CS 540 Introduction to Artificial Intelligence
RNNs, Attention, and Transformers

University of Wisconsin–Madison
Fall 2025, Section 3
October 29, 2025

Announcements

- HW 6 online:
  - Pytorch and neural networks
  - Deadline **Friday, 10/31 11:59PM**
- HW7 released Friday
  - CNNs and transformers

Neural Networks: Perceptron
Neural Networks: MLP
Deep Learning: CNNs
Deep Learning: ResNets
Deep Learning: RNNs and Transformers
Neural Networks & Deep Learning Review
Search, Games, and Reinforcement Learning

Outline for Today's Lecture

- Recurrent Neural Networks
- RNNs for Language Modeling
- The Attention Mechanism
- Transformers

Recurrent Neural Networks (RNNs)

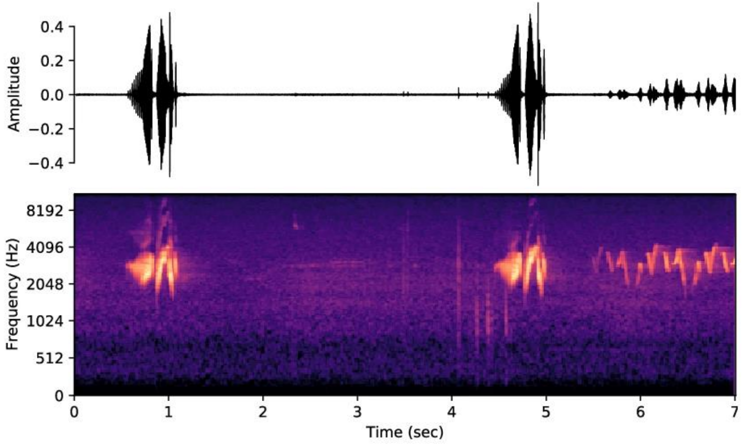
Recurrent Neural Networks (RNNs)

Why Recurrent Neural Networks?

- Handle sequential data (text, audio, speech, time series)
- Allow more general computations

The quick brown fox

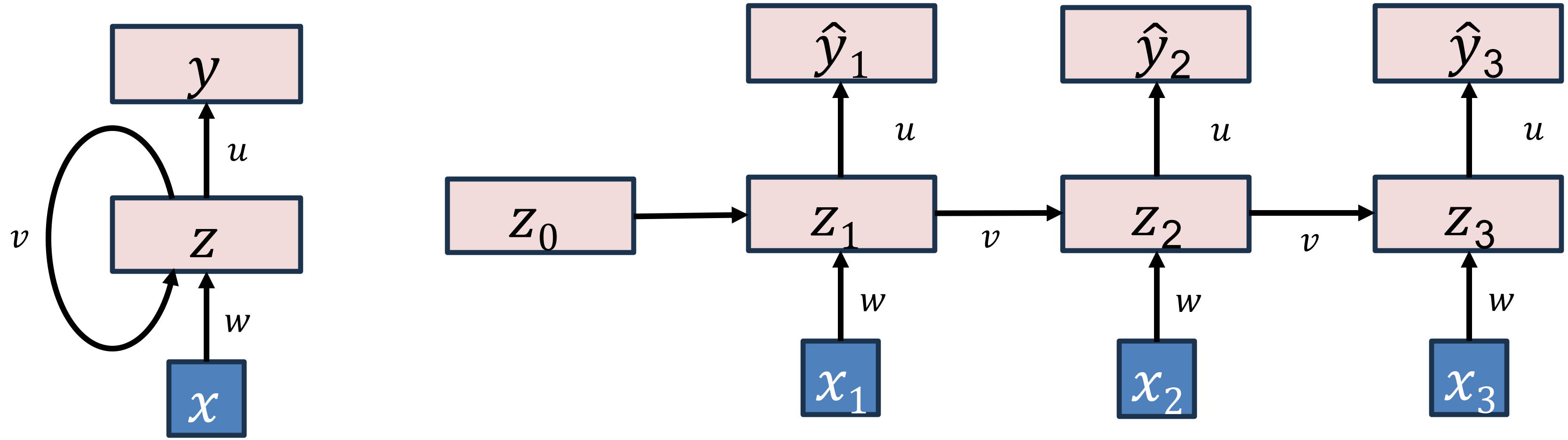
The → **quick** → **brown** → **fox**



Gupta, G., Kshirsagar, M., Zhong, M. et al. Comparing recurrent convolutional neural networks for large scale bird species classification.

