Neural Networks

Transformers

Guest lecture by Dr. Jae Hee Lee

KNOWLEDGE
TECHNOLOGY

http://www.informatik.uni-hamburg.de/WTM/

Outline

- Self-Attention
- Transformer (Architecture, Inference, Training)
- Transformer Applications
- Multimodal Learning

Self-Attention

- Self-Attention
  - Representation Learning
  - Sequence-to-Sequence Models
  - Self-Attention
- Transformers
- Transformer Applications
- Multimodal Learning

Deep Learning as Representation Learning

- Q: What is special about deep learning?

 https://scholar.google.de/citations?view_op=top_venues&hl=en

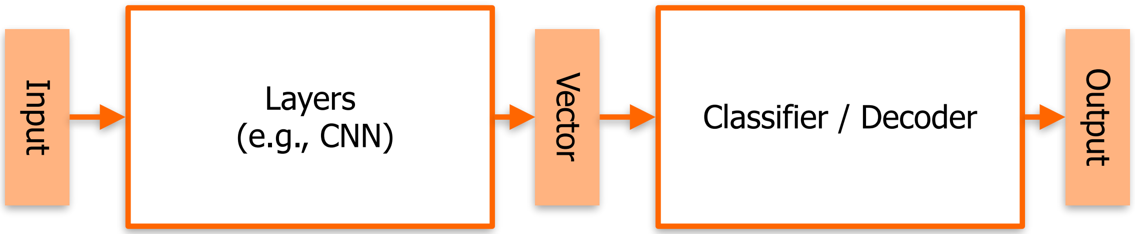
Deep Learning as Representation Learning

- Q: What is special about deep learning?
  - A: Learning meaningful **representations**.

https://scholar.google.de/citations?view_op=top_venues&hl=en

Deep Learning as Representation Learning

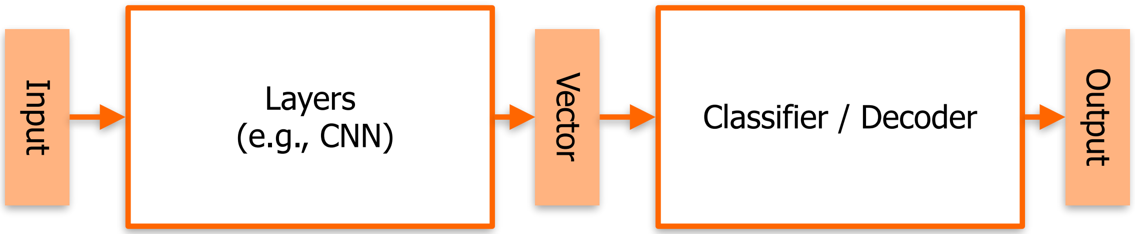
- Q: What is special about deep learning?
  - A: Learning meaningful **representations**.
- In DL, input is transformed to a vector, which
  - contains relevant information to solve a given task and
  - is called a **representation** or a **feature vector.**



https://scholar.google.de/citations?view_op=top_venues&hl=en

Deep Learning as Representation Learning

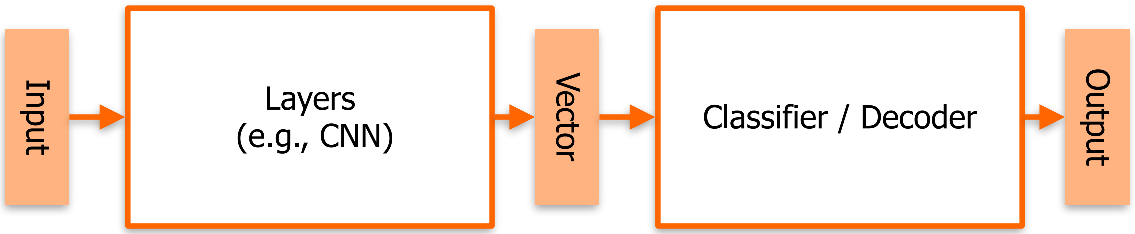
- Q: What is special about deep learning?
  - A: Learning meaningful **representations**.
- In DL, input is transformed to a vector, which
  - contains relevant information to solve a given task and
  - is called a **representation** or a **feature vector.**
- DL research = "**how to learn good representations?**".



https://scholar.google.de/citations?view_op=top_venues&hl=en

Deep Learning as Representation Learning

- Q: What is special about deep learning?
  - A: Learning meaningful **representations**.
- In DL, input is transformed to a vector, which
  - contains relevant information to solve a given task and
  - is called a **representation** or a **feature vector.**
- DL research = "**how to learn good representations?**".

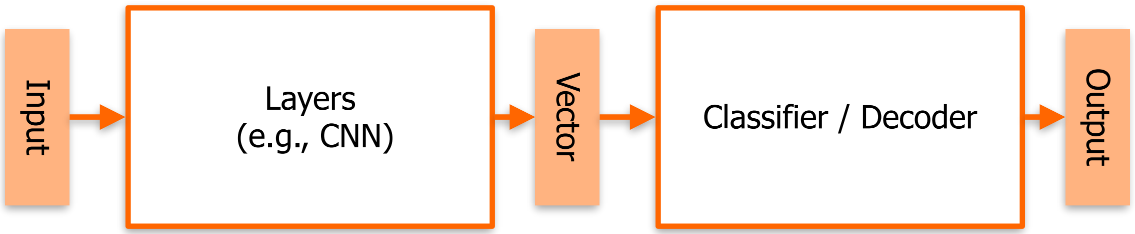


1.	Nature
2.	The New England Journal of Medicine
3.	Science
4.	IEEE/CVF Conference on Computer Vision and Pattern Recognition
5.	The Lancet
6.	Advanced Materials
7.	Nature Communications
8.	Cell
9.	International Conference on Learning Representations
10.	Neural Information Processing Systems

https://scholar.google.de/citations?view_op=top_venues&hl=en

Deep Learning as Representation Learning

- Q: What is special about deep learning?
  - A: Learning meaningful **representations**.
- In DL, input is transformed to a vector, which
  - contains relevant information to solve a given task and
  - is called a **representation** or a **feature vector.**
- DL research = "**how to learn good representations?**".
- Different **building blocks** are introduced to learn good representations.
- In this lecture, we will learn a new building block: **self-attention.**

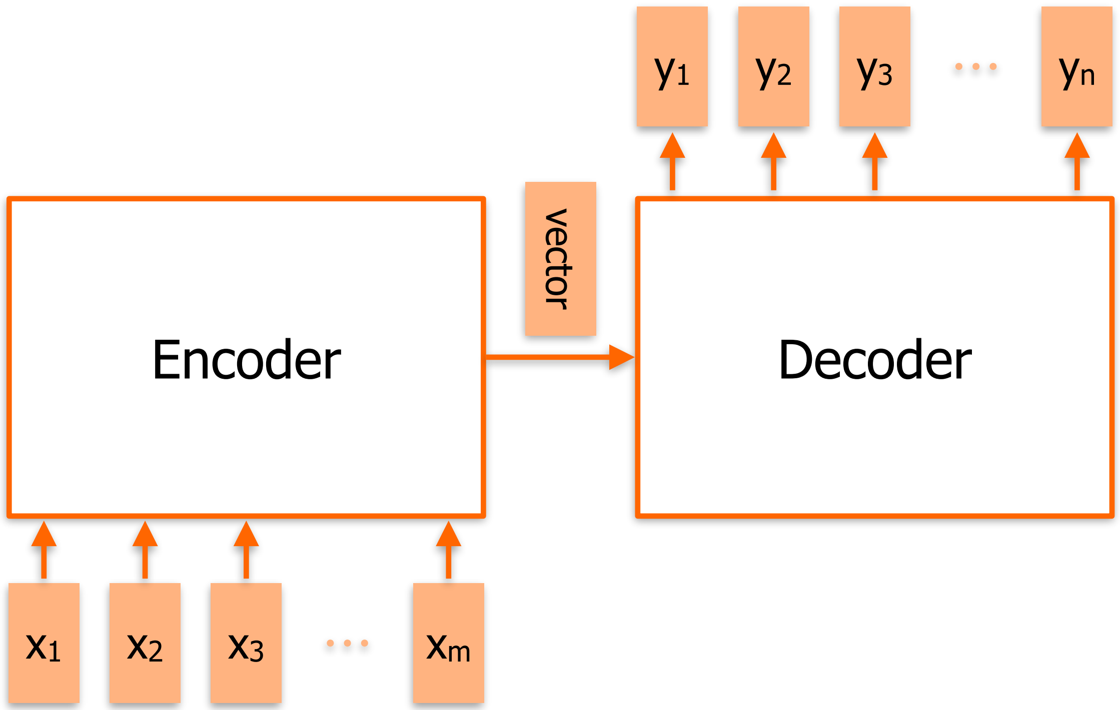


1.	Nature
2.	The New England Journal of Medicine
3.	Science
4.	IEEE/CVF Conference on Computer Vision and Pattern Recognition
5.	The Lancet
6.	Advanced Materials
7.	Nature Communications
8.	Cell
9.	International Conference on Learning Representations
10.	Neural Information Processing Systems

https://scholar.google.de/citations?view_op=top_venues&hl=en

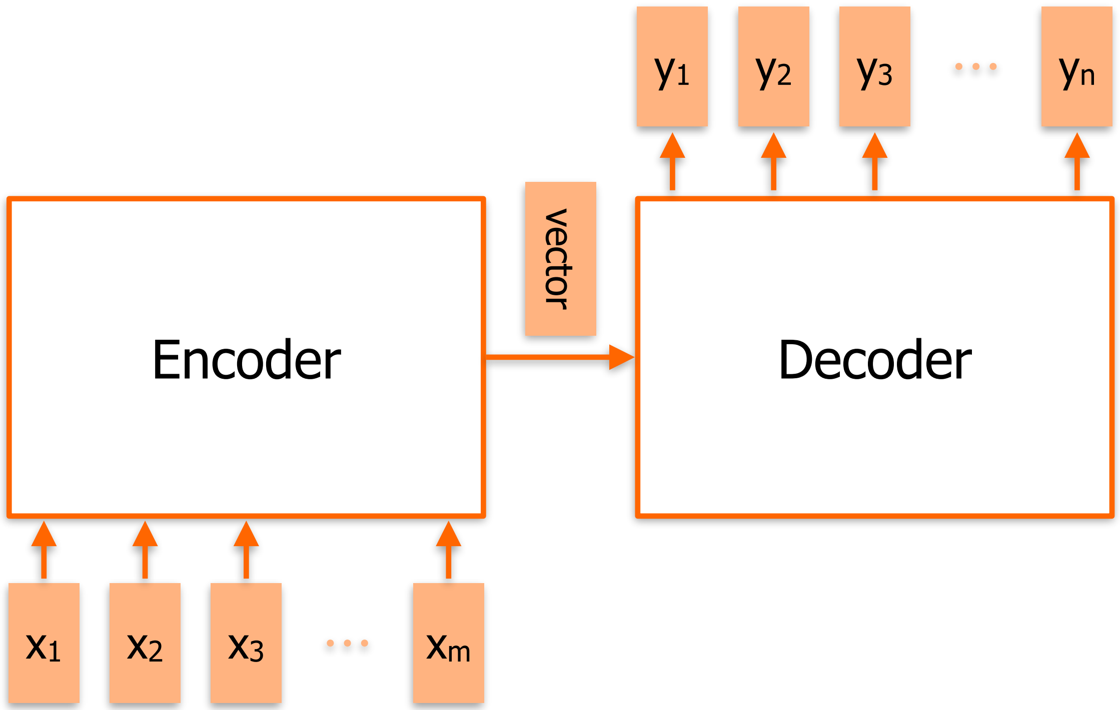
Sequence-to-Sequence Models

- Input: a sequence of vectors.
- Output: a sequence of vectors.
- Example: Machine translation
  - Input: `How are you?`
  - Output: `Wie geht es dir?`



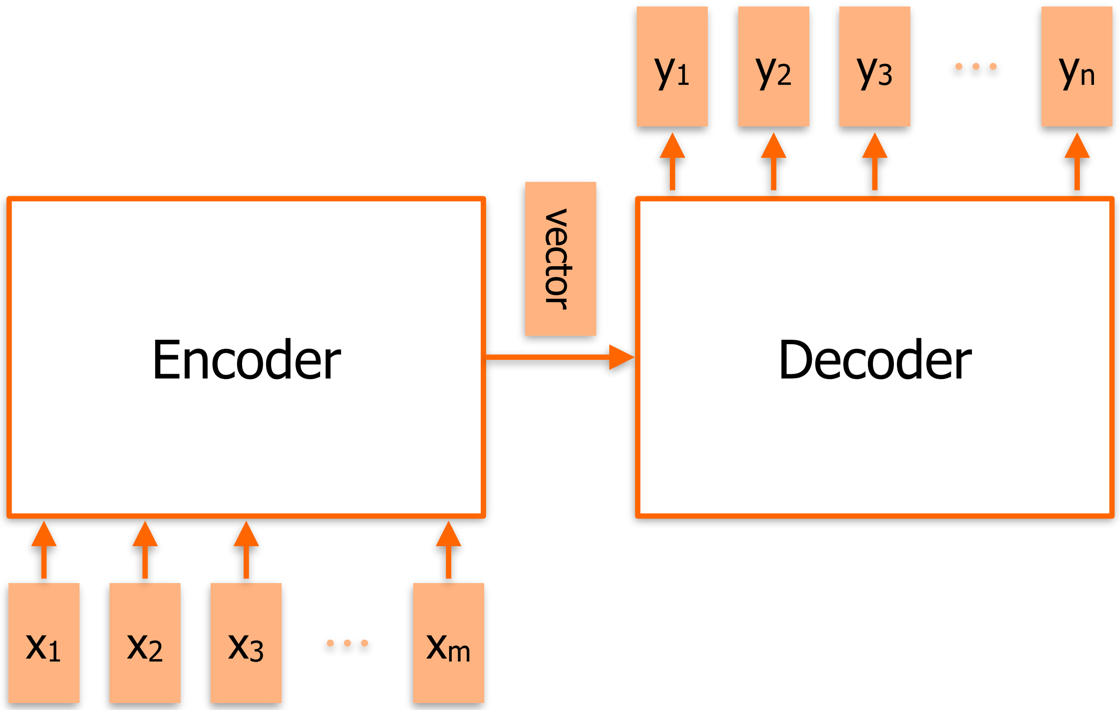
Sequence-to-Sequence Models

- Input: a sequence of vectors.
- Output: a sequence of vectors.
- Example: Machine translation
  - Input: `How are you?`
  - Output: `Wie geht es dir?`
- How to turn words in vocabulary V into a sequence of d-dimensional vectors?



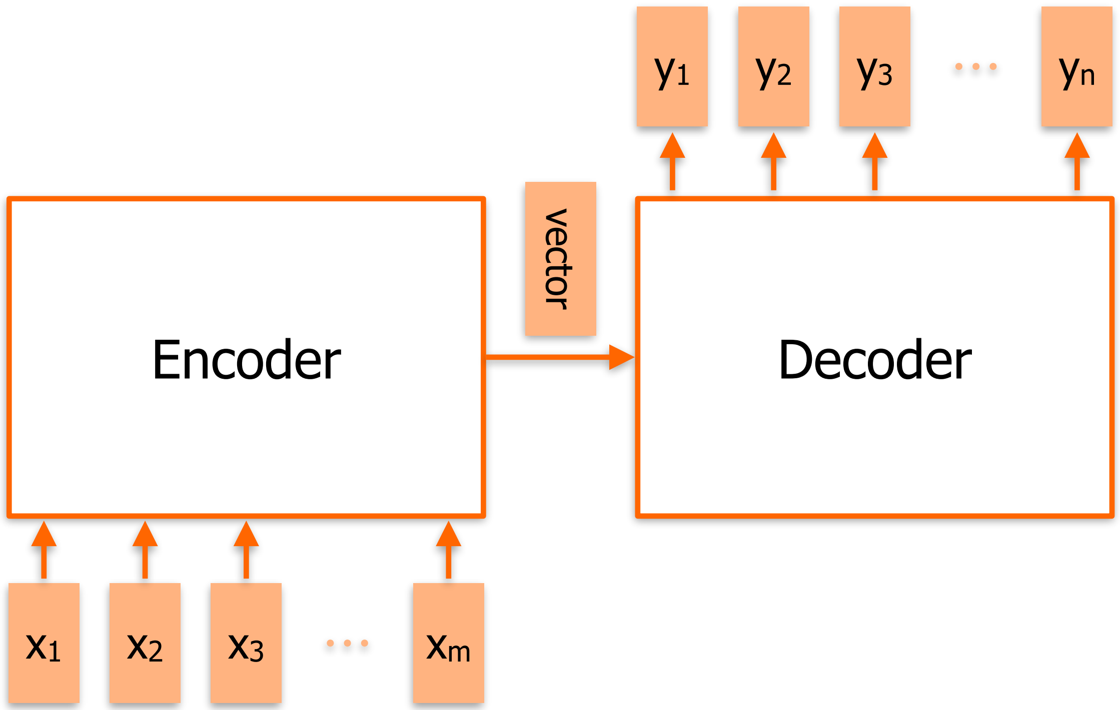
Sequence-to-Sequence Models

- Input: a sequence of vectors.
- Output: a sequence of vectors.
- Example: Machine translation
  - Input: `How are you?`
  - Output: `Wie geht es dir?`
- How to turn words in vocabulary V into a sequence of d-dimensional vectors?
  - $x = W \cdot v$, where $W \in \mathbb{R}^{|V| \times d}$ and v is a **one-hot vector**.



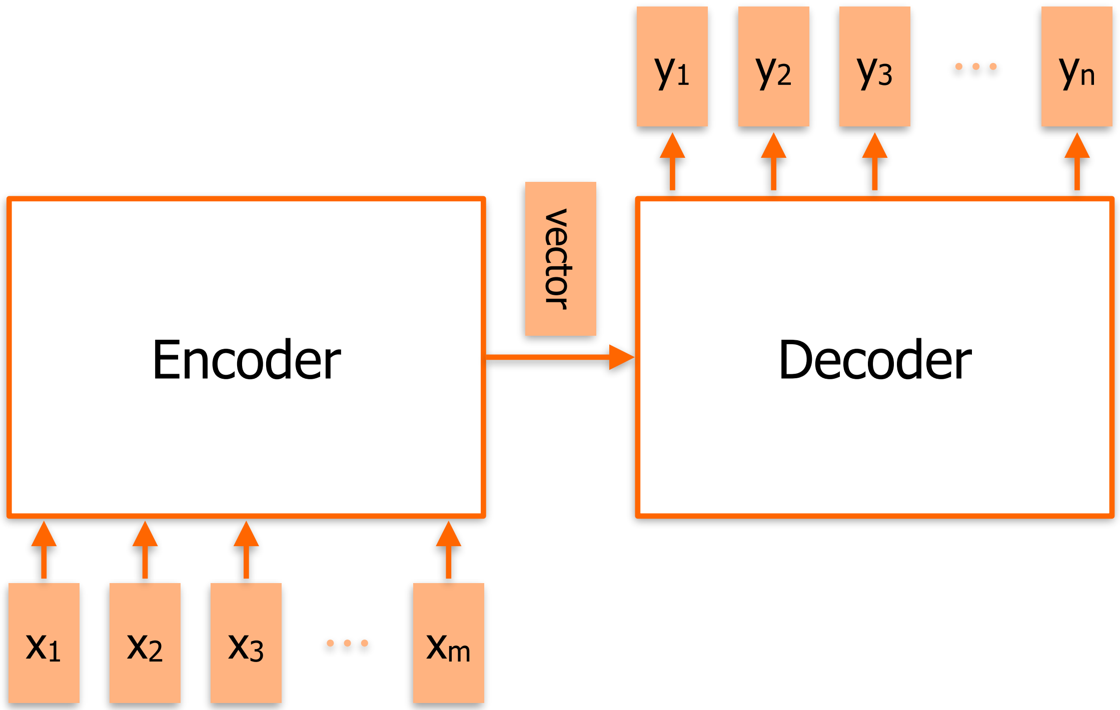
Sequence-to-Sequence Models

- Input: a sequence of vectors.
- Output: a sequence of vectors.
- Example: Machine translation
  - Input: `How are you?`
  - Output: `Wie geht es dir?`
- How to turn words in vocabulary V into a sequence of d-dimensional vectors?
  - $x = W \cdot v$, where $W \in \mathbb{R}^{|V| \times d}$ and v is a **one-hot vector**.
- How to handle different lengths of sequences?



Sequence-to-Sequence Models

- Input: a sequence of vectors.
- Output: a sequence of vectors.
- Example: Machine translation
  - Input: `How are you?`
  - Output: `Wie geht es dir?`
- How to turn words in vocabulary V into a sequence of d-dimensional vectors?
  - $x = W \cdot v$, where $W \in \mathbb{R}^{|V| \times d}$ and v is a **one-hot vector**.
- How to handle different lengths of sequences?
  - One could use an **RNN**.



Limitation of RNNs

- Main Problem: In an RNN (or LSTM) far away words are not much related.
  - `Jane went to the cafeteria to buy a cup of coffee, but she couldn't buy anything because it was closed.`

Limitation of RNNs

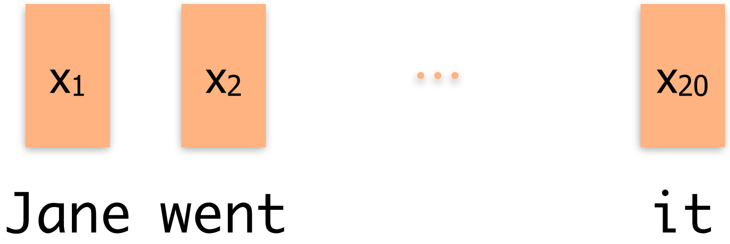
- Main Problem: In an RNN (or LSTM) far away words are not much related.
  - `Jane went to the cafeteria to buy a cup of coffee, but she couldn't buy anything because it was closed.`
- All the information is encoded in one vector: $y_t = f(x_t, y_{t-1})$

Limitation of RNNs

- Main Problem: In an RNN (or LSTM) far away words are not much related.
  - `Jane went to the` <u>`cafeteria`</u> `to buy a cup of coffee, but she couldn't buy anything because` <u>`it`</u> `was closed.`
- All the information is encoded in one vector: $y_t = f(x_t, y_{t-1})$
- Solution: **Self-Attention**

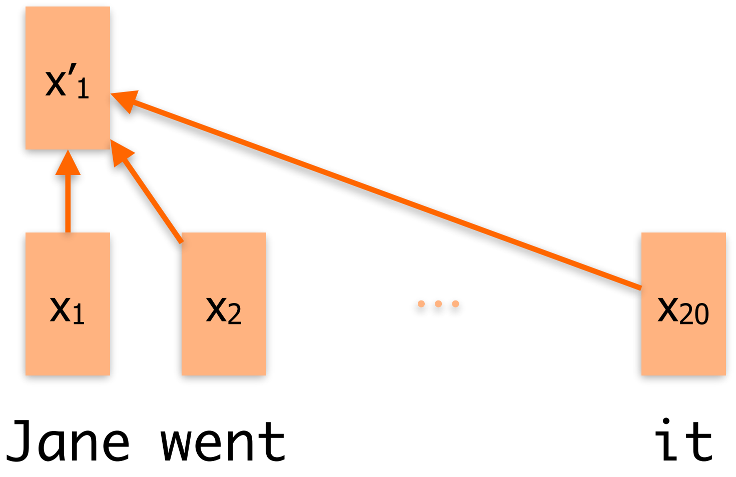
Limitation of RNNs

- Main Problem: In an RNN (or LSTM) far away words are not much related.
  - `Jane went to the cafeteria to buy a cup of coffee, but she couldn't buy anything because it was closed.`
- All the information is encoded in one vector: $y_t = f(x_t, y_{t-1})$
- Solution: **Self-Attention**



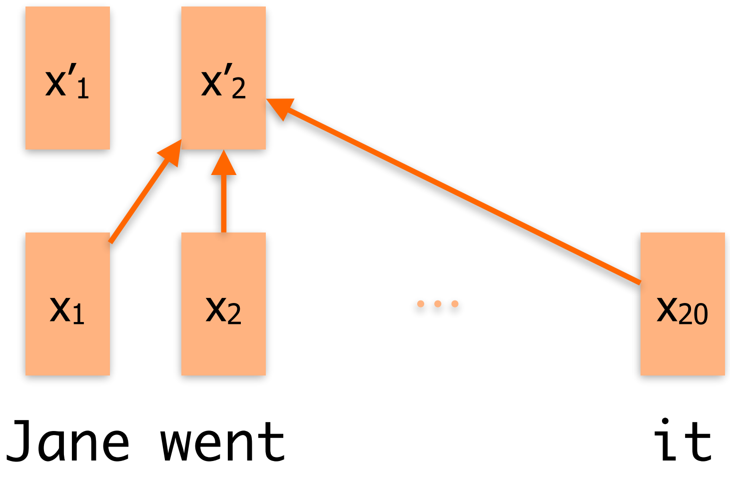
Limitation of RNNs

- Main Problem: In an RNN (or LSTM) far away words are not much related.
  - `Jane went to the cafeteria to buy a cup of coffee, but she couldn't buy anything because it was closed.`
- All the information is encoded in one vector: $y_t = f(x_t, y_{t-1})$
- Solution: **Self-Attention**



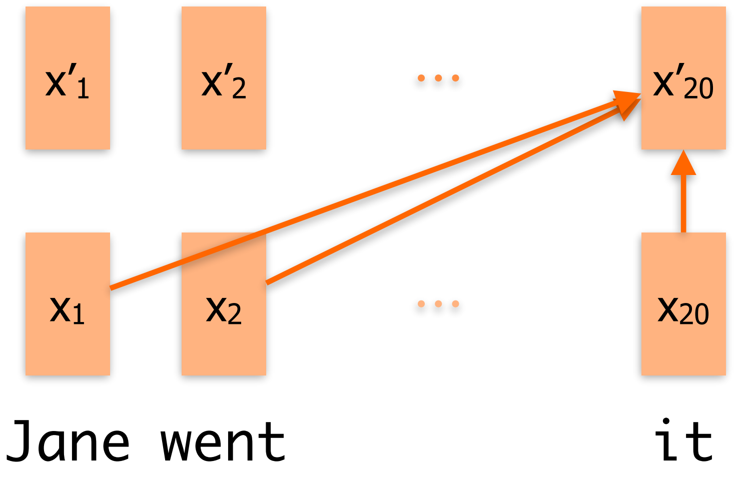
Limitation of RNNs

- Main Problem: In an RNN (or LSTM) far away words are not much related.
  - `Jane went to the cafeteria to buy a cup of coffee, but she couldn't buy anything because it was closed.`
- All the information is encoded in one vector: $y_t = f(x_t, y_{t-1})$
- Solution: **Self-Attention**



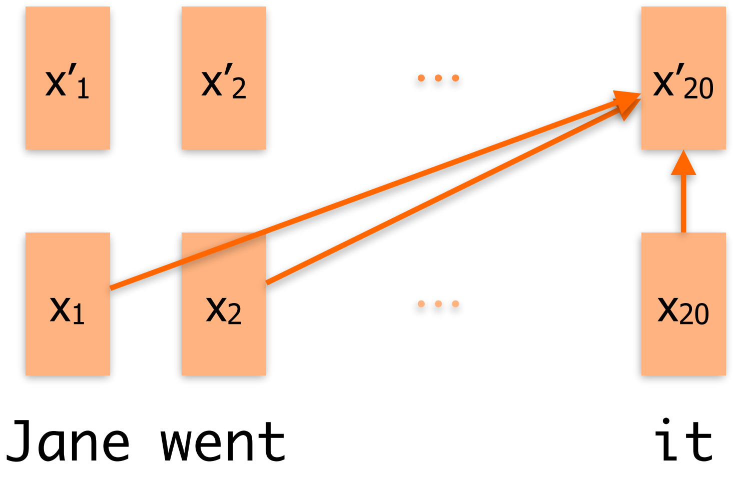
Limitation of RNNs

- Main Problem: In an RNN (or LSTM) far away words are not much related.
  - `Jane went to the` <u>`cafeteria`</u> `to buy a cup of coffee, but she couldn't buy anything because` <u>`it`</u> `was closed.`
- All the information is encoded in one vector: $y_t = f(x_t, y_{t-1})$
- Solution: **Self-Attention**



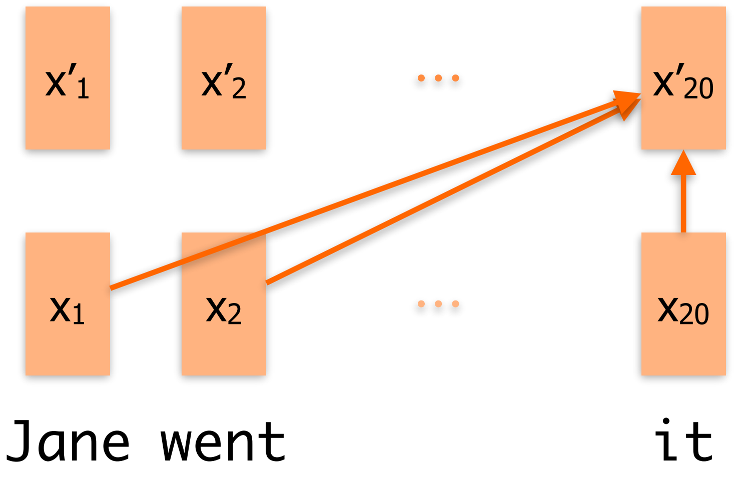
Limitation of RNNs

- Main Problem: In an RNN (or LSTM) far away words are not much related.
  - `Jane went to the cafeteria to buy a cup of coffee, but she couldn't buy anything because it was closed.`
- All the information is encoded in one vector: $y_t = f(x_t, y_{t-1})$
- Solution: **Self-Attention**
  - Represent the **current word** using the **representations of all other words**.



Limitation of RNNs

- Main Problem: In an RNN (or LSTM) far away words are not much related.
  - `Jane went to the cafeteria to buy a cup of coffee, but she couldn't buy anything because it was closed.`
- All the information is encoded in one vector: $y_t = f(x_t, y_{t-1})$
- Solution: **Self-Attention**
  - Represent the **current word** using the **representations of all other words**.
  - But how does self-attention work in detail?



Self-Attention: The Idea

- V is a matrix (trainable).

A. Vaswani et al., "Attention is All you Need," in Advances in Neural Information Processing Systems 30, 2017

Self-Attention: The Idea

- V is a matrix (trainable).
- Compute the **value** vectors
  - $v_i = V \cdot x_i$

A. Vaswani et al., "Attention is All you Need," in Advances in Neural Information Processing Systems 30, 2017

Self-Attention: The Idea

- V is a matrix (trainable).
- Compute the **value** vectors
  - $v_i = V \cdot x_i$
- Prepare weights $\alpha_1^i, \ldots, \alpha_m^i \geq 0$

A. Vaswani et al., “Attention is All you Need,” in Advances in Neural Information Processing Systems 30, 2017

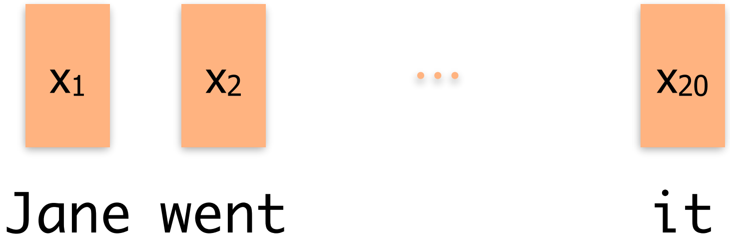
Self-Attention: The Idea

- V is a matrix (trainable).
- Compute the **value** vectors
  - $v_i = V \cdot x_i$
- Prepare weights $\alpha_1^i, \ldots, \alpha_m^i \geq 0$
- New vector $x_i' = \alpha_1^i v_1 + \ldots + \alpha_m^i v_m$.

A. Vaswani et al., “Attention is All you Need,” in Advances in Neural Information Processing Systems 30, 2017

Self-Attention: The Idea

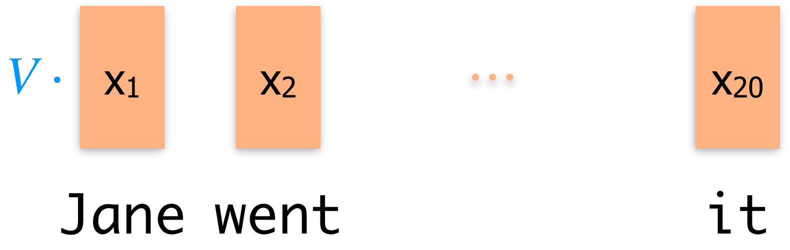
- V is a matrix (trainable).
- Compute the **value** vectors
  - $v_i = V \cdot x_i$
- Prepare weights $\alpha_1^i, \ldots, \alpha_m^i \geq 0$
- New vector $x_i' = \alpha_1^i v_1 + \ldots + \alpha_m^i v_m$.



