CS2109S: Introduction to AI and Machine Learning

Lecture 12:
Attention Neural Networks and AI Ethics

15 April 2025

PollEv.com/conghuihu365

Announcement

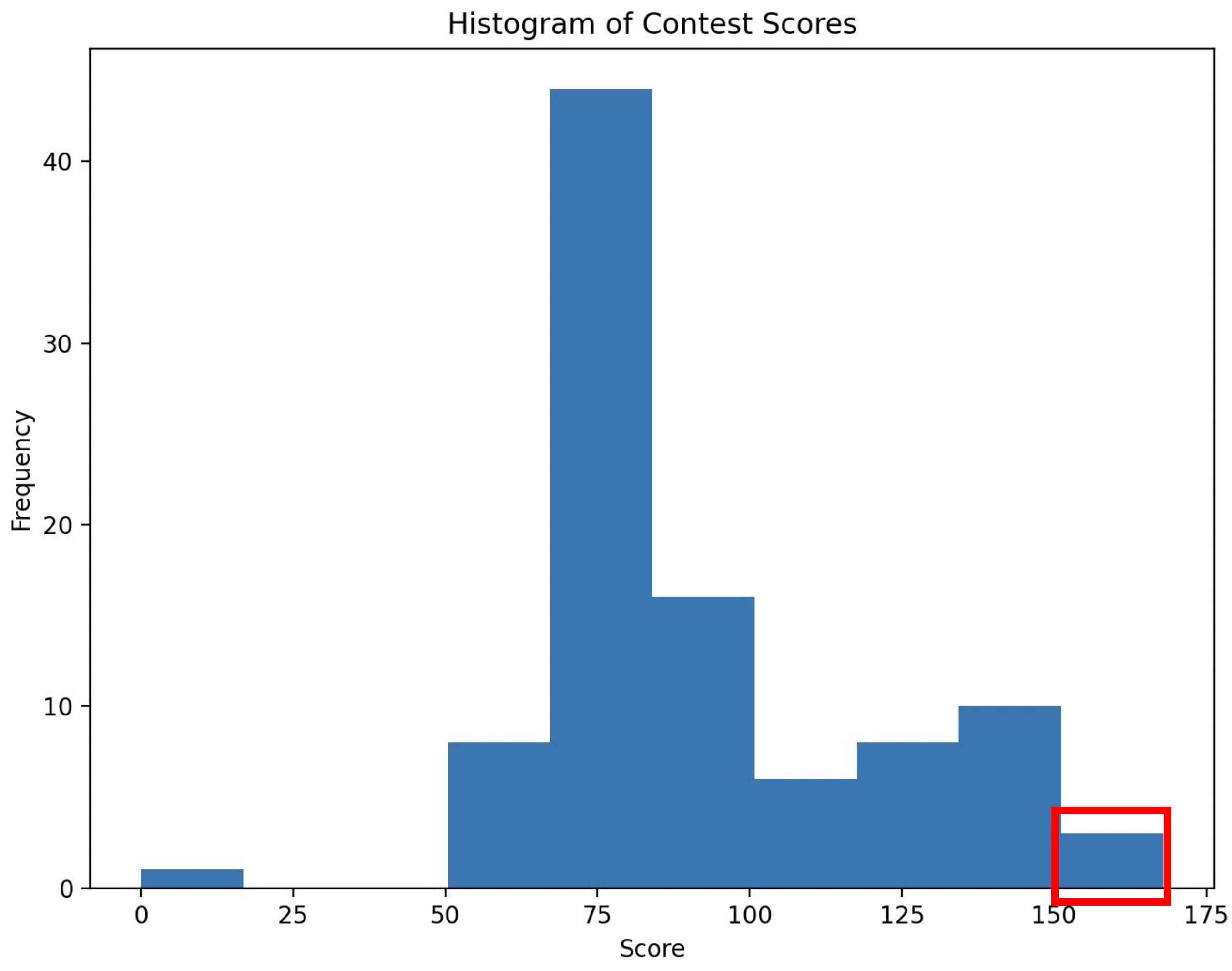
Final Exam

- Date & Time:
  - Monday, 5 May 2025, 9:00am - 11:00am
- Venue:
  - MPSH 1A & 1B
- Format:
  - Digital Assessment (Examplify) – F2F
- Materials:
  - All topics covered in lectures, trainings, tutorials, and problem sets, from Week 1 – 13
- Cheatsheet:
  - 1xA4 paper, both sides
- Calculators:
  - Standard and scientific calculators are allowed.

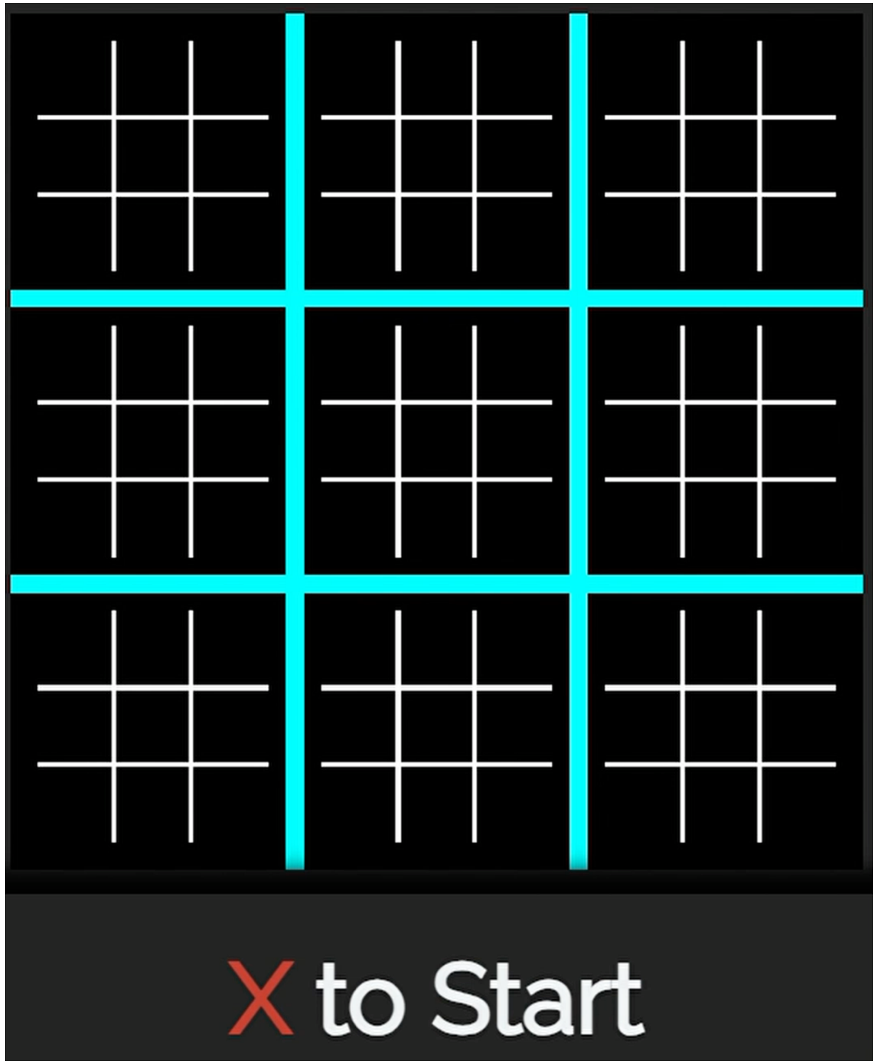
Contest: Top Agents

Top 3

1. Toh Yi Hui
2. Wallace Peck Teong Yee
3. Tan Wee Kean



Contest:



Materials

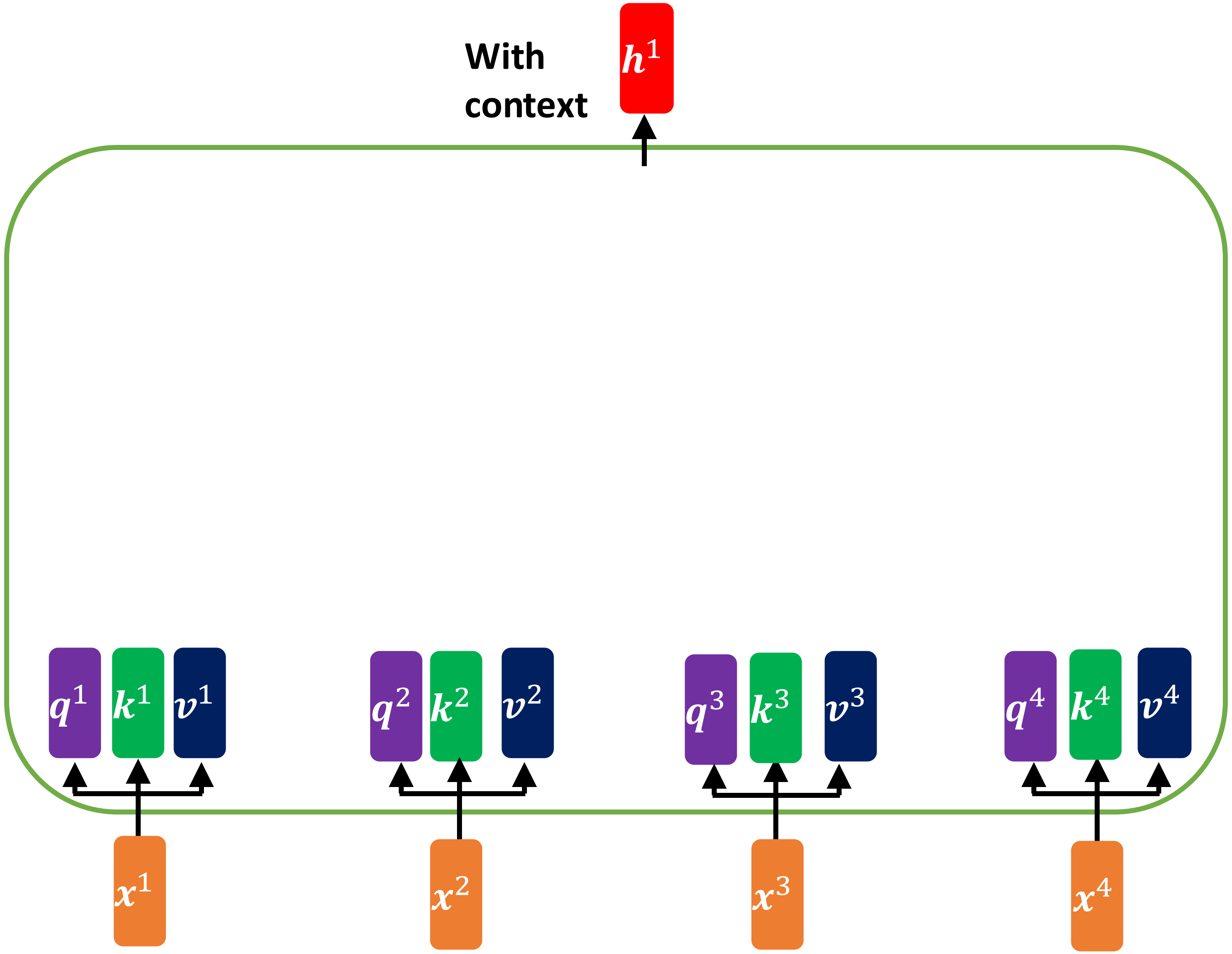
Outline

- Attention Neural Networks:
  - Masked Self-Attention Layer
  - Cross-Attention Layer
  - Transformer
- AI Ethics
  - Ethical Issues
- Course Recap
  - “Classical” AI
  - “Classical” ML
  - “Modern” ML

Outline

- **Attention Neural Networks:**
  - Masked Self-Attention Layer
  - Cross-Attention Layer
  - Transformer
- AI Ethics
  - Ethical Issues
- Course Recap
  - “Classical” AI
  - “Classical” ML
  - “Modern” ML

Self-Attention Layer



Step 1: Linear Projection

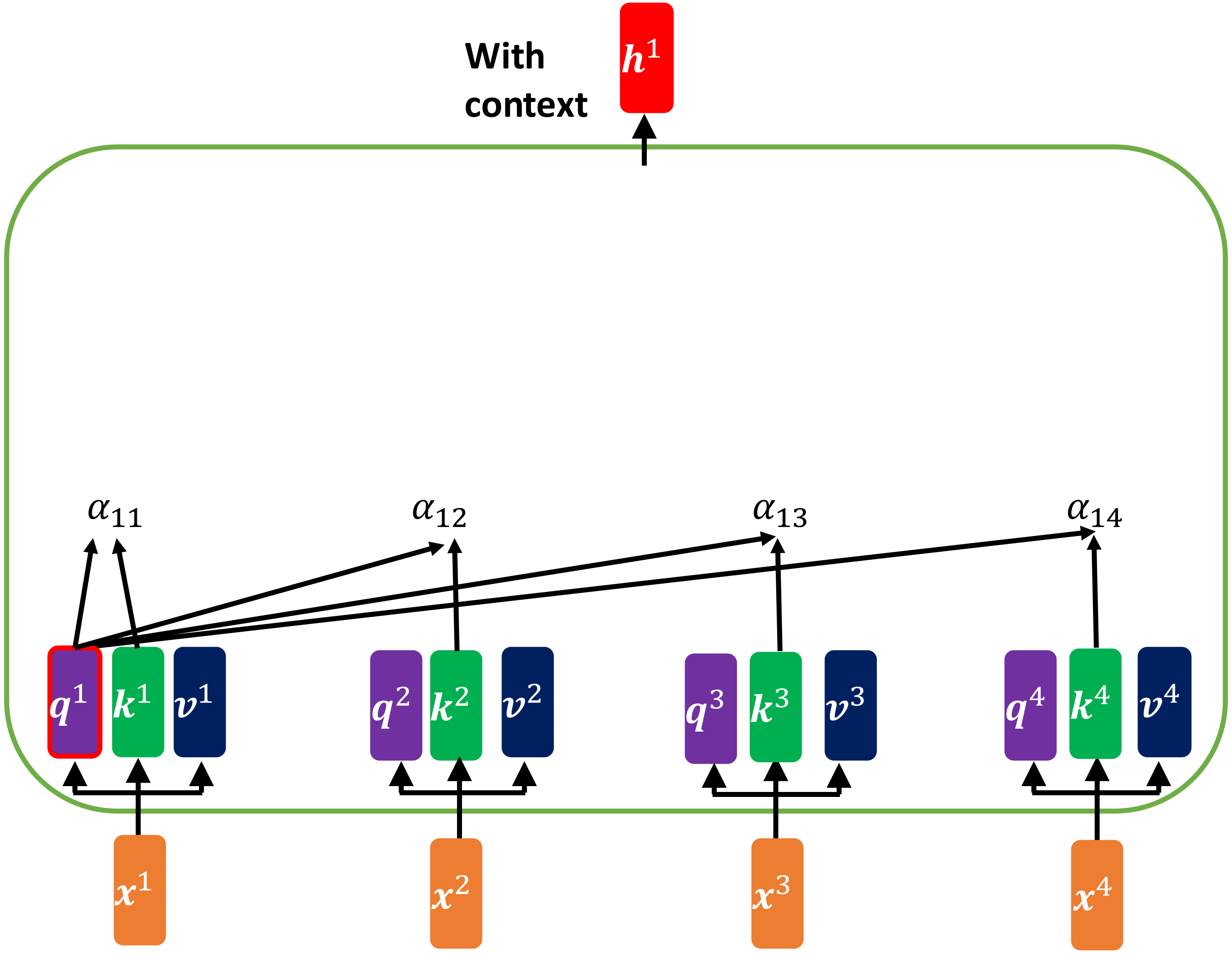
Transform input vectors into Query, Key, and Value using weights matrices (shared across inputs)

Query: $\boldsymbol{q}^i = \boldsymbol{W}^q \boldsymbol{x}^i$

Key: $\boldsymbol{k}^i = \boldsymbol{W}^k \boldsymbol{x}^i$

Value: $\boldsymbol{v}^i = \boldsymbol{W}^v \boldsymbol{x}^i$

Self-Attention Layer



Step 2: Compute the attention scores:
$\alpha_{1j} = \boldsymbol{k}^j \cdot \boldsymbol{q}^1 = (\boldsymbol{k}^j)^\top \boldsymbol{q}^1$

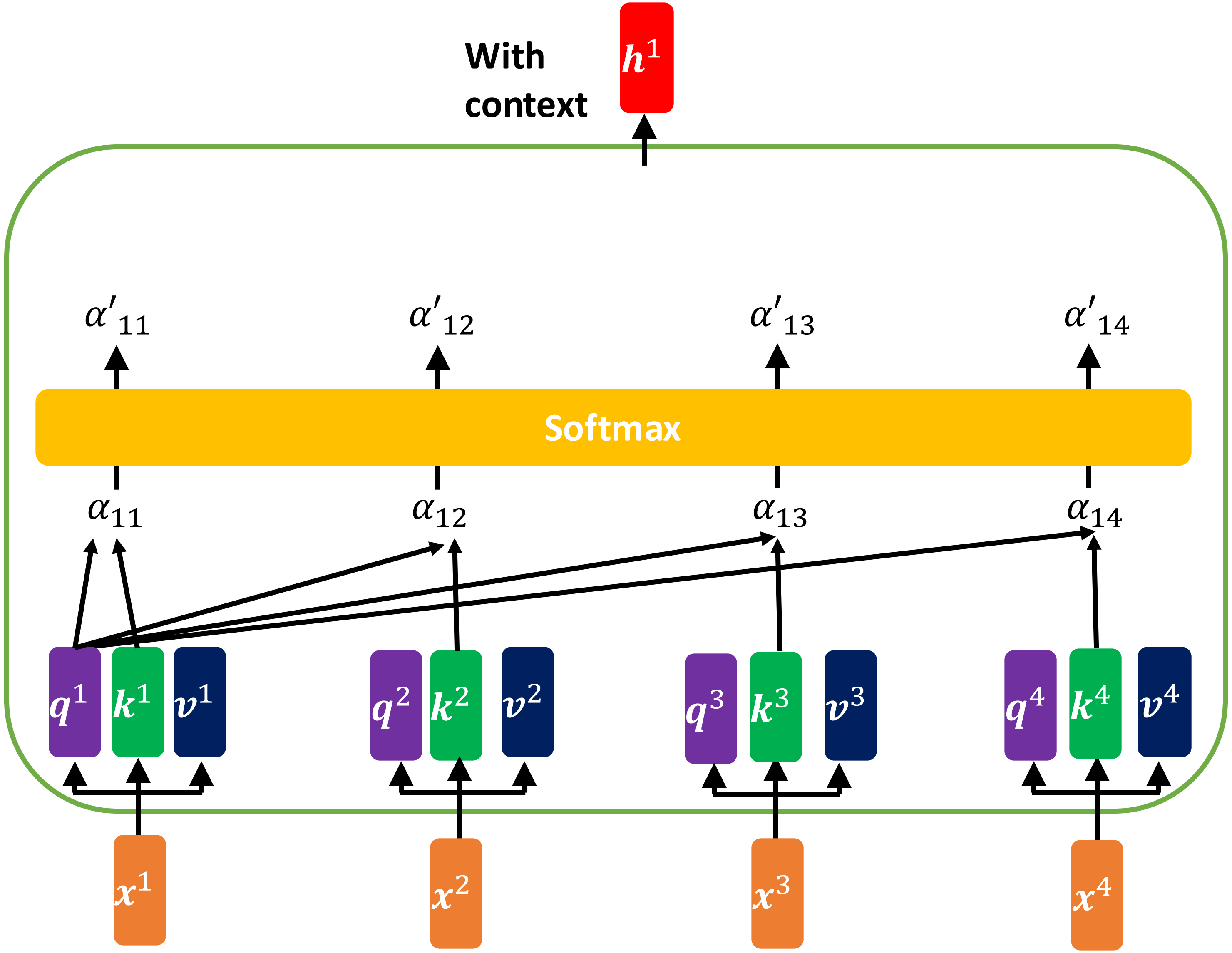
Step 1: Linear Projection
Transform input vectors into Query, Key, and Value using weights matrices (shared across inputs)

Query: $\boldsymbol{q}^i = \boldsymbol{W}^q \boldsymbol{x}^i$

Key: $\boldsymbol{k}^i = \boldsymbol{W}^k \boldsymbol{x}^i$

Value: $\boldsymbol{v}^i = \boldsymbol{W}^v \boldsymbol{x}^i$

Self-Attention Layer



Step 3: Apply Softmax: $\alpha'_{1j} = \frac{e^{\alpha_{1j}}}{\sum_j e^{\alpha_{1j}}}$

Step 2: Compute the attention scores:
$\alpha_{1j} = \boldsymbol{k}^j \cdot \boldsymbol{q}^1 = (\boldsymbol{k}^j)^\top \boldsymbol{q}^1$

