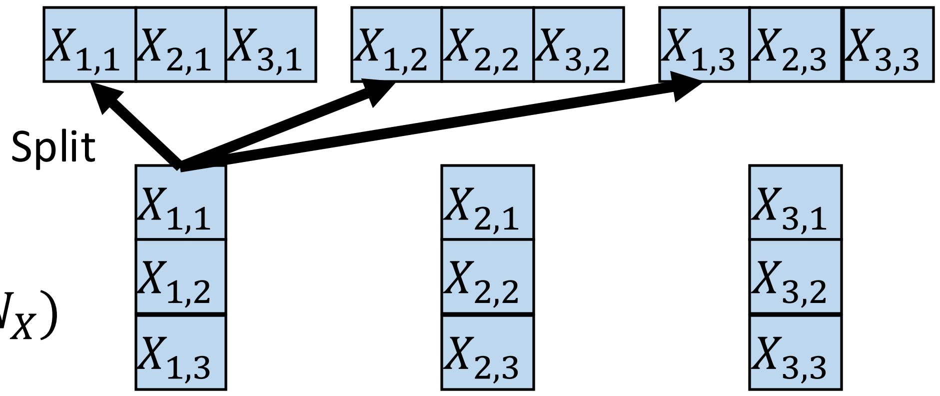
Key vectors: $K = XW_K \ (Shape: \ N_X \times D_Q)$

Value Vectors: $V = XW_V \ (Shape: \ N_X \times D_V)$

Similarities: $E = \frac{QX^T}{\sqrt{D_Q}} \ (Shape: \ N_X \times N_X) E_{i,j} = (Q_i \cdot K_j)/\sqrt{D_q}$

Attention weights: $A = softmax(E, \mathrm{dim} = 1) \ (Shape: \ N_X \times N_X)$

Output vectors: $Y = AV \ (Shape: \ N_X \times D_V) Y_i = \Sigma_j A_{i,j} V_j$



Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Multi-head Self-Attention Layer

Inputs:

Input vectors: X (Shape: $N_X \times D_Q$)

Key matrix: $W_K (Shape: D_X \times D_Q)$

Value matrix: $W_V (Shape: \ D_X \times D_V)$

Query matrix: $\mathbf{W}_Q$ (Shape: $\mathrm{D}_Q \times D_Q$)

Computation:

Query Vectors $Q = XW_Q$

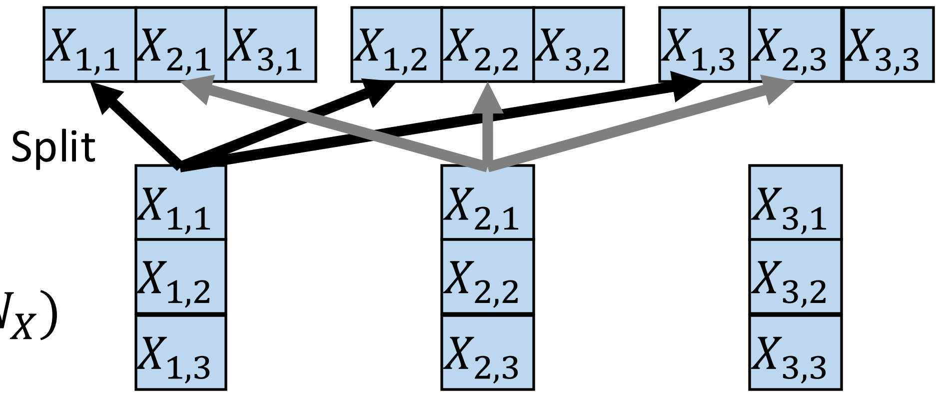
Key vectors: $K = XW_K \ (Shape: \ N_X \times D_Q)$

Value Vectors: $V = XW_V \ (Shape: \ N_X \times D_V)$

Similarities: $E = \frac{QX^T}{\sqrt{D_Q}} \ (Shape: \ N_X \times N_X) E_{i,j} = (Q_i \cdot K_j)/\sqrt{D_q}$

Attention weights: $A = softmax(E, \dim = 1) \ (Shape: \ N_X \times N_X)$

Output vectors: $Y = AV \ (Shape: \ N_X \times D_V) Y_i = \Sigma_j A_{i,j} V_j$



Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Multi-head Self-Attention Layer

Inputs:

Input vectors: X (Shape: $N_X \times D_Q$)

Key matrix: $W_K (Shape: D_X \times D_Q)$

Value matrix: $W_V (Shape:\ D_X \times D_V)$

Query matrix: $\mathbf{W}_Q$ (Shape: $\mathrm{D}_Q \times D_Q$)

Computation:

Query Vectors $Q = XW_Q$

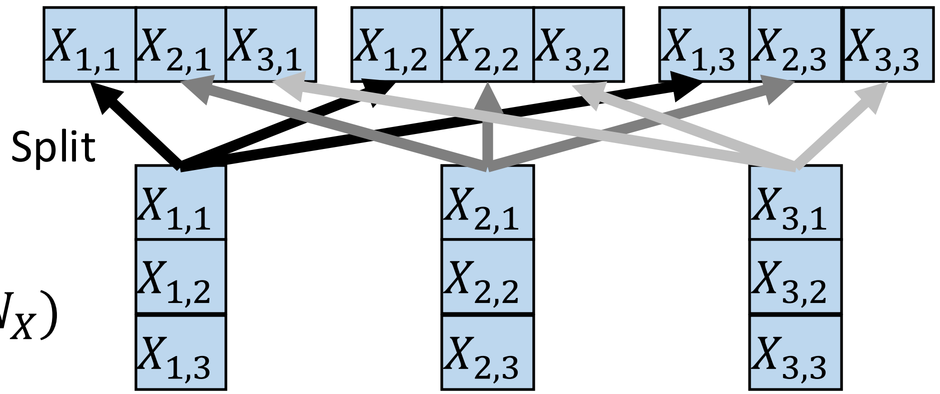
Key vectors: $K = XW_K\ (Shape:\ N_X \times D_Q)$

Value Vectors: $V = XW_V\ (Shape:\ N_X \times D_V)$

Similarities: $E = \frac{QX^T}{\sqrt{D_Q}}\ (Shape:\ N_X \times N_X) E_{i,j} = (Q_i \cdot K_j)/\sqrt{D_q}$

Attention weights: $A = softmax(E, \mathrm{dim} = 1)\ (Shape:\ N_X \times N_X)$

Output vectors: $Y = AV\ (Shape:\ N_X \times D_V) Y_i = \Sigma_j A_{i,j} V_j$



Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Multi-head Self-Attention Layer

Inputs:

Input vectors: X (Shape: $N_X \times D_Q$)

Key matrix: $W_K (Shape: D_X \times D_Q)$

Value matrix: $W_V (Shape: D_X \times D_V)$

Query matrix: $\mathbf{W_Q}$ (Shape: $D_Q \times D_Q$)

Computation:

Query Vectors $Q = XW_Q$

Key vectors: $K = XW_K$ $(Shape: N_X \times D_Q)$

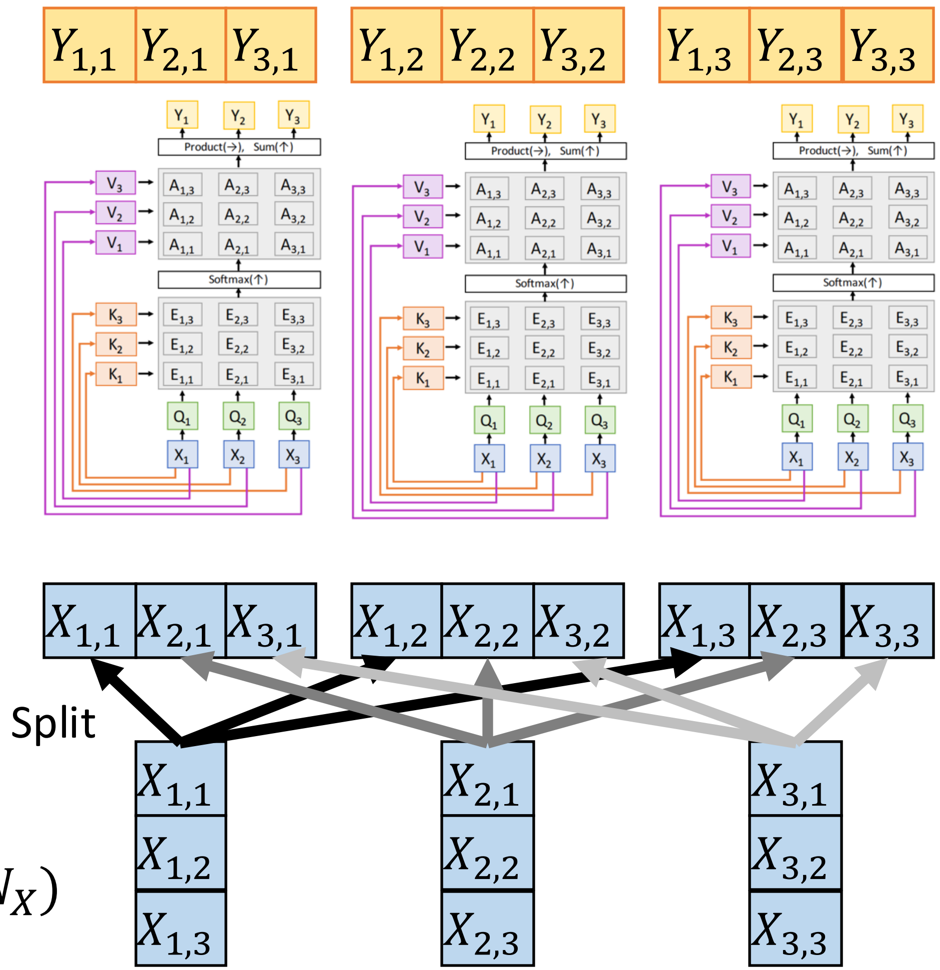
Value Vectors: $V = XW_V$ $(Shape: N_X \times D_V)$

Similarities: $E = \frac{QX^T}{\sqrt{D_Q}}$ $(Shape: N_X \times N_X) E_{i,j} = (Q_i \cdot K_j)/\sqrt{D_q}$

Attention weights: $A = softmax(E, \dim = 1)$ $(Shape: N_X \times N_X)$

Output vectors: $Y = AV$ $(Shape: N_X \times D_V) Y_i = \Sigma_j A_{i,j} V_j$

Run self-attention in parallel on each set of input vectors (different weights per head)



Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Multi-head Self-Attention Layer

Inputs:

Input vectors: X (Shape: $N_X \times D_Q$)

Key matrix: $W_K (Shape: D_X \times D_Q)$

Value matrix: $W_V (Shape: D_X \times D_V)$

Query matrix: $\mathbf{W}_\mathrm{Q}$ (Shape: $\mathrm{D_Q} \times D_Q$)

Computation:

Query Vectors $Q = XW_Q$

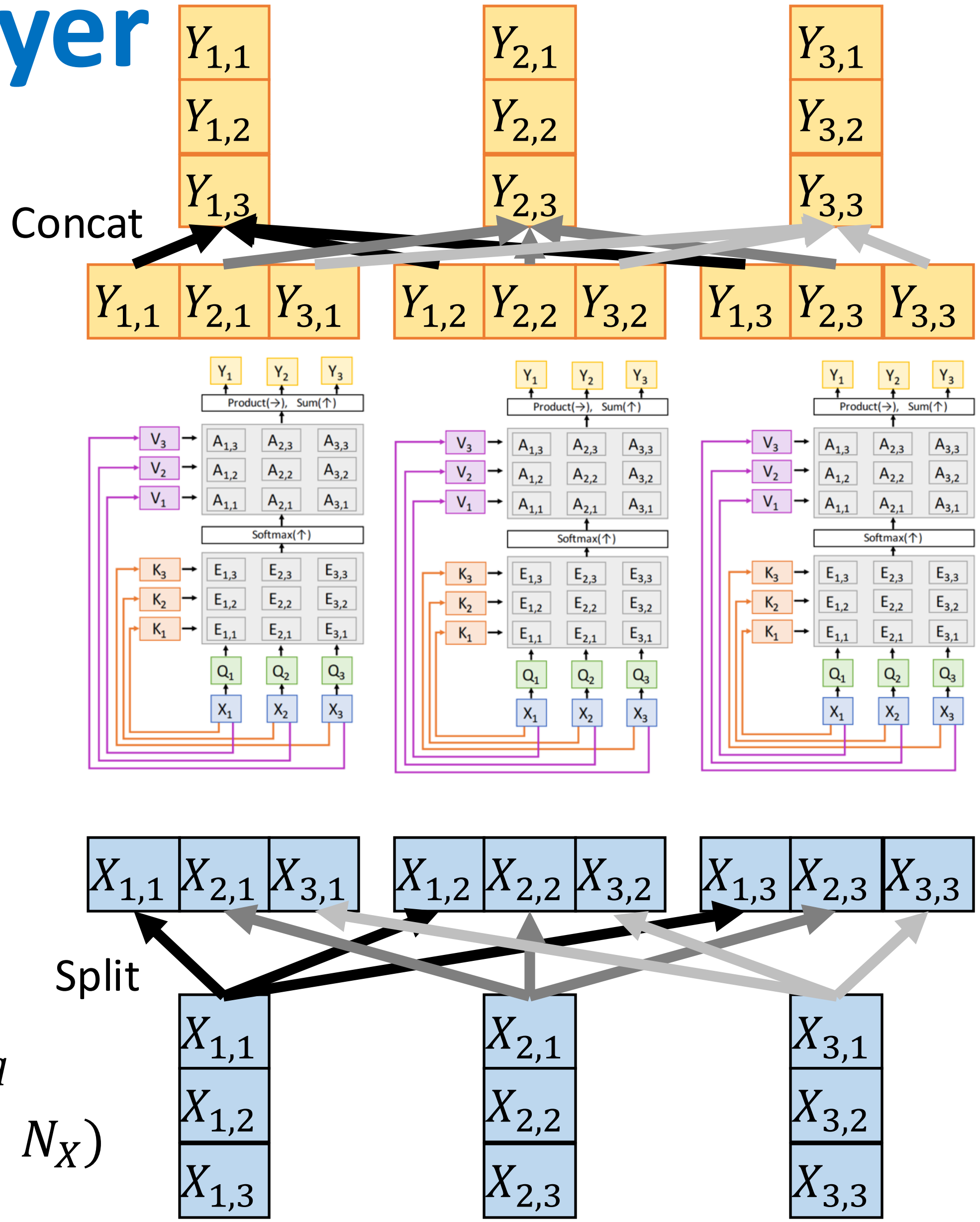
Key vectors: $K = XW_K\ (Shape: N_X \times D_Q)$

Value Vectors: $V = XW_V\ (Shape: N_X \times D_V)$

Similarities: $E = \frac{QX^T}{\sqrt{D_Q}}\ (Shape: N_X \times N_X) E_{i,j} = (Q_i \cdot K_j)/\sqrt{D_q}$

Attention weights: $A = softmax(E, \mathrm{dim} = 1)\ (Shape: N_X \times N_X)$

Output vectors: $Y = AV\ (Shape: N_X \times D_V) Y_i = \Sigma_j A_{i,j} V_j$



Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Multi-head Self-Attention Layer

Inputs:

Input vectors: X (Shape: $N_X \times D_Q$)

Key matrix: $W_K (Shape: D_X \times D_Q)$

Value matrix: $W_V (Shape: D_X \times D_V)$

Query matrix: $\mathbf{W_Q}$ (Shape: $\mathrm{D_Q} \times D_Q$)

Computation:

Query Vectors $Q = XW_Q$

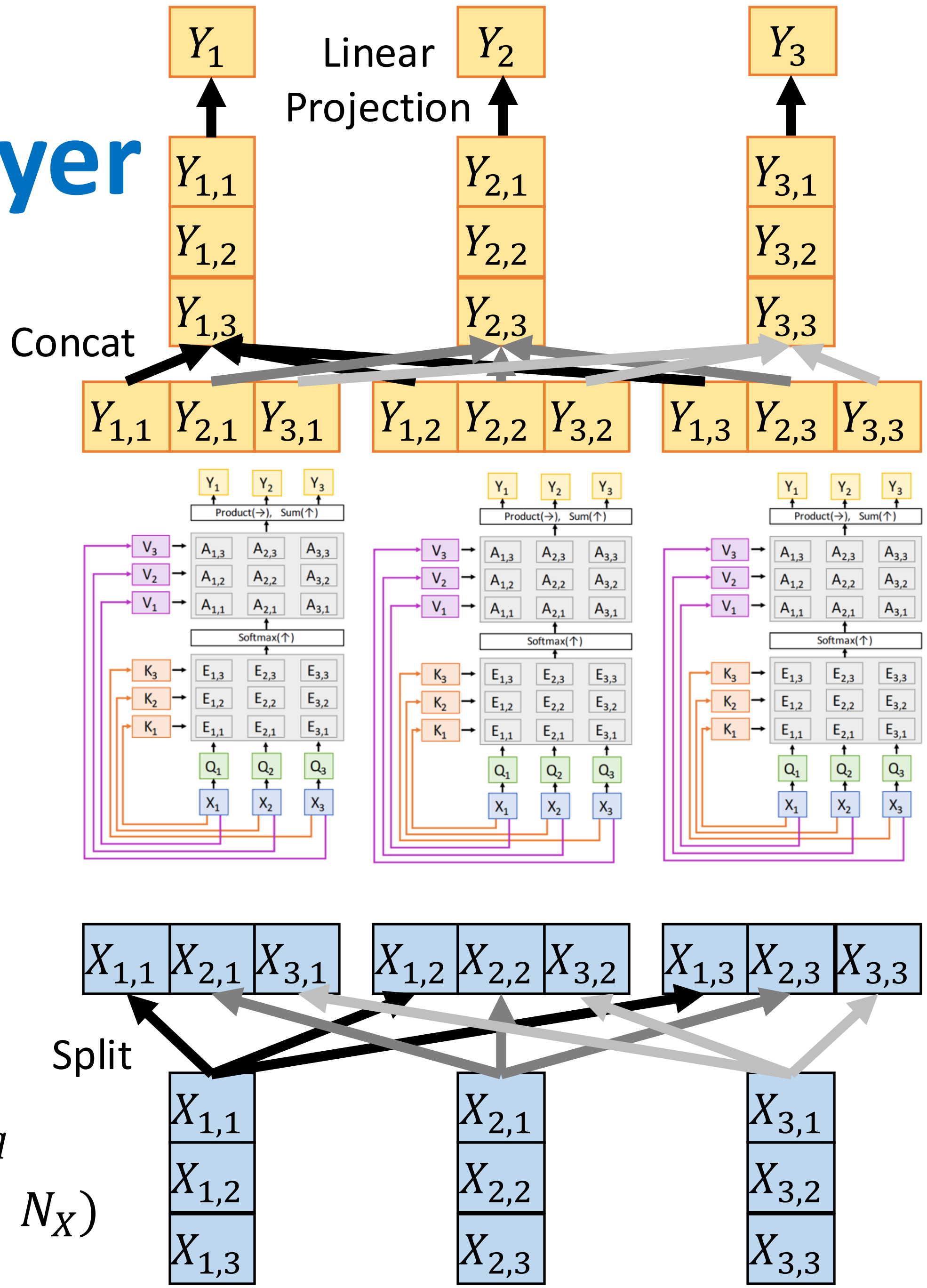
Key vectors: $K = XW_K \ (Shape: N_X \times D_Q)$

Value Vectors: $V = XW_V \ (Shape: N_X \times D_V)$

Similarities: $E = \frac{QX^T}{\sqrt{D_Q}} \ (Shape: N_X \times N_X) E_{i,j} = (Q_i \cdot K_j)/\sqrt{D_q}$

Attention weights: $A = softmax(E, \mathrm{dim} = 1) \ (Shape: N_X \times N_X)$

Output vectors: $Y = AV \ (Shape: N_X \times D_V) Y_i = \Sigma_j A_{i,j} V_j$



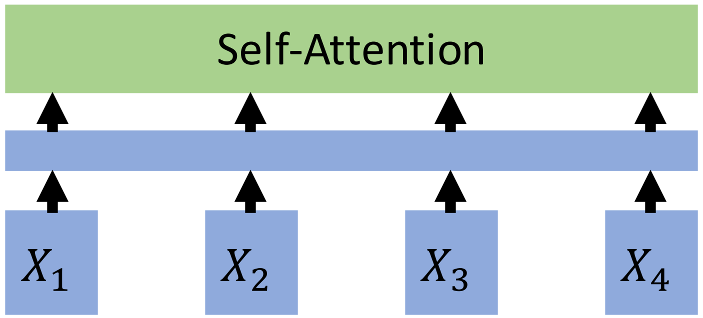
Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

The Transformer

X_1 X_2 X_3 X_4

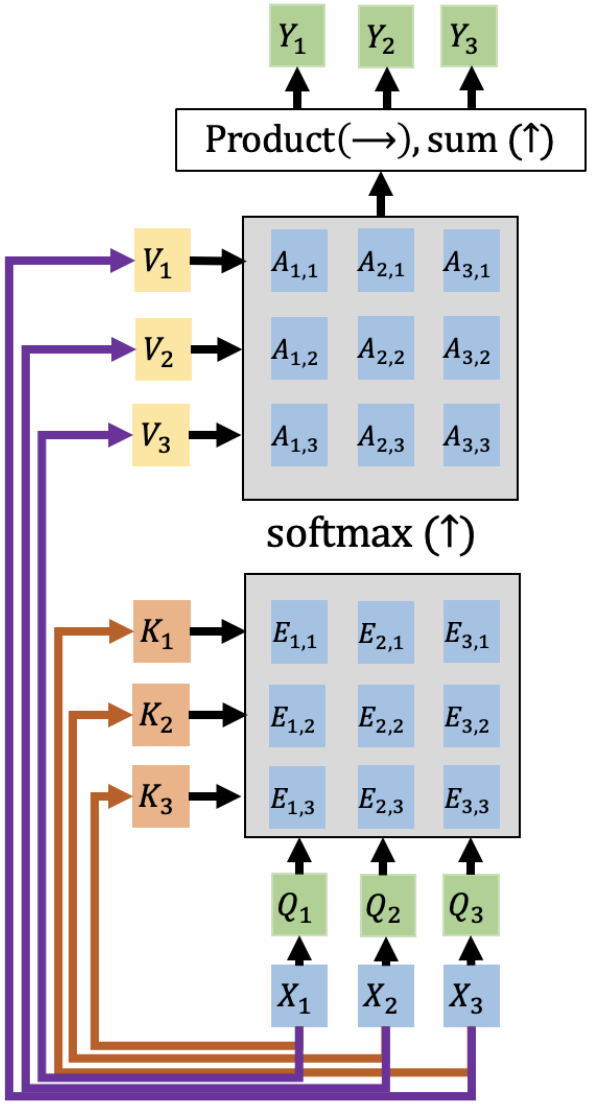
Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

