When AI meets life sciences
当人工智能遇上生命科学

2025 清华大学 AI创新实践课程

张强锋，清华大学

Oct 20, 2025

Outline

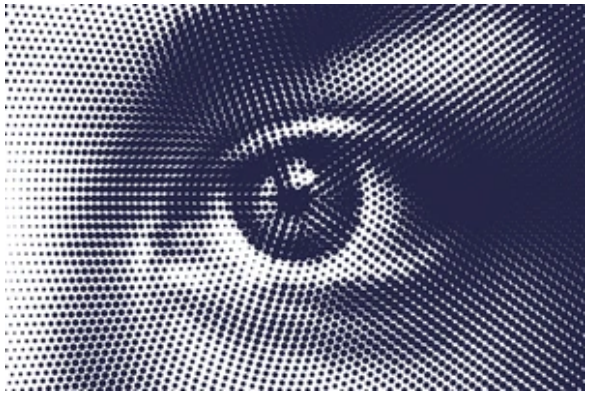
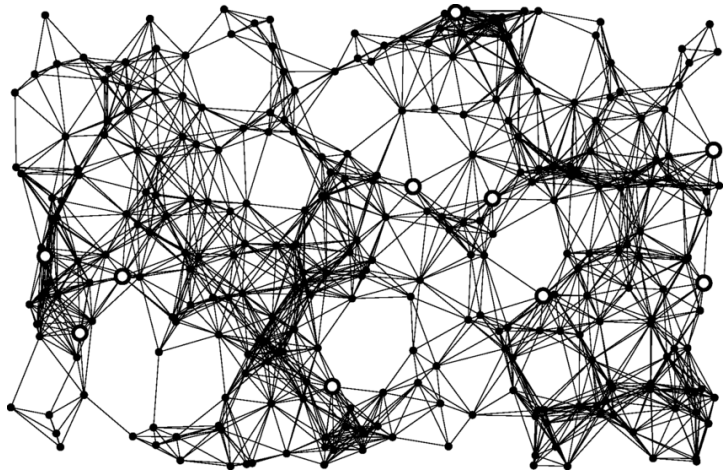


Image
CNN



Sequence
RNN/LSTM
Transformer

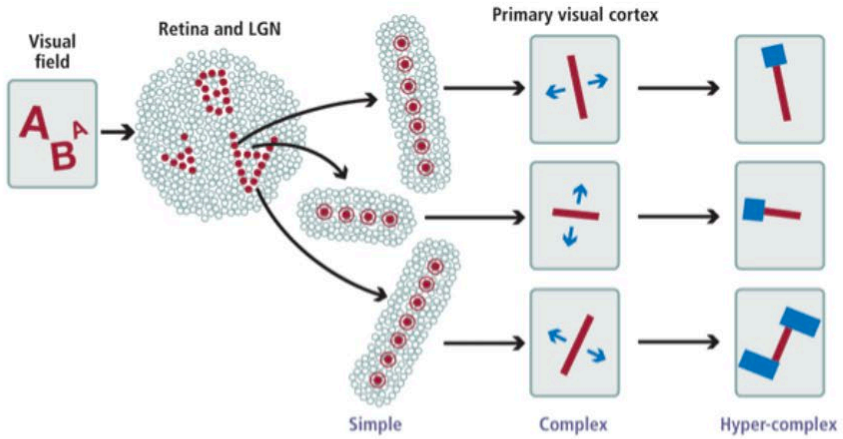


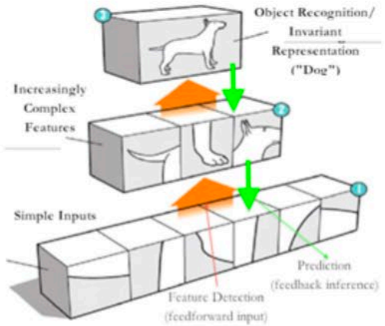
Graph
GNN

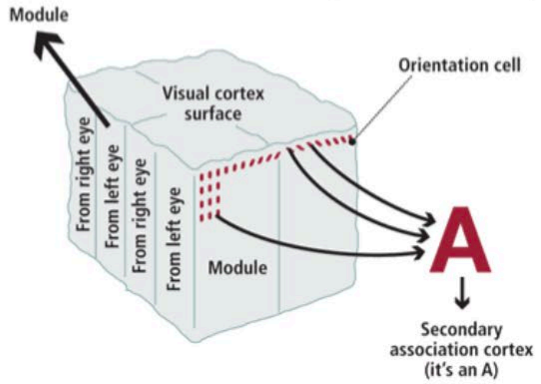
Neuroscience of vision

Human vision system

- **Simple cells → complex cells → hyper-complex cells**

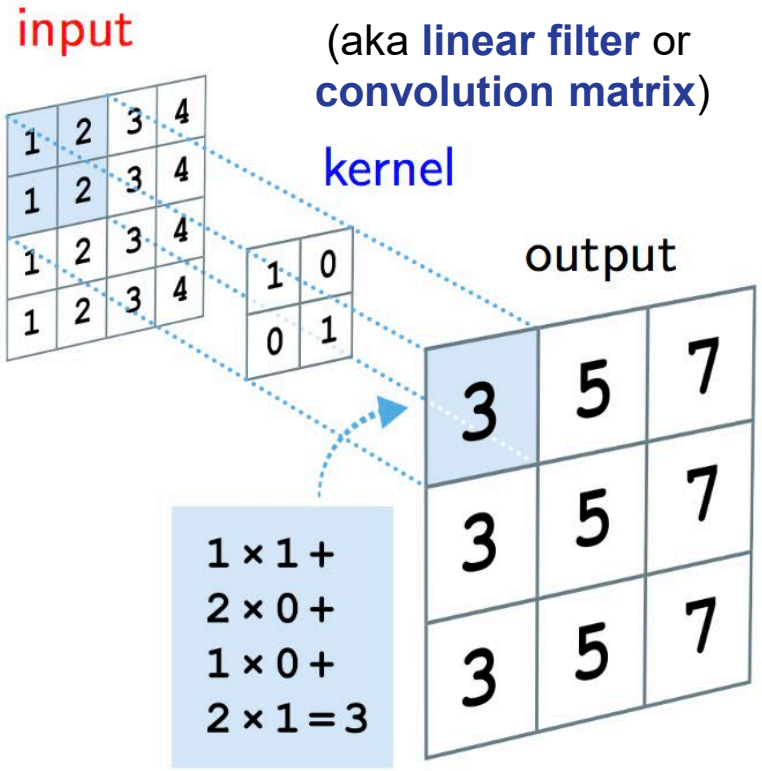


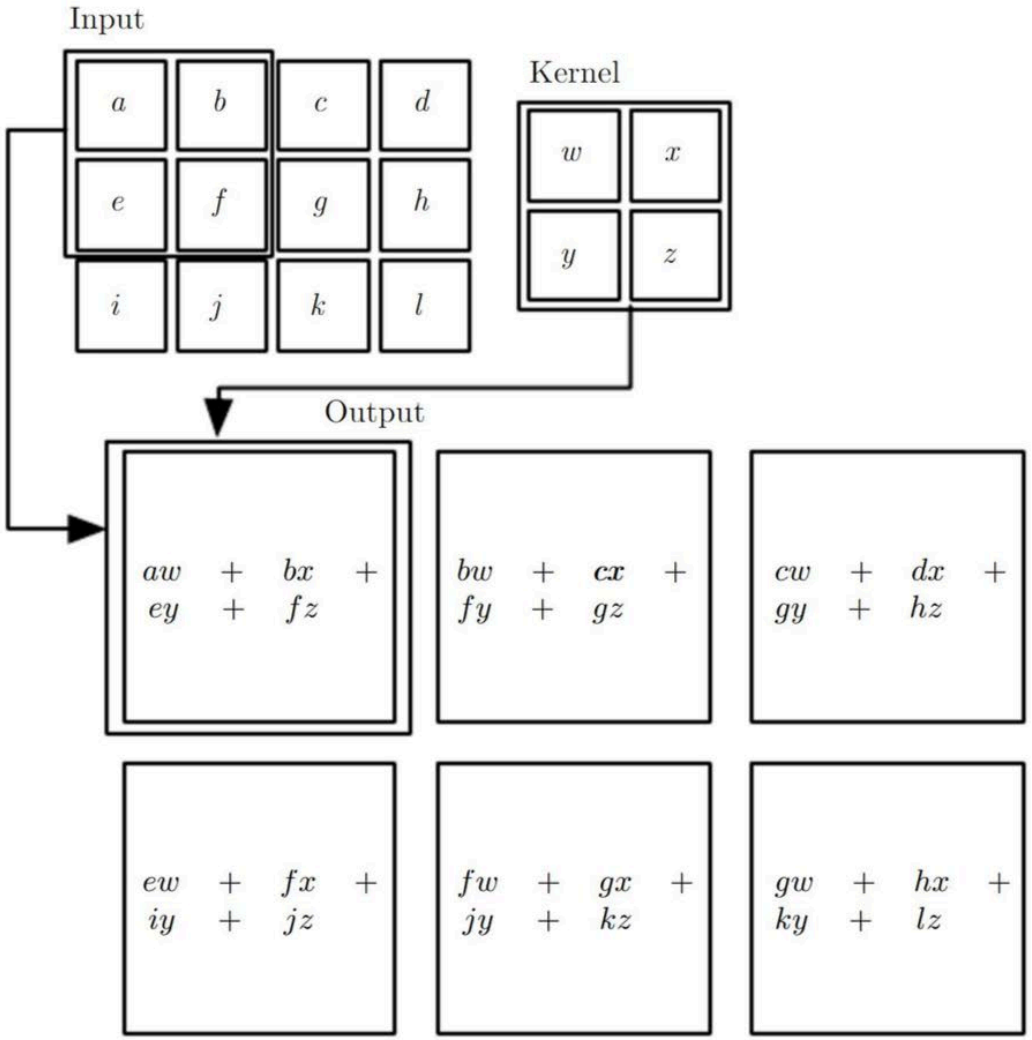




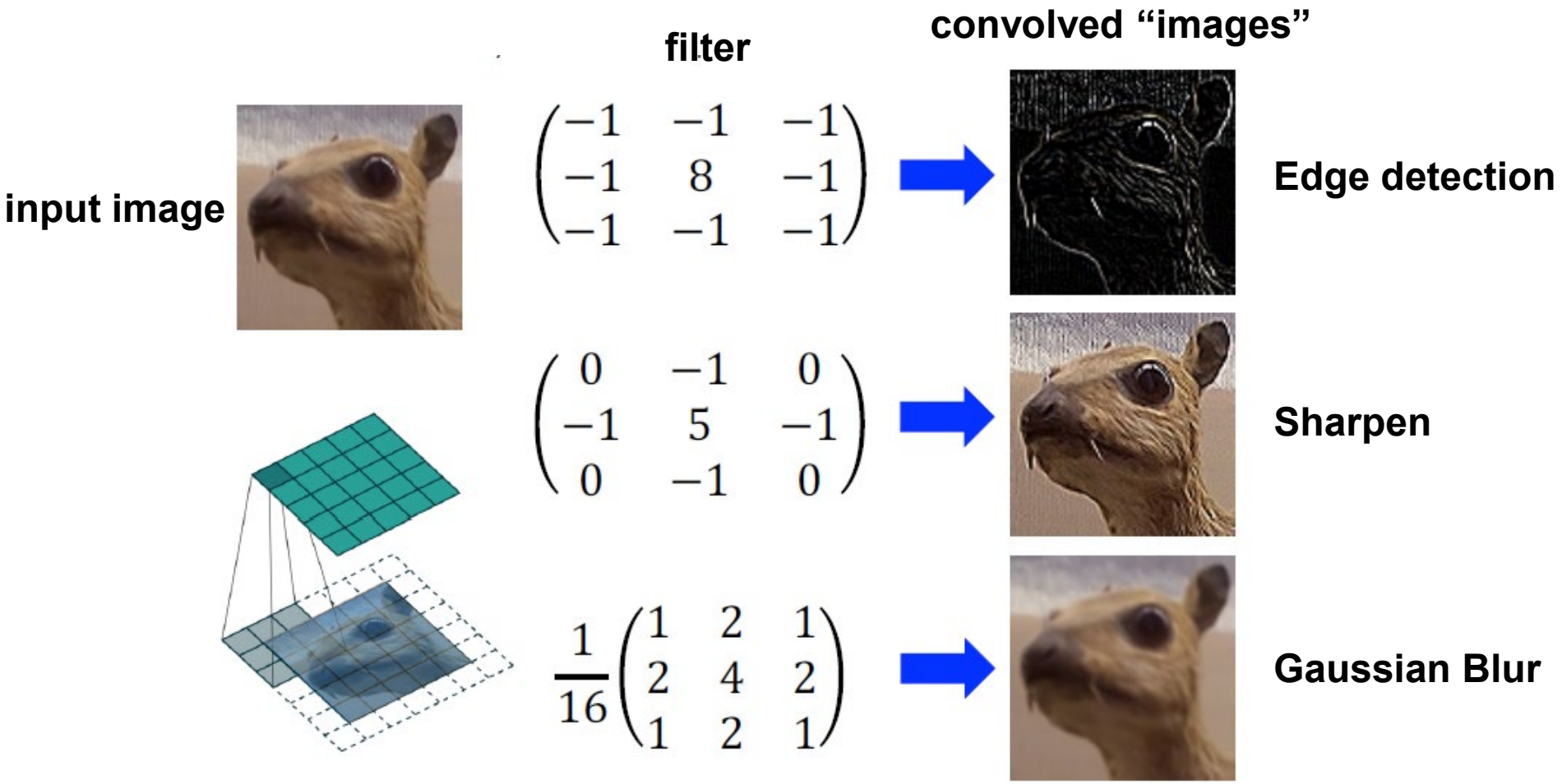
source: Univ. of Colorado, www.eidolonspeak.com

Linear filter or convolution



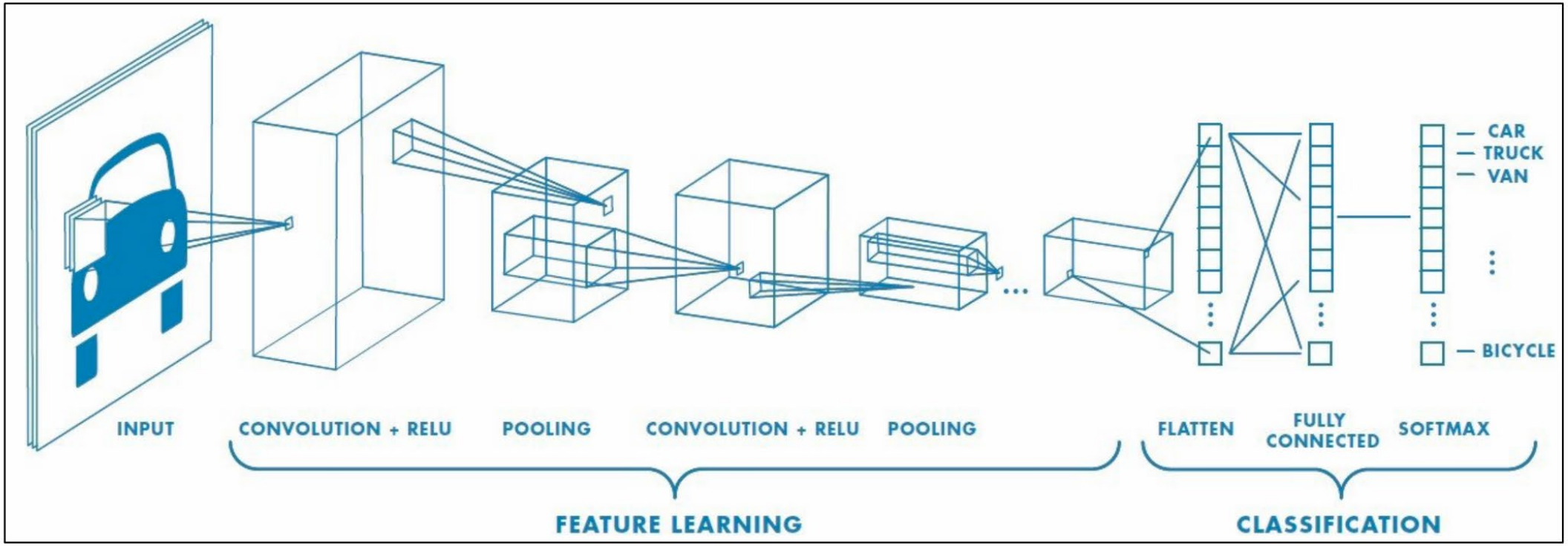


Different linear filters for different tasks



… Combination of convolution!

CNN is a neural network containing a convolutional layer

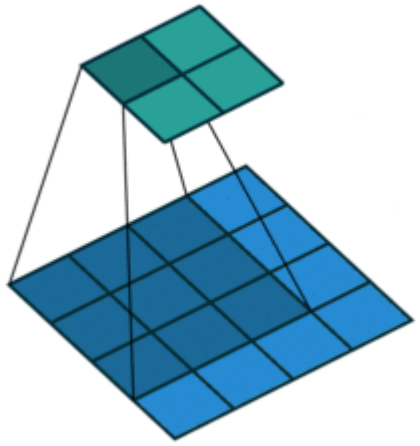


Generate hierarchy of features

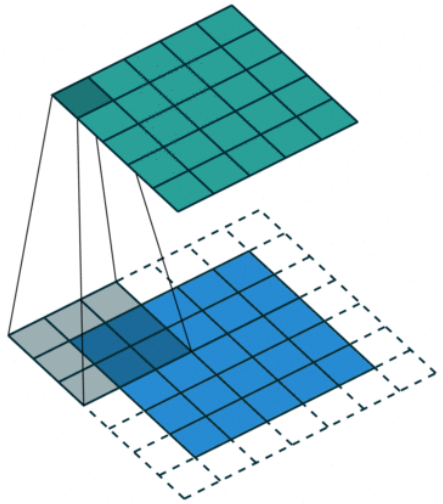
Convolutional Neural Network (CNN)

- Employ a mathematical operation called convolution
  - convolution: a special kind of linear operation
  - in ML, cross-correlation is often used instead
- Simply neural networks that use convolution in their layers in place of general matrix multiplication
- Tremendously successful in practical applications
- Especially for data with a known, grid-like topology
  - time series: 1D grid of samples taken at regular time intervals
  - image: 2D grid of pixels
  - 3D object: 3D grid of voxels

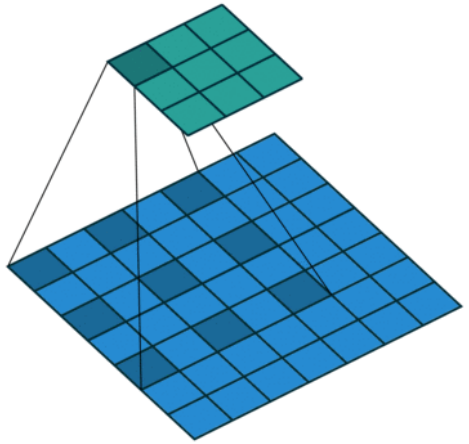
Convolution operators



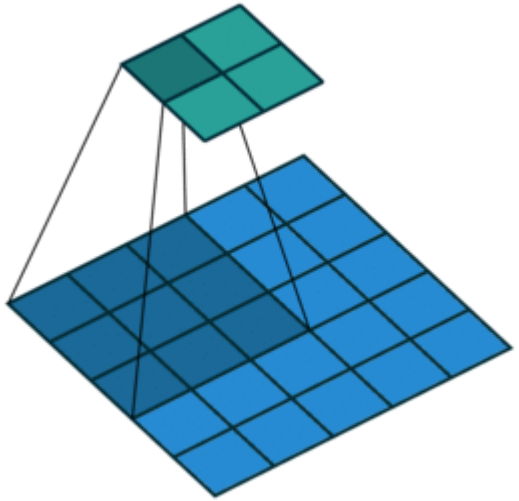
Normal convolution
Kernel size: 3x3
Stride: 1
Padding: 0
Dilation: 1



