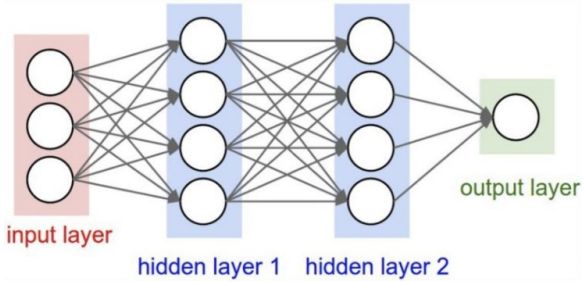
Overview of DNN architectures

TAML 2025

Overview of DNN architectures



Base DNN architectures:

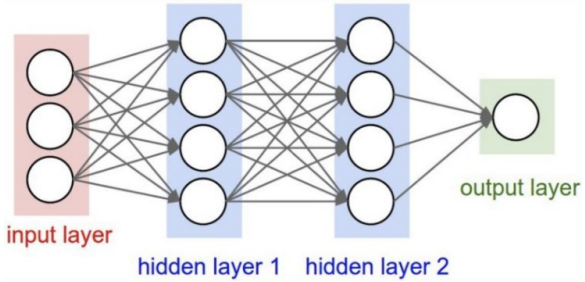
Combined or modified to create more complex architectures:

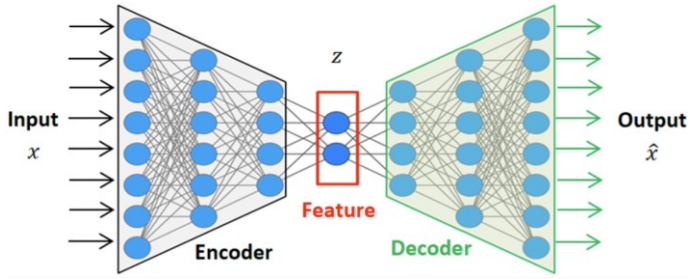
- generative models
- multi-modal models
- reinforcement learning models

Examples:

MLP, CNN, RNN, Transformers, GNN

Overview of DNN architectures





Base DNN architectures:

Combined or modified to create more complex architectures:

- generative models
- multi-modal models
- reinforcement learning models

Examples:

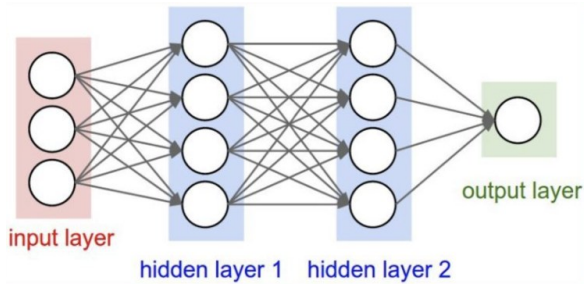
MLP, CNN, RNN, Transformers, GNN

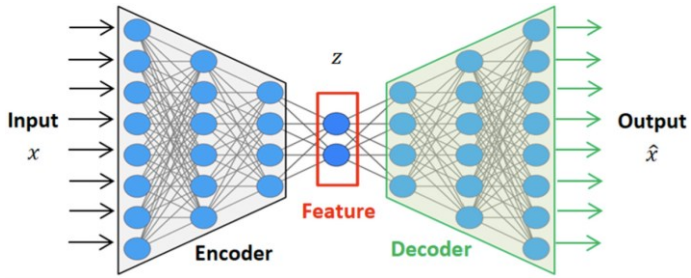
Generative Models:

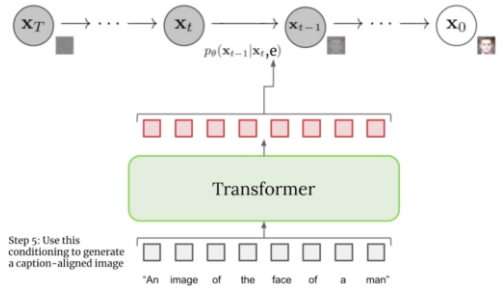
Models that generate new data by learning the distribution of existing data.

- Variational Autoencoders (VAE)
- Generative Adversarial Networks (GAN)
- Normalizing flows
- Diffusion Models

Overview of DNN architectures







Base DNN architectures:

Combined or modified to create more complex architectures:

- generative models
- multi-modal models
- reinforcement learning models

Examples:

MLP, CNN, RNN, Transformers, GNN

Generative Models:

Models that generate new data by learning the distribution of existing data.

- Variational Autoencoders (VAE)
- Generative Adversarial Networks (GAN)
- Normalizing flows
- Diffusion Models

Multi-Modal Models:

Models that combine data from multiple modalities (e.g., text, images, audio)

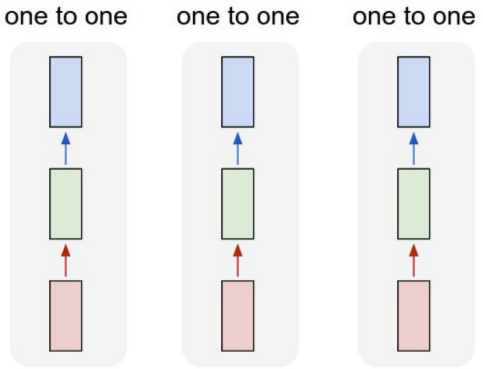
- Text-to-Image Models
- Audio-to-Text Models
- Text-to-Audio
- Multimodal Transformers

Examples:

Whisper, DALL·E, Stable Diffusion, chat GPT ...