A. Vaswani et al., "Attention is All you Need," in Advances in Neural Information Processing Systems 30, 2017

Self-Attention: The Idea

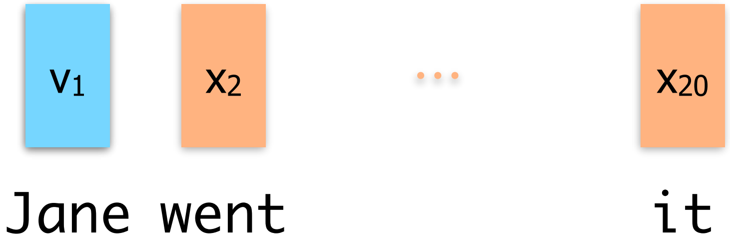
- V is a matrix (trainable).
- Compute the **value** vectors
  - $v_i = V \cdot x_i$
- Prepare weights $\alpha_1^i, \ldots, \alpha_m^i \geq 0$
- New vector $x_i' = \alpha_1^i v_1 + \ldots + \alpha_m^i v_m$.



A. Vaswani et al., "Attention is All you Need," in Advances in Neural Information Processing Systems 30, 2017

Self-Attention: The Idea

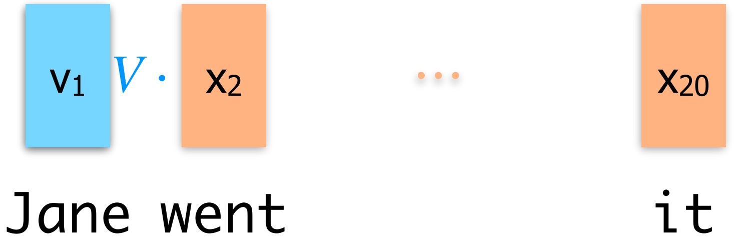
- V is a matrix (trainable).
- Compute the **value** vectors
  - $v_i = V \cdot x_i$
- Prepare weights $\alpha_1^i, \ldots, \alpha_m^i \geq 0$
- New vector $x_i' = \alpha_1^i v_1 + \ldots + \alpha_m^i v_m$.



A. Vaswani et al., "Attention is All you Need," in Advances in Neural Information Processing Systems 30, 2017

Self-Attention: The Idea

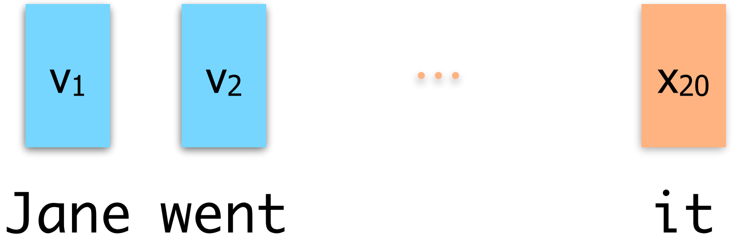
- V is a matrix (trainable).
- Compute the **value** vectors
  - $v_i = V \cdot x_i$
- Prepare weights $\alpha_1^i, \ldots, \alpha_m^i \geq 0$
- New vector $x_i' = \alpha_1^i v_1 + \ldots + \alpha_m^i v_m$.



A. Vaswani et al., “Attention is All you Need,” in Advances in Neural Information Processing Systems 30, 2017

Self-Attention: The Idea

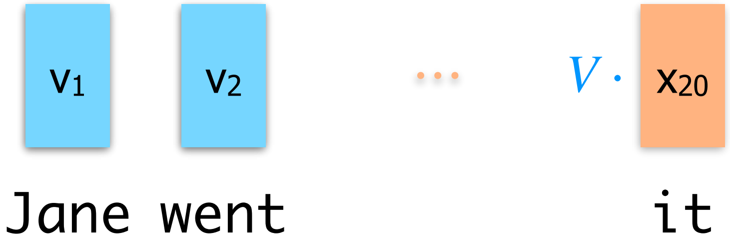
- V is a matrix (trainable).
- Compute the **value** vectors
  - $v_i = V \cdot x_i$
- Prepare weights $\alpha_1^i, \ldots, \alpha_m^i \geq 0$
- New vector $x_i' = \alpha_1^i v_1 + \ldots + \alpha_m^i v_m$.



A. Vaswani et al., “Attention is All you Need,” in Advances in Neural Information Processing Systems 30, 2017

Self-Attention: The Idea

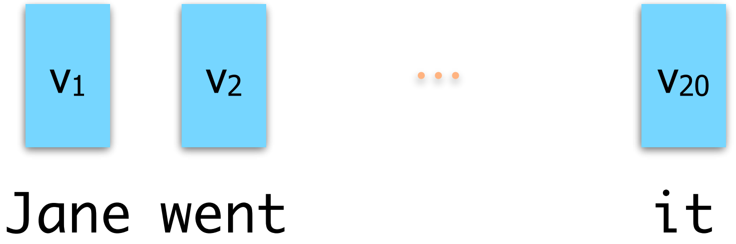
- V is a matrix (trainable).
- Compute the **value** vectors
  - $v_i = V \cdot x_i$
- Prepare weights $\alpha_1^i, \ldots, \alpha_m^i \geq 0$
- New vector $x_i' = \alpha_1^i v_1 + \ldots + \alpha_m^i v_m$.



A. Vaswani et al., “Attention is All you Need,” in Advances in Neural Information Processing Systems 30, 2017

Self-Attention: The Idea

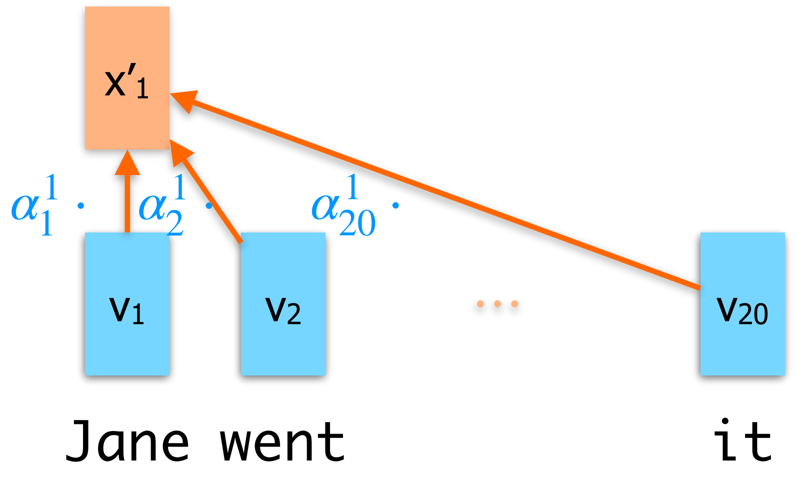
- V is a matrix (trainable).
- Compute the **value** vectors
  - $v_i = V \cdot x_i$
- Prepare weights $\alpha_1^i, \ldots, \alpha_m^i \geq 0$
- New vector $x_i' = \alpha_1^i v_1 + \ldots + \alpha_m^i v_m$.



A. Vaswani et al., "Attention is All you Need," in Advances in Neural Information Processing Systems 30, 2017

Self-Attention: The Idea

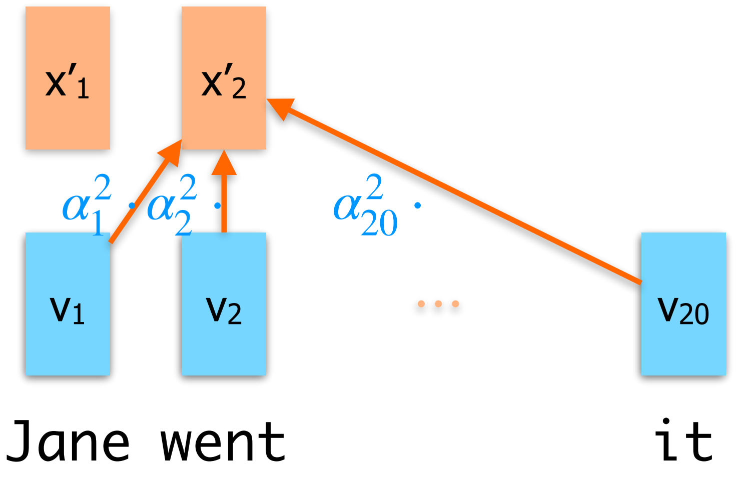
- V is a matrix (trainable).
- Compute the **value** vectors
  - $v_i = V \cdot x_i$
- Prepare weights $\alpha_1^i, \ldots, \alpha_m^i \geq 0$
- New vector $x_i' = \alpha_1^i v_1 + \ldots + \alpha_m^i v_m$.



A. Vaswani et al., “Attention is All you Need,” in Advances in Neural Information Processing Systems 30, 2017

Self-Attention: The Idea

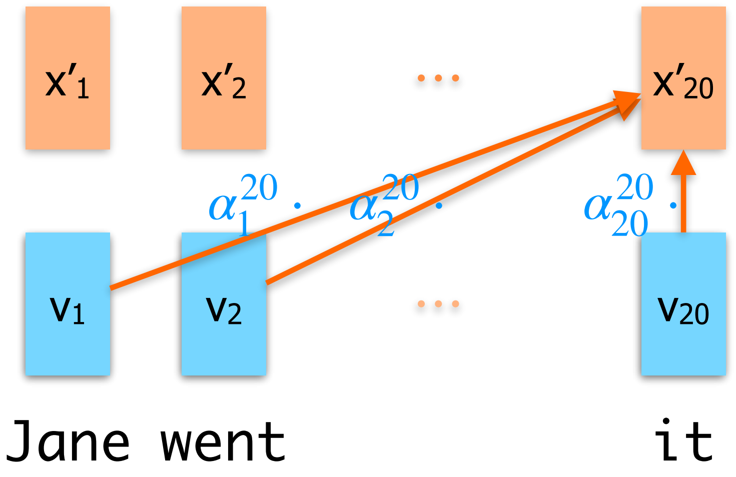
- V is a matrix (trainable).
- Compute the **value** vectors
  - $v_i = V \cdot x_i$
- Prepare weights $\alpha_1^i, \ldots, \alpha_m^i \geq 0$
- New vector $x_i' = \alpha_1^i v_1 + \ldots + \alpha_m^i v_m$.



A. Vaswani et al., "Attention is All you Need," in Advances in Neural Information Processing Systems 30, 2017

Self-Attention: The Idea

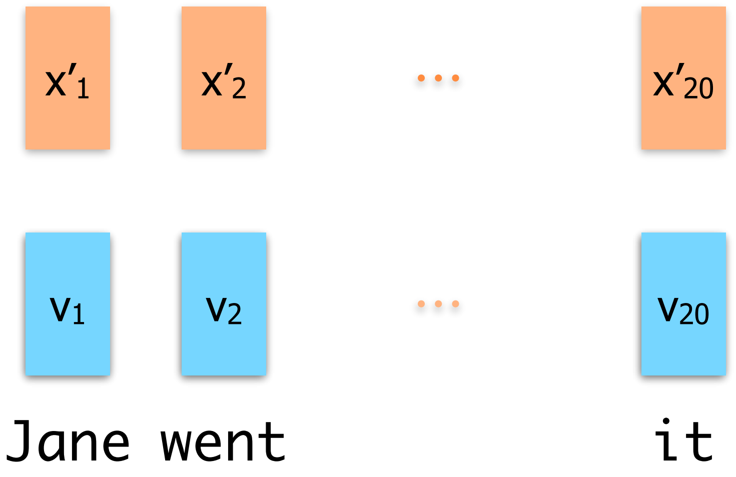
- V is a matrix (trainable).
- Compute the **value** vectors
  - $v_i = V \cdot x_i$
- Prepare weights $\alpha_1^i, \ldots, \alpha_m^i \geq 0$
- New vector $x_i' = \alpha_1^i v_1 + \ldots + \alpha_m^i v_m$.



A. Vaswani et al., "Attention is All you Need," in Advances in Neural Information Processing Systems 30, 2017

Self-Attention: The Idea

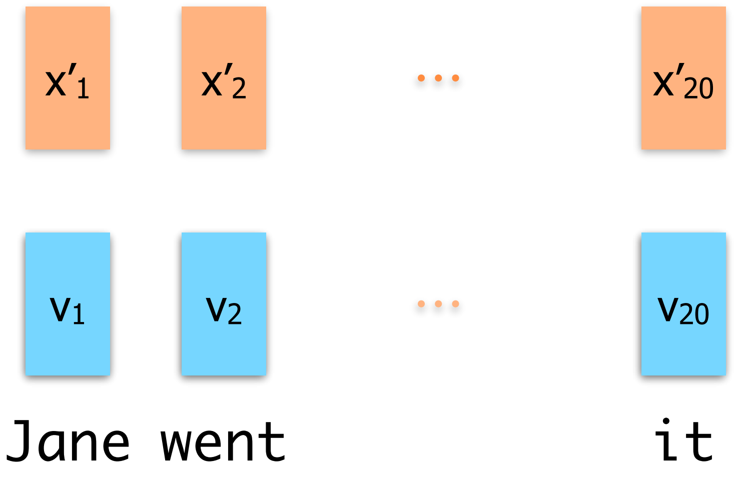
- V is a matrix (trainable).
- Compute the **value** vectors
  - $v_i = V \cdot x_i$
- Prepare weights $\alpha_1^i, \ldots, \alpha_m^i \geq 0$
- New vector $x_i' = \alpha_1^i v_1 + \ldots + \alpha_m^i v_m$.



A. Vaswani et al., "Attention is All you Need," in Advances in Neural Information Processing Systems 30, 2017

Self-Attention: The Idea

- V is a matrix (trainable).
- Compute the **value** vectors
  - $v_i = V \cdot x_i$
- Prepare weights $\alpha_1^i, \ldots, \alpha_m^i \geq 0$
- New vector $x_i' = \alpha_1^i v_1 + \ldots + \alpha_m^i v_m$.
- How to obtain the weights $\alpha_1^i, \ldots, \alpha_m^i$?



A. Vaswani et al., "Attention is All you Need," in Advances in Neural Information Processing Systems 30, 2017

Self-Attention: Computing the Weights α_j^i

- Idea: α_j^i is **high**, if two vectors x_i and x_j are **similar** for a given context.

Self-Attention: Computing the Weights α_j^i

- Idea: α_j^i is **high**, if two vectors x_i and x_j are **similar** for a given context.
- Let Q, K be matrices (trainable):

Self-Attention: Computing the Weights α_j^i

- Idea: α_j^i is **high**, if two vectors x_i and x_j are **similar** for a given context.
- Let Q, K be matrices (trainable):

  (1) Compute the **query** vectors:

  $$q_i = Q \cdot x_i$$

Self-Attention: Computing the Weights α_j^i

- Idea: α_j^i is **high**, if two vectors x_i and x_j are **similar** for a given context.
- Let Q, K be matrices (trainable):

  (1) Compute the **query** vectors:

  $$q_i = Q \cdot x_i$$

  (2) Compute the **key** vectors:

  $$k_j = K \cdot x_j$$

Self-Attention: Computing the Weights α_j^i

- Idea: α_j^i is **high**, if two vectors x_i and x_j are **similar** for a given context.
- Let Q, K be matrices (trainable):

  (1) Compute the **query** vectors:

  $$q_i = Q \cdot x_i$$

  (2) Compute the **key** vectors:

  $$k_j = K \cdot x_j$$

  (3) α_j^i is the **dot product** of q_i and k_j,

  i.e., $a_j^i = q_i^T \cdot k_j$.

Self-Attention: Computing the Weights α_j^i

- Idea: α_j^i is **high**, if two vectors x_i and x_j are **similar** for a given context.
- Let Q, K be matrices (trainable):

  (1) Compute the **query** vectors:

  $$q_i = Q \cdot x_i$$

  (2) Compute the **key** vectors:

  $$k_j = K \cdot x_j$$

  (3) α_j^i is the **dot product** of q_i and k_j, i.e., $a_j^i = q_i^T \cdot k_j$.

  (4) Apply **softmax** to $\alpha_1^i, \ldots, \alpha_m^i$.

Self-Attention: Computing the Weights α_j^i

- Idea: α_j^i is **high**, if two vectors x_i and x_j are **similar** for a given context.
- Let Q, K be matrices (trainable):

  (1) Compute the **query** vectors:
  $q_i = Q \cdot x_i$

  (2) Compute the **key** vectors:
  $k_j = K \cdot x_j$

  (3) α_j^i is the **dot product** of q_i and k_j, i.e., $a_j^i = q_i^T \cdot k_j$.

  (4) Apply **softmax** to $\alpha_1^i, \ldots, \alpha_m^i$.

Example

- $x_1 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, x_2 = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}, x_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 2 \end{bmatrix}$

Self-Attention: Computing the Weights α_j^i

- Idea: α_j^i is **high**, if two vectors x_i and x_j are **similar** for a given context.
- Let Q, K be matrices (trainable):

  (1) Compute the **query** vectors:

  $$q_i = Q \cdot x_i$$

  (2) Compute the **key** vectors:

  $$k_j = K \cdot x_j$$

  (3) α_j^i is the **dot product** of q_i and k_j, i.e., $a_j^i = q_i^T \cdot k_j$.

  (4) Apply **softmax** to $\alpha_1^i, \ldots, \alpha_m^i$.

Example

- $x_1 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, x_2 = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}, x_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 2 \end{bmatrix}$
- $Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 2 & 3 & 4 \end{bmatrix}, K = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 1 & 1 & 1 \end{bmatrix}$

Self-Attention: Computing the Weights α_j^i

- Idea: α_j^i is **high**, if two vectors x_i and x_j are **similar** for a given context.
- Let Q, K be matrices (trainable):

  (1) Compute the **query** vectors:
  $q_i = Q \cdot x_i$

  (2) Compute the **key** vectors:
  $k_j = K \cdot x_j$

  (3) α_j^i is the **dot product** of q_i and k_j, i.e., $a_j^i = q_i^T \cdot k_j$.

  (4) Apply **softmax** to $\alpha_1^i, \ldots, \alpha_m^i$.

Example

- $x_1 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, x_2 = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}, x_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 2 \end{bmatrix}$
- $Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 2 & 3 & 4 \end{bmatrix}, K = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 1 & 1 & 1 \end{bmatrix}$

  (1) $q_1 = \begin{bmatrix} 1 \\ 3 \end{bmatrix}, q_2 = \begin{bmatrix} 1 \\ 30 \end{bmatrix}, q_3 = \begin{bmatrix} 0 \\ 11 \end{bmatrix}$

Self-Attention: Computing the Weights α_j^i

- Idea: α_j^i is **high**, if two vectors x_i and x_j are **similar** for a given context.
- Let Q, K be matrices (trainable):
  (1) Compute the **query** vectors:
  $q_i = Q \cdot x_i$
  (2) Compute the **key** vectors:
  $k_j = K \cdot x_j$
  (3) α_j^i is the **dot product** of q_i and k_j, i.e., $a_j^i = q_i^T \cdot k_j$.
  (4) Apply **softmax** to $\alpha_1^i, \ldots, \alpha_m^i$.

Example

- $x_1 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, x_2 = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}, x_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 2 \end{bmatrix}$
- $Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 2 & 3 & 4 \end{bmatrix}, K = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 1 & 1 & 1 \end{bmatrix}$

  (1) $q_1 = \begin{bmatrix} 1 \\ 3 \end{bmatrix}, q_2 = \begin{bmatrix} 1 \\ 30 \end{bmatrix}, q_3 = \begin{bmatrix} 0 \\ 11 \end{bmatrix}$

  (2) $k_1 = \begin{bmatrix} 3 \\ 2 \end{bmatrix}, k_2 = \begin{bmatrix} 30 \\ 10 \end{bmatrix}, k_3 = \begin{bmatrix} 11 \\ 3 \end{bmatrix}$

Self-Attention: Computing the Weights α_j^i

- Idea: α_j^i is **high**, if two vectors x_i and x_j are **similar** for a given context.
- Let Q, K be matrices (trainable):
  (1) Compute the **query** vectors:
  $q_i = Q \cdot x_i$
  (2) Compute the **key** vectors:
  $k_j = K \cdot x_j$
  (3) α_j^i is the **dot product** of q_i and k_j, i.e., $a_j^i = q_i^T \cdot k_j$.
  (4) Apply **softmax** to $\alpha_1^i, \ldots, \alpha_m^i$.

Example

- $x_1 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, x_2 = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}, x_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 2 \end{bmatrix}$
- $Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 2 & 3 & 4 \end{bmatrix}, K = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 1 & 1 & 1 \end{bmatrix}$

  (1) $q_1 = \begin{bmatrix} 1 \\ 3 \end{bmatrix}, q_2 = \begin{bmatrix} 1 \\ 30 \end{bmatrix}, q_3 = \begin{bmatrix} 0 \\ 11 \end{bmatrix}$

  (2) $k_1 = \begin{bmatrix} 3 \\ 2 \end{bmatrix}, k_2 = \begin{bmatrix} 30 \\ 10 \end{bmatrix}, k_3 = \begin{bmatrix} 11 \\ 3 \end{bmatrix}$

  (3) $\alpha_1^1 = 9, \alpha_2^1 = 60, \alpha_3^1 = 20, \ldots$

Self-Attention: Computing the Weights α_j^i

- Idea: α_j^i is **high**, if two vectors x_i and x_j are **similar** for a given context.
- Let Q, K be matrices (trainable):
  (1) Compute the **query** vectors:
  $q_i = Q \cdot x_i$
  (2) Compute the **key** vectors:
  $k_j = K \cdot x_j$
  (3) α_j^i is the **dot product** of q_i and k_j,
  i.e., $a_j^i = q_i^T \cdot k_j$.
  (4) Apply **softmax** to $\alpha_1^i, \ldots, \alpha_m^i$.

Example

- $x_1 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, x_2 = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}, x_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 2 \end{bmatrix}$
- $Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 2 & 3 & 4 \end{bmatrix}, K = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 1 & 1 & 1 \end{bmatrix}$

  (1) $q_1 = \begin{bmatrix} 1 \\ 3 \end{bmatrix}, q_2 = \begin{bmatrix} 1 \\ 30 \end{bmatrix}, q_3 = \begin{bmatrix} 0 \\ 11 \end{bmatrix}$

  (2) $k_1 = \begin{bmatrix} 3 \\ 2 \end{bmatrix}, k_2 = \begin{bmatrix} 30 \\ 10 \end{bmatrix}, k_3 = \begin{bmatrix} 11 \\ 3 \end{bmatrix}$

  (3) $\alpha_1^1 = 9, \alpha_2^1 = 60, \alpha_3^1 = 20, \ldots$

  (4) $\alpha_1^1 = 0.0, \alpha_2^1 = 1.0, \alpha_3^1 = 0.0, \ldots$

Summary

- Deep Learning allows for learning good representations.
- There are different DL building blocks for learning good representations.
- Self-attention is a DL building block that overcomes limitations of an RNN.

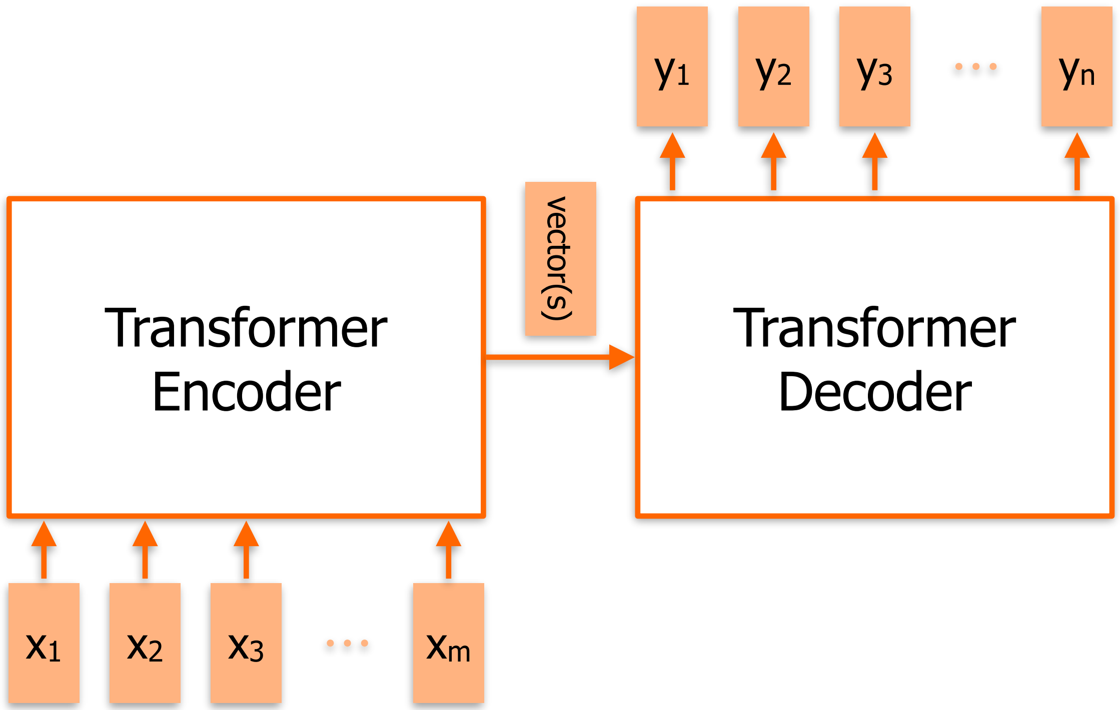
Questions?

Transformers

- Self-Attention
- Transformers
  - Introduction
  - Architecture
  - Inference
  - Training
- Transformer Applications
- Multimodal Learning

Transformer: Introduction

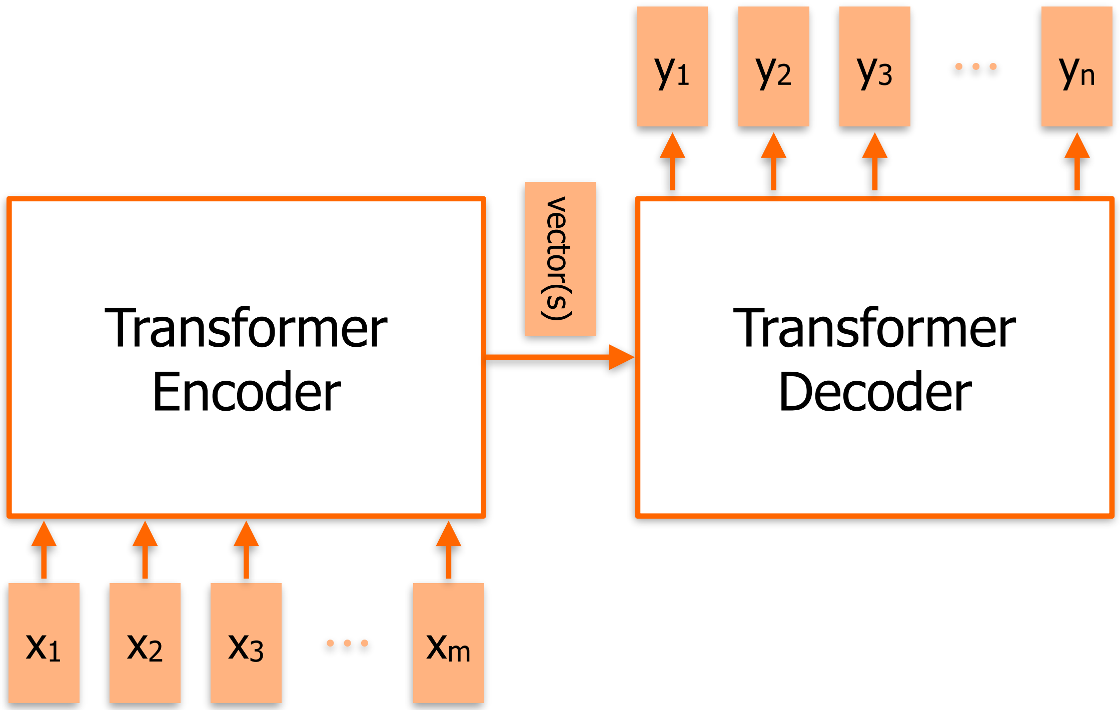
- Sequence-to-Sequence Model.



A. Vaswani et al., "Attention is All you Need," in Advances in Neural Information Processing Systems 30, 2017

Transformer: Introduction

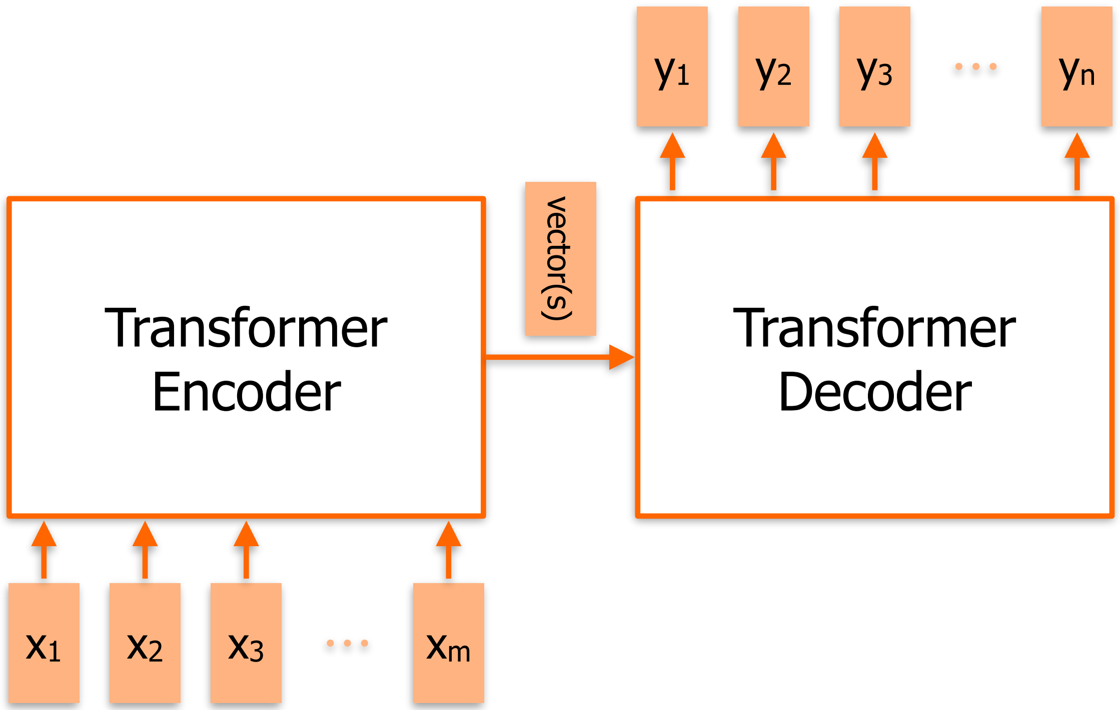
- Sequence-to-Sequence Model.
- Based on self-attention.