Recurrent Neural Networks (RNNs)

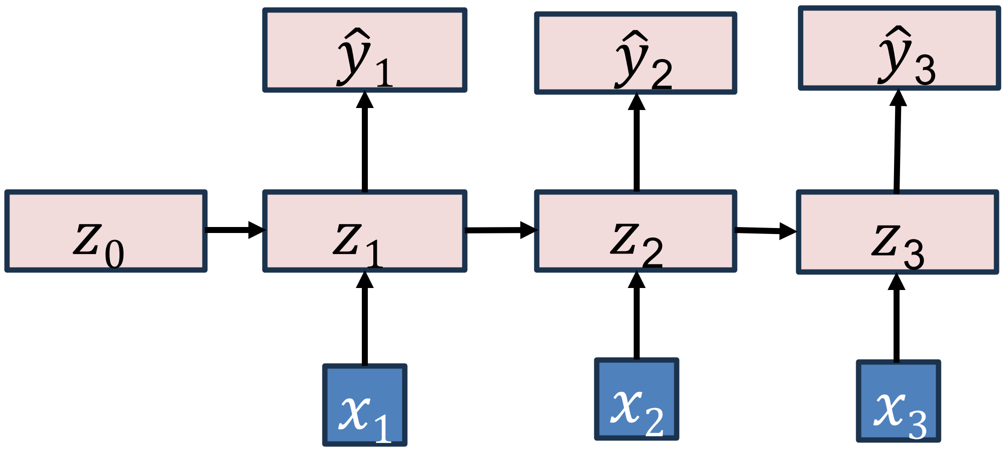
- RNNs introduce **cycles** in the computational graph
- Allowing information to persist; **memory**



RNNs: High Level

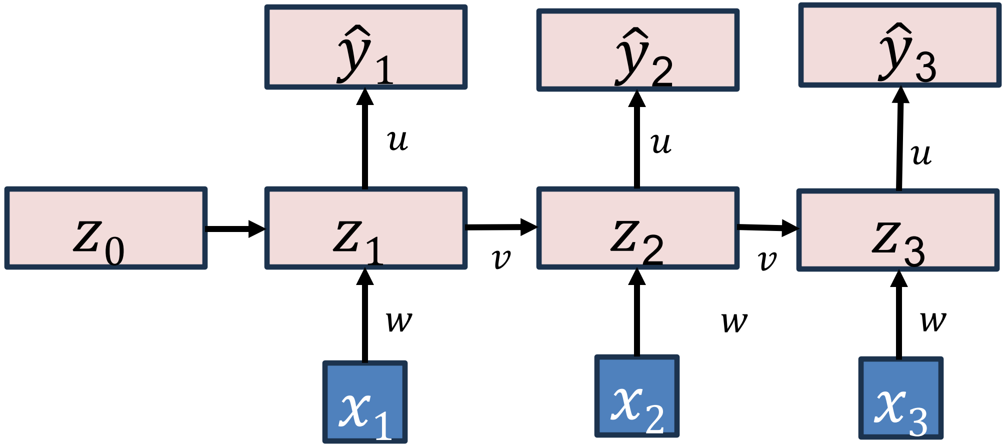
At each time step t:

- Receive input token x_t
- Receive old hidden state z_{t-1}
- Use x_t and z_{t-1} to compute new hidden state z_t
- Use z_t to predict $\hat{y}_t$



Recurrent Neural Networks (RNNs)

- In each time step, the **input value** and the **output of previous hidden state** are used in the computation
- Internal state, **memory** – inputs received at earlier time steps affect the RNN's response to the current input.