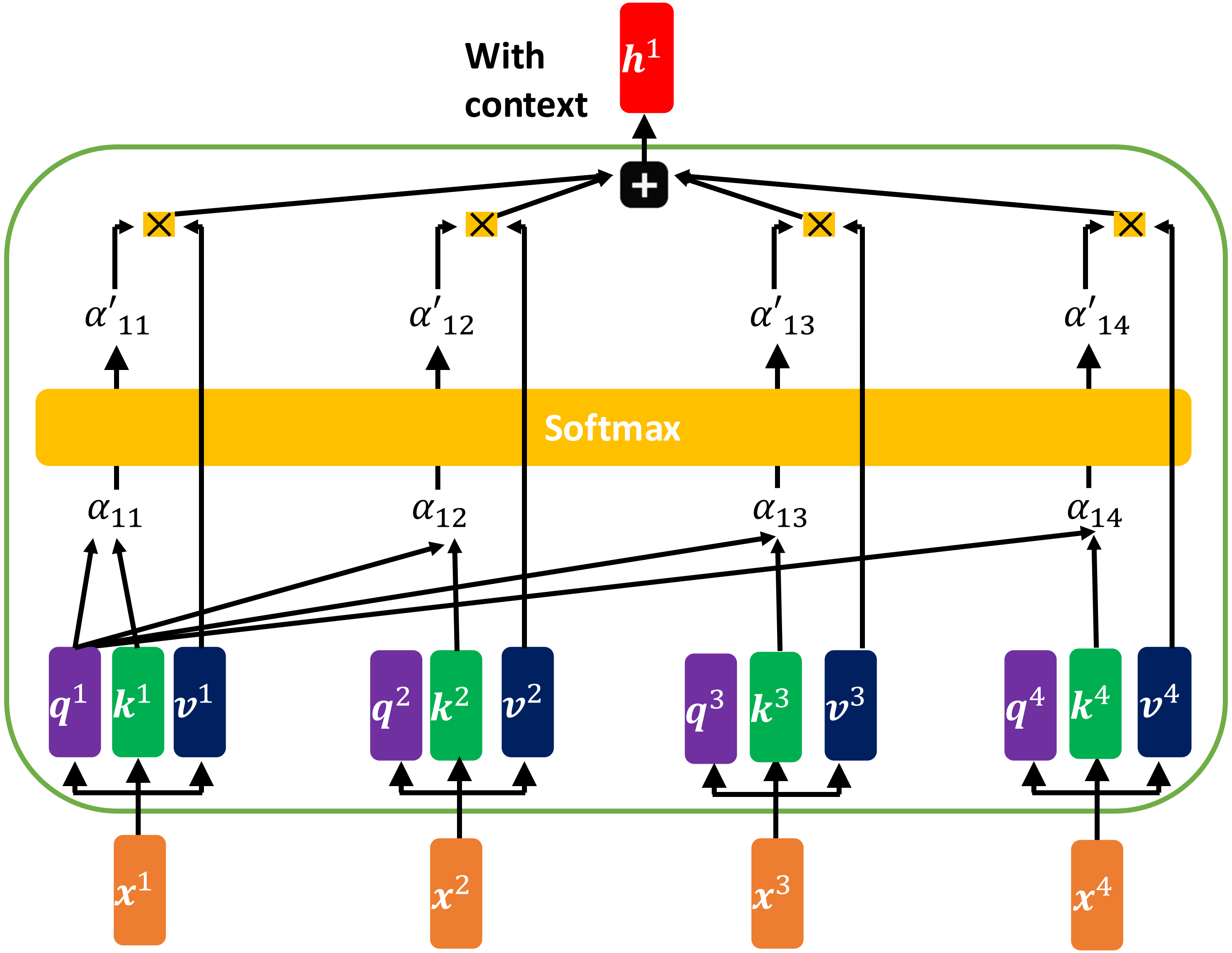
Step 1: Linear Projection
Transform input vectors into Query, Key, and Value using weights matrices (shared across inputs)

Query: $\boldsymbol{q}^i = \boldsymbol{W}^q \boldsymbol{x}^i$

Key: $\boldsymbol{k}^i = \boldsymbol{W}^k \boldsymbol{x}^i$

Value: $\boldsymbol{v}^i = \boldsymbol{W}^v \boldsymbol{x}^i$

Self-Attention Layer



The actual content to aggregate

Step 4: Aggregate information: Multiply Values by attention score (after Softmax)

$$\boldsymbol{h}^1 = \sum_j \alpha'_{1j} \boldsymbol{v}^j$$

Step 3: Apply Softmax: $\alpha'_{1j} = \frac{e^{\alpha_{1j}}}{\sum_j e^{\alpha_{1j}}}$

Step 2: Compute the attention scores:

$\alpha_{1j} = \boldsymbol{k}^j \cdot \boldsymbol{q}^1 = (\boldsymbol{k}^j)^\top \boldsymbol{q}^1$

Step 1: Linear Projection

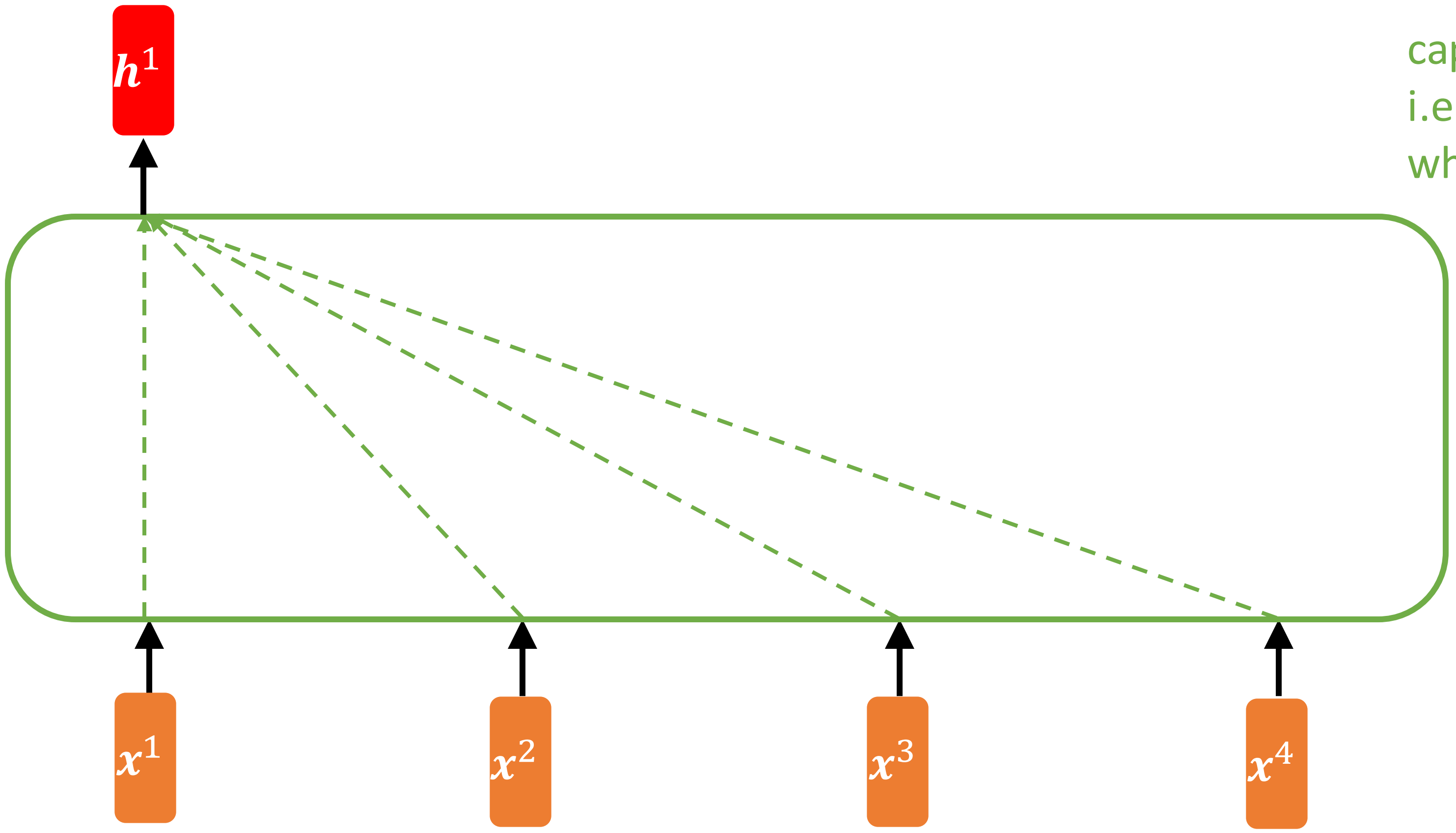
Transform input vectors into Query, Key, and Value using weights matrices (shared across inputs)

Query: $\boldsymbol{q}^i = \boldsymbol{W}^q \boldsymbol{x}^i$

Key: $\boldsymbol{k}^i = \boldsymbol{W}^k \boldsymbol{x}^i$

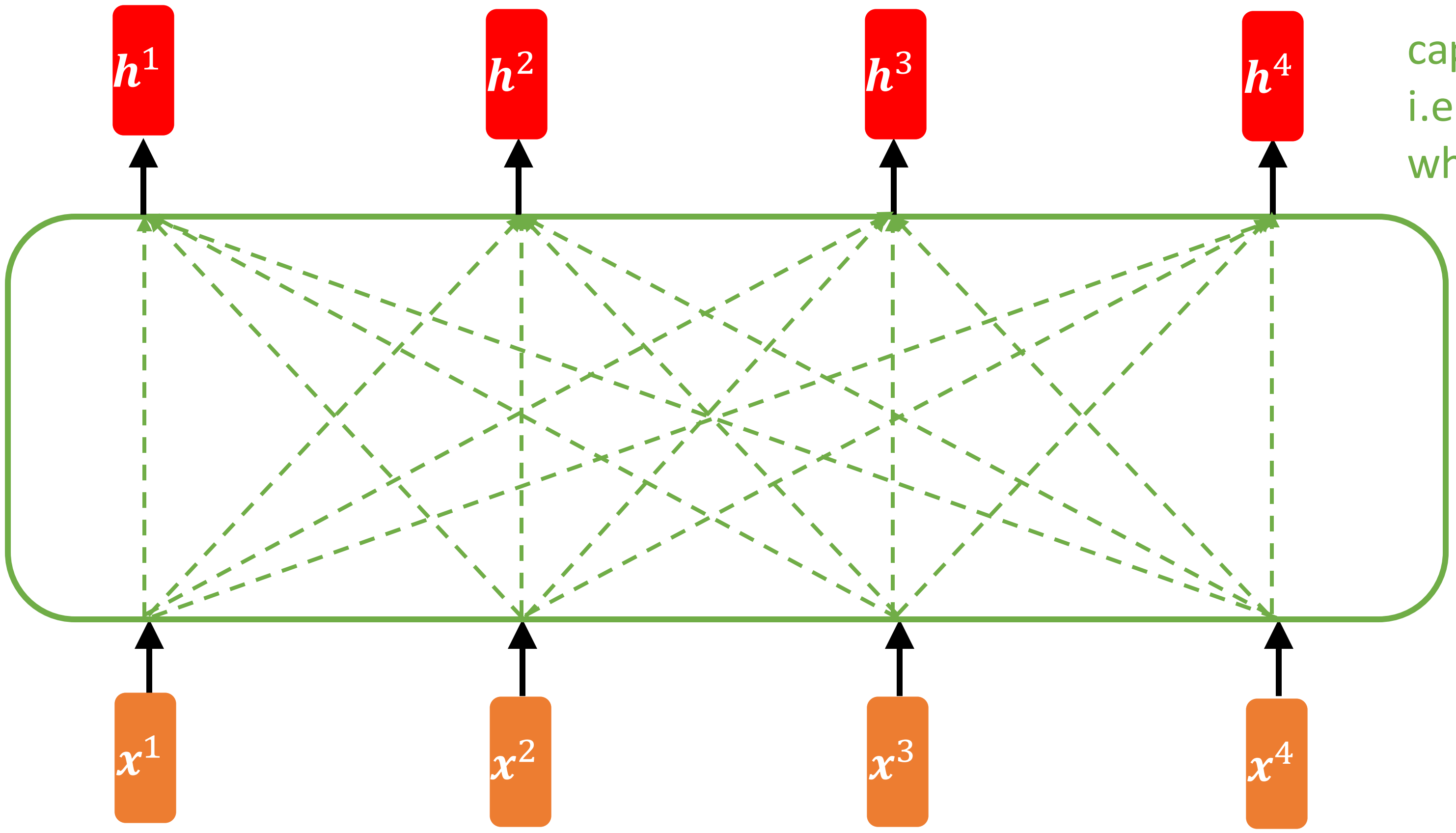
Value: $\boldsymbol{v}^i = \boldsymbol{W}^v \boldsymbol{x}^i$

Self-Attention Layer



capture contextual information, i.e., useful information from the whole input sequence.

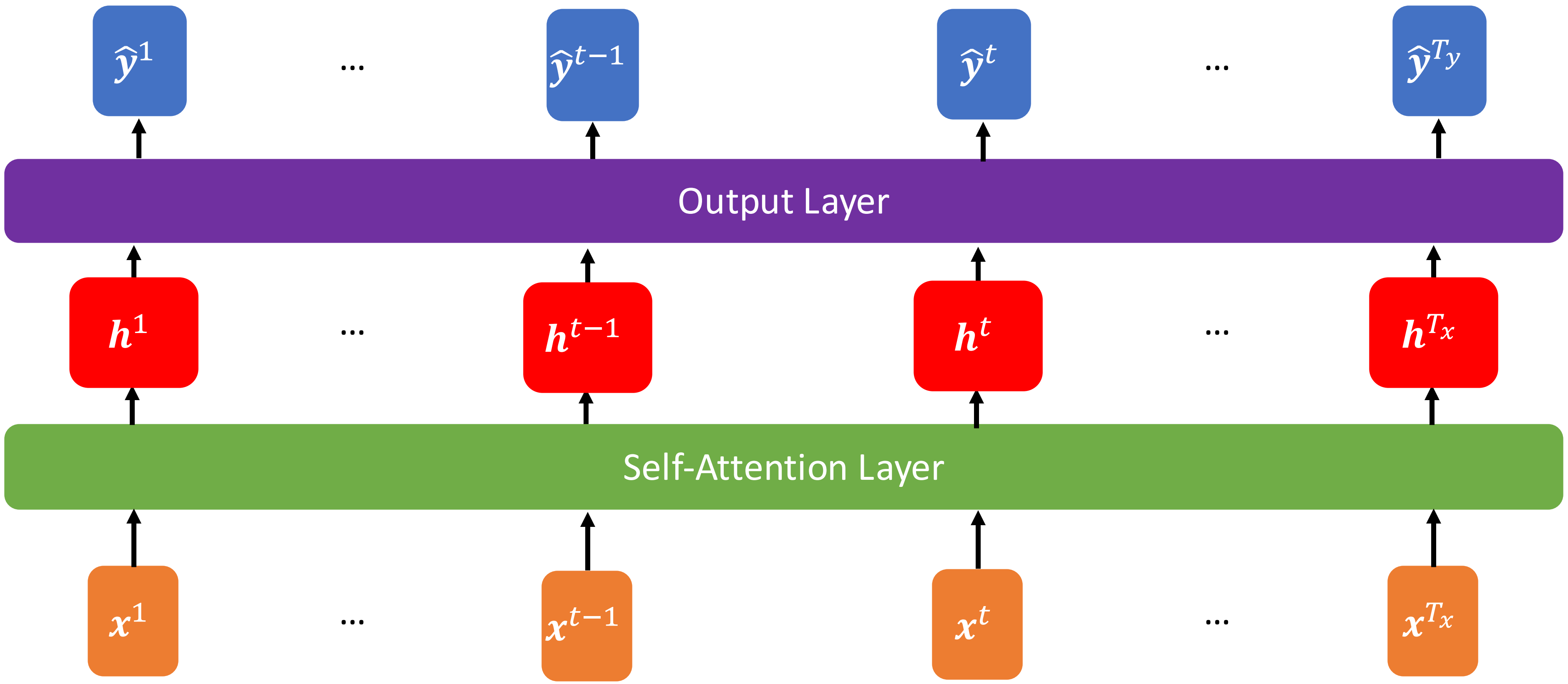
Self-Attention Layer



capture contextual information, i.e., useful information from the whole input sequence.

Attention Neural Network: Many-to-Many

$T_x = T_y > 1$



Attention Neural Network: Many-to-One

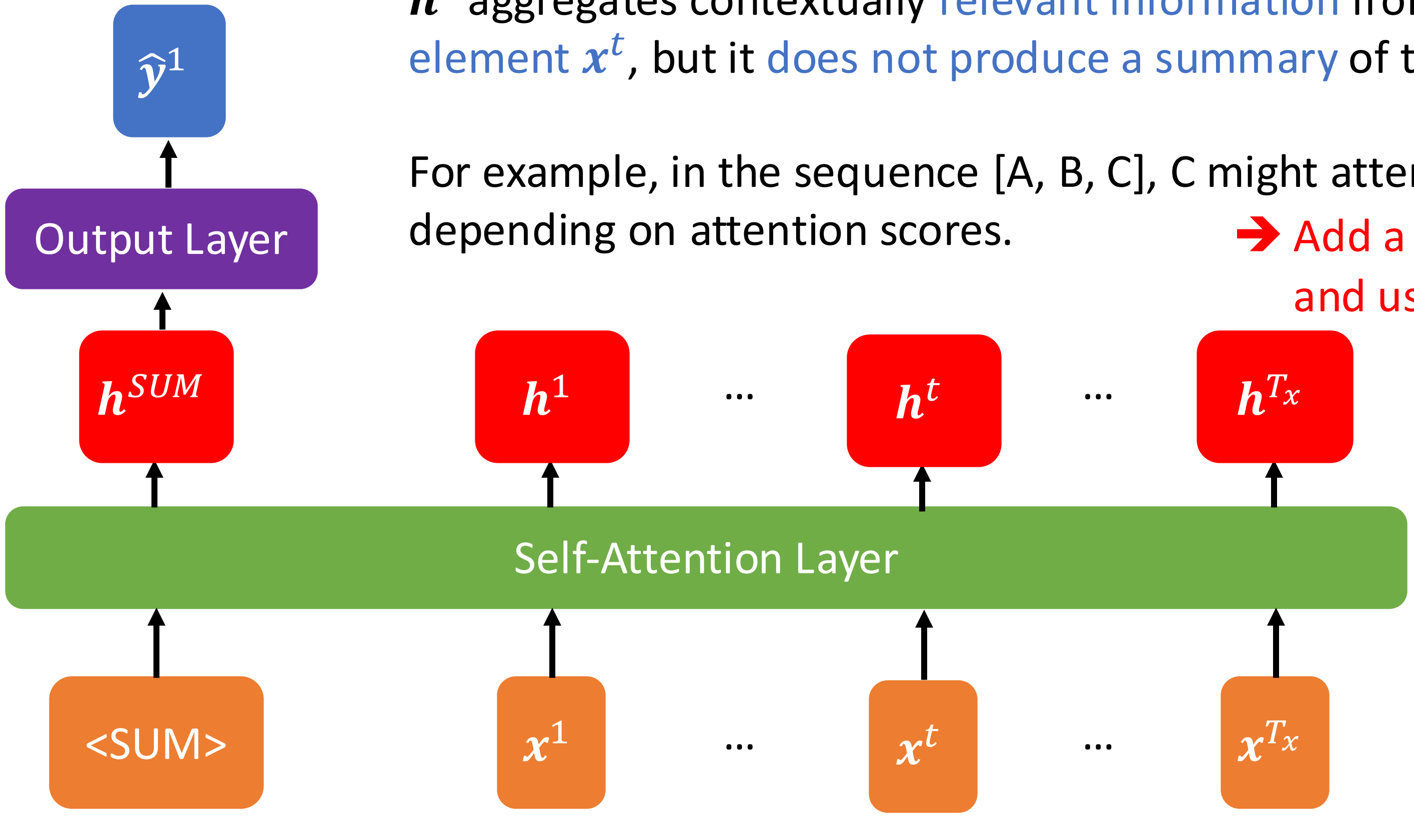
$T_x > 1, T_y = 1$

Which $\boldsymbol{h}^t$ $(1 \leq t \leq T_x)$ should be used for generating the output?

$\boldsymbol{h}^t$ aggregates contextually relevant information from the whole sentence for input element $\boldsymbol{x}^t$, but it does not produce a summary of the entire sentence

For example, in the sequence [A, B, C], C might attend heavily to B but ignore A, depending on attention scores.

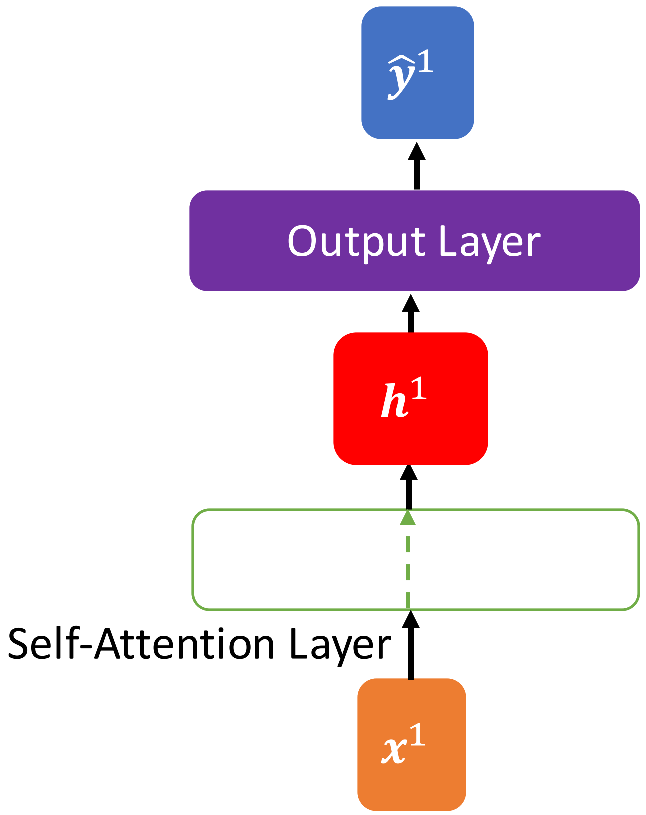
➔ Add a learnable input vector <SUM> and use $\boldsymbol{h}^{SUM}$ to predict the output



The <SUM> vector is optimized to enable the $\boldsymbol{h}^{SUM}$ to capture relevant summary information throughout the input entire sequence, thereby minimizing training loss.