The Transformer

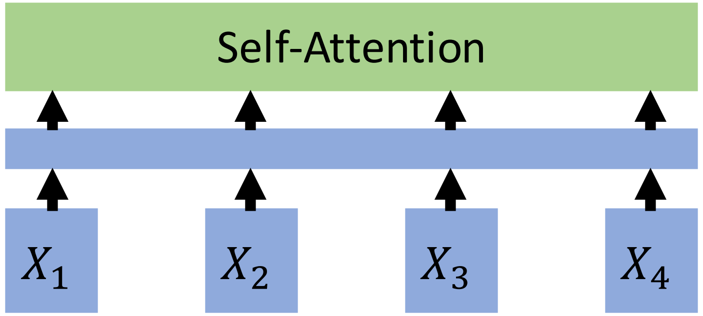


Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

The Transformer



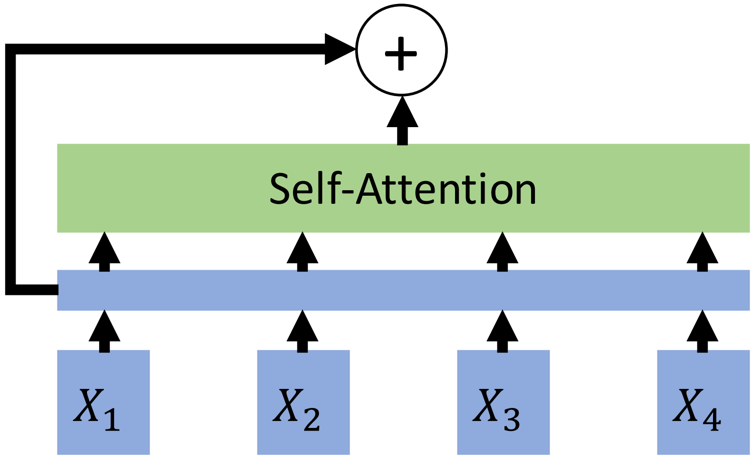
Self-Attention



Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

The Transformer

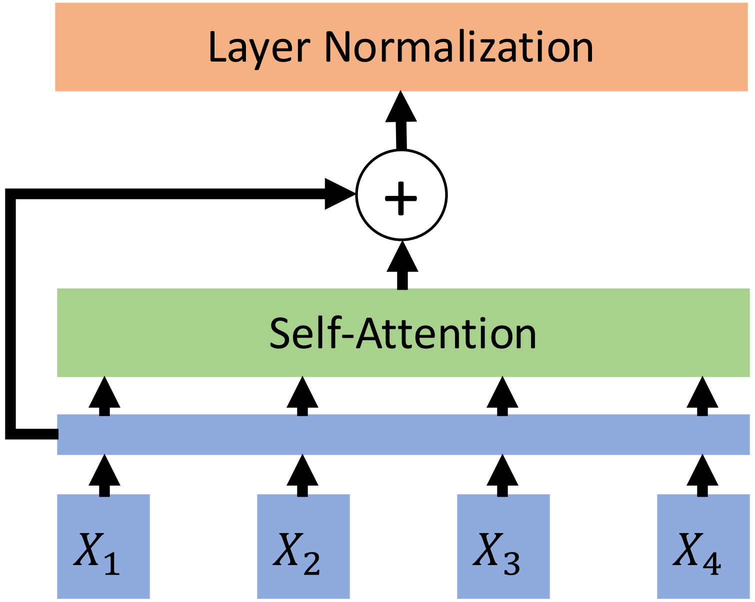
Residual connection
All vectors interact with each other



Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

The Transformer

Residual connection
All vectors interact with each other



Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

The Transformer

Layer Normalization:

Given $h_1, \ldots, h_N$ (Shape: C)
scale: γ (Shape: C)
shift: β (Shape: C)

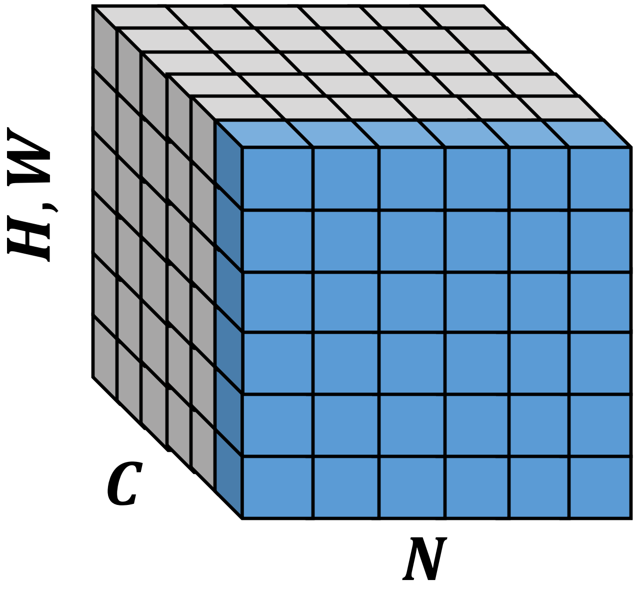
$$\mu_i = \frac{\Sigma_j h_{i,j}}{C} \text{ (scalar)}$$

$$\sigma_i = \left(\frac{\Sigma_j (h_{i,j} - \mu_i)^2}{C}\right)^{1/2} \text{ (scalar)}$$

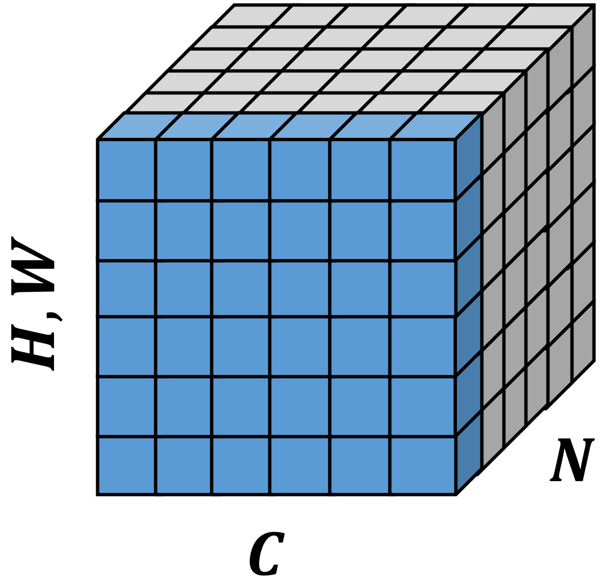
$$z_i = \frac{h_i - \mu_i}{\sigma_i}$$

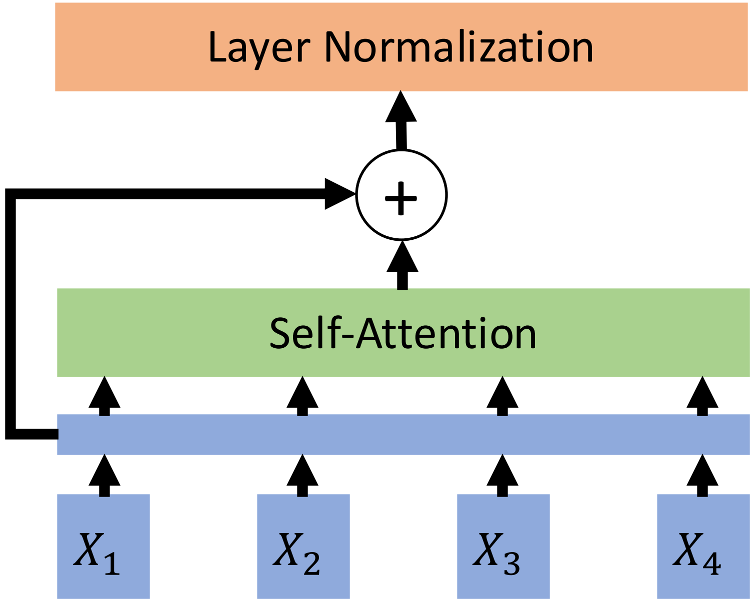
$$y_i = \gamma * z_i + \beta$$

Batch Norm



Layer Norm

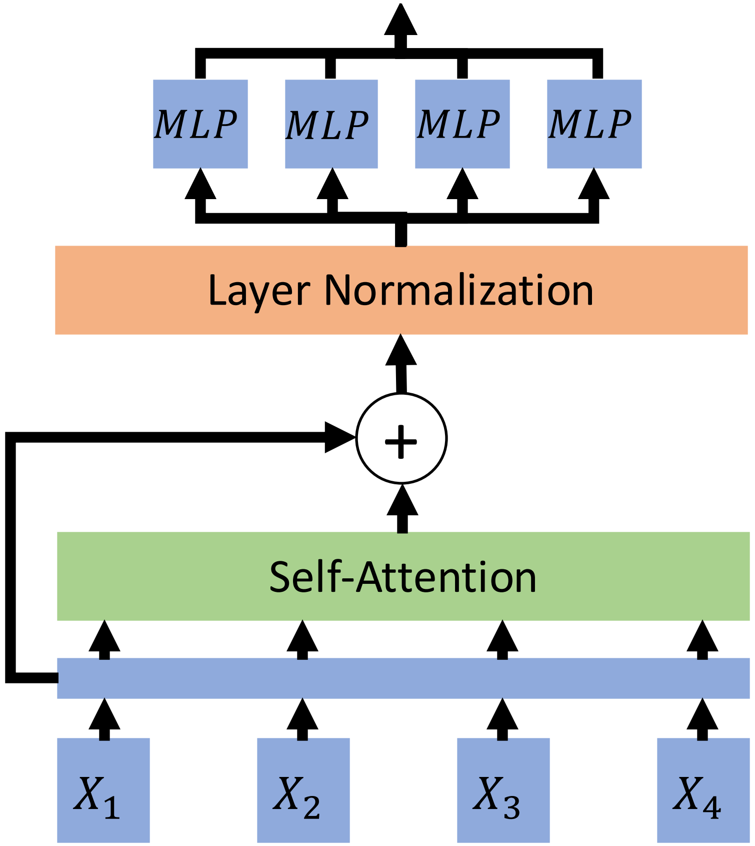




The Transformer

MLP: independently on each vector

Residual connection
All vectors interact with each other

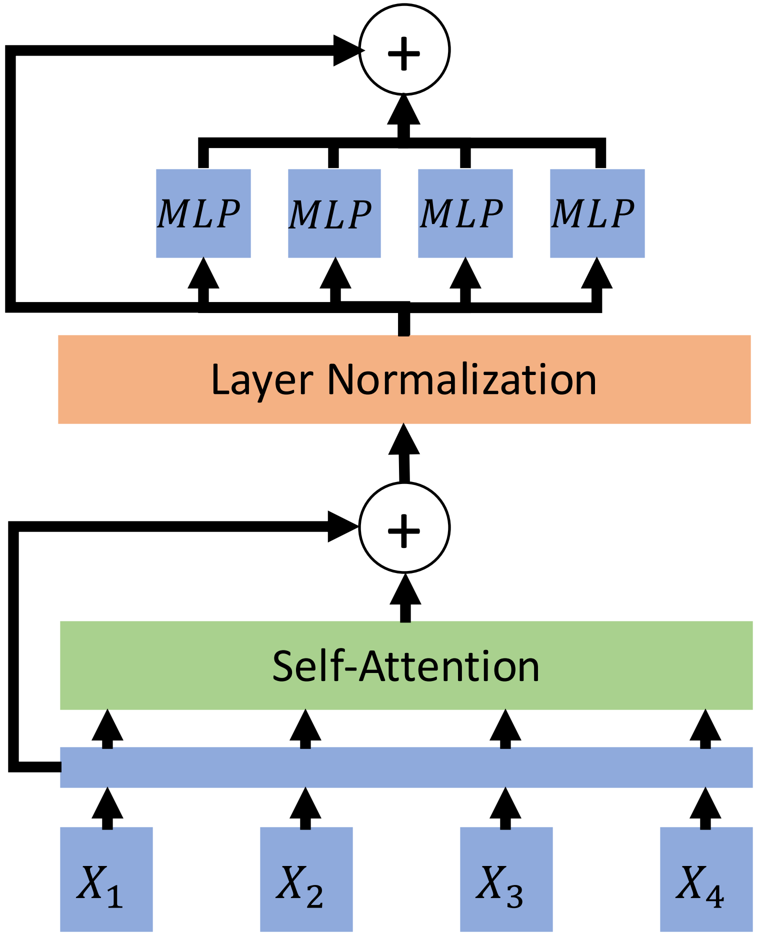


The Transformer

Residual connection

MLP: independently on each vector

Residual connection
All vectors interact with each other



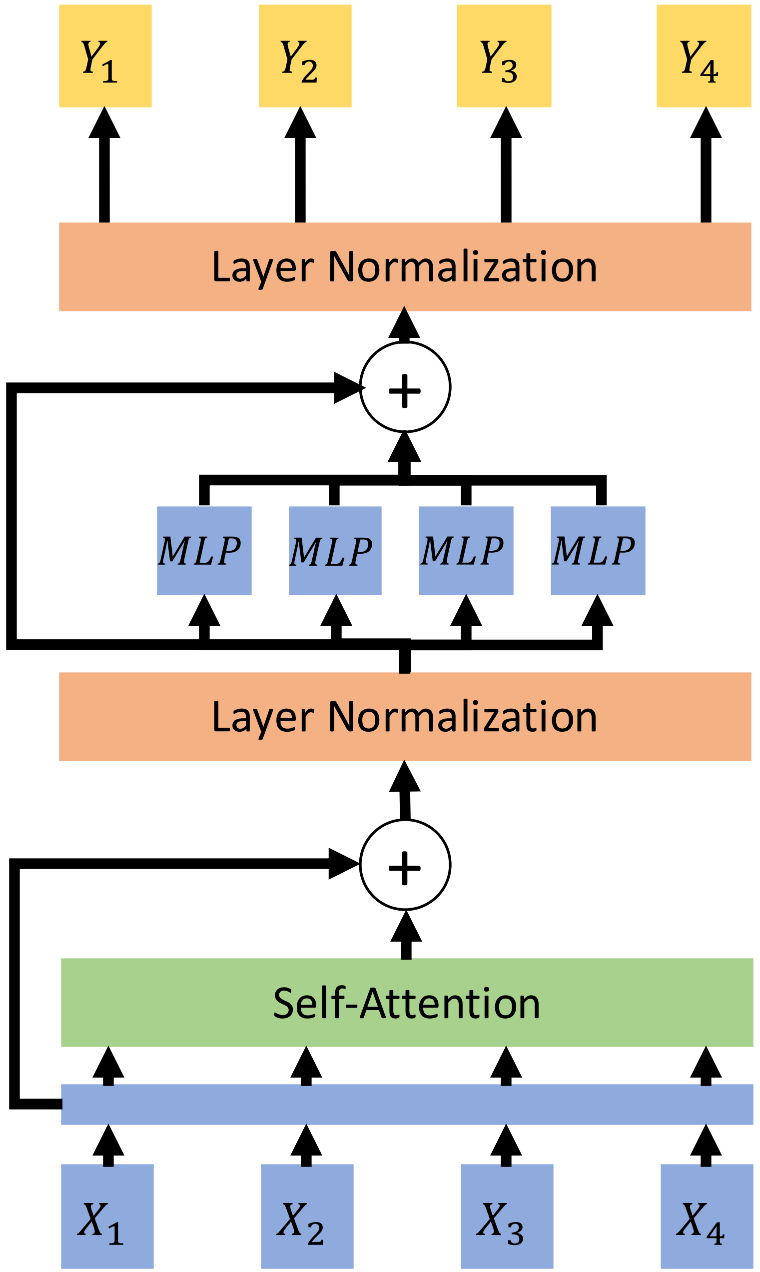
Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

The Transformer

Residual connection

MLP: independently on each vector

Residual connection
All vectors interact with each other

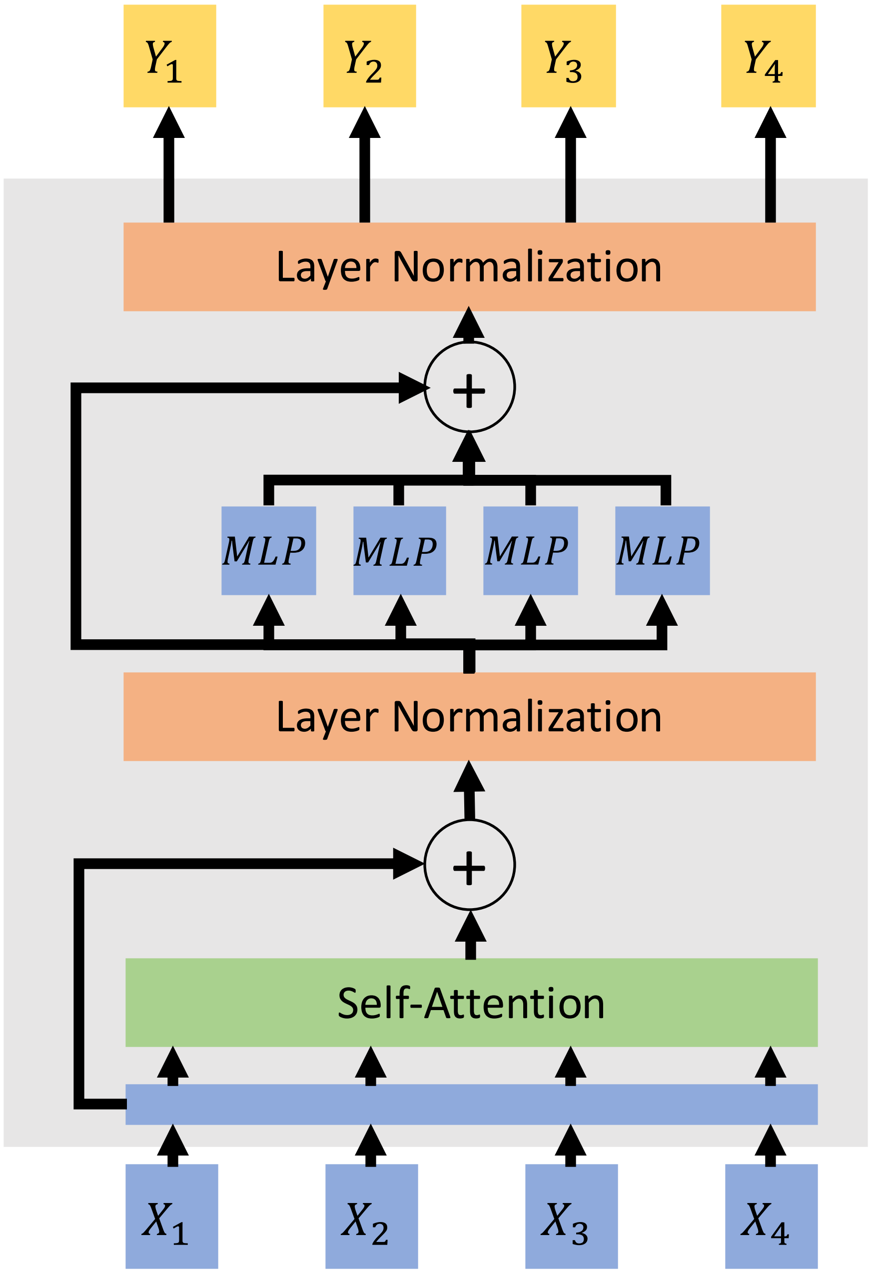


The Transformer

Transformer Block:

- **Input:** Set of vectors x
- **Output:** Set of vectors y

- Self-attention is the only interaction between vectors!
- Layer norm and MLP work independently per vector
- Highly scalable, highly parallelizable