$$\hat{y}_t = g_y(Uz_t)$$

$$z_t = g_z(Vz_{t-1} + Wx + b)$$

RNNs for Language Modeling

Key Application: Language Modeling

- Basic idea: assign probability to text

$$P(w_1, w_2, \dots, w_n)$$

$$P(w_{\text{next}} \mid w_1, \dots, w_{n-1})$$

- Underlies ChatGPT, Claude, etc.
  - **LLM = large language model**

Next-Token Prediction

- Treat text as a sequence of **tokens**
- Simplest: tokens = characters

“Hi, class” ⇒

H	i	,	_	c	l	a	s	s

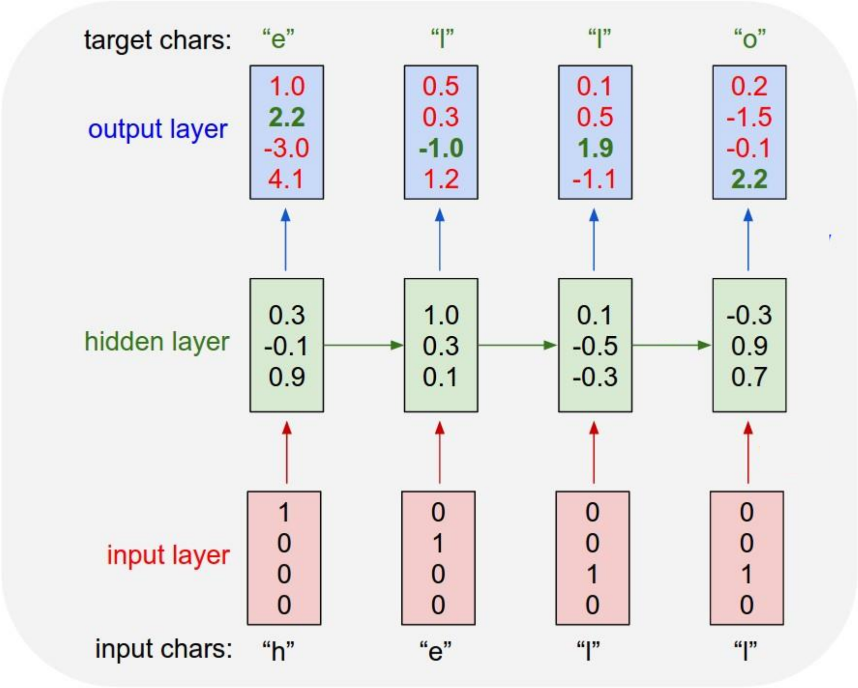
- OpenAI’s GPT4o uses over 200,000 tokens

Hi, class हैलो क्लास 同学们好

https://tiktokenizer.vercel.app

Example: RNNs on Text

- Simple example
  - 4 tokens: “h”, “e”, “l”, “o”
  - Hidden state has 3 dimensions
- Training: try to make output match targets
- Generation: sample!
  - (Same as with n-gram)



https://karpathy.github.io/2015/05/21/rnn-effectiveness/

Q1.1 Quiz Break

What is the primary characteristic that distinguishes Recurrent Neural Networks (RNNs) from standard feedforward networks?

A) They use convolutional layers to process spatial data.
B) They have loops in their architecture, allowing information to persist.
C) They cannot be trained using backpropagation.
D) They can only have a single hidden layer.

Q1.1 Quiz Break Quiz Break

What is the primary characteristic that distinguishes Recurrent Neural Networks (RNNs) from standard feedforward networks?

A) They use convolutional layers to process spatial data.
B) They have loops in their architecture, allowing information to persist.
C) They cannot be trained using backpropagation.
D) They can only have a single hidden layer.

Q1.2 Quiz Break

RNNs (Recurrent Neural Networks) are particularly well-suited for processing which type of data?

A) Tabular data where column order doesn't matter.
B) Static, high-resolution images.
C) Sequential or time-series data.
D) Unlabeled data with no clear structure.

Q1.2 Quiz Break

RNNs (Recurrent Neural Networks) are particularly well-suited for processing which type of data?

A) Tabular data where column order doesn't matter.
B) Static, high-resolution images.
C) Sequential or time-series data.
D) Unlabeled data with no clear structure.