Attention Neural Network: One-to-Many

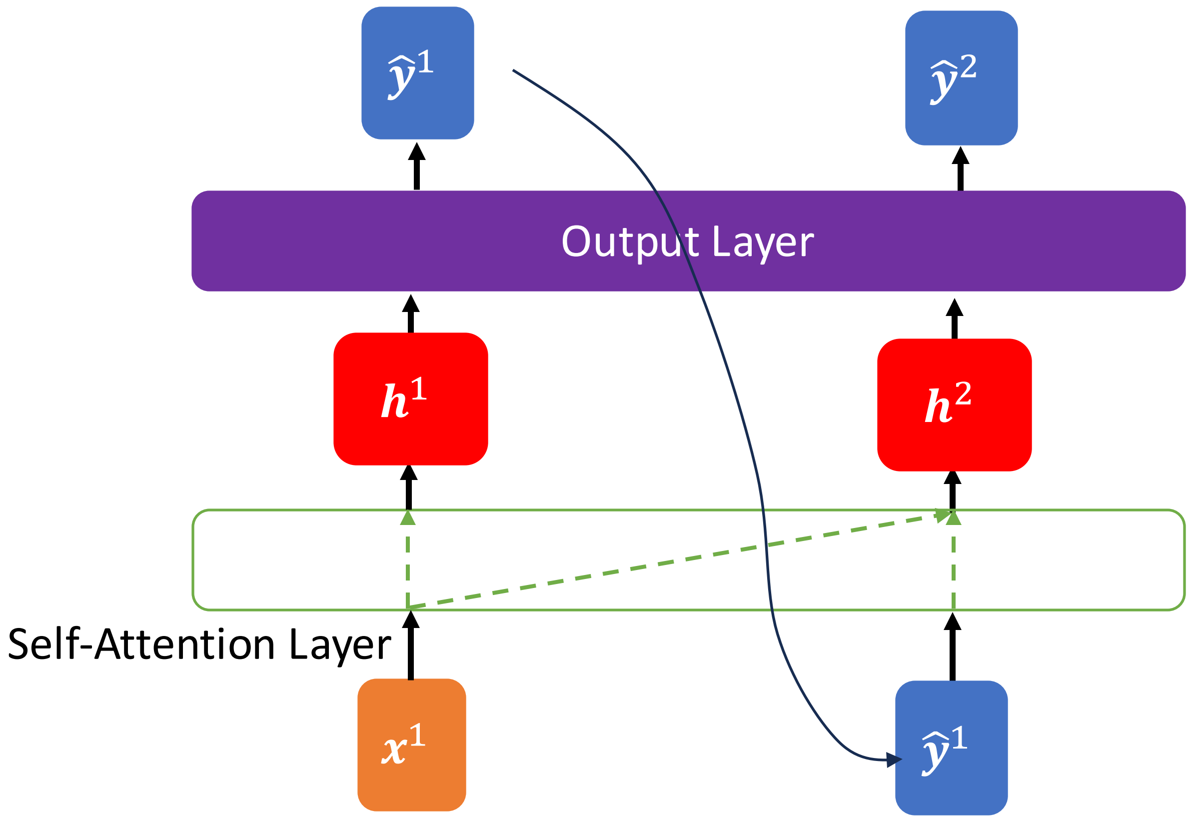
$T_x = 1, T_y > 1$



$\boldsymbol{x}^1$ is used to generate $\boldsymbol{h}^1$.

Attention Neural Network: One-to-Many

$T_x = 1, T_y > 1$

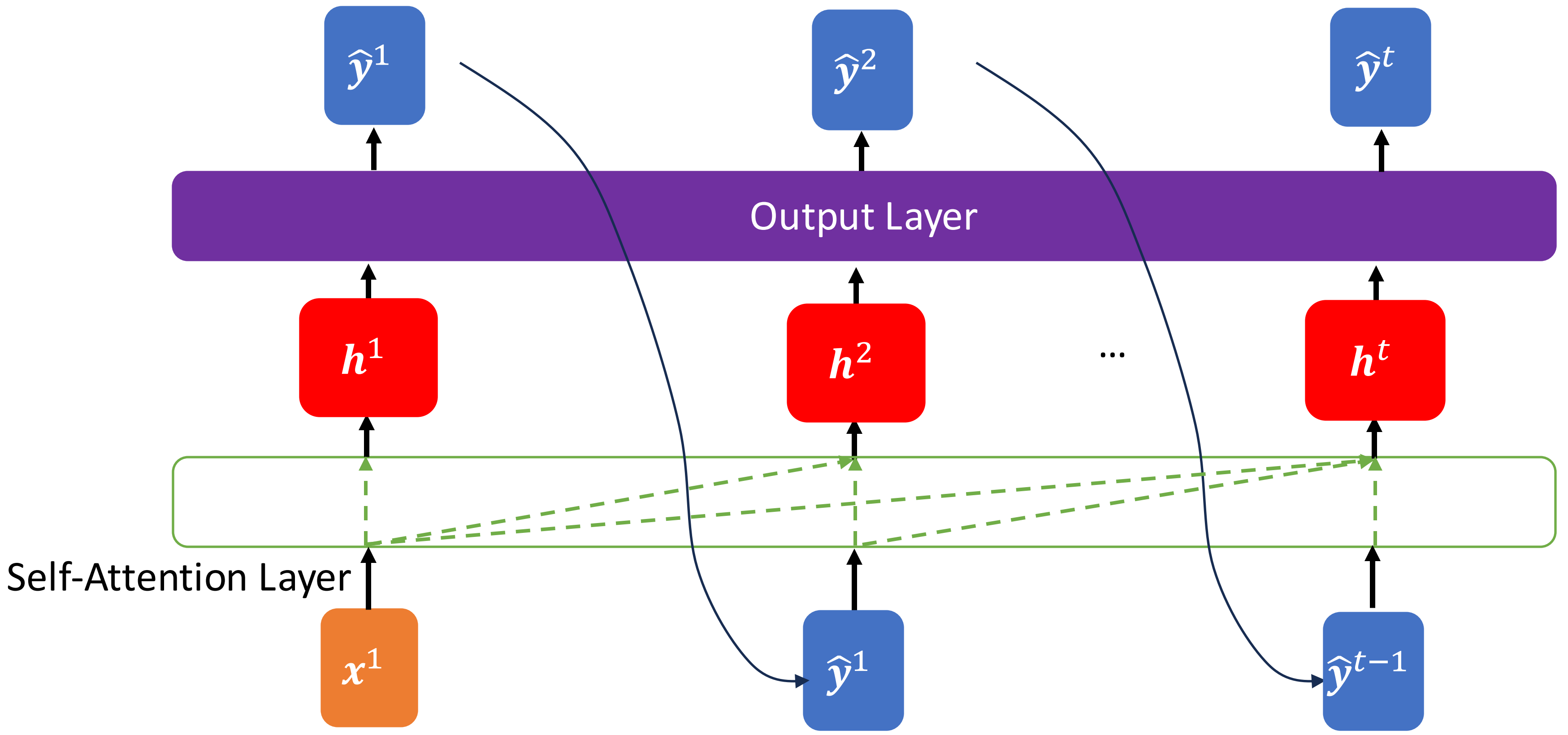


$\boldsymbol{x}^1$ is used to generate $\boldsymbol{h}^1$.

$\boldsymbol{x}^1$ and $\hat{\boldsymbol{y}}^1$ are used to generate $\boldsymbol{h}^2$.

Attention Neural Network: One-to-Many

$T_x = 1, T_y > 1$



$\boldsymbol{x}^1$ is used to generate $\boldsymbol{h}^1$.

$\boldsymbol{x}^1$ and $\hat{\boldsymbol{y}}^1$ are used to generate $\boldsymbol{h}^2$.

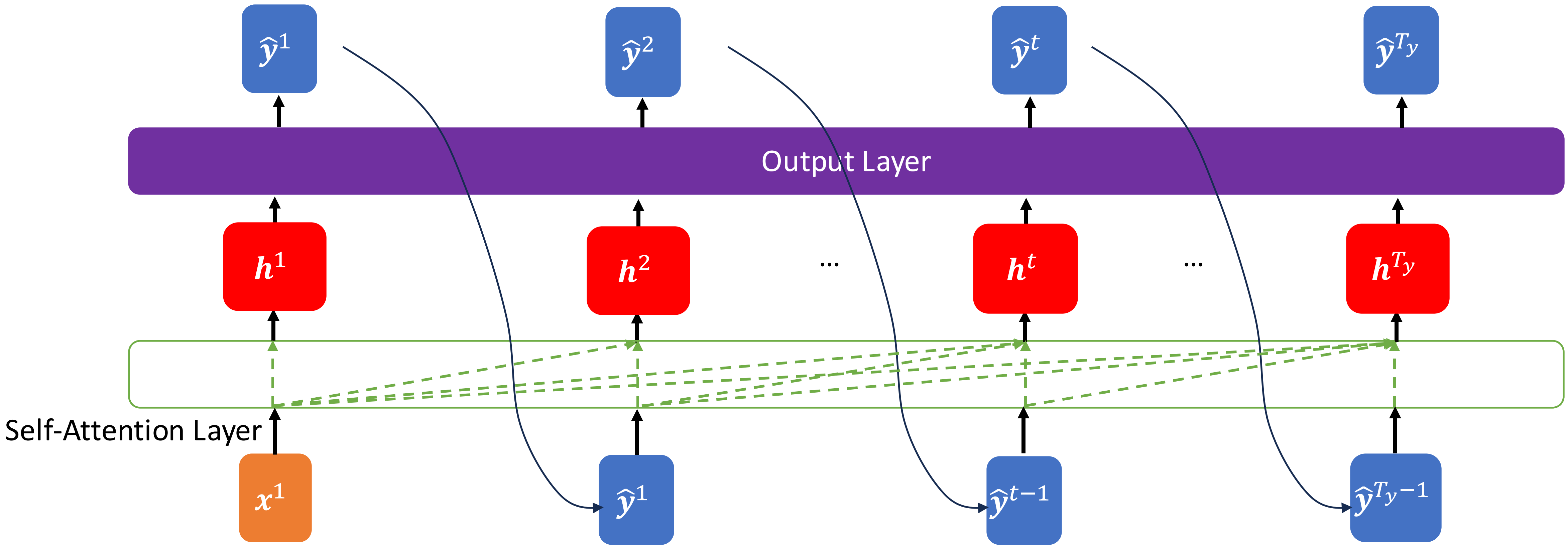
$\boldsymbol{x}^1, \hat{\boldsymbol{y}}^1, \hat{\boldsymbol{y}}^2 \ldots \hat{\boldsymbol{y}}^{t-1}$ are used to generate $\boldsymbol{h}^t$.

Attention Neural Network: One-to-Many

$T_x = 1, T_y > 1$

Given the input sequence $\boldsymbol{x}^1, \widehat{\boldsymbol{y}}^1, \widehat{\boldsymbol{y}}^2 \ldots \widehat{\boldsymbol{y}}^{t-1} \ldots \widehat{\boldsymbol{y}}^{T_y-1}$, if we want to generate $\boldsymbol{h}^1$ (as in the first step) using only $\boldsymbol{x}^1$, how can we ensure this?

Masked Self-Attention Layer!



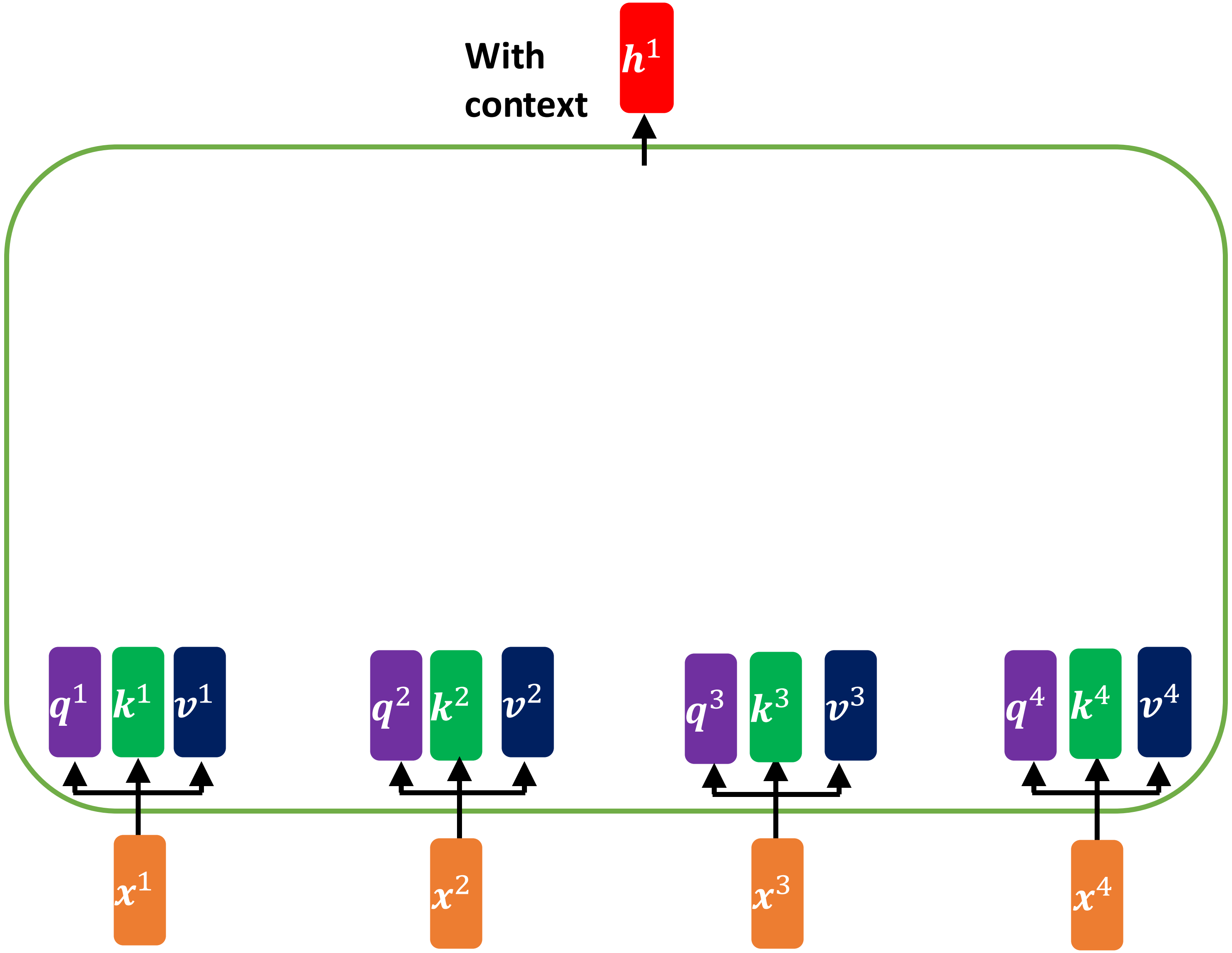
$\boldsymbol{x}^1$ is used to generate $\boldsymbol{h}^1$.

$\boldsymbol{x}^1$ and $\widehat{\boldsymbol{y}}^1$ are used to generate $\boldsymbol{h}^2$.

$\boldsymbol{x}^1, \widehat{\boldsymbol{y}}^1, \widehat{\boldsymbol{y}}^2 \ldots \widehat{\boldsymbol{y}}^{t-1}$ are used to generate $\boldsymbol{h}^t$.

$\boldsymbol{x}^1, \widehat{\boldsymbol{y}}^1, \widehat{\boldsymbol{y}}^2 \ldots \widehat{\boldsymbol{y}}^{T_y-1}$ are used to generate $\boldsymbol{h}^{T_y}$.

Masked Self-Attention Layer



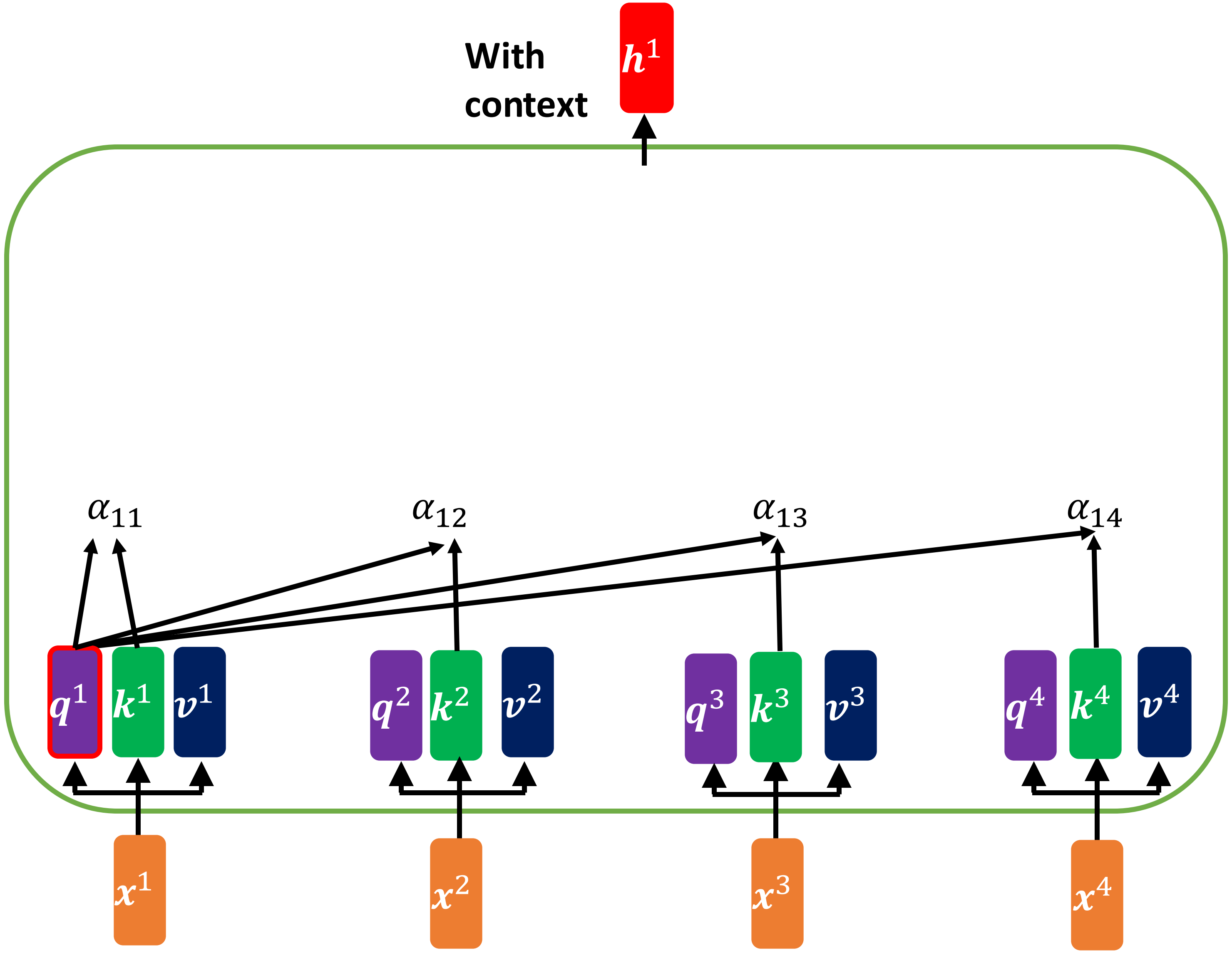
Step 1: Linear Projection
Transform input vectors into Query, Key, and Value using weights matrices (shared across inputs)

Query: $\boldsymbol{q}^i = \boldsymbol{W}^q \boldsymbol{x}^i$

Key: $\boldsymbol{k}^i = \boldsymbol{W}^k \boldsymbol{x}^i$

Value: $\boldsymbol{v}^i = \boldsymbol{W}^v \boldsymbol{x}^i$

Masked Self-Attention Layer



Step 2: Compute the attention scores:

$\alpha_{1j} = \boldsymbol{k}^j \cdot \boldsymbol{q}^1 = (\boldsymbol{k}^j)^\top \boldsymbol{q}^1$