 A. Vaswani et al., “Attention is All you Need,” in Advances in Neural Information Processing Systems 30, 2017

Transformer: Introduction

- Sequence-to-Sequence Model.
- Based on self-attention.
- Transformer literally “transformed” current deep learning architectures.
  - Natural Language Processing (ChatGPT!, BERT, GPT, …)
  - Vision (ViT, …)
  - Speech (Conformer, …)
  - Bioinformatics (AlphaFold, …)
  - Multimodal Learning (LXMERT, ViL-BERT, …)
  - …



A. Vaswani et al., “Attention is All you Need,” in Advances in Neural Information Processing Systems 30, 2017

Transformer Variants

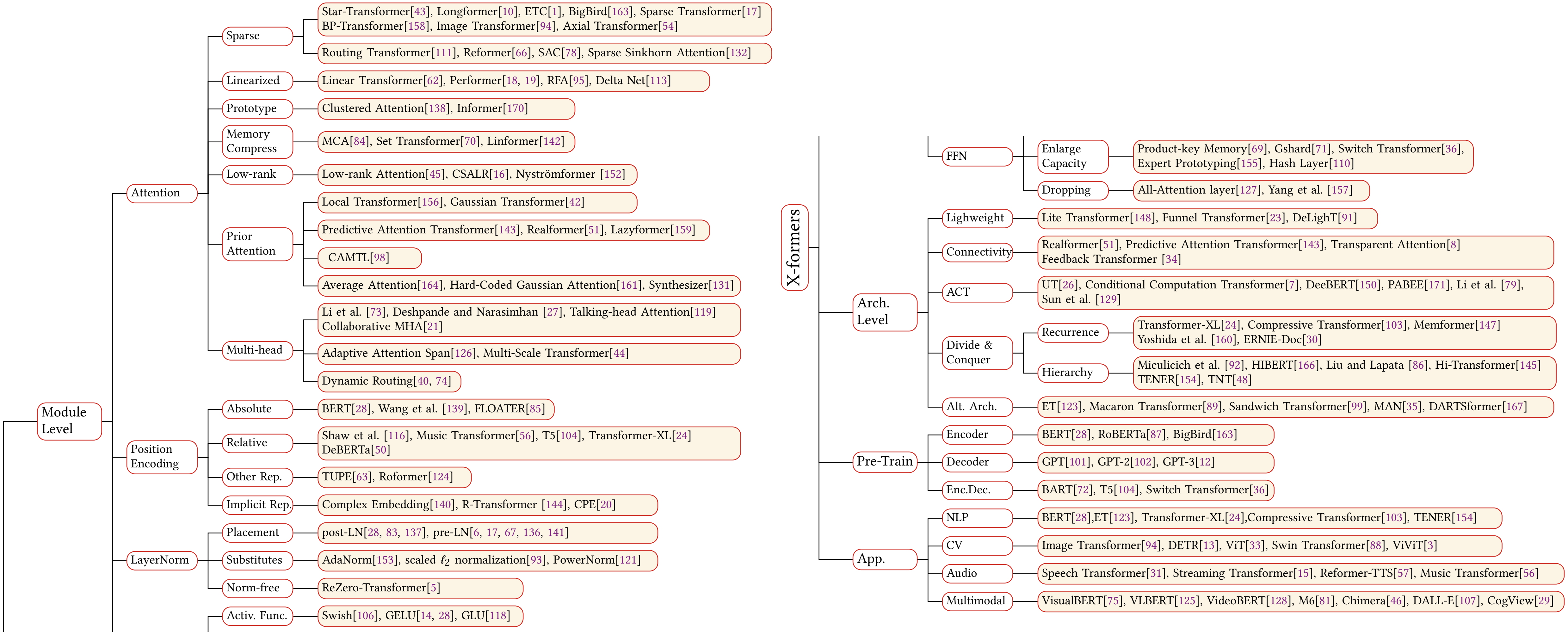
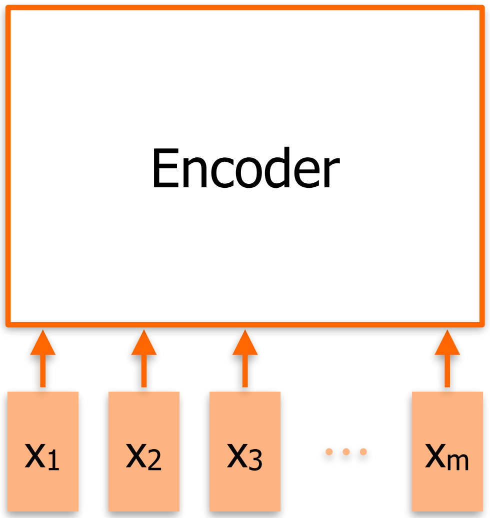


Fig. 3. Taxonomy of Transformers

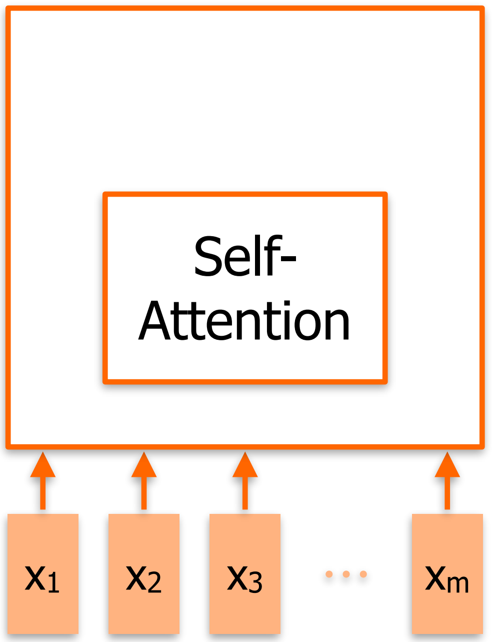
 T. Lin, Y. Wang, X. Liu, and X. Qiu, “A Survey of Transformers.” 2021

Transformer Architecture: Encoder



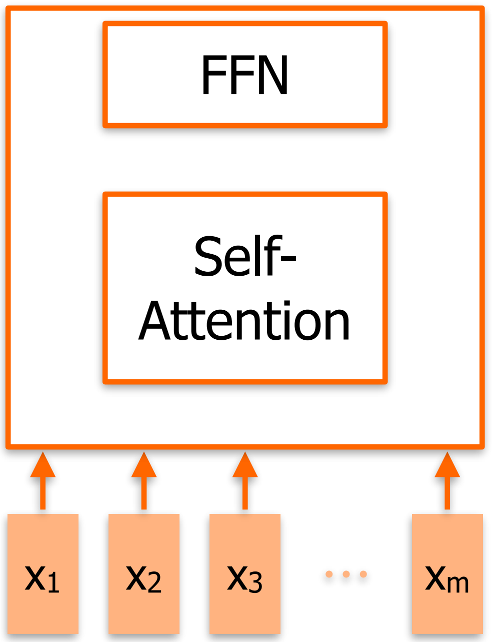
Transformer Architecture: Encoder

- **Self-Attention** is the main component of a Transformer.



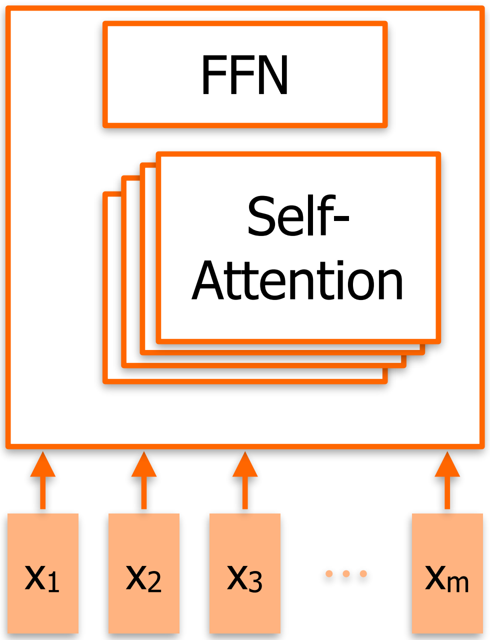
Transformer Architecture: Encoder

- **Self-Attention** is the main component of a Transformer.
- Another component in **FFN** (Feed-forward Network)
  - FFN(x) = Linear(ReLu(Linear(x)))
  - Weights of FFN are **shared**



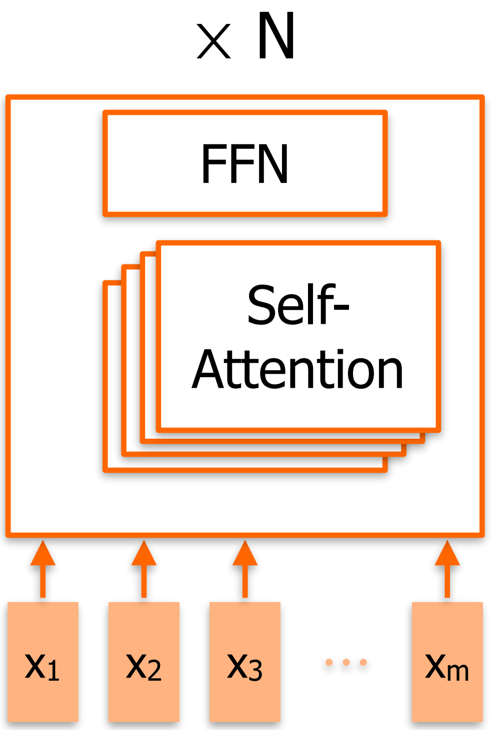
Transformer Architecture: Encoder

- **Self-Attention** is the main component of a Transformer.
- Another component in **FFN** (Feed-forward Network)
  - FFN(x) = Linear(ReLu(Linear(x)))
  - Weights of FFN are **shared**
- **Multi-head attention** + concatenation
  - 8 Self-attention layers.
  - The outputs are concatenated.



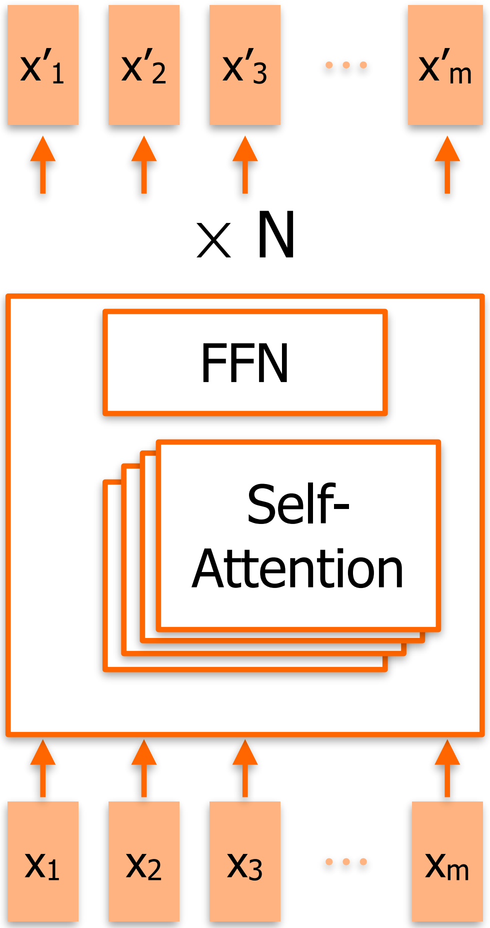
Transformer Architecture: Encoder

- **Self-Attention** is the main component of a Transformer.
- Another component in **FFN** (Feed-forward Network)
  - FFN(x) = Linear(ReLu(Linear(x)))
  - Weights of FFN are **shared**
- **Multi-head attention** + concatenation
  - 8 Self-attention layers.
  - The outputs are concatenated.
- **Stacked** Transformer blocks (make it deep!)



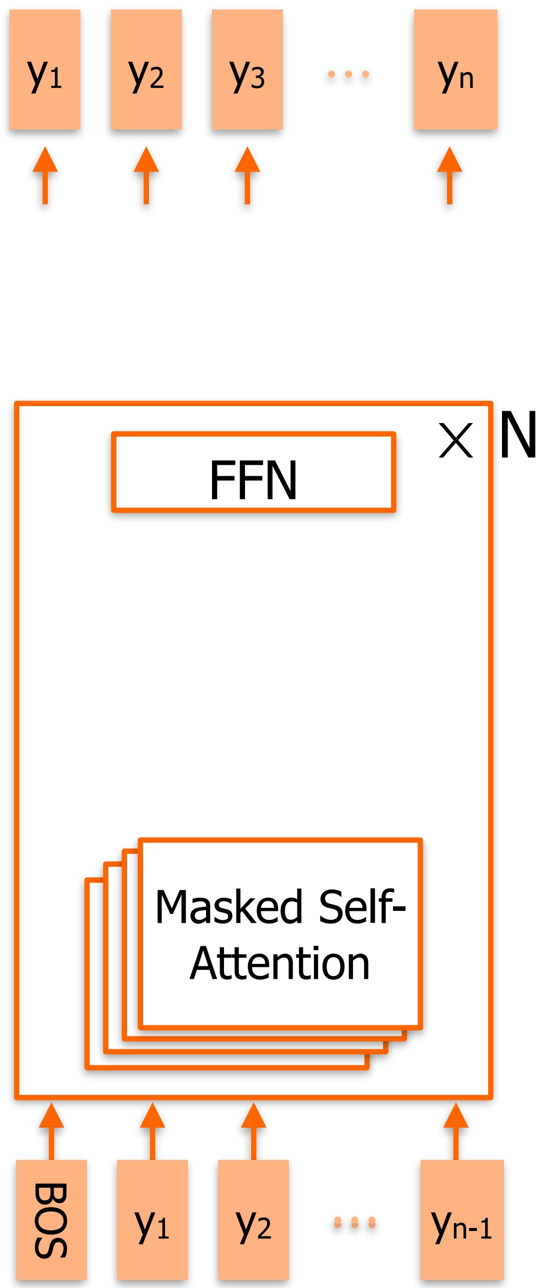
Transformer Architecture: Encoder

- **Self-Attention** is the main component of a Transformer.
- Another component in **FFN** (Feed-forward Network)
  - FFN(x) = Linear(ReLu(Linear(x)))
  - Weights of FFN are **shared**
- **Multi-head attention** + concatenation
  - 8 Self-attention layers.
  - The outputs are concatenated.
- **Stacked** Transformer blocks (make it deep!)

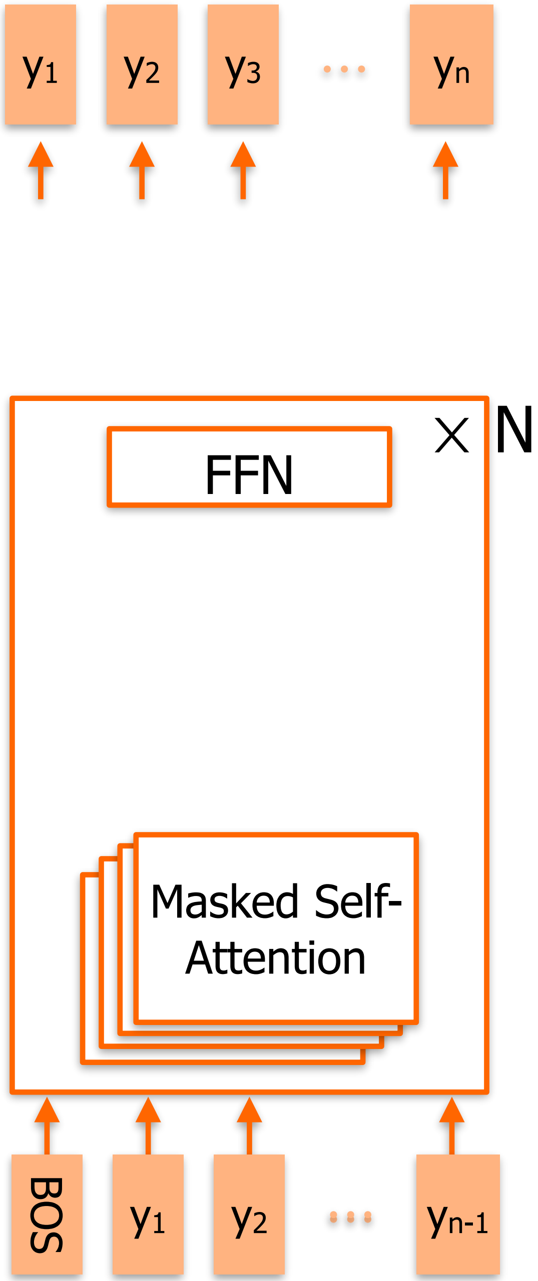


Transformer Architecture: Decoder

- The decoder is similar to the encoder.



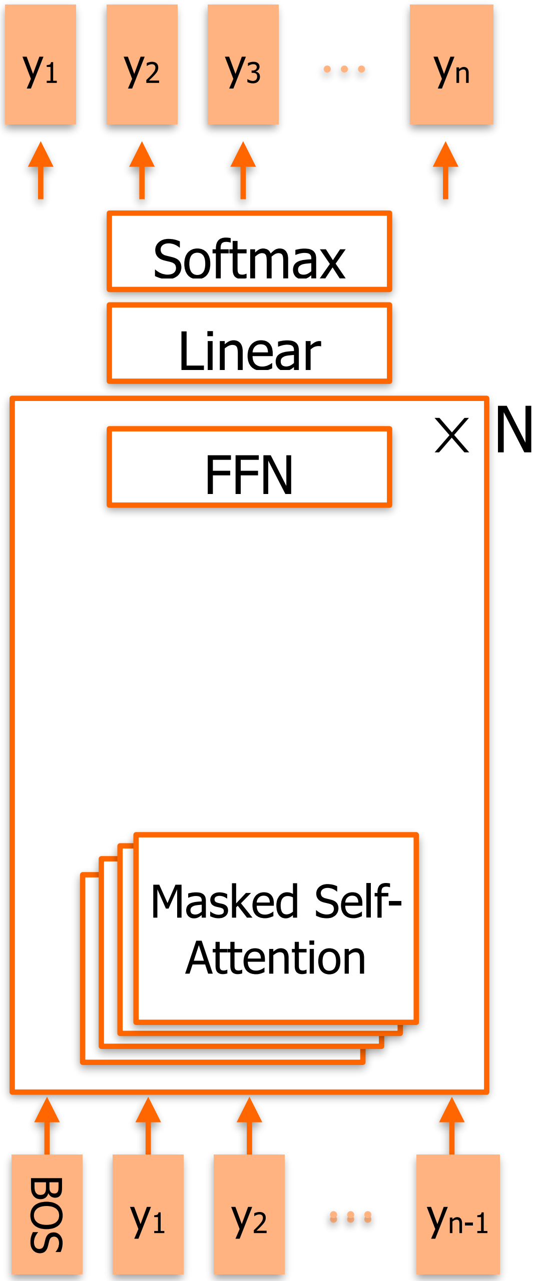
Transformer Architecture: Decoder



- The decoder is similar to the encoder.
- **Masked self-attention**
  - Later inputs are **not attended** to (i.e., attention weights α_j^i for later inputs are zero) → Transformer Training

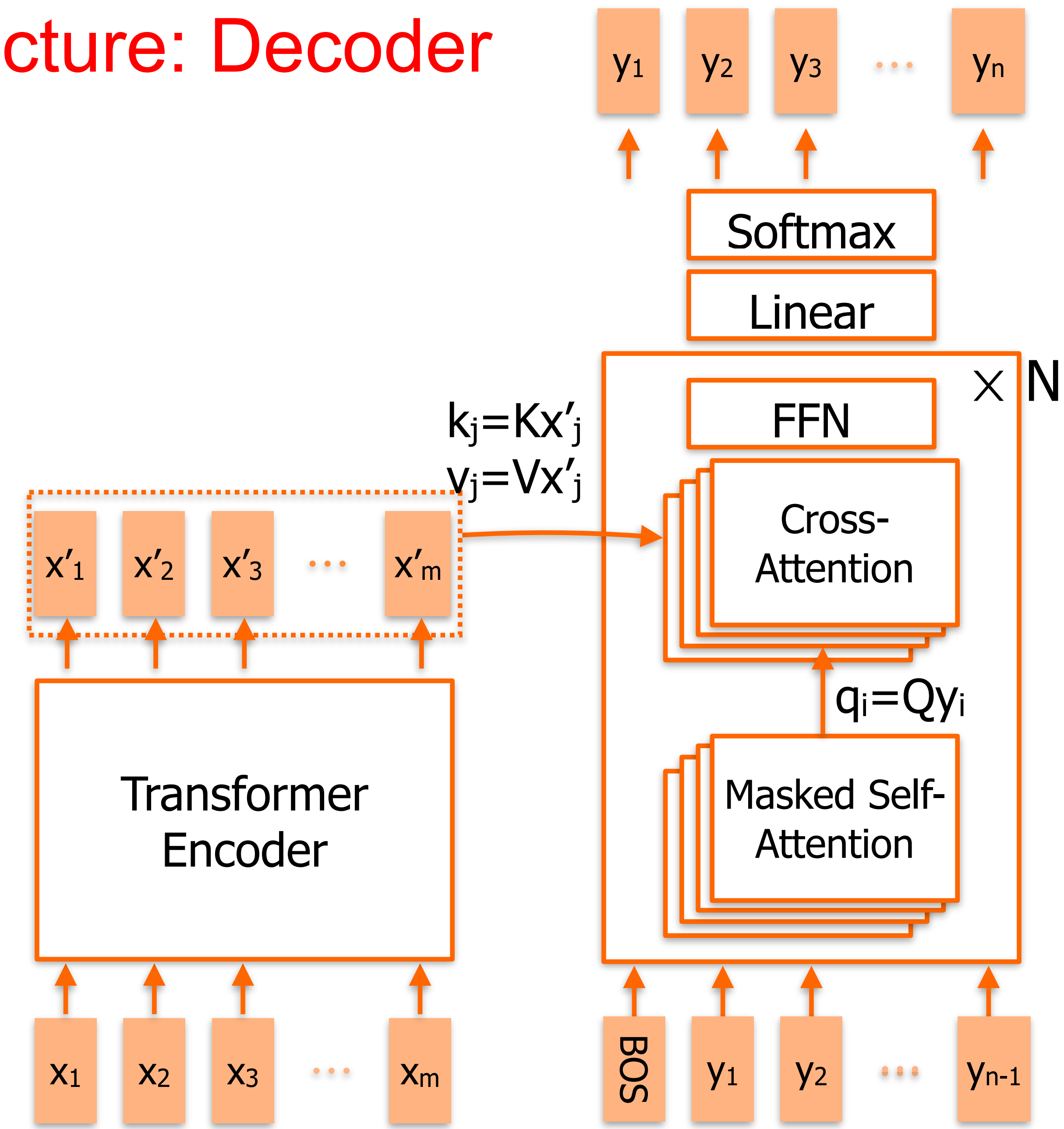
Transformer Architecture: Decoder

- The decoder is similar to the encoder.
- **Masked self-attention**
  - Later inputs are **not attended** to (i.e., attention weights α_j^i for later inputs are zero) → Transformer Training
- **Linear + Softmax**



Transformer Architecture: Decoder

- The decoder is similar to the encoder.
- **Masked self-attention**
  - Later inputs are **not attended** to (i.e., attention weights α_j^i for later inputs are zero) → Transformer Training
- **Linear + Softmax**
- **Cross-attention**
  - The **queries** are obtained from the output vectors of the previous **decoder** layer.
  - The **keys** and **values** are from the output vectors of the **encoder**.

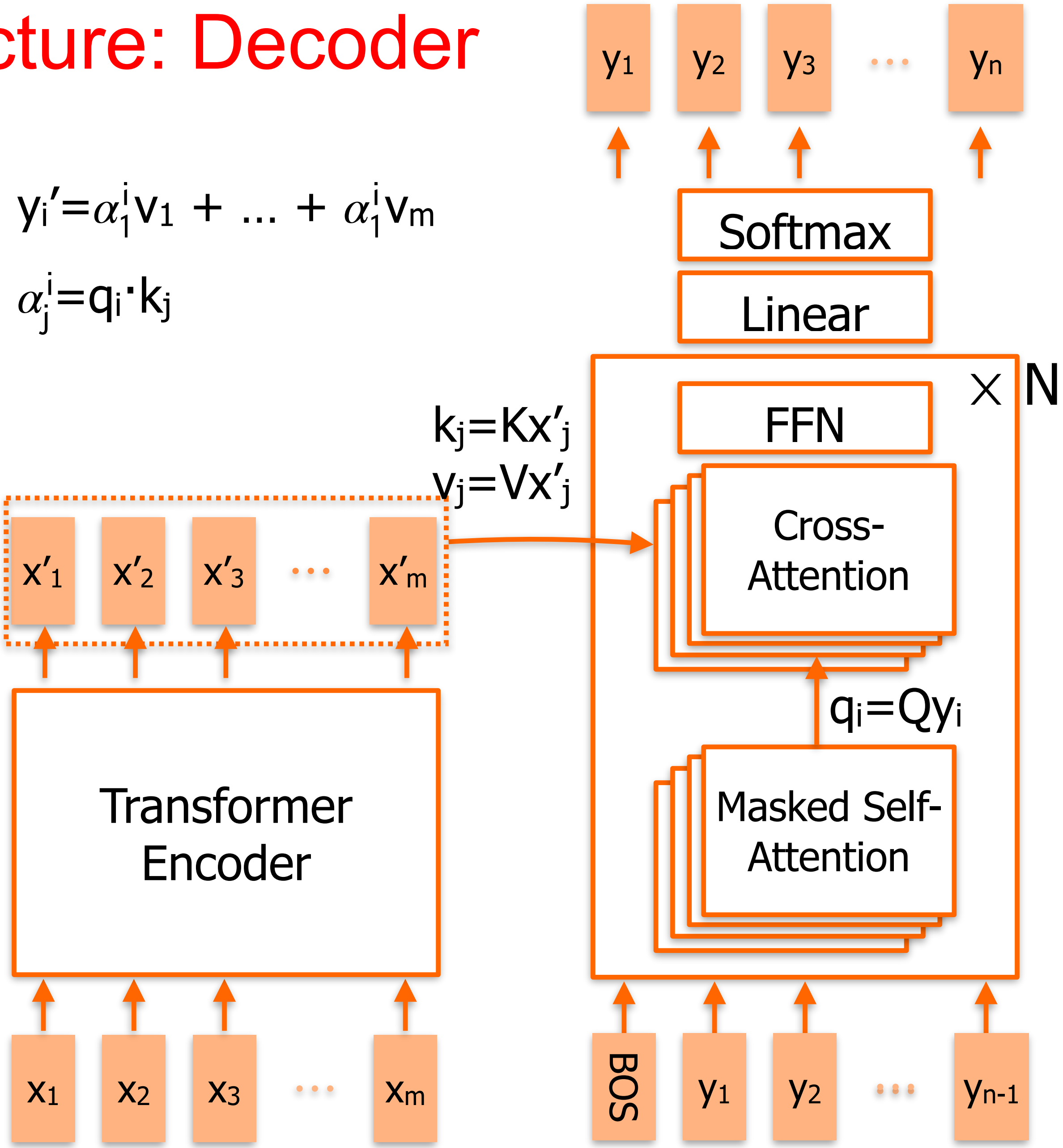


Transformer Architecture: Decoder

- The decoder is similar to the encoder.
- **Masked self-attention**
  - Later inputs are **not attended** to (i.e., attention weights α_j^i for later inputs are zero) → Transformer Training
- **Linear + Softmax**
- **Cross-attention**
  - The **queries** are obtained from the output vectors of the previous **decoder** layer.
  - The **keys** and **values** are from the output vectors of the **encoder**.

$y_i' = \alpha_1^i v_1 + \ldots + \alpha_1^i v_m$

$\alpha_j^i = q_i \cdot k_j$

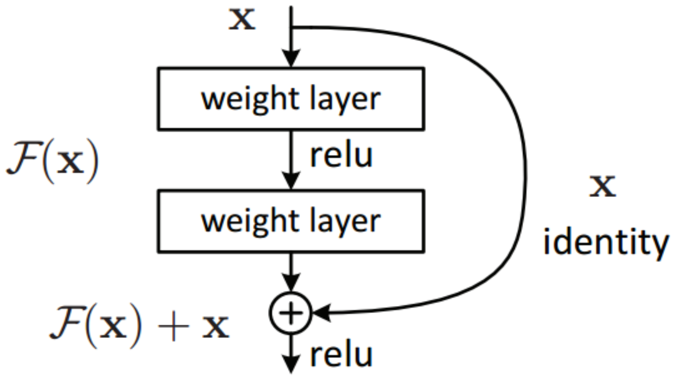


Transformer Architictre: Further details

1. https://towardsdatascience.com/residual-blocks-building-blocks-of-resnet-fd90ca15d6ec
2. https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

Transformer Architictre: Further details

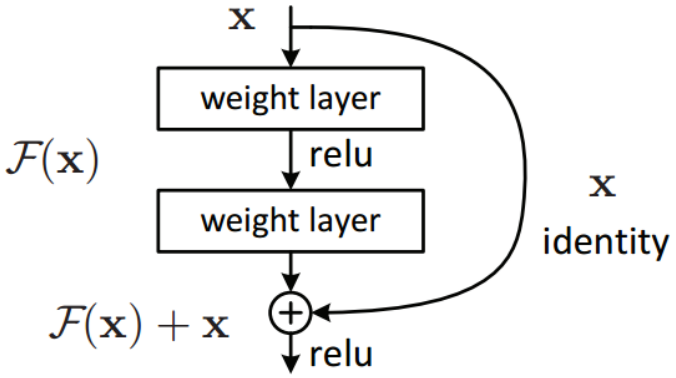
- Residual connection



1. https://towardsdatascience.com/residual-blocks-building-blocks-of-resnet-fd90ca15d6ec
2. https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

Transformer Architictre: Further details

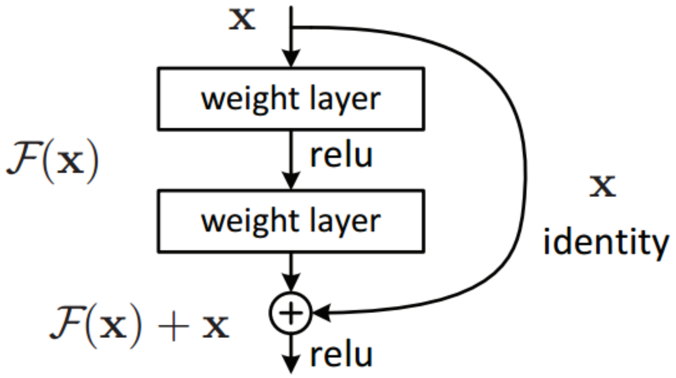
- Residual connection
  - Model is less dependent on the number of layers.



1. https://towardsdatascience.com/residual-blocks-building-blocks-of-resnet-fd90ca15d6ec
2. https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

Transformer Architictre: Further details

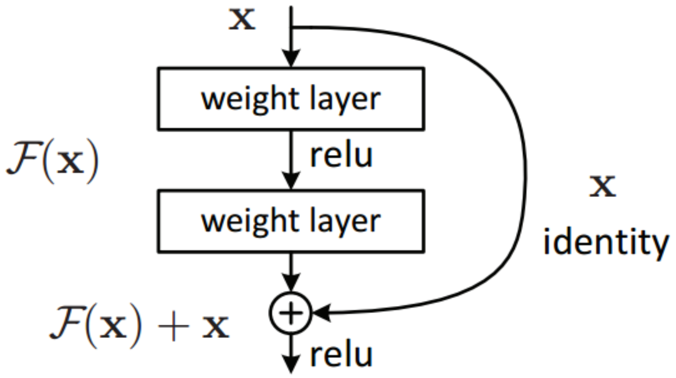
- Residual connection
  - Model is less dependent on the number of layers.
- Layer normalization



1. https://towardsdatascience.com/residual-blocks-building-blocks-of-resnet-fd90ca15d6ec
2. https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

Transformer Architictre: Further details

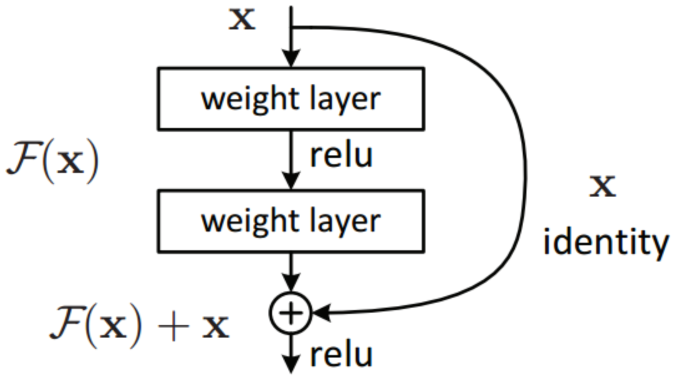
- Residual connection
  - Model is less dependent on the number of layers.
- Layer normalization
  - Normalize features of each vector using their mean and variance.



1. https://towardsdatascience.com/residual-blocks-building-blocks-of-resnet-fd90ca15d6ec
2. https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

Transformer Architictre: Further details

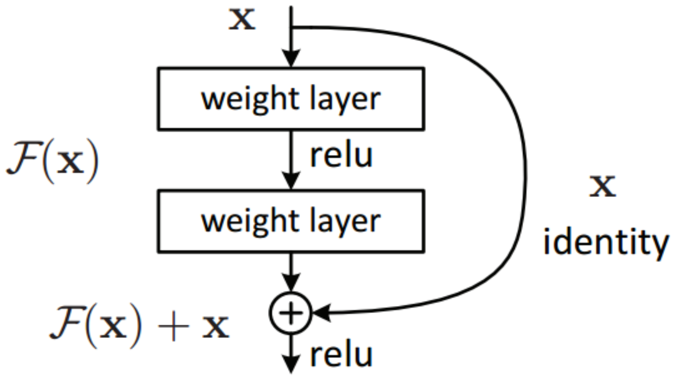
- Residual connection
  - Model is less dependent on the number of layers.
- Layer normalization
  - Normalize features of each vector using their mean and variance.
  - Faster and more stable training.



1. https://towardsdatascience.com/residual-blocks-building-blocks-of-resnet-fd90ca15d6ec
2. https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

Transformer Architictre: Further details

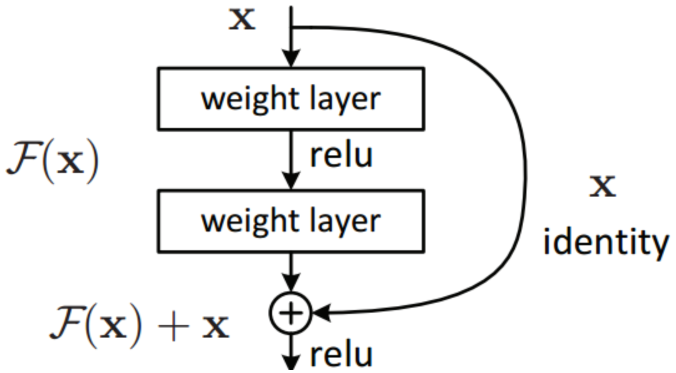
- Residual connection
  - Model is less dependent on the number of layers.
- Layer normalization
  - Normalize features of each vector using their mean and variance.
  - Faster and more stable training.
- Positional Encoding
  - For preserving the input order.