The Attention Mechanism

From RNNS to Transformers

- **RNNs** handle sequences but **struggle with long-term dependencies**
- Transformers allow parallelization and efficient context handling
- Example: “The cat the dog chased ran away.”
  - Who ran away?
  - Need to remember/attend to earlier words
- Goal: **decide which words matter most!**

Word Representations & Context

- We use vectors to represent words ("embeddings")
- Recall:
  - One-hot representation
  - Dense embedding
    - Vector captures **meaning**

"dog"

$$[0 \;\; 1 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0]$$

$$[0.13 \quad 0.87 \quad -0.23 \quad 0.46]$$

Problem: the meaning of a word depends on the words around it

Word Representations & Context

"The monkey ate the banana. It was _____, wasn't it?"

Does "it" mean **monkey** or **banana**?

Could be:

- The monkey ate the banana. It was ripe, wasn't it?
- The monkey ate the banana. It was hungry, wasn't it?

Meaning depends on past words (**context**)

The attention mechanism produces **contextual embeddings.**

Attempt 1: Naïve Contextual Embedding

- Each token has a fixed embedding vector x_i
- A crude attempt at contextual embedding: average over context
- Equal "attention" to every previous token

Tokens		Fixed Embeddings
1	the	[0.45 0.23]
2	monkey	[0.39 0.72]
3	ate	[0.83 0.61]
4	the	[0.45 0.23]
5	banana	[0.25 0.18]
6	it	[0.63 0.41]
7	was	[0.63 0.41]
8	ripe	[0.70 0.67]
9	wasn't	[0.14 0.61]
10	it	+ [0.63 0.41]

[4.98 4.93]

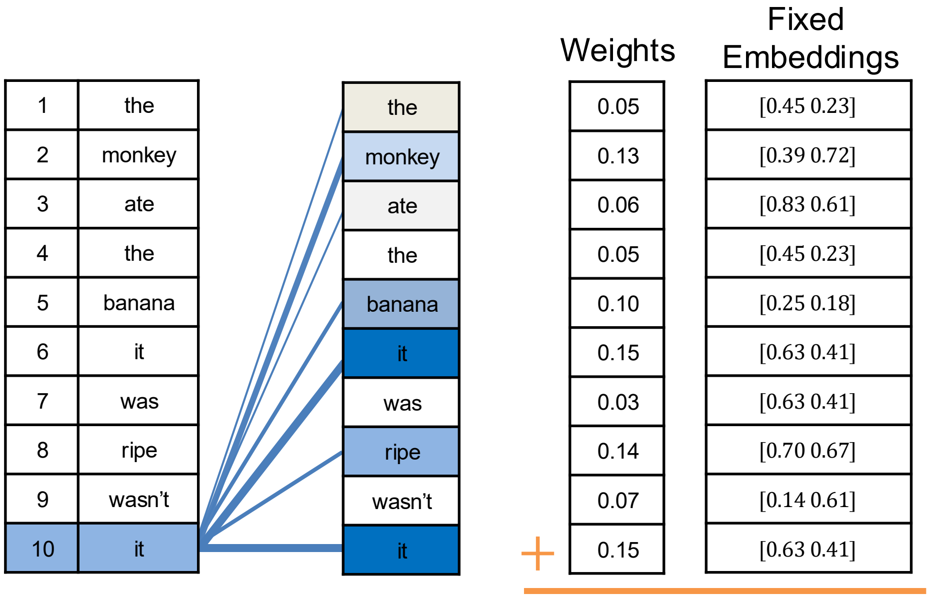
In math:
for the i-th token

$$c_i = \frac{1}{i}\sum_{j=1}^{i} x_j$$

Contextual embedding for "it": [0.498 0.493]

Attempt 2: Assigning Weights

- Humans focus selectively — machines can too
- We can assign weights based on relevance
  - Idea: weight similar words highly
  - If $\langle x_i, x_j \rangle$ large, assign large weight
- Then take weighted sum



Contextual embedding for "it": [0.37 0.42]

In math: for i-th token

$$r_{ij} = \frac{1}{\sqrt{d}} \langle x_i, x_j \rangle$$

$$p_{i,:} = \mathrm{softmax}(r_{i,:})$$

$$c_i = \sum_{j=1}^{i} p_{ij} \cdot x_j$$

Final Attempt: The Attention Mechanism

Previous attempt:

- Used fixed embeddings in three locations.