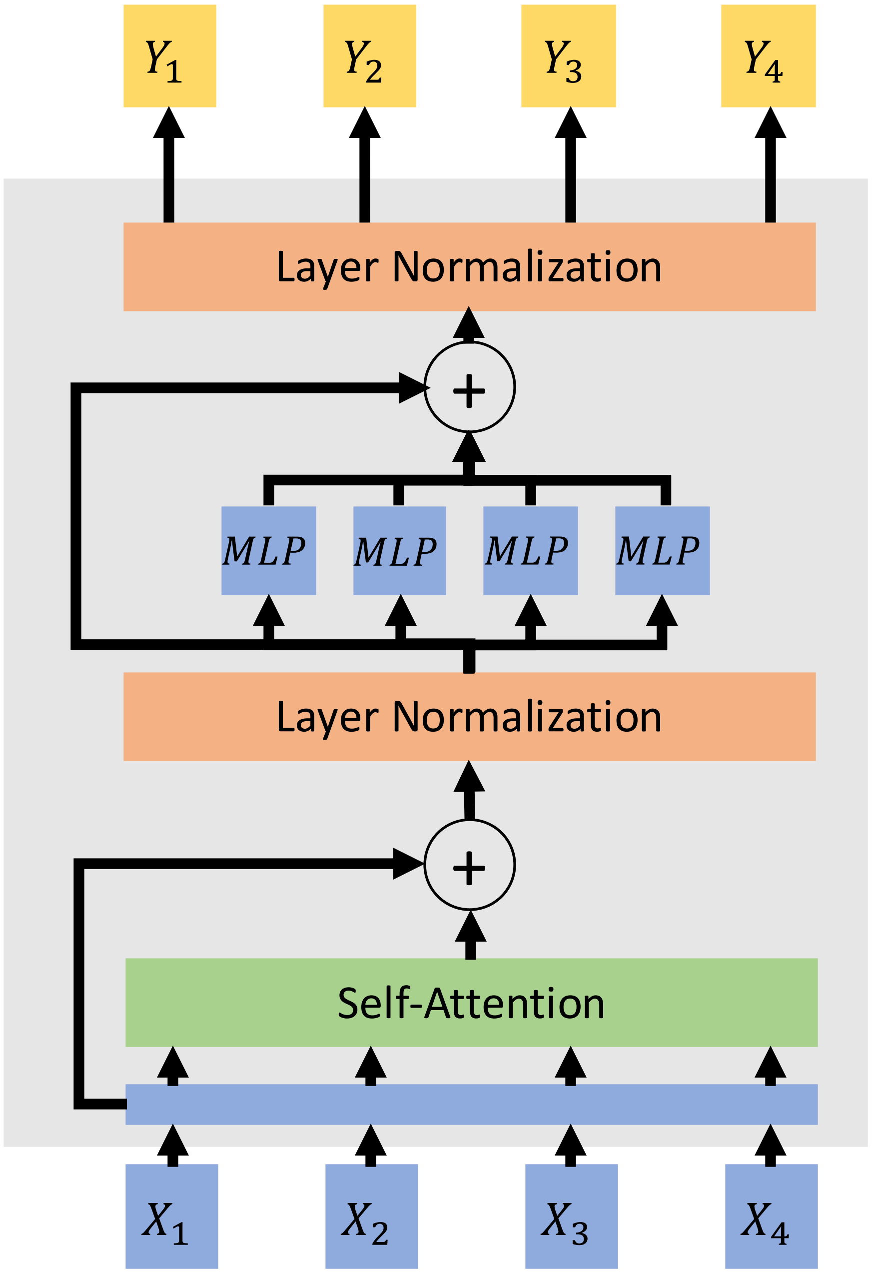


Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Post-Norm Transformer

- Layer Normalization is after the residual connections

- Gives more stable training, commonly used in practice



Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

The Transformer

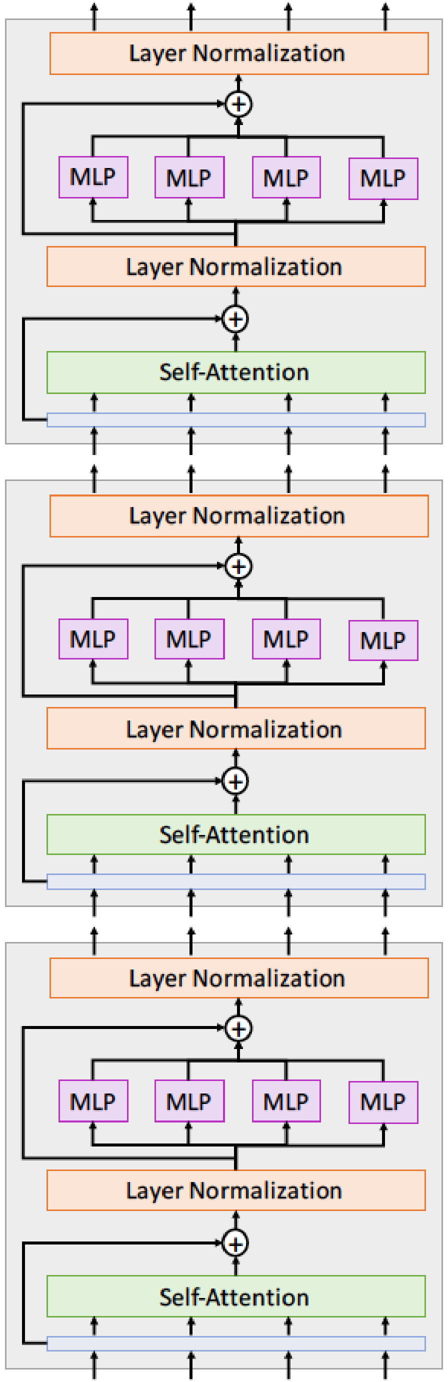
Transformer Block:

- **Input:** Set of vectors x
- **Output:** Set of vectors y

A **Transformer** is a sequence of transformer blocks

Vaswani et al:
12 blocks, $D_Q = 512$, 6 heads

- Self-attention is the only interaction between vectors!
- Layer norm and MLP work independently per vector
- Highly scalable, highly parallelizable



Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

The Transformer: Transfer Learning

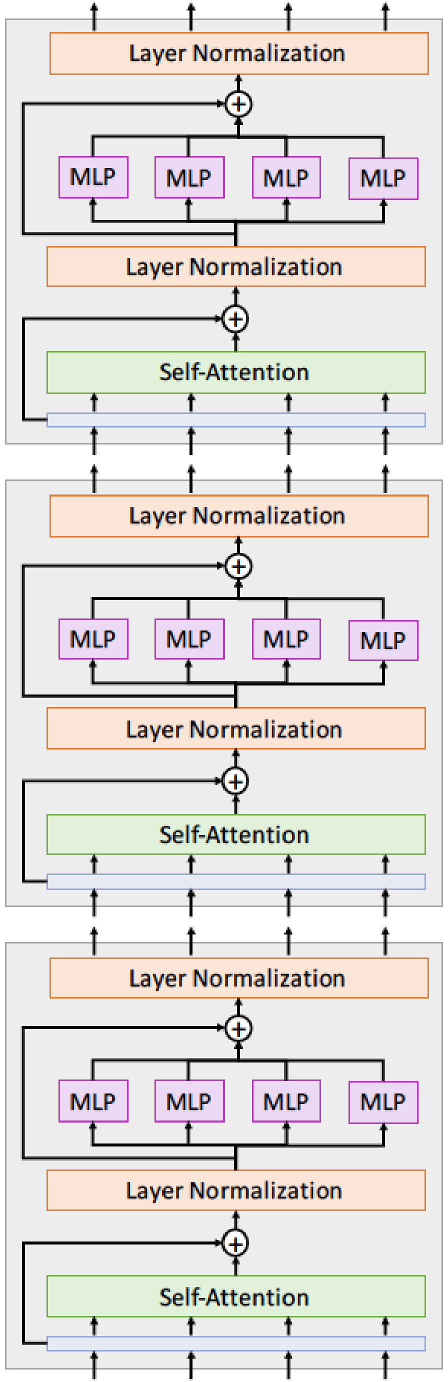
"ImageNet Moment for Natural Language Processing"

Pretraining:

- Download a lot of text from the internet
- Train a giant Transformer model for language modeling

Finetuning:

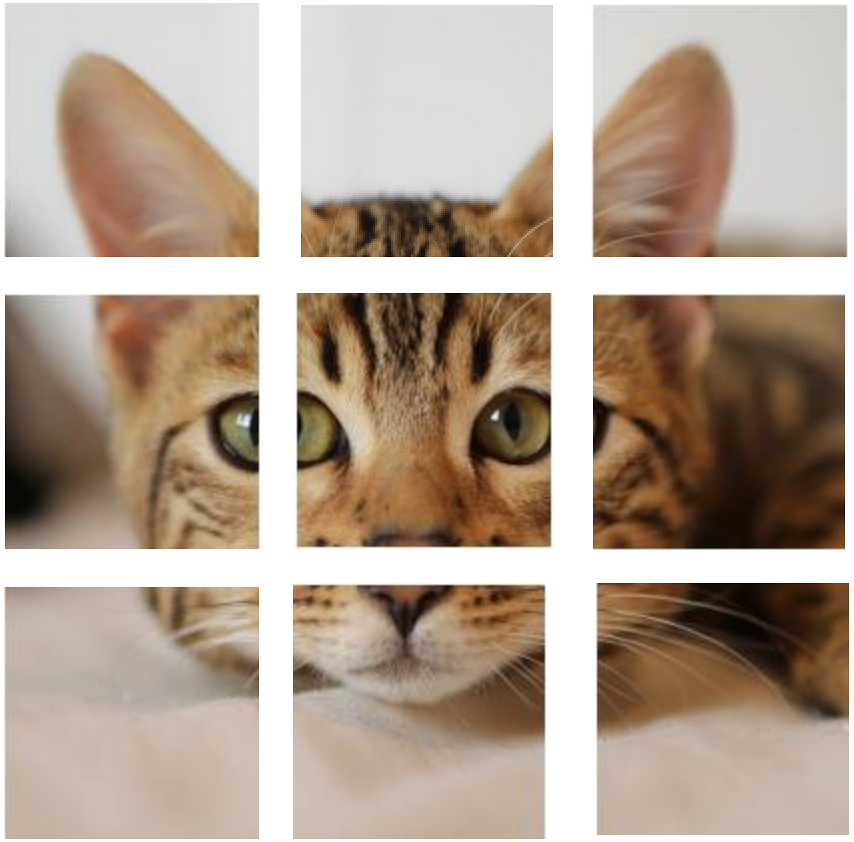
- Fine-tune the Transformer on your own NLP task



Vision Transformer (ViT)

Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Vision Transformer (ViT)

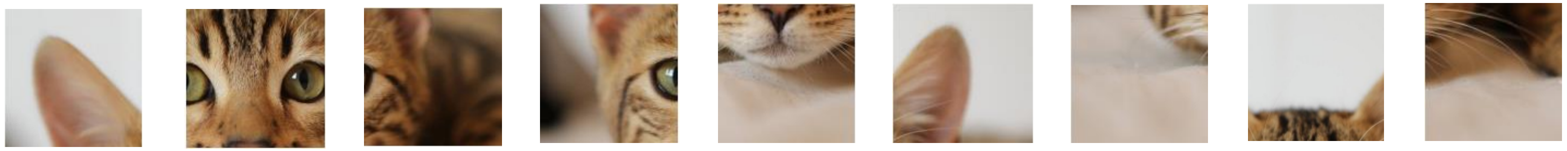


Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Vision Transformer (ViT)

N input patches, each of shape $3 \times 16 \times 16$



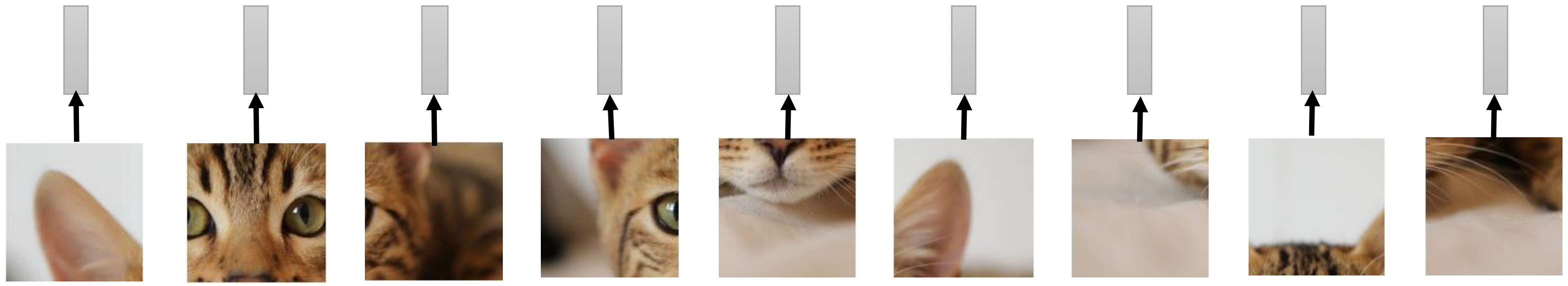
Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Vision Transformer (ViT)

Linear projection to
D-dimensional vector

N input patches, each
of shape $3 \times 16 \times 16$



Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

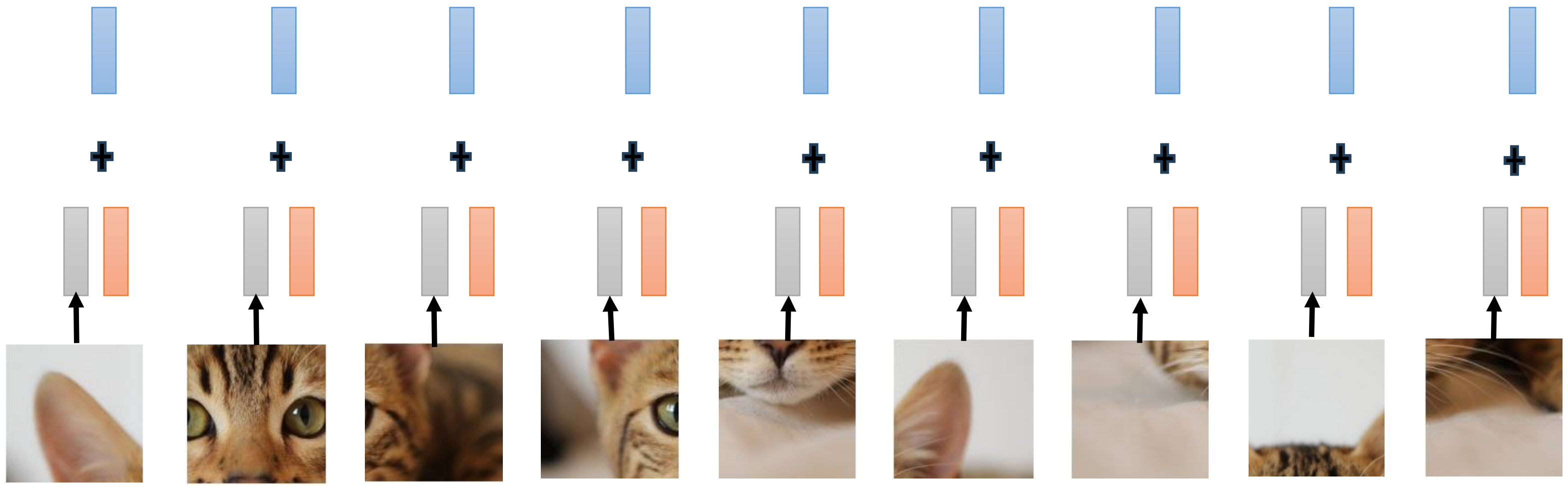
Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Vision Transformer (ViT)

Add positional embedding: learned D-dim vector per position

Linear projection to D-dimensional vector

N input patches, each of shape $3 \times 16 \times 16$



Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Vision Transformer (ViT)

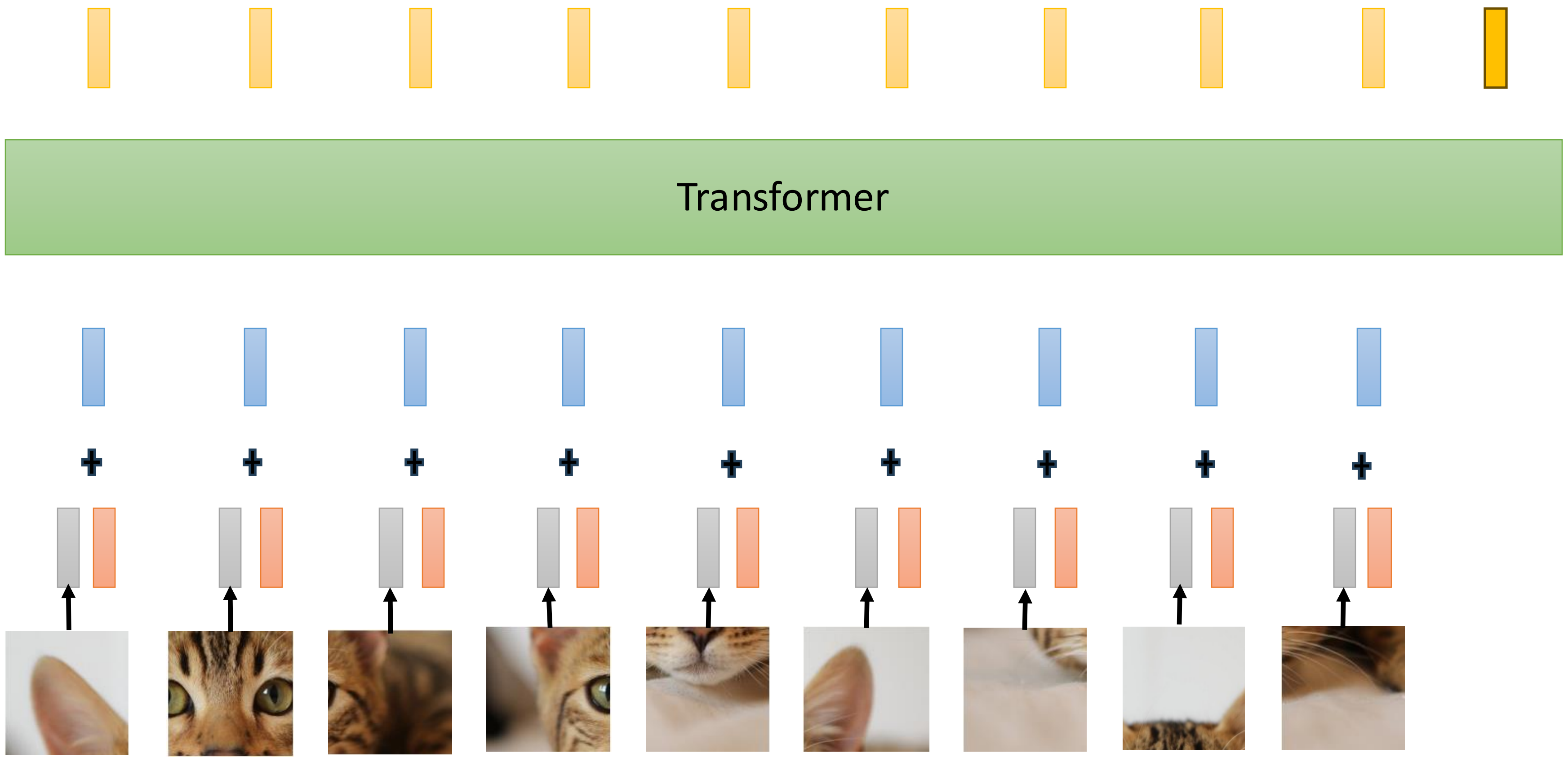
Output vectors

Exact same as
NLP Transformer!

Add positional embedding: learned D-dim vector per position

Linear projection to D-dimensional vector

N input patches, each of shape $3 \times 16 \times 16$



Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Vision Transformer (ViT)

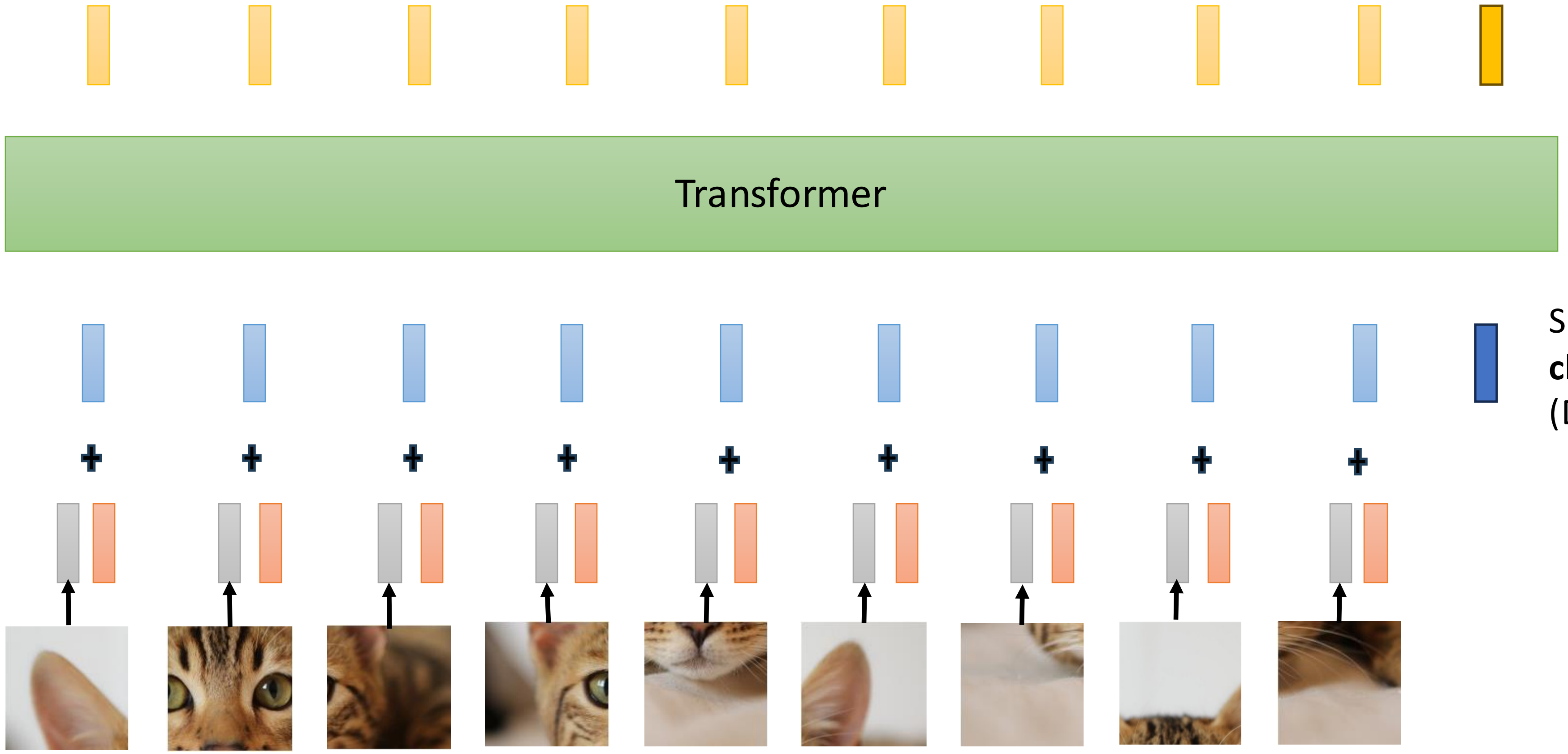
Output vectors

Exact same as
NLP Transformer!

Add positional
embedding: learned D-
dim vector per position

Linear projection to
D-dimensional vector

N input patches, each
of shape $3 \times 16 \times 16$



Special extra input:
classification token
(D dims, learned)

Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Vision Transformer (ViT)

Computer vision model with
no convolutions!