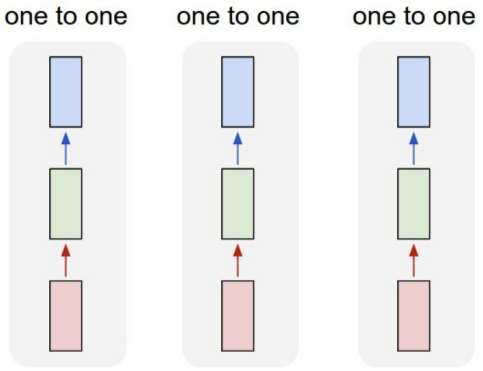
Base DNN architectures

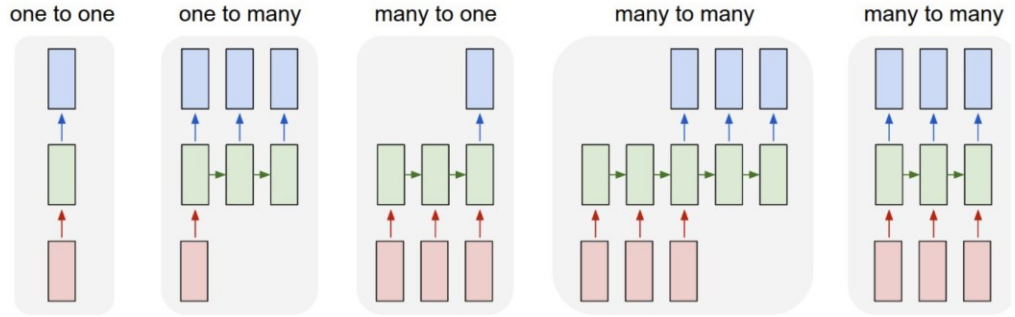
- Multi-Layer Perceptron (MLP)
- Convolutional Neural Networks (CNNs)
  - Processes items one by one
  - No relations between items
  - Structured data (fixed size 1D, 2D, nD tensors)



Base DNN architectures

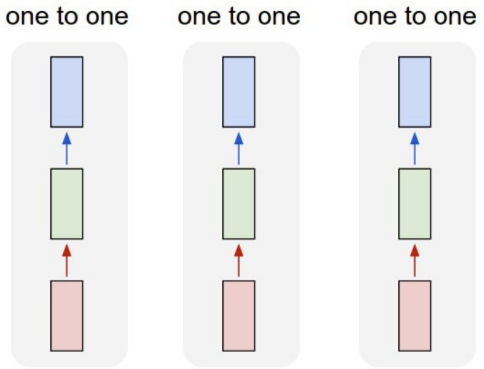
- Multi-Layer Perceptron (MLP)
- Convolutional Neural Networks (CNNs)
  - Processes items one by one
  - No relations between items
  - Structured data (fixed size 1D, 2D, nD tensors)
- Recurrent Neural Networks (RNNs)
- Transformer Networks
  - Collection of items
  - Sequential data for RNN
  - Global relationships for Transformer

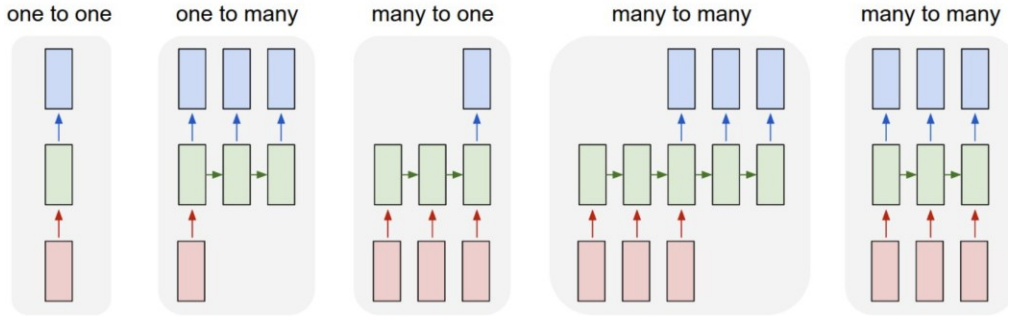


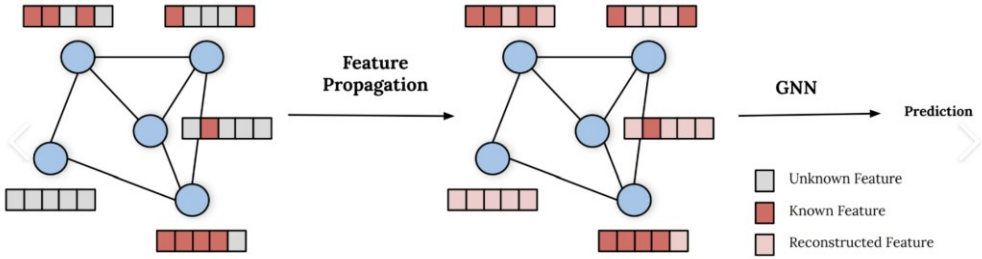


Base DNN architectures

- Multi-Layer Perceptron (MLP)
- Convolutional Neural Networks (CNNs)
  - Processes items one by one
  - No relations between items
  - Structured data (fixed size 1D, 2D, nD tensors)
- Recurrent Neural Networks (RNNs)
- Transformer Networks
  - Collection of items
  - Sequential data for RNN
  - Global relationships for Transformer
- Graph Neural Networks (GNNs)
  - No fixed spatial or sequential order
  - Relationships are defined by edges