$$r_{ij} = \frac{1}{\sqrt{d}} \cdot \langle x_i, x_j \rangle$$

$$p_{i,:} = \mathrm{softmax}(r_{i,:})$$

$$c_i = \sum_{j=1}^{i} p_{ij} \cdot x_j$$

In the attention mechanism:

- Each token is associated with **three** vectors
- Query: q_i, the one attended from
- Key: k_j, the one attended to
- Value: v_j, the context being generated

Query Key

$$r_{ij} = \frac{1}{\sqrt{d}} \cdot \langle q_i, k_j \rangle$$

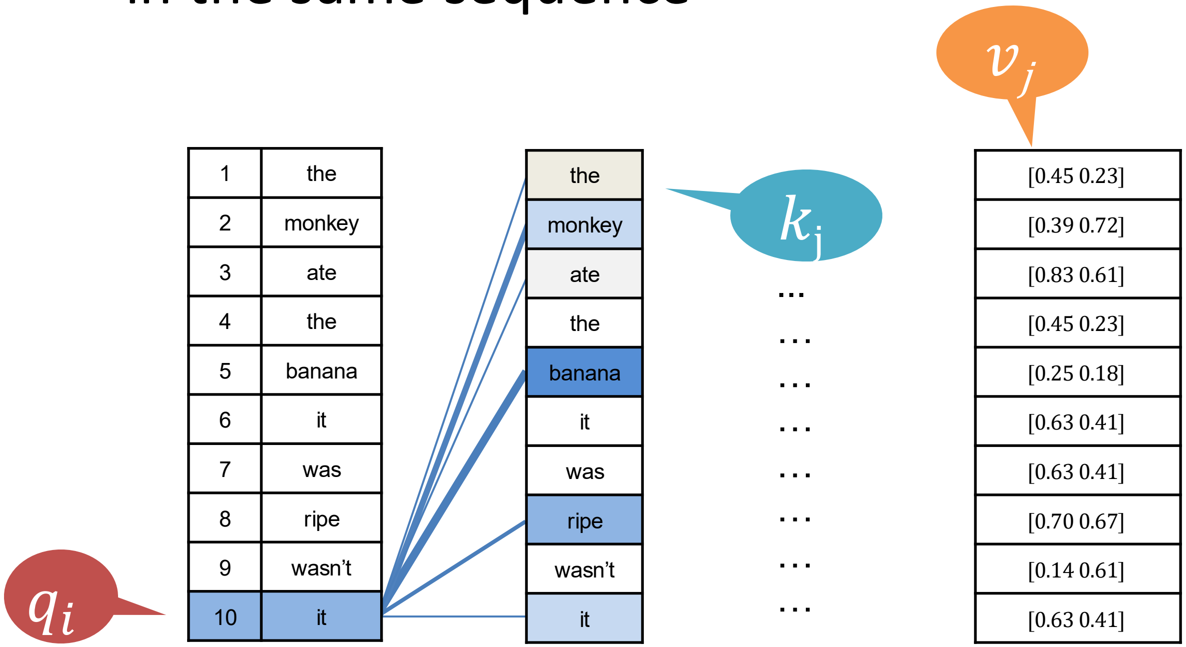
$$p_{i,:} = \mathrm{softmax}(r_{i,:})$$