Linear projection to C-dim vector of predicted class scores

Output vectors

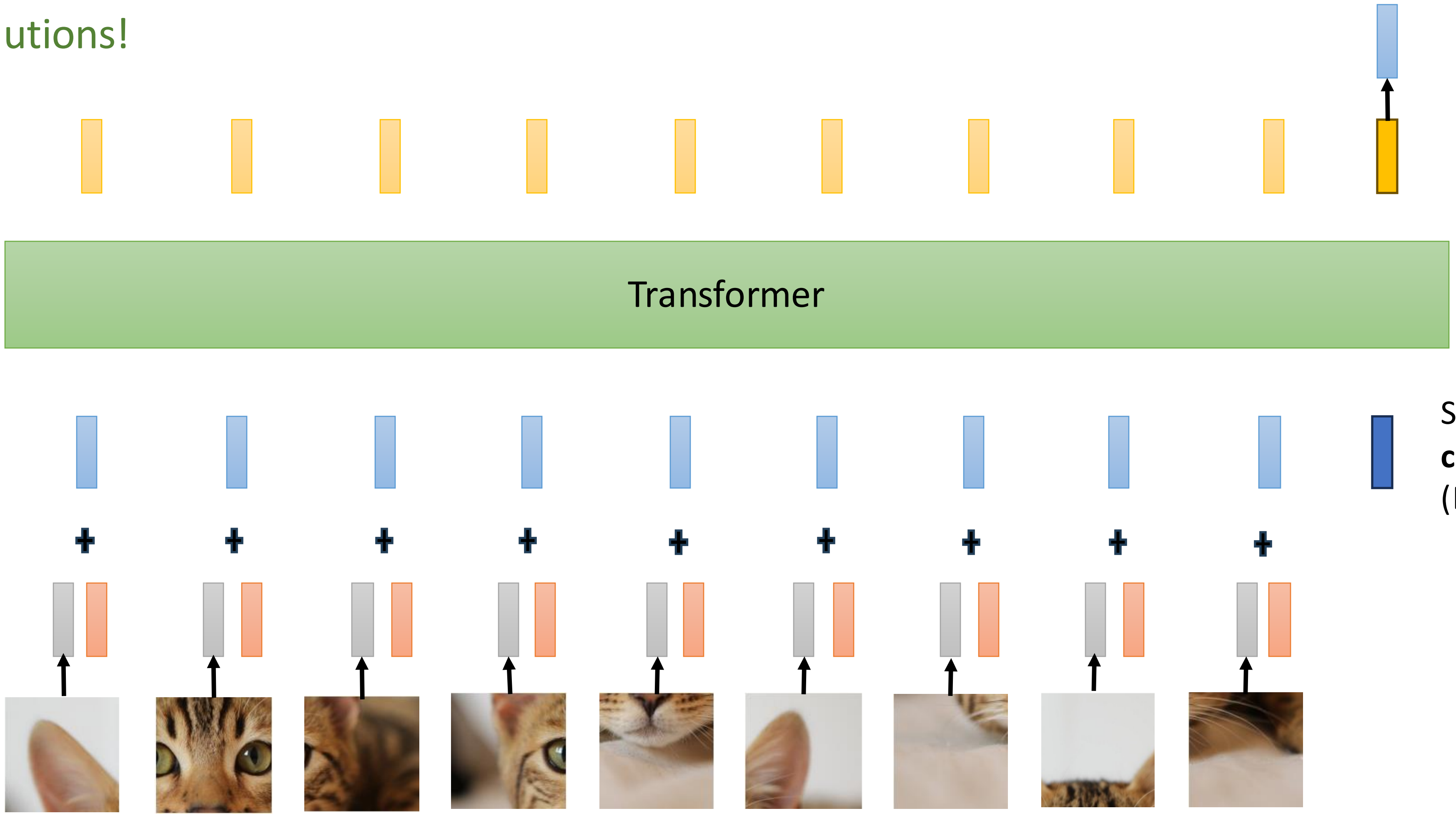
Exact same as NLP Transformer!

Add positional embedding: learned D-dim vector per position

Special extra input: **classification token** (D dims, learned)

Linear projection to D-dimensional vector

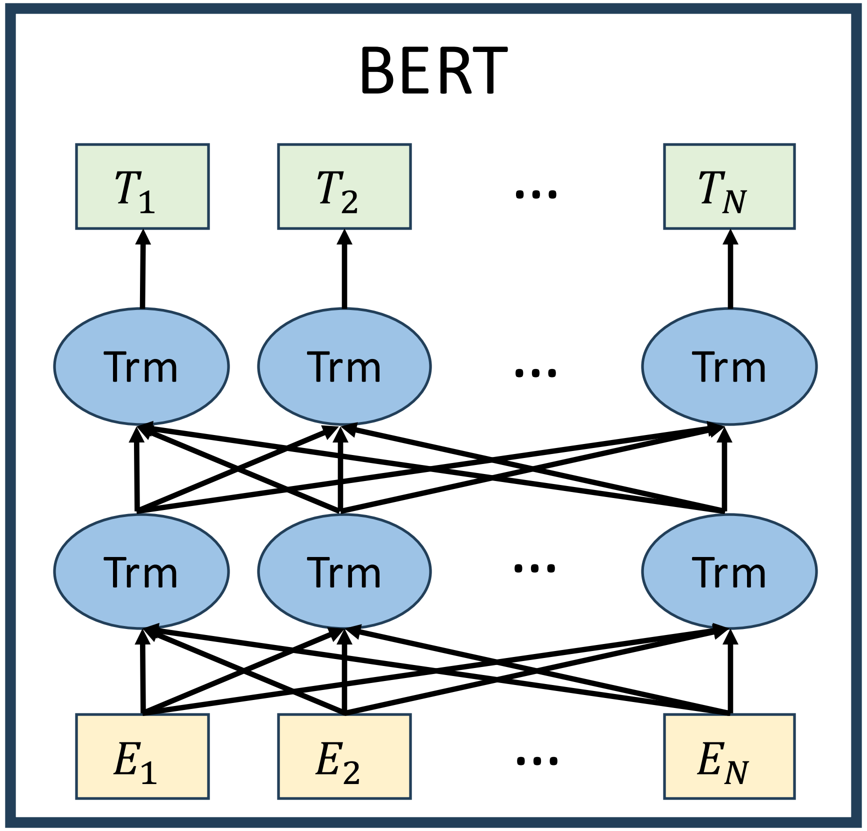
N input patches, each of shape 3x16x16

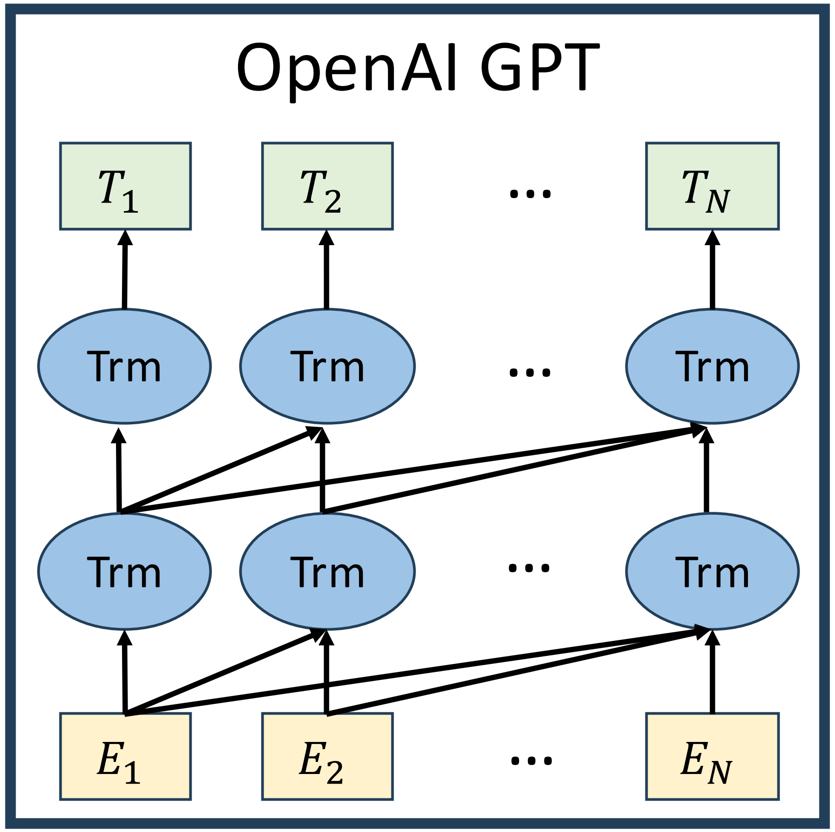


Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Pretraining Transformers: BERT Vs GPT





Parameter Efficient Fine-Tuning (PEFT)

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Full Fine-tuning in Foundational Models

Model Name	η_{params}	η_{layers}	d_{model}	η_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	$6.0\ X 10^{-4}$
GPT-3 Medium	350M	24	1024	16	64	0.5M	$3.0\ X\ 10^{-4}$
GPT-3 Large	760M	24	1536	16	96	0.5M	$2.5\ X\ 10^{-4}$
GPT-3 XL	1.3B	24	2048	24	128	IM	$2.0\ X\ 10^{-4}$
GPT-3 2.7B	2.7B	32	2560	32	80	IM	$1.6\ X\ 10^{-4}$
GPT-3 6.7B	6.7B	32	4096	32	128	2M	$1.2\ X\ 10^{-4}$
GPT-3 13B	13.0B	40	5140	40	128	2M	$1.0\ X\ 10^{-4}$
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	$0.6\ X\ 10^{-4}$

Brown et al. Language Models are Few-Shot Learners. NeurIPS 2020.
McCandlish et al. An Empirical Model of Large-Batch Training

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Full Fine-tuning in Foundational Models

Temp Memory

Forward Activations

Gradients

Optimizer States

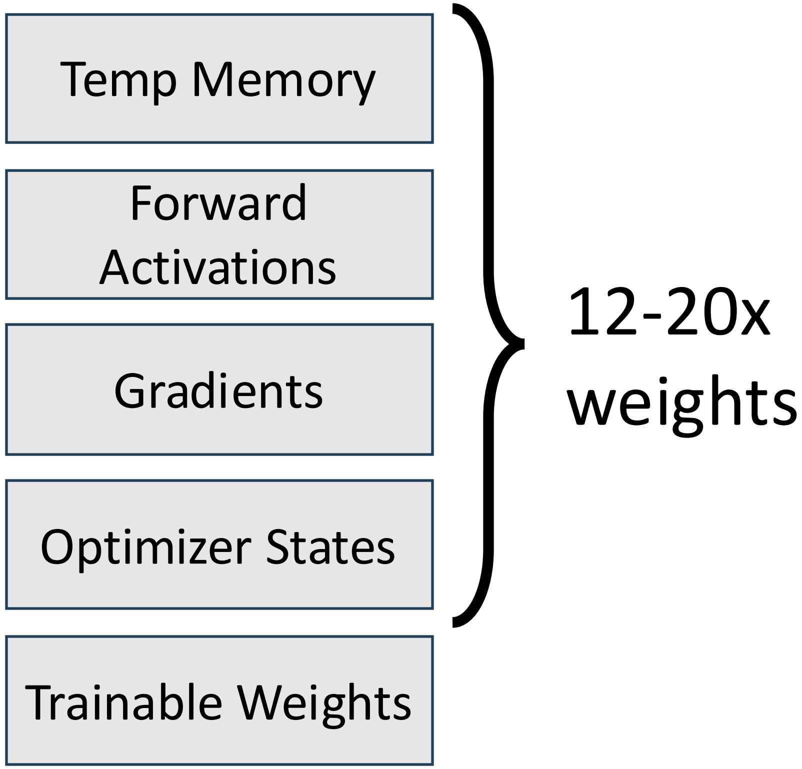
Trainable Weights

12-20x weights

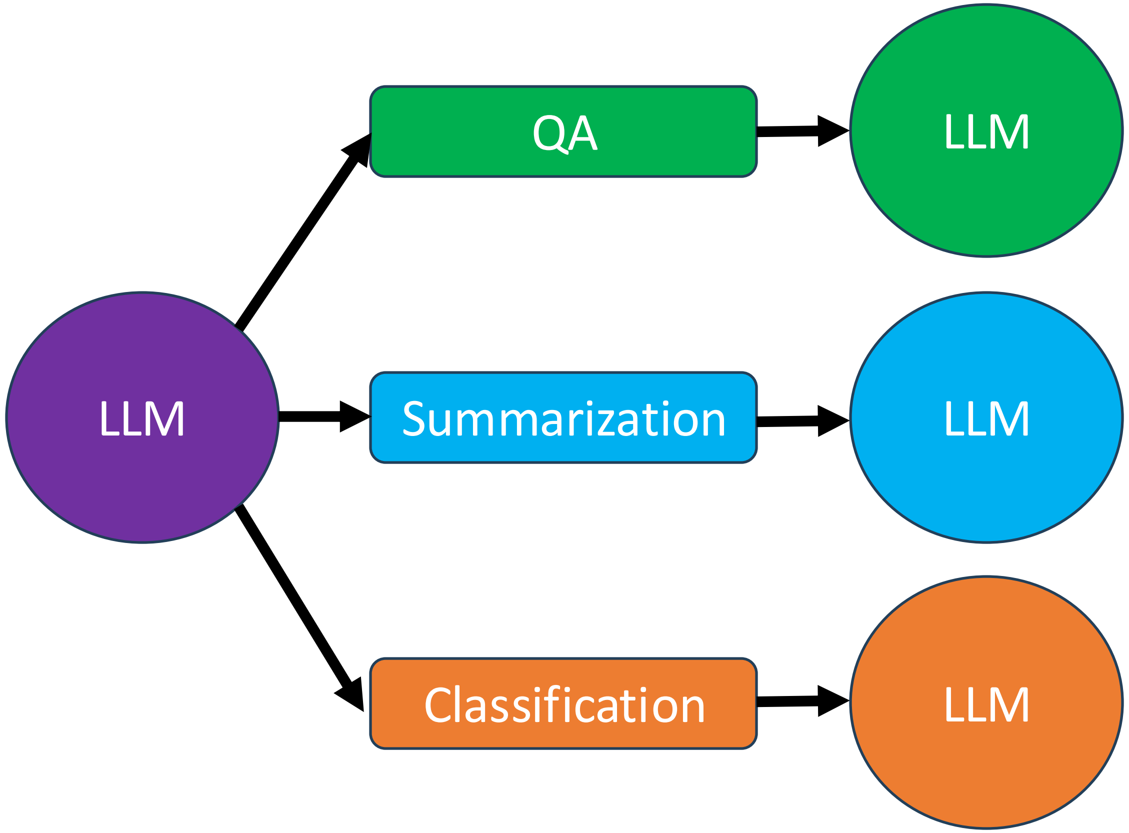
1. Hardware Requirements

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Full Fine-tuning in Foundational Models



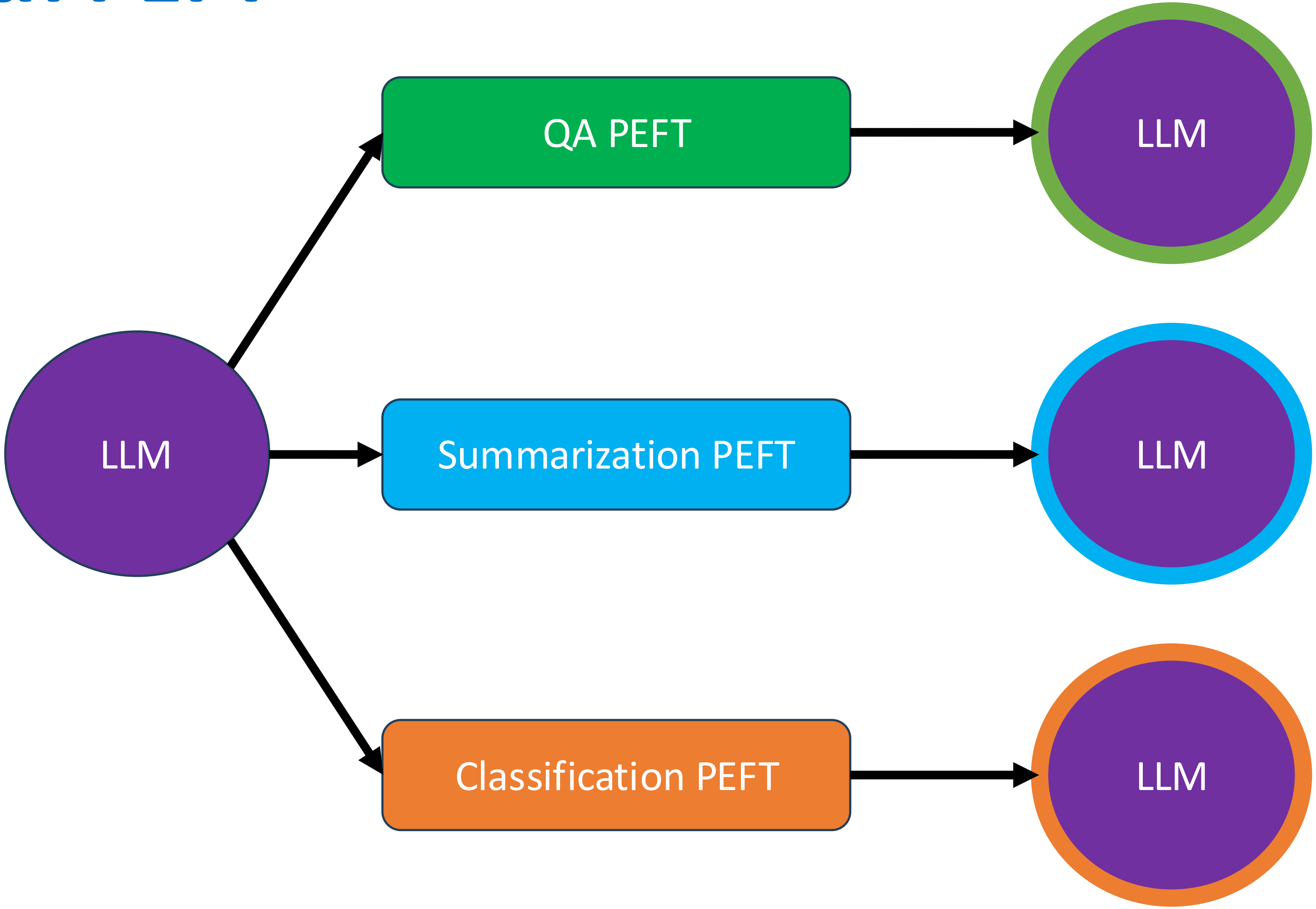
1. Hardware Requirements



2. Storage

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

With PEFT



Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

PEFT Benefits

- **Reduced computational costs**
  - Requires fewer GPUs and GPU time
- **Lower hardware requirements**
  - Works with smaller GPUs & less memory
- **Better modelling performance**
  - Reduces overfitting by preventing catastrophic forgetting
- **Less storage**
  - Majority of weights can be shared across different tasks

Prompts

- Prompts include instructions and, optionally, examples (latter called “In-Context Learning”)

- **Zero Shot:** The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
Translate English to French:

cheese =>
```

Brown et al. Language Models are Few-Shot Learners. NeurIPS 2020.

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Prompts

- Prompts include instructions and, optionally, examples (latter called "In-Context Learning")

- **Single Shot:** In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
Translate English to French:

sea otter => loutre de mer

cheese =>
```

Brown et al. Language Models are Few-Shot Learners. NeurIPS 2020.

Prompts

- Prompts include instructions and, optionally, examples (latter called "In-Context Learning")

- **Few Shot:** In addition to the task description, the model sees a **few** examples of the task. No gradient updates are performed.

```
Translate English to French:
sea otter => loutre de mer
peppermint => menthe poivre
plush giraffe => giraffe peluche
cheese =>
```

Brown et al. Language Models are Few-Shot Learners. NeurIPS 2020.

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

What Prompts to Use?

- Chain-of-thought (COT) prompting can help by guiding model to show its intermediate reasoning steps!

Standard Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

The answer is 27.

incorrect

Chain of Thought Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9.

Challenge: What Prompts to Use?

Why COT prompting works? Examples may reveal the target output format as performance still improves with invalid examples; e.g.

COT	Originally, Leah had 32 chocolates and her sister had 42. So in total they had 32 + 42 = 74. After eating 35, they had 74 - 35 = 39 pieces left in total. The answer is 39.	Julie is reading a 120-page book. Yesterday, she read 12 pages and today, she read 24 pages. So she read a total of 12 + 24 = 36 pages. Now she has 120 - 36 = 84 pages left. Since she wants to read half of the remaining pages, she should read 84 / 2 = 42 pages. The answer is 42
Invalid Reasoning **Yet, correct answer**	Originally, Leah had 32 chocolates, and her sister had 42. So her sister had 42 – 32 = 10 chocolates more than Leah had. After eating 35, since 10 + 35 = 45, they had 45 – 6 = 39 pieces left in total. The answer is 39.	Yesterday, Julie read 12 pages. Today, she read 12 * 2 = 24 pages. So she read a total of 12 + 24 = 36 pages. Now she needs to read 120 - 36 = 84 more pages. She wants to read half of the remaining pages tomorrow, so she needs to read 84/2 = 42 pages tomorrow. The answer is 42.

Brown et al. Language Models are Few-Shot Learners. Neurips 2020.

PEFT Techniques

- P-Tuning
- Prefix Tuning
- Adapters
- Low Rank Adaptation

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

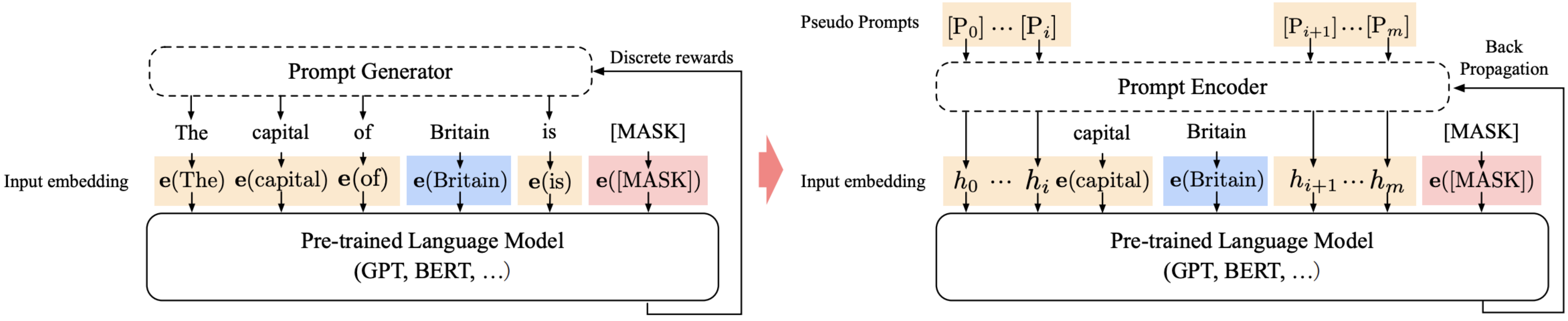
PEFT Techniques

- P-Tuning
- Prefix Tuning
- Adapters
- Low Rank Adaptation

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

P-Tuning

- Appends a trainable tensor to the model's input embeddings, creating a soft prompt.
- The model weights of the LLM are frozen.
- In contrast to the regular (hard) prompt tuning, in (soft) prompt tuning the prompts are vectors instead of discrete prompts.