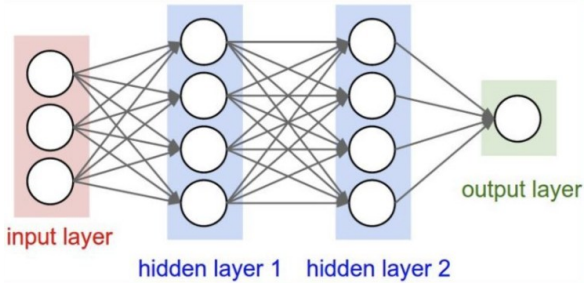




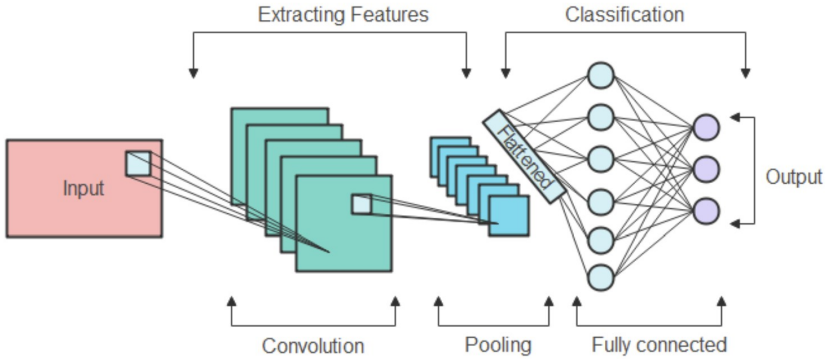


Base DNN architectures

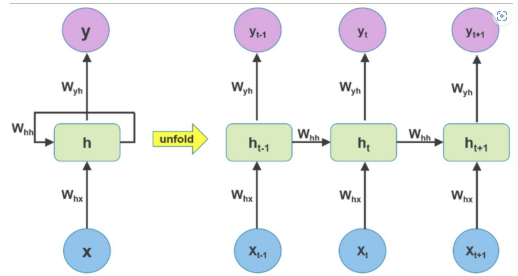
MLP



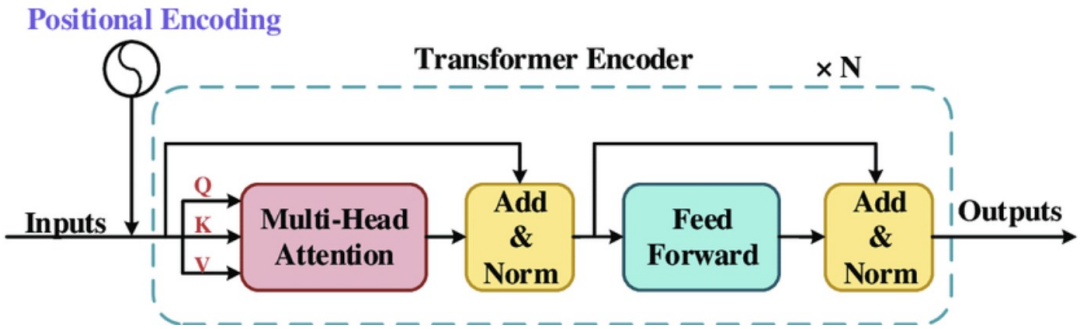
CNN



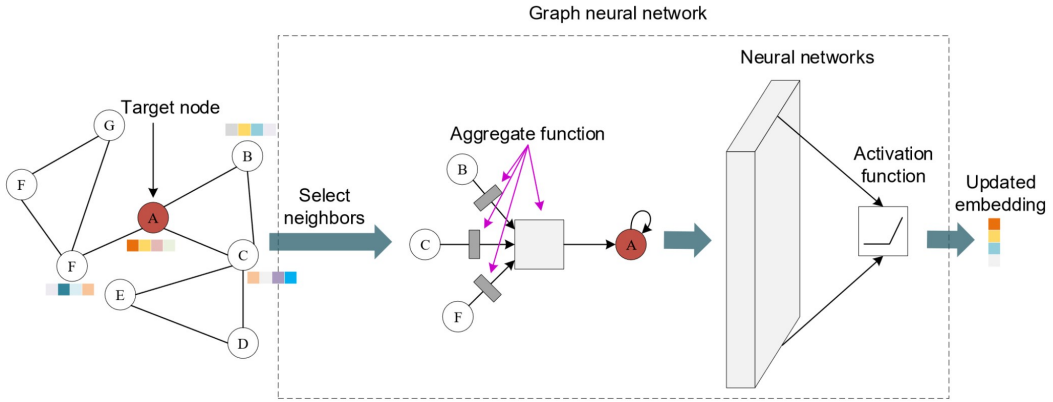
RNN



Transformer

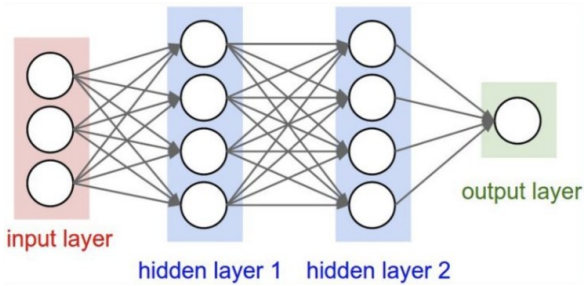


GNN