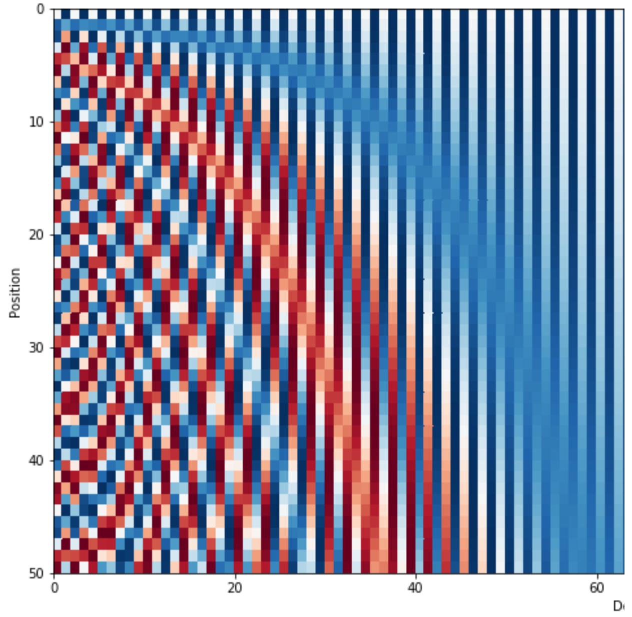
1. https://towardsdatascience.com/residual-blocks-building-blocks-of-resnet-fd90ca15d6ec
2. https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

Transformer Architictre: Further details

- Residual connection
  - Model is less dependent on the number of layers.
- Layer normalization
  - Normalize features of each vector using their mean and variance.
  - Faster and more stable training.
- Positional Encoding
  - For preserving the input order.
  - Use sine and cosine (similar to binary representation of numbers)



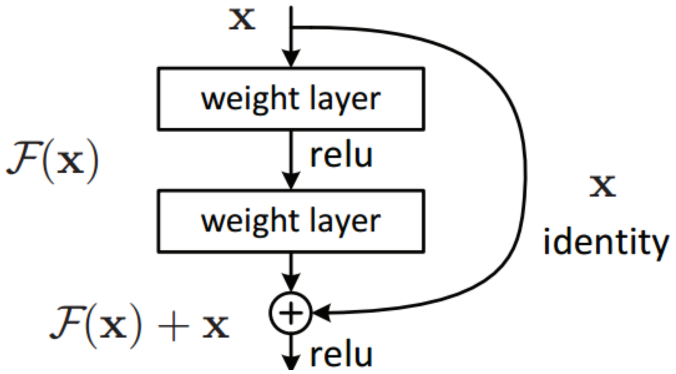
$$\begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \\ \vdots \\ \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}$$



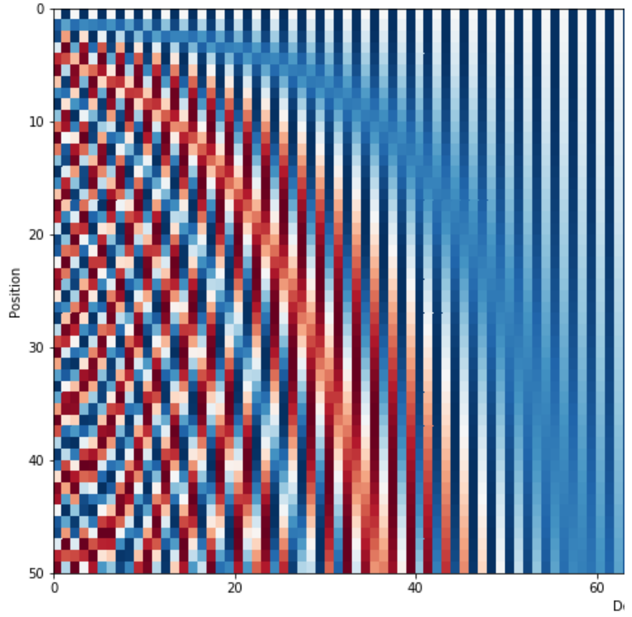
1. https://towardsdatascience.com/residual-blocks-building-blocks-of-resnet-fd90ca15d6ec
2. https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

Transformer Architictre: Further details

- Residual connection
  - Model is less dependent on the number of layers.
- Layer normalization
  - Normalize features of each vector using their mean and variance.
  - Faster and more stable training.
- Positional Encoding
  - For preserving the input order.
  - Use sine and cosine (similar to binary representation of numbers)
- Vectorization
  - Faster training and inference due to parallel processing.



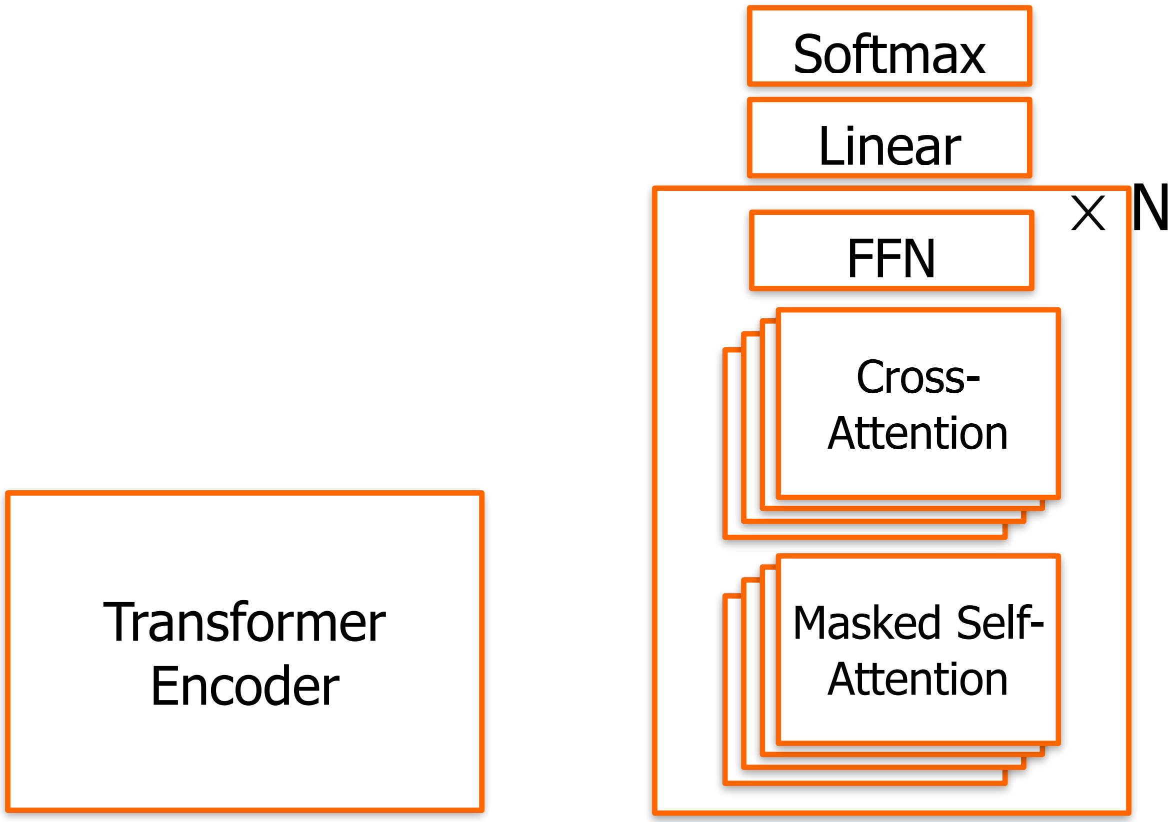
$$\begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \\ \vdots \\ \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}$$



1. https://towardsdatascience.com/residual-blocks-building-blocks-of-resnet-fd90ca15d6ec
2. https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

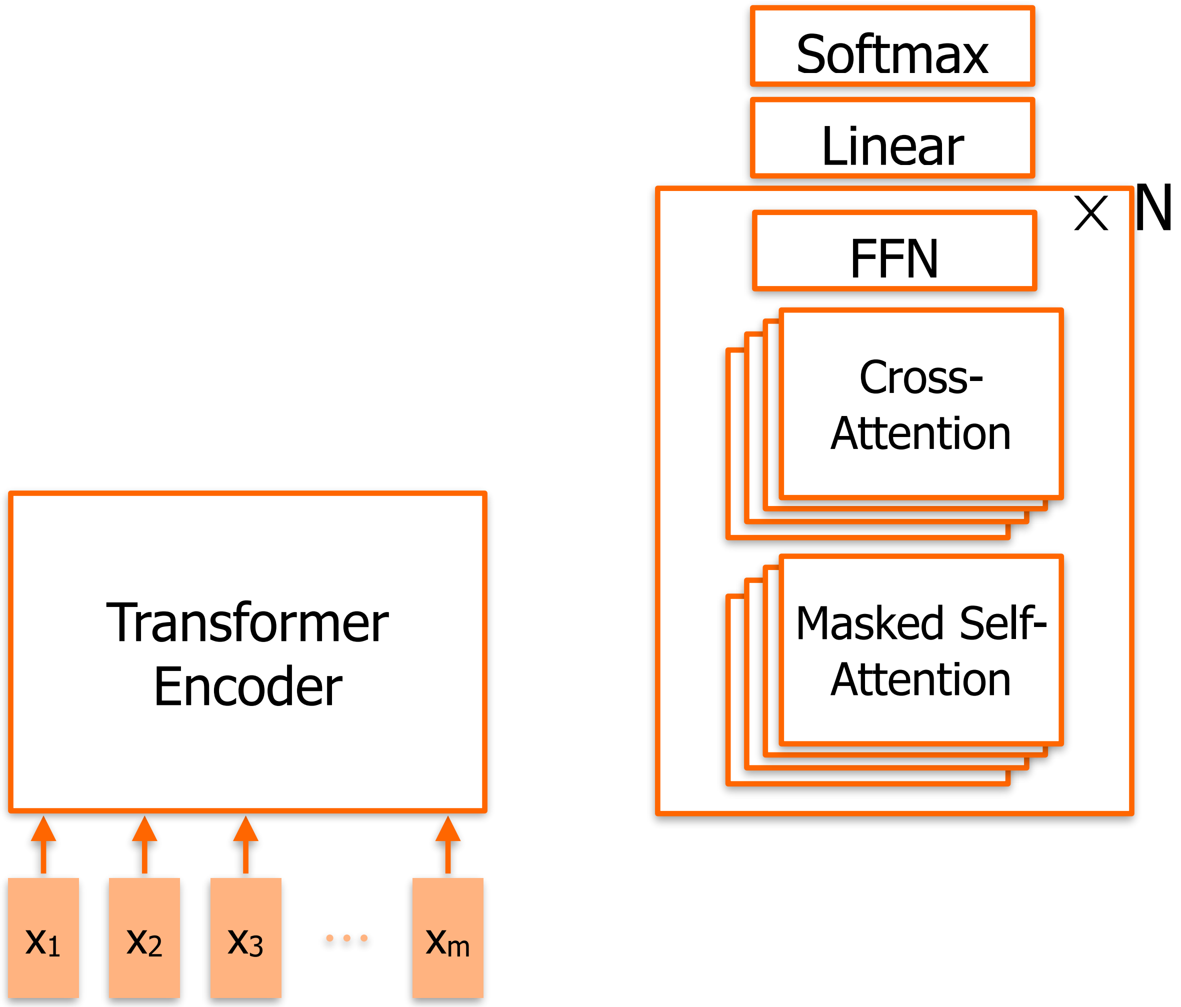
Transformer Inference

- First, encoder outputs $x'_1, x'_2, \ldots, x'_m$



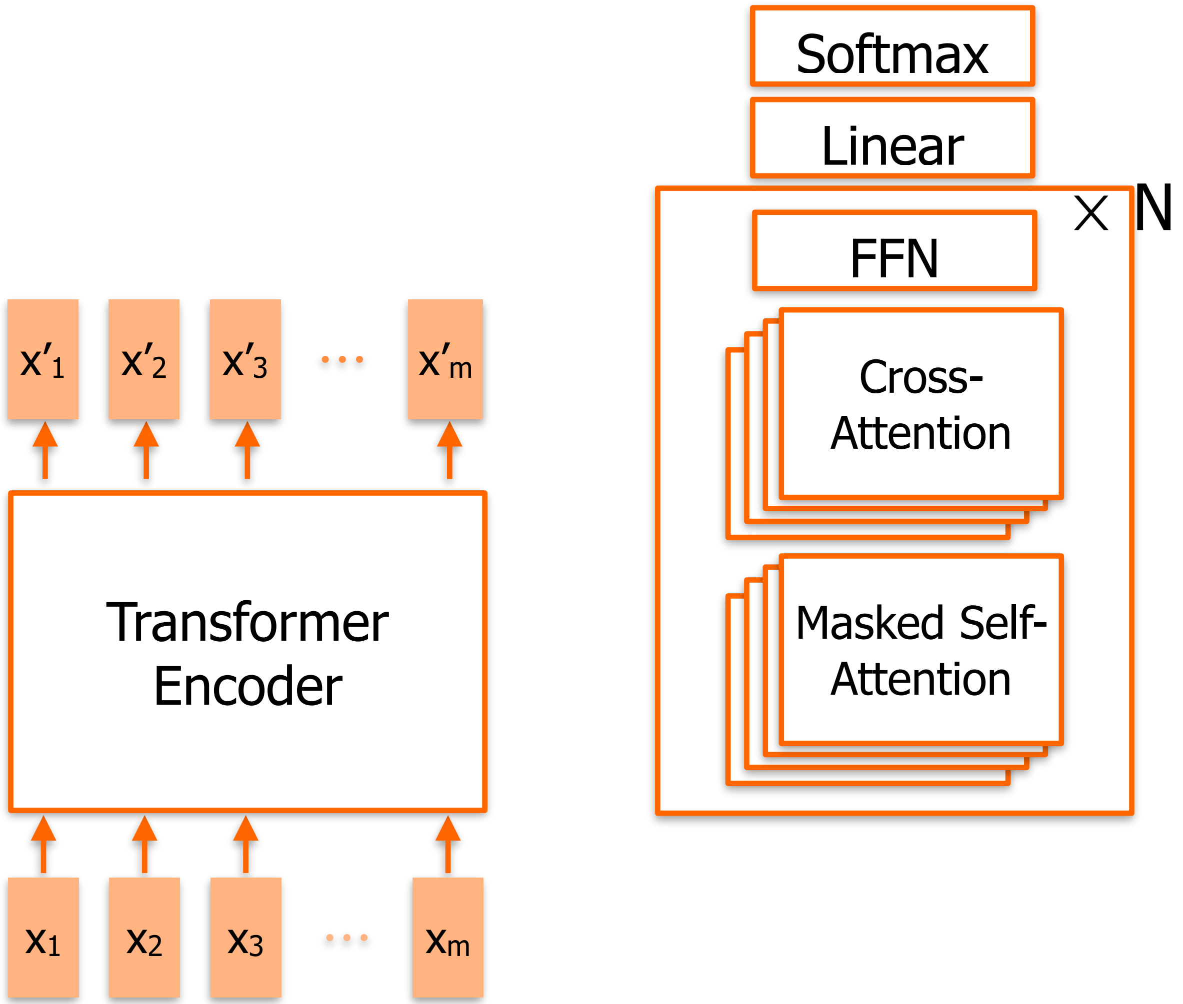
Transformer Inference

- First, encoder outputs $x'_1, x'_2, \ldots, x'_m$



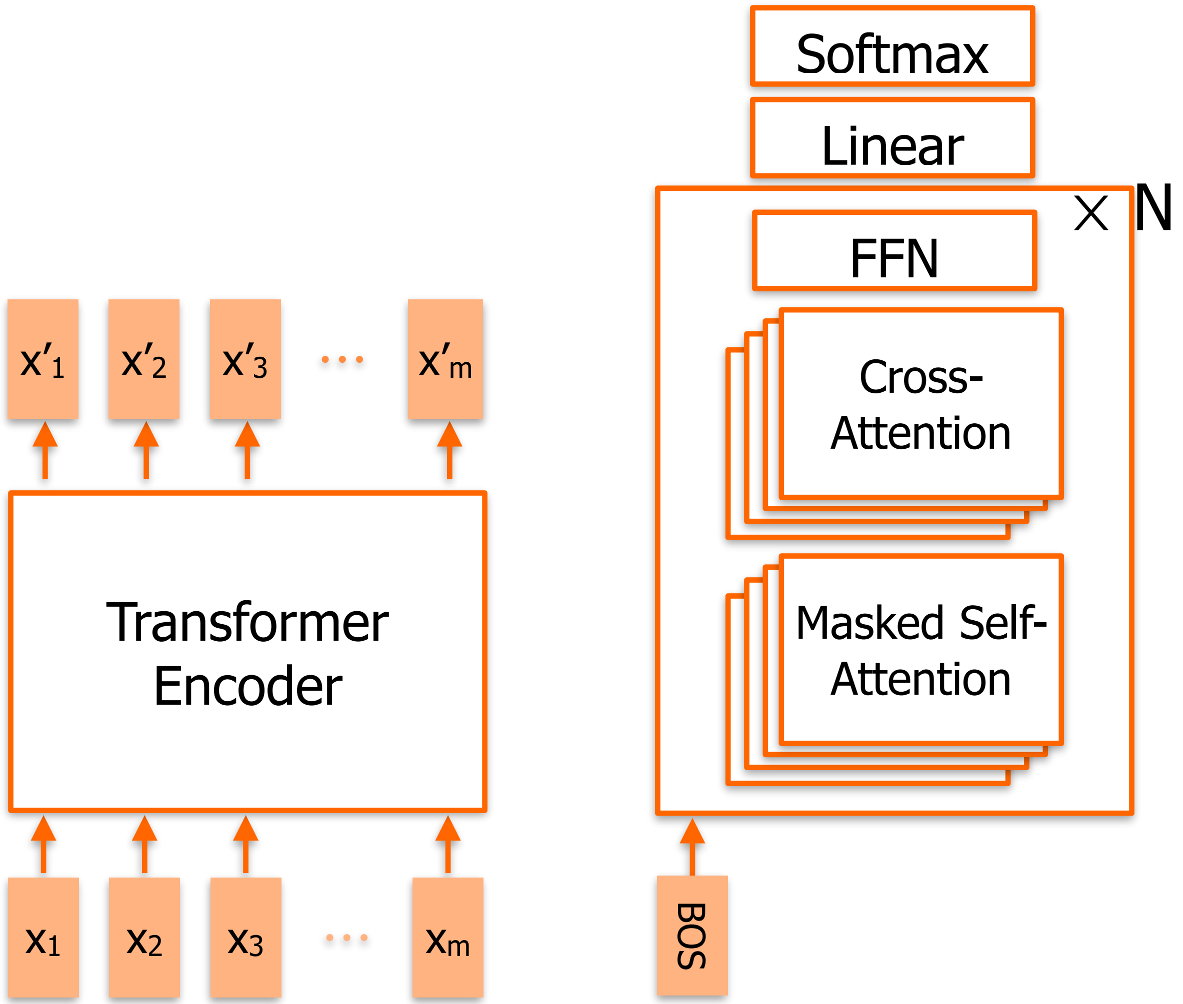
Transformer Inference

- First, encoder outputs $x'_1, x'_2, \ldots, x'_m$



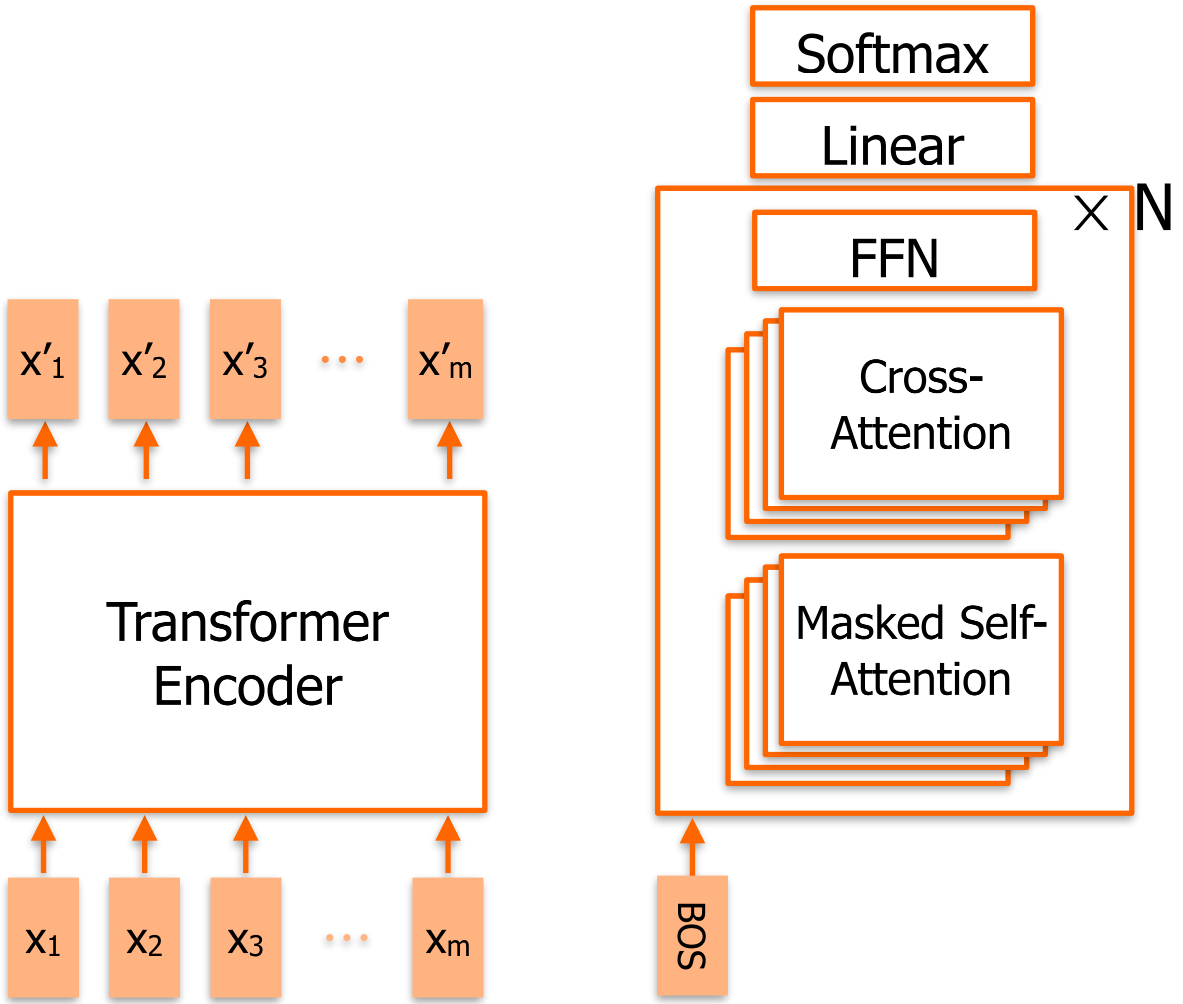
Transformer Inference

- First, encoder outputs $x'_1, x'_2, \ldots, x'_m$
- The "**BOS**" vector is the **first input** to the decoder.
  - "BOS": a vector representing the **beginning of sentence**.



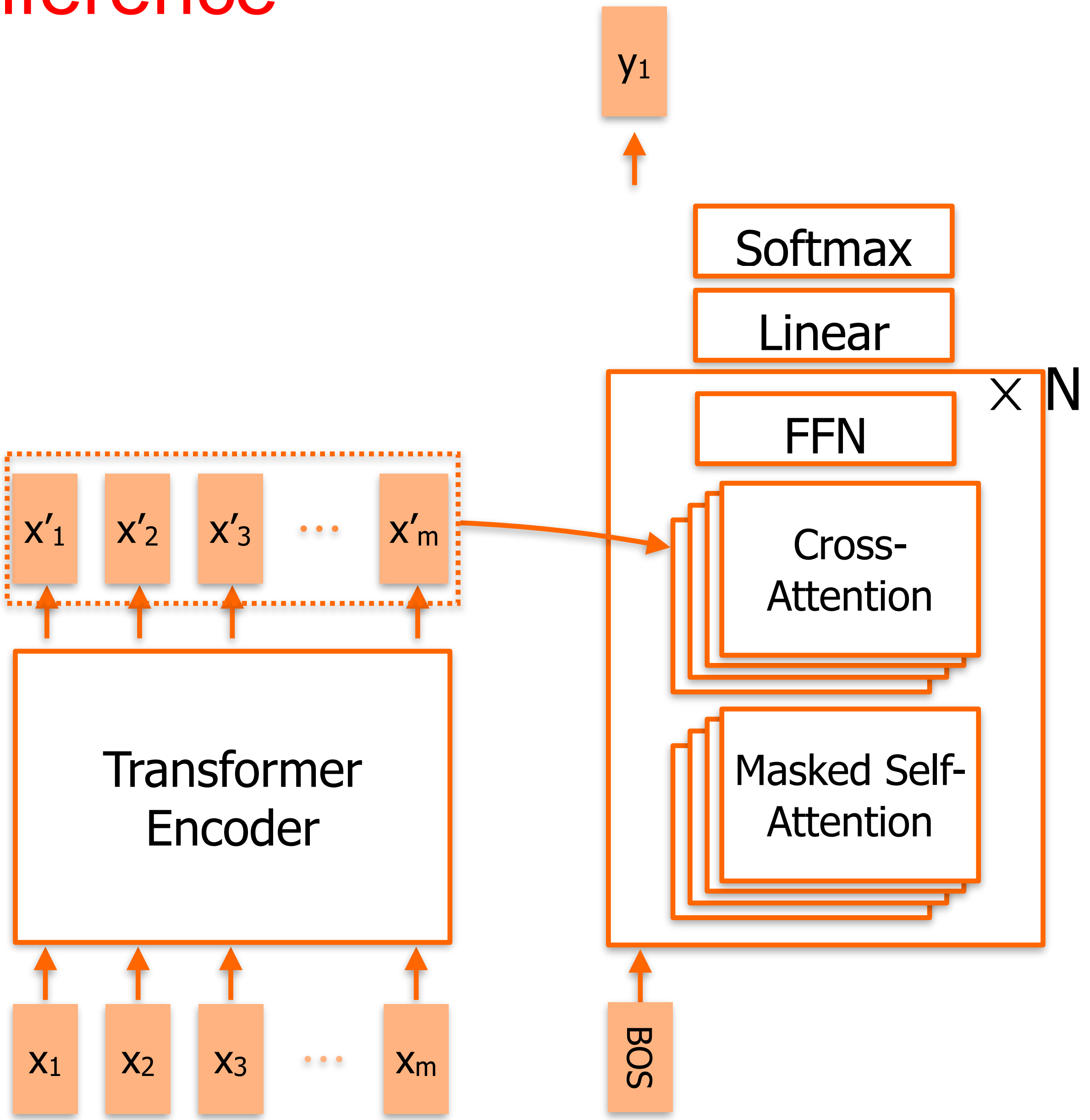
Transformer Inference

- First, encoder outputs $x'_1, x'_2, \ldots, x'_m$
- The "**BOS**" vector is the **first input** to the decoder.
  - "BOS": a vector representing the **beginning of sentence**.
- Decoder is **autoregressive**, i.e.,
  - output y_t of the decoder becomes the input to the decoder at the next step.



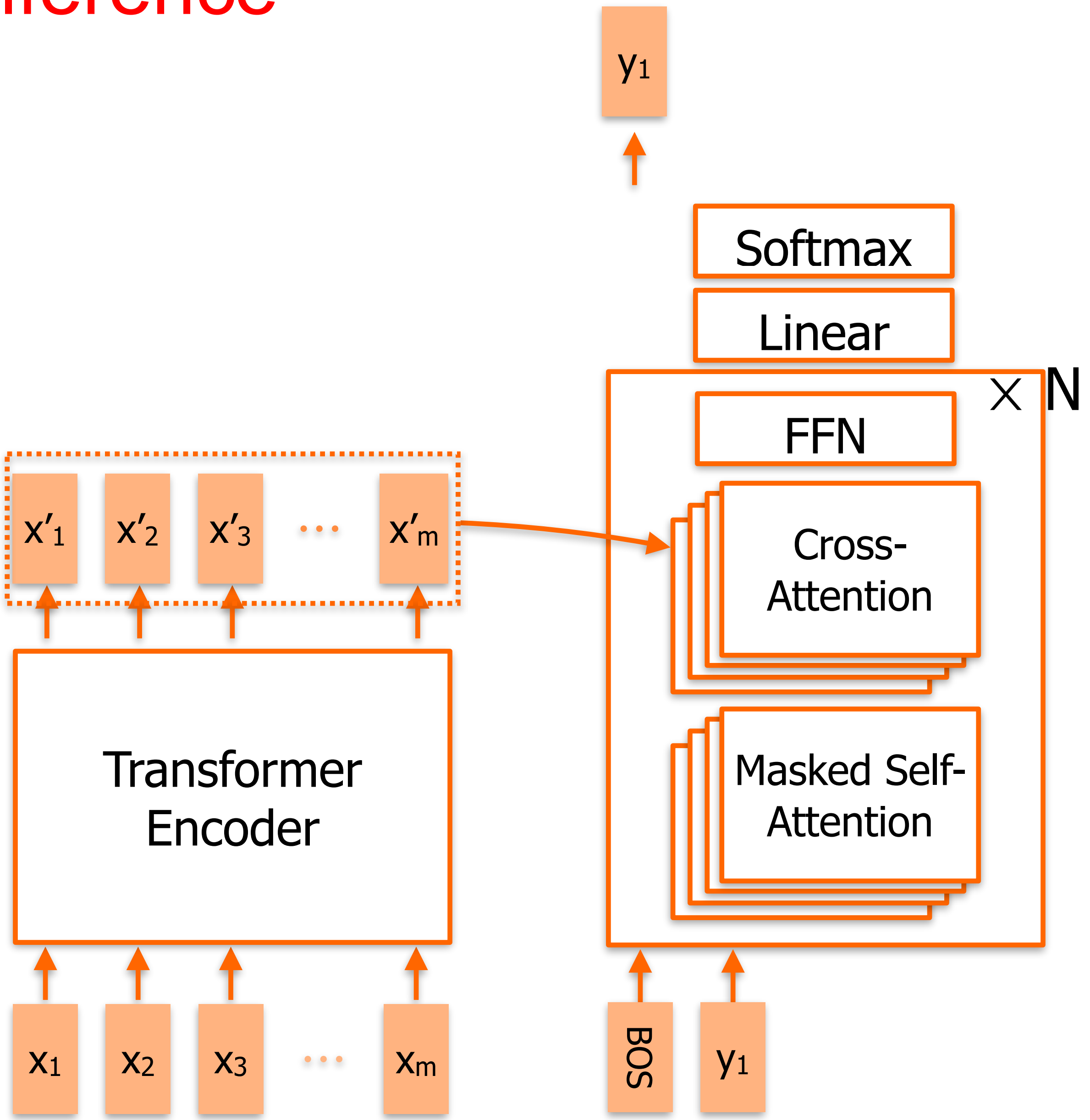
Transformer Inference

- First, encoder outputs $x'_1, x'_2, \ldots, x'_m$
- The "**BOS**" vector is the **first input** to the decoder.
  - "BOS": a vector representing the **beginning of sentence**.
- Decoder is **autoregressive**, i.e.,
  - output y_t of the decoder becomes the input to the decoder at the next step.



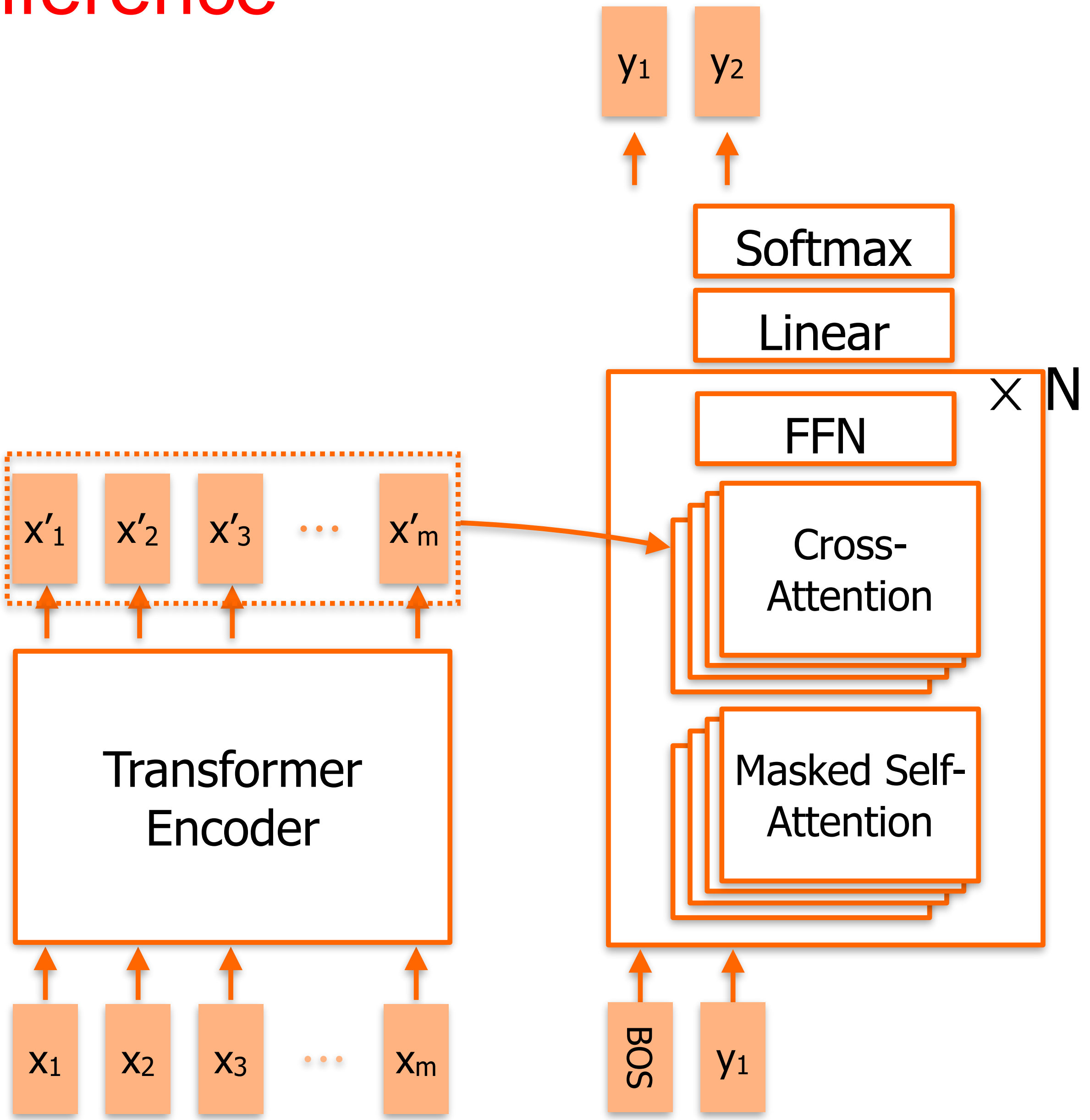
Transformer Inference

- First, encoder outputs $x'_1, x'_2, \ldots, x'_m$
- The “**BOS**” vector is the **first input** to the decoder.
  - “BOS”: a vector representing the **beginning of sentence**.
- Decoder is **autoregressive**, i.e.,
  - output y_t of the decoder becomes the input to the decoder at the next step.