Step 1: Linear Projection

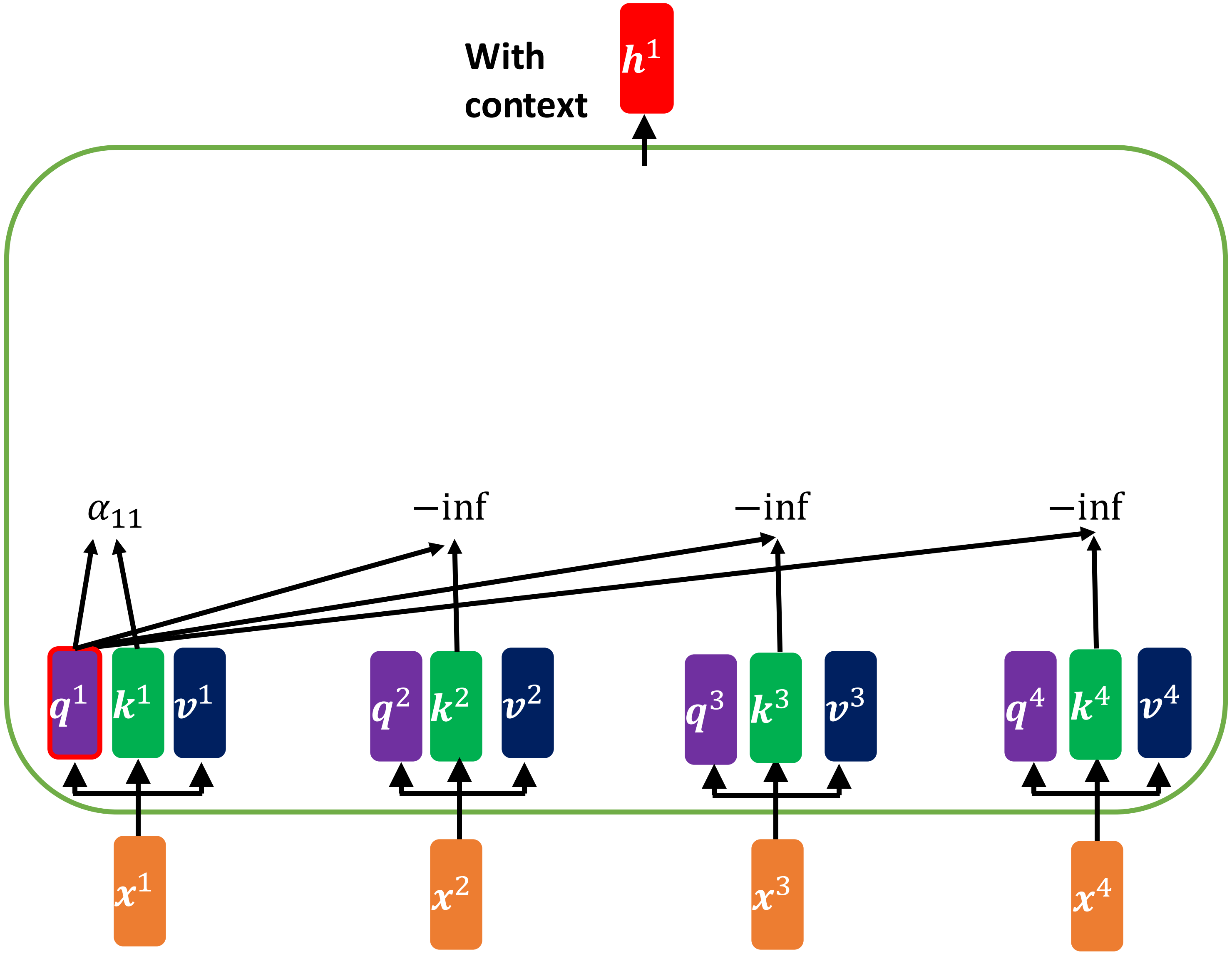
Transform input vectors into Query, Key, and Value using weights matrices (shared across inputs)

Query: $\boldsymbol{q}^i = \boldsymbol{W}^q \boldsymbol{x}^i$

Key: $\boldsymbol{k}^i = \boldsymbol{W}^k \boldsymbol{x}^i$

Value: $\boldsymbol{v}^i = \boldsymbol{W}^v \boldsymbol{x}^i$

Masked Self-Attention Layer



Step 3: Add mask to attention scores:
$[0, -\text{inf}, -\text{inf}, -\text{inf}]$

Step 2: Compute the attention scores:
$\alpha_{1j} = \boldsymbol{k}^j \cdot \boldsymbol{q}^1 = (\boldsymbol{k}^j)^\top \boldsymbol{q}^1$

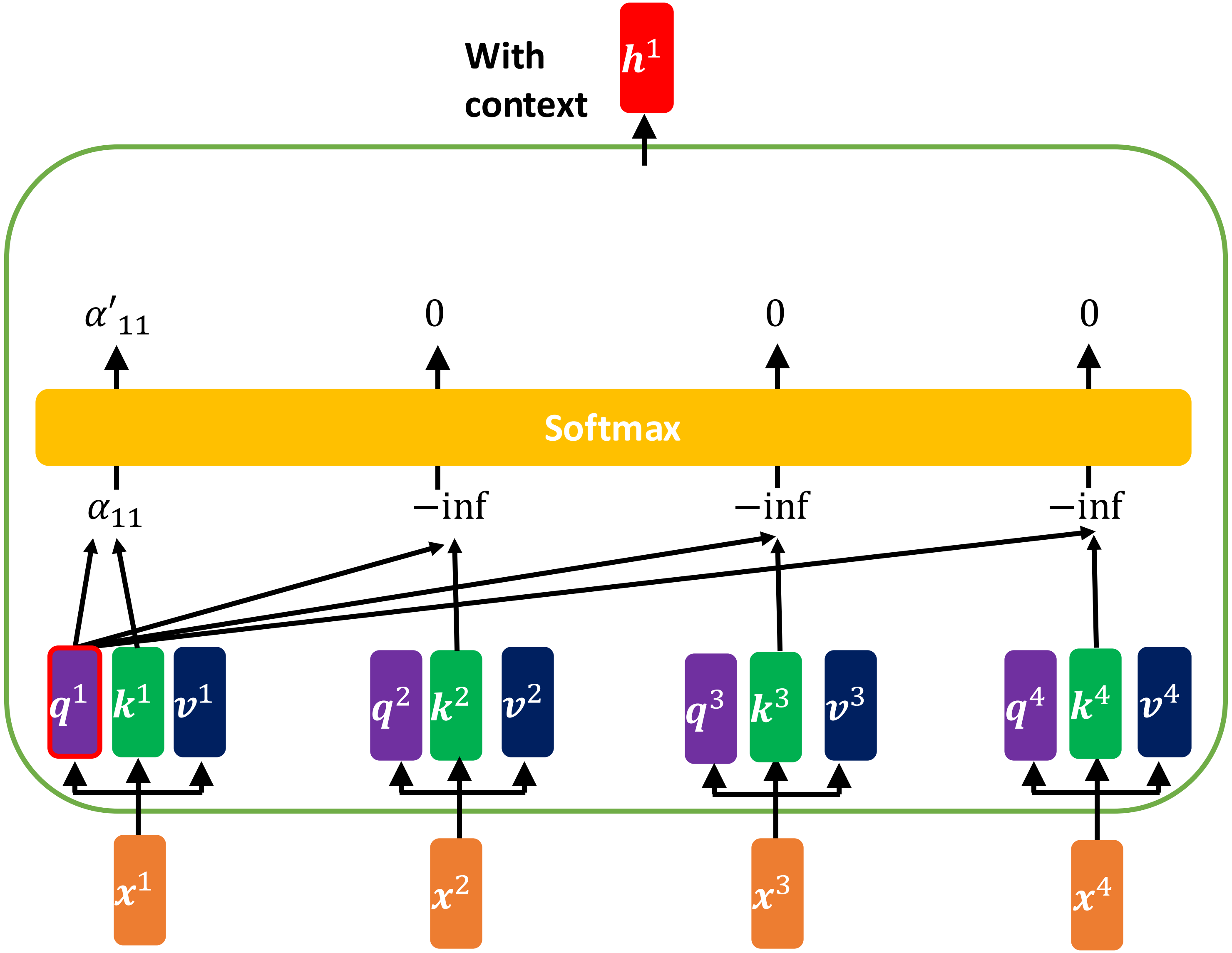
Step 1: Linear Projection
Transform input vectors into Query, Key, and Value using weights matrices (shared across inputs)

Query: $\boldsymbol{q}^i = \boldsymbol{W}^q \boldsymbol{x}^i$

Key: $\boldsymbol{k}^i = \boldsymbol{W}^k \boldsymbol{x}^i$

Value: $\boldsymbol{v}^i = \boldsymbol{W}^v \boldsymbol{x}^i$

Masked Self-Attention Layer



Step 4: Apply Softmax: $\alpha'_{1j} = \frac{e^{\alpha_{1j}}}{\sum_j e^{\alpha_{1j}}}$ $\quad e^{-\text{inf}} = 0$

Step 3: Add mask to attention scores:
$[0, -\text{inf}, -\text{inf}, -\text{inf}]$

Step 2: Compute the attention scores:
$\alpha_{1j} = \boldsymbol{k}^j \cdot \boldsymbol{q}^1 = (\boldsymbol{k}^j)^\top \boldsymbol{q}^1$

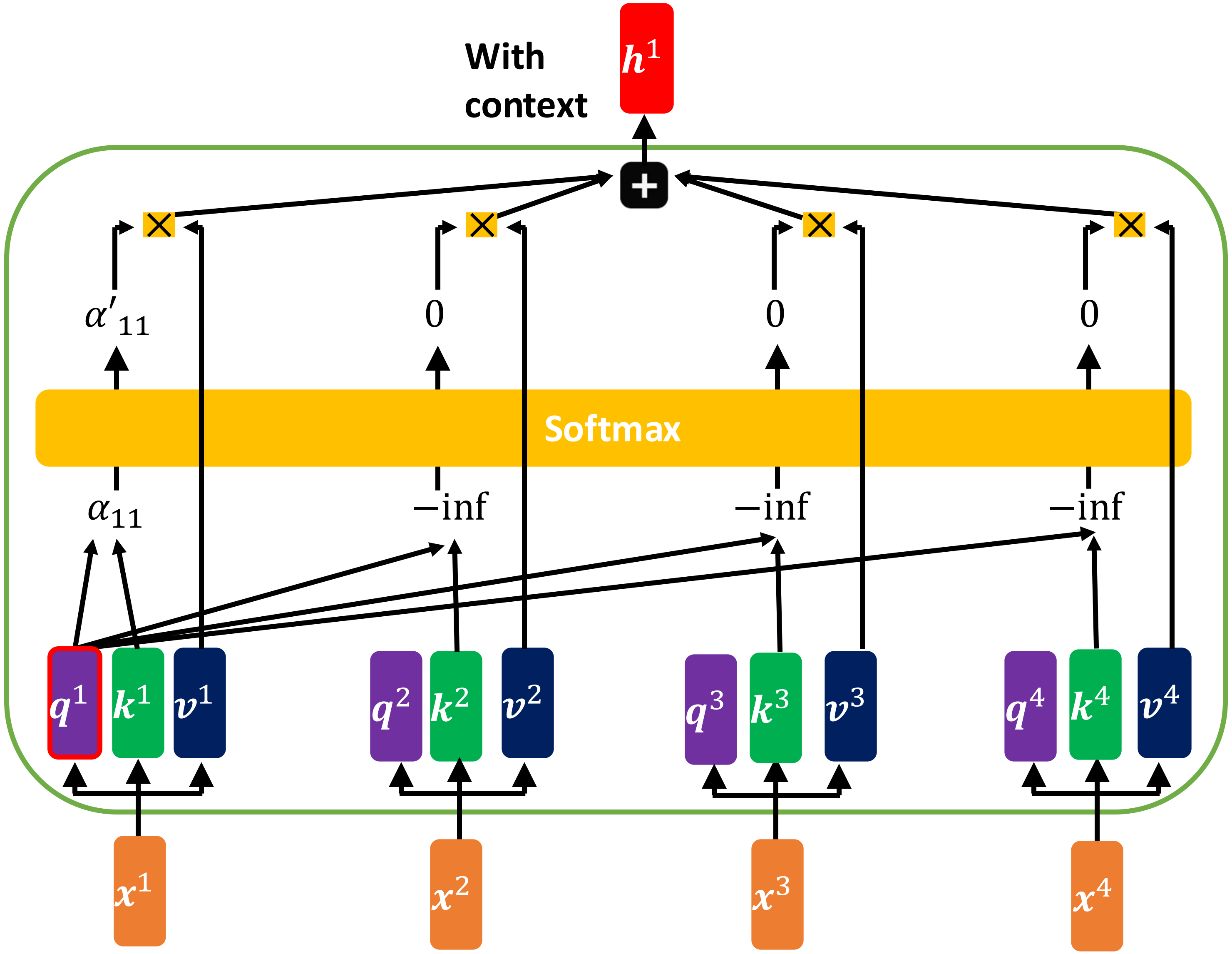
Step 1: Linear Projection
Transform input vectors into Query, Key, and Value using weights matrices (shared across inputs)

Query: $\boldsymbol{q}^i = \boldsymbol{W}^q \boldsymbol{x}^i$

Key: $\boldsymbol{k}^i = \boldsymbol{W}^k \boldsymbol{x}^i$

Value: $\boldsymbol{v}^i = \boldsymbol{W}^v \boldsymbol{x}^i$

Masked Self-Attention Layer



Step 5: Aggregate information: Multiply Values by attention score (after Softmax) $\boldsymbol{h}^1 = \sum_j \alpha'_{1j} \boldsymbol{v}^j$

Step 4: Apply Softmax: $\alpha'_{1j} = \frac{e^{\alpha_{1j}}}{\sum_j e^{\alpha_{1j}}}$ $\quad e^{-\text{inf}} = 0$

Step 3: Add mask to attention scores: $[0, -\text{inf}, -\text{inf}, -\text{inf}]$

Step 2: Compute the attention scores: $\alpha_{1j} = \boldsymbol{k}^j \cdot \boldsymbol{q}^1 = (\boldsymbol{k}^j)^\top \boldsymbol{q}^1$

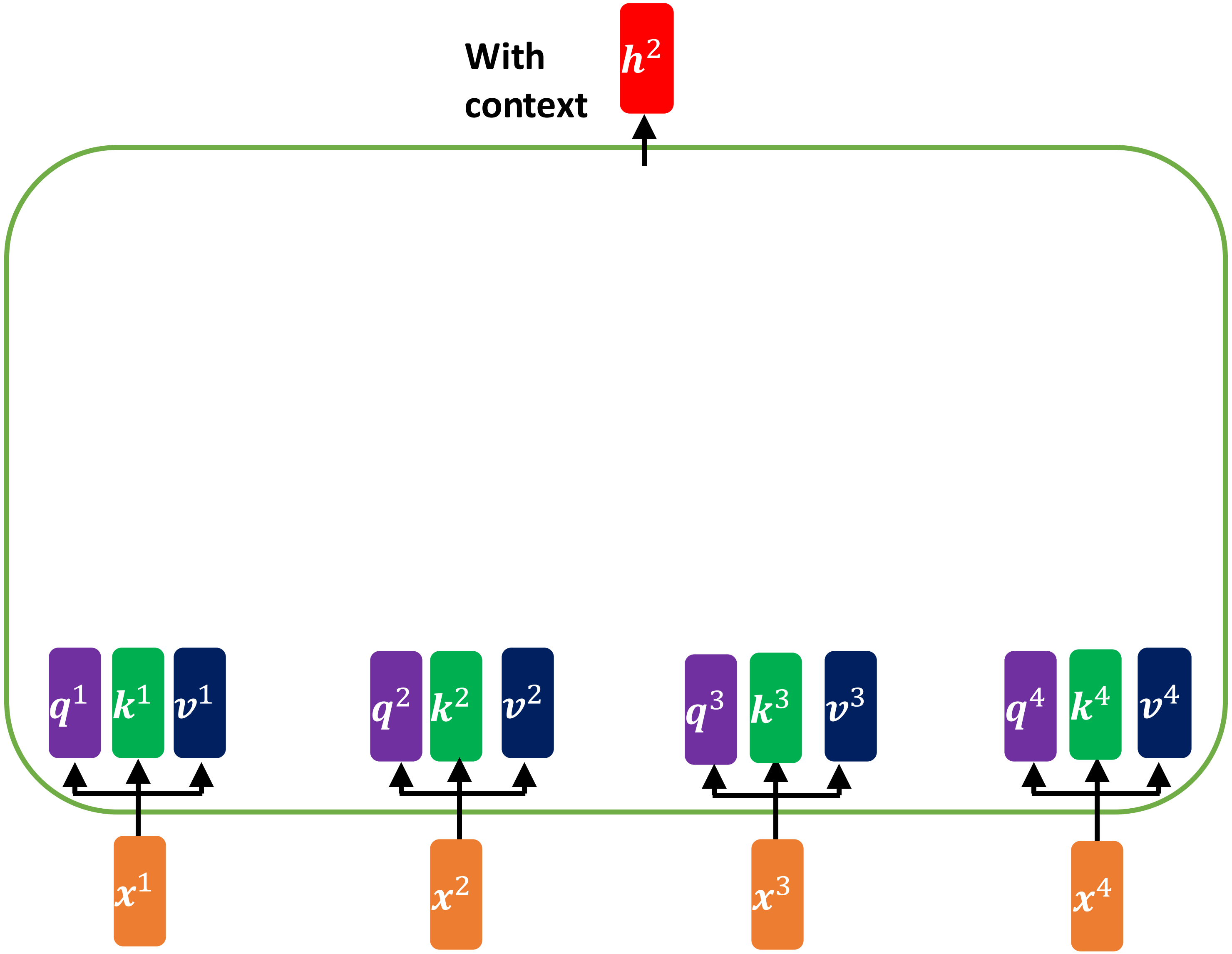
Step 1: Linear Projection
Transform input vectors into Query, Key, and Value using weights matrices (shared across inputs)

Query: $\boldsymbol{q}^i = \boldsymbol{W}^q \boldsymbol{x}^i$

Key: $\boldsymbol{k}^i = \boldsymbol{W}^k \boldsymbol{x}^i$

Value: $\boldsymbol{v}^i = \boldsymbol{W}^v \boldsymbol{x}^i$

Masked Self-Attention Layer



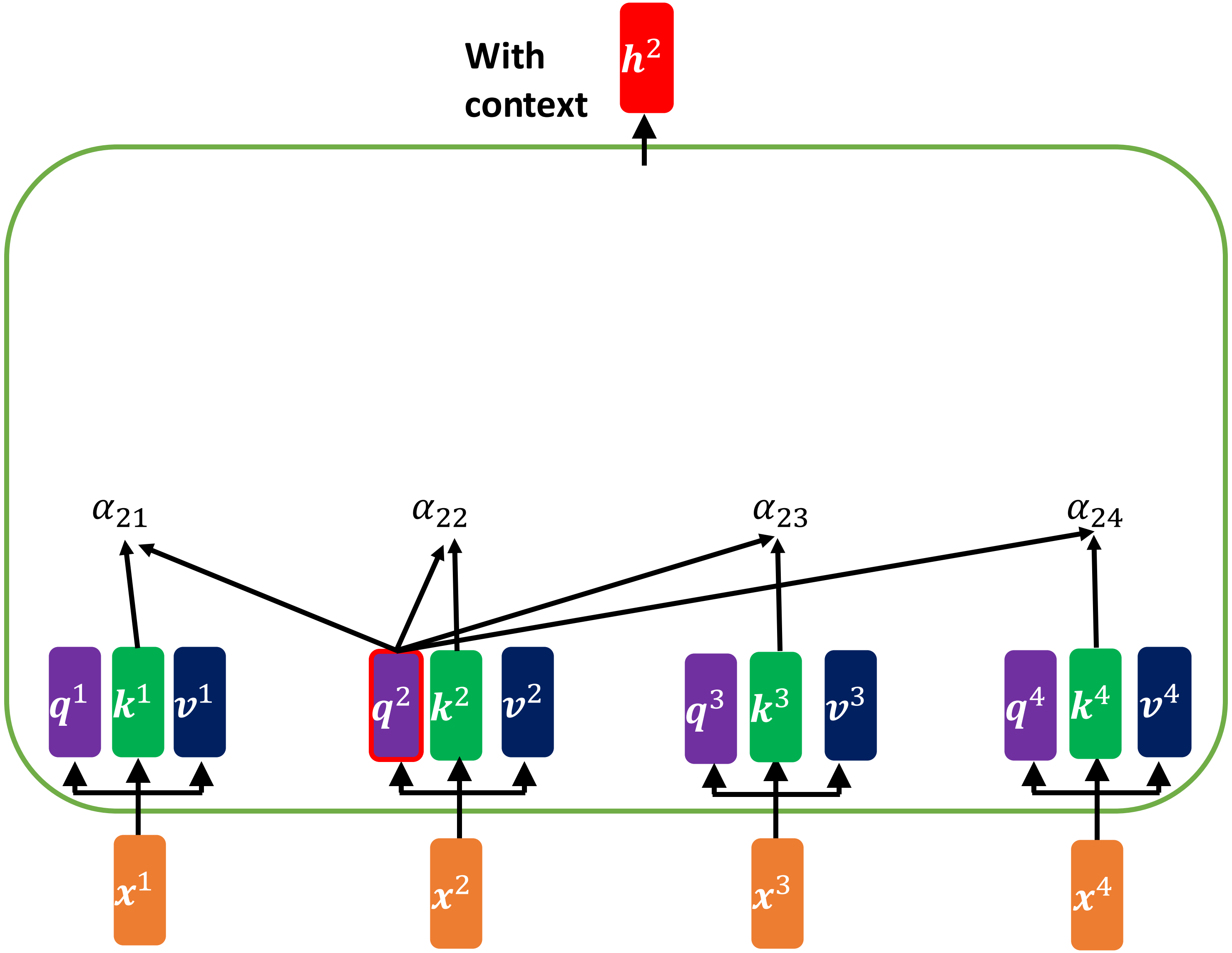
Step 1: Linear Projection
Transform input vectors into Query, Key, and Value using weights matrices (shared across inputs)