(a) Discrete Prompt Search (b) P-tuning

Liu et al. GPT Understands, Too. CoRR abs/2103.10385. 2021

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

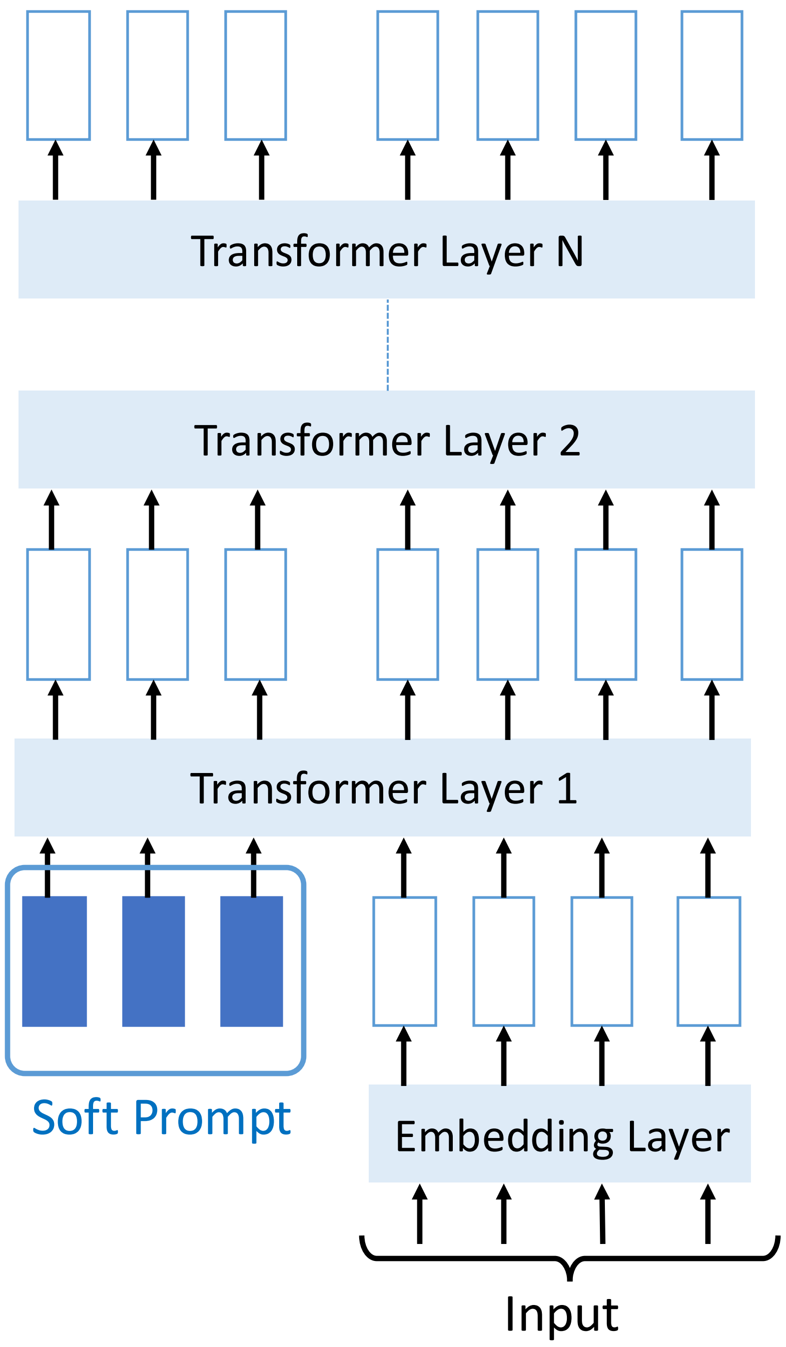
P-Tuning

- Treats prompt as a set of learnable parameters that are updated by backpropagation.

- For a specific task, only a small task-specific soft prompt needs to be stored

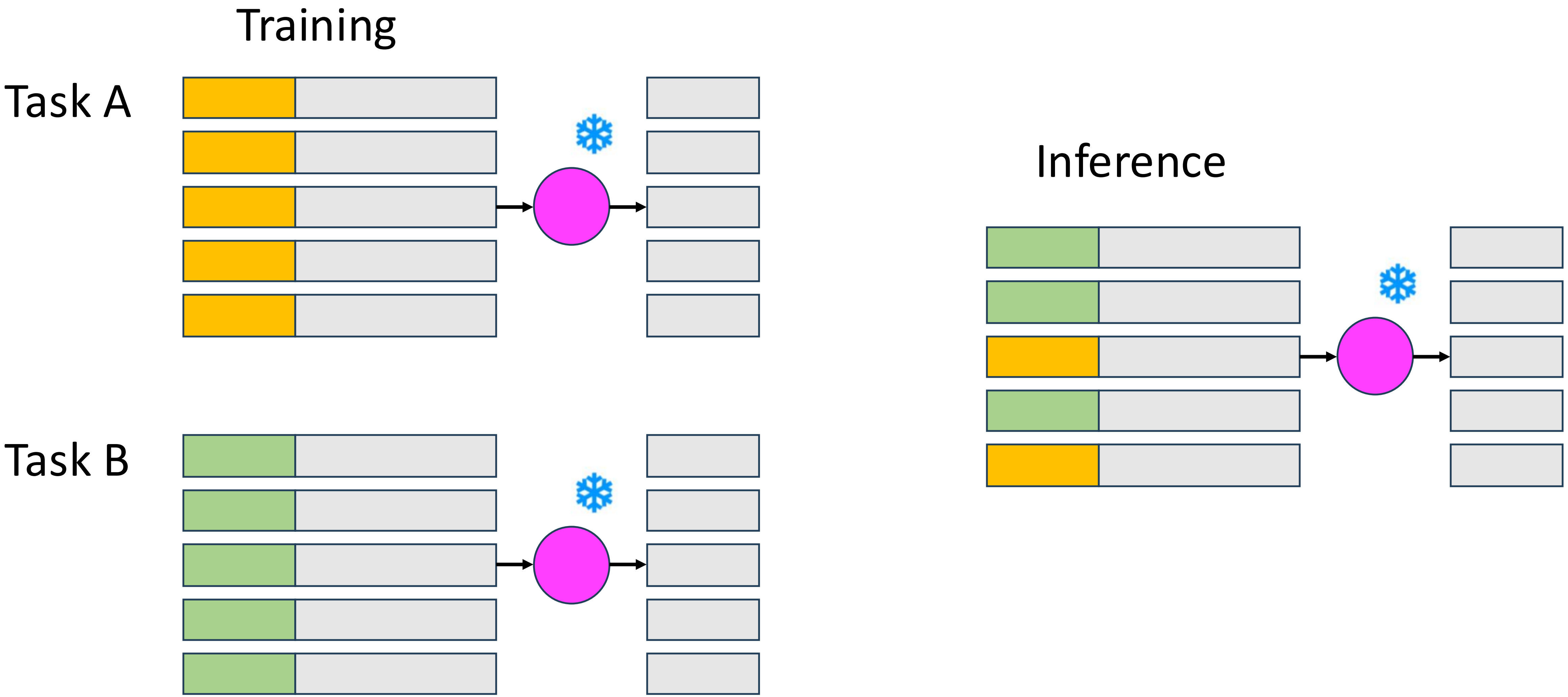
- Significantly more parameter-efficient than full-finetuning

- Additionally, a prompt encoder can also be used which can be an LSTM or a Multi-Layer Perceptron.



Lester et al., The Power of Scale for Parameter-Efficient Prompt Tuning. EMNLP 2021.

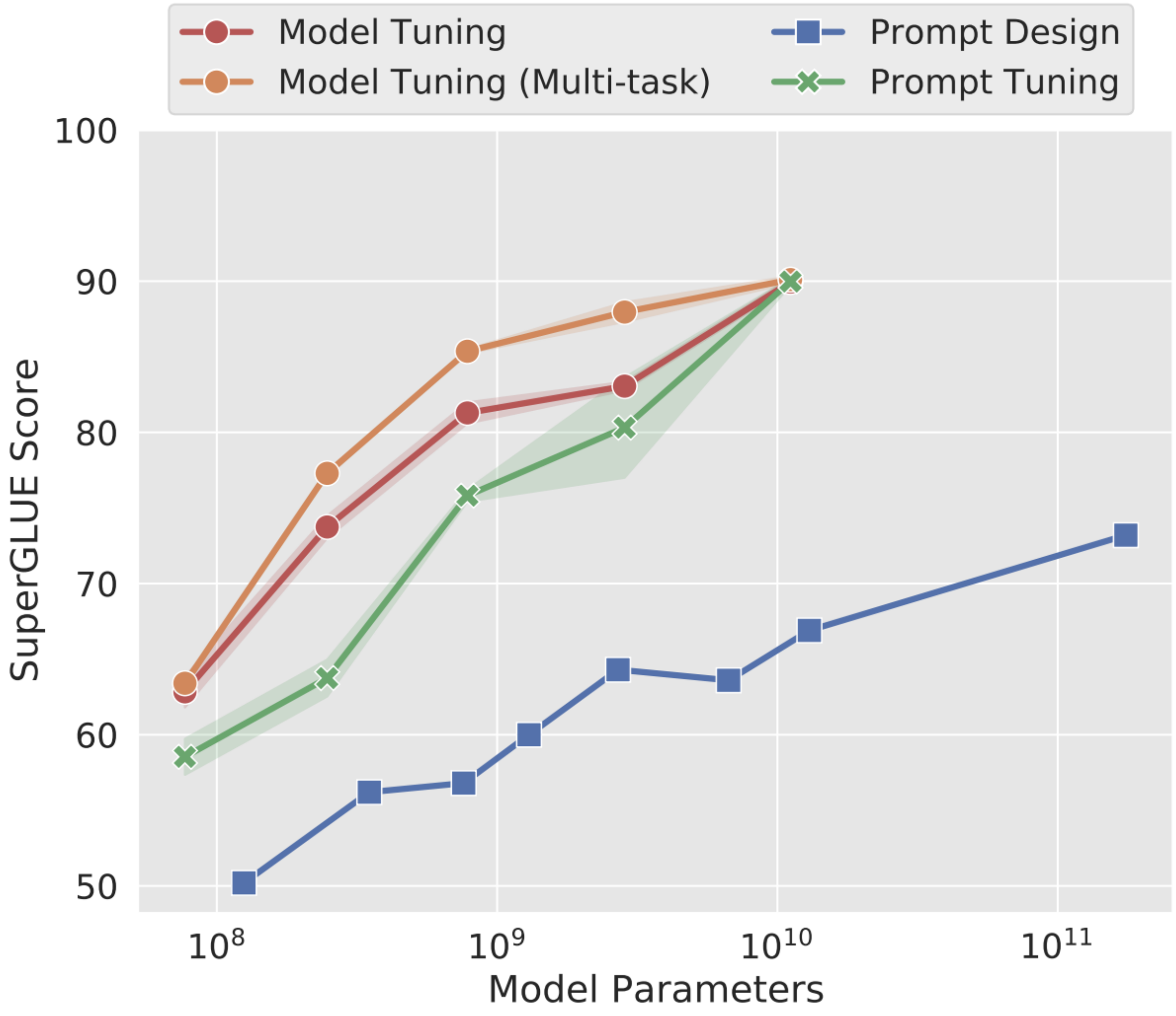
P-Tuning



Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

(Soft) Prompt Tuning: Pros and Cons

- May perform poorly at smaller model sizes and on harder tasks.
- Increasing prompt length improves the performance but increasing beyond 20 tokens may only yield marginal gains.



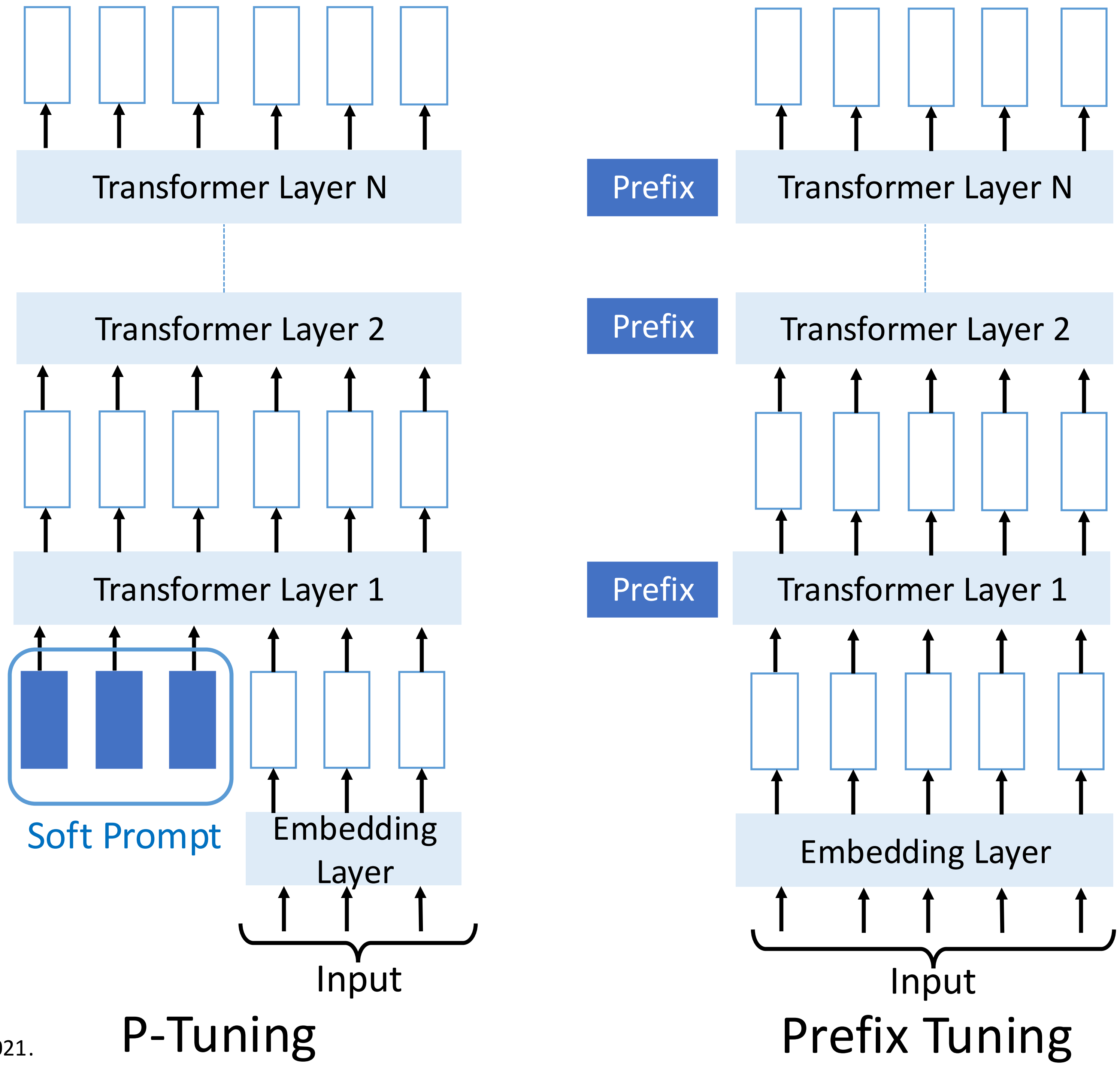
PEFT Techniques

- P-Tuning
- Prefix Tuning
- Adapters
- Low Rank Adaptation

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Prefix Tuning

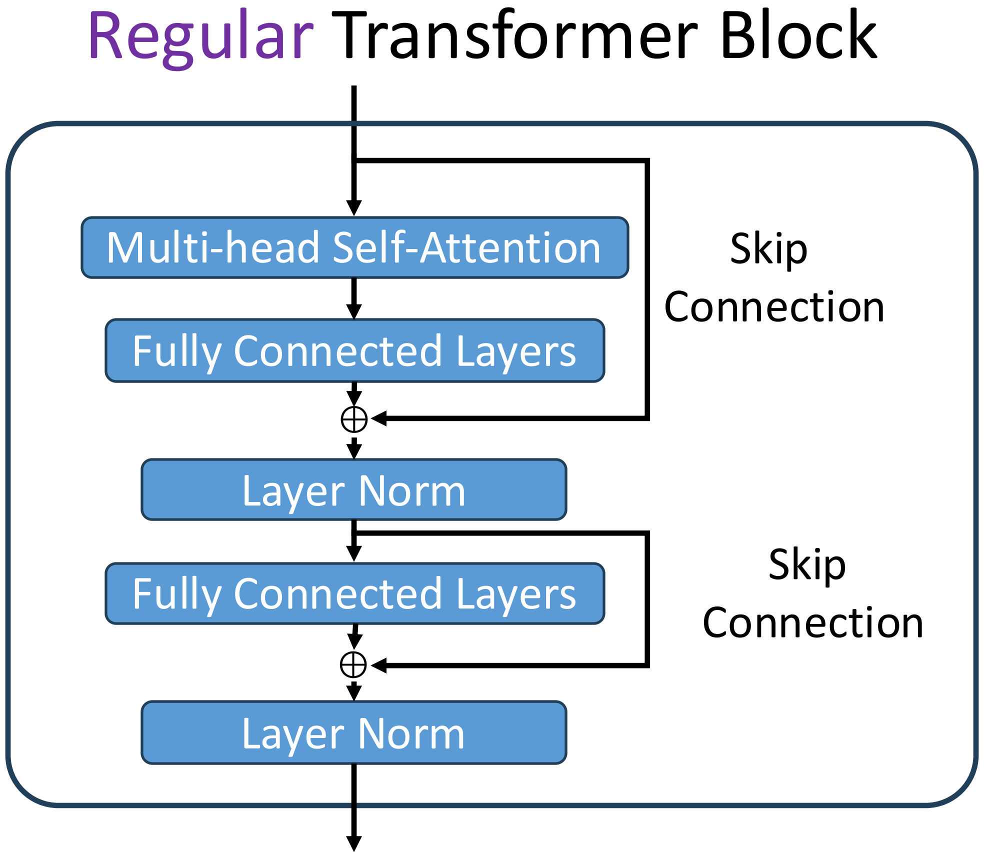
- Add a trainable tensor to **<u>each</u>** transformer block instead of only the input embeddings, as in soft prompt tuning.
- Add learnable component to each K/V vectors.



Li and Liang, Prefix-Tuning: Optimizing Continuous Prompts for Generation. 2021.