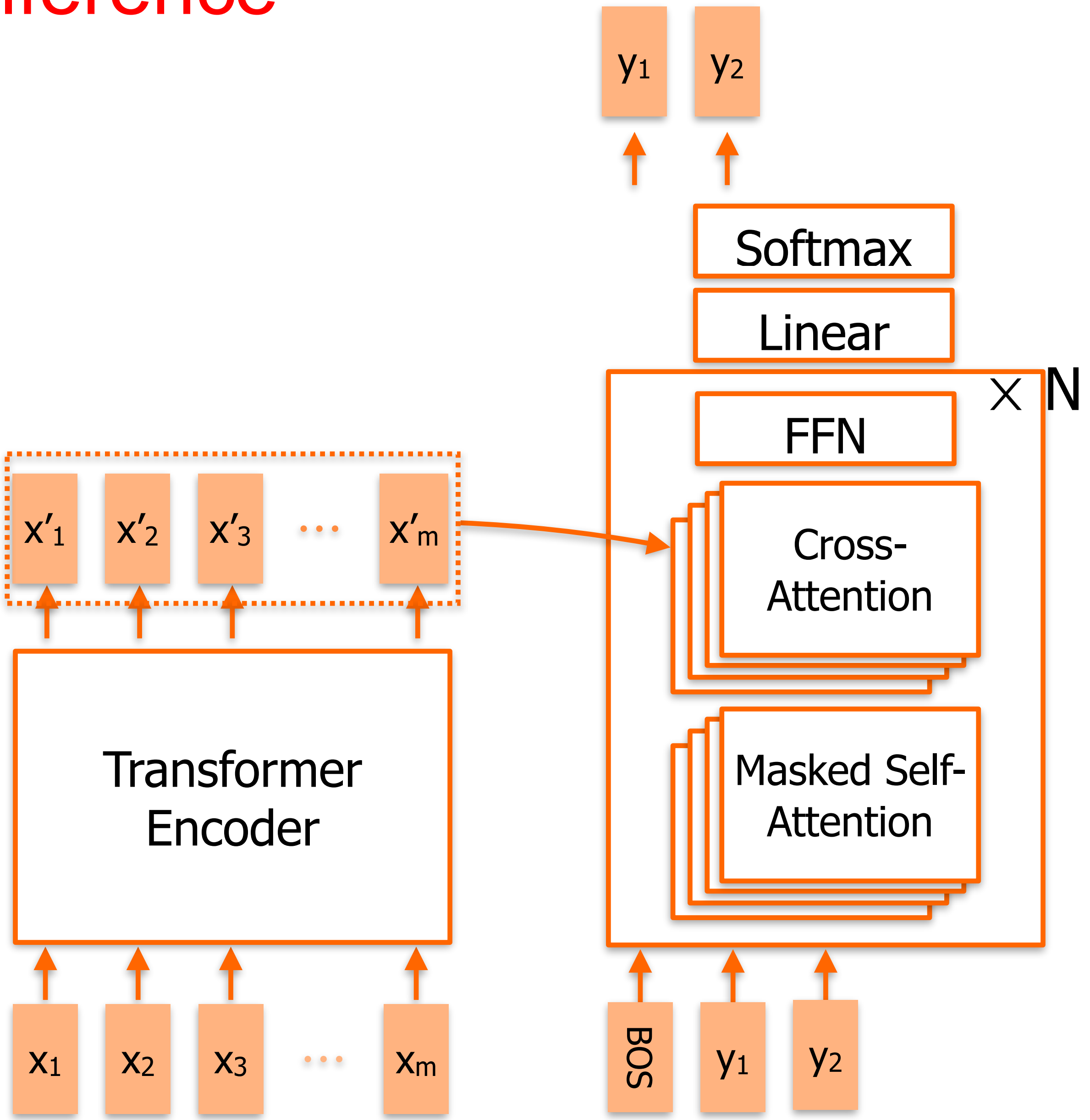
Transformer Inference

- First, encoder outputs $x'_1, x'_2, \ldots, x'_m$
- The "**BOS**" vector is the **first input** to the decoder.
  - "BOS": a vector representing the **beginning of sentence**.
- Decoder is **autoregressive**, i.e.,
  - output y_t of the decoder becomes the input to the decoder at the next step.



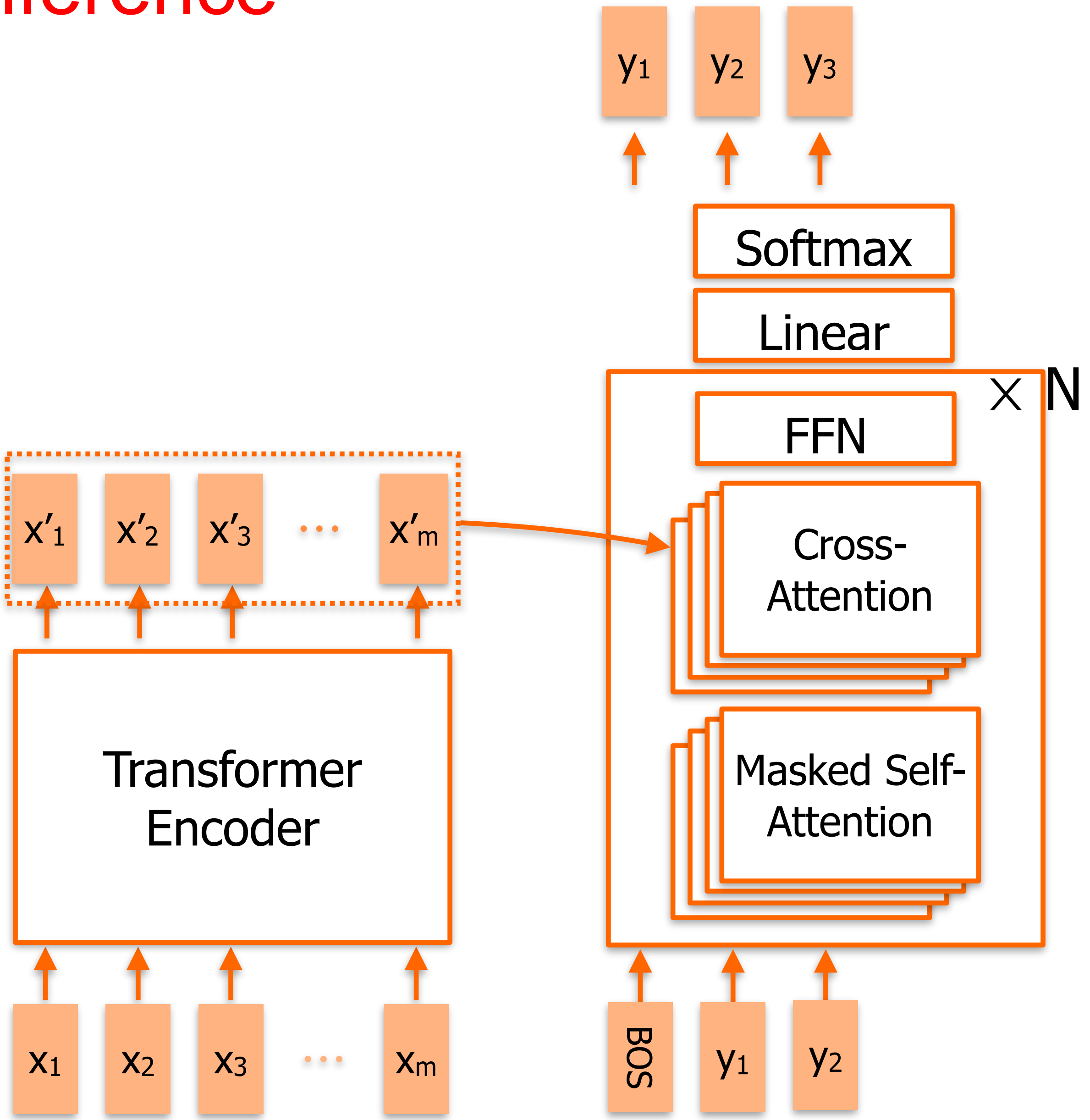
Transformer Inference

- First, encoder outputs $x'_1, x'_2, \ldots, x'_m$
- The "**BOS**" vector is the **first input** to the decoder.
  - "BOS": a vector representing the **beginning of sentence**.
- Decoder is **autoregressive**, i.e.,
  - output y_t of the decoder becomes the input to the decoder at the next step.



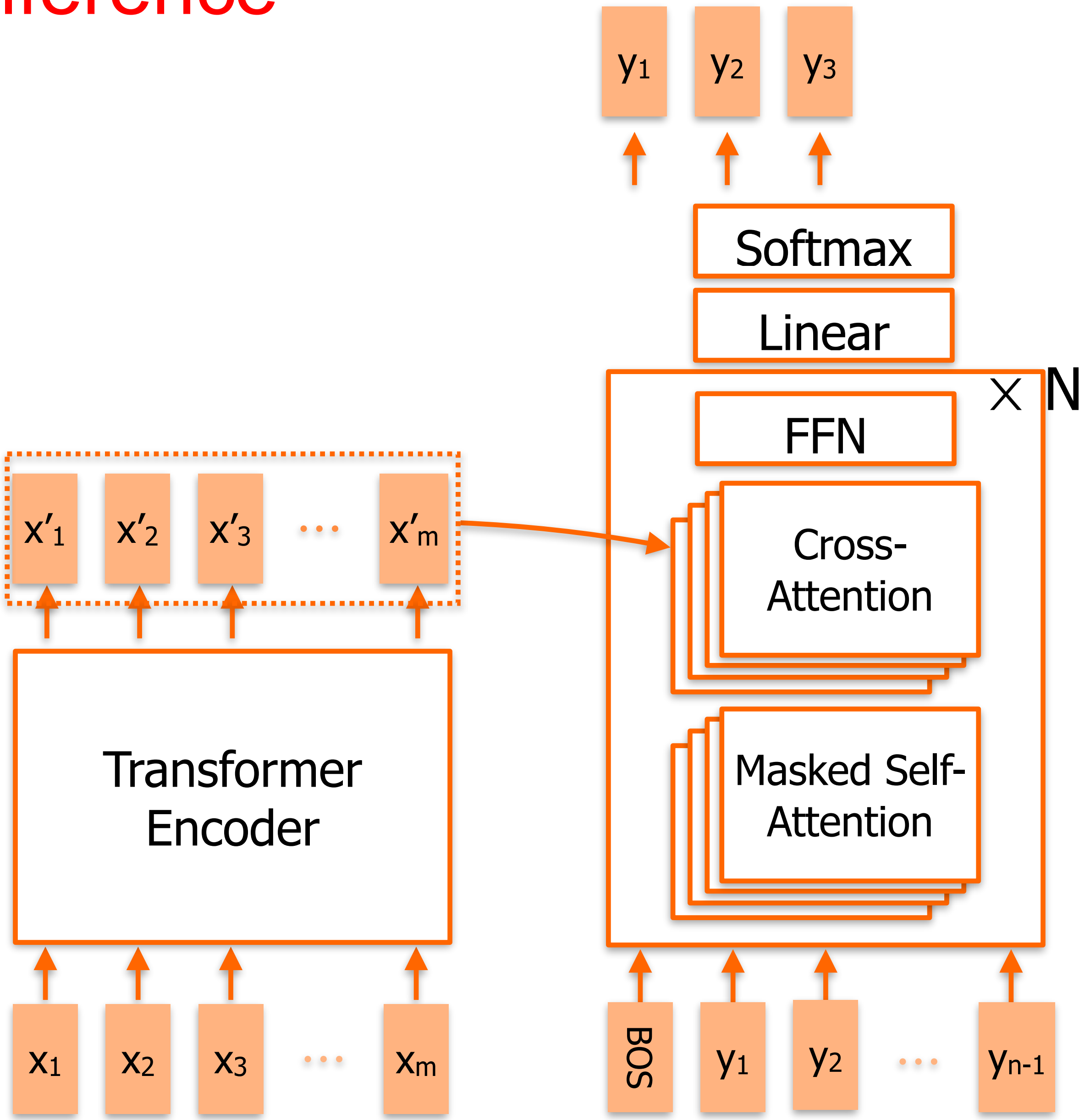
Transformer Inference

- First, encoder outputs $x'_1, x'_2, \ldots, x'_m$
- The "**BOS**" vector is the **first input** to the decoder.
  - "BOS": a vector representing the **beginning of sentence**.
- Decoder is **autoregressive**, i.e.,
  - output y_t of the decoder becomes the input to the decoder at the next step.



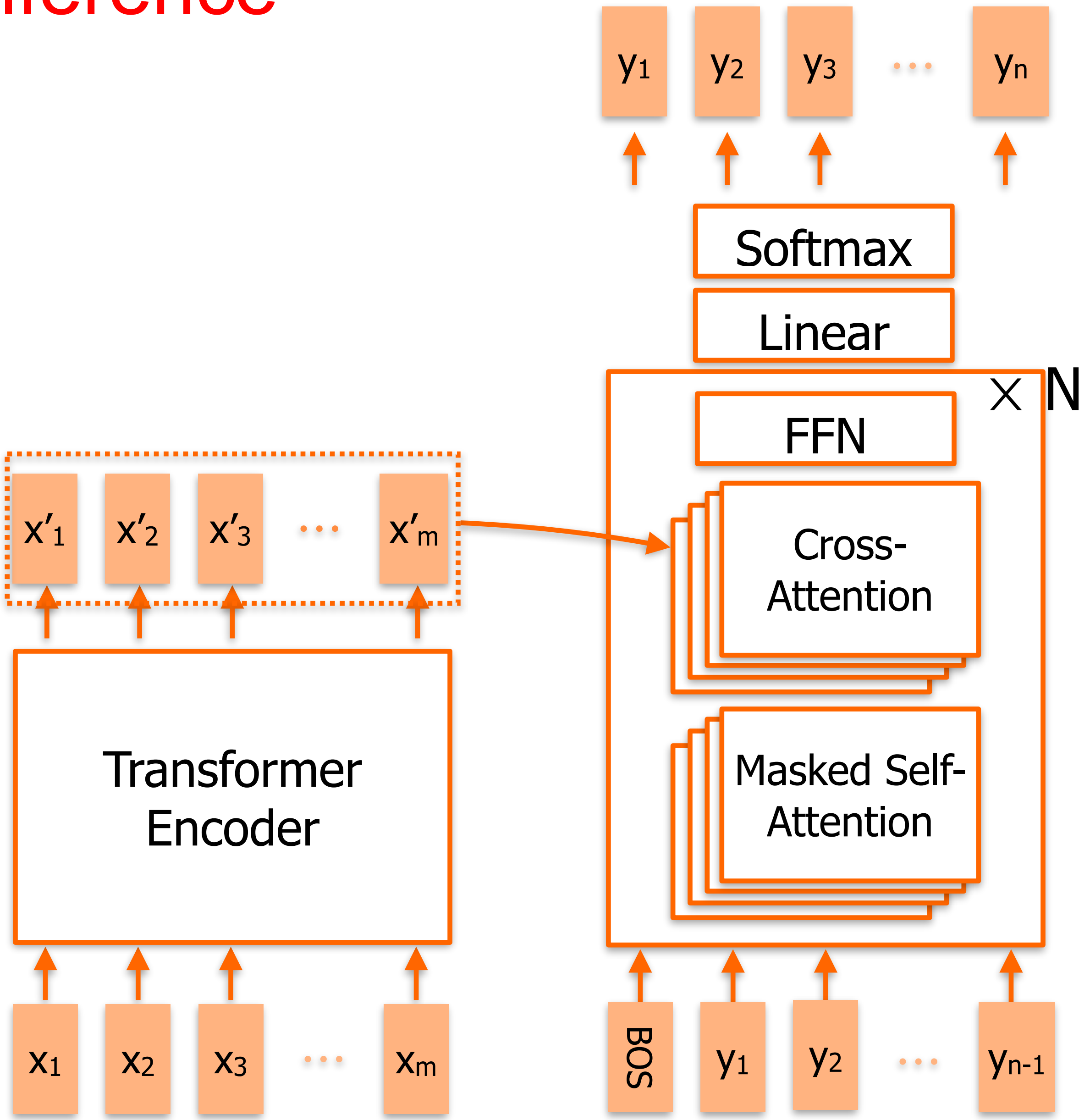
Transformer Inference

- First, encoder outputs $x'_1, x'_2, \ldots, x'_m$
- The "**BOS**" vector is the **first input** to the decoder.
  - "BOS": a vector representing the **beginning of sentence**.
- Decoder is **autoregressive**, i.e.,
  - output y_t of the decoder becomes the input to the decoder at the next step.



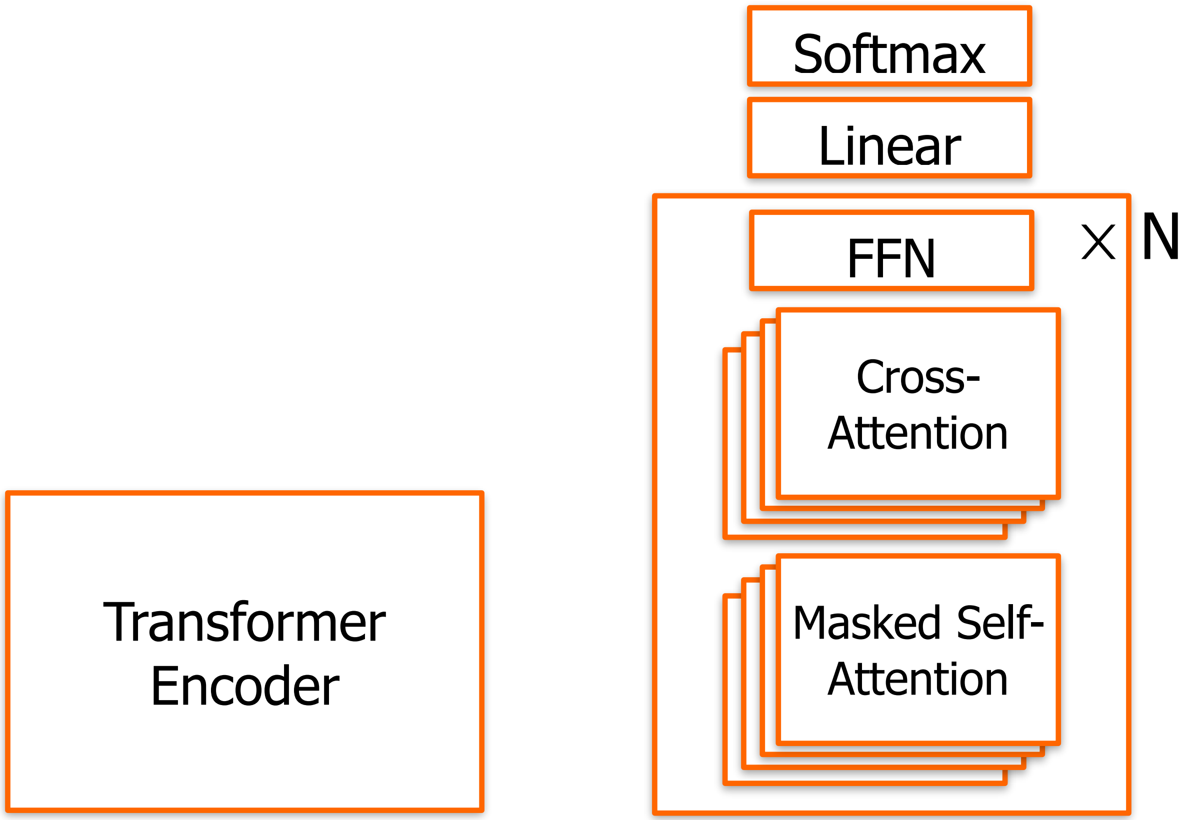
Transformer Inference

- First, encoder outputs $x'_1, x'_2, \ldots, x'_m$
- The "**BOS**" vector is the **first input** to the decoder.
  - "BOS": a vector representing the **beginning of sentence**.
- Decoder is **autoregressive**, i.e.,
  - output y_t of the decoder becomes the input to the decoder at the next step.



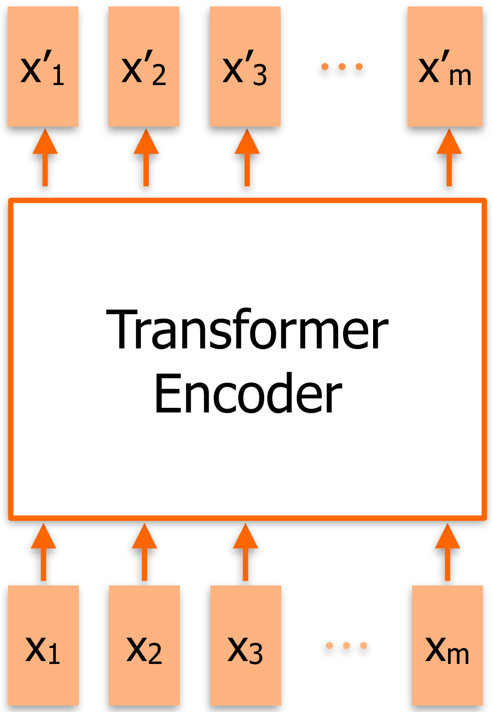
Transformer Training

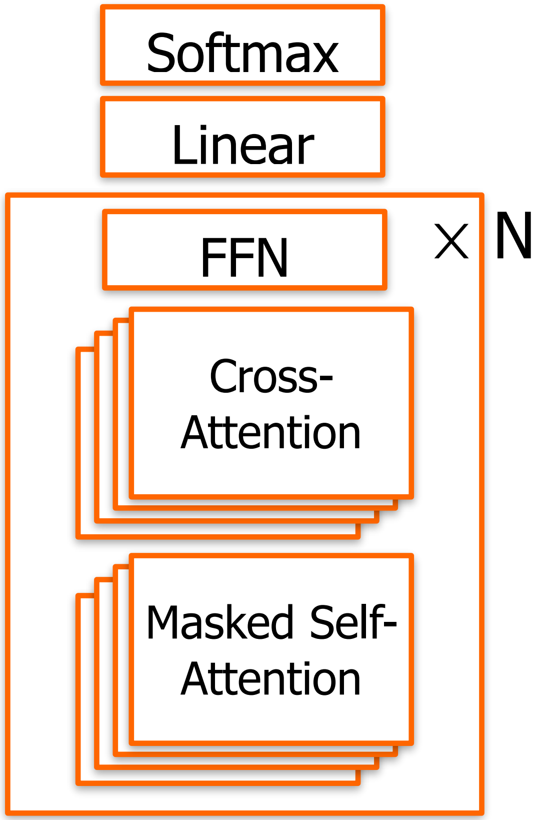
- First, encoder outputs $x'_1, x'_2, \ldots, x'_m$



Transformer Training

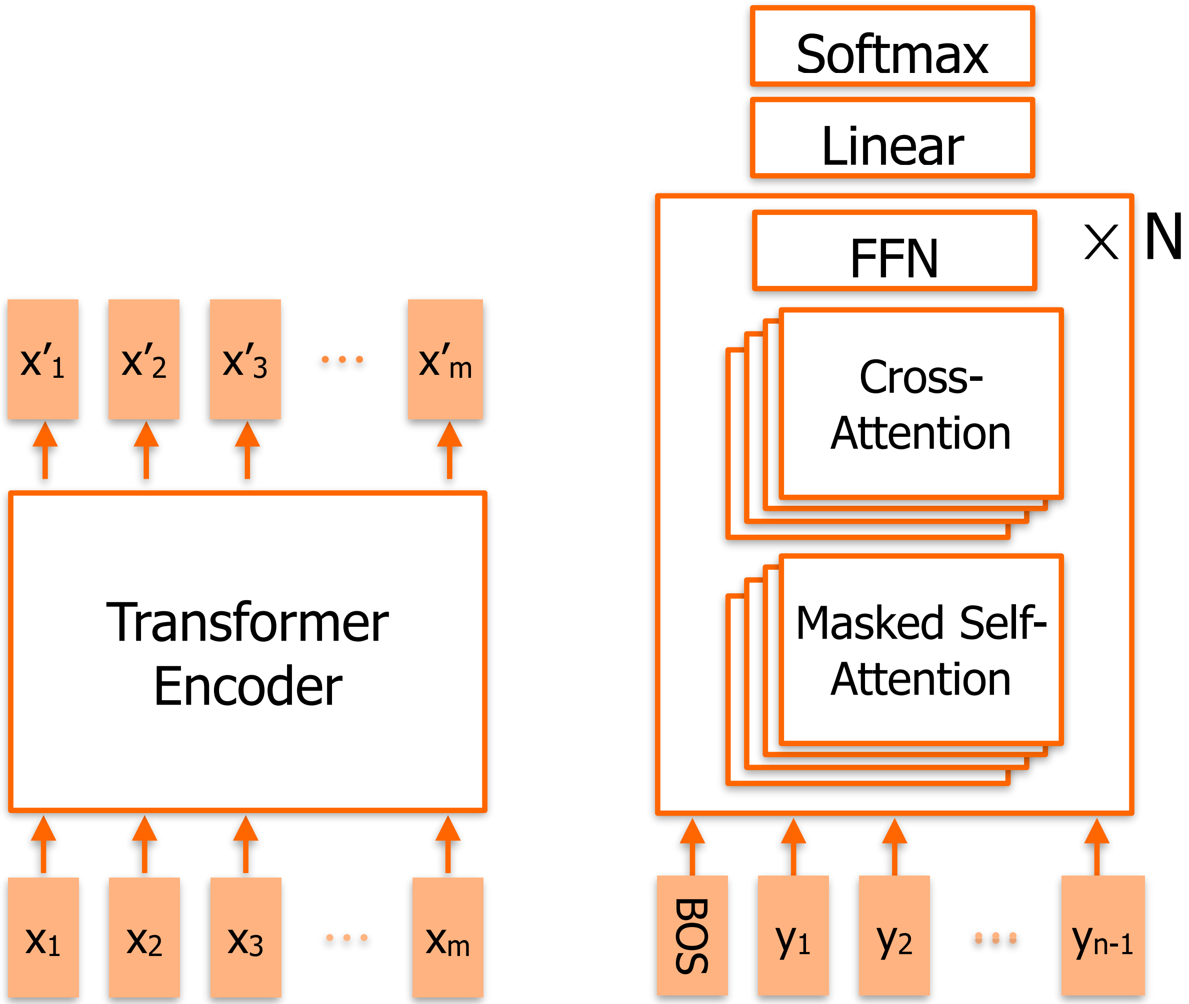
- First, encoder outputs $x'_1, x'_2, \ldots, x'_m$





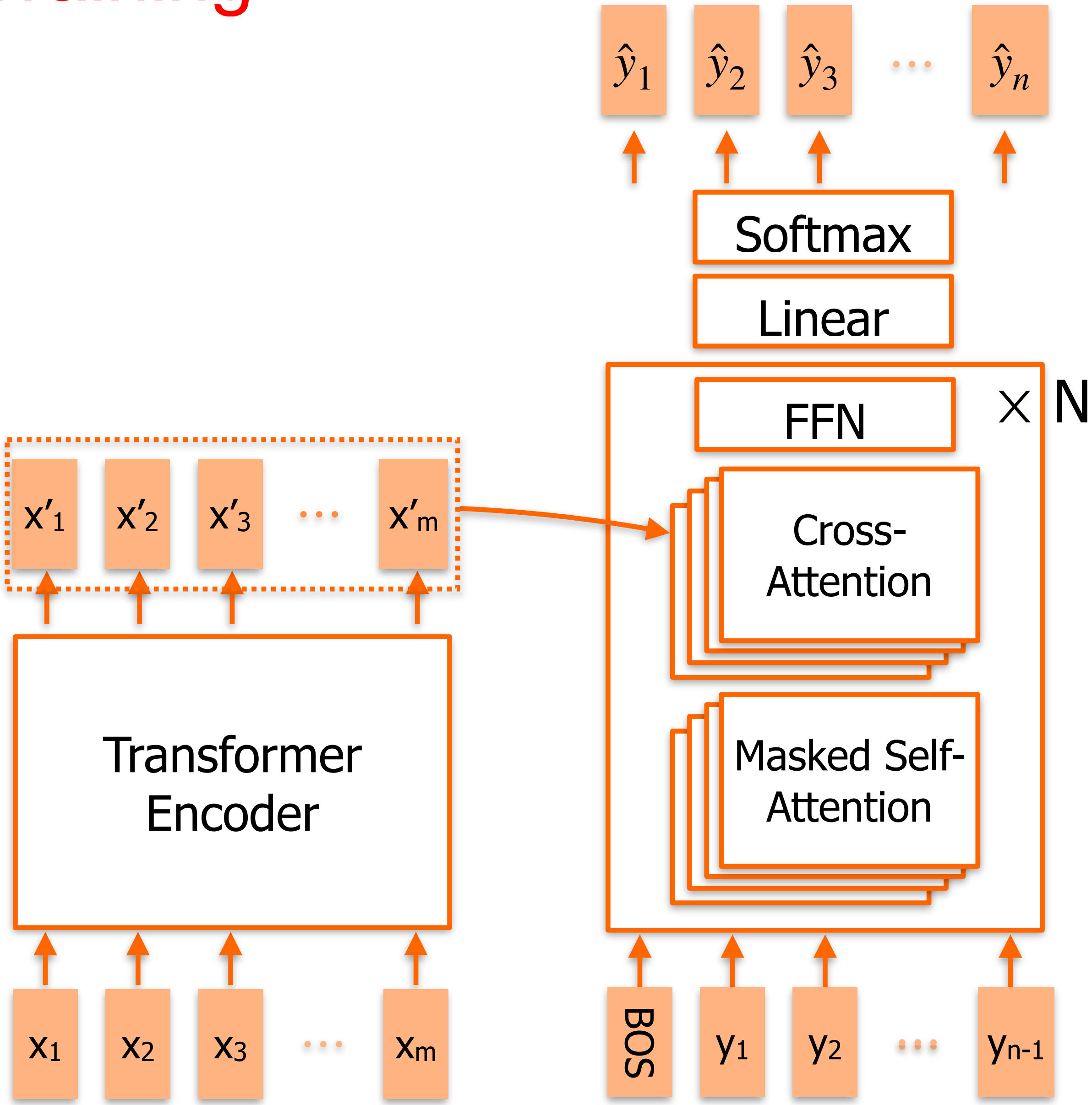
Transformer Training

- First, encoder outputs $x'_1, x'_2, \ldots, x'_m$
- **Ground-truth one-hot vectors** $y_1, y_2, \ldots, y_{n-1}$ with the "BOS" vector are fed to the decoder **simultaneously**.