Padding convolution
Kernel size: 3x3
Stride: 1
Padding: 1
Dilation: 1



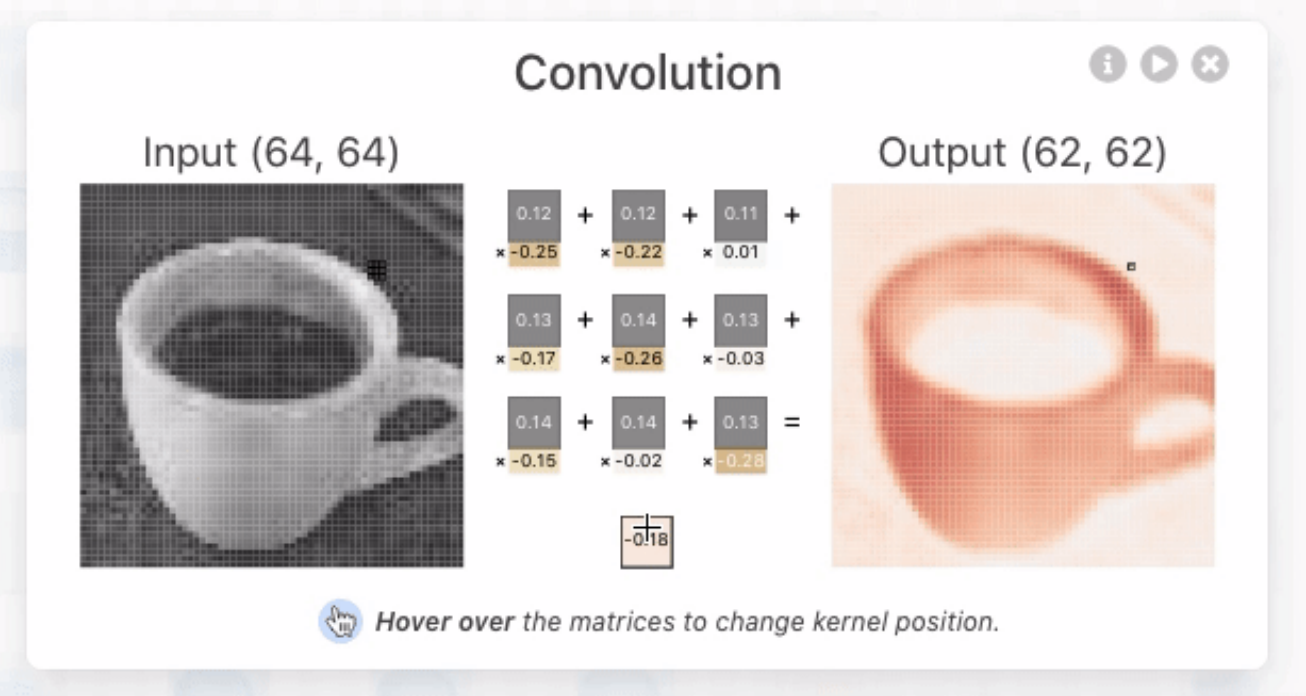
Dilated convolution
Kernel size: 3x3
Stride: 1
Padding: 0
Dilation: 2



Stride convolution
Kernel size: 3x3
Stride: 2
Padding: 0
Dilation: 1

Convolution on an image

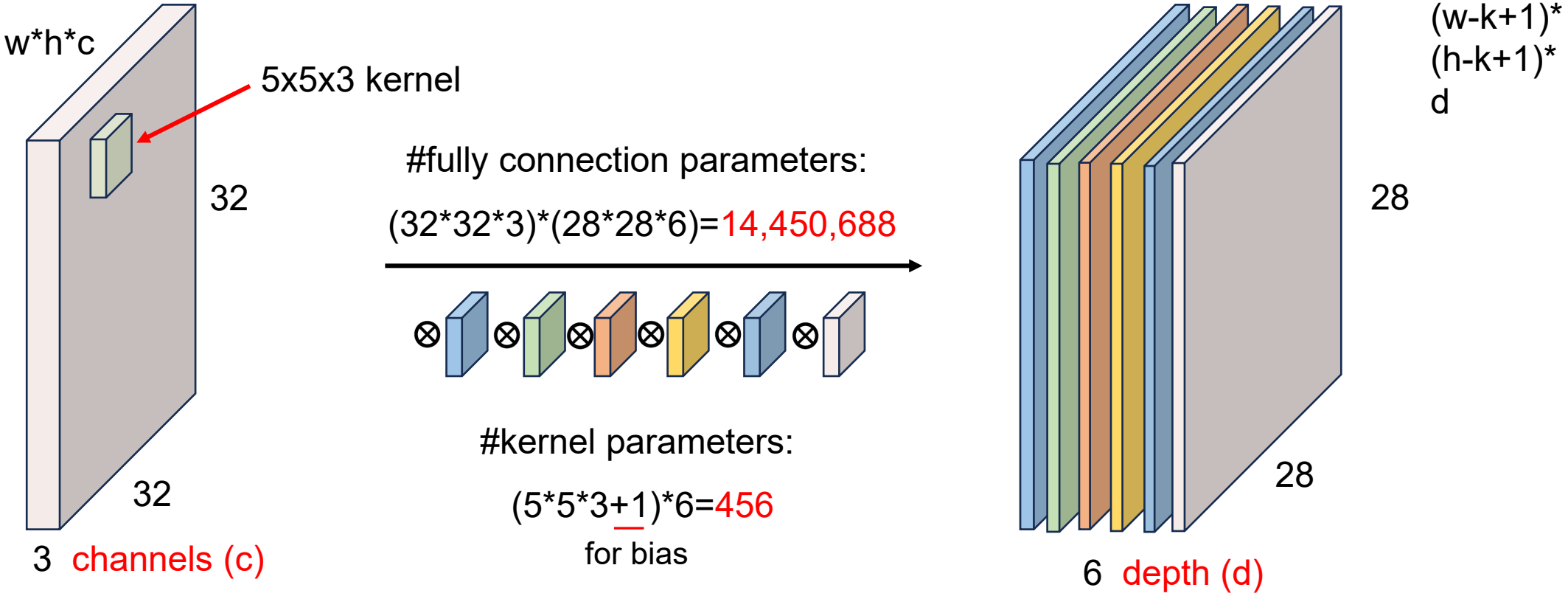
- Input image is convolved with a set of filters, giving feature maps



Try it yourself: https://poloclub.github.io/cnn-explainer/

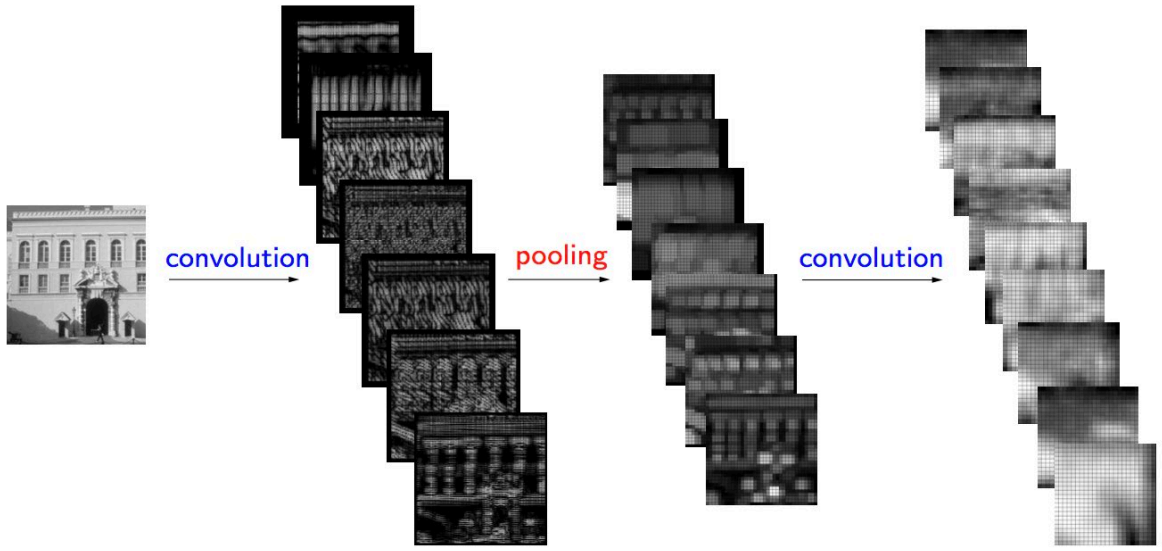
- Filter (aka kernel or convolution matrix)
  - Filter weights: trained (by backprop) or pre-determined
  - Weight sharing & local connectivity
  - Different filters extract different features

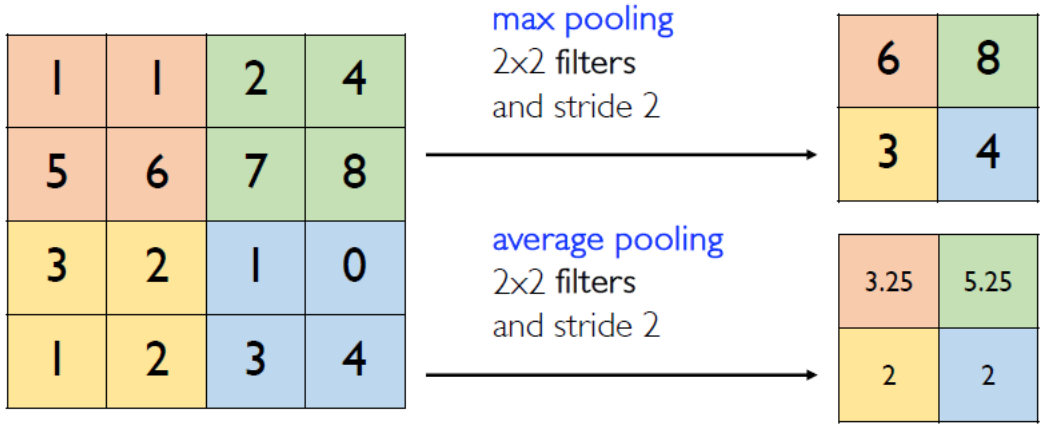
Convolution: weight sharing



Pooling/subsampling stage

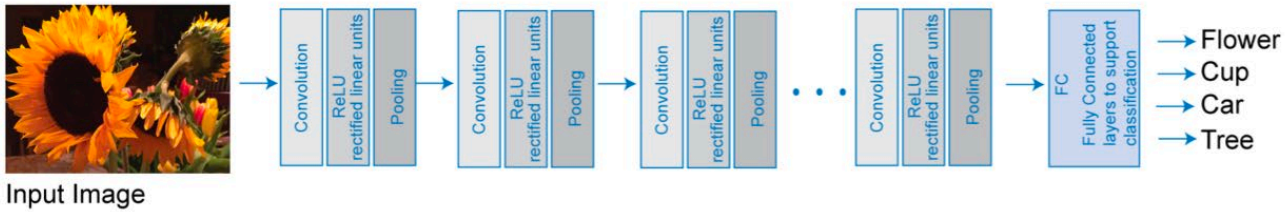
- Nonlinear down-sampling
  - Aggregates statistics of local features
  - Reduced variance: provides invariance to local transformations
  - Reduce the resolution of images through the processing pipeline

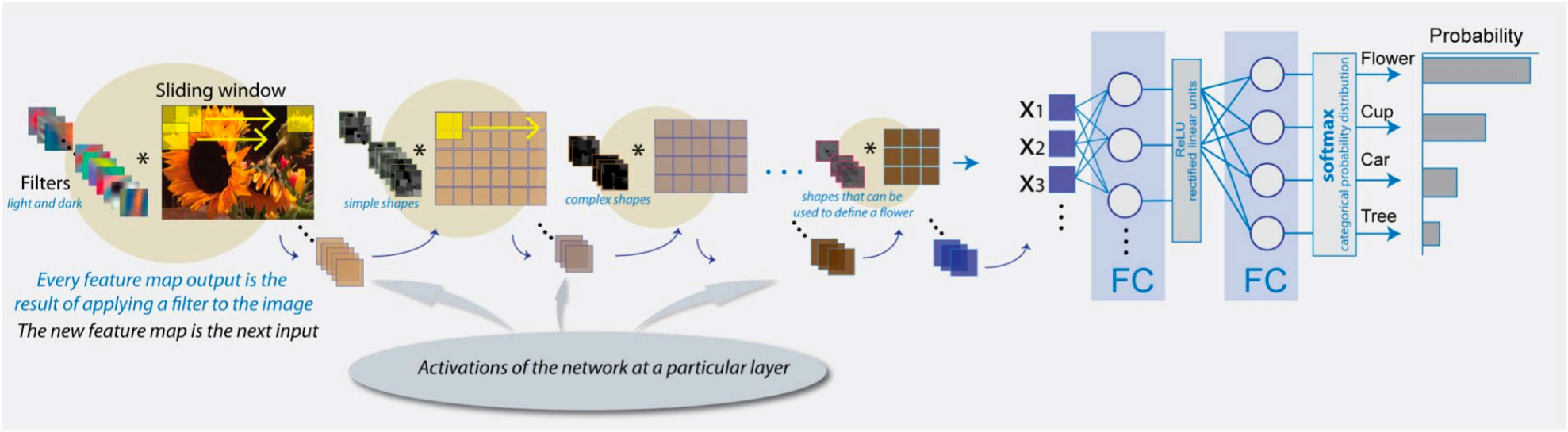




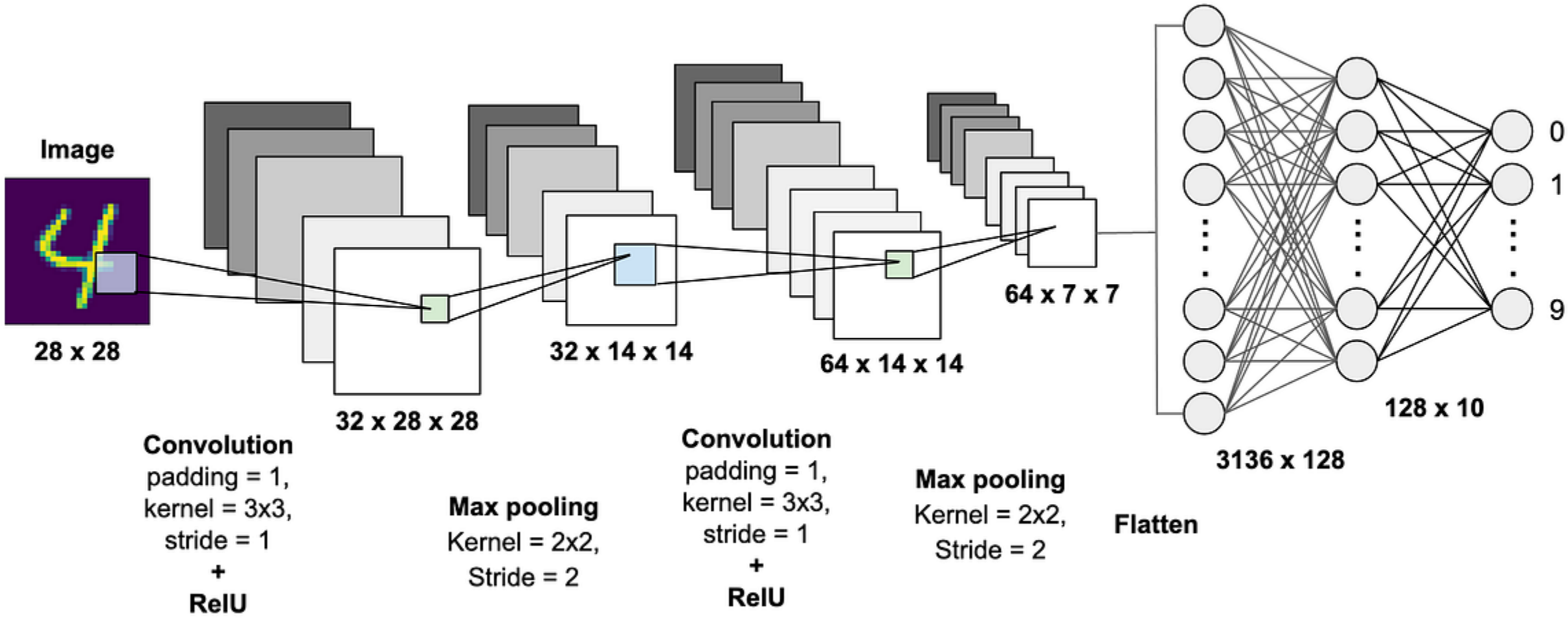
Fully connected (FC) layer

- For high-level reasoning such as classification and regression
  - ➢ Often with dropout regularization & softmax output





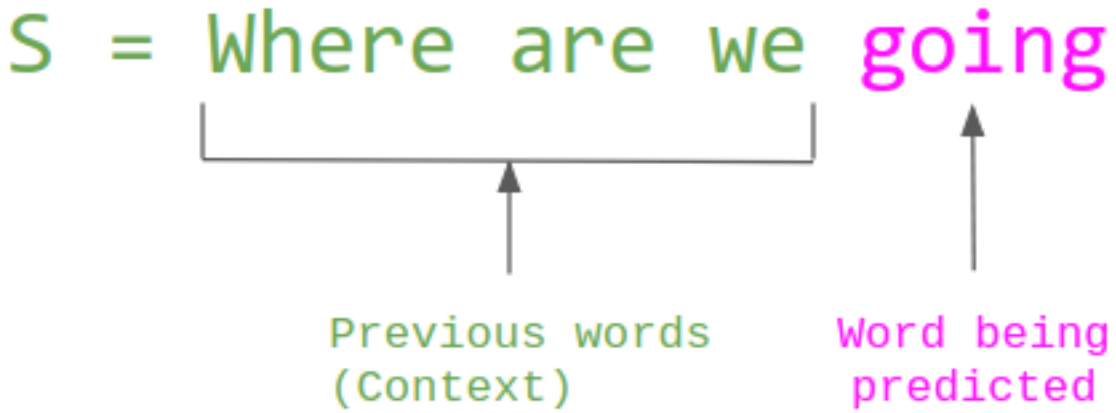
Putting it all together: digit classification



Sequence data (language) modeling



Sequence data (language) modeling



$$P(S) = P(\text{Where}) \times P(\text{are} \mid \text{Where}) \times P(\text{we} \mid \text{Where are}) \times P(\text{going} \mid \text{Where are we})$$

N-gram models

Text: “In consciousness studies the consciousness studies are fundamental”

Unigram	In	consciousness	studies	the	are	fundamental

Bigram	In consciousness	consciousness studies	studies the	the consciousness	studies are	are fundamental

Trigram	In consciousness studies	consciousness studies the	studies the consciousness	the consciousness studies	consciousness studies are	studies are fundamental

© AIML.com Research

N-gram models: writing a poem

unigram:

'<start>', '轻', '谁', '沉', '<start>', '我', '夺', '网', '若', '<end>'

'<start>', '毫', '千', '物', '观', '闲', '日', '烧', '<start>', '<end>',

由于 unigram 模型不考虑词的相互关系，且＜start＞的比例较大，因此即使在句子中间也可能会出现＜start＞。

bigram:

'<start>', '惆', '怅', '落', '飞', '而', '今', '日', '见', '<end>'

'<start>', '故', '人', '<end>'

'<start>', '风', '任', '公', '见', '客', '<end>'

trigram:

'<start>', '忽', '闻', '悲', '风', '四', '边', '来', '<end>'

'<start>', '起', '立', '明', '灯', '前', '<end>'

'<start>', '故', '山', '有', '松', '月', '<end>'

'<start>', '白', '犬', '离', '村', '吠', '<end>'

N-gram models: writing a poem

(2) 给定第一个字是“君”,三个模型表现如下。

unigram:

'<start>','君', '半' '<start>', '昔', '吹', '无', '方', '嵘', '<end>'