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Prefix Tuning



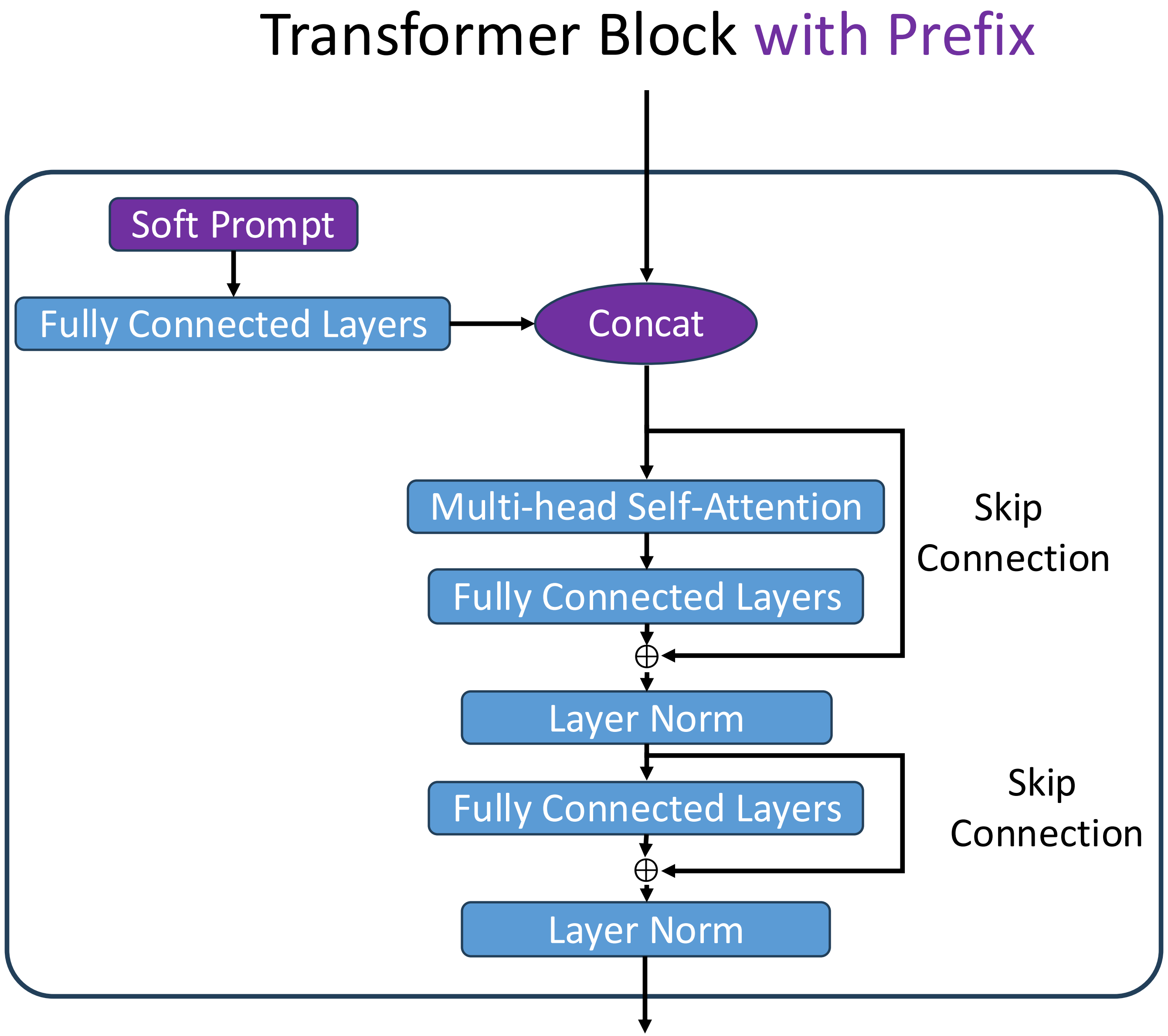


Image Credit: https://sebastianraschka.com/blog/2023/llm-finetuning-llama-adapter.html

PEFT Techniques

- P-Tuning
- Prefix Tuning
- Adapters
- Low Rank Adaptation

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Adapter

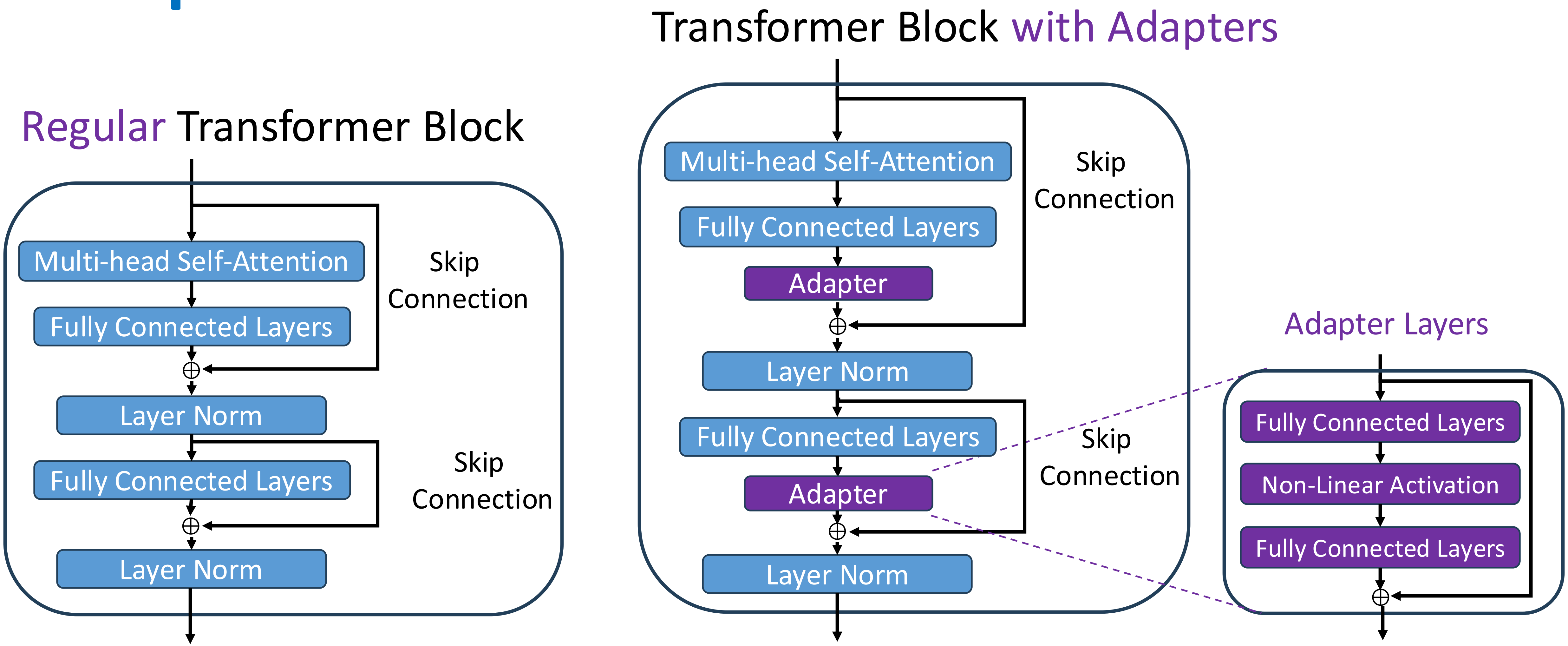
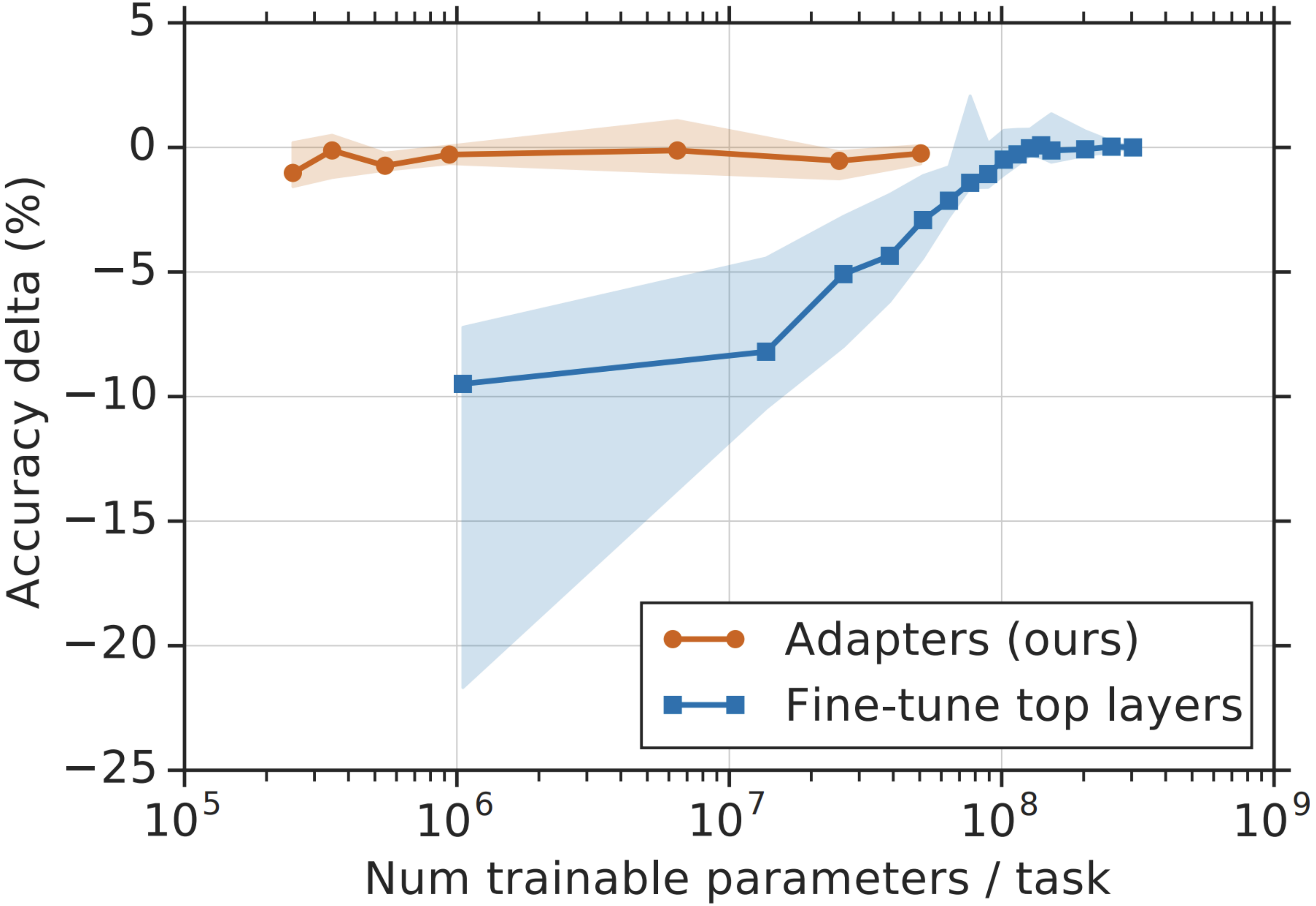


Image Credit: https://sebastianraschka.com/blog/2023/llm-finetuning-llama-adapter.html

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Adapter

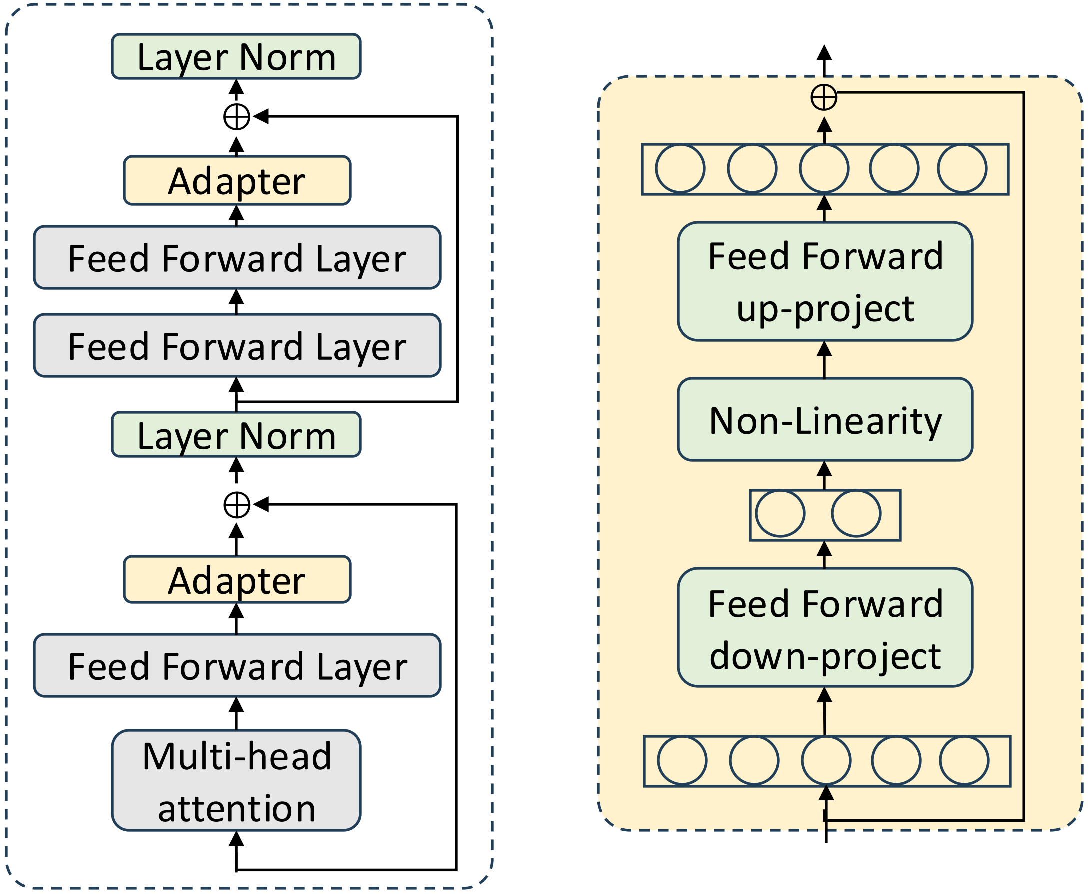


Houlsby et al., Parameter-Efficient Transfer Learning for NLP. 2019.

Adapter: Architecture

Bottleneck Structure

- Reduces the number of parameters
- Reduces d-dimensional features into a smaller m-dimensional vector
  - Example: $d = 1024$ and $m = 24$
  - 1024×1024 requires 1,048,576 parameters
  - $2 \times (1024 \times 24)$ requires 49,152 parameters



- m determines the number of optimizable parameters and hence poses a parameter vs performance tradeoff.

Houlsby et al., Parameter-Efficient Transfer Learning for NLP. 2019.

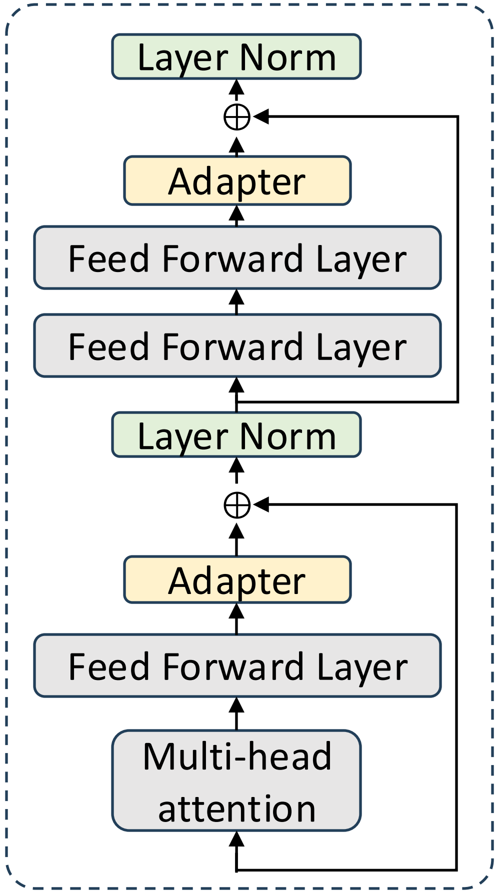
Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

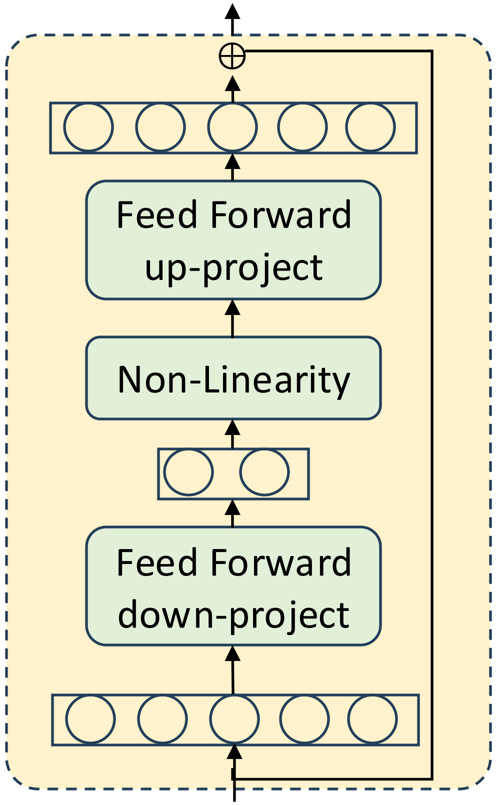
Adapter: Architecture

Inference Overhead

- Additional adapter in each transformer layer increases the inference latency

- Unlike Prompt tuning, same pre-trained model can't be used when fine-tuned with an adapter layer.



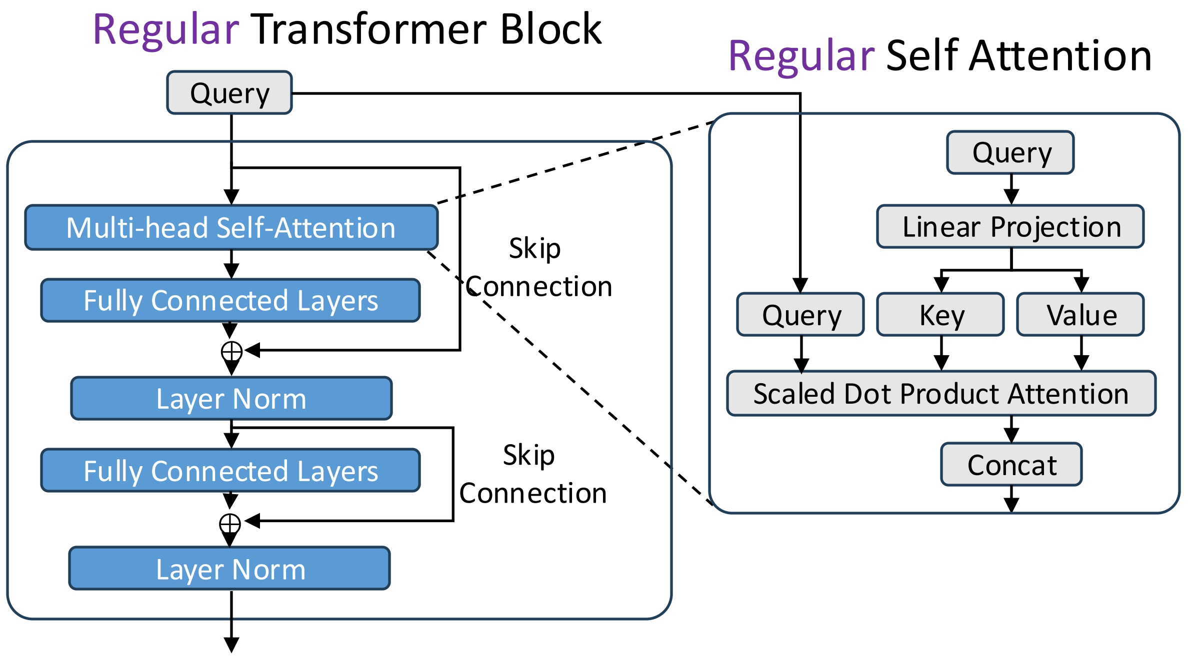


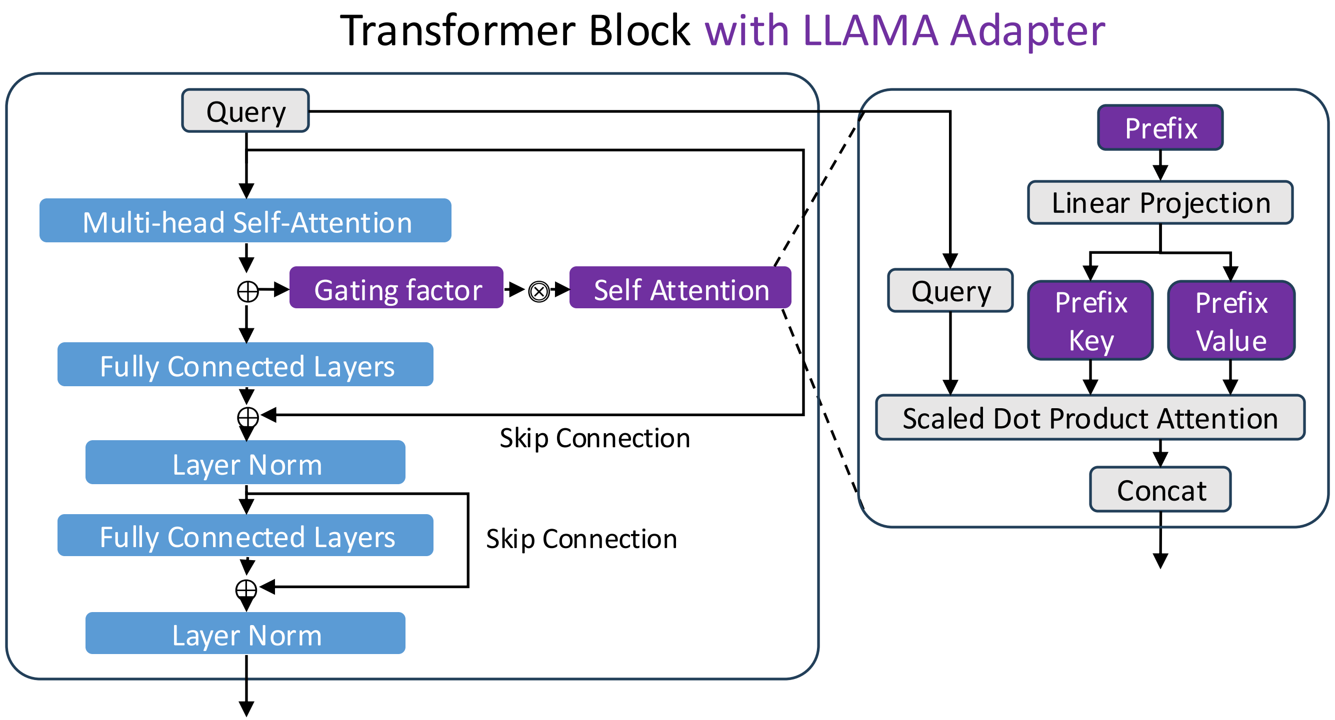
Houlsby et al., Parameter-Efficient Transfer Learning for NLP. 2019.

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Example: Llama Adapter

- Prepends tunable prompt tensors to the embedded inputs.
- The prefix is learned and maintained within an embedding table rather than being provided externally.
- Each transformer block in the model has its own distinct learned prefix, allowing for more tailored adaptation across different model layers.