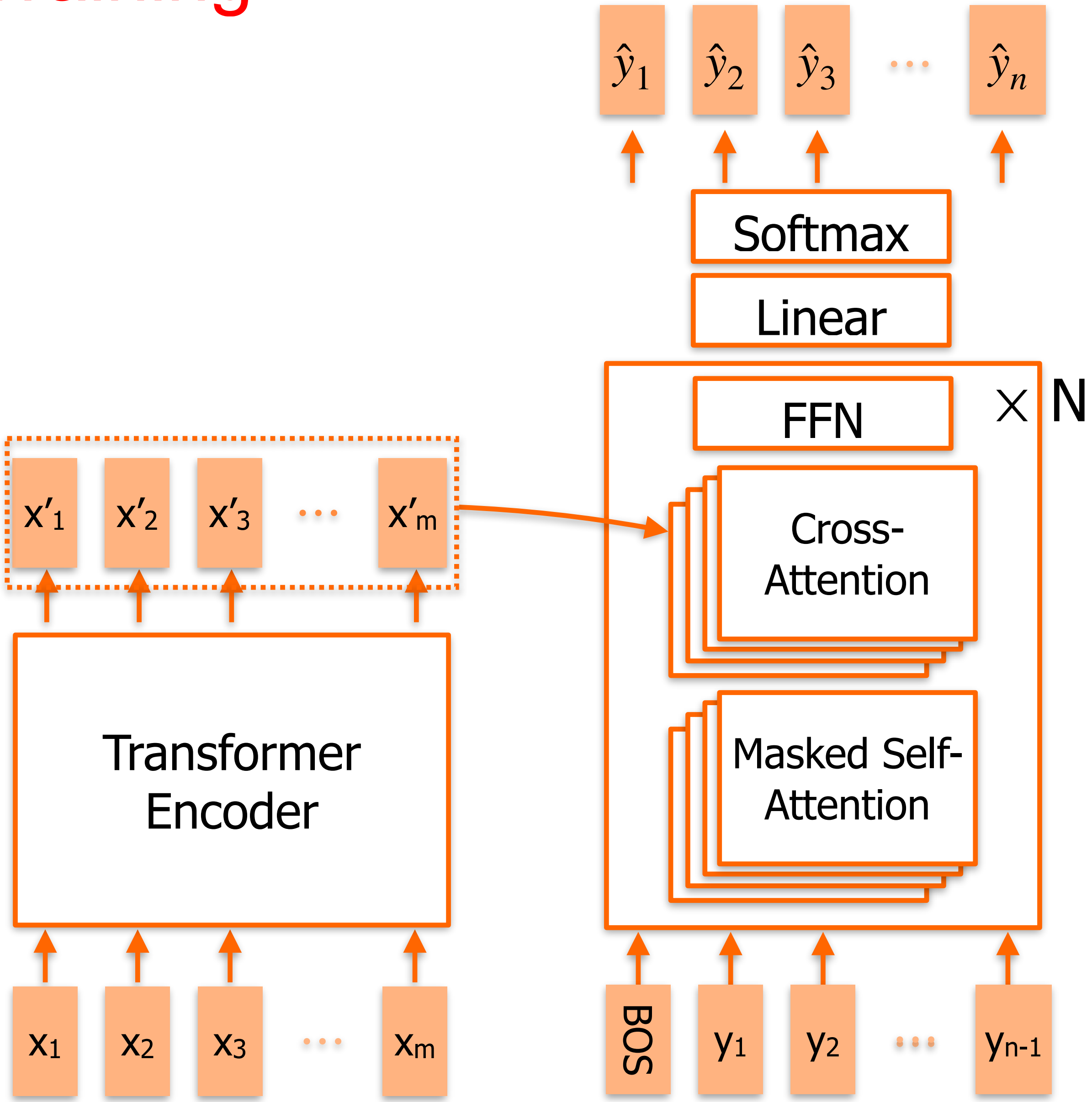
Transformer Training

- First, encoder outputs $x'_1, x'_2, \ldots, x'_m$
- **Ground-truth one-hot vectors** $y_1, y_2, \ldots, y_{n-1}$ with the "BOS" vector are fed to the decoder **simultaneously**.
- Output $\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2, \ldots, \hat{\mathbf{y}}_n$ is compared against ground-truth $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n$ to **compute the loss**.



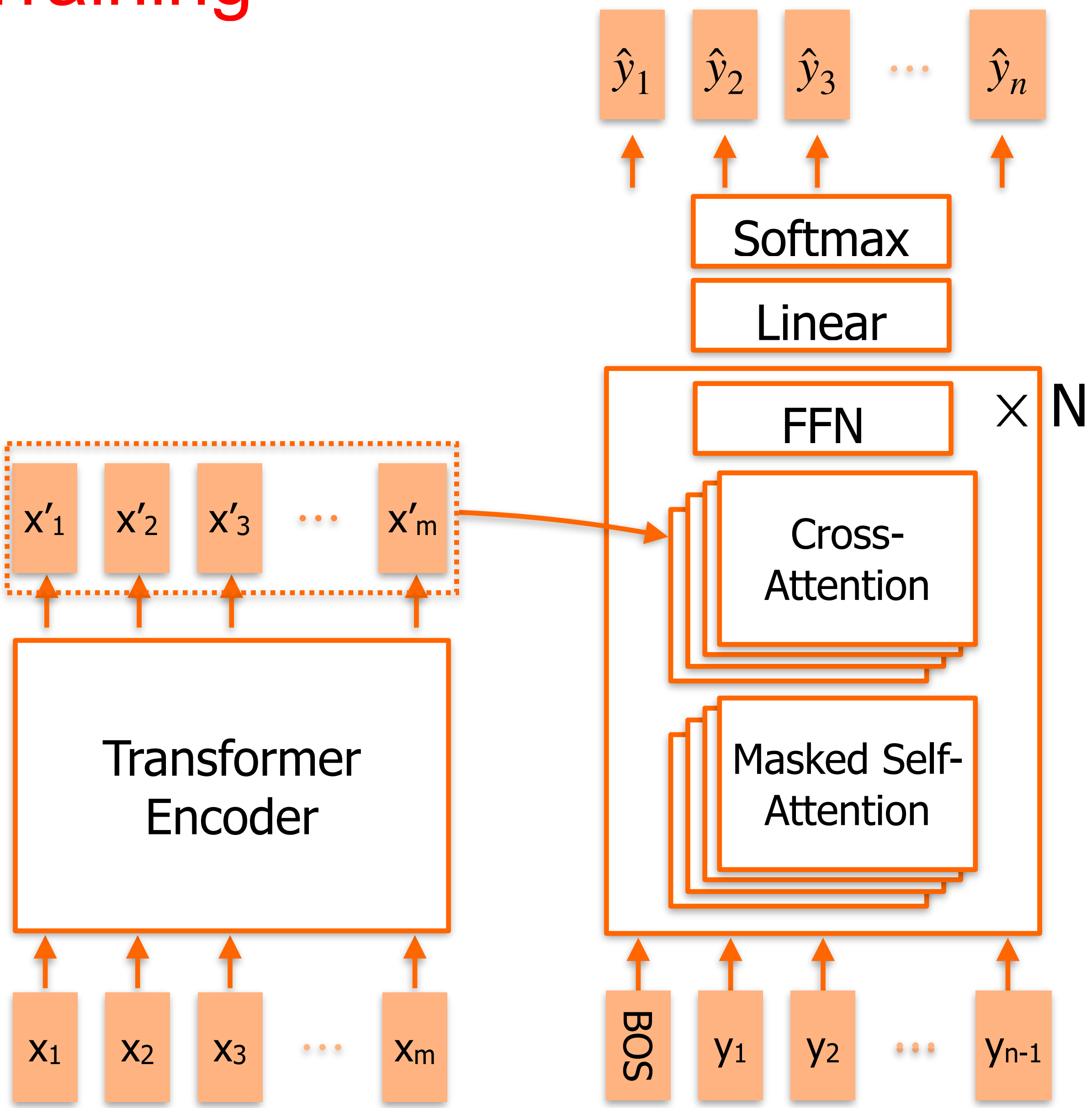
Transformer Training

- First, encoder outputs $x'_1, x'_2, \ldots, x'_m$
- **Ground-truth one-hot vectors** $y_1, y_2, \ldots, y_{n-1}$ with the "BOS" vector are fed to the decoder **simultaneously**.
- Output $\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2, \ldots, \hat{\mathbf{y}}_n$ is compared against ground-truth $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n$ to **compute the loss**.
- Q: If $y_1, y_2, \ldots, y_{n-1}$ is fed to the decoder simultaneously, how does the model (avoid cheating and) learn to predict y_{t+1} from y_t?



Transformer Training

- First, encoder outputs $x'_1, x'_2, \ldots, x'_m$
- **Ground-truth one-hot vectors** $y_1, y_2, \ldots, y_{n-1}$ with the "BOS" vector are fed to the decoder **simultaneously**.
- Output $\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2, \ldots, \hat{\mathbf{y}}_\mathbf{n}$ is compared against ground-truth $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_\mathbf{n}$ to **compute the loss**.
- Q: If $y_1, y_2, \ldots, y_{n-1}$ is fed to the decoder simultaneously, how does the model (avoid cheating and) learn to predict y_{t+1} from y_t?
  - A: Masked self-attention prevents using the information of y_{t+i} with $i \geq 1$.



Summary

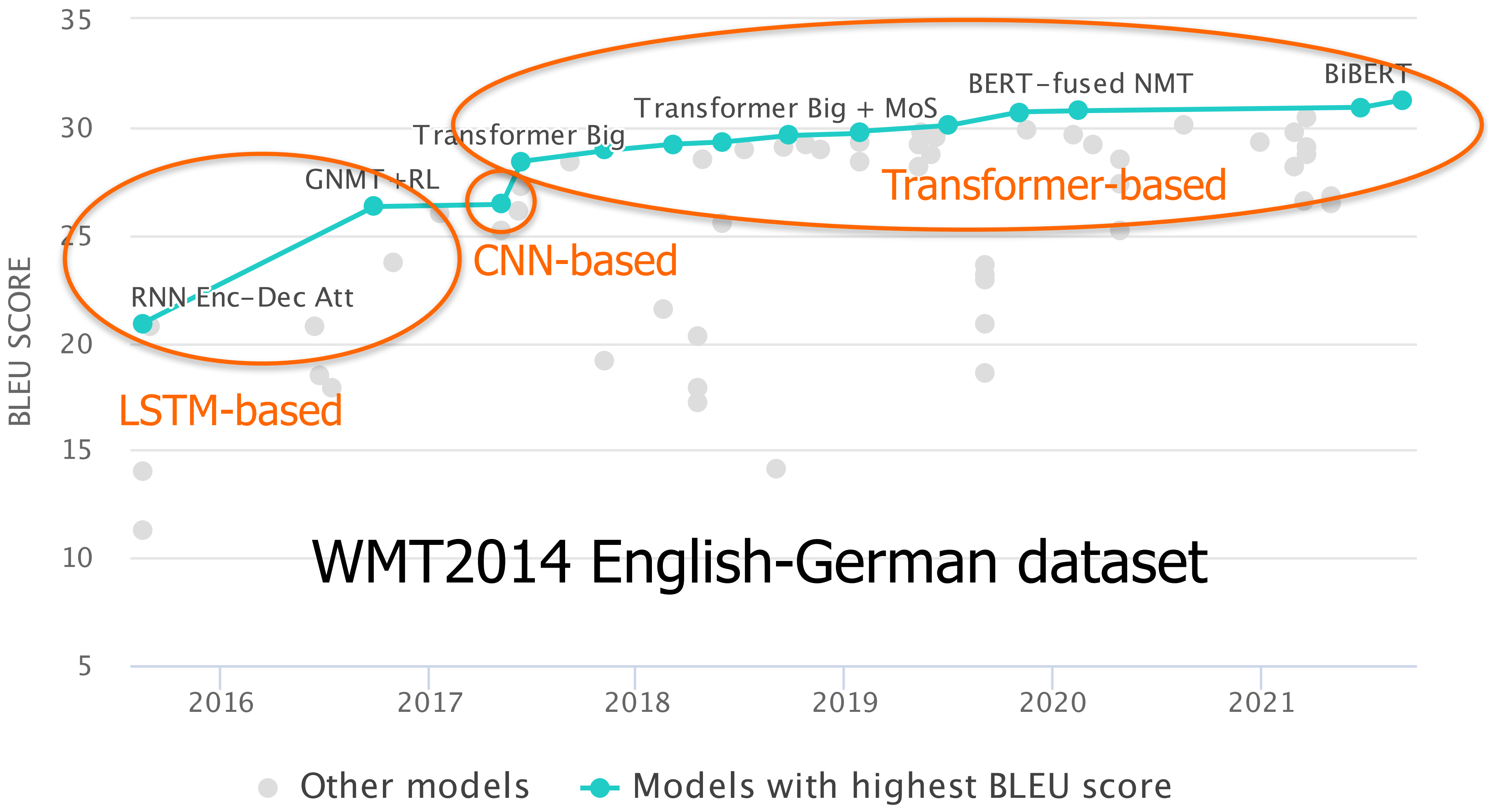
- Transformer encoder and decoder are based on self-attention.
- Decoder uses cross-attention.
- Masked self-attention in the decoder allows for parallel processing.

Questions?

Transformer Applications

- Background
- Transformers
- Transformer Applications
  - Machine Translation (vanilla Transformer)
  - Text Classification (BERT)
  - Text Generation (GPT, ChatGPT)
  - Image Classification (ViT)
- Multimodal Learning

Machine Translation (Transformer)



https://paperswithcode.com/sota/machine-translation-on-wmt2014-english-german

Text Classification (BERT)

- BERT (**Bidirectional Encoder Representations from Transformers**)

1. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” NAACL 2019.
2. https://huggingface.co/blog/bert-101

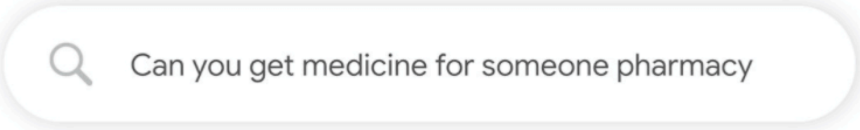
Text Classification (BERT)

- BERT (**Bidirectional Encoder Representations from Transformers**)
- BERT helps **Google provide better results** since November of 2020.

1. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," NAACL 2019.
2. https://huggingface.co/blog/bert-101

Text Classification (BERT)

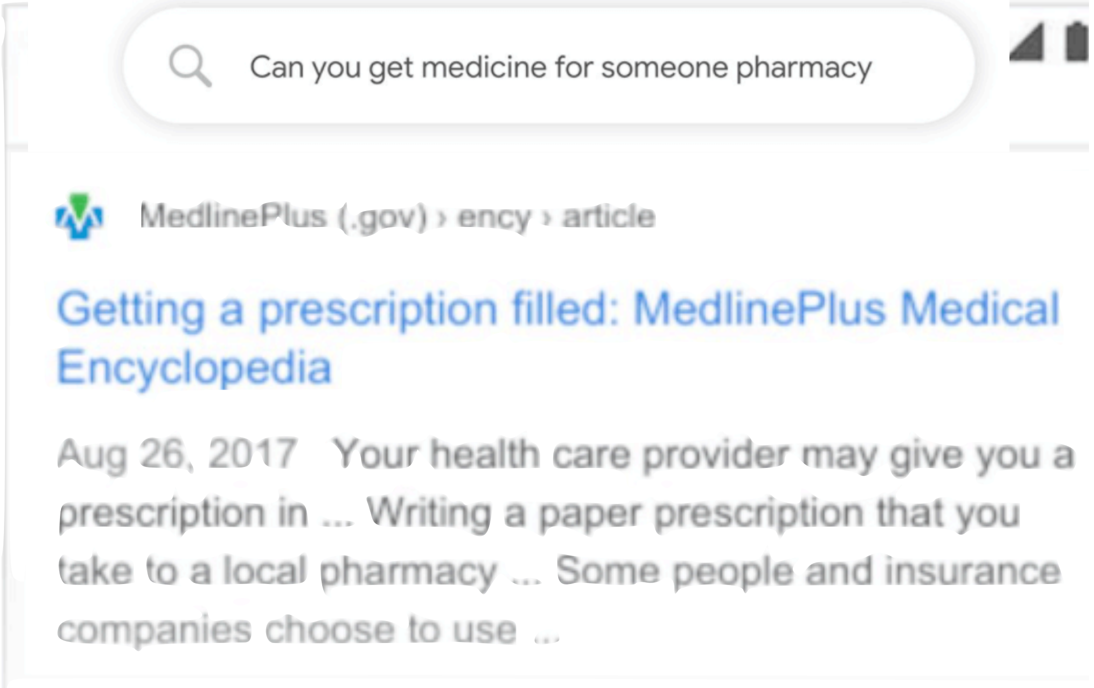
- BERT (**Bidirectional Encoder Representations from Transformers**)
- BERT helps **Google provide better results** since November of 2020.
- Example: “Can you get medicine for someone pharmacy”



1. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” NAACL 2019.
2. https://huggingface.co/blog/bert-101

Text Classification (BERT)

- BERT (**Bidirectional Encoder Representations from Transformers**)
- BERT helps **Google provide better results** since November of 2020.
- Example: "Can you get medicine for someone pharmacy"
  - Pre-BERT: (**Irrelevant**) Information about getting a prescription filled.

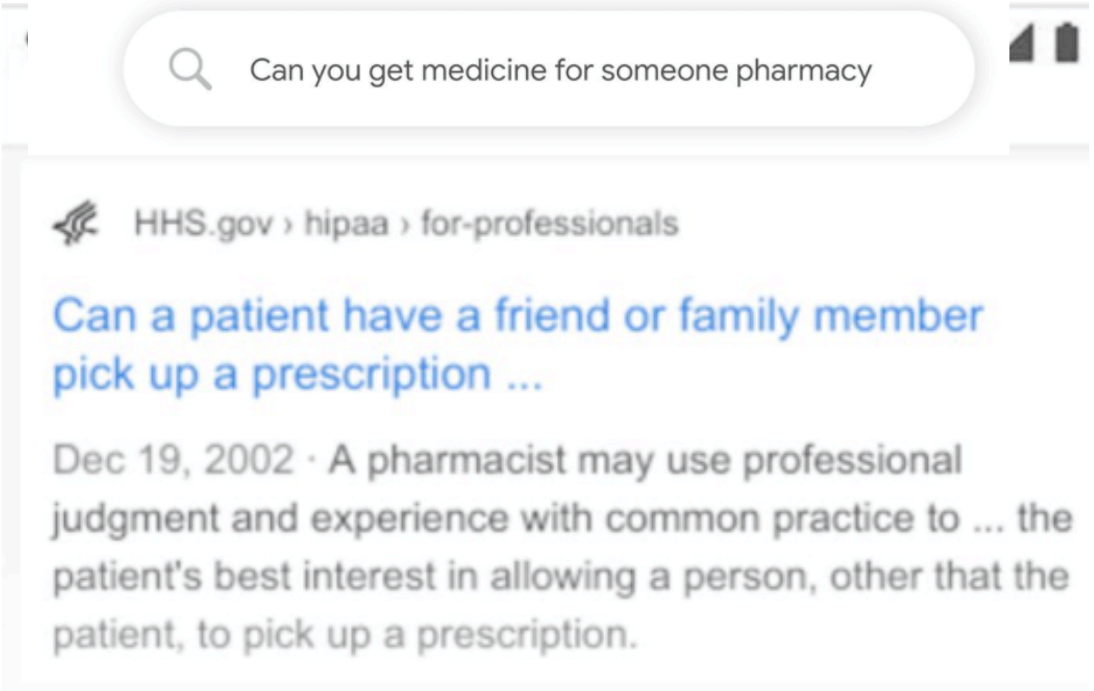


Before

1. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," NAACL 2019.
2. https://huggingface.co/blog/bert-101

Text Classification (BERT)

- BERT (**Bidirectional Encoder Representations from Transformers**)
- BERT helps **Google provide better results** since November of 2020.
- Example: “Can you get medicine for someone pharmacy”
  - Pre-BERT: (**Irrelevant**) Information about getting a prescription filled.
  - Post-BERT: Google **understands** that “for someone” relates to picking up a prescription for someone else.



After

1. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” NAACL 2019.
2. https://huggingface.co/blog/bert-101

Background: Pre-Training and Fine-Tuning

- Pre-Training
  - **Train** a model on a **large set of data**.
  - The model learns **good representations** of the input.

Background: Pre-Training and Fine-Tuning

- Pre-Training
  - **Train** a model on a **large set of data**.
  - The model learns **good representations** of the input.
- Fine-tuning
  - **Train** the model again on the **target task** with usually less data.
  - Modify the model if needed for the target task (e.g., add a different classification layer to the original model).

Background: Pre-Training and Fine-Tuning

- Pre-Training
  - **Train** a model on a **large set of data**.
  - The model learns **good representations** of the input.
- Fine-tuning
  - **Train** the model again on the **target task** with usually less data.
  - Modify the model if needed for the target task (e.g., add a different classification layer to the original model).

Example

- Train a image classification model (e.g., ResNet) on the ImageNet dataset.
- Fine-tune the model on robotic scenes.

Background: Pre-Training and Fine-Tuning

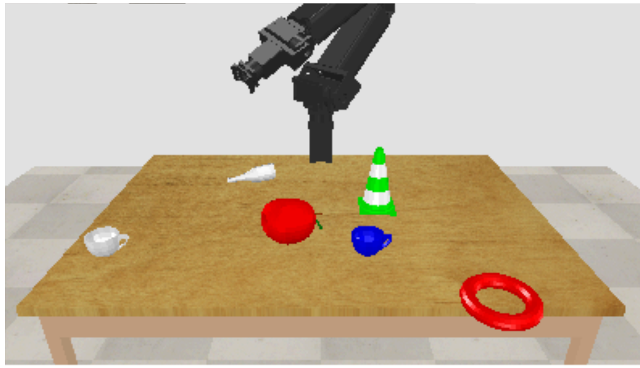
- Pre-Training
  - **Train** a model on a **large set of data**.
  - The model learns **good representations** of the input.
- Fine-tuning
  - **Train** the model again on the **target task** with usually less data.
  - Modify the model if needed for the target task (e.g., add a different classification layer to the original model).

Example

- Train a image classification model (e.g., ResNet) on the ImageNet dataset.
- Fine-tune the model on robotic scenes.

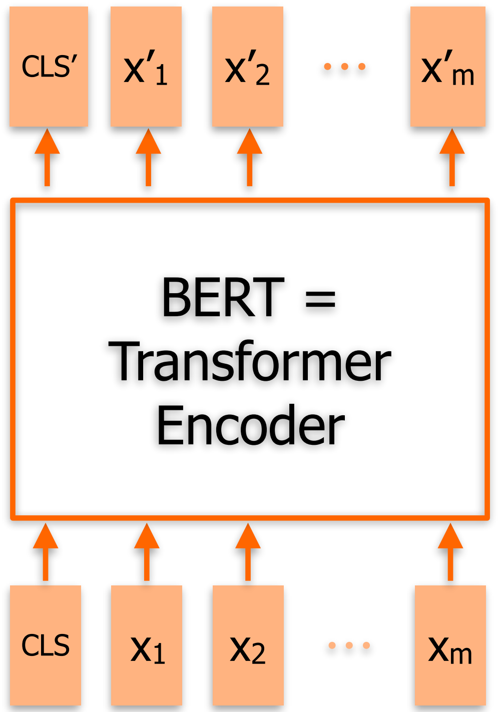
Imagenet

Target task



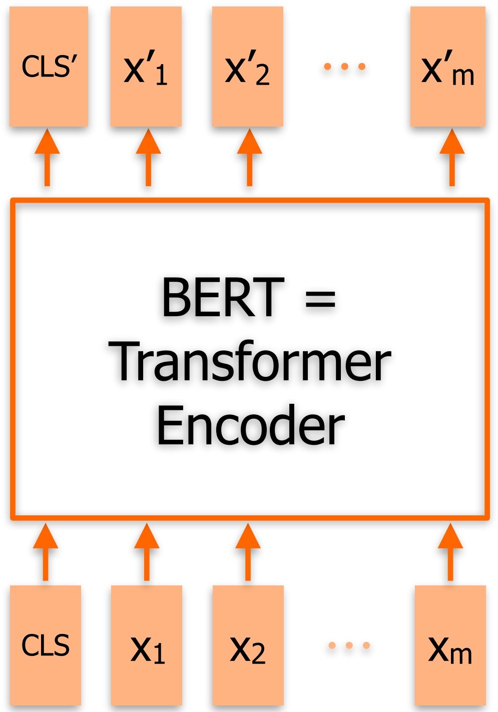
BERT: Architecture and Pre-training

- Transformer Encoder



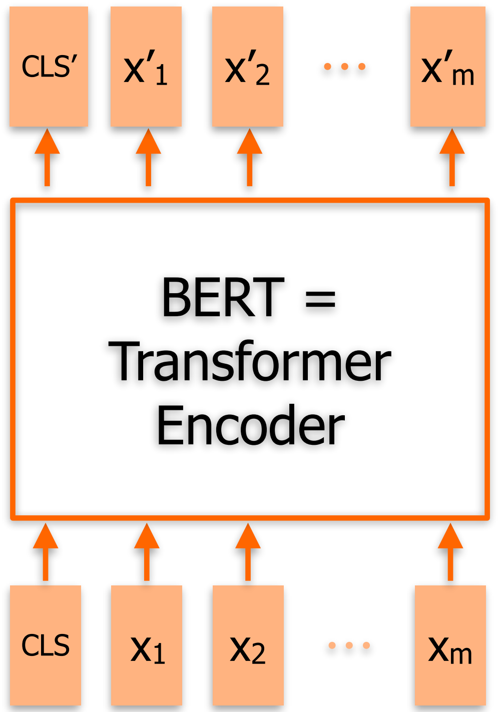
BERT: Architecture and Pre-training

- Transformer Encoder
- Pre-training dataset
  - BooksCorpus (800M words)
  - English Wikipedia (2,500M words)



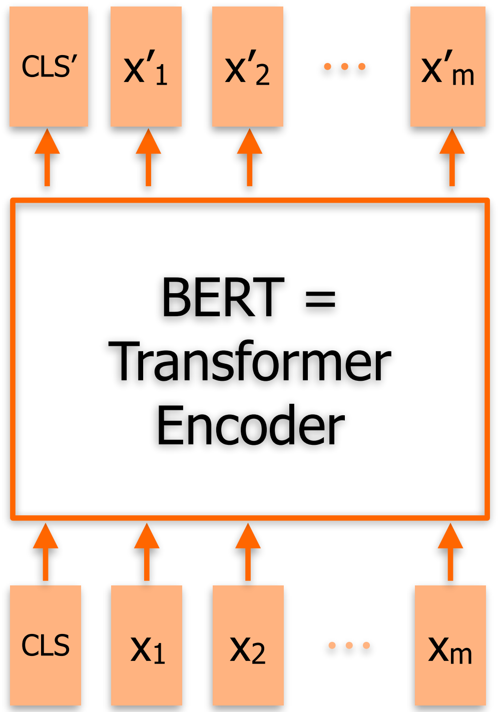
BERT: Architecture and Pre-training

- Transformer Encoder
- Pre-training dataset
  - BooksCorpus (800M words)
  - English Wikipedia (2,500M words)
- Pre-training task 1: **Masked Language Model**
  - Input: a sentence from a the dataset.
  - Mask some input tokens at random.
  - Predict those masked tokens.



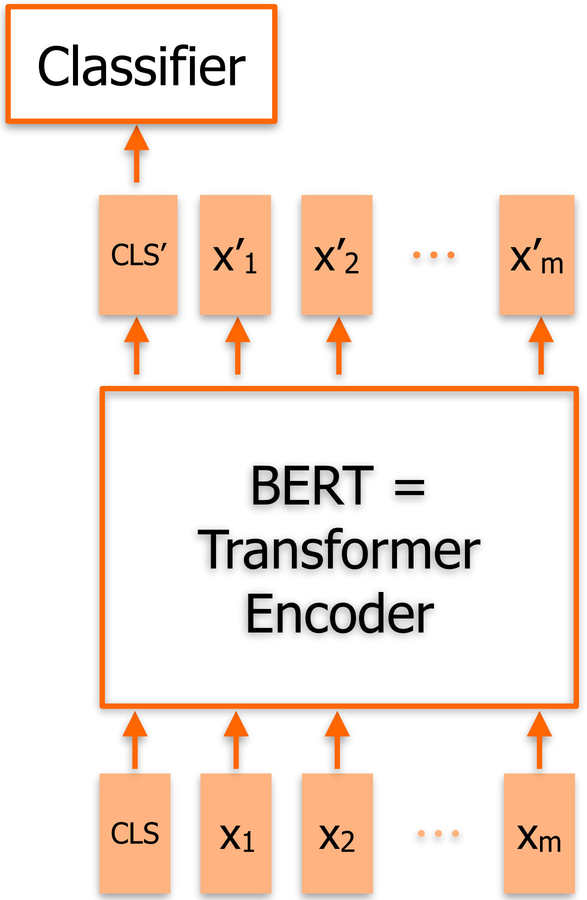
BERT: Architecture and Pre-training

- Transformer Encoder
- Pre-training dataset
  - BooksCorpus (800M words)
  - English Wikipedia (2,500M words)
- Pre-training task 1: **Masked Language Model**
  - Input: a sentence from a the dataset.
  - Mask some input tokens at random.
  - Predict those masked tokens.
- Pre-training task 2: **Next Sentence Prediction**
  - Input: concatenation of two sentences A and B.
  - 50% of the time B is A's next sentence.
  - 50% of the time B is a random sentence.
  - Predict whether B is the next sentence.



BERT: Fine-Tuning

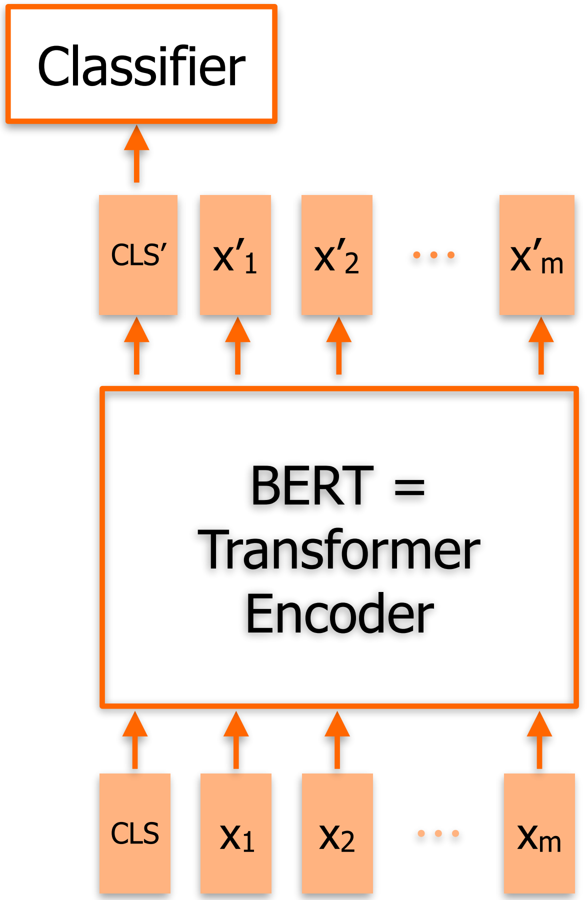
- Fine-Tuning: Use CLS for prediction.



https://huggingface.co/blog/bert-101

BERT: Fine-Tuning

- Fine-Tuning: Use CLS for prediction.
- It achieved state-of-the-art performance on three classification tasks:
  - SQuAD (Stanford Question Answering Dataset);
  - SWAG (Situations With Adversarial Generations);
  - GLUE (General Language Understanding Evaluation) a benchmark suit of nine tasks.



https://huggingface.co/blog/bert-101

Text Generation (GPT)

- GPT (Generative Pre-trained Transformer)

1. https://transformer.huggingface.co/doc/gpt2-large
2. https://blog.andrewcantino.com/blog/2021/04/21/prompt-engineering-tips-and-tricks/

Text Generation (GPT)

- GPT (Generative Pre-trained Transformer)
  - A **language model**
    - Look at part of a sentence and predict the next word.

1. https://transformer.huggingface.co/doc/gpt2-large
2. https://blog.andrewcantino.com/blog/2021/04/21/prompt-engineering-tips-and-tricks/

Text Generation (GPT)

- GPT (Generative Pre-trained Transformer)
  - A **language model**
    - Look at part of a sentence and predict the next word.

This is a lecture on Transformers and Crossmodal Learning. It covers several topics in detail, such as crossmodal information transfer, multilevel modeling, and object representation , and it is based on a recent paper " Theories of Cross mod al Learning " which has now been published as a working paper (pdf).

1. https://transformer.huggingface.co/doc/gpt2-large
2. https://blog.andrewcantino.com/blog/2021/04/21/prompt-engineering-tips-and-tricks/

Text Generation (GPT)

- GPT (Generative Pre-trained Transformer)
  - A **language model**
    - Look at part of a sentence and predict the next word.
- Based on the **Transformer decoder**.
  - Trained with next token prediction.

This is a lecture on Transformers and Crossmodal Learning. It covers several topics in detail, such as crossmodal information transfer, multilevel modeling, and object representation , and it is based on a recent paper " Theories of Cross mod al Learning " which has now been published as a working paper (pdf).

1. https://transformer.huggingface.co/doc/gpt2-large
2. https://blog.andrewcantino.com/blog/2021/04/21/prompt-engineering-tips-and-tricks/

Text Generation (GPT)

- GPT (Generative Pre-trained Transformer)
  - A **language model**
    - Look at part of a sentence and predict the next word.
- Based on the **Transformer decoder**.
  - Trained with next token prediction.
- Three versions
    - GPT (2018) 117 M parameters
    - GPT-2 (2019) 1.5 B parameters
    - GPT-3 (2020) 175 B parameters
    - GPT-3.5 (2022) 175 B parameters
    - GPT 4 (2023) ?