'<start>','君', '极', '人', '<start>', '欲', '里', '青', '<end>'

bigram:

'<start>', '君', '青', '条', '脱', '宝', '鞭', '从', '广', '陵', '绕', '床', '<end>'

'<start>', '君', '留', '人', '如', '飘', '若', '未', '穷', '<end>'

'<start>', '君', '糠', '养', '之', '可', '掇', '仙', '人', '间', '<end>'

trigram:

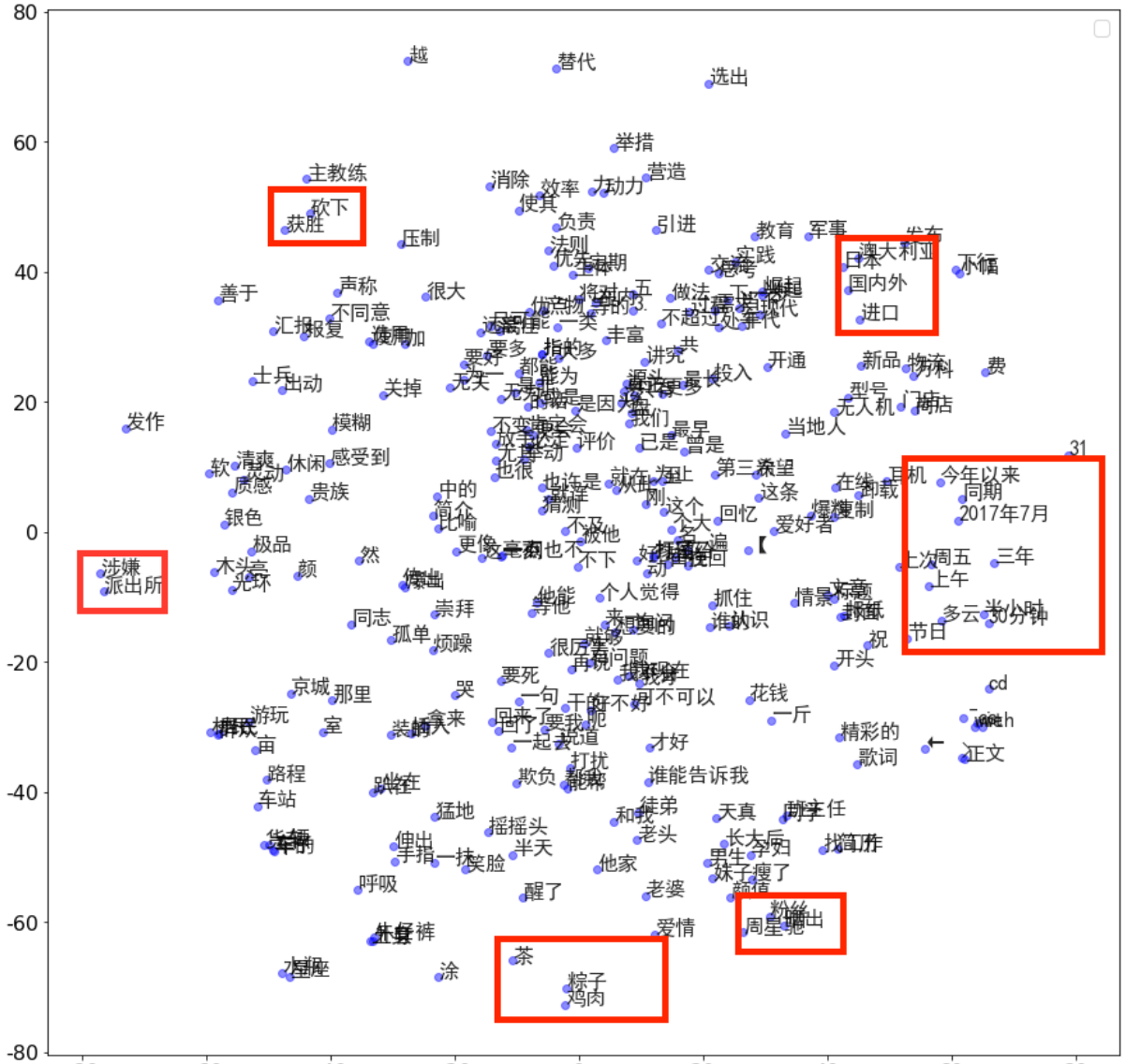
'<start>', '君', '为', '进', '士', '不', '得', '意', '<end>'

'<start>', '君', '夸', '通', '塘', '好', '<end>'

'<start>', '君', '莫', '驯', '<end>'

'<start>', '君', '从', '此', '谢', '情', '人', '<end>'

Where is the meaning: word vector

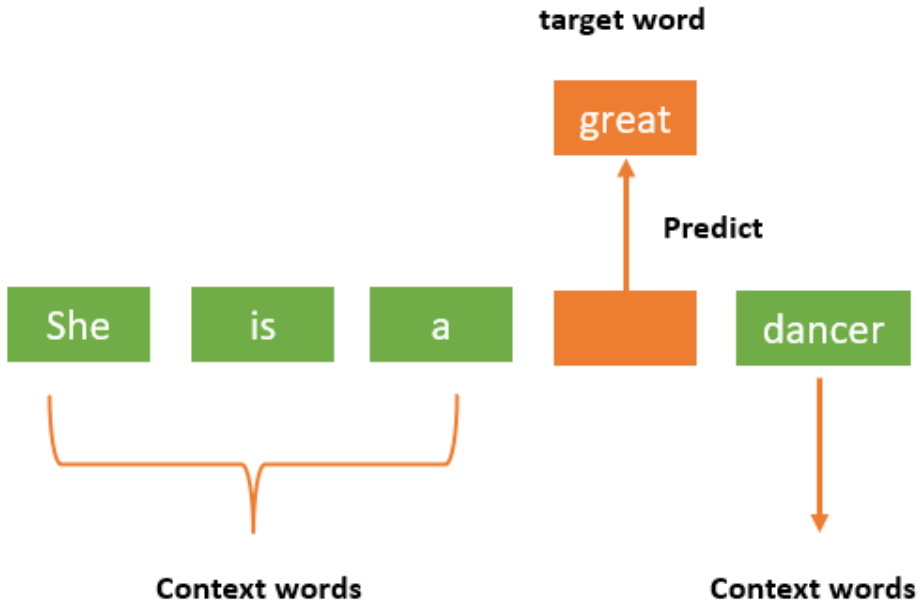


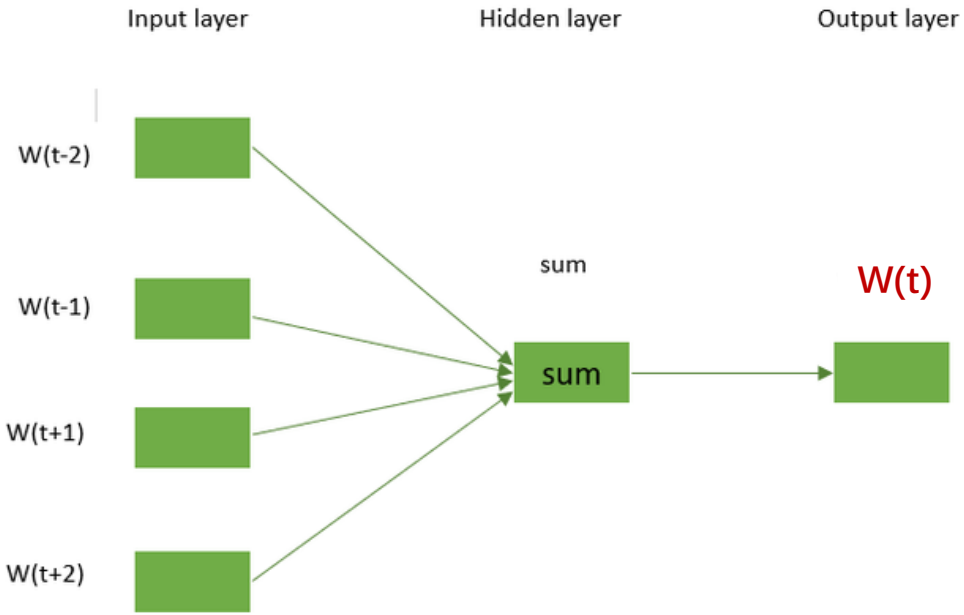
Where is similarity

Word embedding

- one-hot embedding
- Word2Vec
  - CBOW
  - Skip-gram
  - FastText
- GloVe

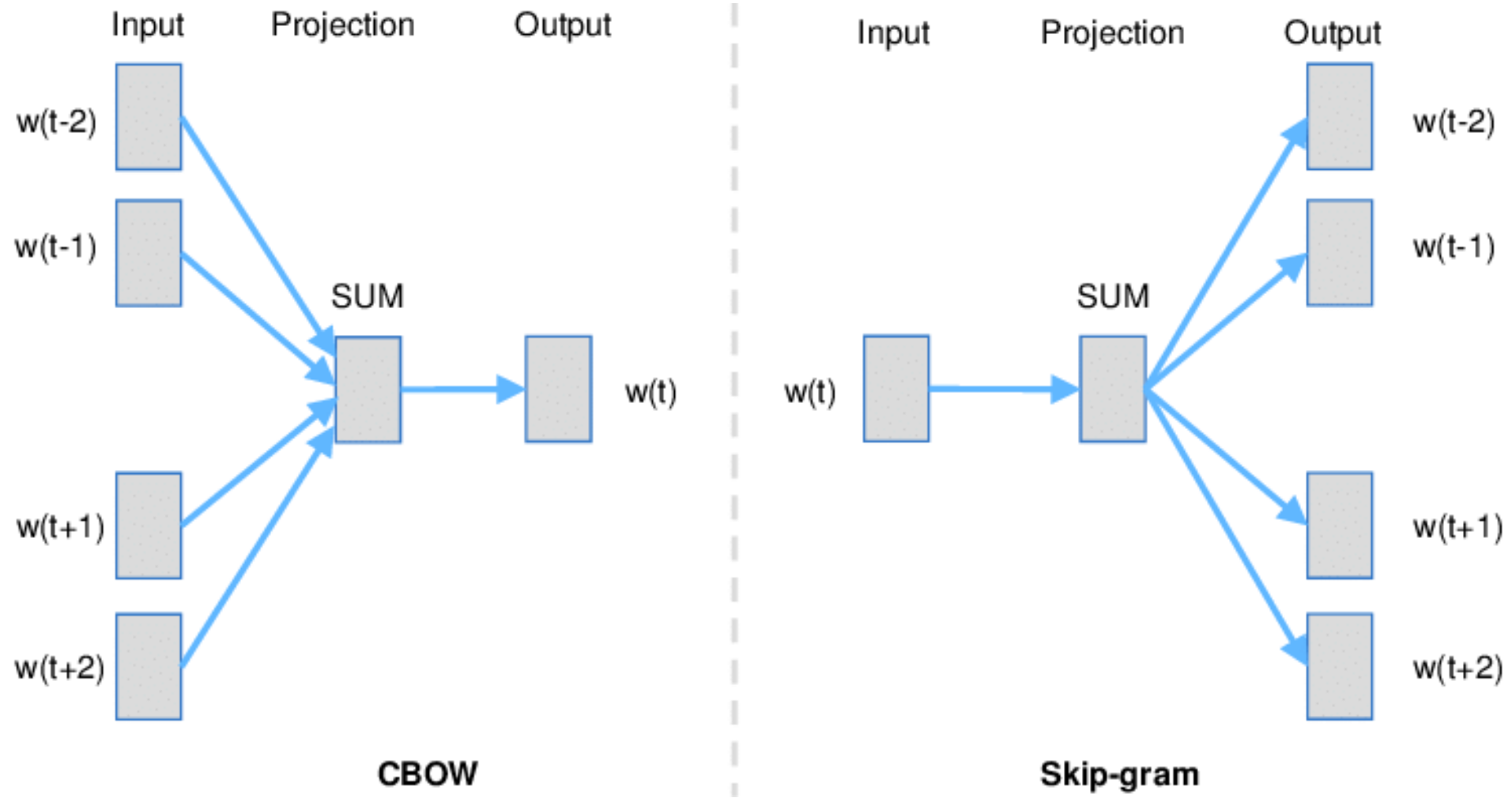
CBOW (Continuous Bag of Words)





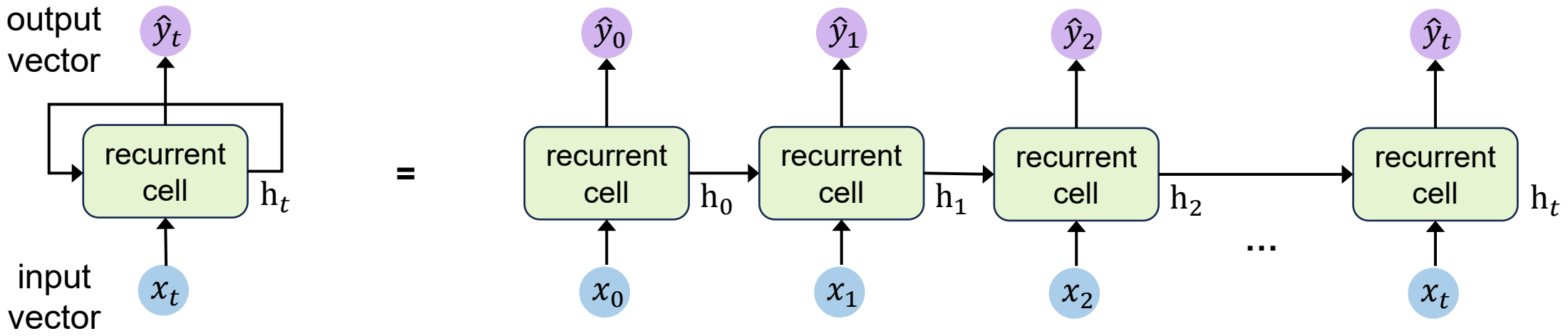
The real goal is not to complete the prediction task, but to learn the useful by-product that emerges from the process—the word embedding matrix.

CBOW vs. Skip-gram



Recurrent Neural Network (RNN)

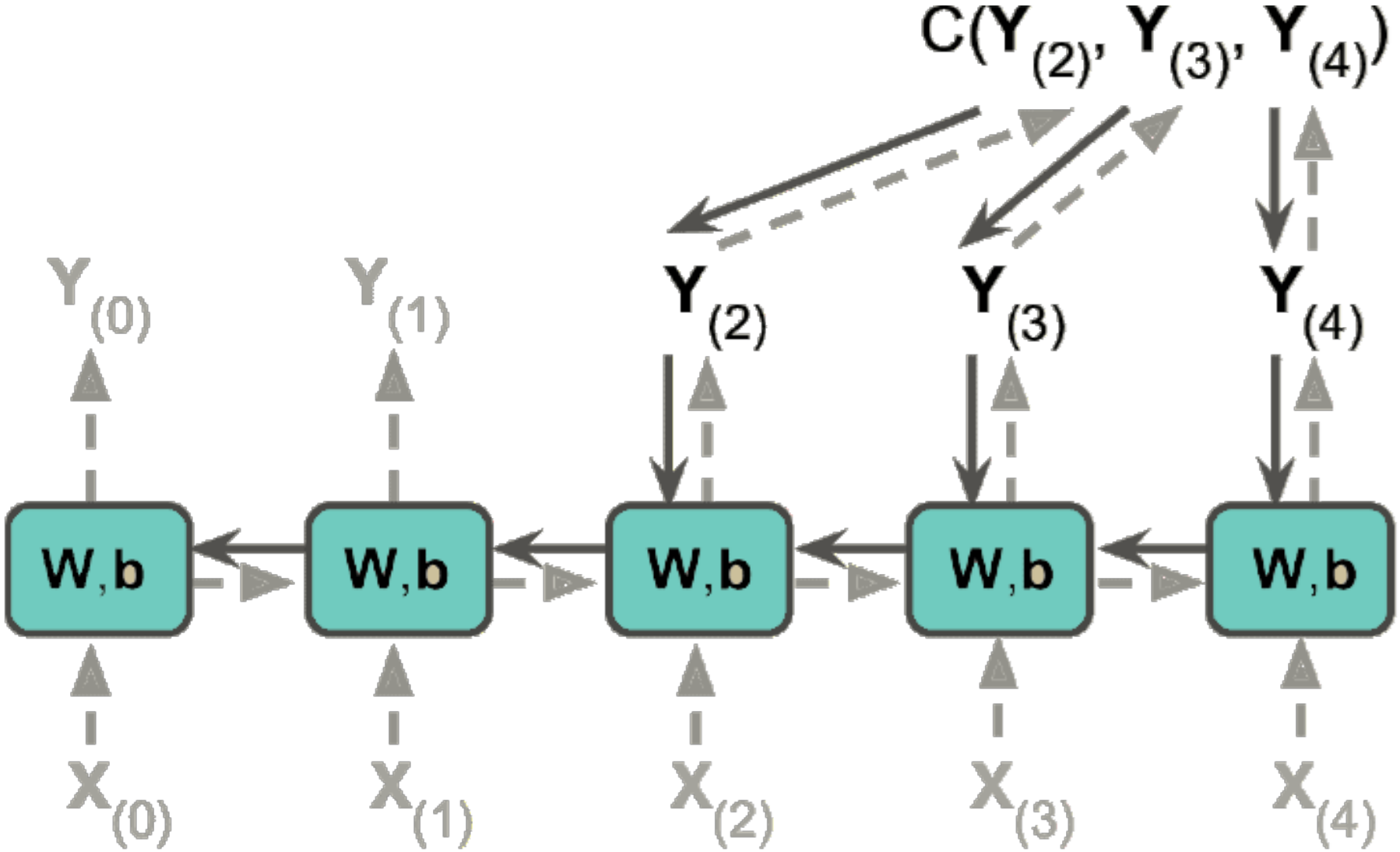
- Hidden state h_t is used to encode the information from all past timesteps
- Use temporally shared parameters.