Query: $\boldsymbol{q}^i = \boldsymbol{W}^q \boldsymbol{x}^i$

Key: $\boldsymbol{k}^i = \boldsymbol{W}^k \boldsymbol{x}^i$

Value: $\boldsymbol{v}^i = \boldsymbol{W}^v \boldsymbol{x}^i$

Masked Self-Attention Layer



Step 2: Compute the attention scores:

$$\alpha_{2j} = \boldsymbol{k}^j \cdot \boldsymbol{q}^2 = (\boldsymbol{k}^j)^\top \boldsymbol{q}^2$$

Step 1: Linear Projection

Transform input vectors into Query, Key, and Value using weights matrices (shared across inputs)

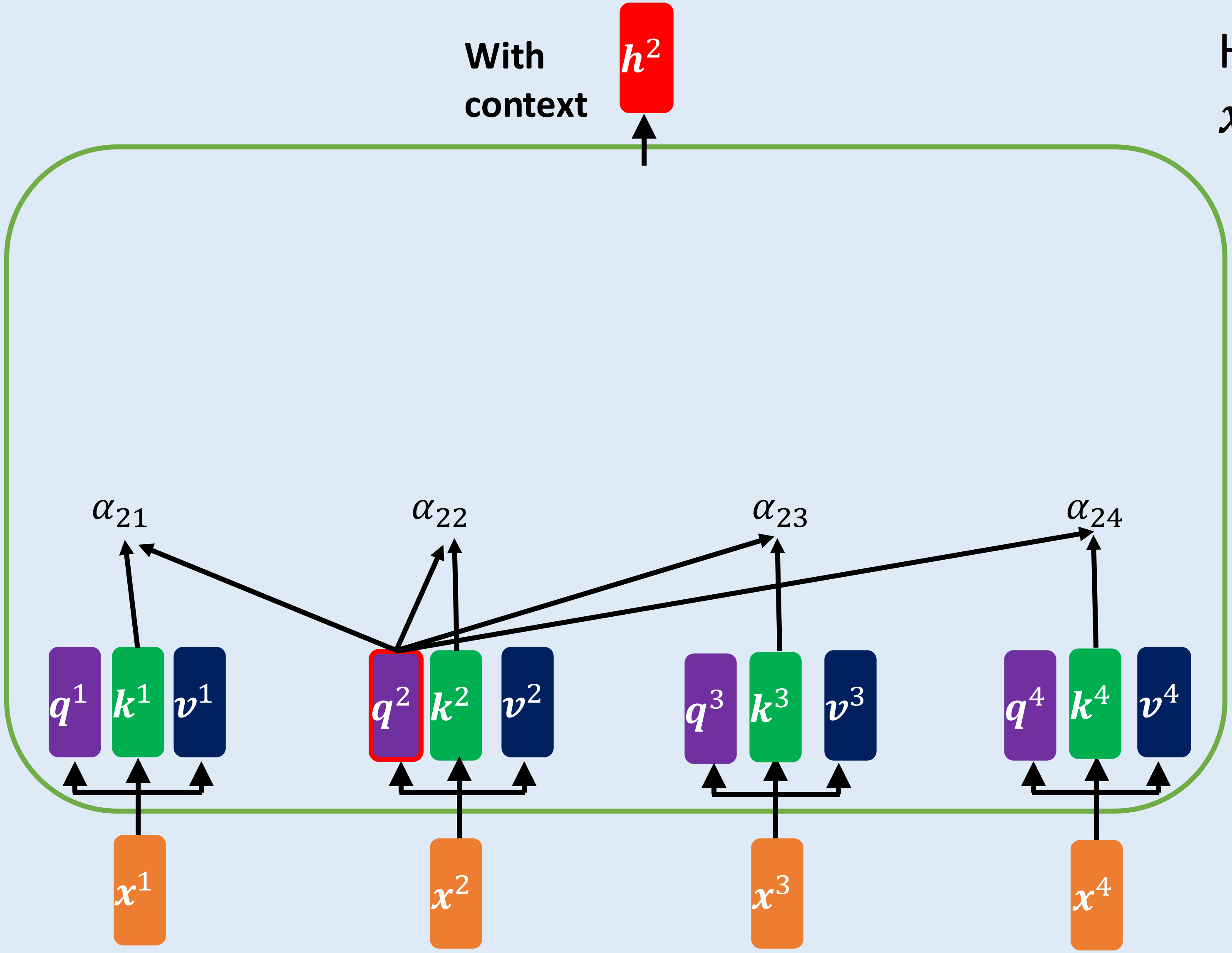
Query: $\boldsymbol{q}^i = \boldsymbol{W}^q \boldsymbol{x}^i$

Key: $\boldsymbol{k}^i = \boldsymbol{W}^k \boldsymbol{x}^i$

Value: $\boldsymbol{v}^i = \boldsymbol{W}^v \boldsymbol{x}^i$

Poll Everywhere

PollEv.com/conghuihu365



How should the mask be set so that only $\boldsymbol{x}^1$ and $\boldsymbol{x}^2$ are used to generate $\boldsymbol{h}^2$?

Step 3: Add mask to attention scores

Step 2: Compute the attention scores:

$\alpha_{2j} = \boldsymbol{k}^j \cdot \boldsymbol{q}^2 = (\boldsymbol{k}^j)^\top \boldsymbol{q}^2$

Step 1: Linear Projection

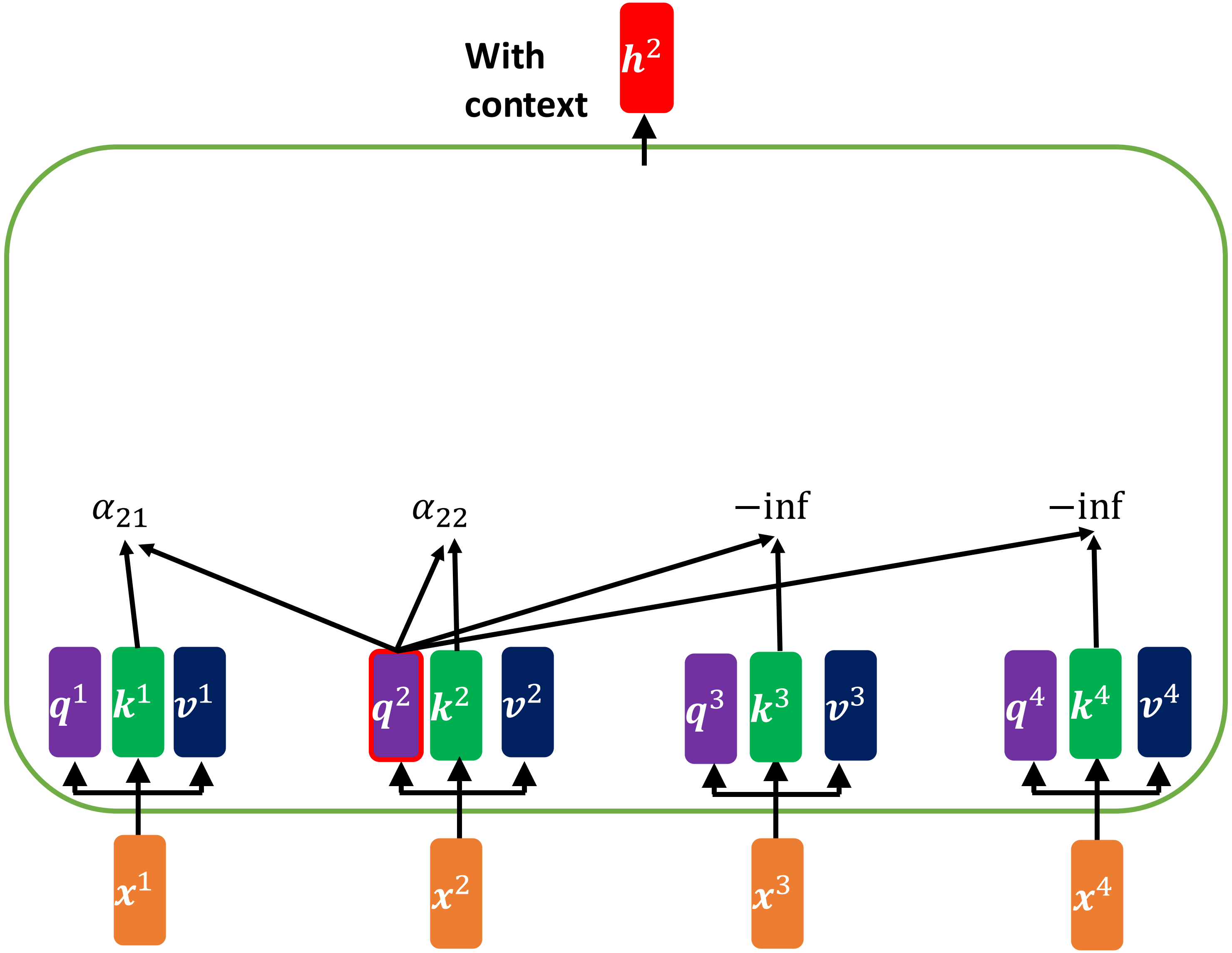
Transform input vectors into Query, Key, and Value using weights matrices (shared across inputs)

Query: $\boldsymbol{q}^i = \boldsymbol{W}^q \boldsymbol{x}^i$

Key: $\boldsymbol{k}^i = \boldsymbol{W}^k \boldsymbol{x}^i$

Value: $\boldsymbol{v}^i = \boldsymbol{W}^v \boldsymbol{x}^i$

Masked Self-Attention Layer



Step 3: Add mask to attention scores:
$[0, 0, -\inf, -\inf]$

Step 2: Compute the attention scores:
$\alpha_{2j} = \boldsymbol{k}^j \cdot \boldsymbol{q}^2 = (\boldsymbol{k}^j)^\top \boldsymbol{q}^2$

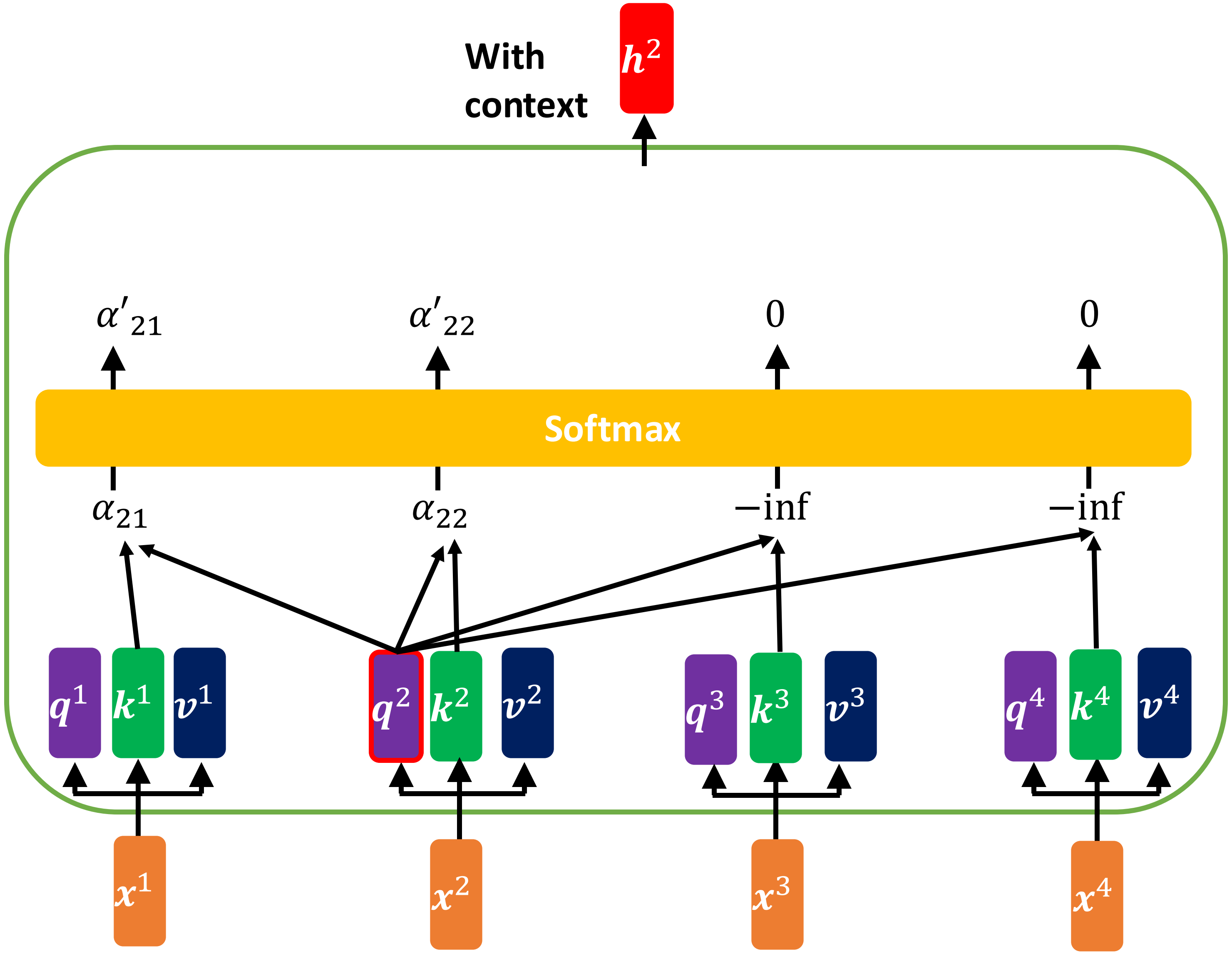
Step 1: Linear Projection
Transform input vectors into Query, Key, and Value using weights matrices (shared across inputs)

Query: $\boldsymbol{q}^i = \boldsymbol{W}^q \boldsymbol{x}^i$

Key: $\boldsymbol{k}^i = \boldsymbol{W}^k \boldsymbol{x}^i$

Value: $\boldsymbol{v}^i = \boldsymbol{W}^v \boldsymbol{x}^i$

Masked Self-Attention Layer



Step 4: Apply Softmax: $\alpha'_{2j} = \frac{e^{\alpha_{2j}}}{\sum_j e^{\alpha_{2j}}}$ $e^{-\mathrm{inf}} = 0$

Step 3: Add mask to attention scores:
$[0, 0, -\mathrm{inf}, -\mathrm{inf}]$

Step 2: Compute the attention scores:
$\alpha_{2j} = \boldsymbol{k}^j \cdot \boldsymbol{q}^2 = (\boldsymbol{k}^j)^\top \boldsymbol{q}^2$

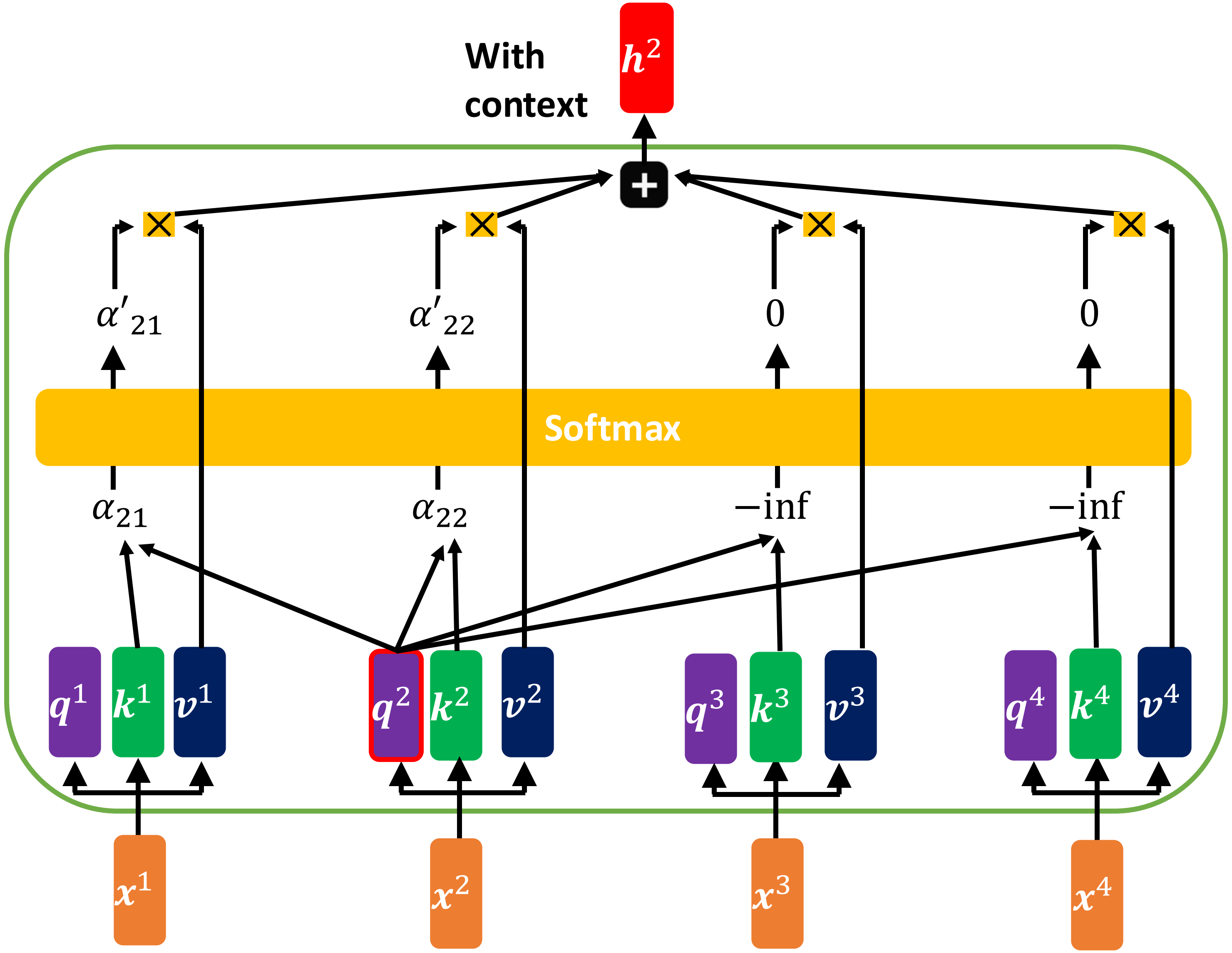
Step 1: Linear Projection
Transform input vectors into Query, Key, and Value using weights matrices (shared across inputs)

Query: $\boldsymbol{q}^i = \boldsymbol{W}^q \boldsymbol{x}^i$

Key: $\boldsymbol{k}^i = \boldsymbol{W}^k \boldsymbol{x}^i$

Value: $\boldsymbol{v}^i = \boldsymbol{W}^v \boldsymbol{x}^i$

Masked Self-Attention Layer



Step 5: Aggregate information: Multiply Values by attention score (after Softmax) $\boldsymbol{h}^2 = \sum_j \alpha'_{2j} \boldsymbol{v}^j$

Step 4: Apply Softmax: $\alpha'_{2j} = \frac{e^{\alpha_{2j}}}{\sum_j e^{\alpha_{2j}}}$ $\quad e^{-\text{inf}} = 0$

Step 3: Add mask to attention scores: $[0, 0, -\text{inf}, -\text{inf}]$

Step 2: Compute the attention scores: $\alpha_{2j} = \boldsymbol{k}^j \cdot \boldsymbol{q}^2 = (\boldsymbol{k}^j)^\top \boldsymbol{q}^2$

Step 1: Linear Projection

Transform input vectors into Query, Key, and Value using weights matrices (shared across inputs)