Value

$$c_i = \sum_{j=1}^{i} p_{ij} \cdot v_j$$

The Attention Mechanism

Each token attends to all previous tokens in the same sequence

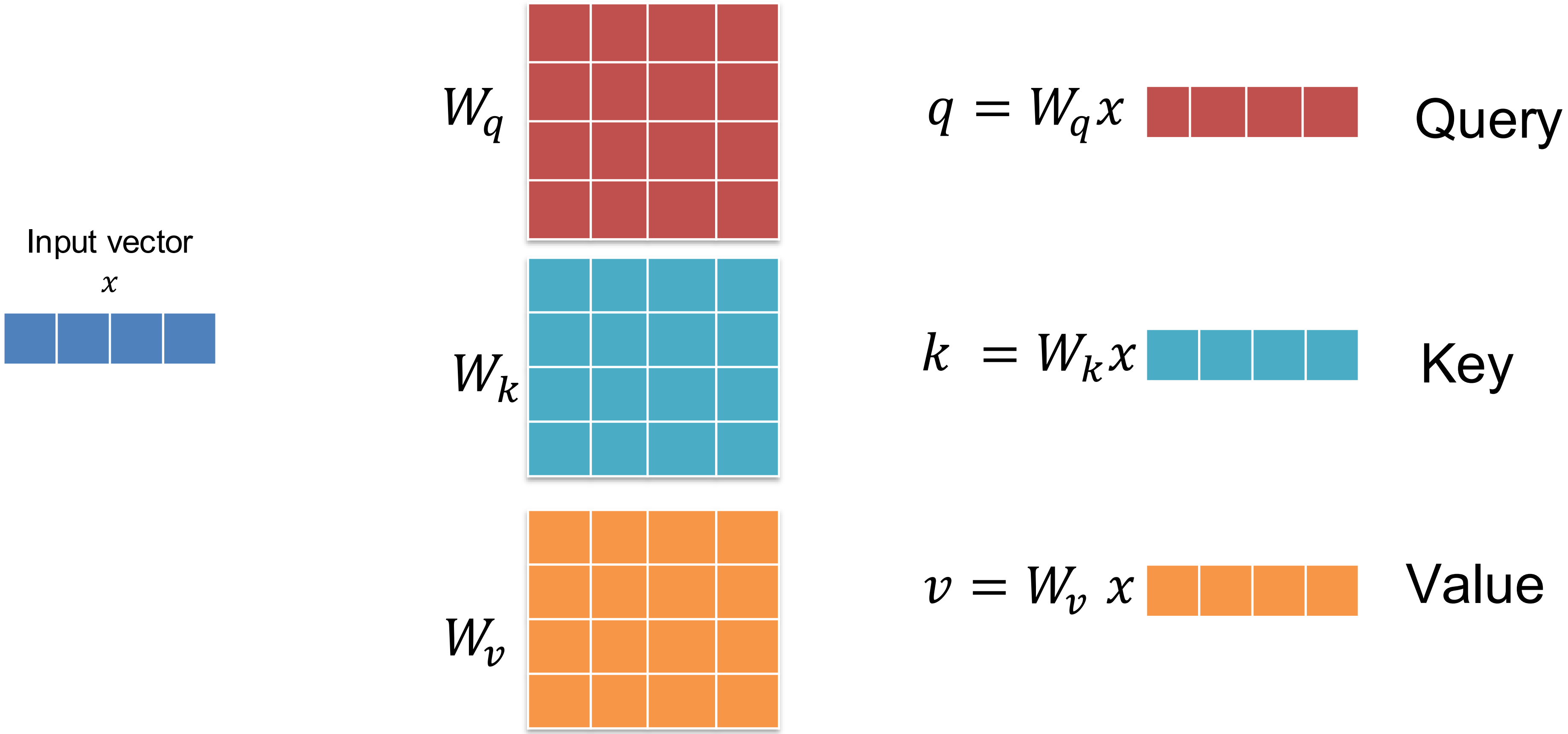


$$r_{ij} = \frac{\langle q_i, k_j \rangle}{\sqrt{d}}$$

$$p_{i,:} = \mathrm{softmax}(r_{i,:})$$

$$c_i = \sum_j p_{ij} \cdot v_j$$

Query, Key, and Value Matrices



Notation for Attention

Queries, keys and values are written as matrices $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$