$$\underline{\hat{y}_t} = f(\underline{x_t}, \underline{h_{t-1}})$$

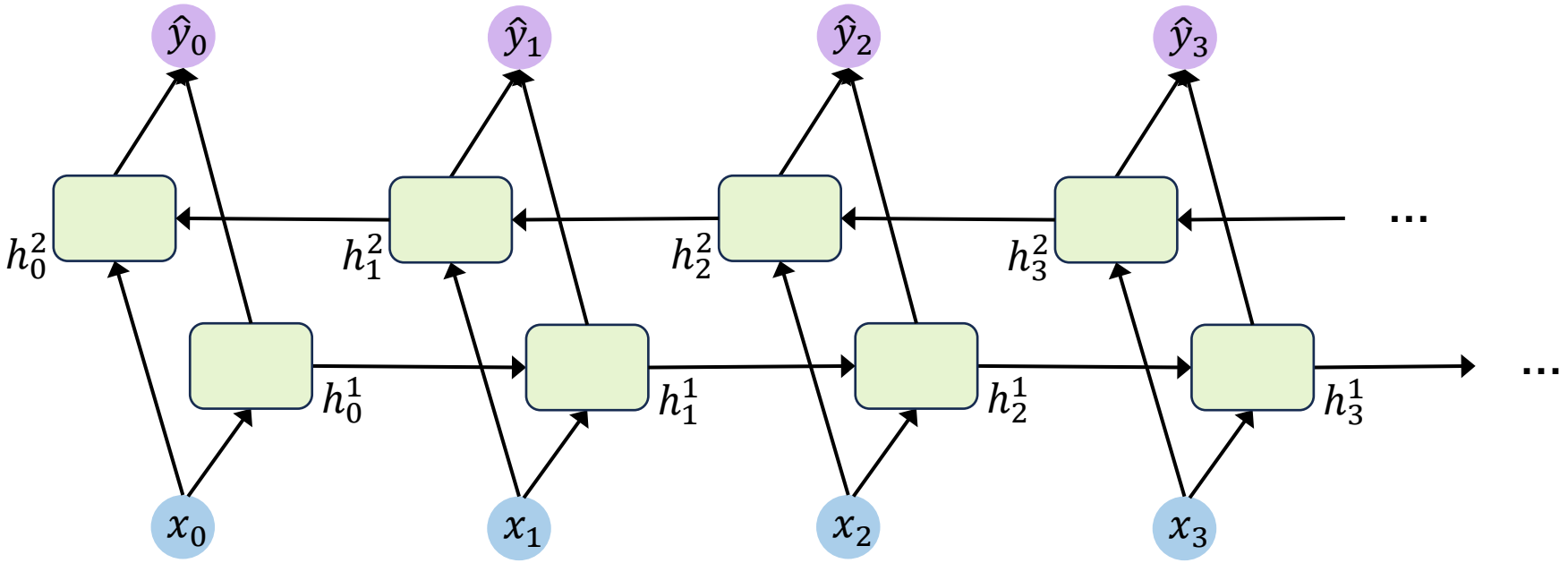
output input past memory

Train an RNN



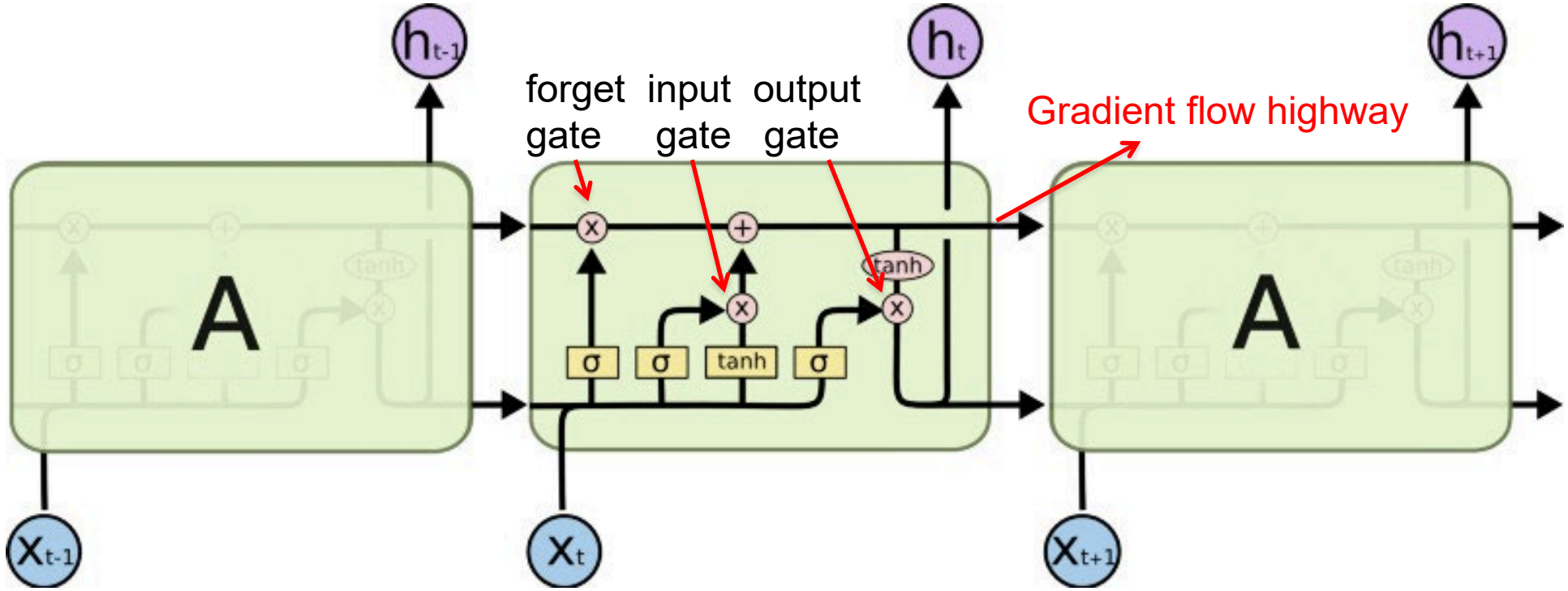
Bidirectional RNN

- When conditioning on a full input sequence, traversing from both directions can enhance the temporal dependency

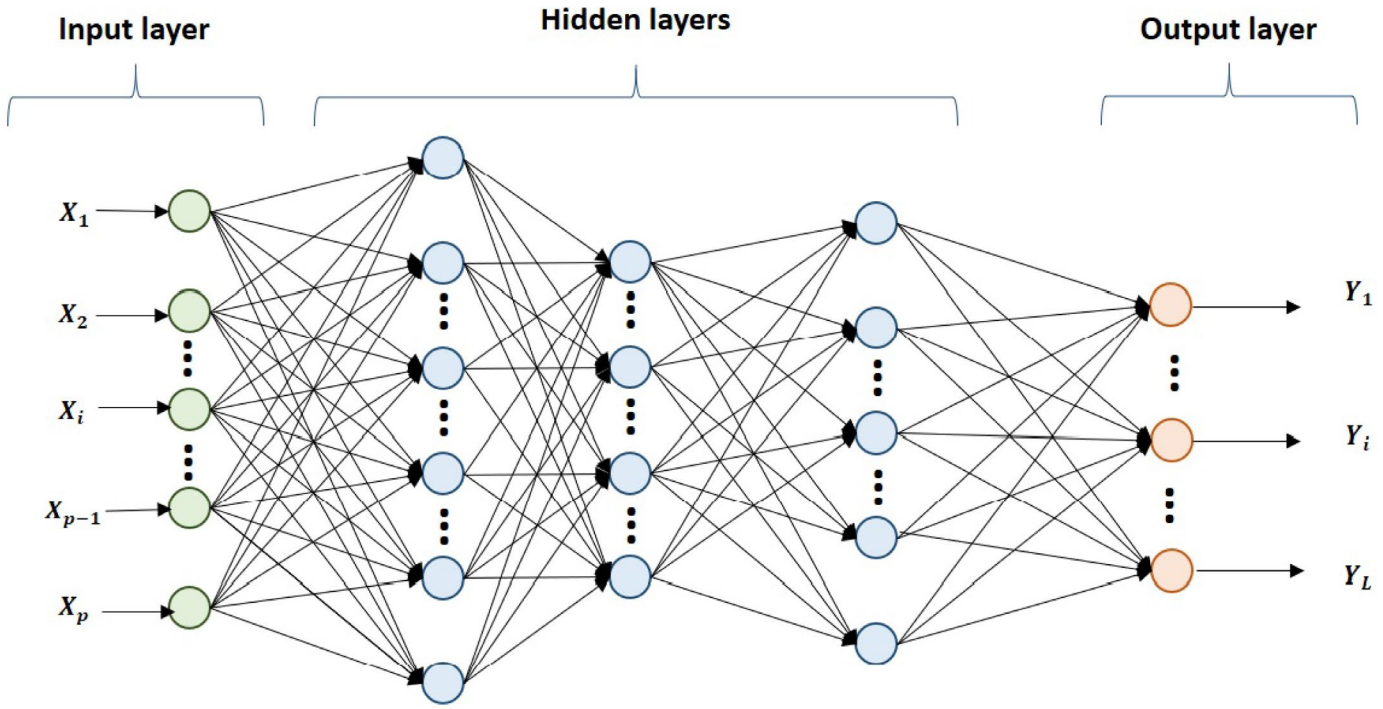


Long Short-Term Memory (LSTM)

- **LSTM** is a type of recurrent neural network (RNN) aimed at mitigating the **vanishing gradient problem** commonly encountered by traditional RNNs.
- Use special hidden units called the memory cells



What if MLP for sequence modeling?

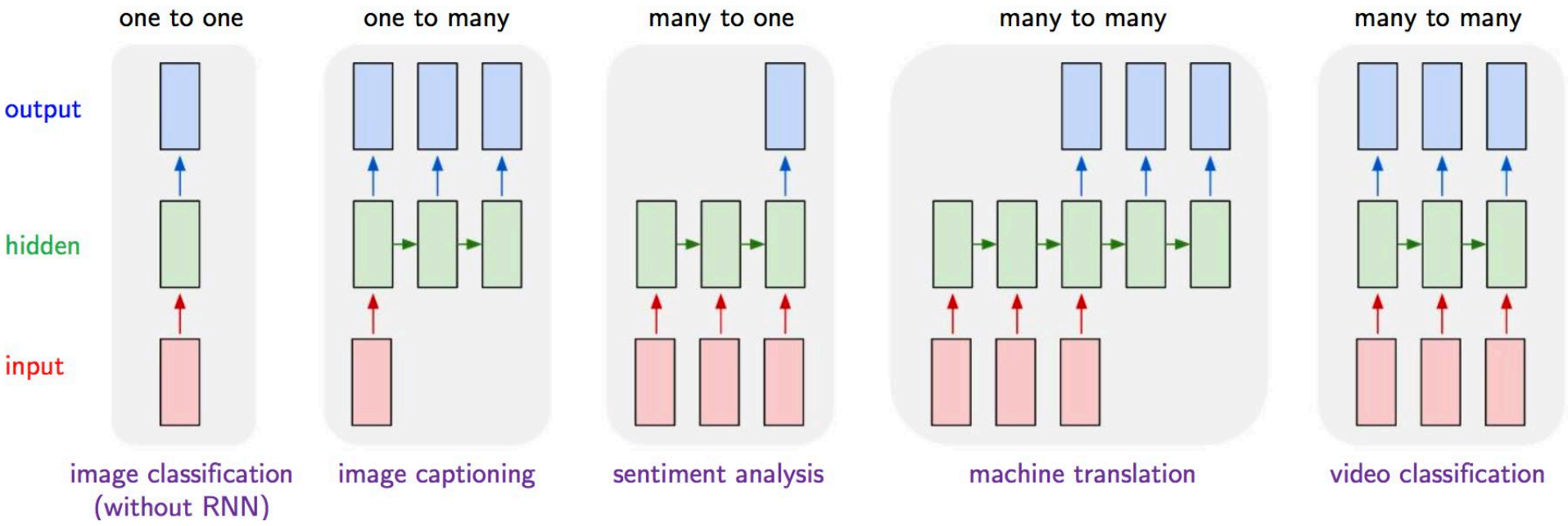


- Fixed input length.
- If input length is very large and growing, this network would become too large.
- The **order** of sequence data is discarded.

Key idea of RNN: input one xi at a time, and re-use the same edge weights.

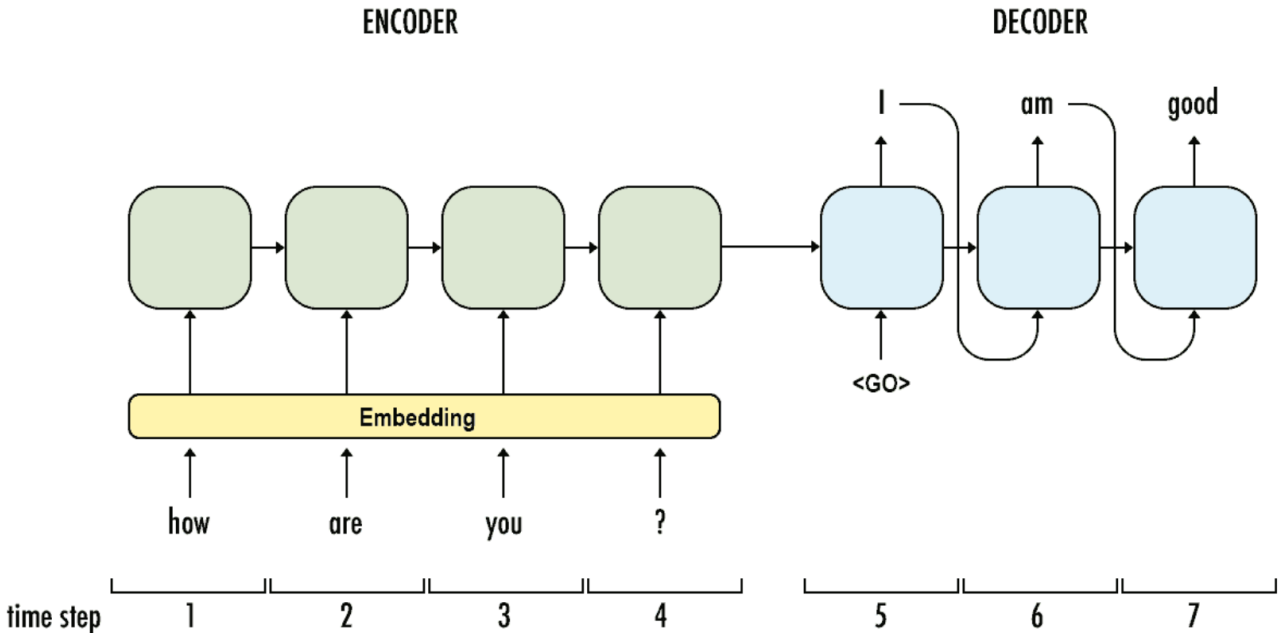
Sequence modeling

- **Sequential data**: language, time series, video, biological sequences…
- The key to sequence modeling is capturing the **sequential context (order important)**
- Various sequencing modeling tasks

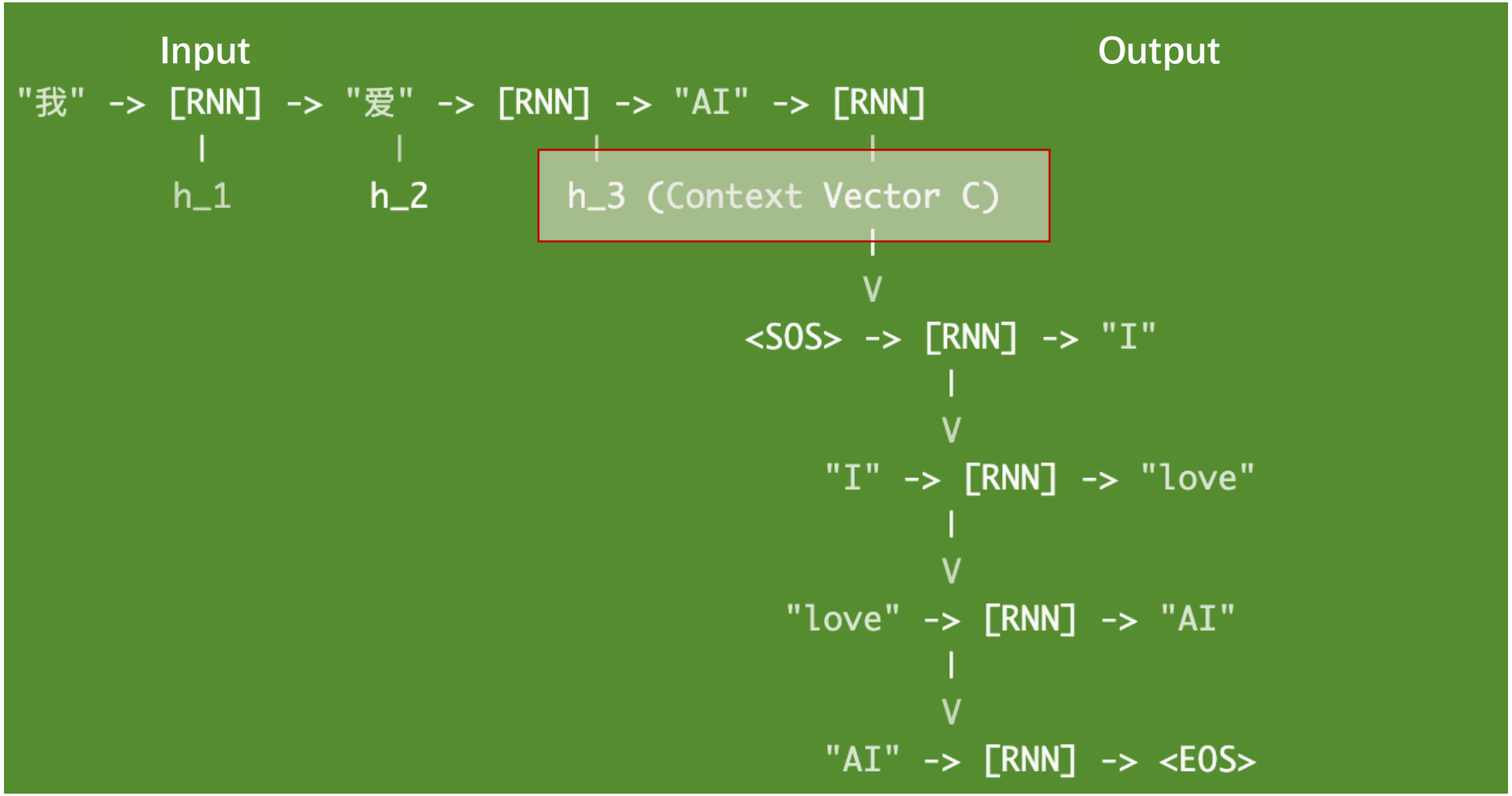


Encoder-decoder model

- When applying RNN/LSTM on NLP tasks such as machine translation, we leverage a structure called an **encoder-decoder model** (or **sequence-to-sequence model**).
- The **encoder** encodes the input sequence into a fixed-length representation, also known as a context vector or latent space representation.
- The **decoder** takes over and generates an output sequence based on the encoded representation.