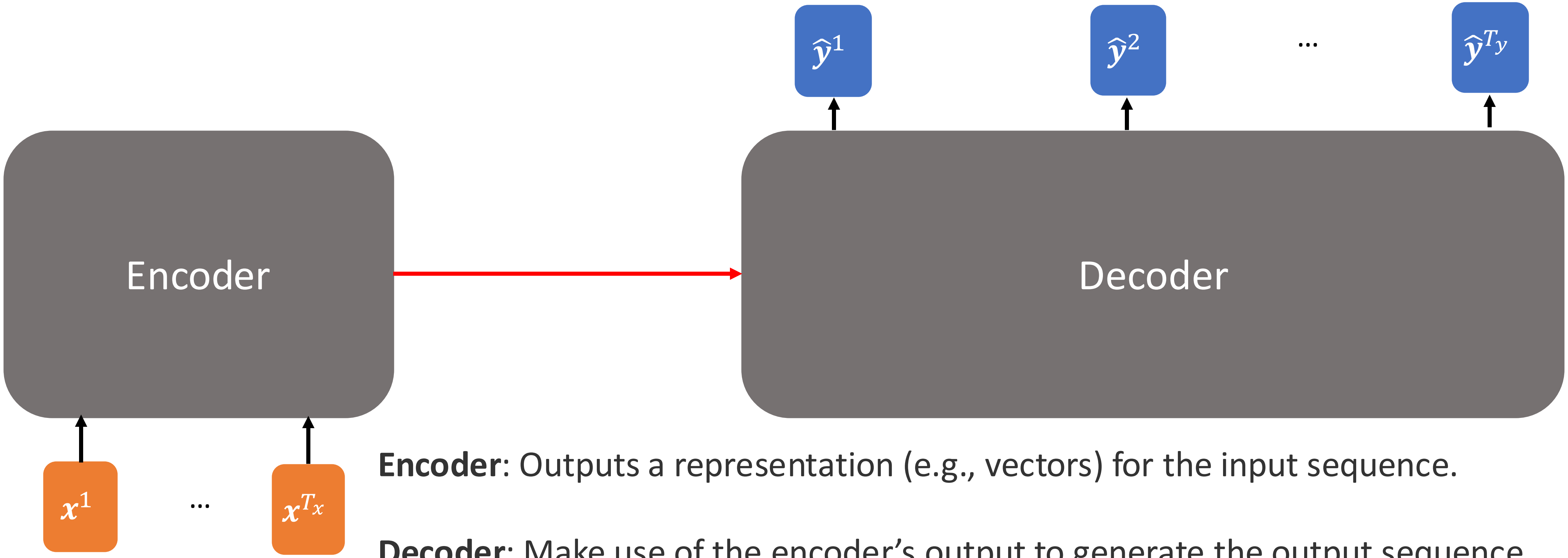
Query: $\boldsymbol{q}^i = \boldsymbol{W}^q \boldsymbol{x}^i$

Key: $\boldsymbol{k}^i = \boldsymbol{W}^k \boldsymbol{x}^i$

Value: $\boldsymbol{v}^i = \boldsymbol{W}^v \boldsymbol{x}^i$

Attention Neural Network: Many-to-Many

$T_x \neq T_y, T_x > 1, T_y > 1$

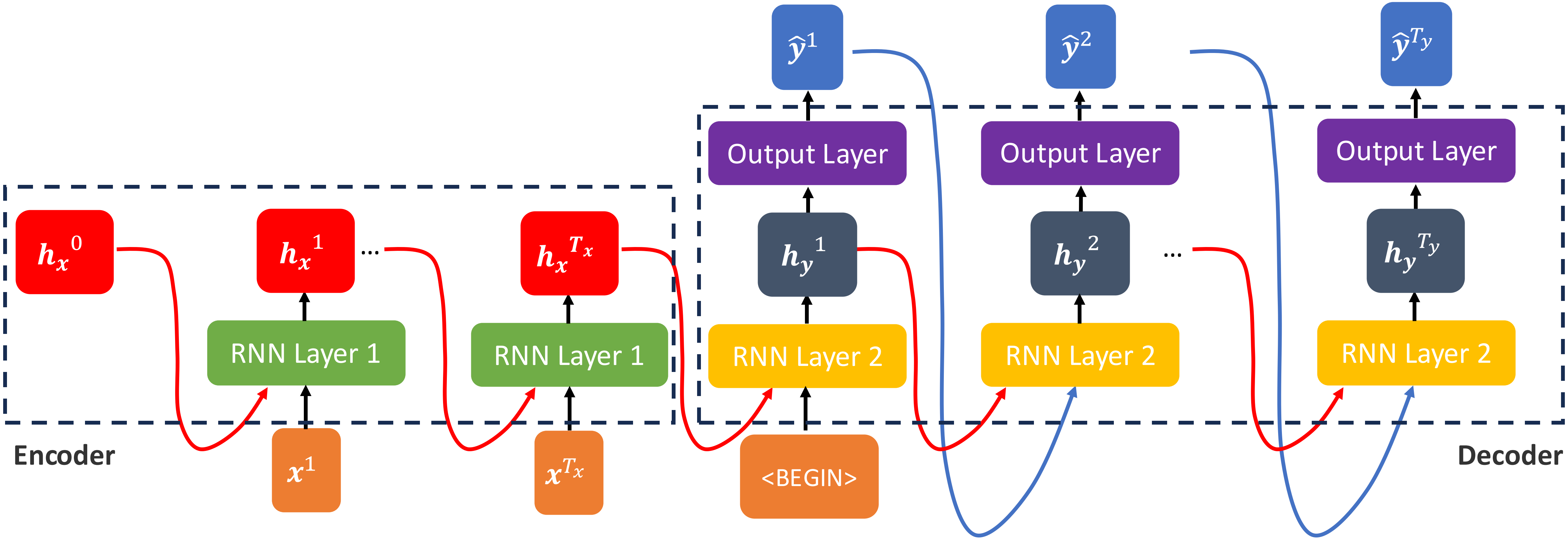


Encoder: Outputs a representation (e.g., vectors) for the input sequence.

Decoder: Make use of the encoder's output to generate the output sequence.

Recurrent Neural Network: Many-to-Many

$T_x \neq T_y, T_x > 1, T_y > 1$



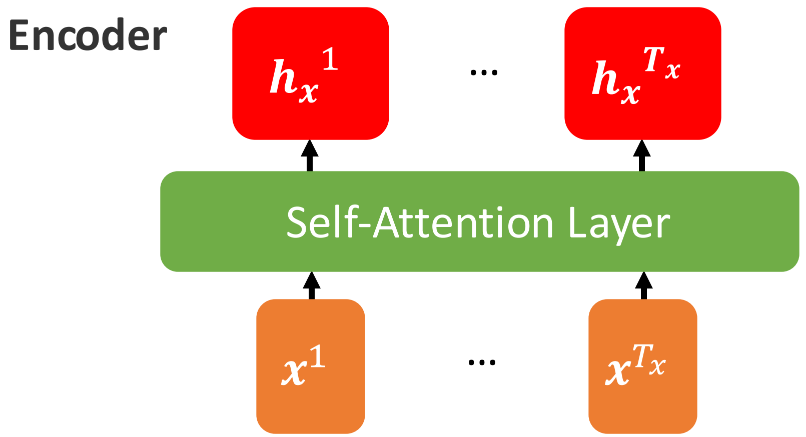
<BEGIN> should be added into the vocabulary and encoded like other words.

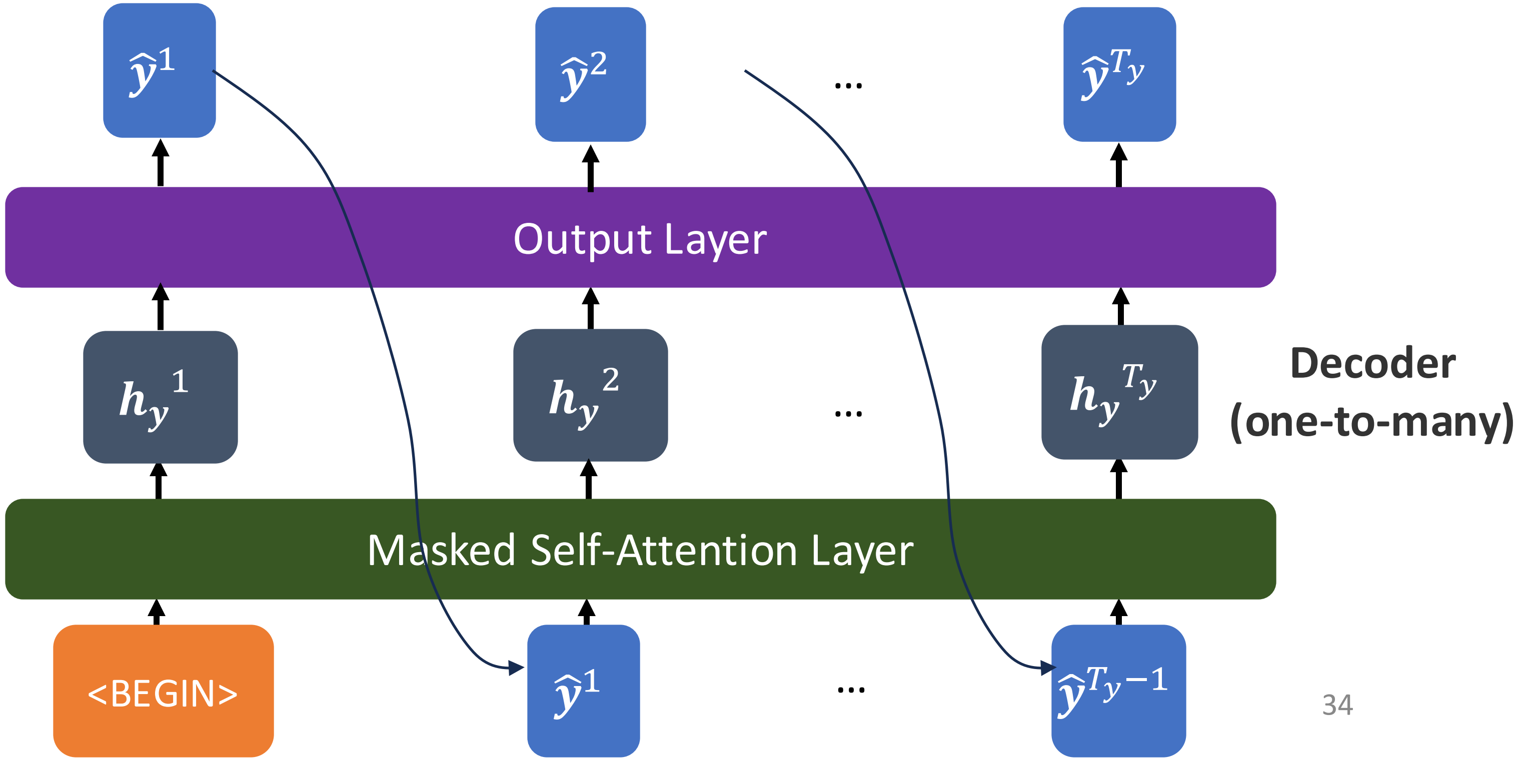
Attention Neural Network: Many-to-Many

$T_x \neq T_y, T_x > 1, T_y > 1$

How can the input information be utilized by the Decoder during prediction?

Use Cross-Attention Layer!



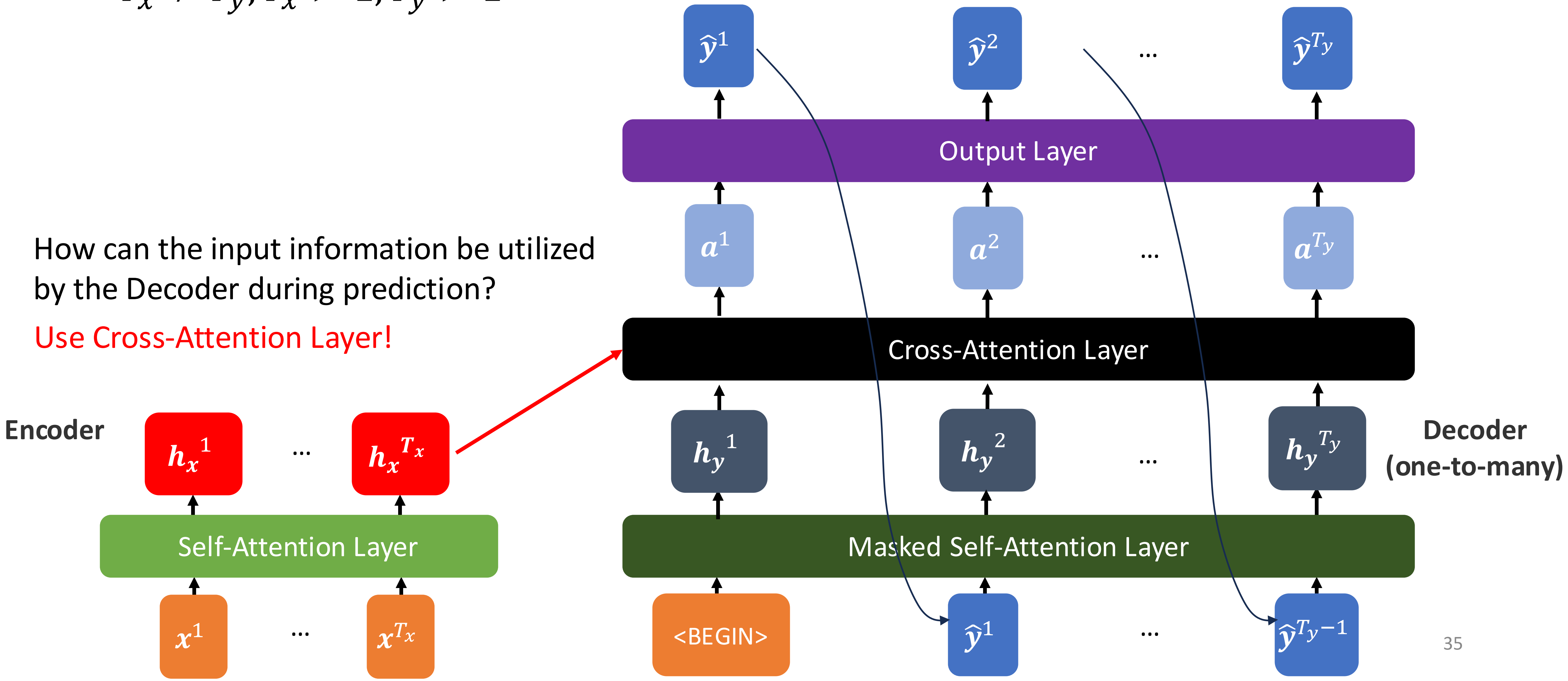


Attention Neural Network: Many-to-Many

$T_x \neq T_y, T_x > 1, T_y > 1$

How can the input information be utilized by the Decoder during prediction?

Use Cross-Attention Layer!



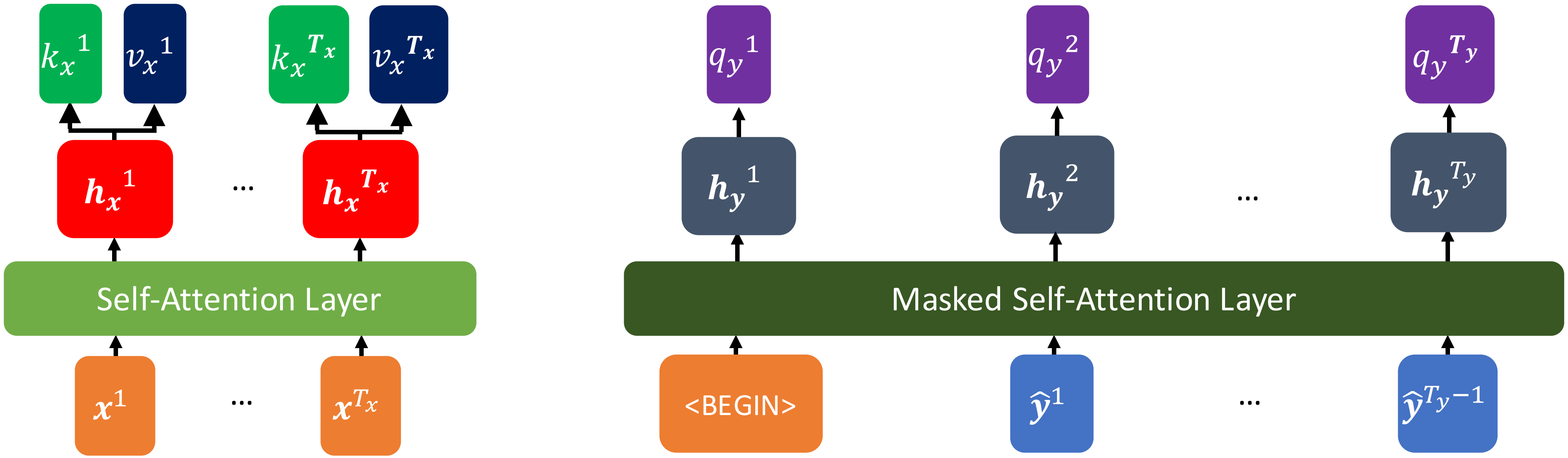
Cross-Attention Layer

Step 1: Linear Projection

Transform input vectors into Query, Key, and Value

Key: $\boldsymbol{k_x}^i = \boldsymbol{W}^k \boldsymbol{h_x}^i$ Query: $\boldsymbol{q_y}^i = \boldsymbol{W}^q \boldsymbol{h_y}^i$

Value: $\boldsymbol{v_x}^i = \boldsymbol{W}^v \boldsymbol{h_x}^i$



Cross-Attention Layer

Step 1: Linear Projection

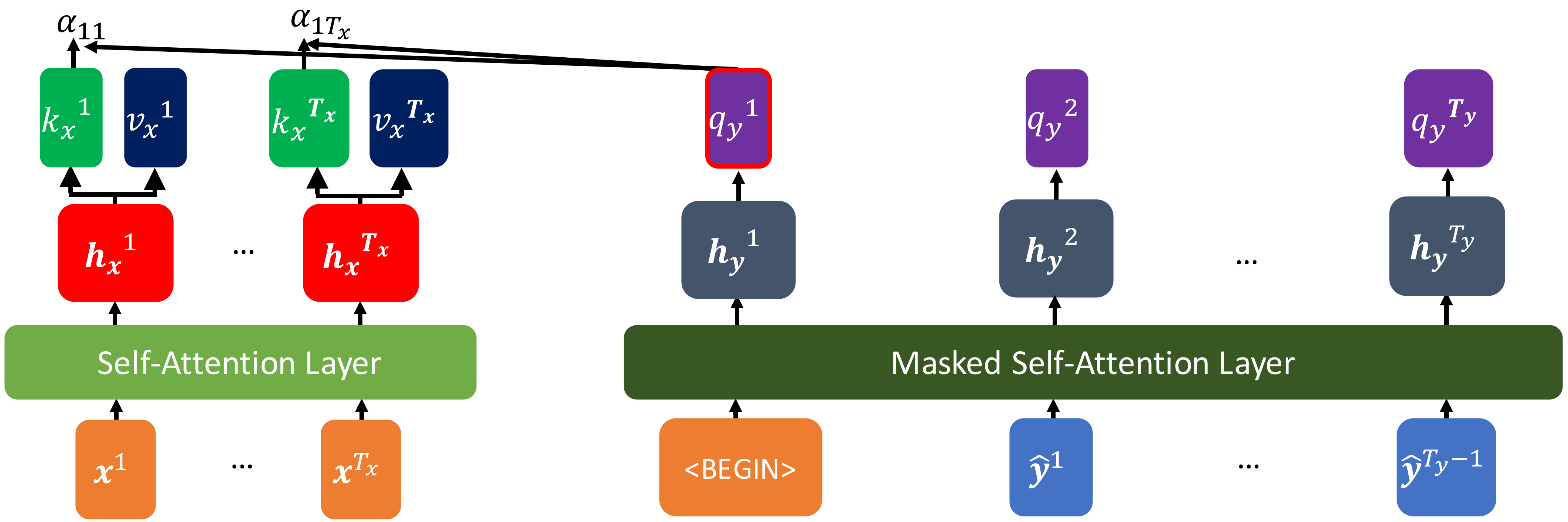
Transform input vectors into Query, Key, and Value

Key: $\boldsymbol{k_x}^i = \boldsymbol{W}^k \boldsymbol{h_x}^i$ Query: $\boldsymbol{q_y}^i = \boldsymbol{W}^q \boldsymbol{h_y}^i$

Value: $\boldsymbol{v_x}^i = \boldsymbol{W}^v \boldsymbol{h_x}^i$

Step 2: Compute the attention scores:

$$\alpha_{1j} = \boldsymbol{k_x}^j \cdot \boldsymbol{q_y}^1 = (\boldsymbol{k_x}^j)^\top \boldsymbol{q_y}^1$$



Cross-Attention Layer

Step 1: Linear Projection

Transform input vectors into Query, Key, and Value

Key: $\boldsymbol{k_x}^i = \boldsymbol{W}^k \boldsymbol{h_x}^i$ Query: $\boldsymbol{q_y}^i = \boldsymbol{W}^q \boldsymbol{h_y}^i$