This is a lecture on Transformers and Crossmodal Learning. It covers several topics in detail, such as crossmodal information transfer, multilevel modeling, and object representation , and it is based on a recent paper " Theories of Cross mod al Learning " which has now been published as a working paper (pdf).

1. https://transformer.huggingface.co/doc/gpt2-large
2. https://blog.andrewcantino.com/blog/2021/04/21/prompt-engineering-tips-and-tricks/

Text Generation (GPT)

- GPT (Generative Pre-trained Transformer)
  - A **language model**
    - Look at part of a sentence and predict the next word.
- Based on the **Transformer decoder**.
  - Trained with next token prediction.
- Three versions
    - GPT (2018) 117 M parameters
    - GPT-2 (2019) 1.5 B parameters
    - GPT-3 (2020) 175 B parameters
    - GPT-3.5 (2022) 175 B parameters
    - GPT 4 (2023) ?
- **Prompt engineering** with GPT-3
  - The **description** of the task is **embedded in the input**.
  - The **output** is the **solution**.

This is a lecture on Transformers and Crossmodal Learning. It covers several topics in detail, such as crossmodal information transfer, multilevel modeling, and object representation , and it is based on a recent paper " Theories of Cross mod al Learning " which has now been published as a working paper (pdf).

1. https://transformer.huggingface.co/doc/gpt2-large
2. https://blog.andrewcantino.com/blog/2021/04/21/prompt-engineering-tips-and-tricks/

Text Generation (GPT)

- GPT (Generative Pre-trained Transformer)
  - A **language model**
    - Look at part of a sentence and predict the next word.
- Based on the **Transformer decoder**.
  - Trained with next token prediction.
- Three versions
  - GPT (2018) 117 M parameters
  - GPT-2 (2019) 1.5 B parameters
  - GPT-3 (2020) 175 B parameters
  - GPT-3.5 (2022) 175 B parameters
  - GPT 4 (2023) ?
- **Prompt engineering** with GPT-3
  - The **description** of the task is **embedded in the input**.
  - The **output** is the **solution**.

This is a lecture on Transformers and Crossmodal Learning. It covers several topics in detail, such as crossmodal information transfer, multilevel modeling, and object representation , and it is based on a recent paper " Theories of Cross mod al Learning " which has now been published as a working paper (pdf).

English: This sandwich is very tasty.
Spanish: E

1. https://transformer.huggingface.co/doc/gpt2-large
2. https://blog.andrewcantino.com/blog/2021/04/21/prompt-engineering-tips-and-tricks/

Text Generation (GPT)

- GPT (Generative Pre-trained Transformer)
  - A **language model**
    - Look at part of a sentence and predict the next word.
- Based on the **Transformer decoder**.
  - Trained with next token prediction.
- Three versions
    - GPT (2018) 117 M parameters
    - GPT-2 (2019) 1.5 B parameters
    - GPT-3 (2020) 175 B parameters
    - GPT-3.5 (2022) 175 B parameters
    - GPT 4 (2023) ?
- **Prompt engineering** with GPT-3
  - The **description** of the task is **embedded in the input**.
  - The **output** is the **solution**.

This is a lecture on Transformers and Crossmodal Learning. It covers several topics in detail, such as crossmodal information transfer, multilevel modeling, and object representation , and it is based on a recent paper " Theories of Cross mod al Learning " which has now been published as a working paper (pdf).

English: This sandwich is very tasty.
Spanish: Este sándwich es muy rico.

1. https://transformer.huggingface.co/doc/gpt2-large
2. https://blog.andrewcantino.com/blog/2021/04/21/prompt-engineering-tips-and-tricks/

Text Generation (GPT)

- GPT (Generative Pre-trained Transformer)
  - A **language model**
    - Look at part of a sentence and predict the next word.
- Based on the **Transformer decoder**.
  - Trained with next token prediction.
- Three versions
    - GPT (2018) 117 M parameters
    - GPT-2 (2019) 1.5 B parameters
    - GPT-3 (2020) 175 B parameters
    - GPT-3.5 (2022) 175 B parameters
    - GPT 4 (2023) ?
- **Prompt engineering** with GPT-3
  - The **description** of the task is **embedded in the input**.
  - The **output** is the **solution**.

This is a lecture on Transformers and Crossmodal Learning. It covers several topics in detail, such as crossmodal information transfer, multilevel modeling, and object representation , and it is based on a recent paper " Theories of Cross mod al Learning " which has now been published as a working paper (pdf).

English: This sandwich is very tasty.
Spanish: Este sándwich es muy rico.

This is a poem written by Robert Frost about the perils of machine learning.

1. https://transformer.huggingface.co/doc/gpt2-large
2. https://blog.andrewcantino.com/blog/2021/04/21/prompt-engineering-tips-and-tricks/

Text Generation (GPT)

- GPT (Generative Pre-trained Transformer)
  - A **language model**
    - Look at part of a sentence and predict the next word.
- Based on the **Transformer decoder**.
  - Trained with next token prediction.
- Three versions
    - GPT (2018) 117 M parameters
    - GPT-2 (2019) 1.5 B parameters
    - GPT-3 (2020) 175 B parameters
    - GPT-3.5 (2022) 175 B parameters
    - GPT 4 (2023) ?
- **Prompt engineering** with GPT-3
  - The **description** of the task is **embedded in the input**.
  - The **output** is the **solution**.

This is a lecture on Transformers and Crossmodal Learning. It covers several topics in detail, such as crossmodal information transfer, multilevel modeling, and object representation , and it is based on a recent paper " Theories of Cross mod al Learning " which has now been published as a working paper (pdf).

English: This sandwich is very tasty.
Spanish: Este sándwich es muy rico.

This is a poem written by Robert Frost about the perils of machine learning.
Alas! The machines are here.
They'll eat our brains and take our jobs,
They'll do our thinking for us,
And all that we'll be able to do
Is program them.
Alas! Here comes the Machine

1. https://transformer.huggingface.co/doc/gpt2-large
2. https://blog.andrewcantino.com/blog/2021/04/21/prompt-engineering-tips-and-tricks/

ChatGPT

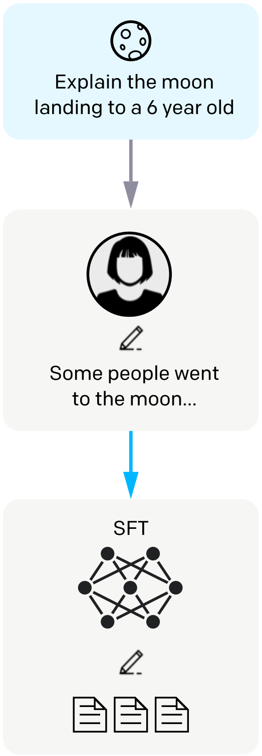
Step 1

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3 with supervised learning.



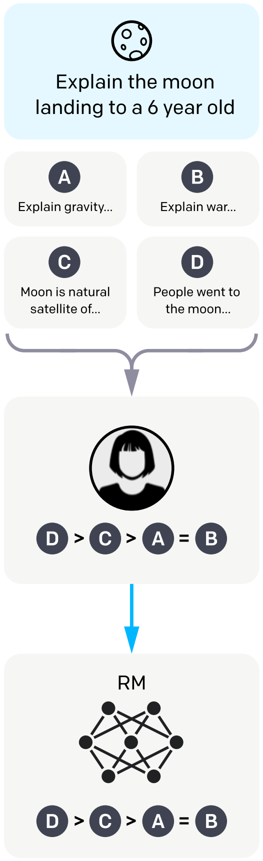
Step 2

Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.



Step 3

Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.

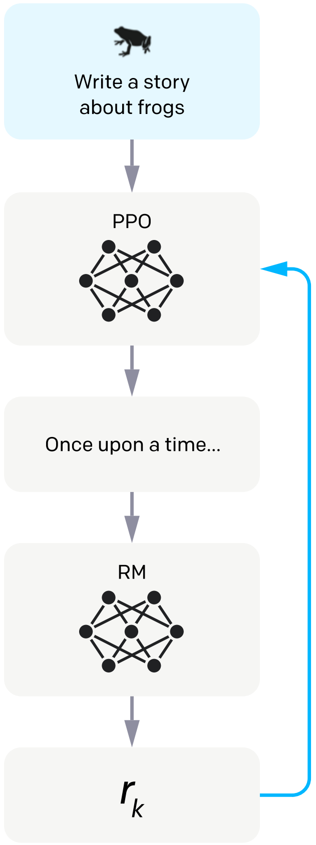
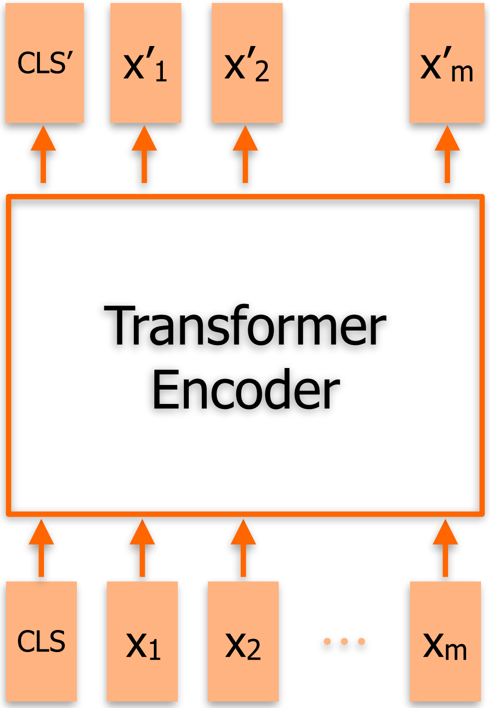


Image Classification (Vision Transformer)

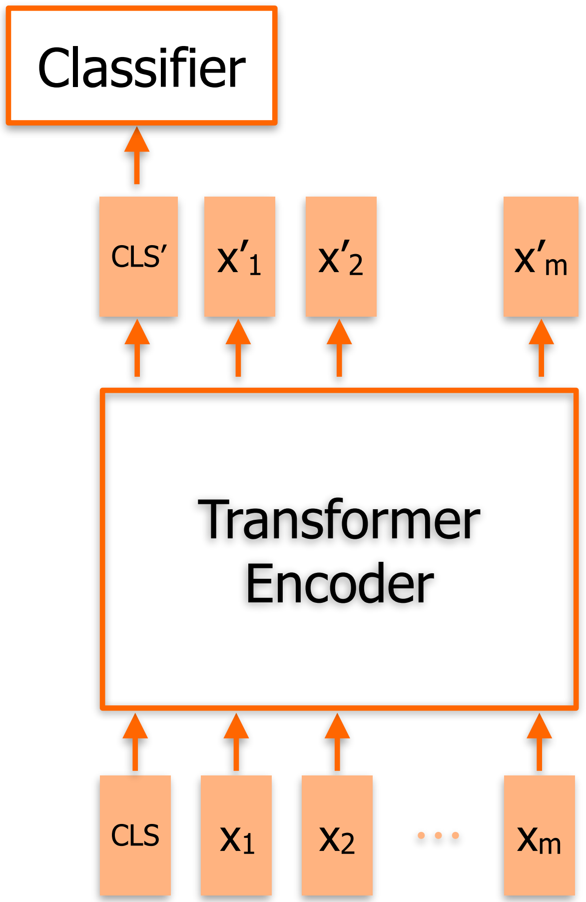
- Uses Transformer encoder.



 https://en.wikipedia.org/wiki/Vision_transformer#/media/File:Vision_Transformer.gif

Image Classification (Vision Transformer)

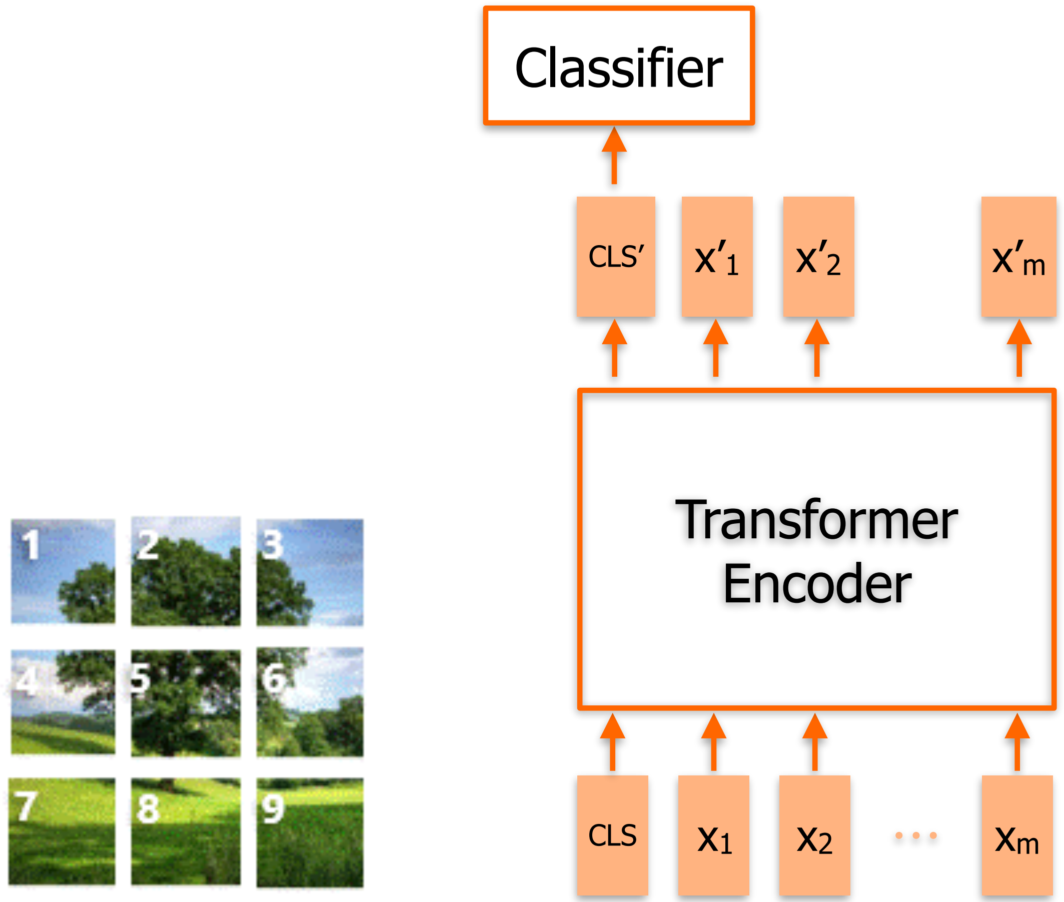
- Uses Transformer encoder.



https://en.wikipedia.org/wiki/Vision_transformer#/media/File:Vision_Transformer.gif

Image Classification (Vision Transformer)

- Uses Transformer encoder.
- Input image is tiled into sections.

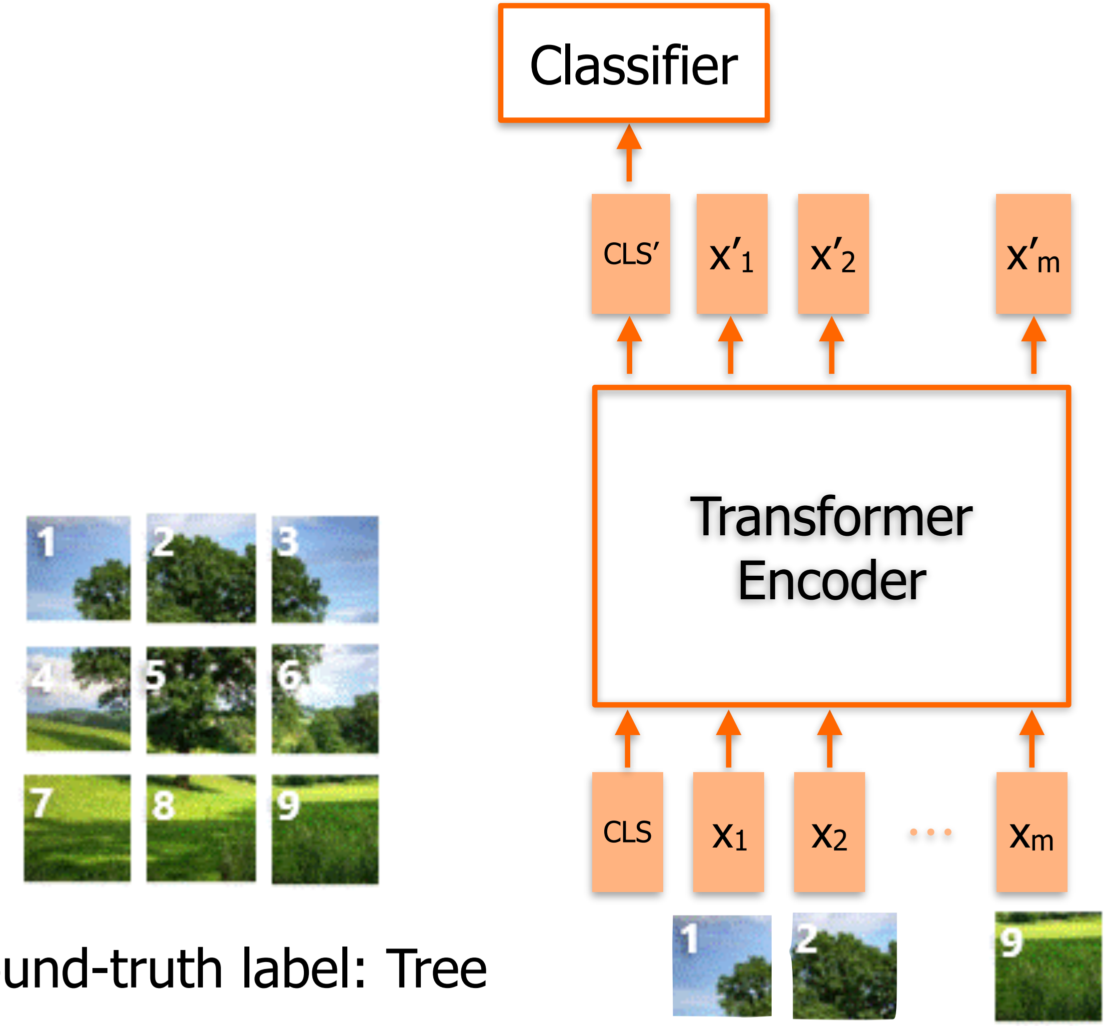


Ground-truth label: Tree

https://en.wikipedia.org/wiki/Vision_transformer#/media/File:Vision_Transformer.gif

Image Classification (Vision Transformer)

- Uses Transformer encoder.
- Input image is tiled into sections.
- The sections is turned into an embedding using a linear layer
- The results are fed to the Transformer encoder.

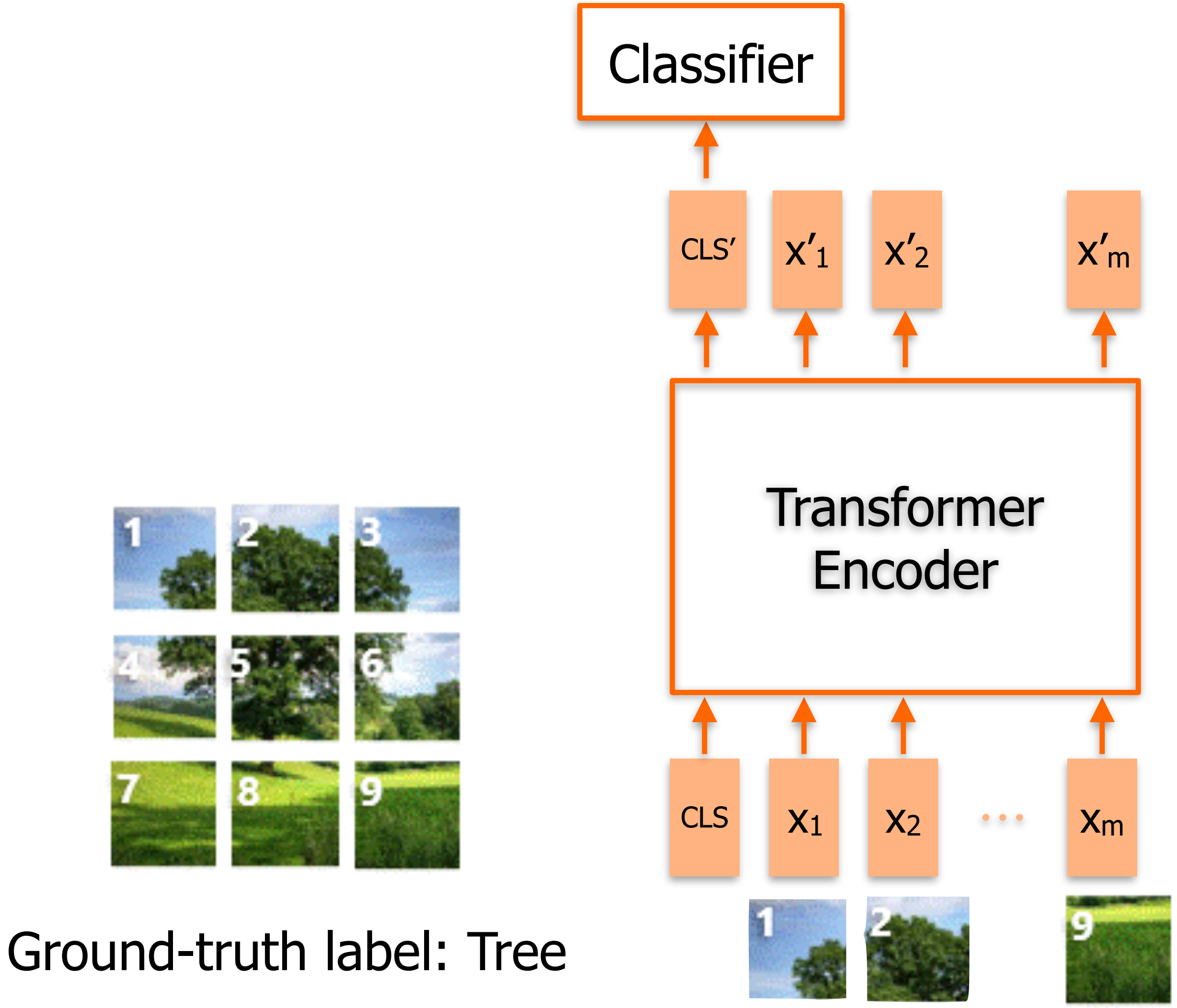


Ground-truth label: Tree

https://en.wikipedia.org/wiki/Vision_transformer#/media/File:Vision_Transformer.gif

Image Classification (Vision Transformer)

- Uses Transformer encoder.
- Input image is tiled into sections.
- The sections is turned into an embedding using a linear layer
- The results are fed to the Transformer encoder.
- Vision Transformers are able to capture global and wider range relations.
- However, more training data is needed.



https://en.wikipedia.org/wiki/Vision_transformer#/media/File:Vision_Transformer.gif

Summary

- The Transformer architecture has been used in different applications.
- BERT is based on the Transformer encoder.
- GPT is based on the Transformer decoder.
- Vision Transformer uses image patches as input tokens.

Multimodal Learning

- Background
- Transformers
- Transformer Applications
- Multimodal Learning
  - Introduction
  - Vision and Language Integration Methods

Multimodal Learning

- In DL language and vision have been tackled separately until 2014.
- Integrating two or more modalities has recently gained increased attention.
  - language, vision, speech, sound, proprioception, …
- Some Multimodal (vision and language) tasks:
  - Image Captioning
  - Visual Question Answering
  - Image Retrieval
  - Language-to-Image Generation

Image Captioning

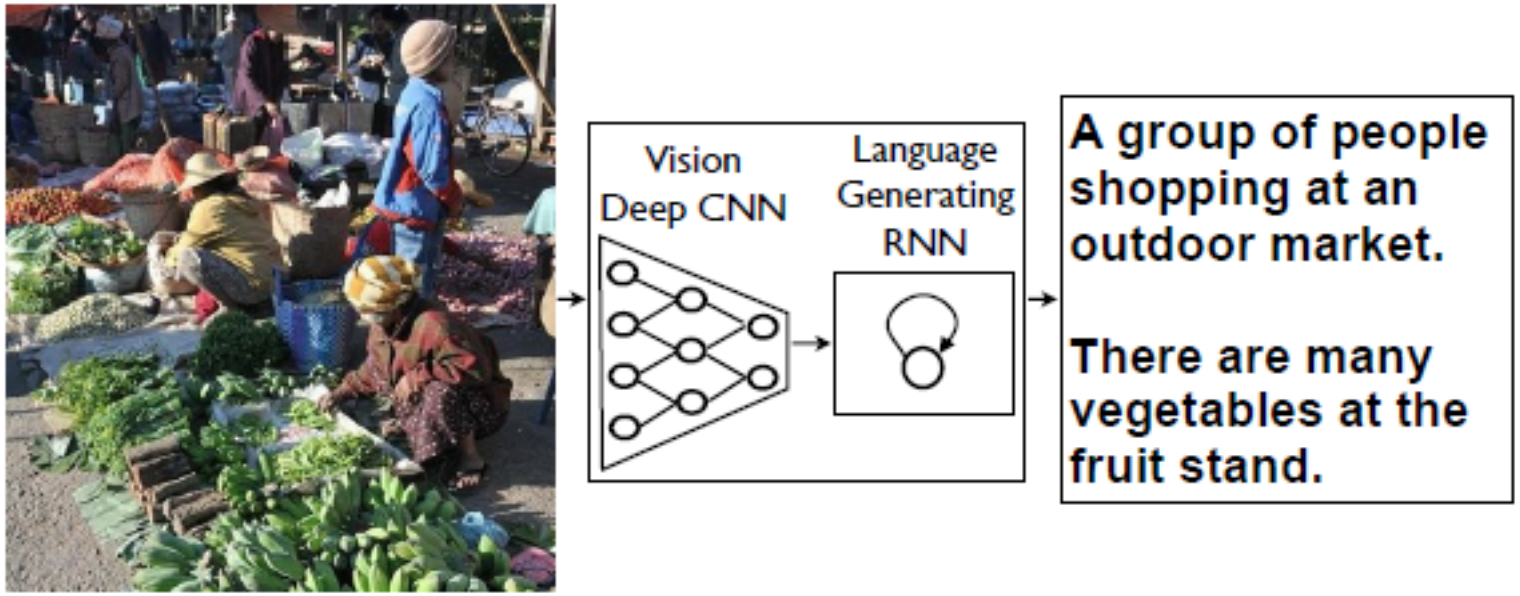
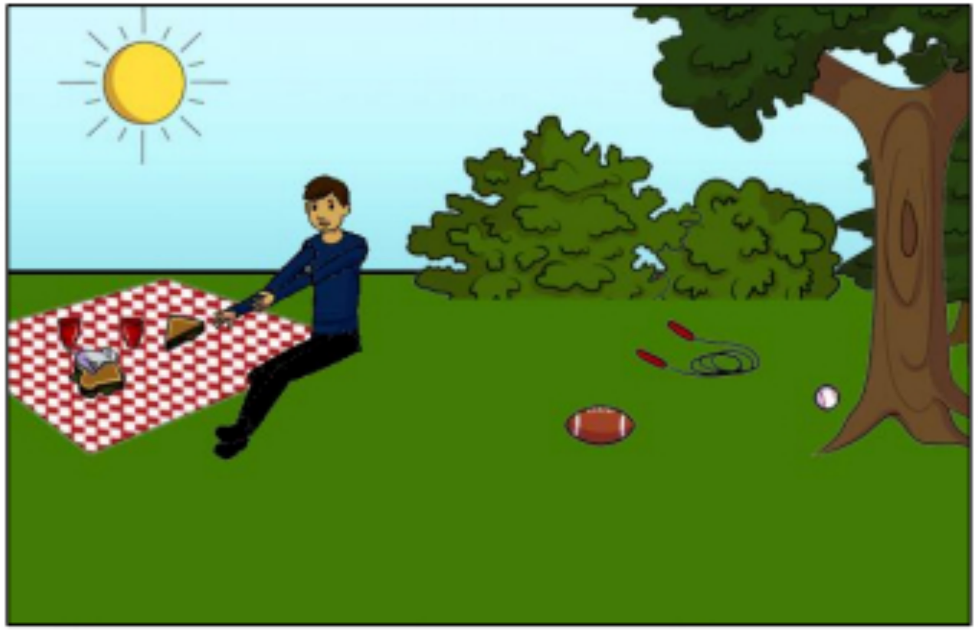


Image Captioning

O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and Tell: A Neural Image Caption Generator," CVPR 2015

Visual Question Answering



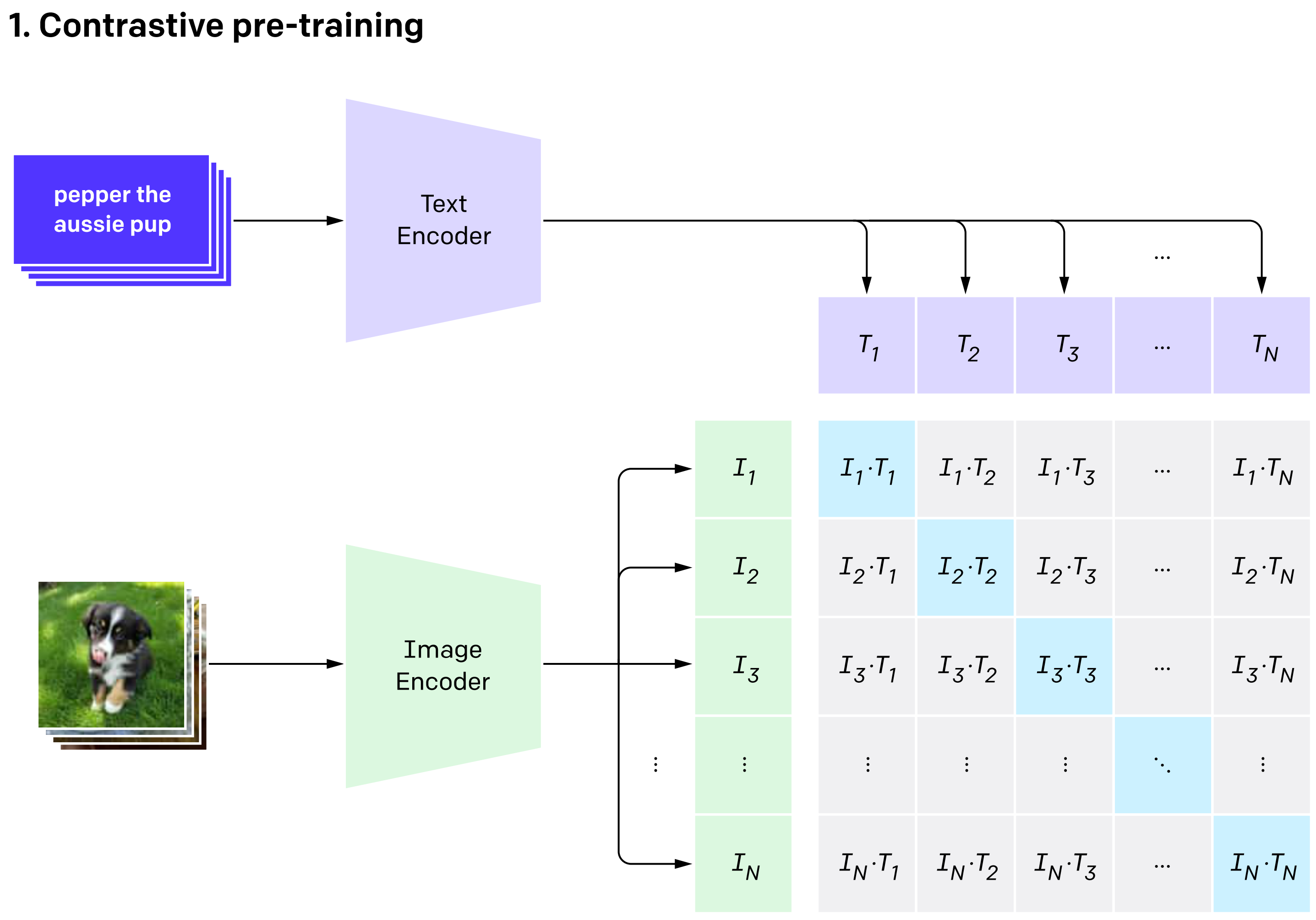
Is this person expecting company?
What is just under the tree?

Does it appear to be rainy?
Does this person have 20/20 vision?