Encoder-decoder model



Applications

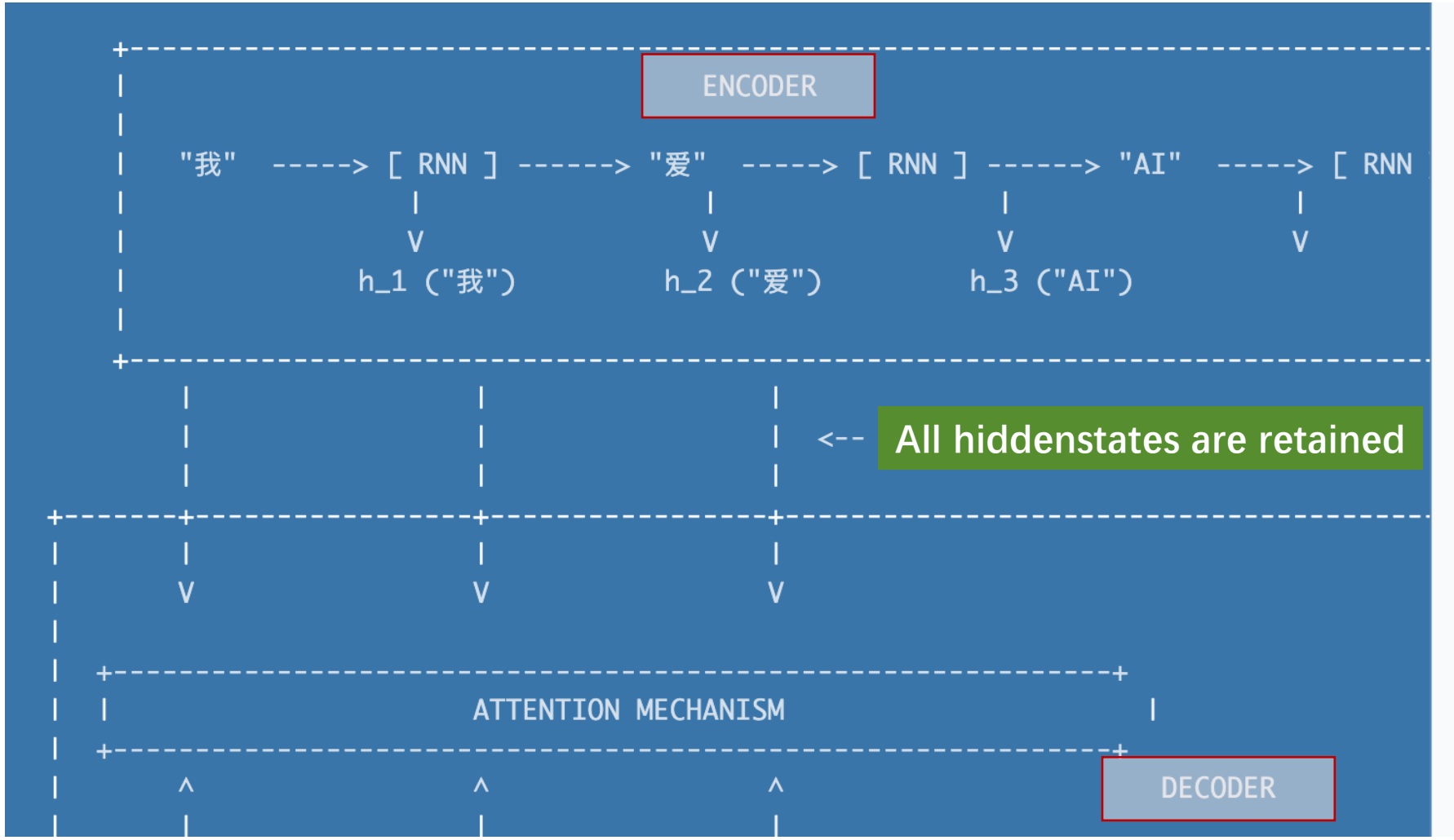
- Machine Translation
  - Input: Hello World
  - Output: 你好世界
- Text Summarization
  - Input: An article, a paragraph
  - Output: A summary sentence
- Chatbots
  - Input: How’s the weather
  - Output: It’s chilly!
- Speech Recognition
  - Input: an audio
  - Output: text

Attention mechanism

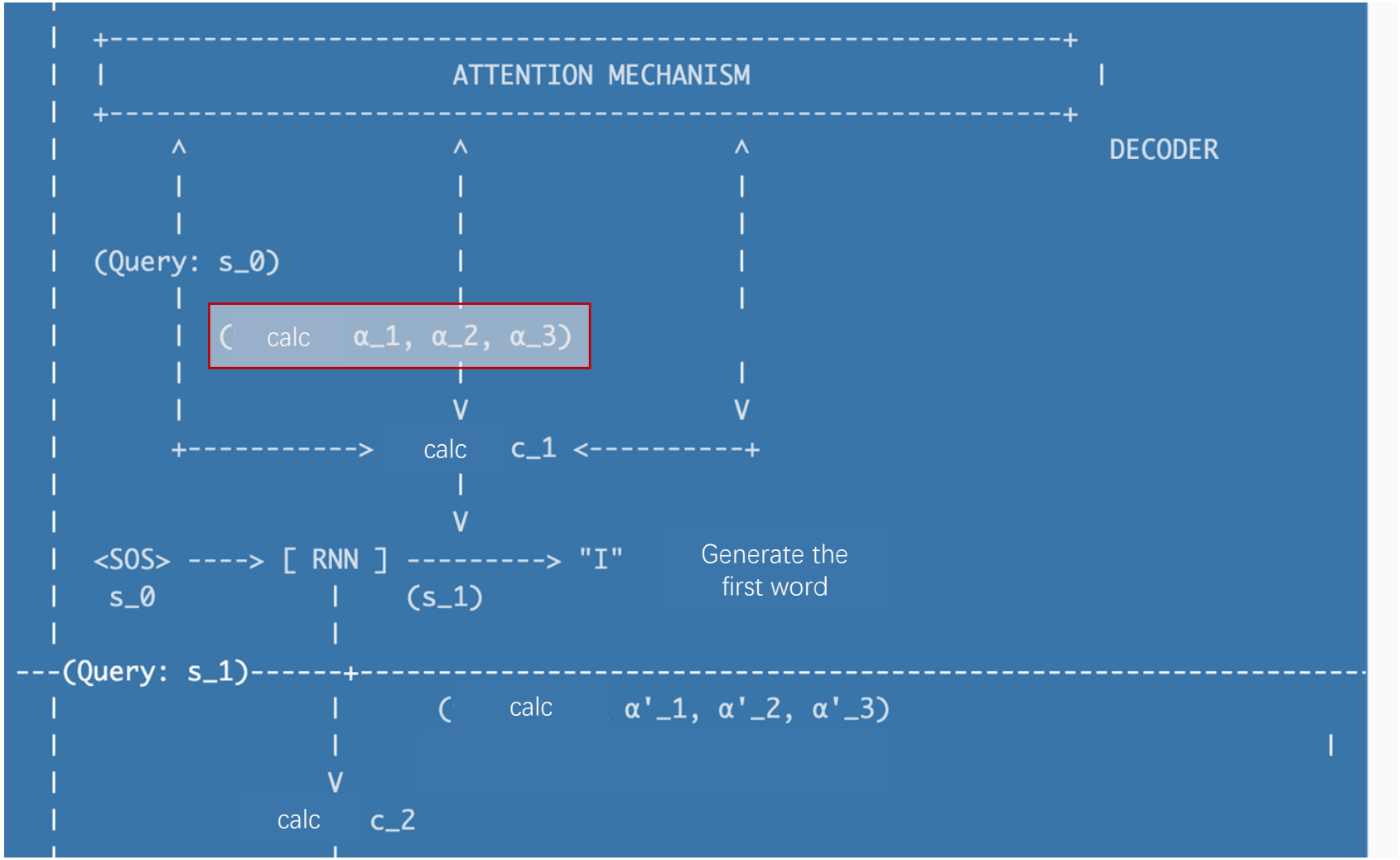
- The bottolneck of the conventional seq-to-seq model is the context vector
- Solution
  - Leverage all hidden states
  - Combine the hidden states into the context vector when decoding
    - How? **Attention!**

      Look backward and calculate a weight

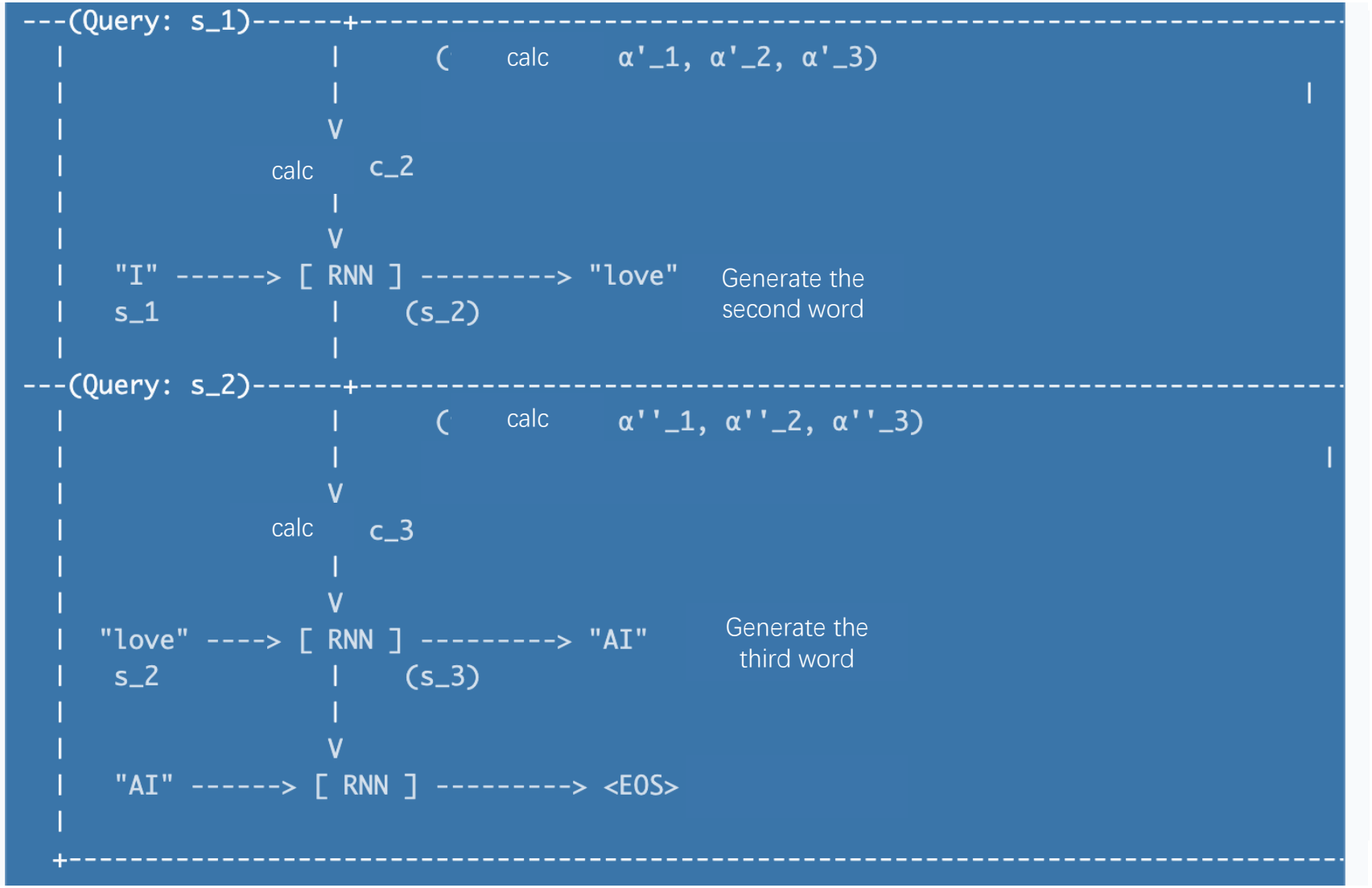
Attention mechanism



Attention mechanism



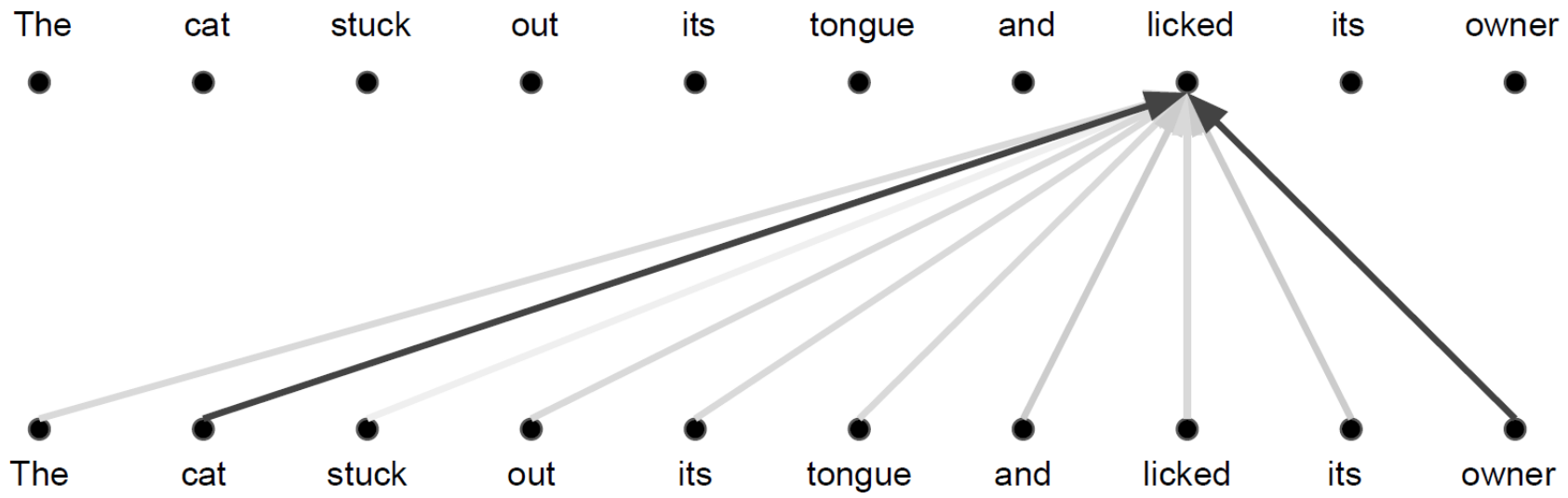
Attention mechanism



Attention is all you need

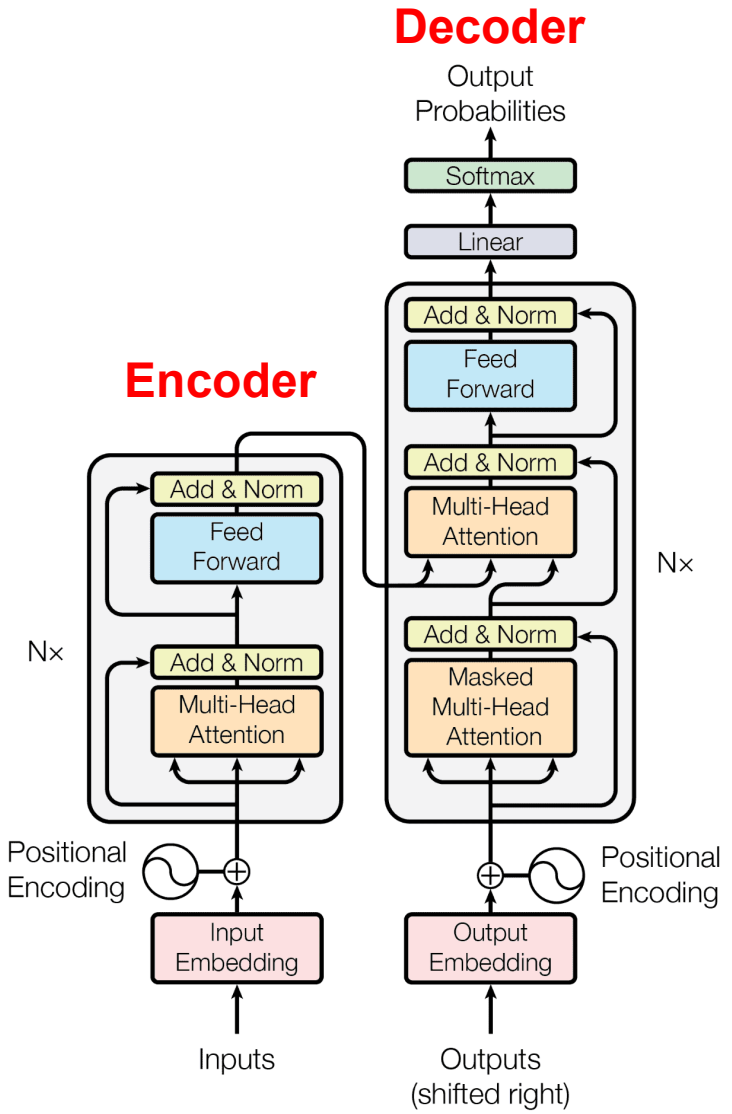


- RNN + attention
- But with attention, we don't need RNN!
- $Attention(Q, K, V) = Softmax(\frac{QK^\top}{\sqrt{d_k}})V$
  - Query $\mathrm{Q} = [q_1, \ldots, q_n]$, key $\mathrm{K} = [k_1, \ldots, k_n]$, value $\mathrm{V} = [v_1, \ldots, v_n]$
  - Compute weights with each others in the whole sequence



From RNN to transformer

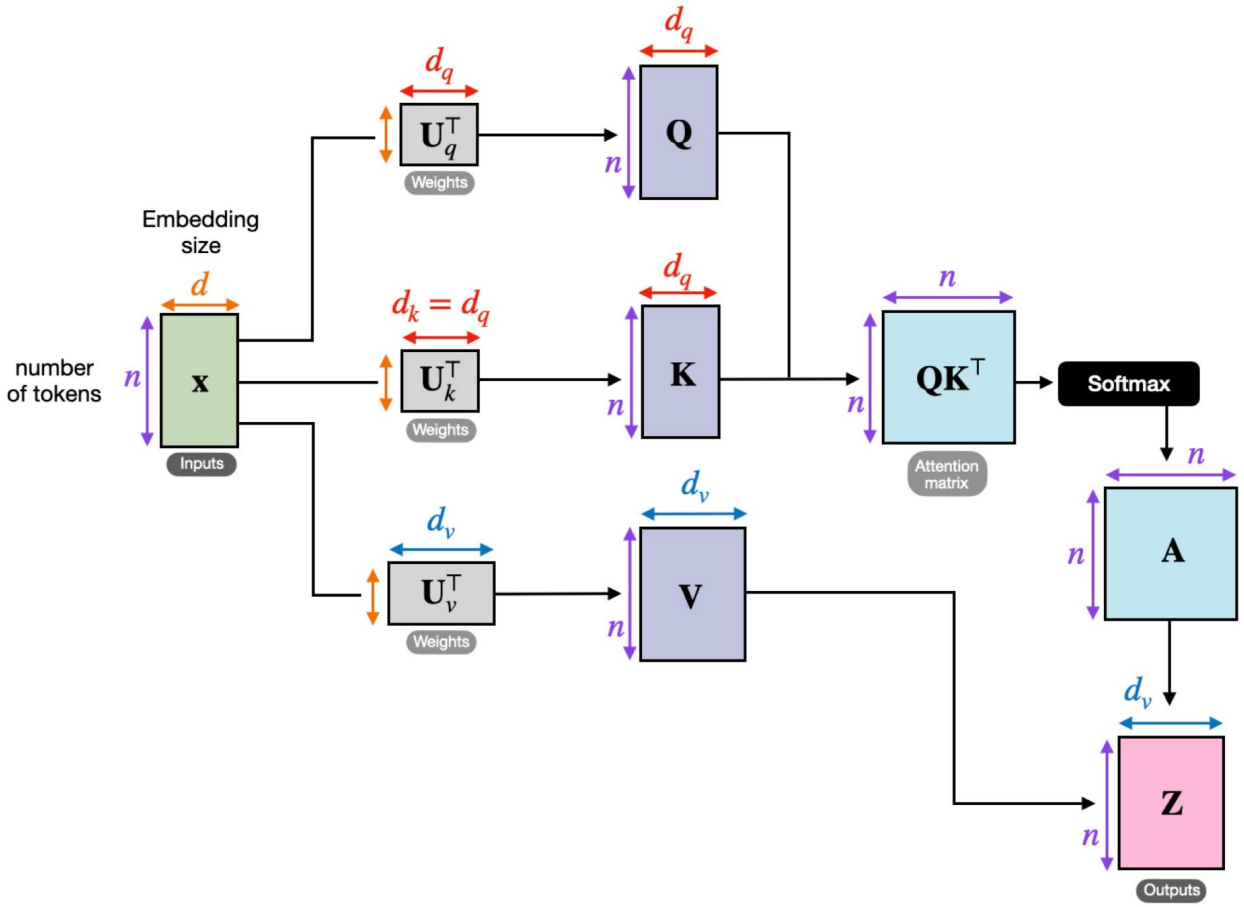
- Disadvantages of RNN
  - ➢ Sequential computation of RNN inhibits parallelization.
  - ➢ Not every piece of the input sequence's context is required at every decoding stage for all text production activities.
- Unlike RNNs, **Transformers** do not rely on recurrence but instead operate on **self-attention**.