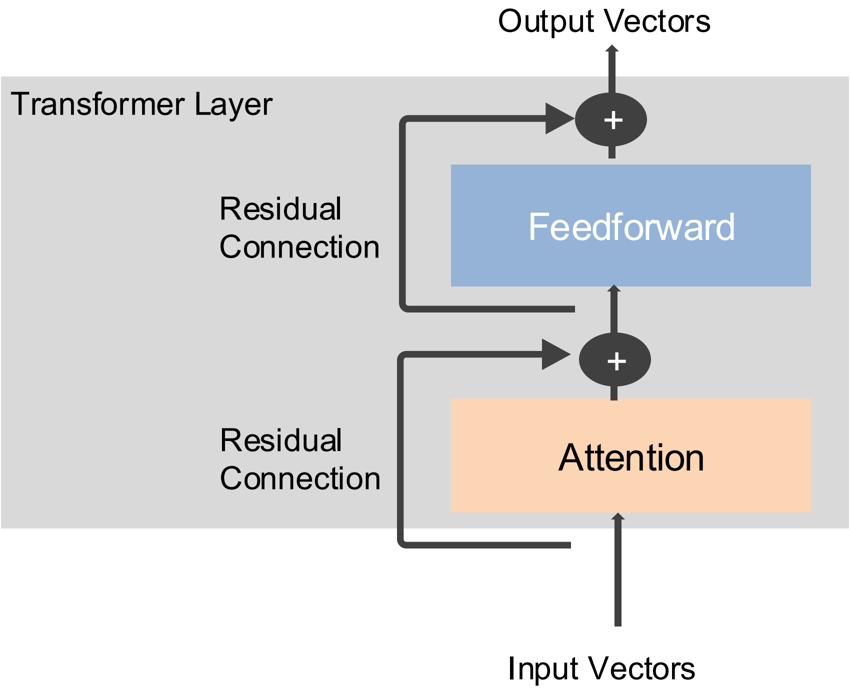
$$\text{Attention}\,(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)\text{V}$$

The Transformer Architecture

From Attention to Transformer

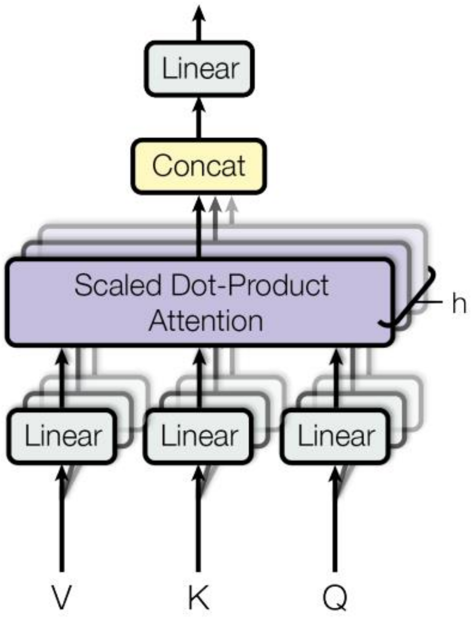
A single layer transformer consists of:

- Attention Mechanism
- Feed-Forward Network
- Residual Connections



Multi-Head Attention

- Outputs combined for richer representations
- Multiple heads learn different relationships (syntax, meaning, position)



[Vaswani, A., et al. (2017). Attention Is All You Need](#)

Positional Encoding

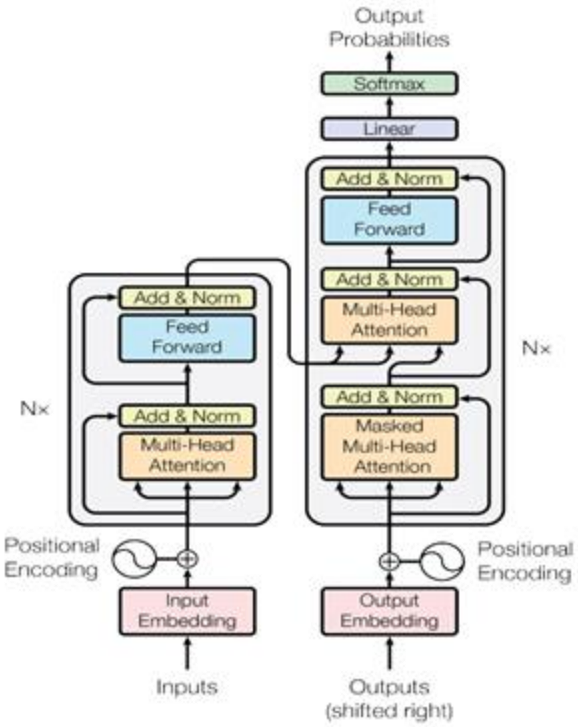
- Transformers have no recurrence — order must be added explicitly
- **Positional Encoding**: Information about the relative or absolute position of the tokens in the sequence
- Added to the input embeddings