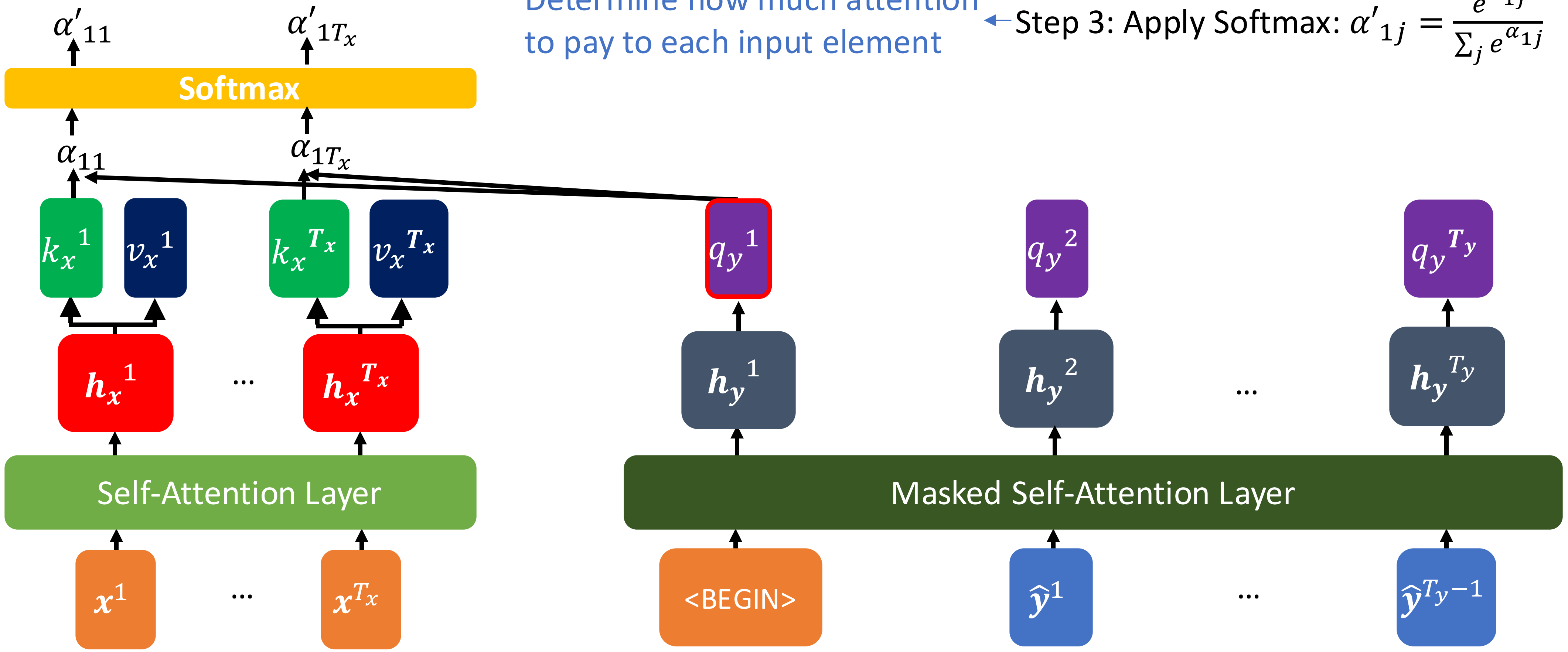
Value: $\boldsymbol{v_x}^i = \boldsymbol{W}^v \boldsymbol{h_x}^i$

Step 2: Compute the attention scores:

$\alpha_{1j} = \boldsymbol{k_x}^j \cdot \boldsymbol{q_y}^1 = (\boldsymbol{k_x}^j)^\top \boldsymbol{q_y}^1$

Determine how much attention to pay to each input element

Step 3: Apply Softmax: $\alpha'_{1j} = \frac{e^{\alpha_{1j}}}{\sum_j e^{\alpha_{1j}}}$



Cross-Attention Layer

Step 1: Linear Projection

Transform input vectors into Query, Key, and Value

Key: $\boldsymbol{k_x}^i = \boldsymbol{W}^k \boldsymbol{h_x}^i$ Query: $\boldsymbol{q_y}^i = \boldsymbol{W}^q \boldsymbol{h_y}^i$

Value: $\boldsymbol{v_x}^i = \boldsymbol{W}^v \boldsymbol{h_x}^i$

Step 2: Compute the attention scores:

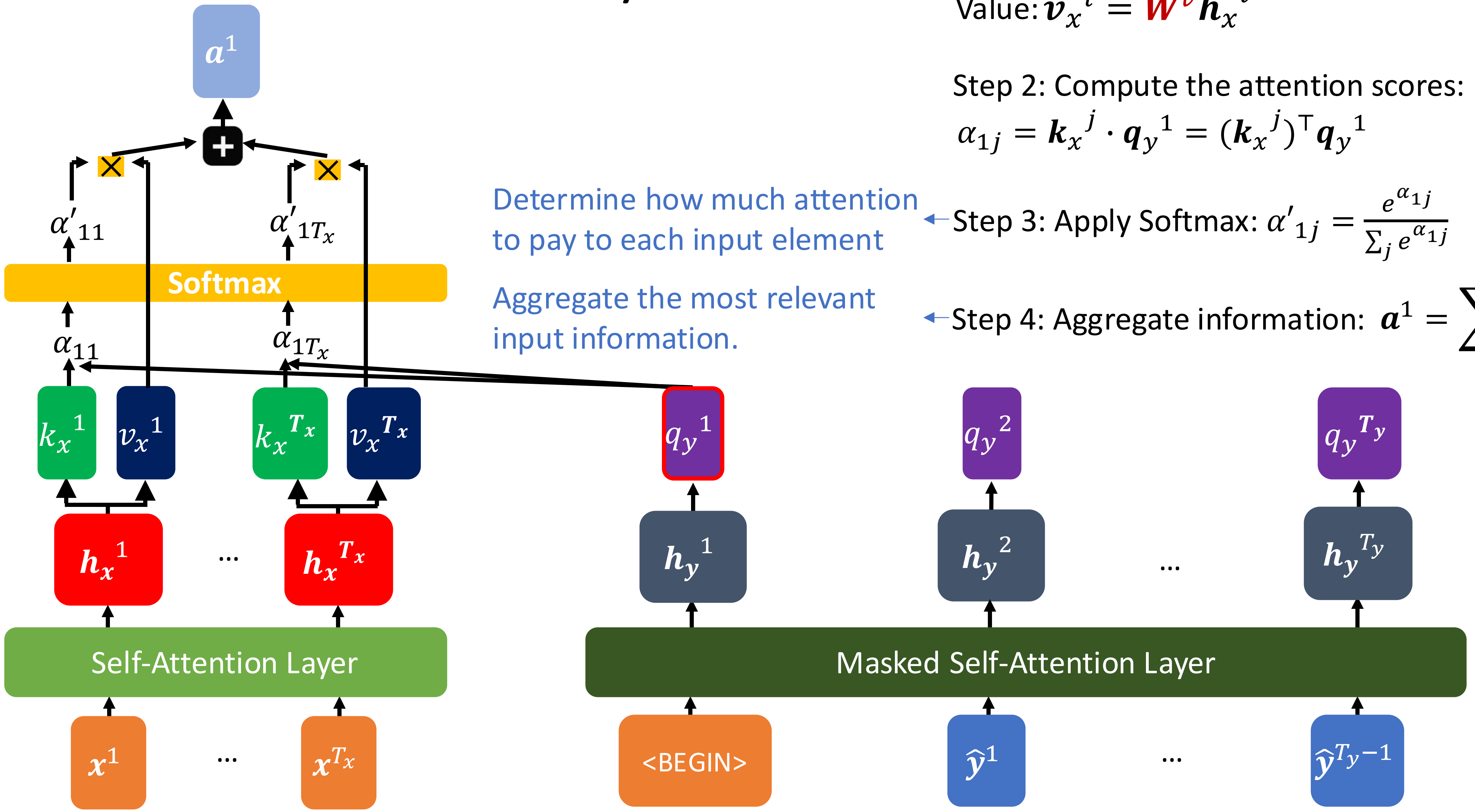
$$\alpha_{1j} = \boldsymbol{k_x}^j \cdot \boldsymbol{q_y}^1 = (\boldsymbol{k_x}^j)^\top \boldsymbol{q_y}^1$$

Determine how much attention to pay to each input element

Step 3: Apply Softmax: $\alpha'_{1j} = \frac{e^{\alpha_{1j}}}{\sum_j e^{\alpha_{1j}}}$

Aggregate the most relevant input information.

Step 4: Aggregate information: $\boldsymbol{a}^1 = \sum_j \alpha'_{1j} \boldsymbol{v_x}^j$



Cross-Attention Layer

Step 1: Linear Projection

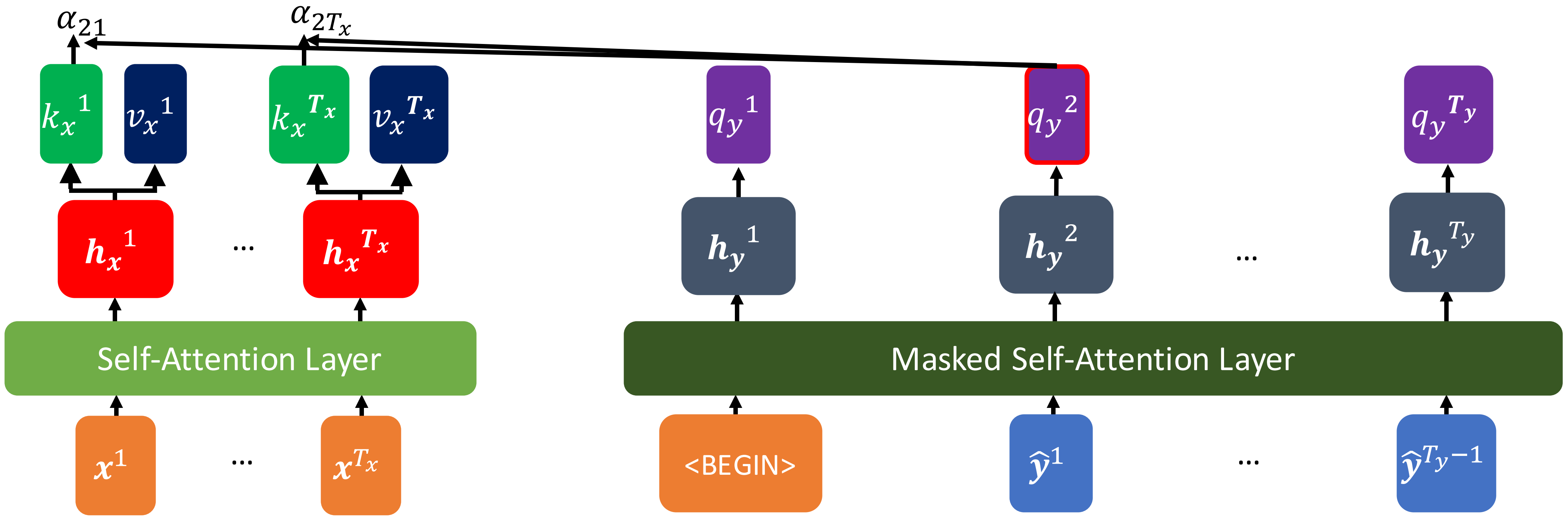
Transform input vectors into Query, Key, and Value

Key: $\boldsymbol{k_x}^i = \boldsymbol{W}^k \boldsymbol{h_x}^i$ Query: $\boldsymbol{q_y}^i = \boldsymbol{W}^q \boldsymbol{h_y}^i$

Value: $\boldsymbol{v_x}^i = \boldsymbol{W}^v \boldsymbol{h_x}^i$

Step 2: Compute the attention scores:

$$\alpha_{2j} = \boldsymbol{k_x}^j \cdot \boldsymbol{q_y}^1 = (\boldsymbol{k_x}^j)^\top \boldsymbol{q_y}^1$$



Cross-Attention Layer

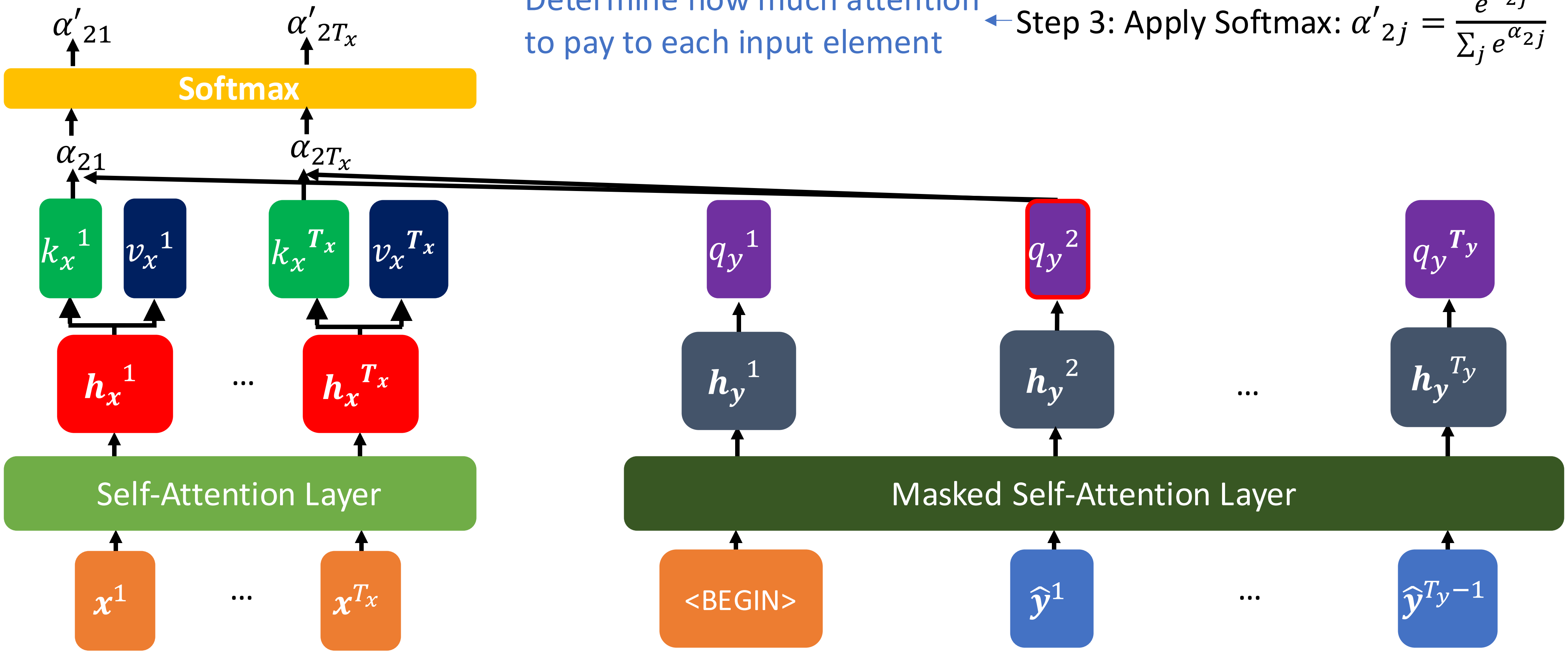
Step 1: Linear Projection
Transform input vectors into Query, Key, and Value

Key: $\boldsymbol{k_x}^i = \boldsymbol{W}^k \boldsymbol{h_x}^i$ Query: $\boldsymbol{q_y}^i = \boldsymbol{W}^q \boldsymbol{h_y}^i$

Value: $\boldsymbol{v_x}^i = \boldsymbol{W}^v \boldsymbol{h_x}^i$

Step 2: Compute the attention scores:
$\alpha_{2j} = \boldsymbol{k_x}^j \cdot \boldsymbol{q_y}^1 = (\boldsymbol{k_x}^j)^\top \boldsymbol{q_y}^1$

Determine how much attention to pay to each input element ← Step 3: Apply Softmax: $\alpha'_{2j} = \frac{e^{\alpha_{2j}}}{\sum_j e^{\alpha_{2j}}}$



Cross-Attention Layer

Step 1: Linear Projection

Transform input vectors into Query, Key, and Value

Key: ${\boldsymbol{k}_x}^i = \boldsymbol{W}^k {\boldsymbol{h}_x}^i$ Query: ${\boldsymbol{q}_y}^i = \boldsymbol{W}^q {\boldsymbol{h}_y}^i$

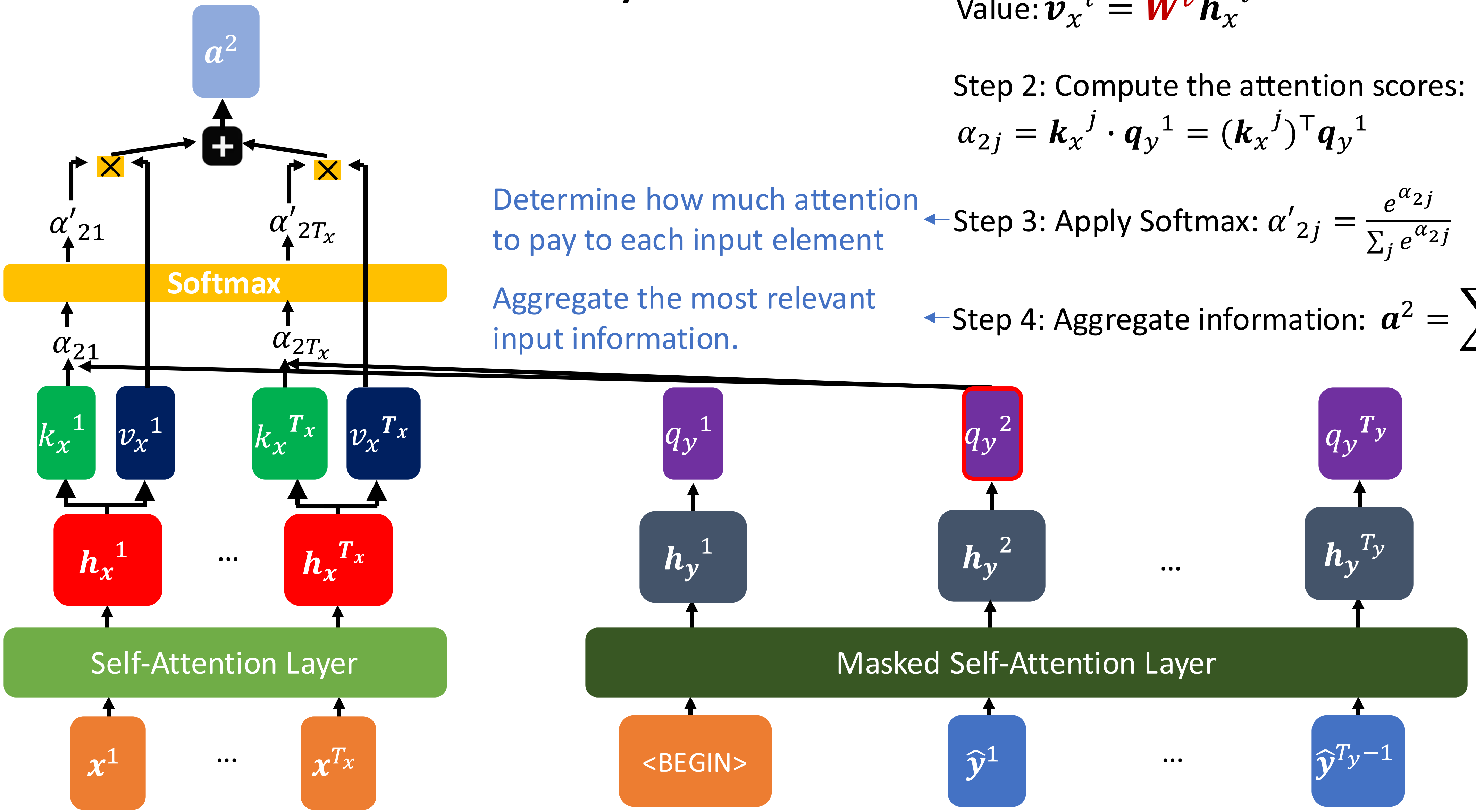
Value: ${\boldsymbol{v}_x}^i = \boldsymbol{W}^v {\boldsymbol{h}_x}^i$

Step 2: Compute the attention scores:

$\alpha_{2j} = {\boldsymbol{k}_x}^j \cdot {\boldsymbol{q}_y}^1 = ({\boldsymbol{k}_x}^j)^\top {\boldsymbol{q}_y}^1$

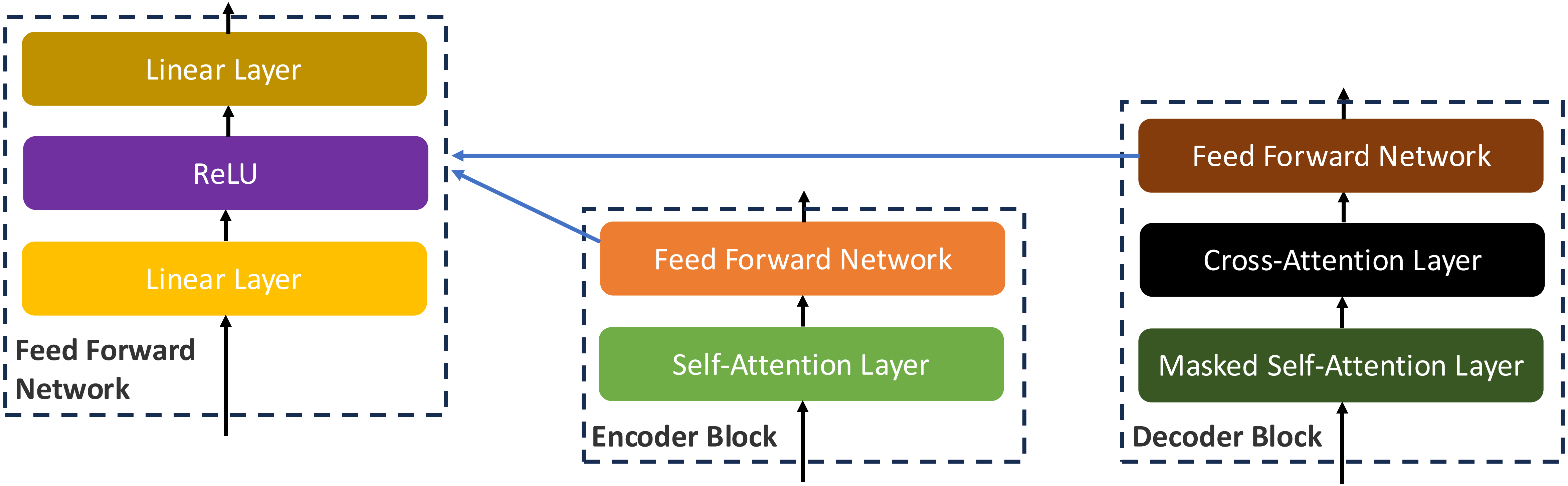
Step 3: Apply Softmax: $\alpha'_{2j} = \frac{e^{\alpha_{2j}}}{\sum_j e^{\alpha_{2j}}}$