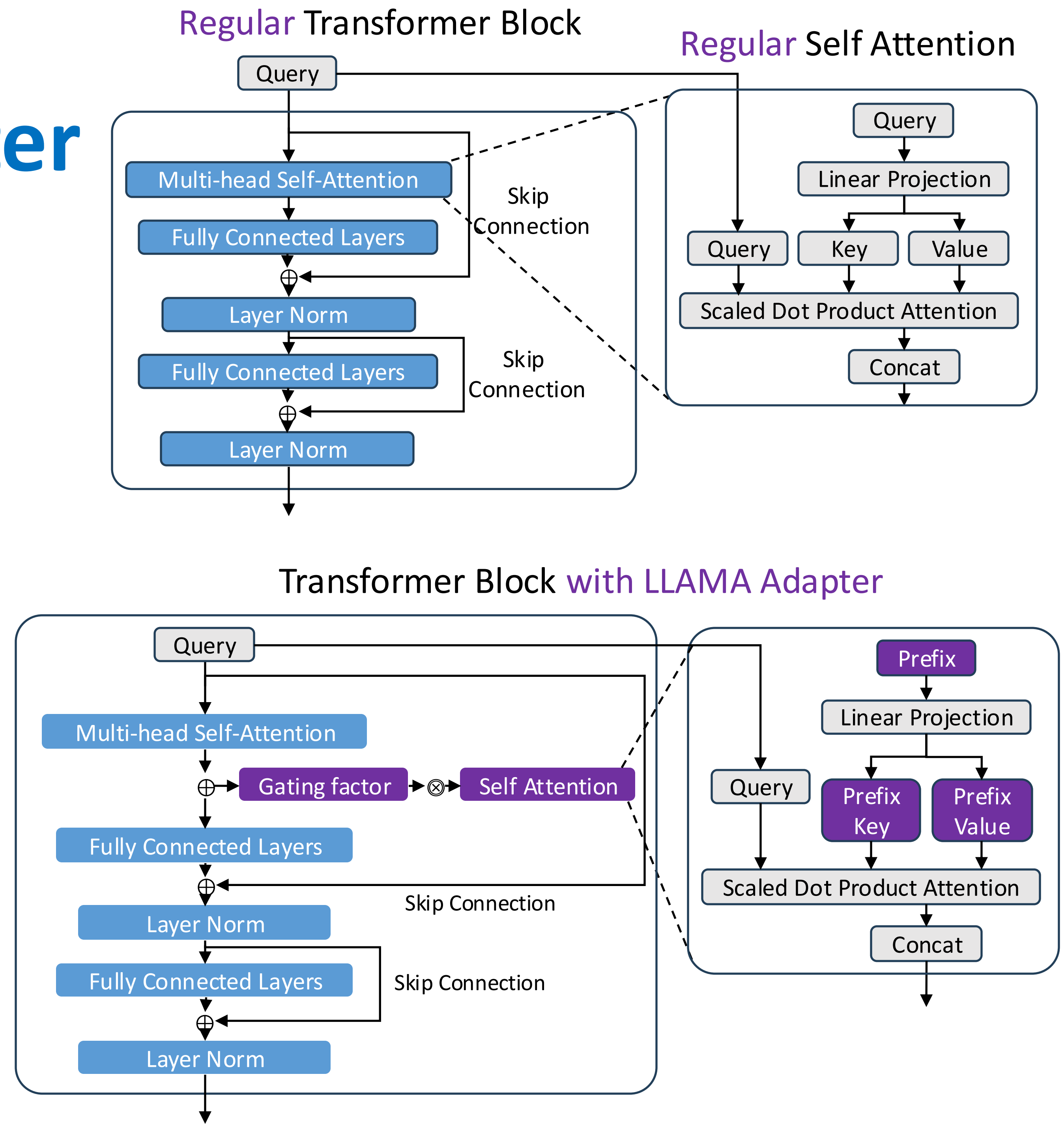




Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Example: Llama Adapter

- Introduces a zero-initialized attention mechanism coupled with gating.
- Prevents adapters and prefix tuning from potentially disrupting the linguistic knowledge of the pretrained LLM during initial training phases.
- Adds the learnable adaption prompts only to the L topmost transformer layers instead of all transformer layers.

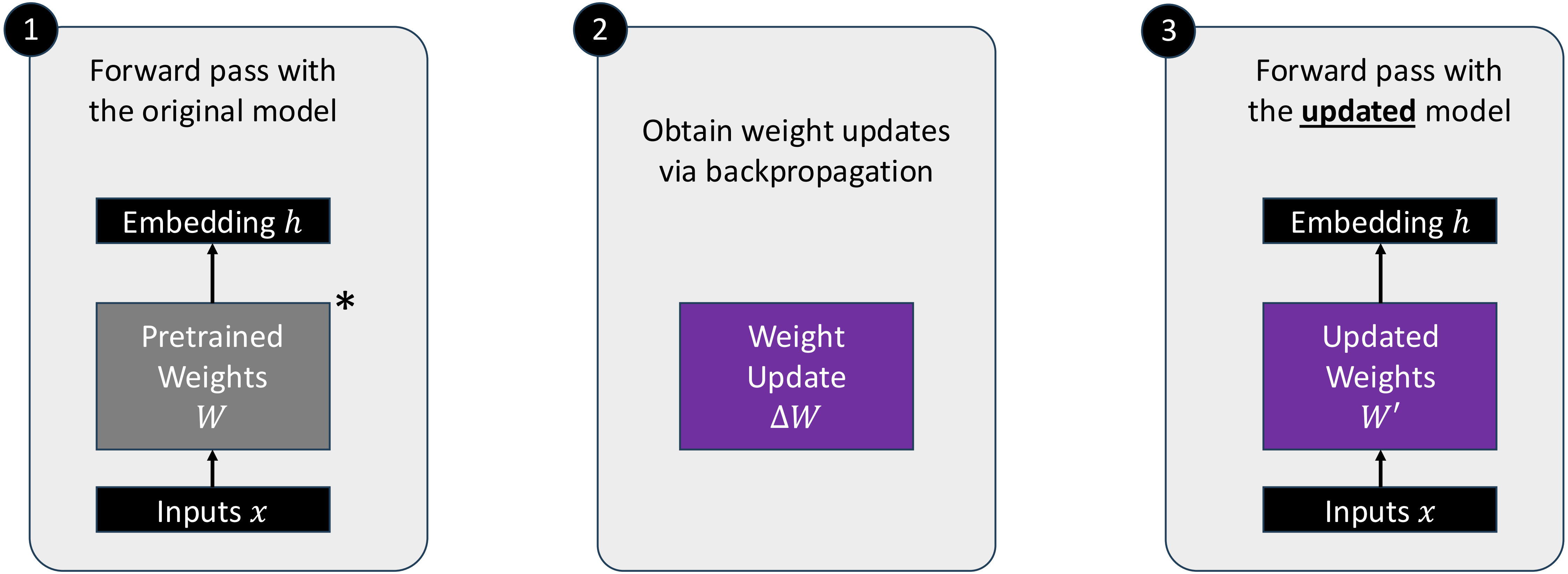


PEFT Techniques

- P-Tuning
- Prefix Tuning
- Adapters
- Low Rank Adaptation

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Regular Finetuning

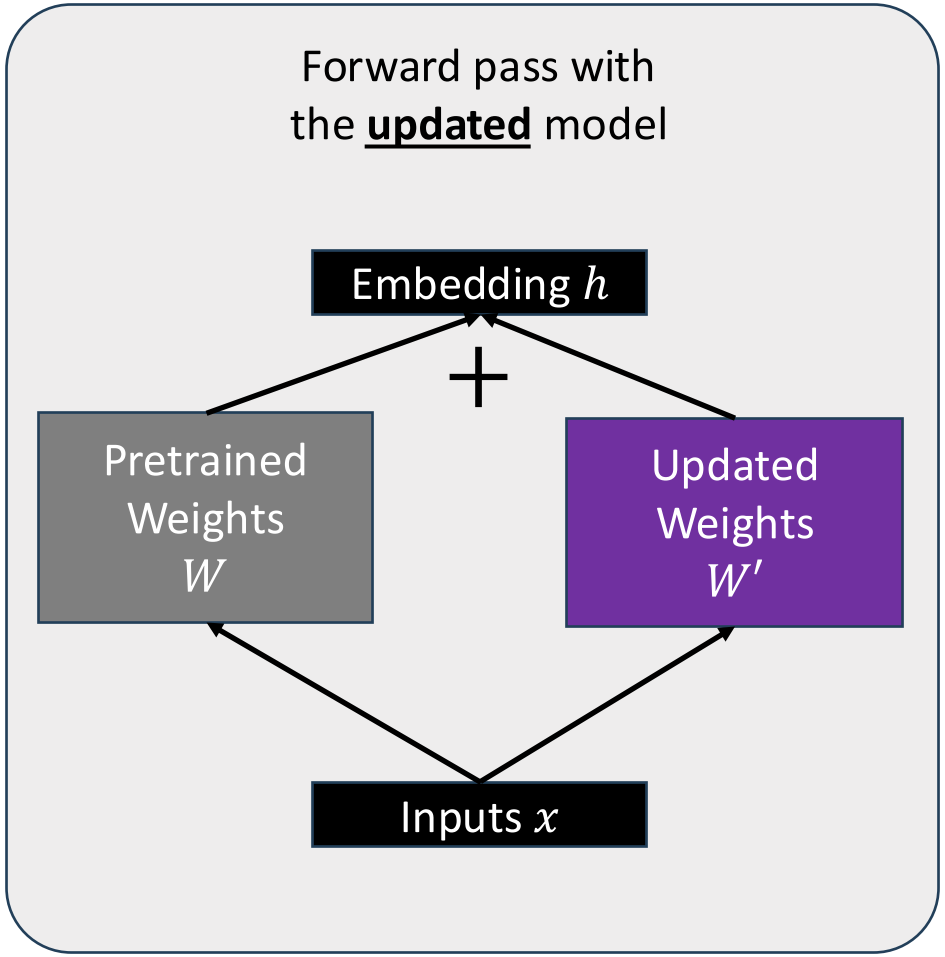


* The pretrained model could be any LLM, e.g. an encoder-style LLM (Like BERT) or a generative decoder-style LLM (like GPT)

https://sebastianraschka.com/blog/2023/llm-finetuning-lora.html

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Regular Finetuning: Alternate Visualization



https://sebastianraschka.com/blog/2023/llm-finetuning-lora.html

Intrinsic Dimension

- While the weights of a pretrained model have full rank on the pretrained tasks, pretrained large language models have a low "intrinsic dimension" when they are adapted to a new task.

- By optimizing only 200 trainable parameters randomly projected back into the full space, one can tune a RoBERTa model to achieve 90% of the full parameter performance.

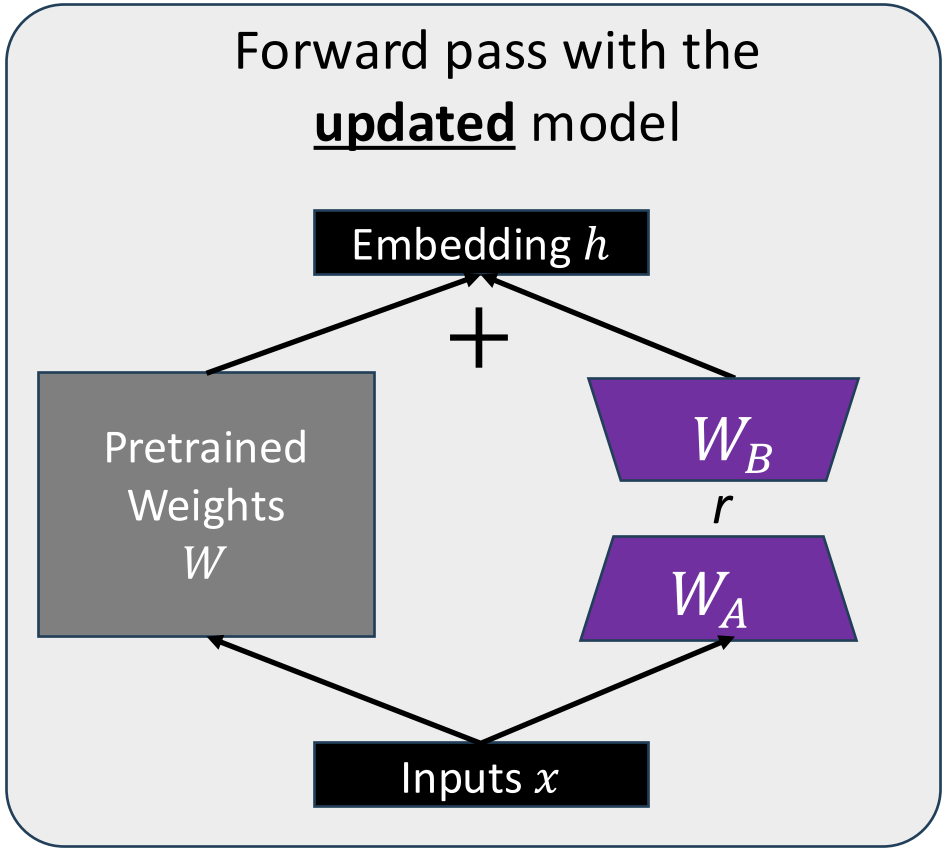
- **Intrinsic dimension of a task**: Minimum dimension/number-of-parameters where a model achieves within 90% of the full-parameter model performance

Aghajanyan et al., Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning. 2020.

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Low Rank Adaptation (LoRA)

LoRA weights W_A and W_B represent ΔW



$$h = W_0 x + \Delta W x = W_0 x + BAx$$

Learns two low-rank matrices A and B that are applied to the self-attention weights

Rank r is a hyperparameter that is used to specify the rank of the low-rank matrices used for adaptation

https://sebastianraschka.com/blog/2023/llm-finetuning-lora.html

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

LoRA: Choosing Rank

Smaller Rank r

- Simpler low-rank matrix, and fewer parameters to learn during adaptation.
- Faster training and reduced computational requirements.
- Decreased capacity of the low-rank matrix to capture task-specific information. Lower adaptation quality. Inferior performance

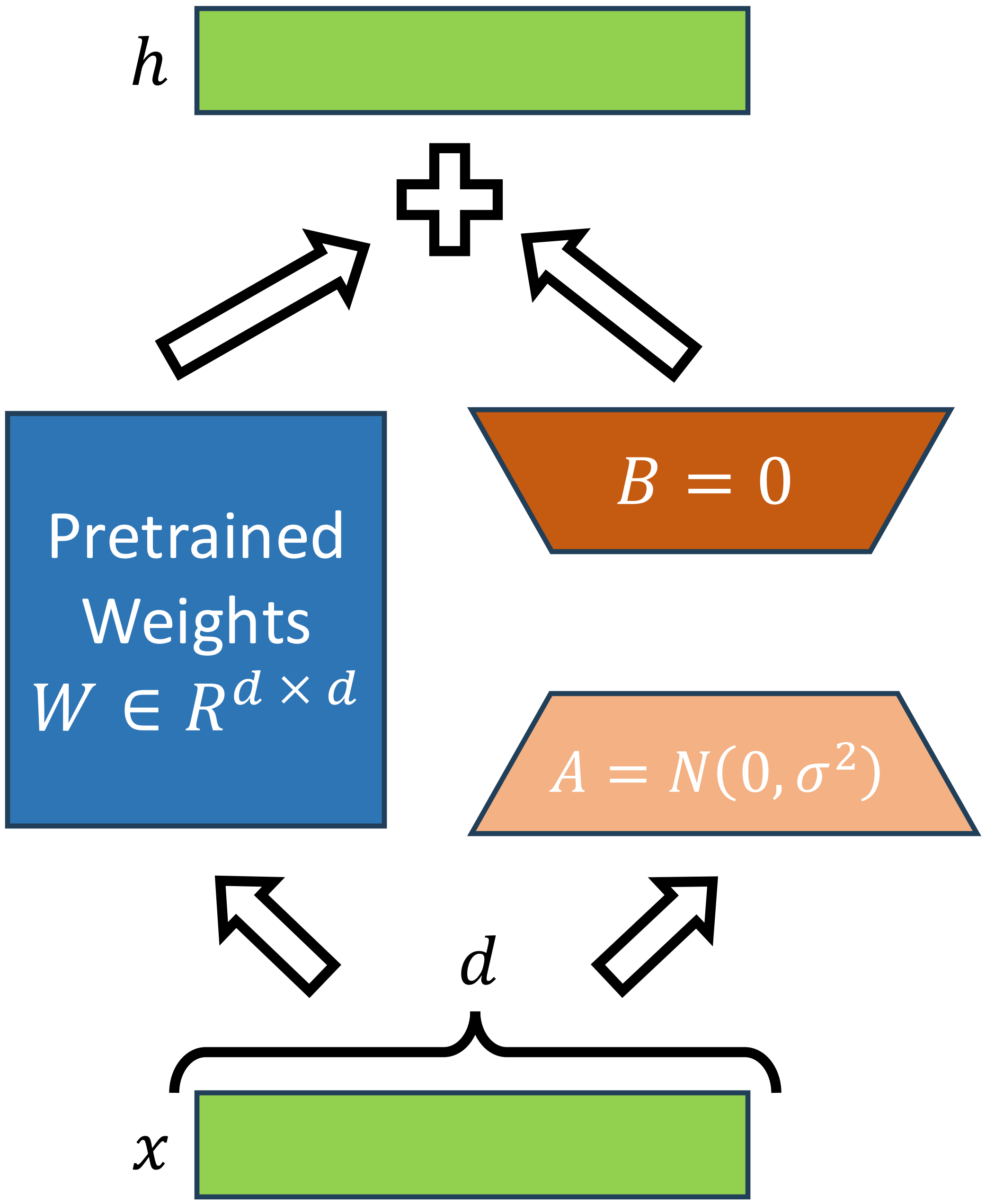
- Rank in LoRA represents trade-off between model complexity, adaptation capacity, and the risk of underfitting or overfitting.
- Important to experiment with different rank values to find the right balance to achieve the desired performance on the new task.

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

LoRA Weight Initialization

- By setting B to zero, the product $\Delta W = BA$ initially equals zero. This preserves the behavior of the original model at the start of fine-tuning

- Gaussian distribution helps ensure that the values in A are neither too large nor too biased in any direction, which could lead to disproportionate influence on the updates when B begins to change.



LoRA Variants

QLoRA [Dettmers et al., 2023]

- Backpropagates gradients through 4-bit quantized model for reducing memory usage.

LoRA+ [Hayou et al., 2024]

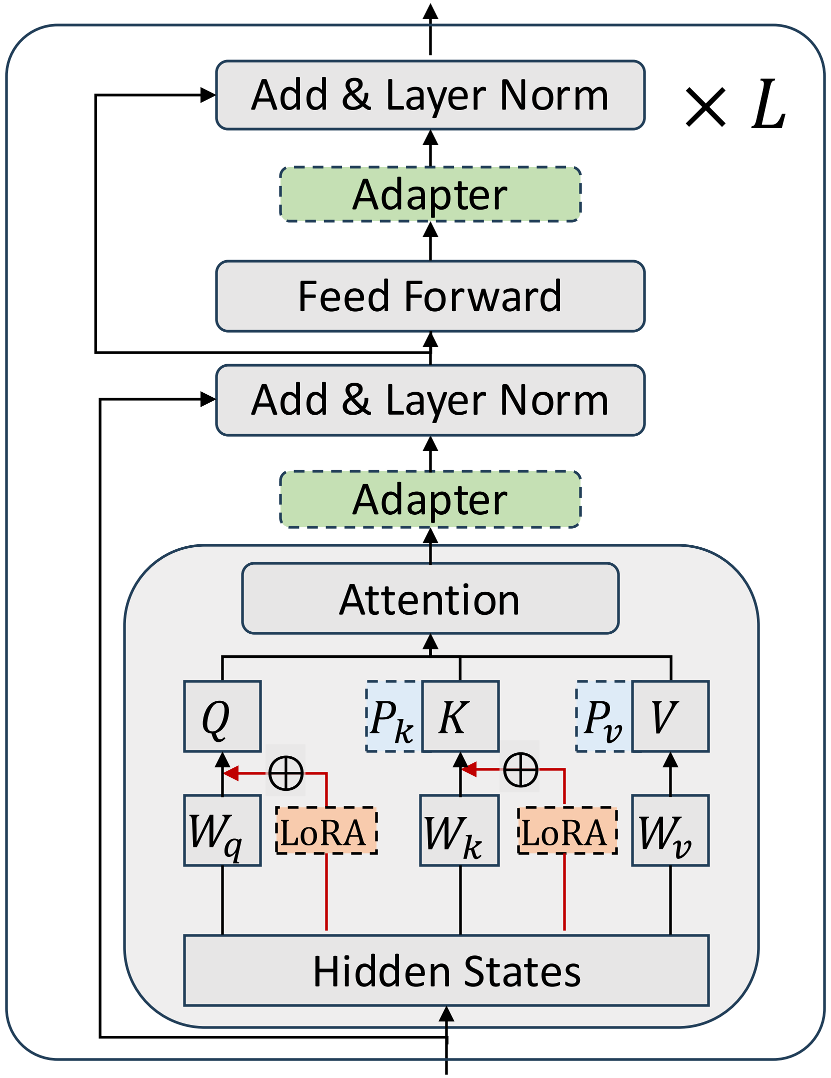
- Different learning rates for the LoRA adapter matrices A and B. Improves finetuning speed.

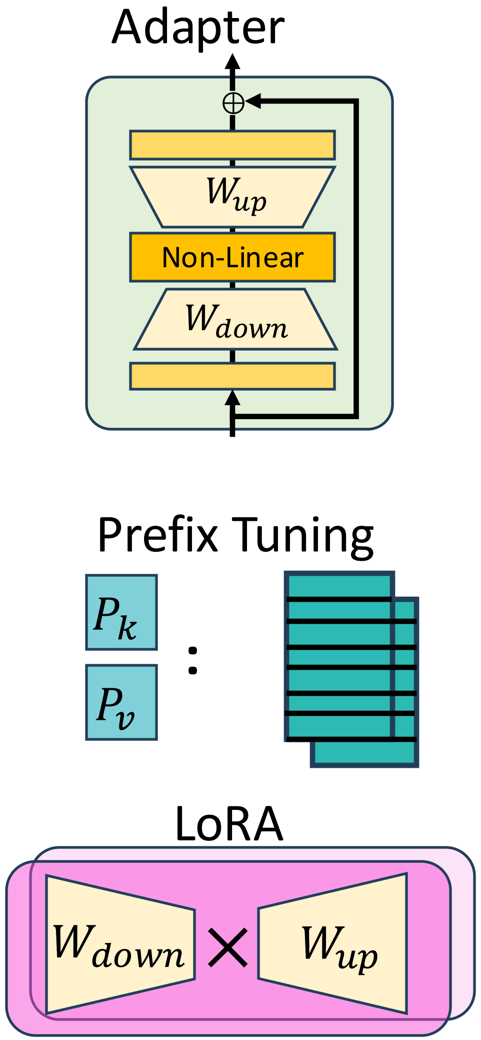
DyLoRA [Valipou et al., 2023]

- Selects rank without requiring multiple runs of training.