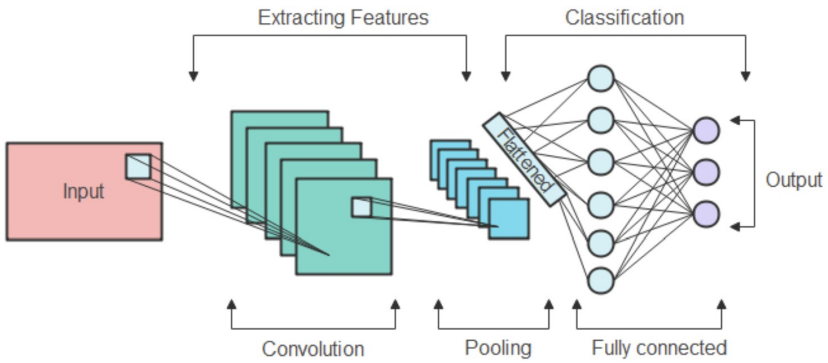


Base DNN architectures

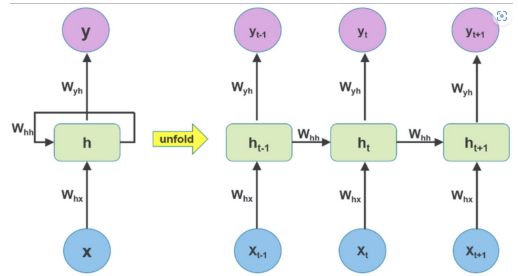
MLP



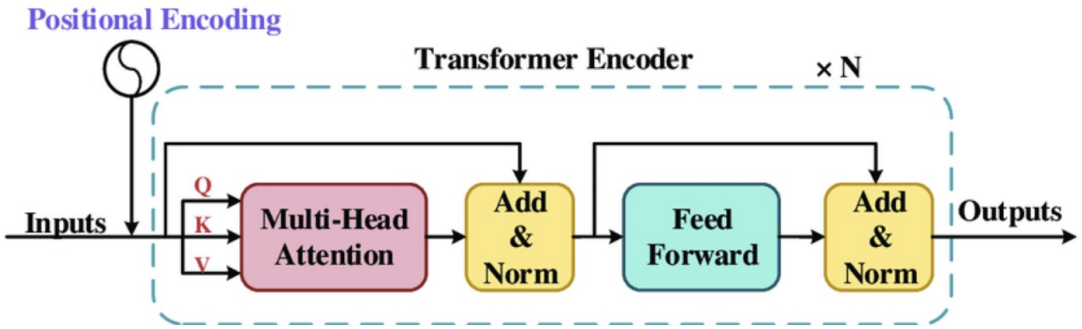
CNN



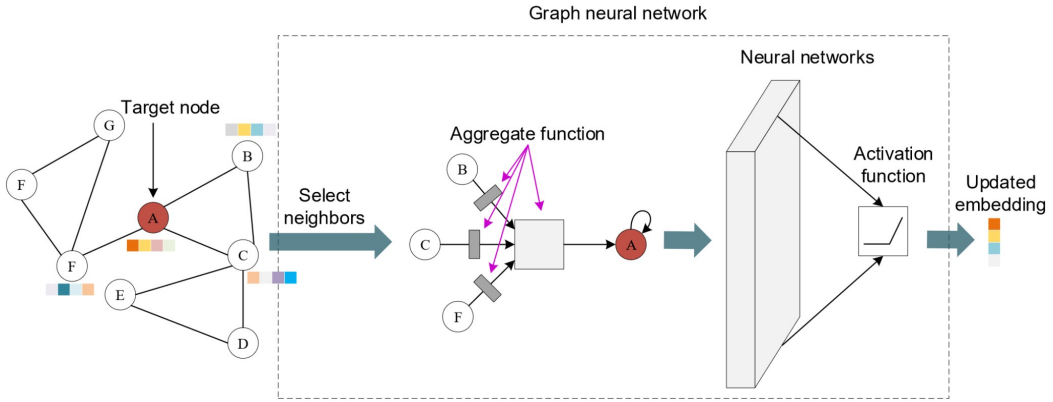
RNN



Transformer

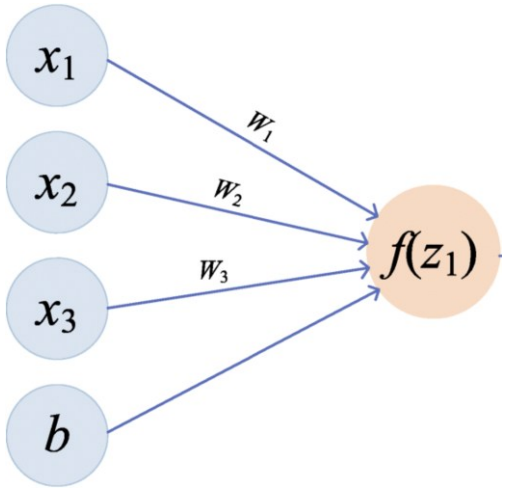


GNN



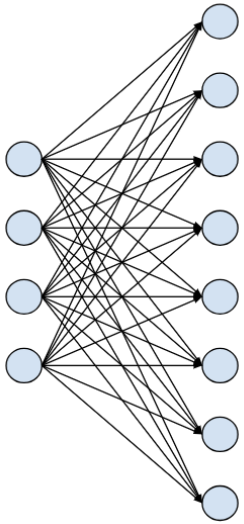
Multi-Layer Perceptron (MLP)