position dimension

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

Transformer Architecture

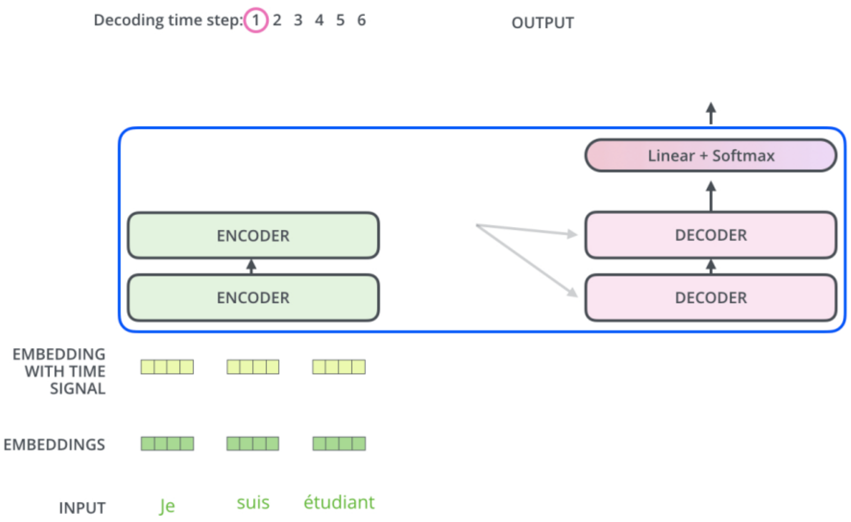
- Encoder–Decoder structure
- **Encoder**: maps an input sequence to a sequence of continuous representations z.
  - Useful for classification/translation
- **Decoder**: Given z, the decoder generates an output sequence of symbols one element at a time.
  - Useful for generation

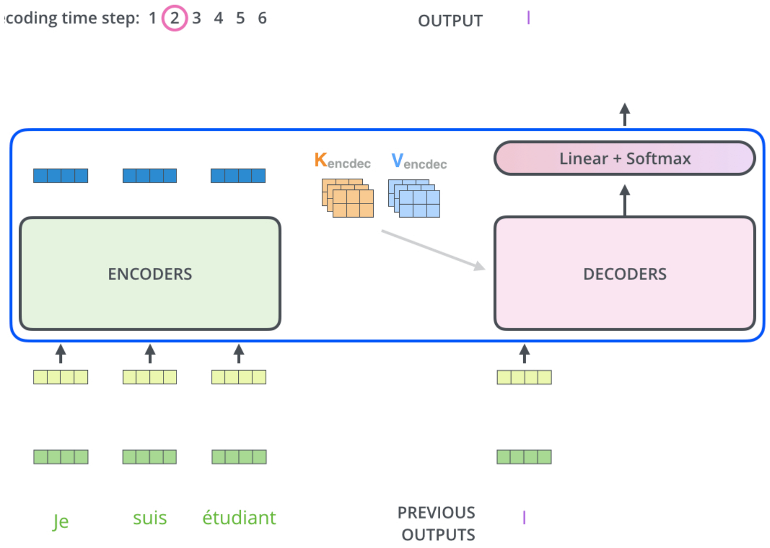


[Vaswani, A., et al. (2017). Attention Is All You Need]

Decoder

- **Masked multi-head attention**: each word attends to the words before it
- A **second attention** module that **attends the output of the encoder**





[The Illustrated Transformer]

Applications

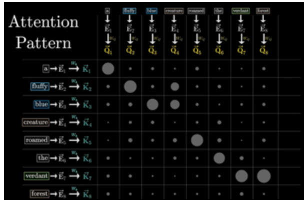
- Language Models: GPT, BERT, T5
- Vision: ViT (Vision Transformer)
- Multimodal: CLIP, DALL·E, GPT-4v
- Scientific: AlphaFold, time-series modeling, robotics

Further Reading/Viewing

- Jurafsky & Martin, Chapter 8
  - https://web.stanford.edu/~jurafsky/slp3/ed3book_aug25.pdf
- Russell & Norvig, Chapter 24

- Andrej Karpathy tutorial
  - https://karpathy.ai/zero-to-hero.html

- 3Blue1Brown:
  - https://www.youtube.com/watch?v=eMlx5fFNoYc



- The Illustrated Transformer
  - https://jalammar.github.io/illustrated-transformer/