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Parameter Efficient Tuning: Summary





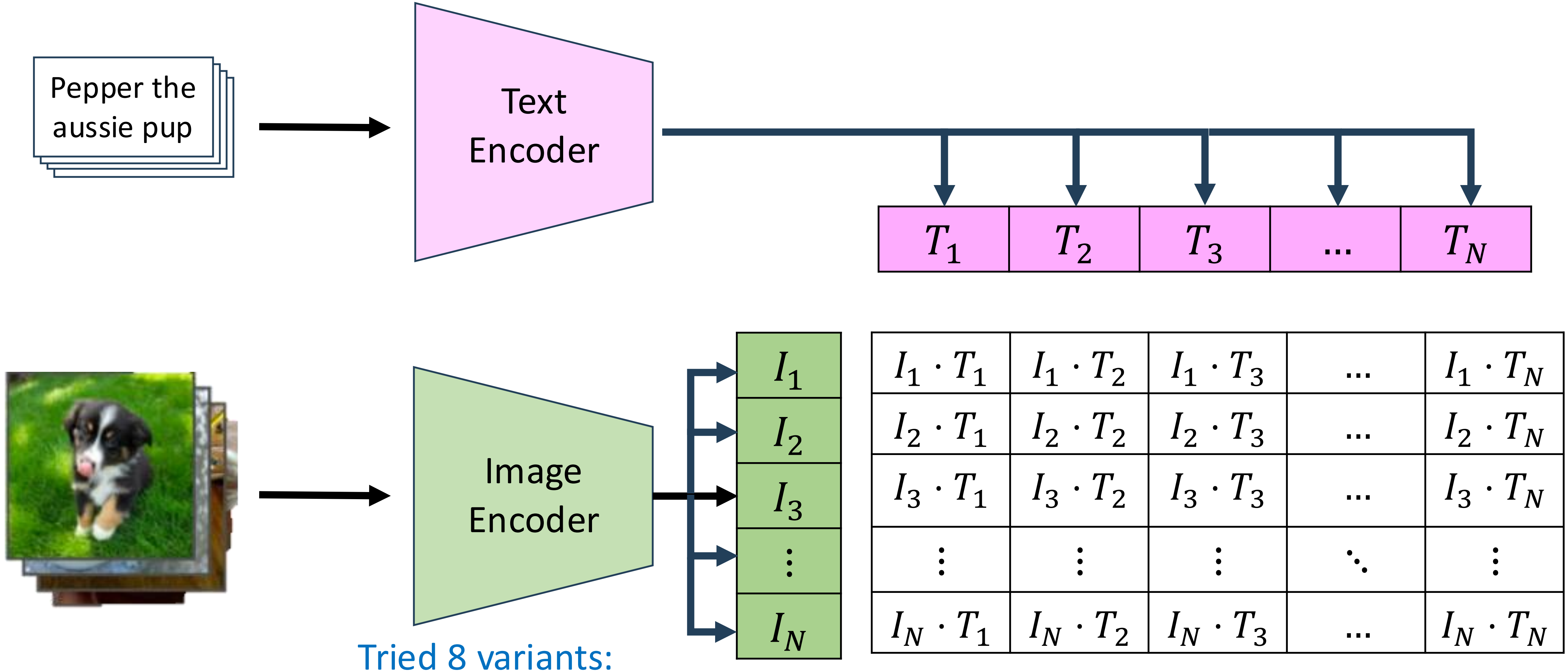
He et al. Towards a Unified View of Parameter-Efficient Transfer Learning. ICLR 2022

Computer Vision Applications

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Contrastive Language Image Pre-training (CLIP)

Text Transformer (GPT-2)

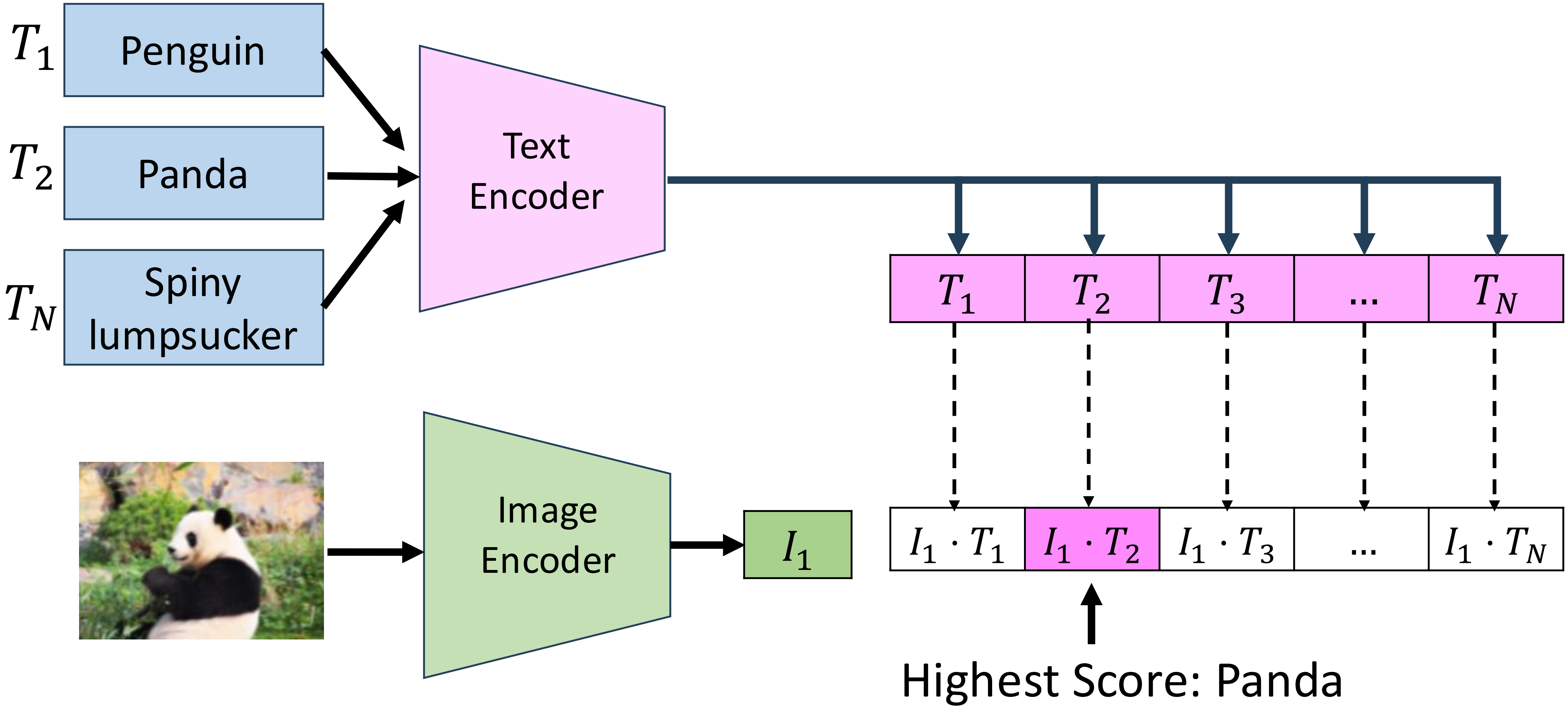


Tried 8 variants:
3 VIT and 5 ResNet

Radford et al. Learning Transferable Visual Models From Natural Language Supervision. ICML 2021.

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

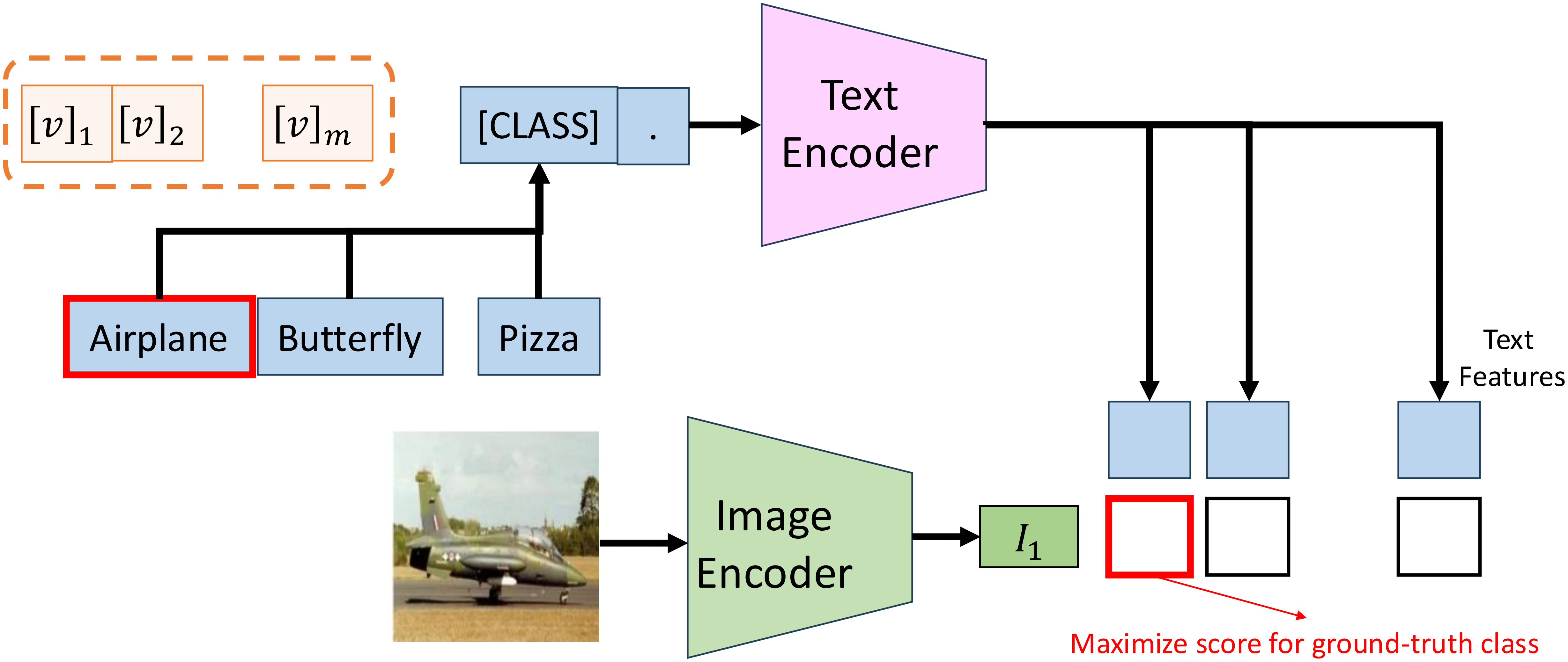
CLIP Inference



https://towardsdatascience.com/understanding-zero-shot-learning-making-ml-more-human-4653ac35ccab

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

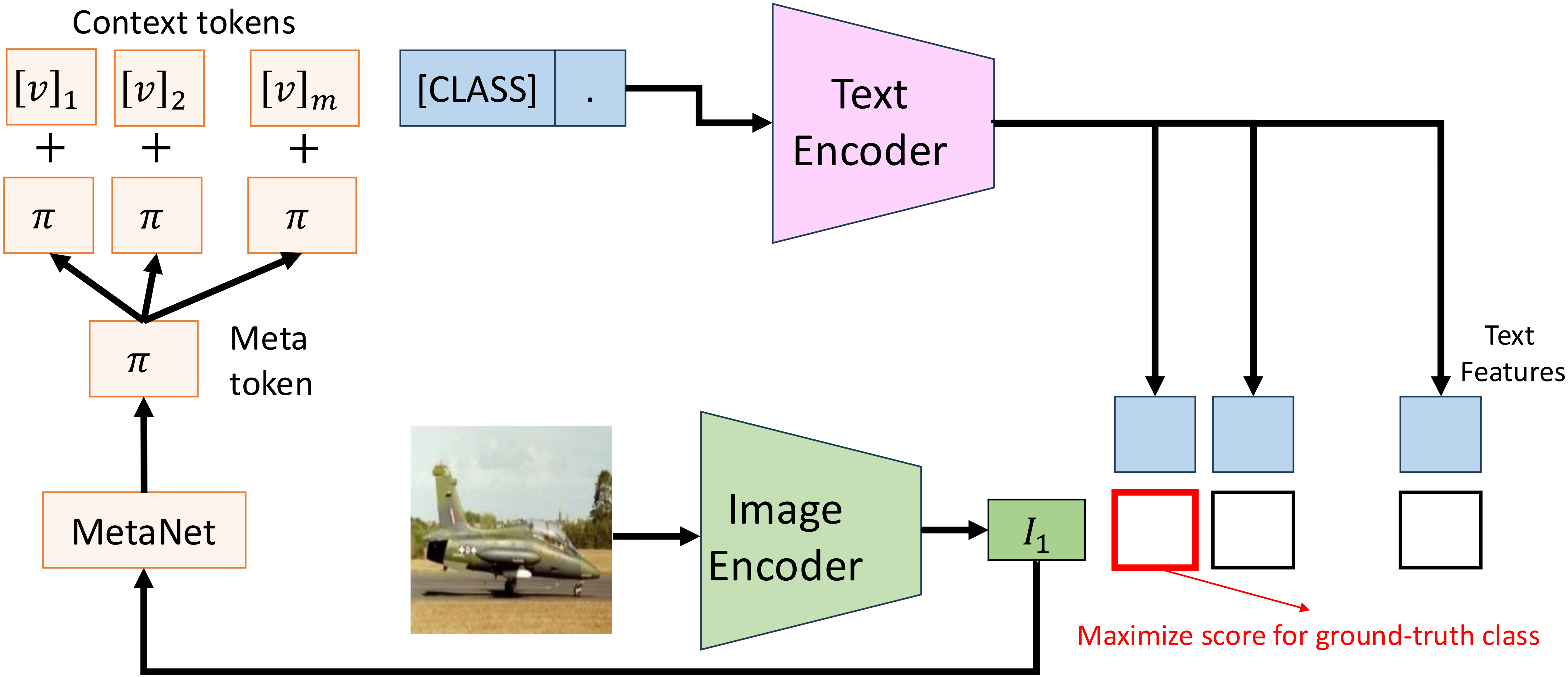
Context Optimization (CoOp)



Zhou et al., Learning to Prompt for Vision-Language Models. 2021.

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

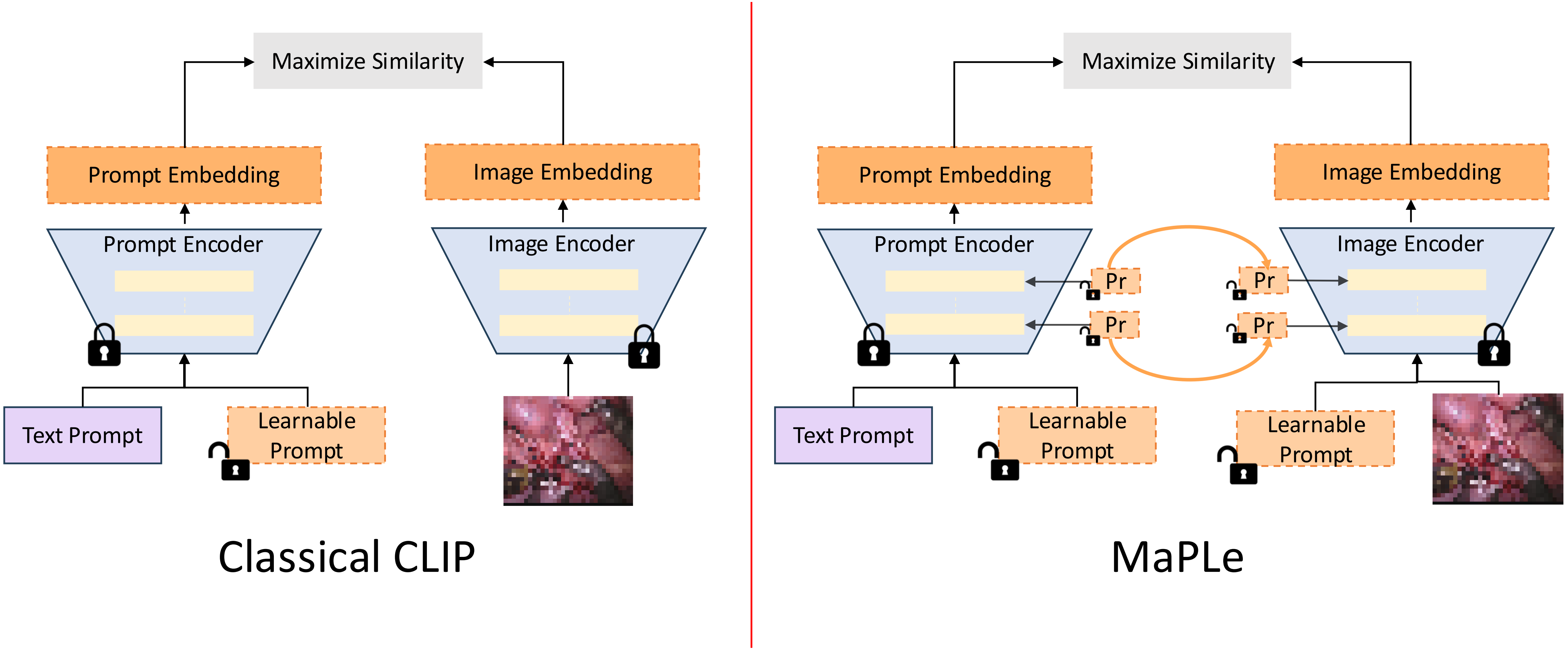
Conditional Context Opt. (CoCoOp)



Zhou et al., Conditional Prompt Learning for Vision-Language Models. CVPR 2022

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Multi-modal Prompt Learning (MaPLe)

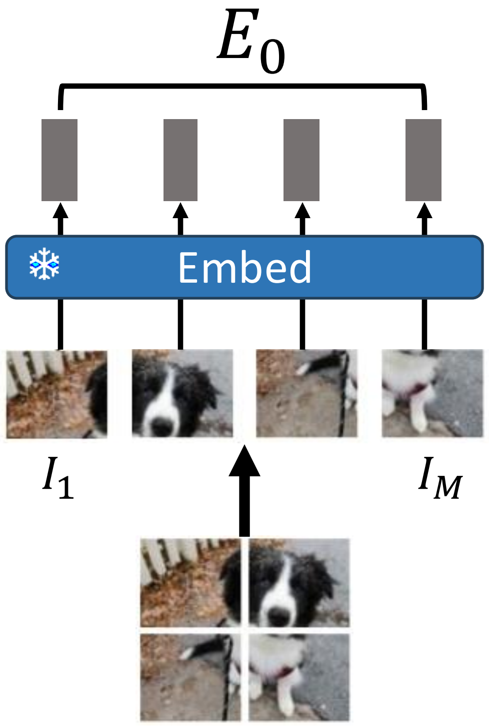


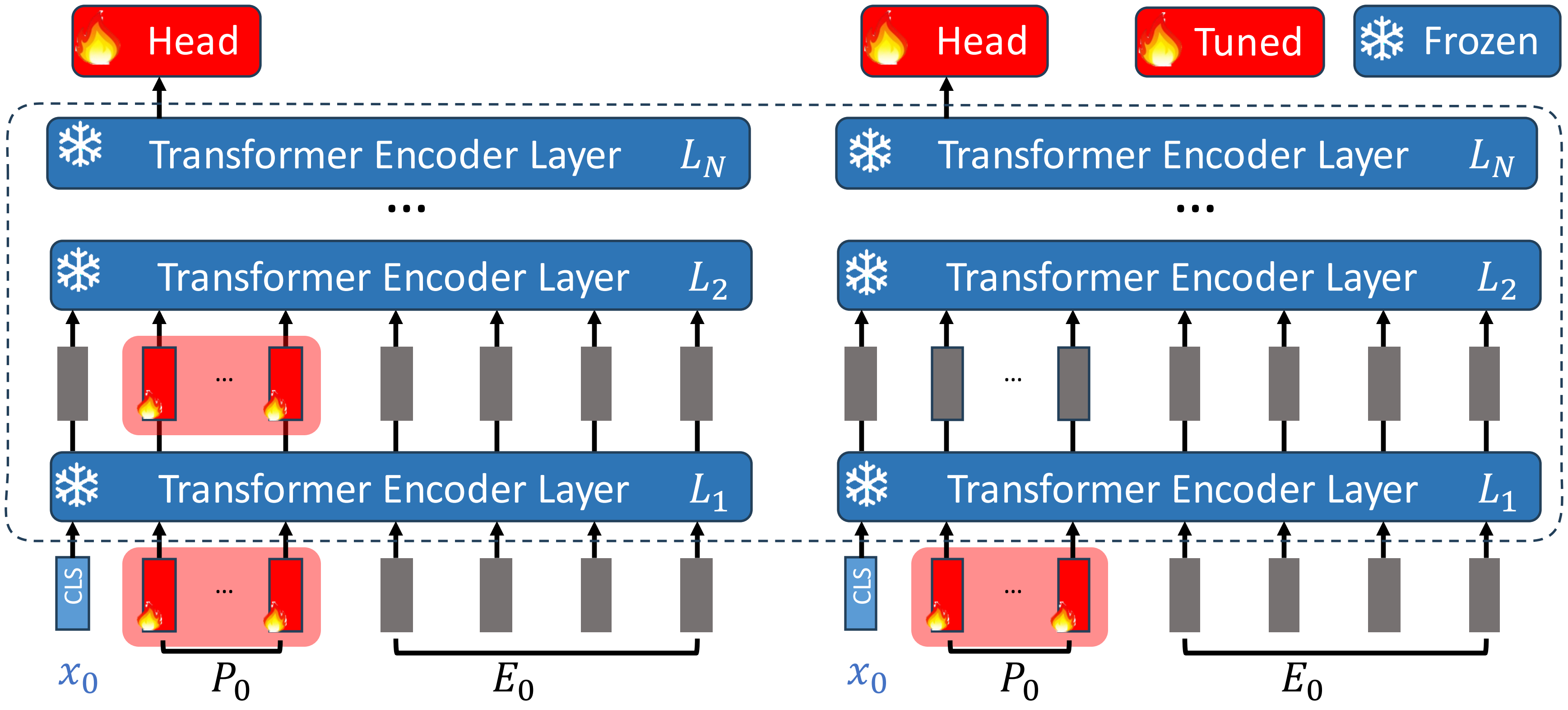
Khattak et al., Multi-modal Prompt Learning (MaPLe). CVPR 2023.

Chetan Arora
Department of Computer Science and Engineering, IIT Delhi

Visual Prompt Tuning

- Learned prompts adapt frozen model (e.g., no fine-tuning required) to different target tasks.





Visual-Prompt Tuning: Deep

Visual-Prompt Tuning: Shallow

Jia et al. Visual Prompt Tuning. ECCV 2022