Step 4: Aggregate information: $\boldsymbol{a}^2 = \sum_j \alpha'_{2j} {\boldsymbol{v}_x}^j$



Transformer

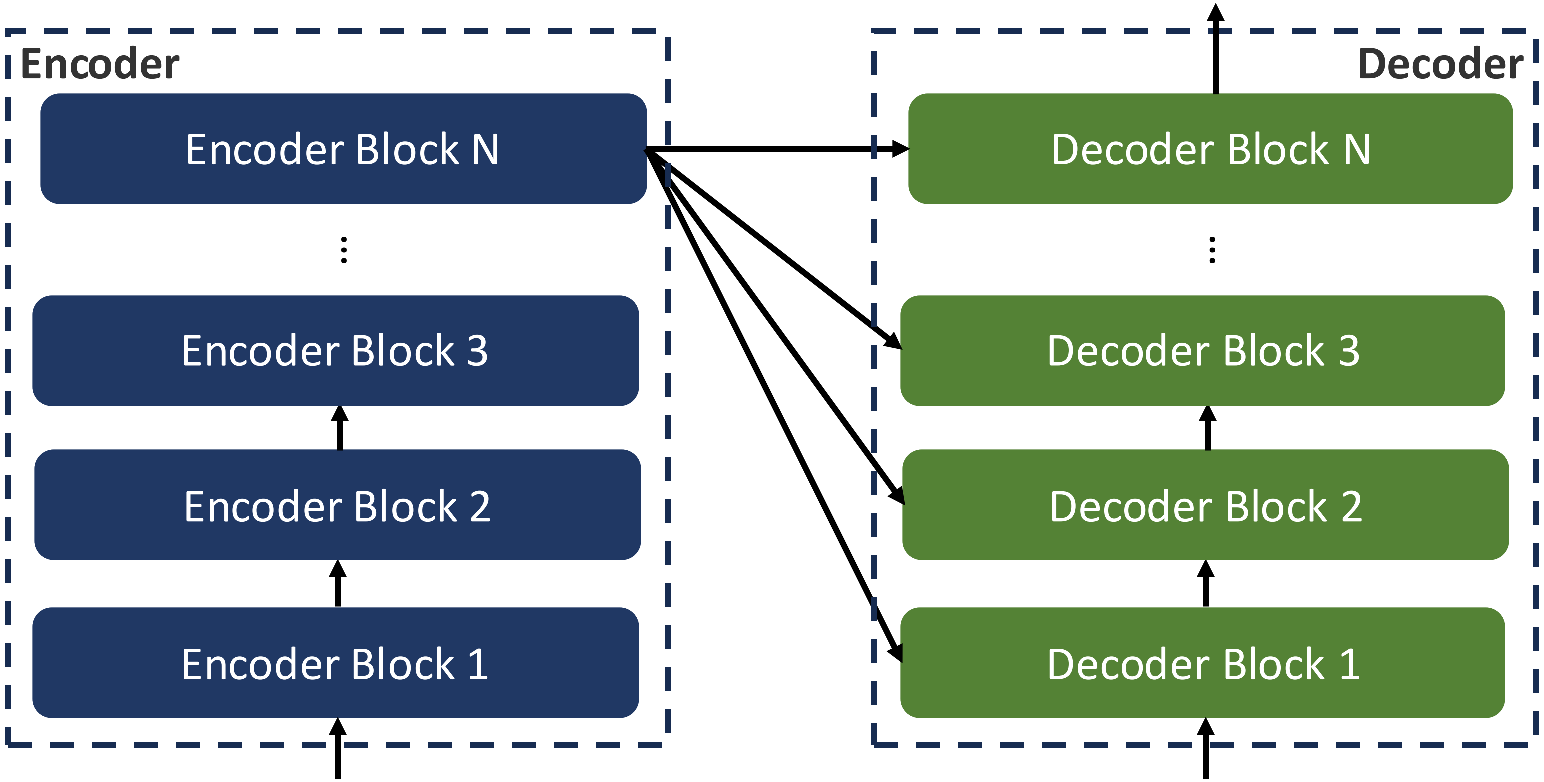
- Deep attention neural network
- Basic building blocks: Encoder Block and Decoder Block



Vaswani, Ashish, et al. Attention is all you need. NeurIPS, 2017.

Transformer

- Encoder: A stack of Encoder Blocks
- Decoder: A stack of Decoder Blocks

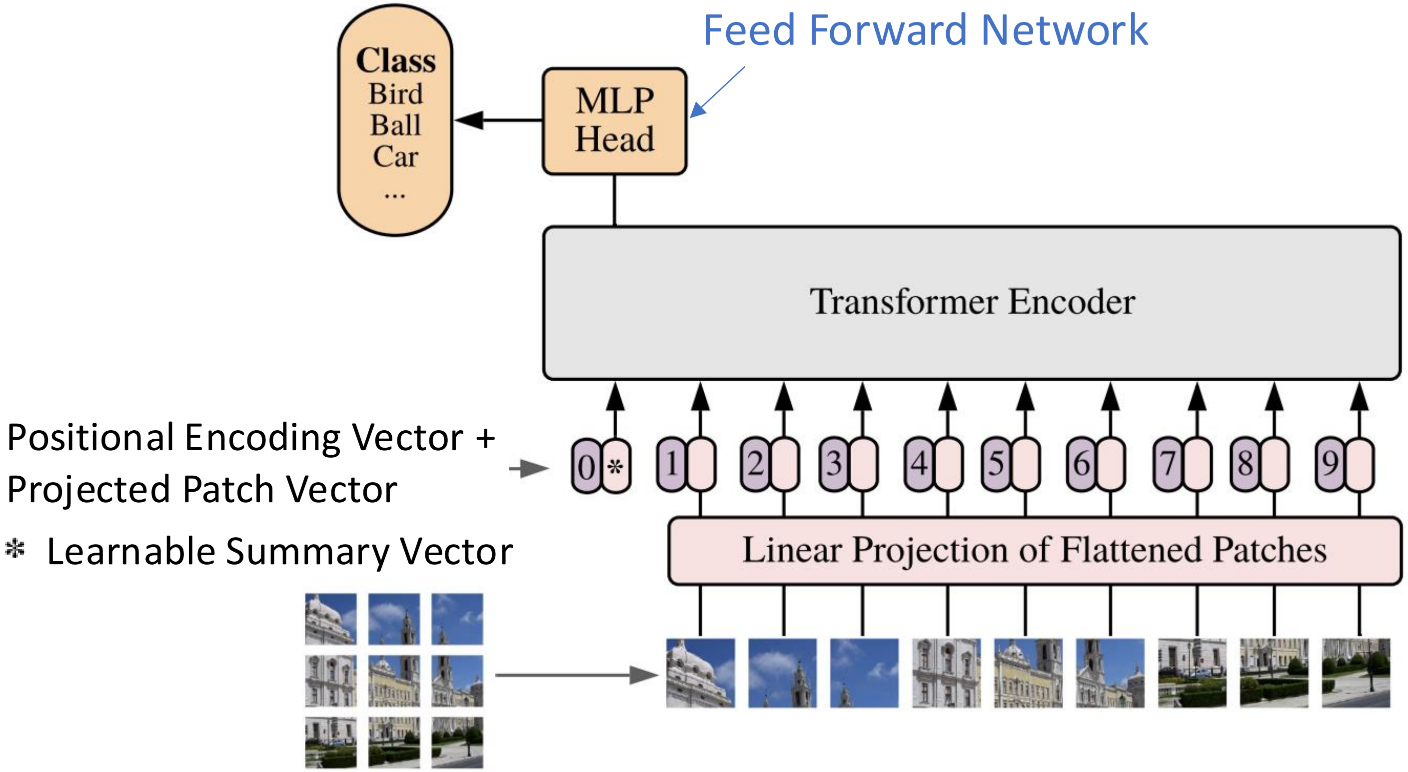


Generative Pretrained Transformers (GPT) are built on the transformer architecture to produce new text based on input prompts.

Vaswani, Ashish, et al. Attention is all you need. NeurIPS, 2017.

Vision Transformer

Image Classification:



Dosovitskiy, Alexey, et al. An image is worth 16x16 words: Transformers for image recognition at scale. ICLR 2021

Outline

- Attention Neural Networks:
  - Masked Self-Attention Layer
  - Cross-Attention Layer
  - Transformer
- **AI Ethics**
  - Ethical Issues
- Course Recap
  - “Classical” AI
  - “Classical” ML
  - “Modern” ML

AI Ethics

- Ethics: Principles that guide human behaviour, distinguishing right from wrong.

- AI Ethics: Principles that guide developers, manufacturers, authorities, and operators in mitigating the ethical issues arising from AI.

What are the ethical issues?

Sources:
https://www.theguardian.com/technology/2018/oct/10/amazon-hiring-ai-gender-bias-recruiting-engine
https://www.bloomberg.com/graphics/2023-generative-ai-bias/

Biased Decision

- AI can generate biased outputs and make unfair judgements.

Amazon ditched AI recruiting tool that favored men for technical jobs

Specialists had been building computer programs since 2014 to review résumés in an effort to automate the search process

Amazon's automated hiring tool was found to be inadequate after penalizing the résumés of female candidates. Photograph: Brian Snyder/Reuters

Bias in AI Recruiting Tool

Explore Images of Workers Generated by Stable Diffusion

A color photograph of a **lawyer**

STABLE DIFFUSION RESULTS

SKIN TONE	I	II	III	IV	V	VI
SHARE (%)	66	15	8	3	5	2

GENDER	MEN	WOM.	AMB.
SHARE (%)	83	15	2

Bias in Generative AI

Source:
https://www.youtube.com/shorts/dVrHLZtvr5g

Manipulation of Behaviour

- AI can be used to generate fake content to influence human behaviour.

Source: https://www.businessinsider.com/openai-chatgpt-contractors-train-ai-software-engineering-autonomous-vehicles-report-2023-1

Automation and Employment

- AI can replace humans in some jobs.

BUSINESS INSIDER

Newsletters Log in Subscribe

US MARKETS OPEN
In the news

Dow Jones -0.23% Nasdaq -0.24% S&P 500 -0.14% META -0.72% TSLA -2.47% BABA -9.26%

HOME > TECH

ChatGPT creator OpenAI might be training its AI technology to replace some software engineers, report says

Samantha Delouya Jan 28, 2023, 5:06 AM SGT

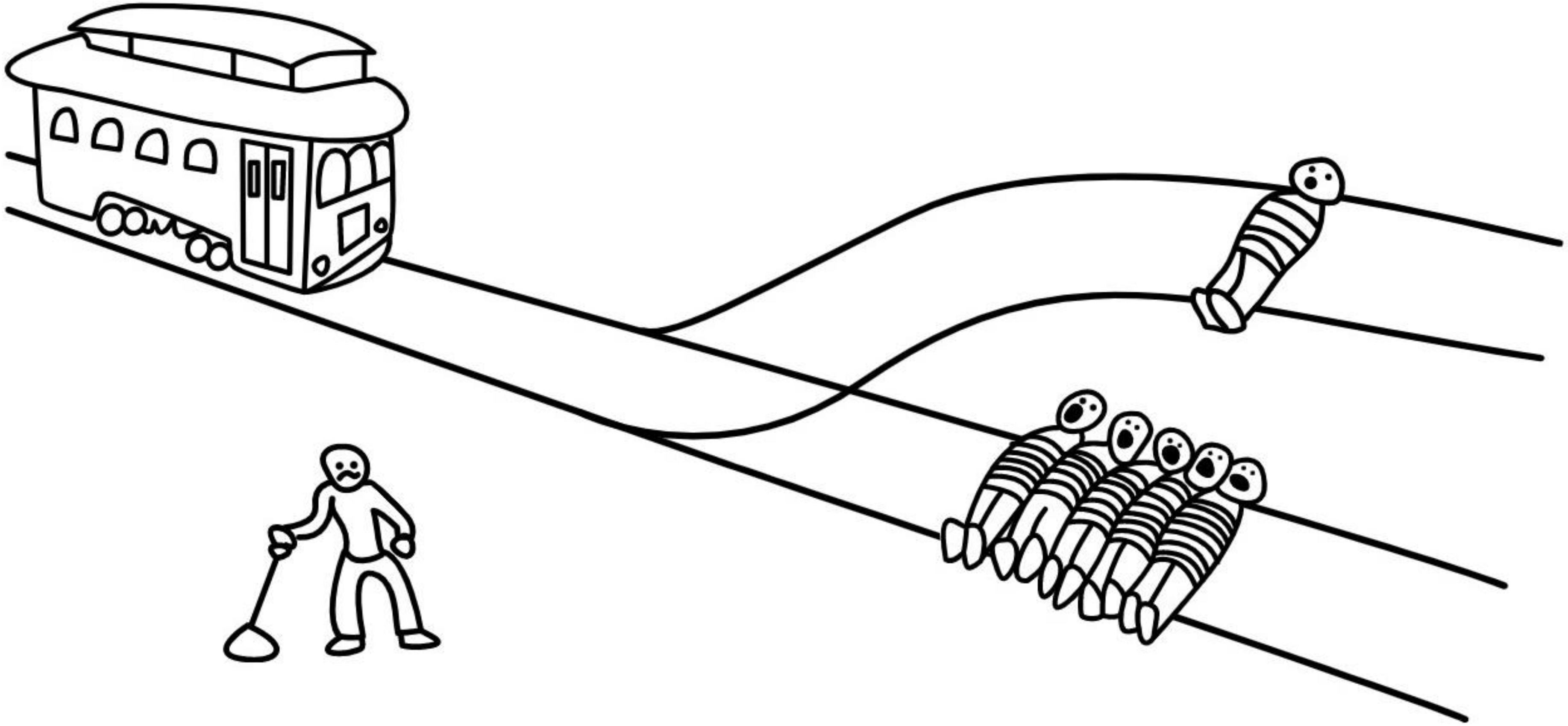
Autonomous Systems

- Autonomous systems can make decision independently.

- When decisions are delegated to autonomous system, two key issues arise:
  - How should autonomous system behave?
  - Who is accountable for harmful actions taken by autonomous systems?

Source:
https://troypancake.substack.com/p/reflections-on-the-trolley-problem

Autonomous Systems: Trolley Problem



Source:
https://www.youtube.com/watch?v=-N_RZJUAQY4

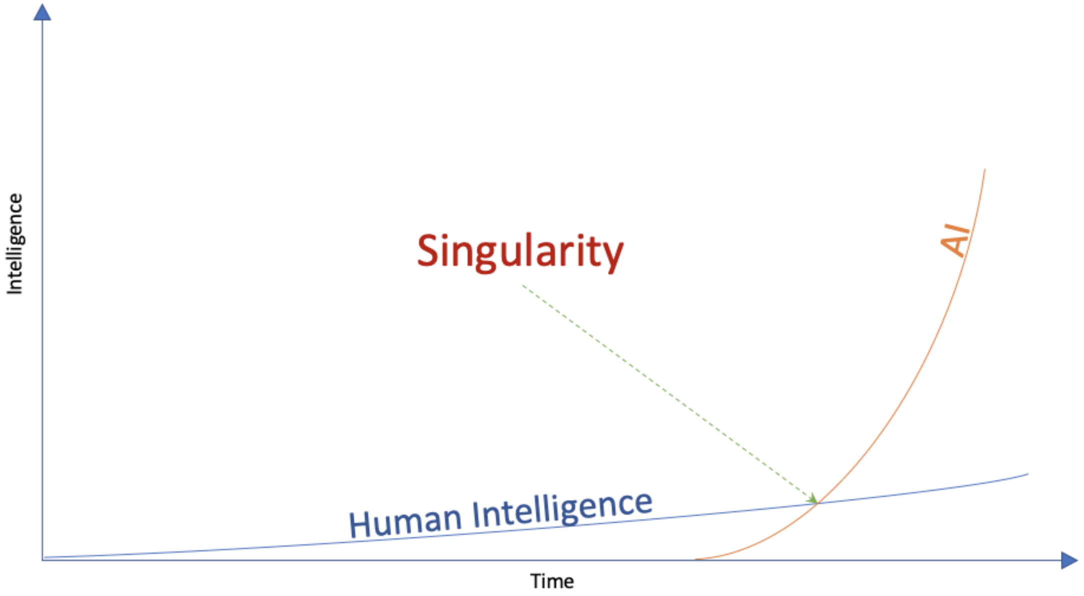
Trolley Problem: A Naïve Solution

Source:
https://www.reddit.com/r/nextfuckinglevel/comments/u7n5wa/solution_to_the_trolley_problem/

Trolley Problem: A Better Solution

Singularity

- The singularity in AI is a hypothetical future point when artificial intelligence not only reaches but exceeds human intelligence.



Will the singularity ever occur? We don't know.
What would happen if it did occur? We still don't know.

Outline

- Attention Neural Networks:
  - Masked Self-Attention Layer
  - Cross-Attention Layer
  - Transformer
- AI Ethics
  - Ethical Issues
- **Course Recap**
  - “Classical” AI
  - “Classical” ML
  - “Modern” ML

"Classical" AI

Agent

- PEAS framework:
  Performance Measure,
  Environment, Actuators, Sensors

- Properties of Task Environment:
  Fully v.s. Partially observable;
  Deterministic v.s. Stochastic;
  Episodic v.s. Sequential

- Agent structures: Reflex, Goal-based, Utility-based, Learning

Search Algorithms

- Uninformed search: BFS, UCS, DFS, DLS, IDS

- Informed search: A* search

- Local search: Hill climbing

- Adversarial search: Minimax, Alph-Beta pruning

"Classical" ML

Supervised Learning

- Tasks: Regression and Classification
- Models: Decision Tree, Linear/Logistic Regression, Support Vector Machine
- Learning Algorithms: Decision tree learning and gradient descent
- Performance Measure: MSE, MAE, Accuracy, Precision, Recall, F1

Unsupervised Learning

- Clustering: K-Means Clustering
- Dimension Reduction: SVD and PCA

Regularization

- Prevent overfitting

Kernel

- Compute feature transformation efficiently

“Modern” ML

Non-sequential data

- Perceptron
- Fully-connected neural network
- Convolution neural network

Sequential data

- Recurrent neural network
- Attention neural network
- Transformer

Backpropagation

Calculate the gradients of the loss function with respect to each weight in the network.

Coming Up Next Week

- **None**

To Do

- **None**
- Actually: PS5 is due **Saturday**

Thanks to the teaching team!

Instructors

Muhammad **Rizki** Maulana

rizki@nus.edu.sg

Deep Reinforcement Learning

Conghui Hu

conghui@nus.edu.sg

Deep Learning, Computer Vision

Patrick Rebentrost

cqtfpr@nus.edu.sg

Quantum Machine Learning

Teaching Assistant

Aprup

Nikhil Sultania

Sarji

Wilson

Ruiheng

Malaika Afra Taj

John Russell Himawan

Jalil

Gavin

Khoi Nguyen

Teaching Assistant

Chenrui Tie

Zihao XU

Ivan

Aditya

Nguyen Pham

Anton

Si Yuan

Pang Yen

Benson

Wai Kin

Teaching Assistant


Laksh

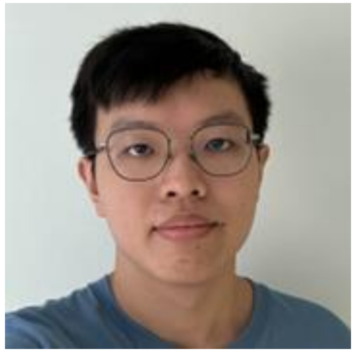

Chun Jie


Thanh Chu


Daryl


Ian


Jiacheng

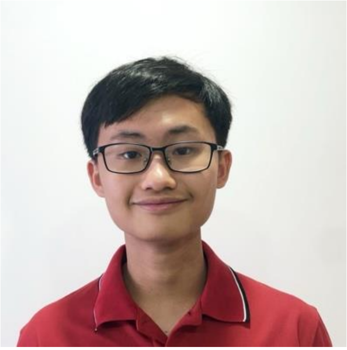

Zheng Long


Diwen


Yi Hong


Phi

Teaching Assistant



Shaun Quek

Kum Chai Yin

Wei Hao

Miguel

Chowdhury Rafeed Rahman

Junting Chen

Lim Jing Yu

Come join the teaching team!

We are open for recruitment

That’s it!