Neuron



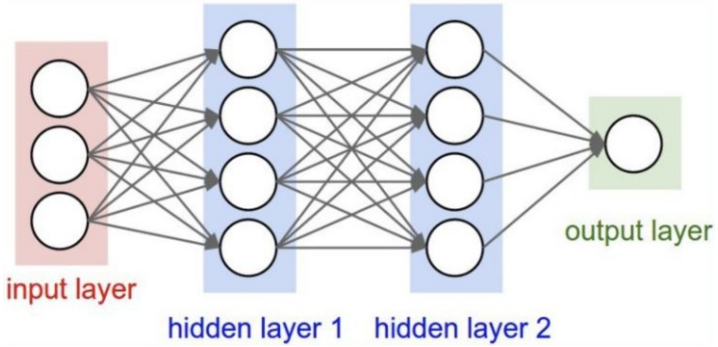
Fully connected (FC) layer

stack



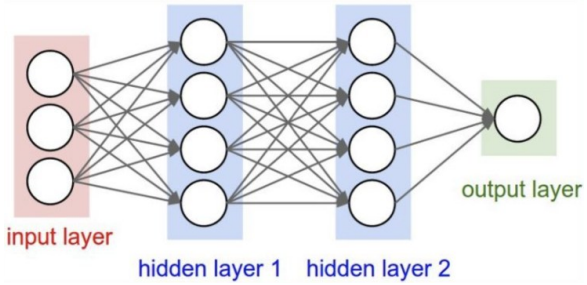
stack

Multi-Layer Perceptron

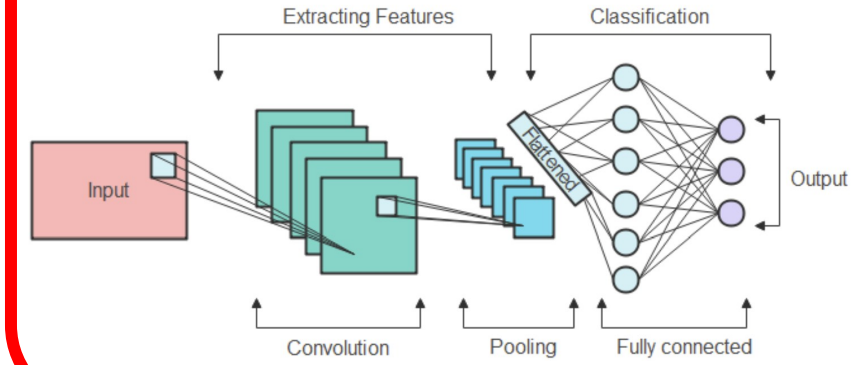


Base DNN architectures

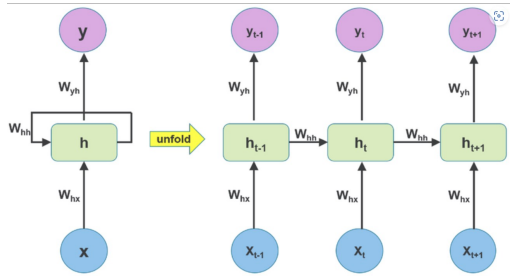
MLP



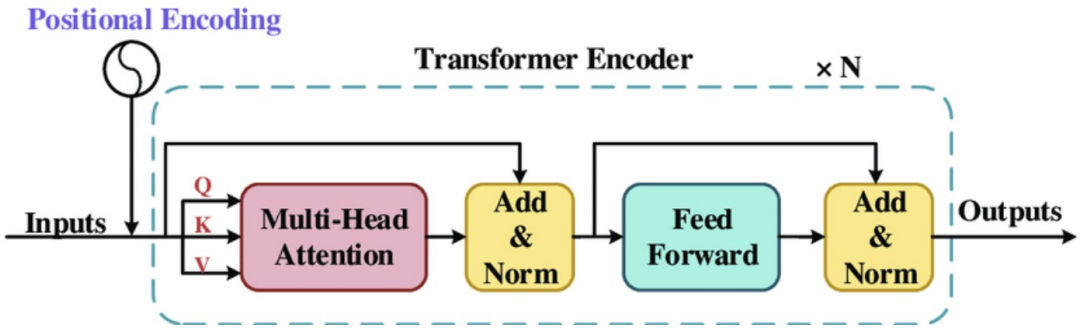
CNN



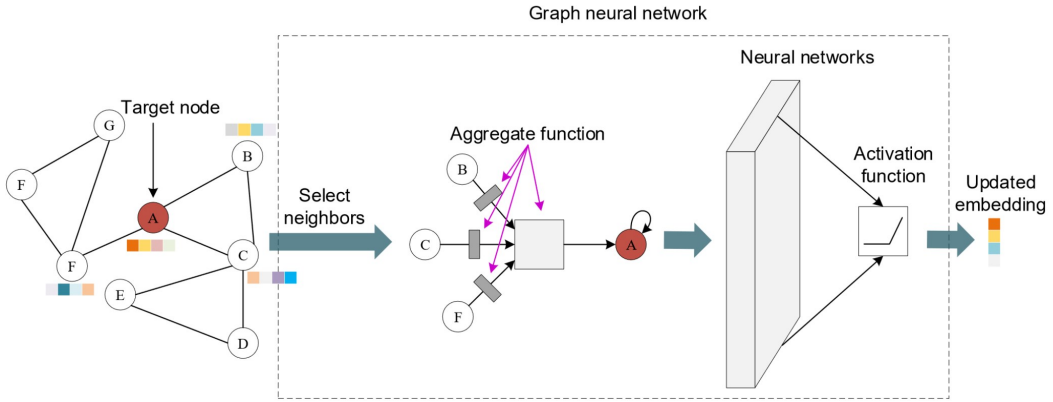
RNN



Transformer

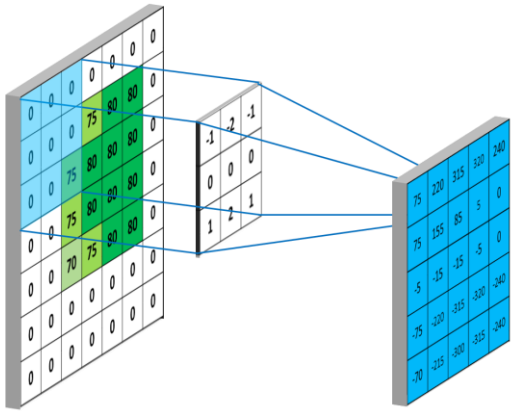


GNN

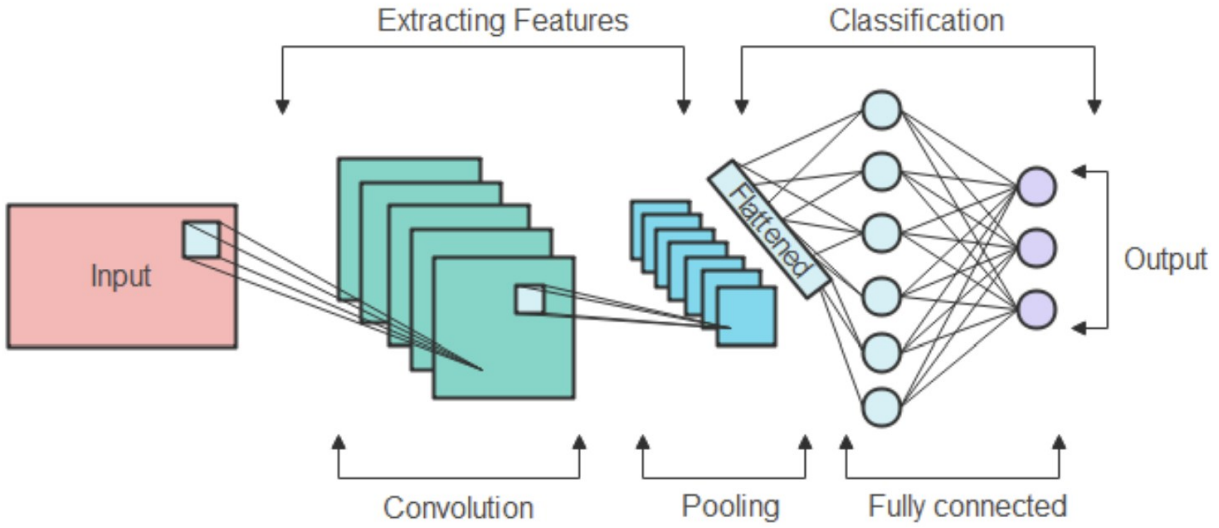


Convolutional Neural Networks (CNNs)

Convolution



CNN

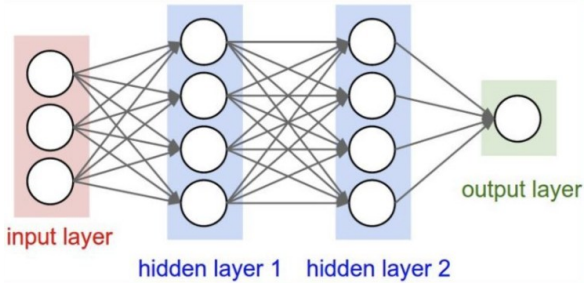


Convolution is central operation

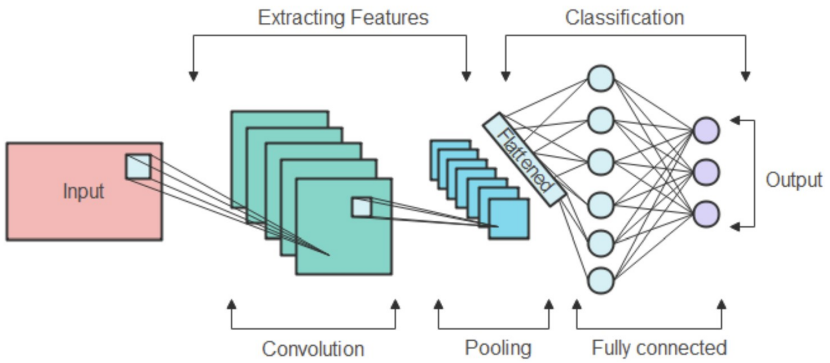
CNNs combine convolution, pooling, FC layers, …