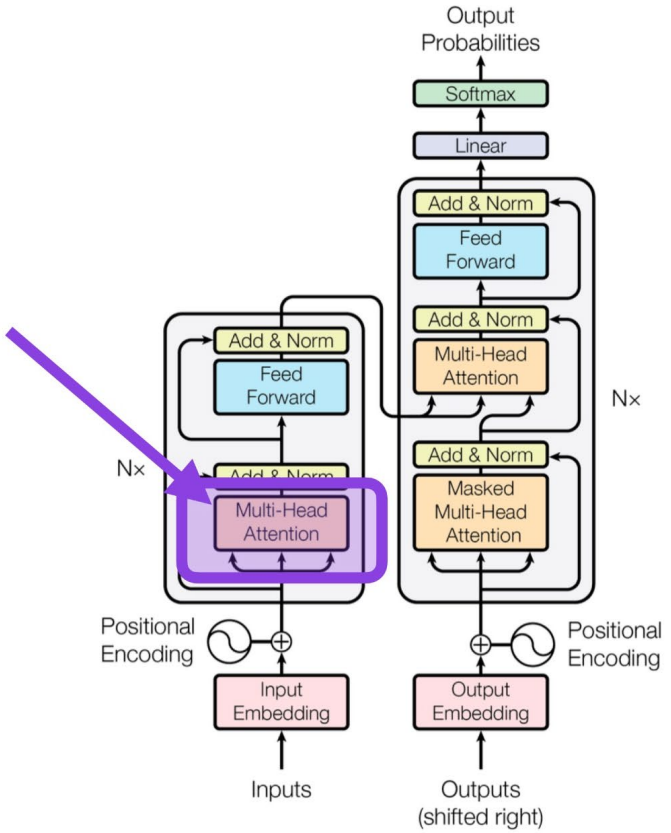


more reading: https://jalammar.github.io/illustrated-transformer/

Self-Attention in Transformer

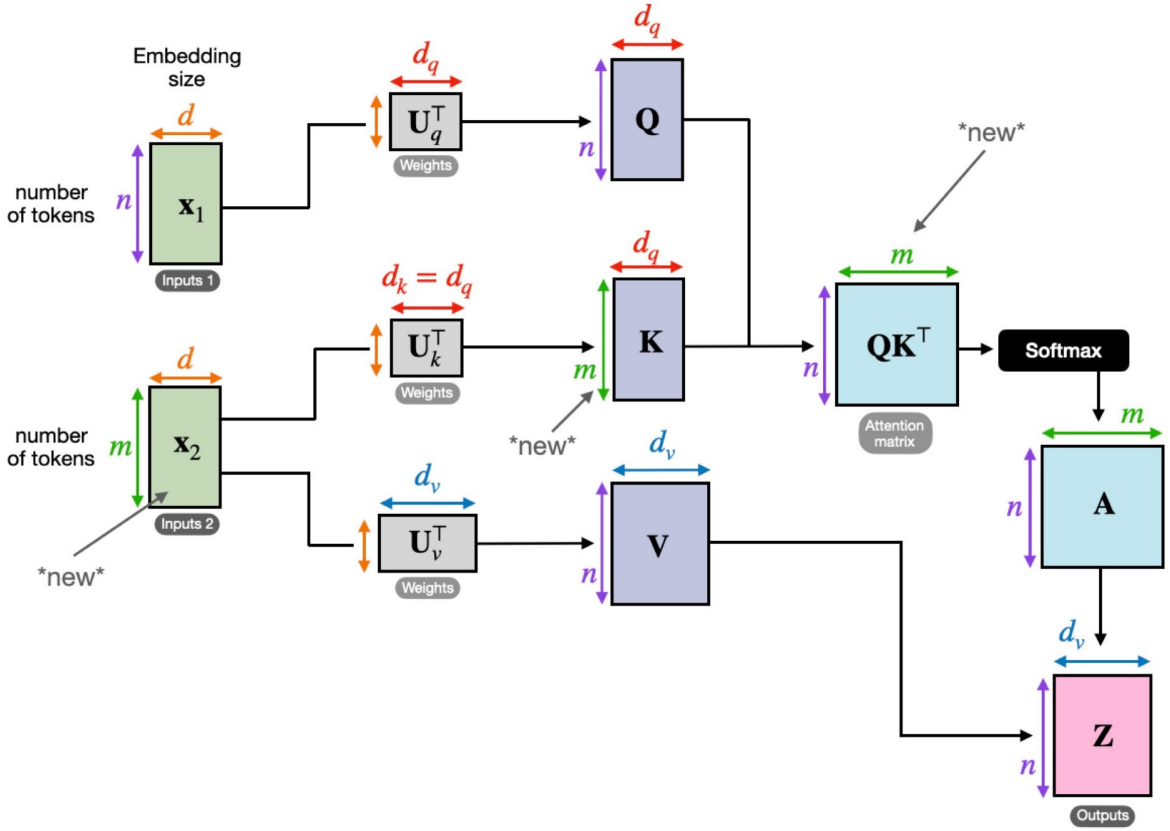
- Q, K, V are from the same sequence

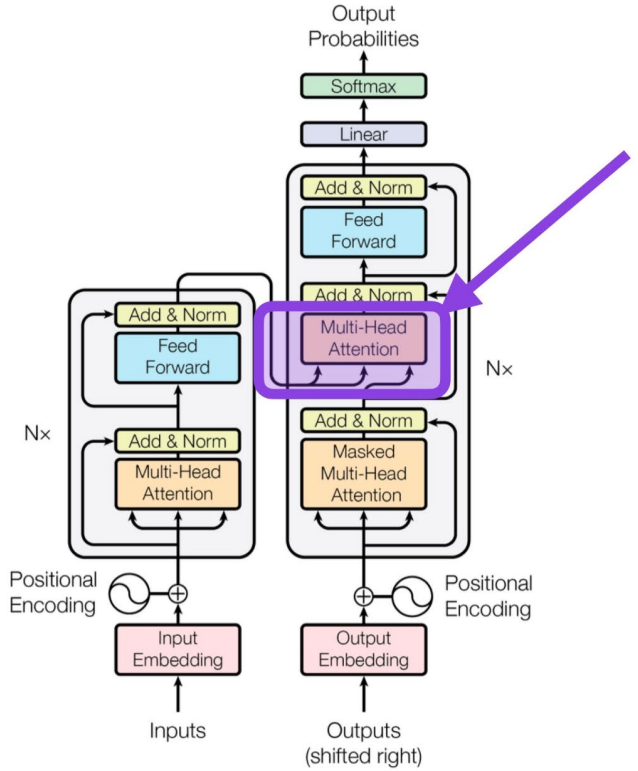




Cross-Attention in Transformer

- K, V are from the decoder outputs

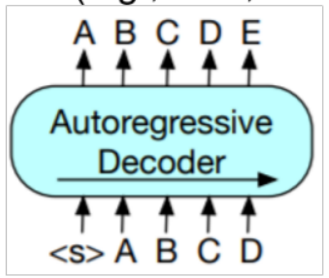
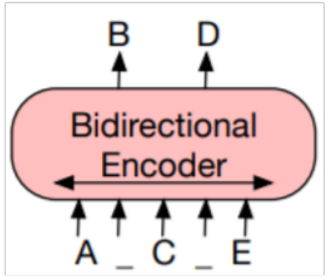
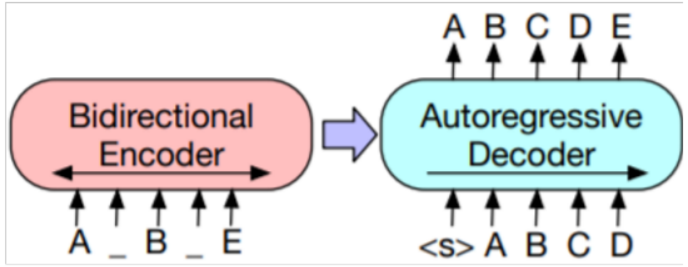
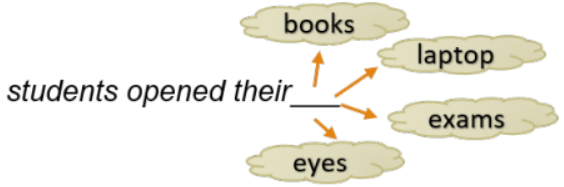
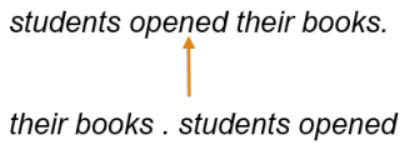




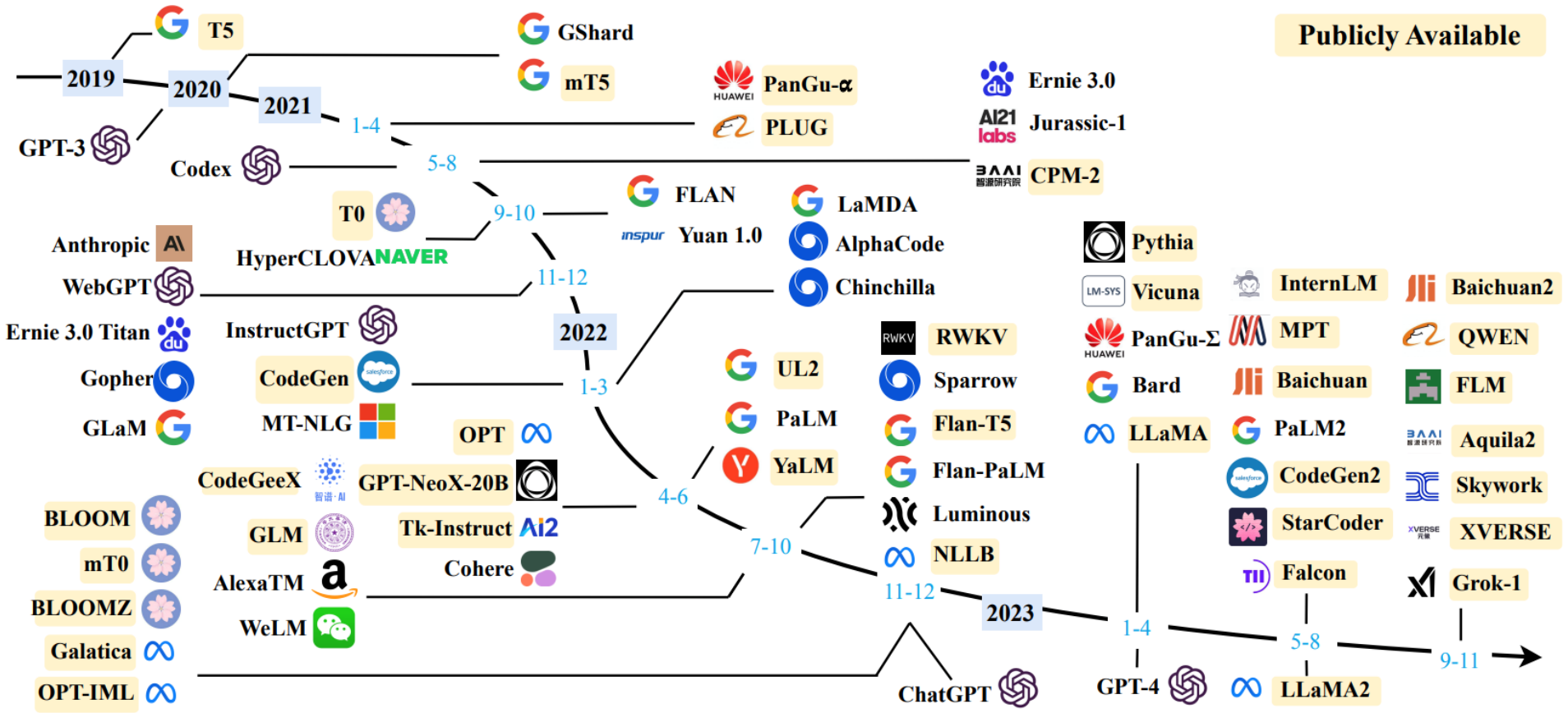
RNN vs Transformer

- RNN
  - Strength: Powerful at modeling sequences (Turing complete).
  - Drawback: Their sequential nature makes it quite slow to train.
  - Fail on large-scale language understanding tasks like translation.
- Transformer
  - Strength: Process in parallel thus much faster to train.
  - Drawback: Fails on smaller and more structured language understanding tasks, or even simple algorithmic tasks such as copying a string while RNNs perform well.
  - Requires a lot of memory: $n \times m$ alignment and attention scalers need to be calculated and stored for a single self-attention head.

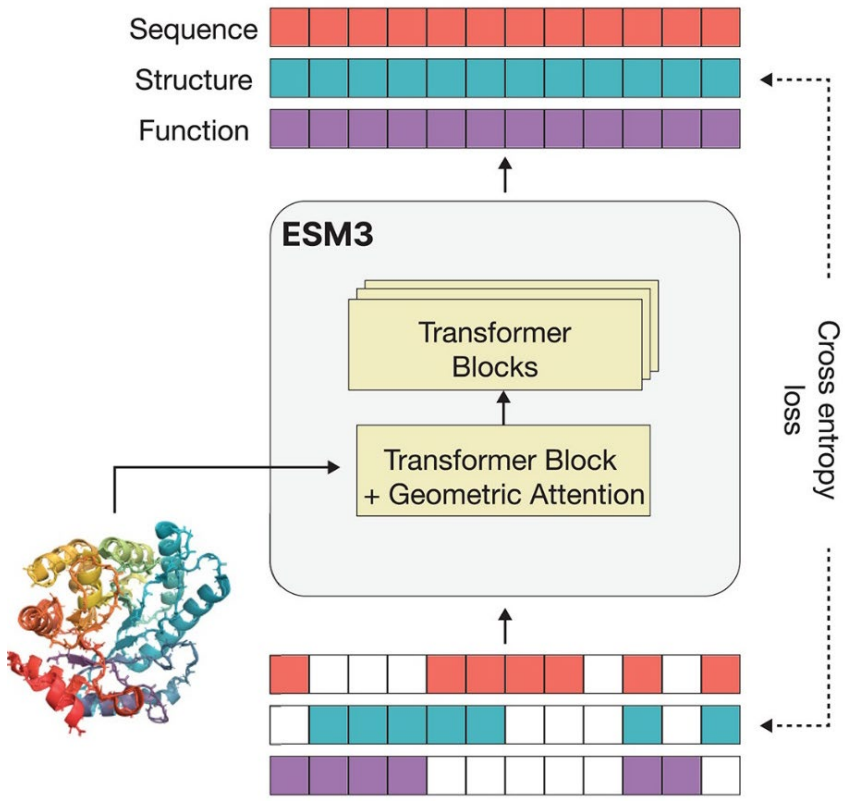
Large language models: self-supervision

	Autoregressive language model (e.g., GPT, GPT-2/3)	**Masked** language model (e.g., BERT, RoBERTa, XLM-R)	**Encoder-Decoder** (e.g., BART, T5)
Model & illustration			
Training objective	Predicting what word comes next given previous words	Predicting masked words given other words in the sequence	Corrupting a sequence and then predicting the original sequence
Example			

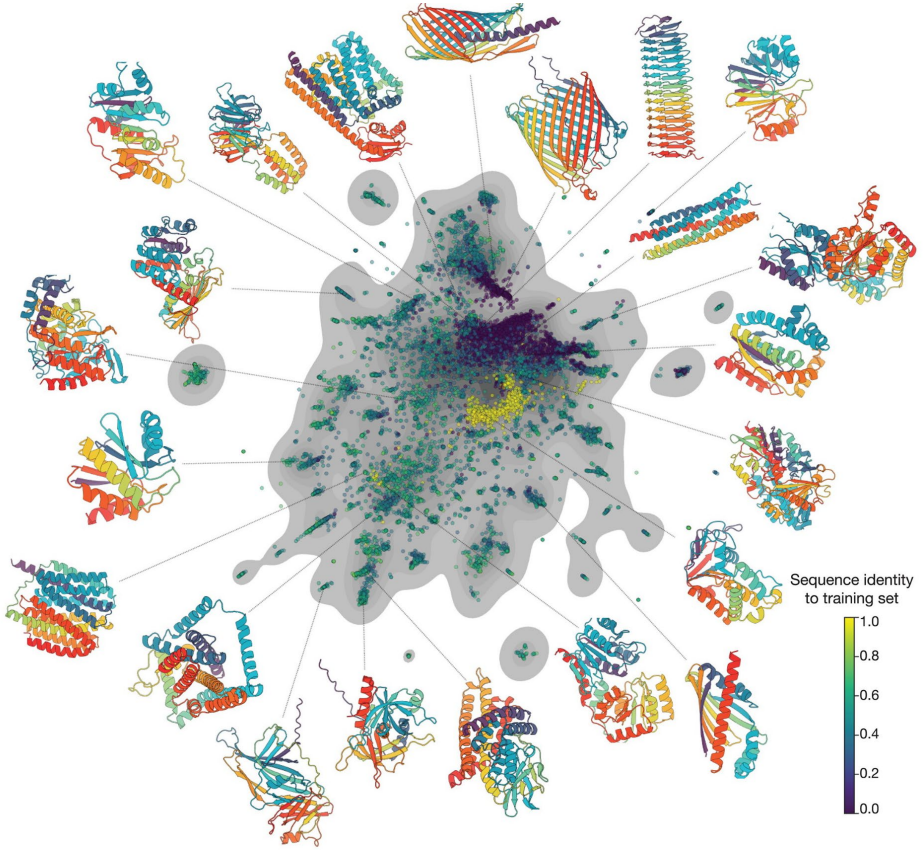
Large language models for natural language



Example: generative language model ESM3 reasons over the sequence, structure, and function of proteins.



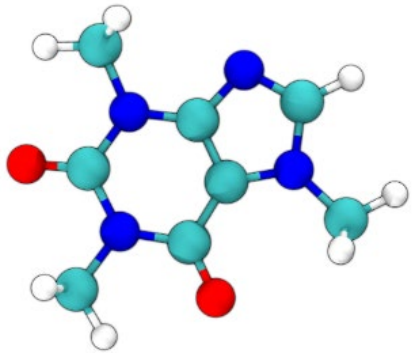
ESM3 architecture



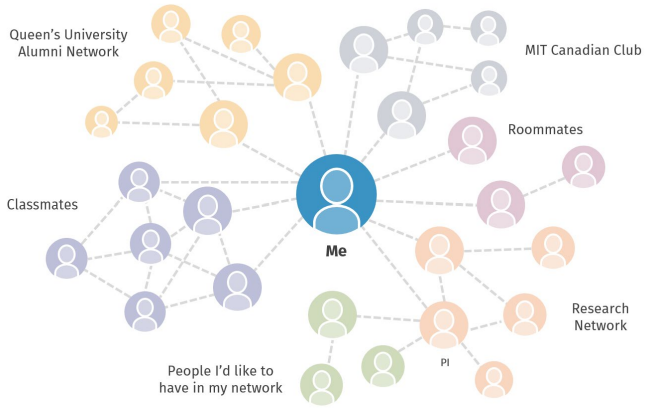
Unconditional generated sequences by ESM3 are diverse, high-quality, and cover the distribution of natural sequences. (*Science*, 2025)