Visual Question Answering

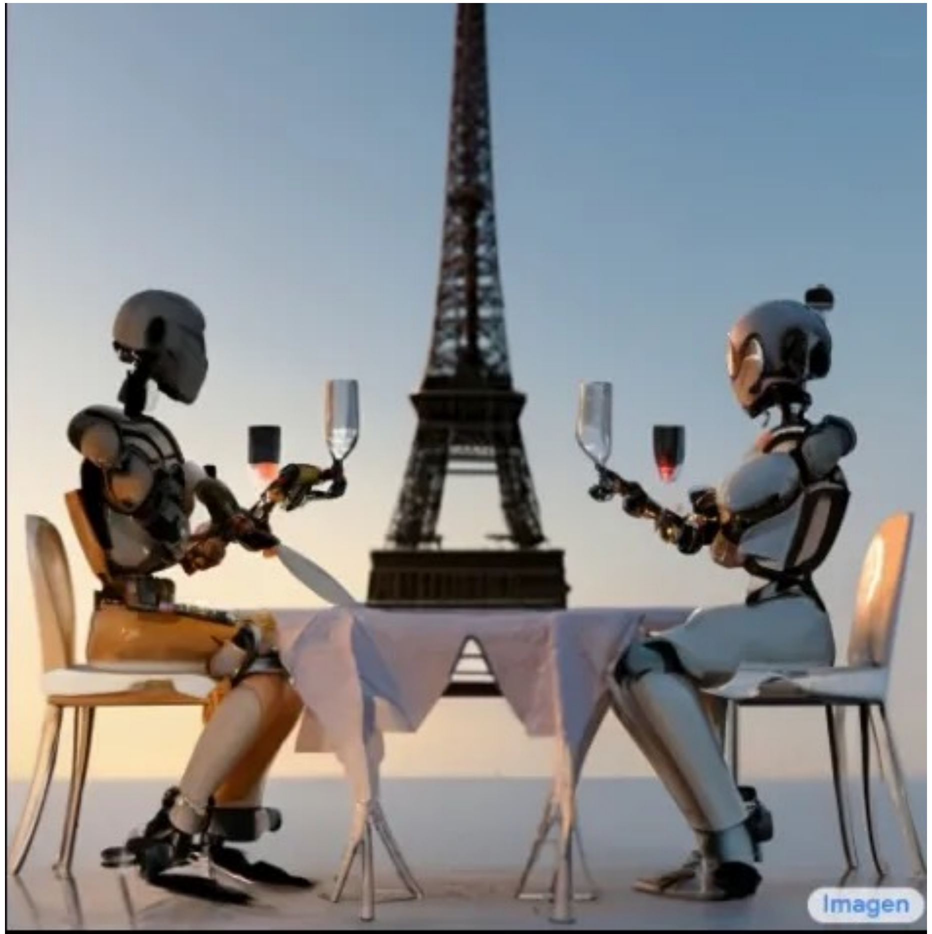
S. Antol et al., "VQA: Visual Question Answering," ICCV 2015

Image Retrieval



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

Language-to-Image Generation

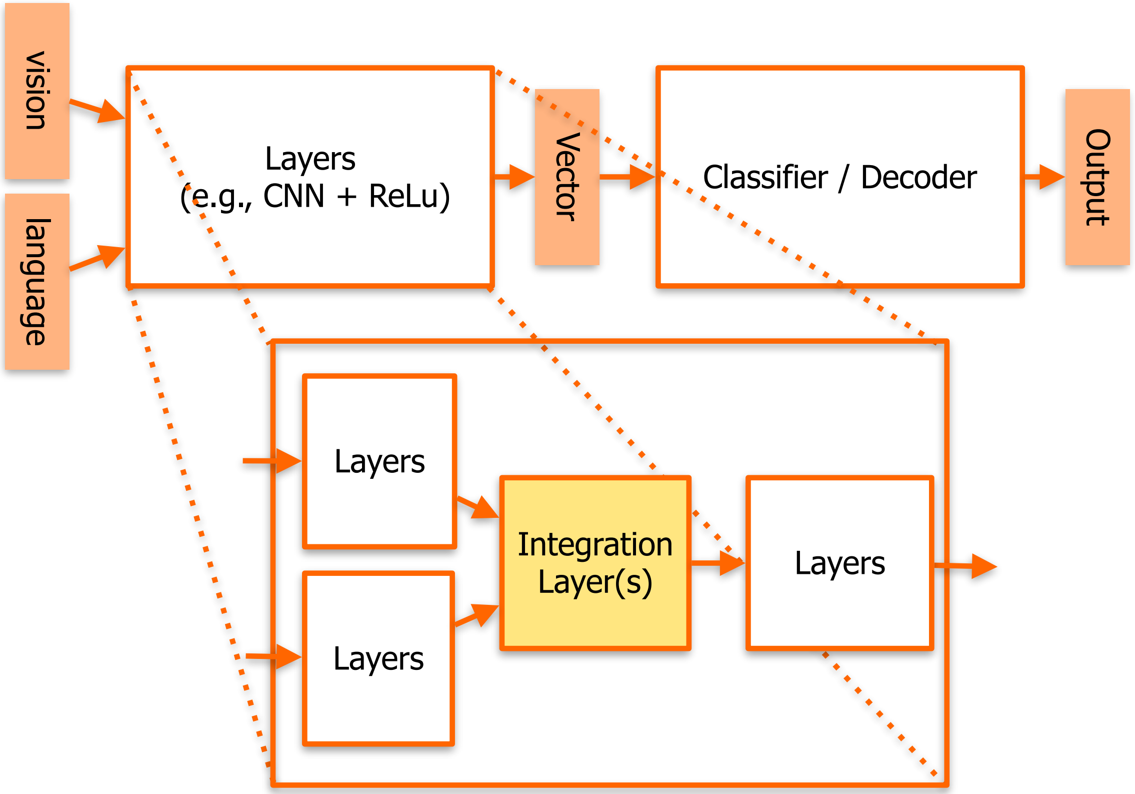


A robot couple fine dining with Eiffel Tower in the background.

An alien octopus floats through a portal reading a newspaper.

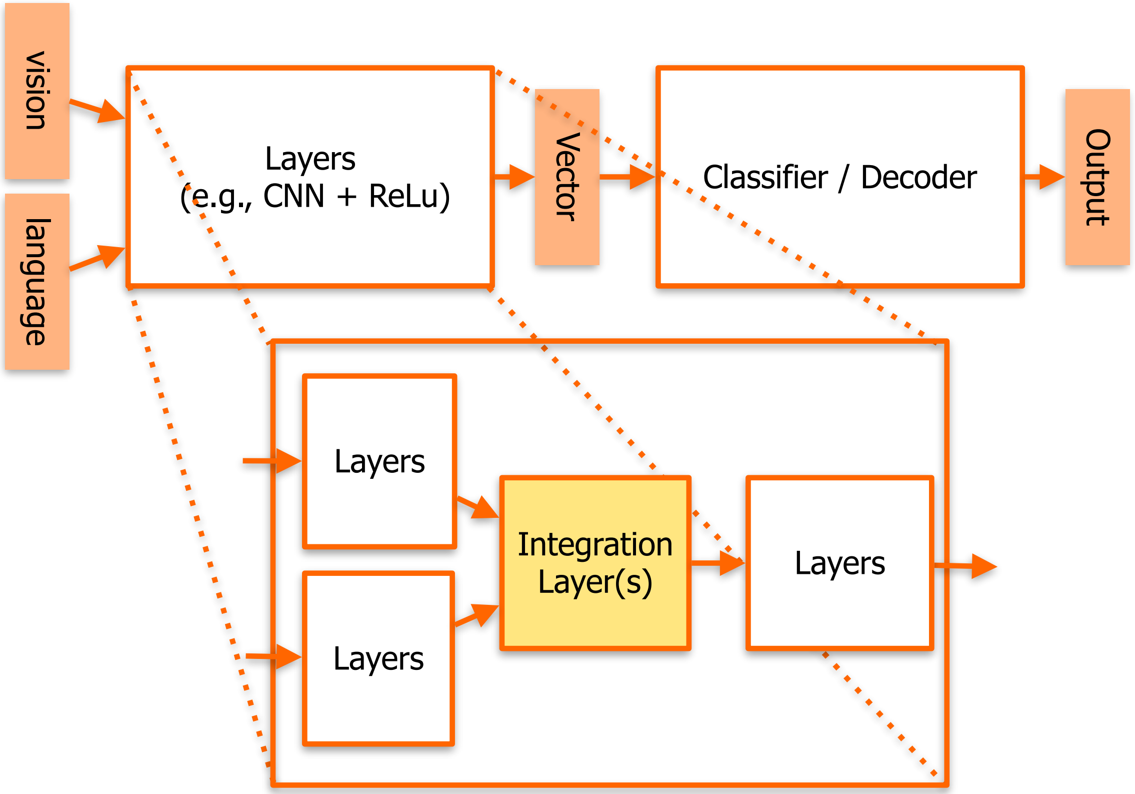
https://imagen.research.google/

Vision Language Integration Techniques



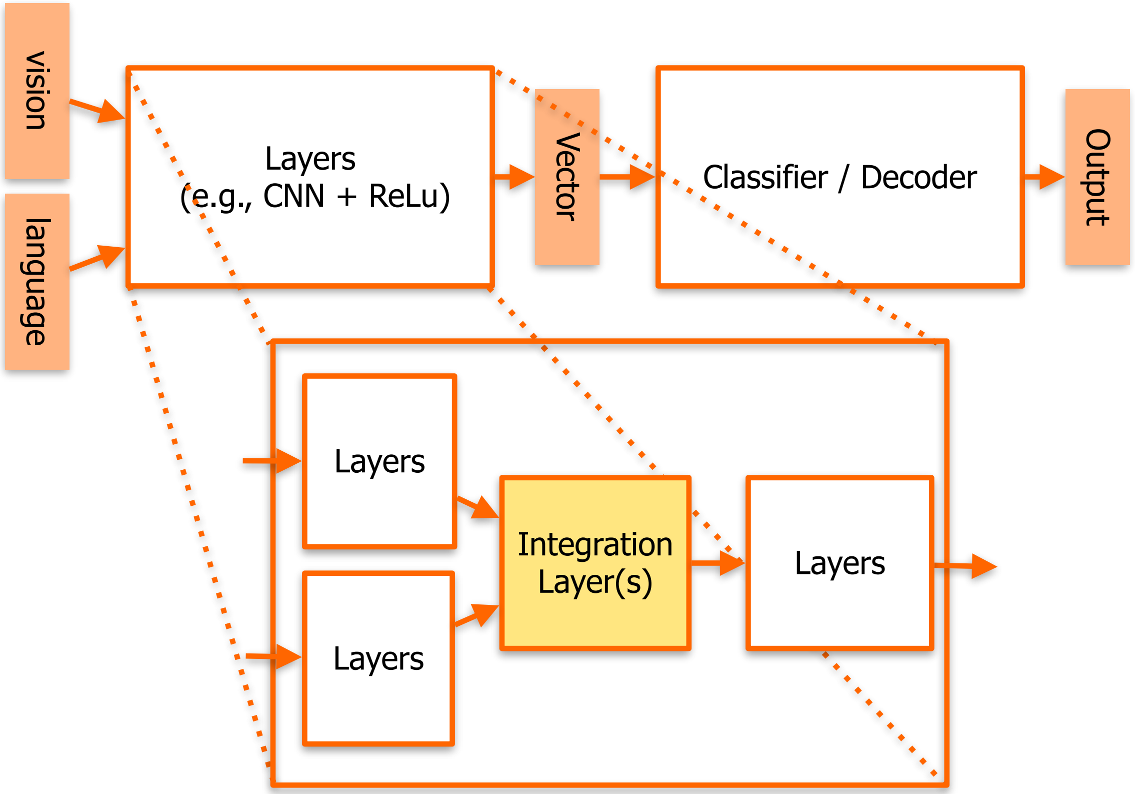
Vision Language Integration Techniques

- Q: Given two vectors from two different modalities (e.g., vision and language) how would you integrate them?



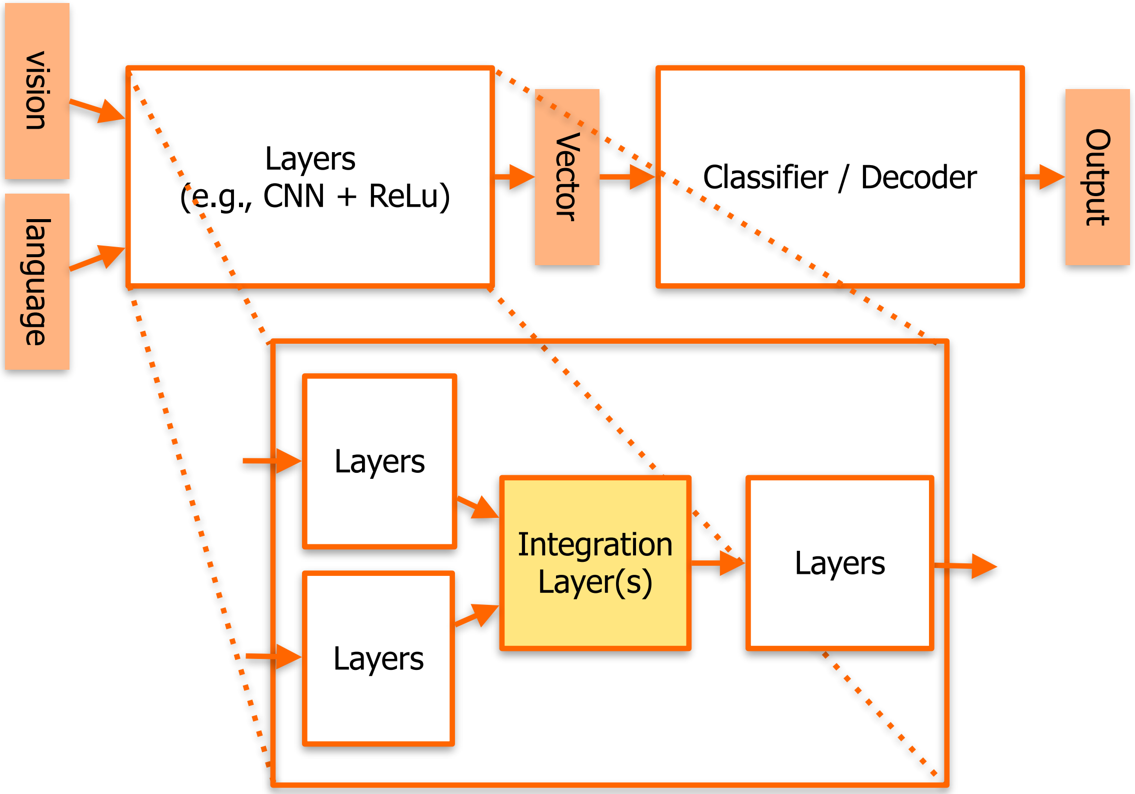
Vision Language Integration Techniques

- Q: Given two vectors from two different modalities (e.g., vision and language) how would you integrate them?
- Concatenation
  - $f(v, l) = [v; l]$



Vision Language Integration Techniques

- Q: Given two vectors from two different modalities (e.g., vision and language) how would you integrate them?
- Concatenation
  - $f(v, l) = [v; l]$
- Element-wise Multiplication
  - $f(v, l) = v \odot l$
  - Example: $\begin{bmatrix} 1 \\ 2 \end{bmatrix} \odot \begin{bmatrix} 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 2 \\ 6 \end{bmatrix}$



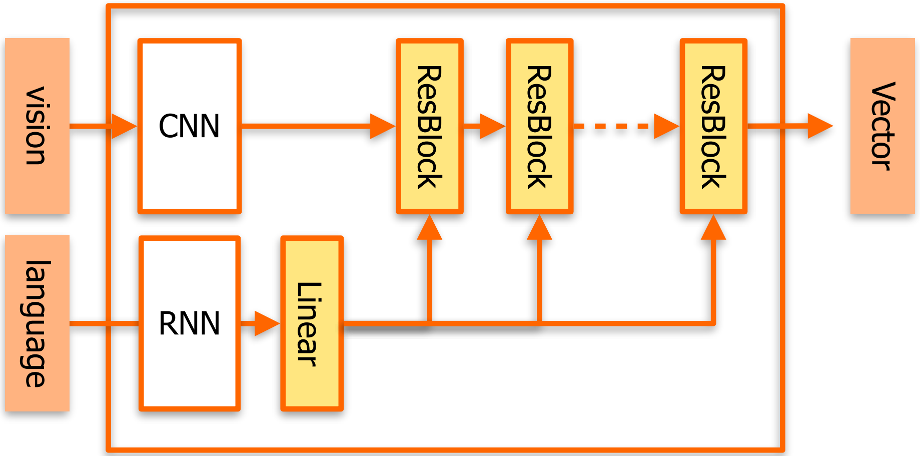
Feature-Wise Transformation

- Feature-Wise Transformation
  - The language input “modulates” how the image input is processed.
  - $f(v, l) = (\alpha_l \odot v) + \beta_l$
  - α_l and β_l are vectors computed from language vector l (e.g., using a linear layer)

1. https://distill.pub/2018/feature-wise-transformations/
2. E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. Courville, “FiLM: Visual Reasoning with a General Conditioning Layer,” AAAI 2018

Feature-Wise Transformation

- Feature-Wise Transformation
  - The language input “modulates” how the image input is processed.
  - $f(v, l) = (\alpha_l \odot v) + \beta_l$
  - α_l and β_l are vectors computed from language vector l (e.g., using a linear layer)



Example: FiLM architecture

1. https://distill.pub/2018/feature-wise-transformations/
2. E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. Courville, “FiLM: Visual Reasoning with a General Conditioning Layer,” AAAI 2018

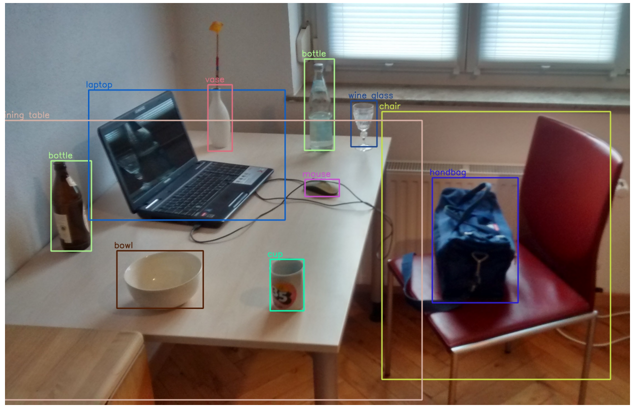
Transformer-Based Models
1. VQA

- Q: How would you integrate vision and language using Transformers?

1. H. Tan and M. Bansal, “LXMERT: Learning Cross-Modality Encoder Representations from Transformers,” EMNLP 2019
2. J. Lu, D. Batra, D. Parikh, and S. Lee, “ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks,” NeurIPS 2019

Transformer-Based Models
1. VQA

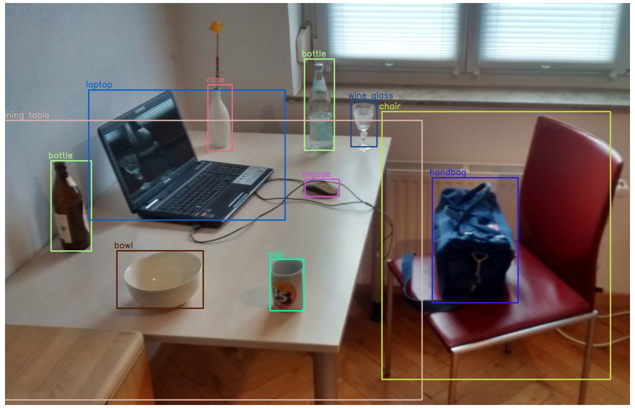
- Q: How would you integrate vision and language using Transformers?
- v_i: detected bounding boxes in image.



1. H. Tan and M. Bansal, "LXMERT: Learning Cross-Modality Encoder Representations from Transformers," EMNLP 2019
2. J. Lu, D. Batra, D. Parikh, and S. Lee, "ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks," NeurIPS 2019

Transformer-Based Models
1. VQA

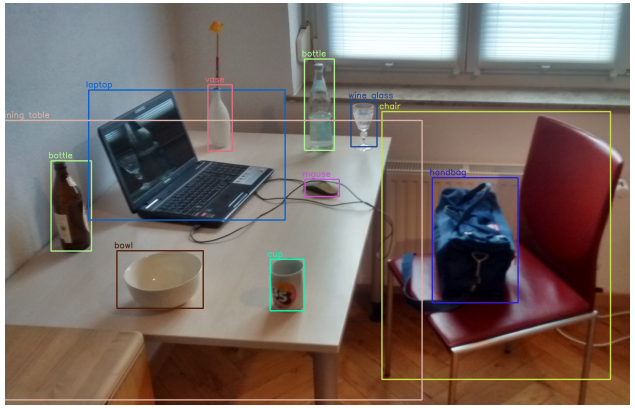
- Q: How would you integrate vision and language using Transformers?
- v_i: detected bounding boxes in image.
- l_i: language tokens (e.g., words).
  - e.g., "What is in front of the laptop?"

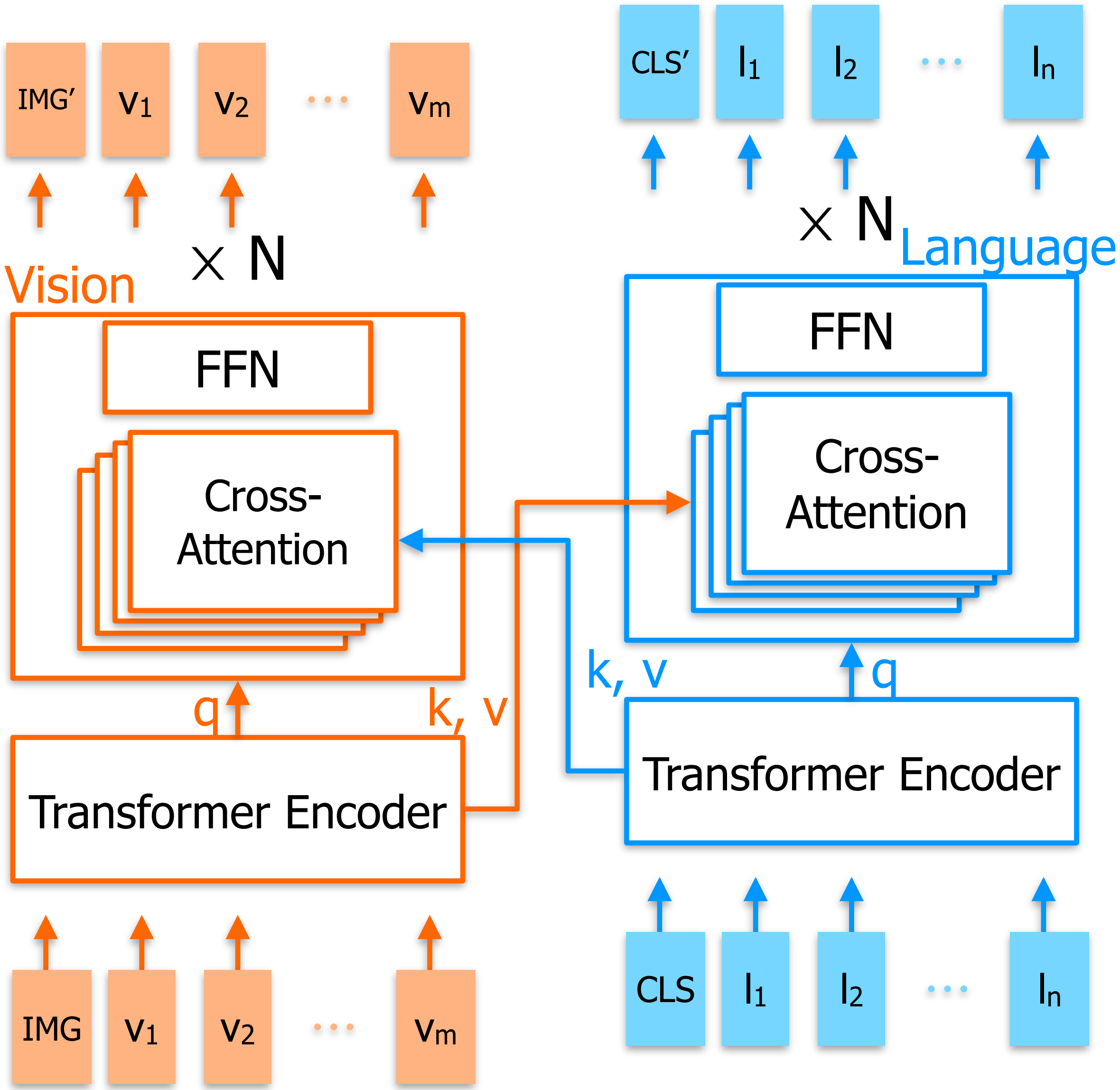


1. H. Tan and M. Bansal, "LXMERT: Learning Cross-Modality Encoder Representations from Transformers," EMNLP 2019
2. J. Lu, D. Batra, D. Parikh, and S. Lee, "ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks," NeurIPS 2019

Transformer-Based Models
1. VQA

- Q: How would you integrate vision and language using Transformers?
- v_i: detected bounding boxes in image.
- l_i: language tokens (e.g., words).
  - e.g., “What is in front of the laptop?”
- IMG and CLS are used for prediction

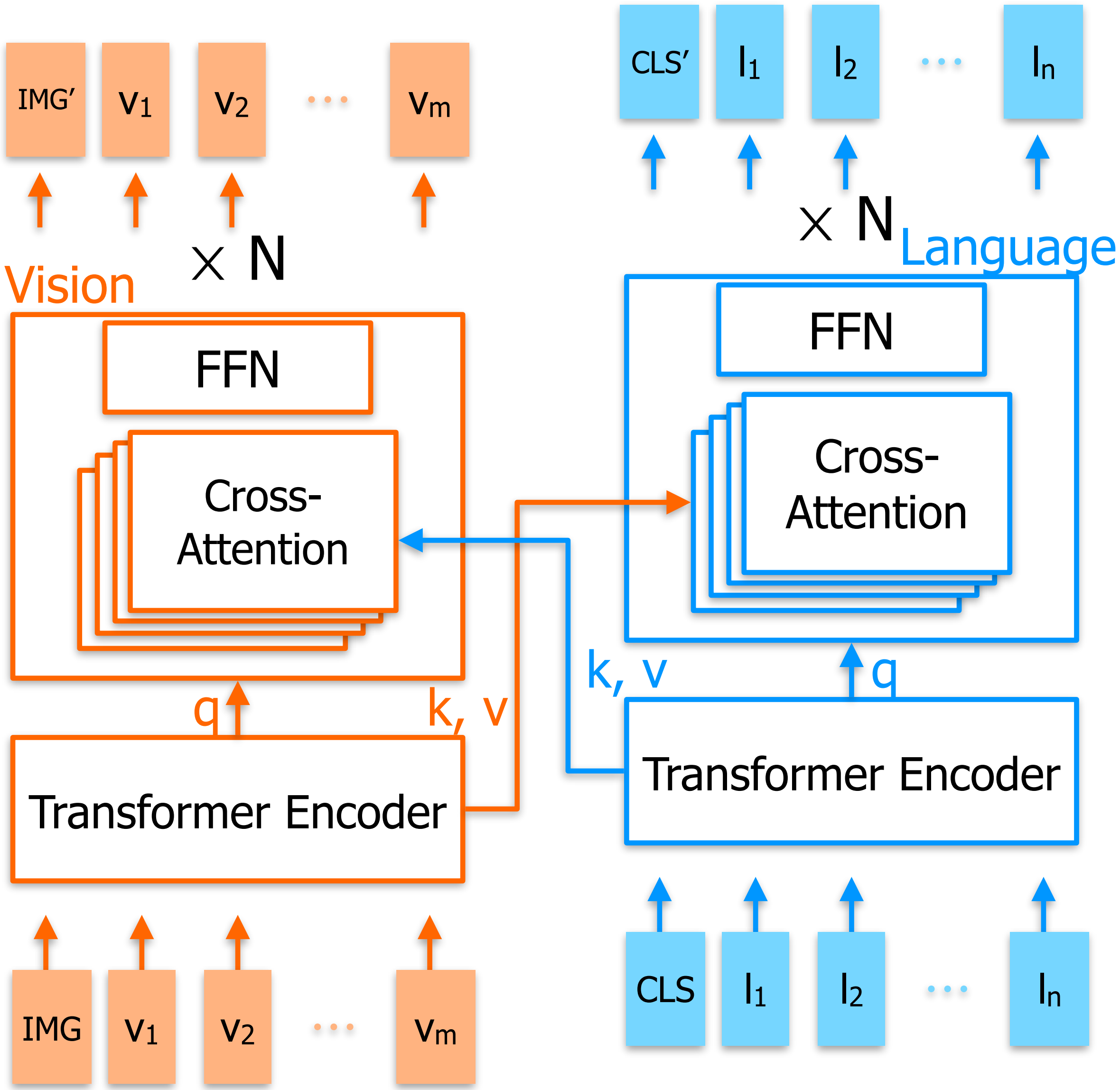




1. H. Tan and M. Bansal, “LXMERT: Learning Cross-Modality Encoder Representations from Transformers,” EMNLP 2019
2. J. Lu, D. Batra, D. Parikh, and S. Lee, “ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks,” NeurIPS 2019

Transformer-Based Models
1. VQA

- Q: How would you integrate vision and language using Transformers?
- v_i: detected bounding boxes in image.
- l_i: language tokens (e.g., words).
  - e.g., "What is in front of the laptop?"
- IMG and CLS are used for prediction
- Allows better representation of relationships between objects and words.

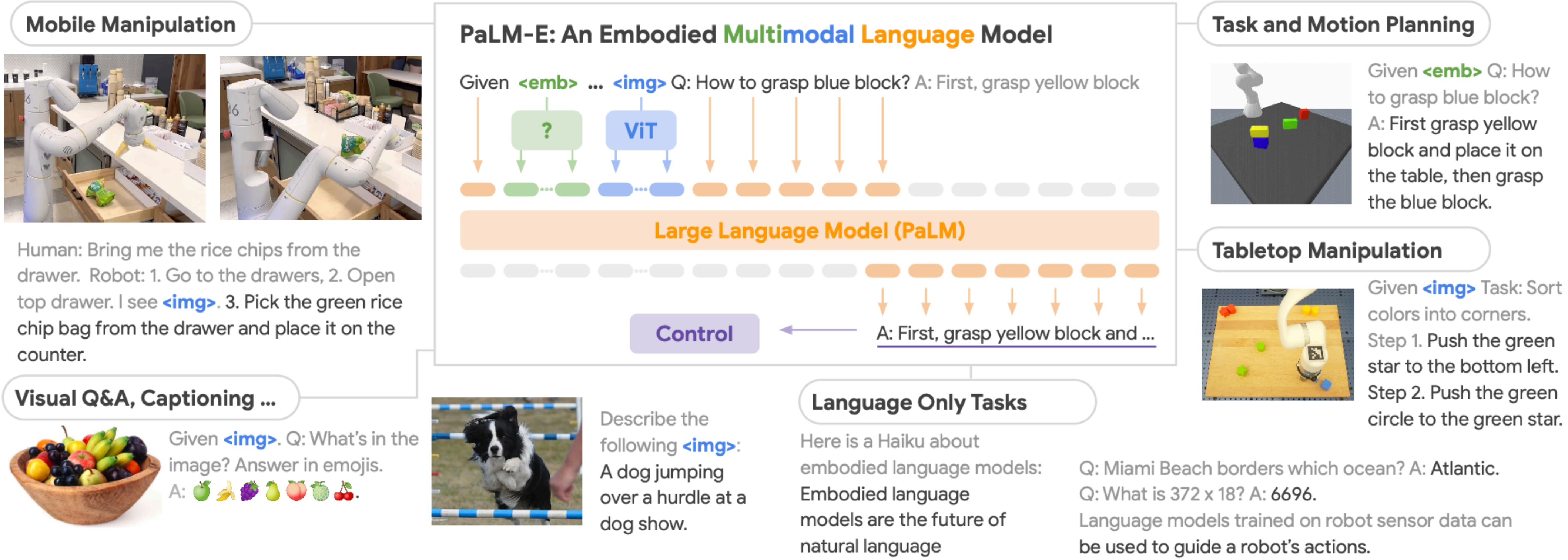


1. H. Tan and M. Bansal, "LXMERT: Learning Cross-Modality Encoder Representations from Transformers," EMNLP 2019
2. J. Lu, D. Batra, D. Parikh, and S. Lee, "ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks," NeurIPS 2019

Advantages of Transformer for Multimodal Learning

- RNNs and CNNs are constrained by the input space (i.e., 1D, 2D), where the order of the input matters.
- Transformer operates on sets: input order does not matter.
  - Adding new modalities is easier.
- How would you combine vision and language using a transformer?
  - Add new modalities and introduce modality-specific embeddings / flags.

Transformer-Based Models
2. PaLM-E (Pathways Language Model with Embodiment)



D. Driess et al., "PaLM-E: An Embodied Multimodal Language Model." http://arxiv.org/abs/2303.03378
https://palm-e.github.io/

Summary

- Multimodal learning is an active research area.
- There are several ways to integrate different modalities.
- Transformer cross-attention can be used to integrated different modalities.

Open Questions in Deep Learning Research

- Generalizability
  - How to make models generalize to new situations?
- Continual learning
  - How can the models learn new data without forgetting previous ones?
- Explainability
  - How do the models come to the decisions?
- Ethical Issues
  - How can the models be aligned with human values?

Questions?

Resources

- Transformer
  - The Illustrated Transformer
  - Ch 13 Transformers in “Deep Learning: Foundations and Concepts”
  - Formal Algorithms for Transformers
  - Dive into Deep Learning - Chapter 11: Attention Mechanisms and Transformers
- Vision and Language Integration
  - A. Mogadala, M. Kalimuthu, and D. Klakow, “Trends in Integration of Vision and Language Research: A Survey of Tasks, Datasets, and Methods,” JAIR 2021