Base DNN architectures

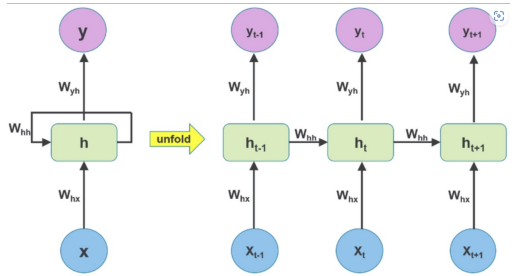
MLP



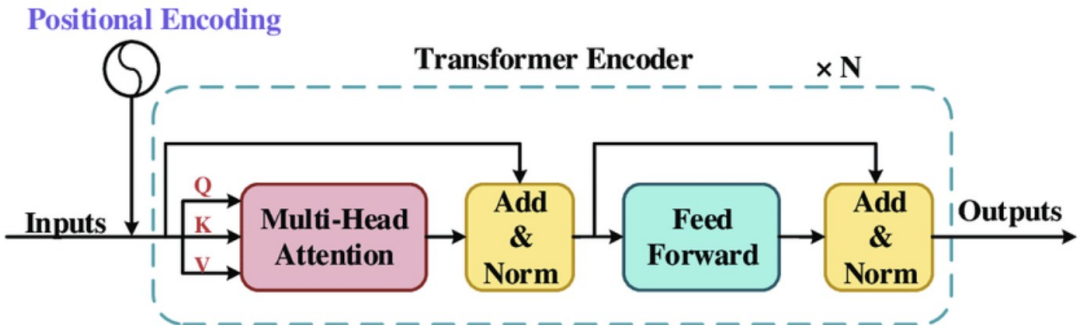
CNN



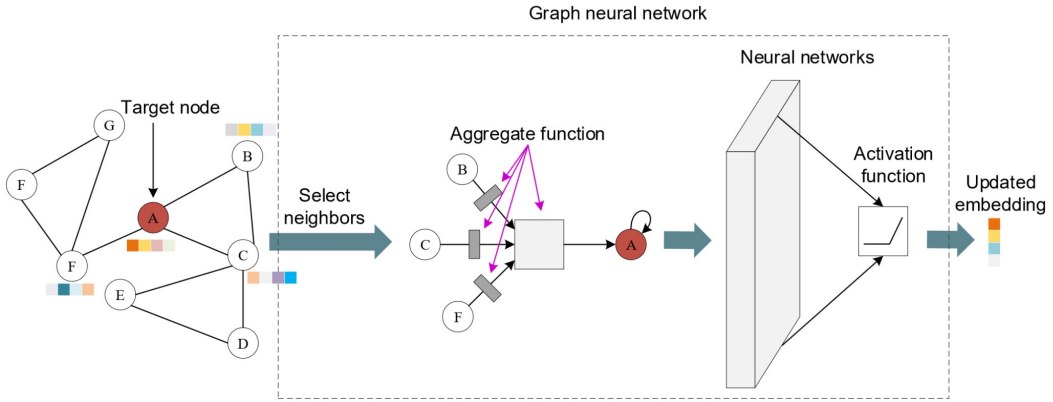
RNN



Transformer

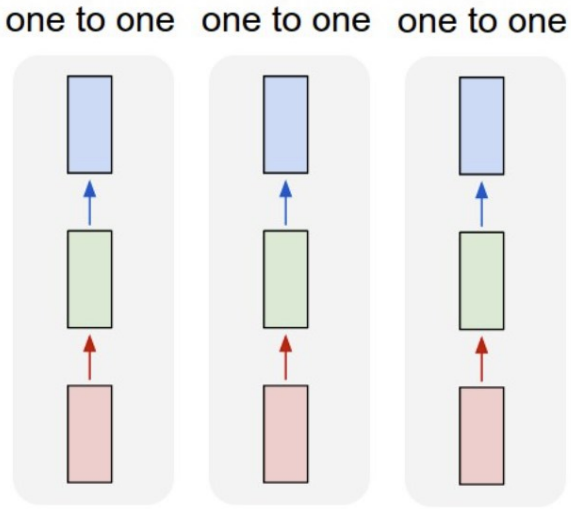


GNN

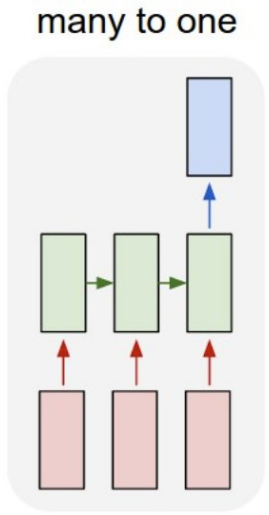


Recurrent Neural Networks (RNNs)

Working with sequencies: Prediction depends on the past



One by one processing:
relation between items lost



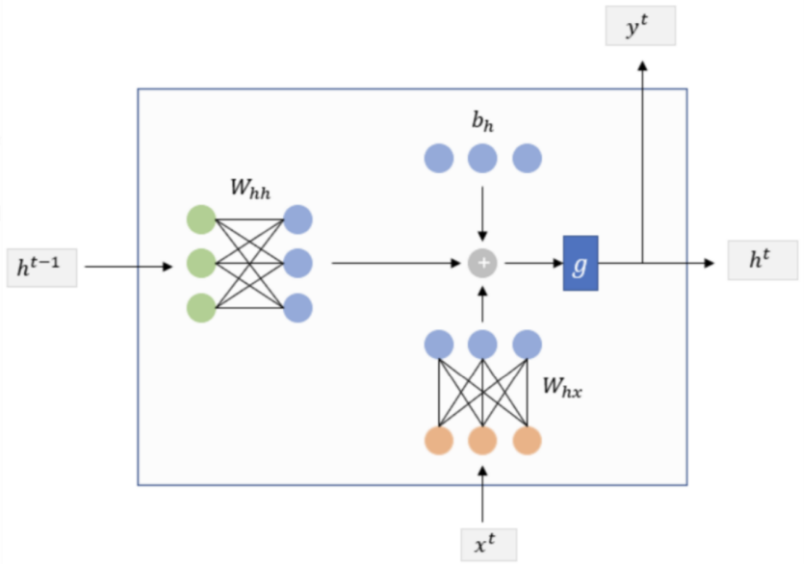
Solution: keep history inside of NN
NN remembers previous items