Graphs are ubiquitous

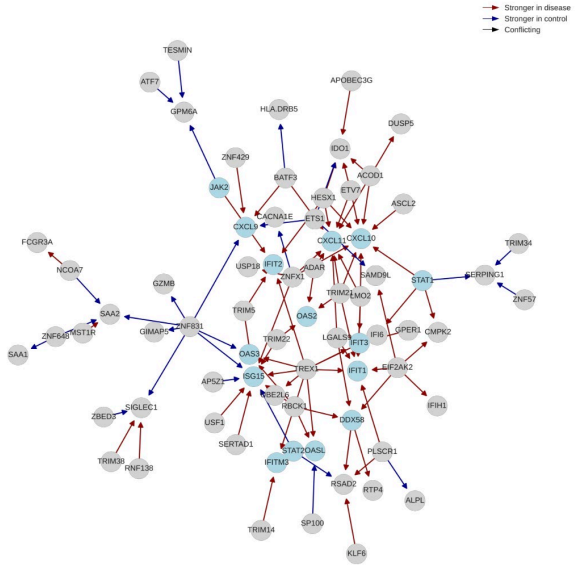
- A **graph** represents the relations (**edges**) between a collection of entities (**nodes**)



molecule



social network

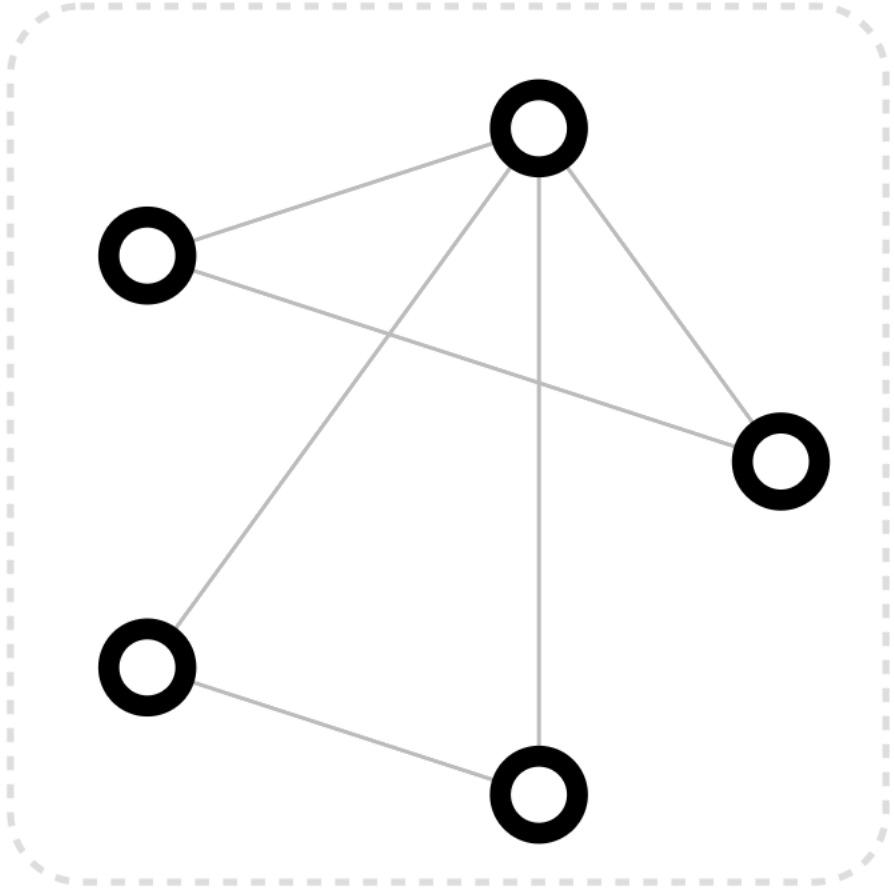


biomedical network



Internet

Three types of attributes in a graph



V Vertex (or node) attributes

e.g., node identity, number of neighbors

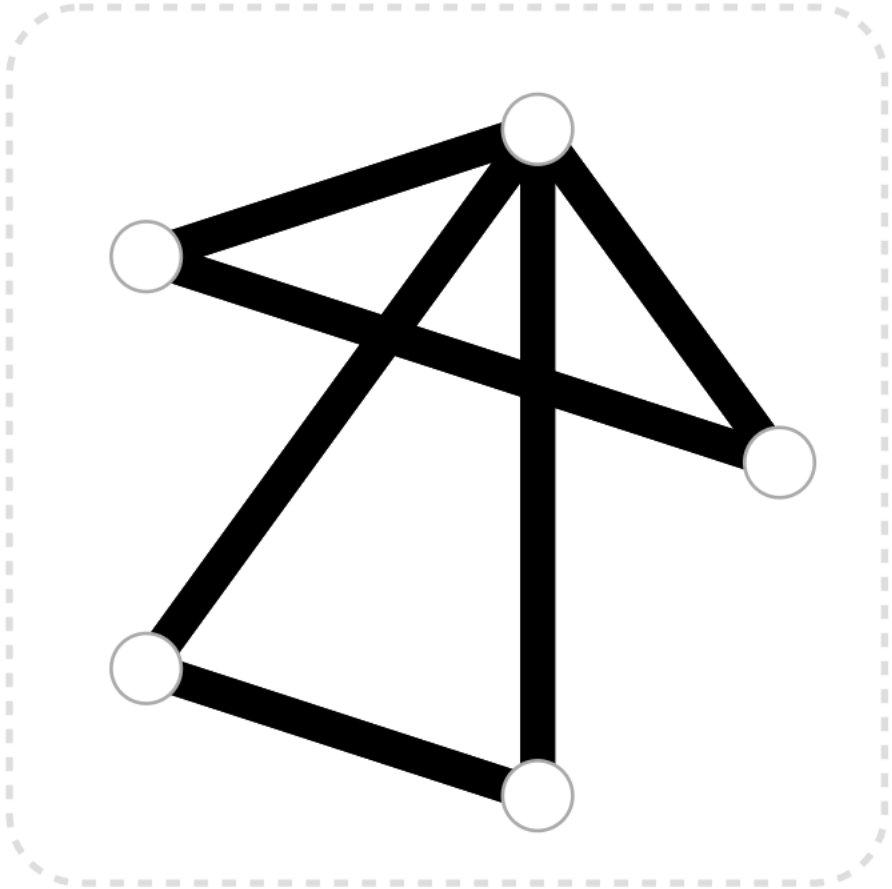
E Edge (or link) attributes and directions

e.g., edge identity, edge weight

U Global (or master node) attributes

e.g., number of nodes, longest path

Three types of attributes in a graph



V Vertex (or node) attributes

e.g., node identity, number of neighbors

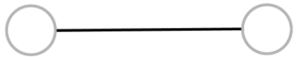
E Edge (or link) attributes and directions

e.g., edge identity, edge weight

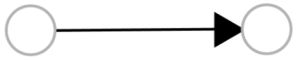
U Global (or master node) attributes

e.g., number of nodes, longest path

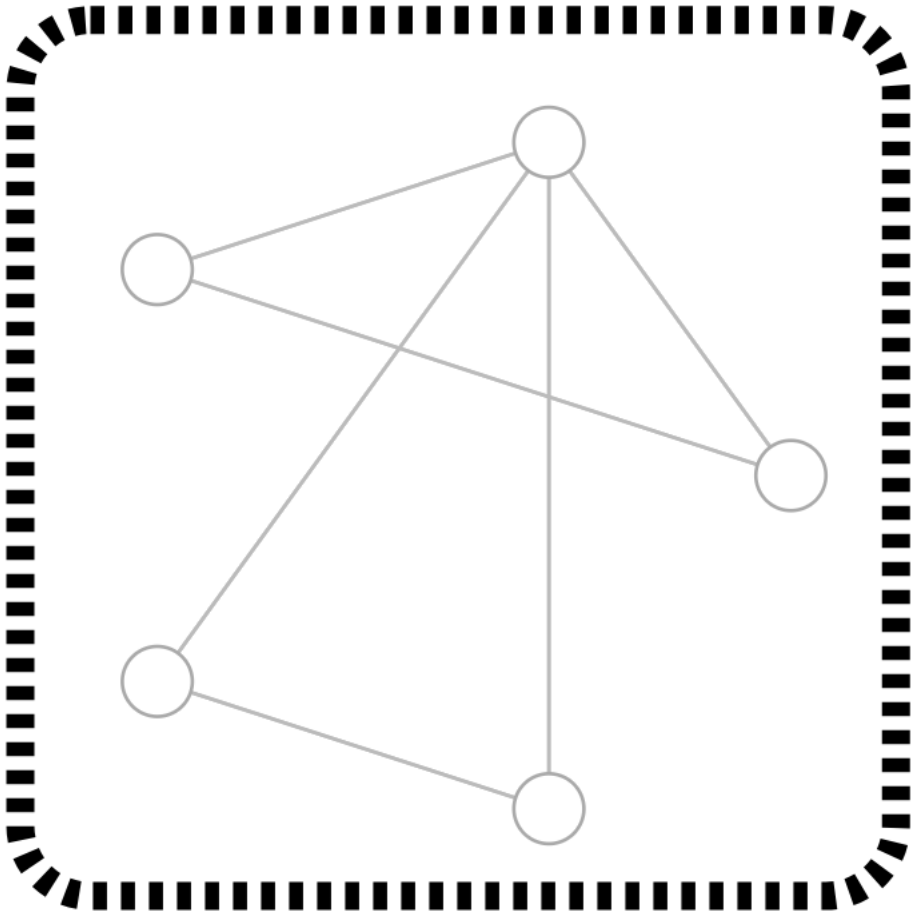
Undirected edge



Directed edge



Three types of attributes in a graph



V Vertex (or node) attributes

e.g., node identity, number of neighbors

E Edge (or link) attributes and directions

e.g., edge identity, edge weight

U Global (or master node) attributes

e.g., number of nodes, longest path

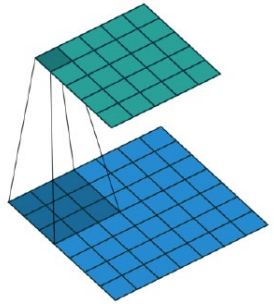
More reading: https://distill.pub/2021/understanding-gnns/

Deep learning on graph

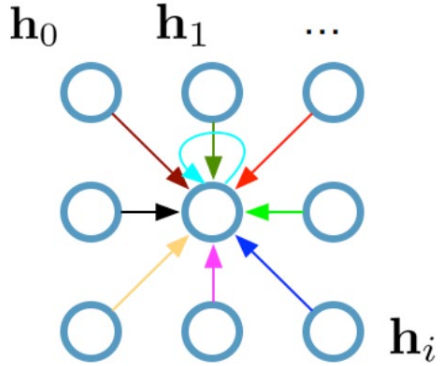
- Tasks for graph structured data
  - Node-level task: e.g. member classification
  - Edge-level task: e.g. relationship classification
  - Graph-level task: e.g. molecule classification
- Existing toolbox is designed for simple sequence or grids
  - CNN for fixed-size images/grids
  - RNN for text/sequences
- Graphs are more complex
  - Arbitrary size and complex topographical structure
  - No fixed node ordering or reference point
  - Often dynamic and have multimodal features

From image to graph

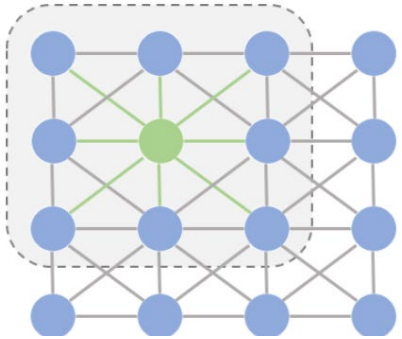
- Idea: transform information at the **neighbors** and combine it
  - Transform messages h_i from neighbors: $W_i h_i$
  - Add them up: $\sum_{i \in Neighborhood} W_i h_i$
- Graphs are **permutation invariant**



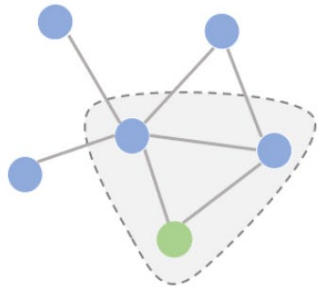
image



graph



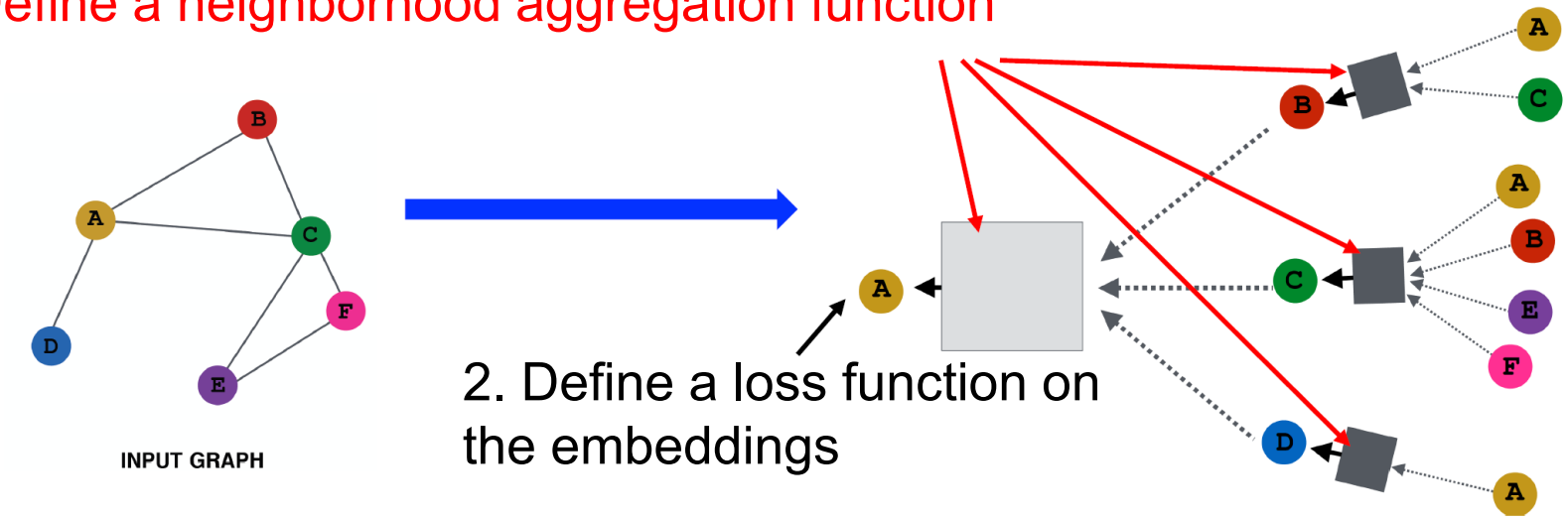
Lattices structure



No fixed notion of locality or sliding window

GNN overview

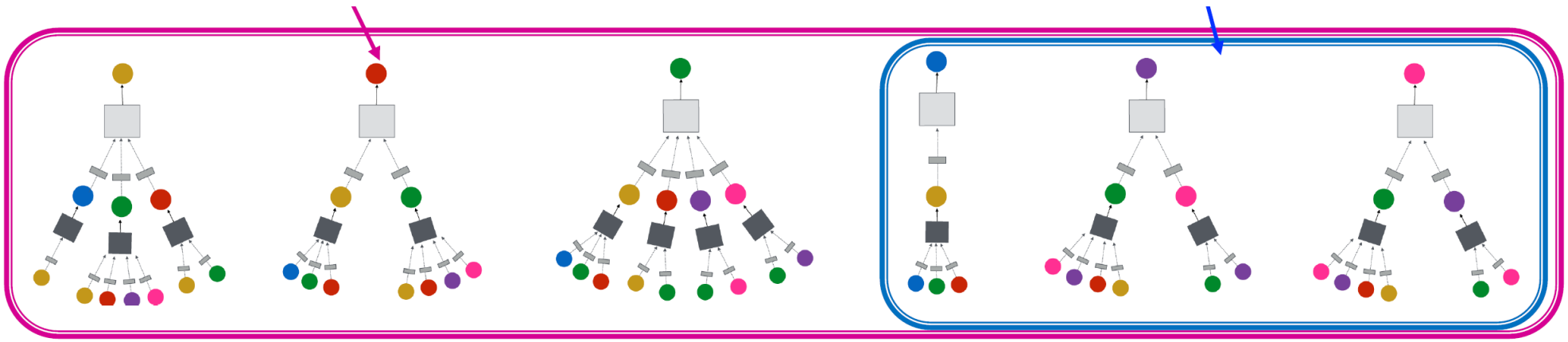
1. Define a neighborhood aggregation function



2. Define a loss function on the embeddings

3. Train on a set of nodes

4. Generate embeddings for nodes needed



GNN layer

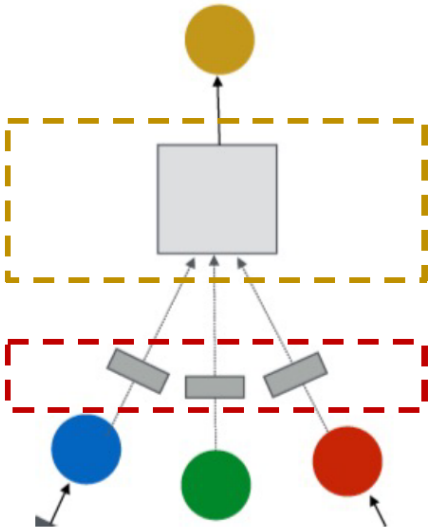
- Message computation: Each node will create a message, which will be sent to other nodes $m_u^l = MSG^l(h_u^{l-1})$
- Aggregation: Each node will aggregate messages from its neighbors $h_v^l = AGG^l(\{m_u^l, u \in N(v)\})$
- Classical GNN layer: Graph Convolutional Network (GCN)

$$\mathbf{h}_v^l = \sigma\left(\sum_{u \in N(v)} \mathbf{W}^l \frac{\mathbf{h}_u^{l-1}}{|N(v)|}\right)$$

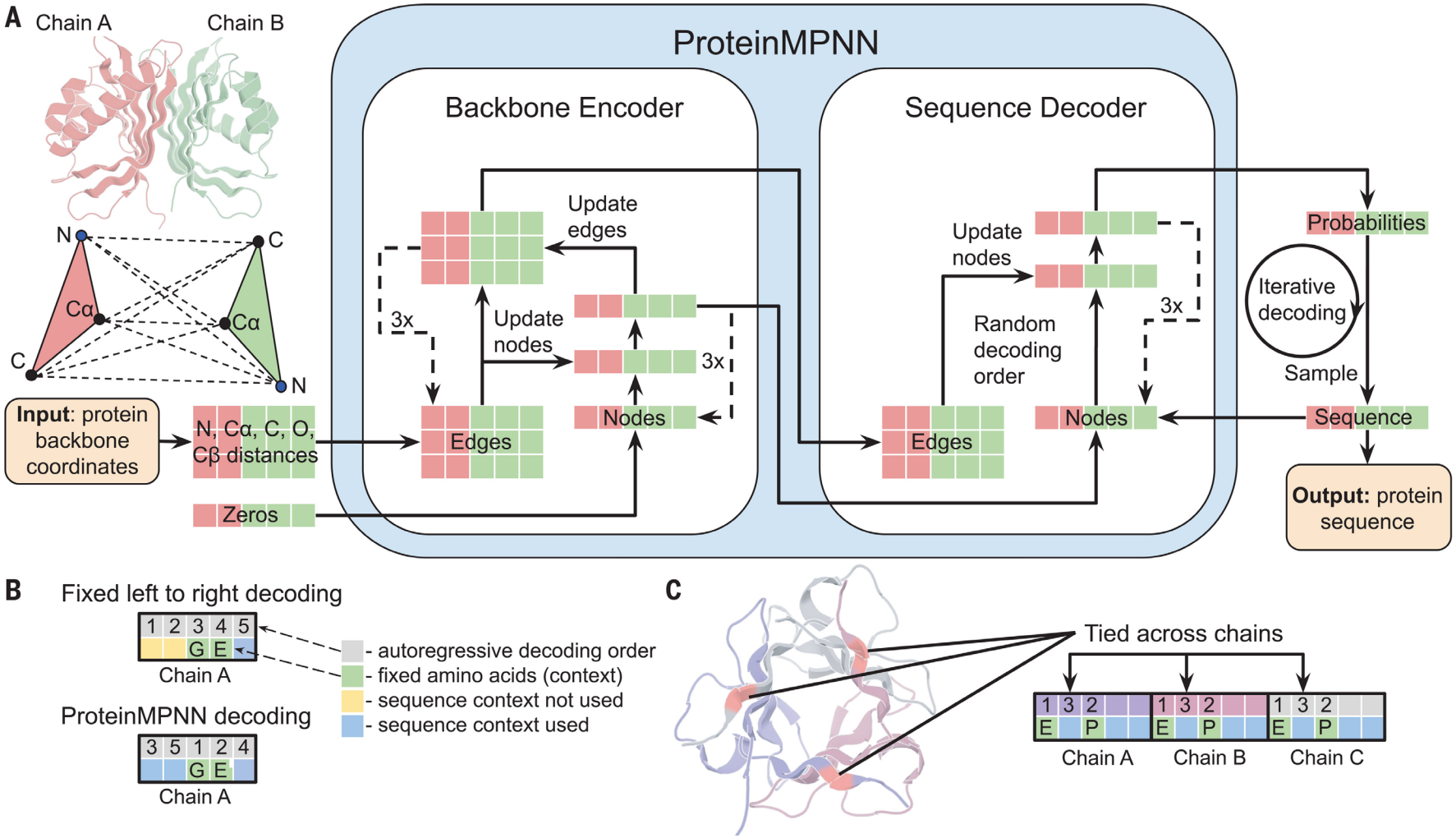
$$\mathbf{h}_v^l = \sigma\left(\text{Sum}\left(\{\mathbf{m}_u^l, u \in N(v)\}\right)\right)$$