Example: "The cat is __"

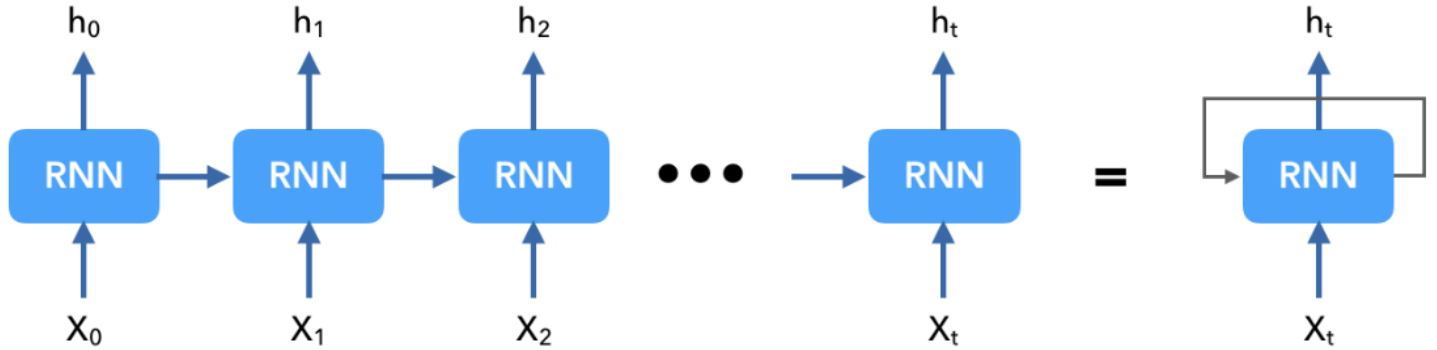
DNN needs to predict the next word: **"sleeping"**

Recurrent Neural Networks (RNNs)

RNN cell



RNN



RNN cell combines hidden state **h** with input **X**

$$h_t = tanh(W_{hh}h_{t-1} + W_{xh}X_t + b_h)$$

RNN keep history of past inputs in **hidden state h**

Recurrent Neural Networks (RNNs)

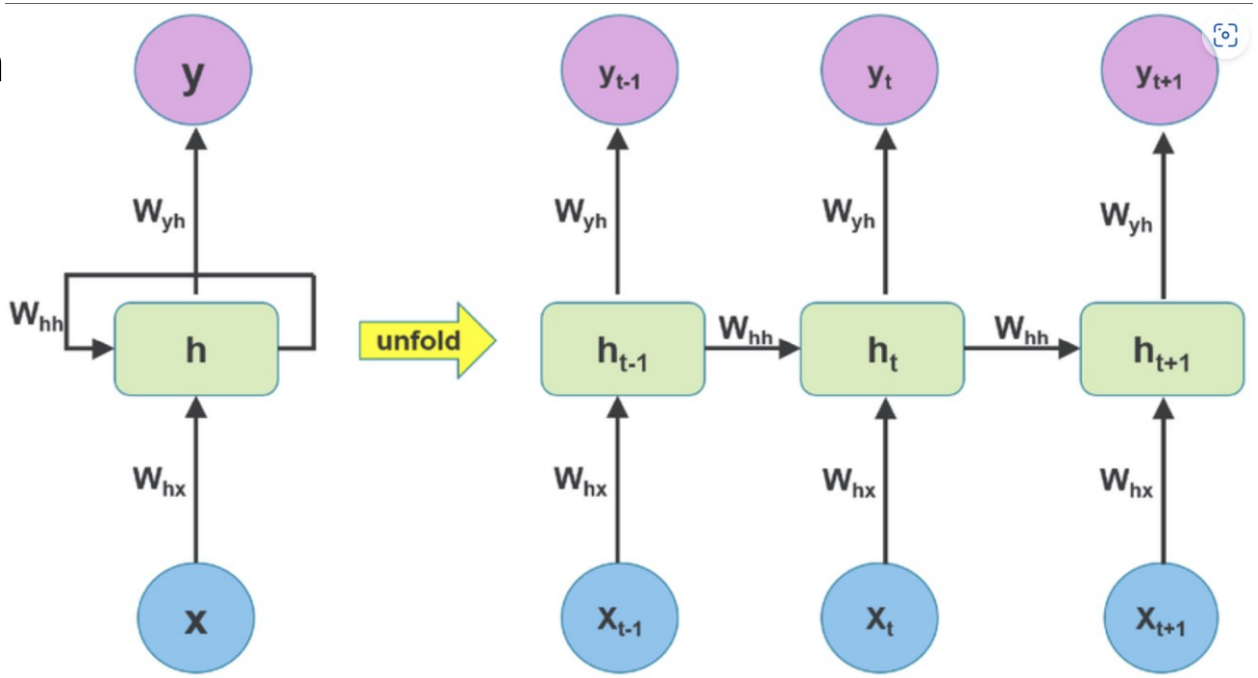
RNN keep history of past inputs in hidden state **h**

- Hidden state **h** is computed using its own previous value with *recurrence relation*