② Aggregation

① Message computation



Example: structure-based protein sequence design with ProteinMPNN

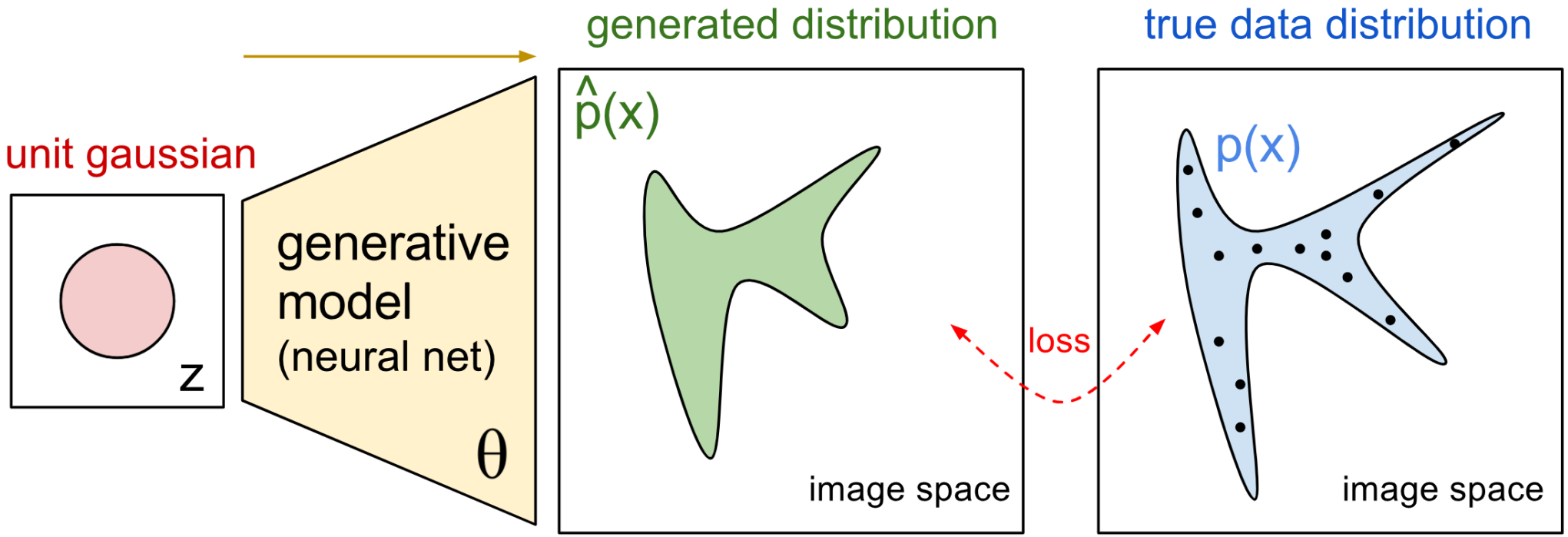


(*Science*, 2022)

Generative modeling

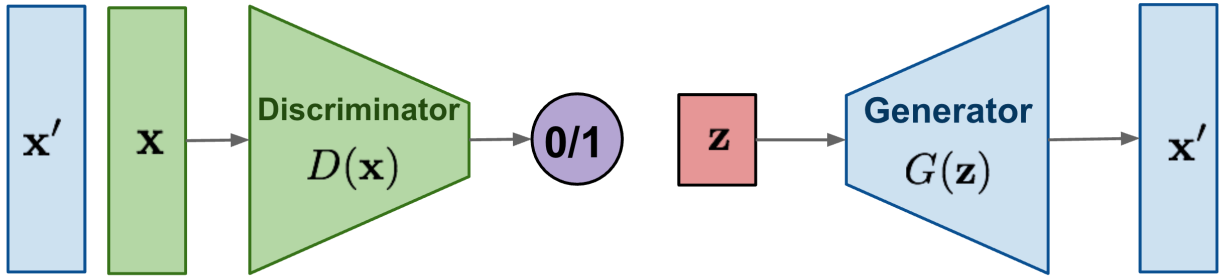
"What I cannot create, I do not understand." — Richard Feynman

- **Generative modeling** are designed to learn the underlying distribution of data and generate new data samples that resemble the original data.

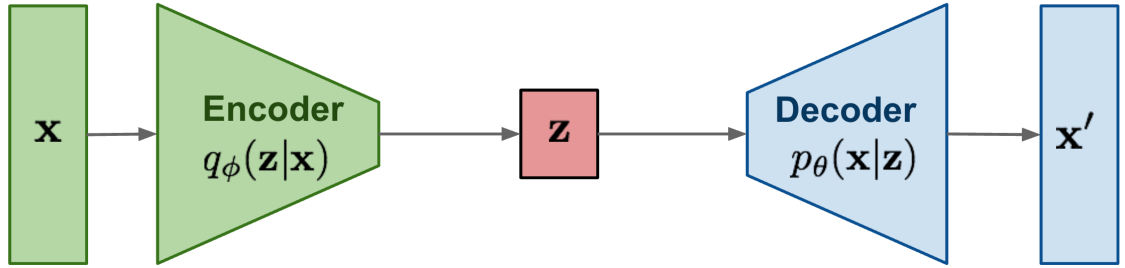


Types of generative models

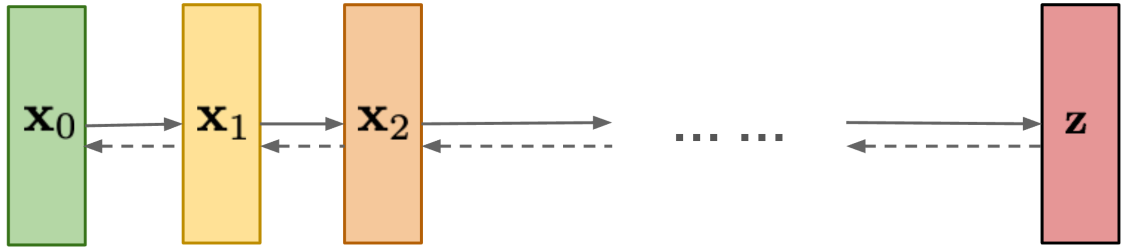
GAN: Adversarial training



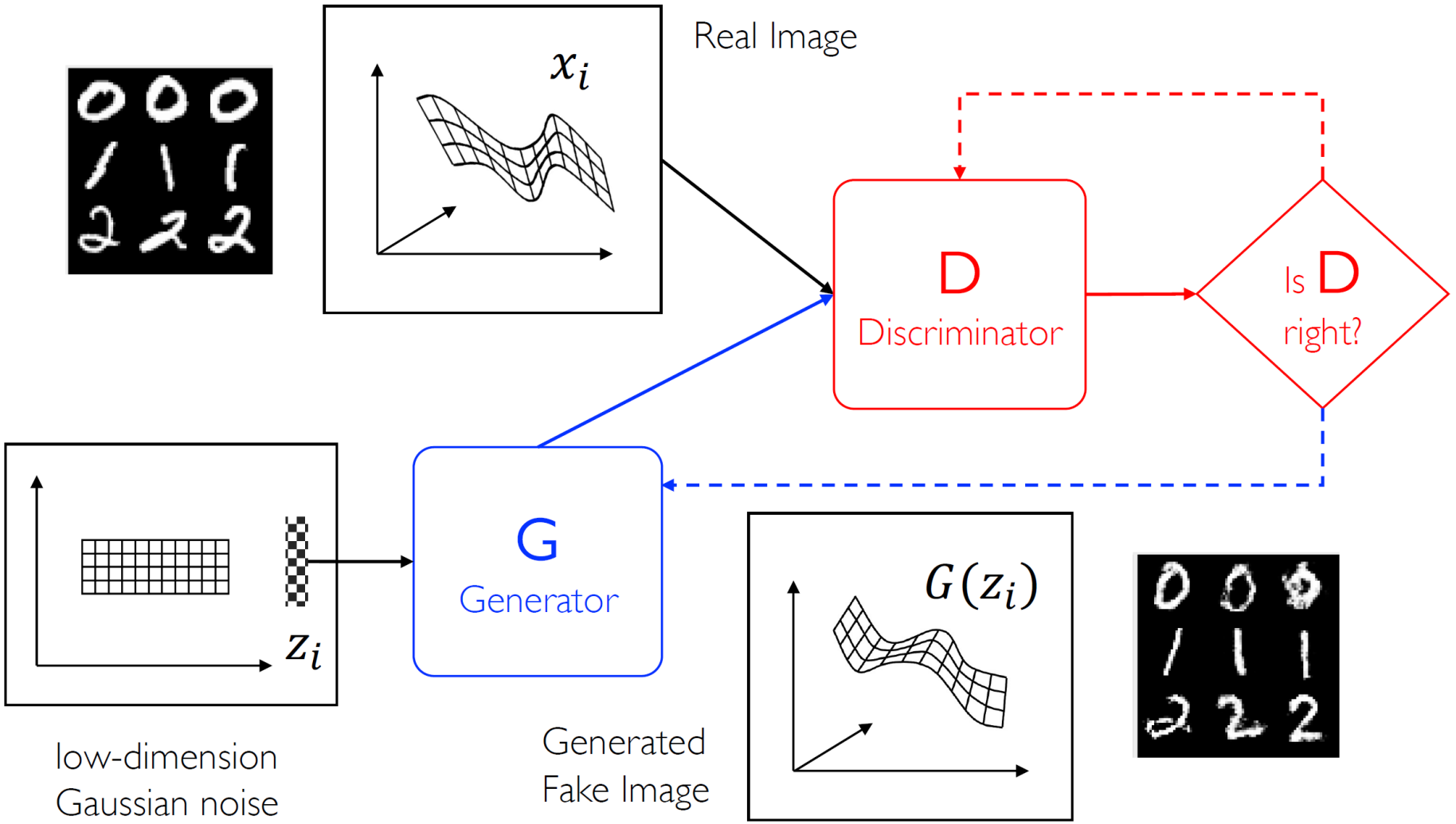
VAE: maximize variational lower bound



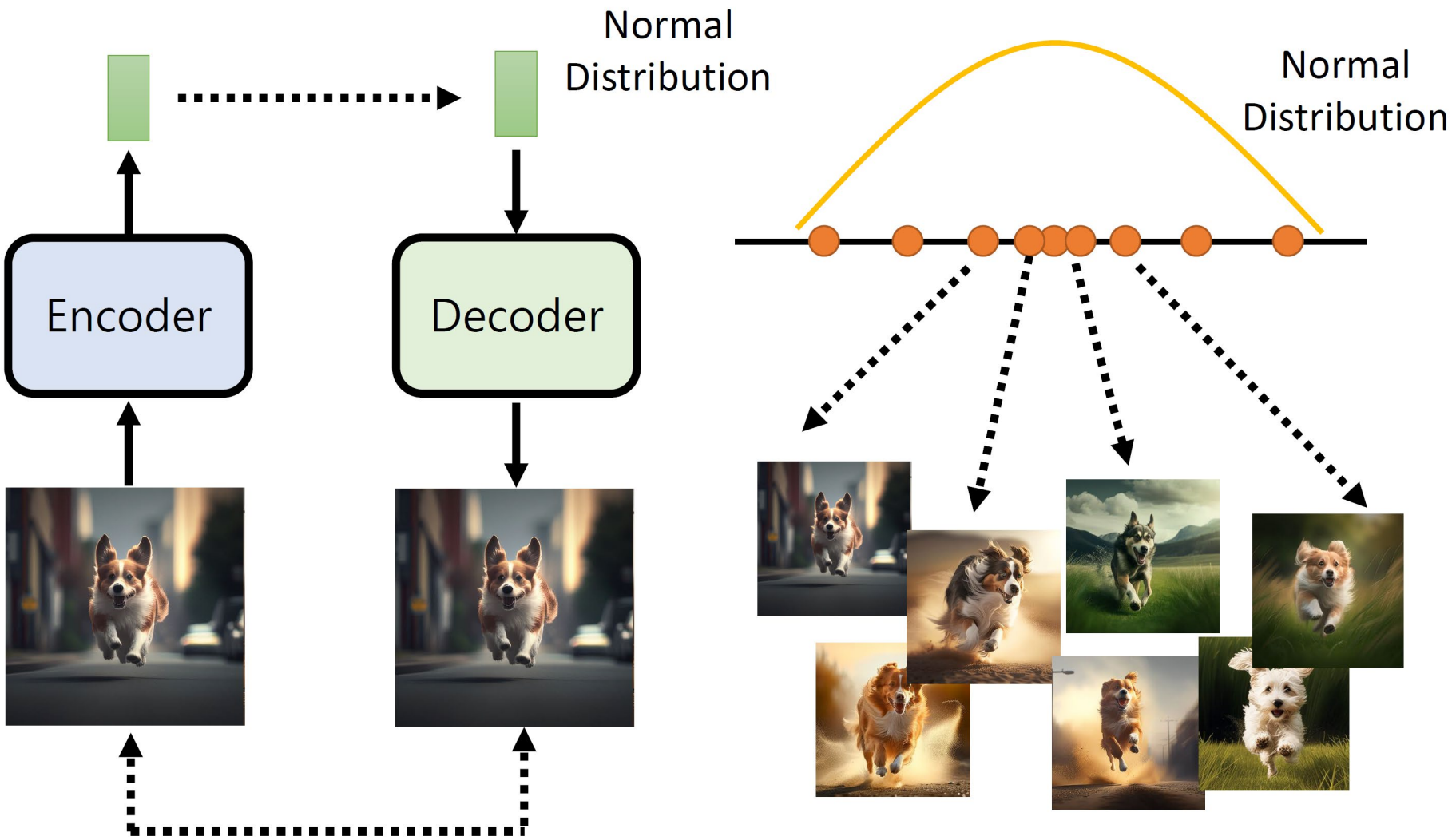
Diffusion models: Gradually add Gaussian noise and then reverse



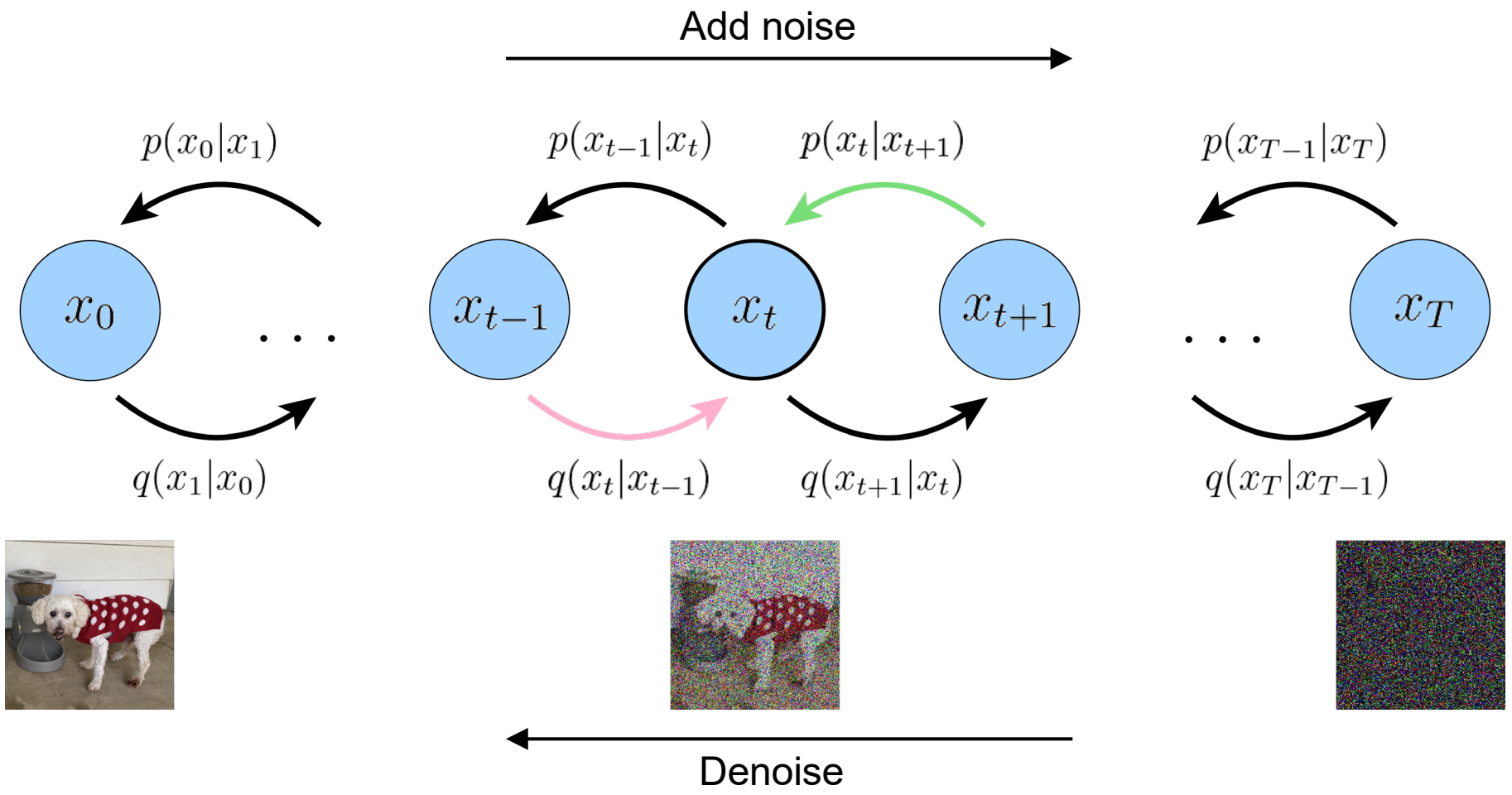
Generative Adversarial Network (GAN)



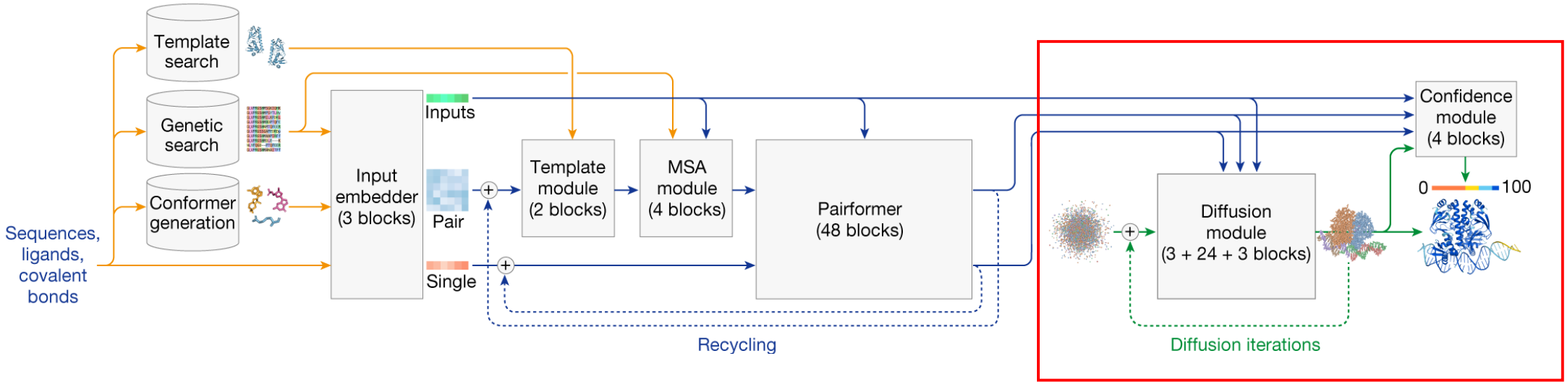
Variational Autoencoder (VAE)



Diffusion models



Example: AlphaFold3 enables accurate structure prediction with a diffusion-based architecture



(*Nature*, 2024)

Q&A