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

- **h** is used to predict output

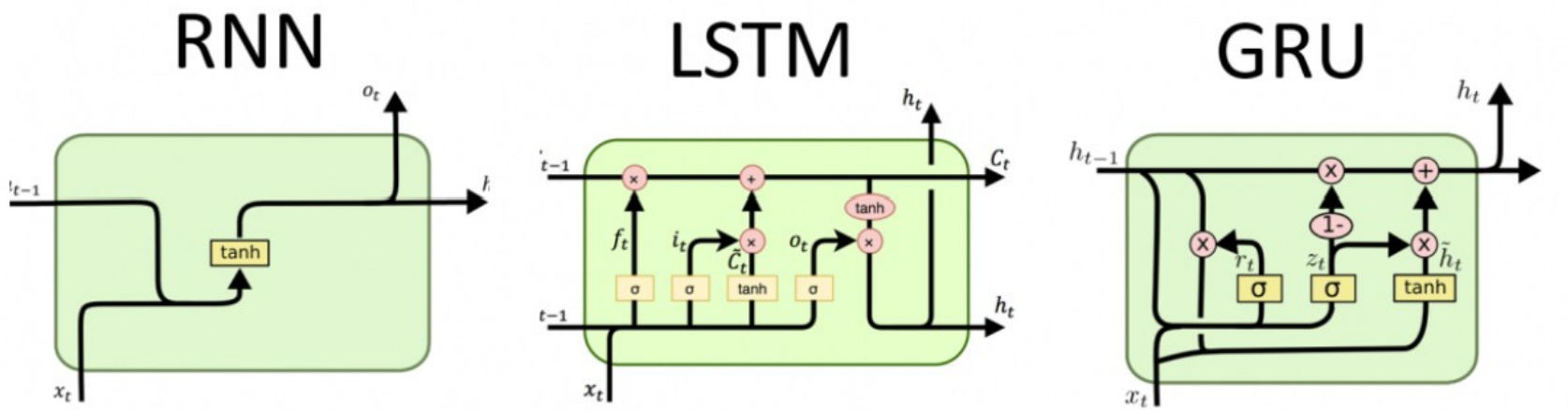
$$y_t = W_{hy}h_t + b_y$$



Recurrent Neural Networks (RNNs)

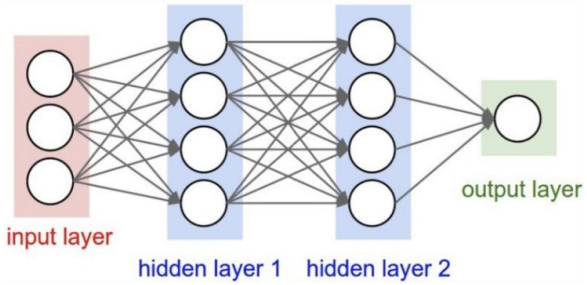
Recurrent Neural Networks types

- Vanilla RNN - *problem with vanishing gradients*
- Long Short-Term Memory (LSTM) - *allows long-term dependencies*
- Bidirectional LSTM (BiLSTM) - *both directions for better context understanding*
- Gated Recurrent Units (GRUs) - *simpler, faster alternative to LSTMs*

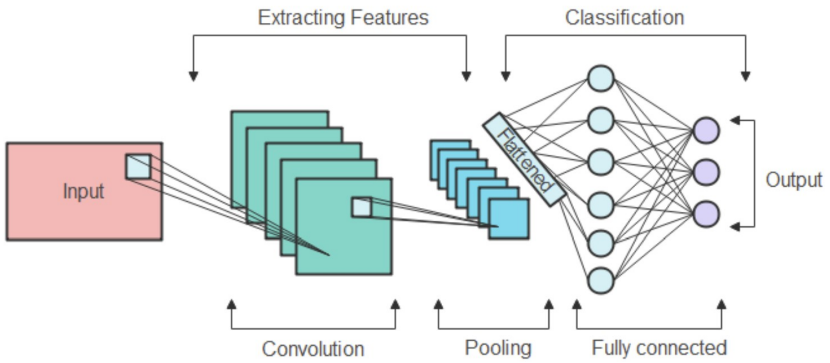


Base DNN architectures

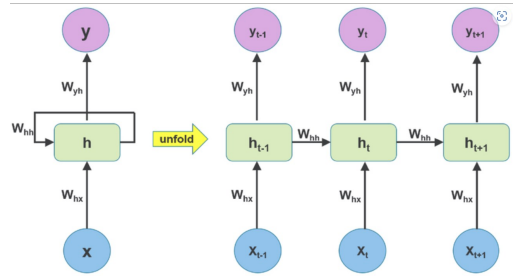
MLP



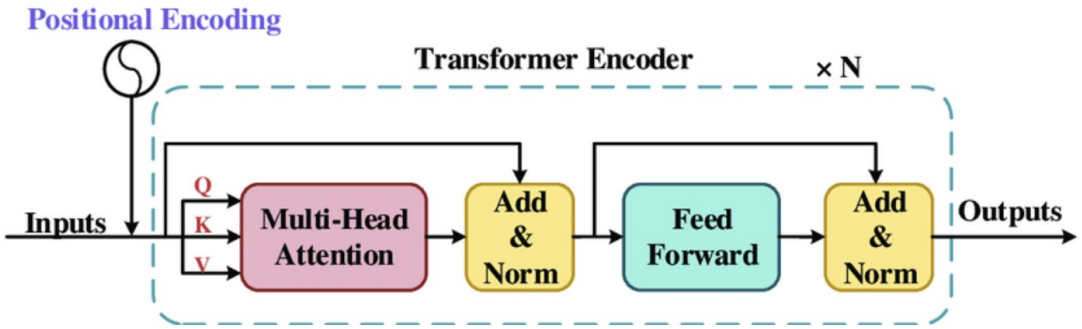
CNN



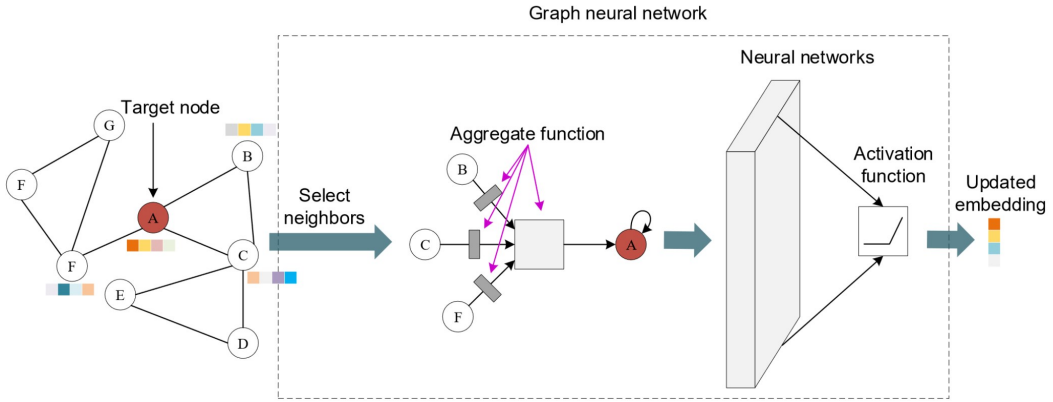
RNN



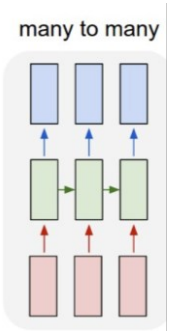
Transformer



GNN

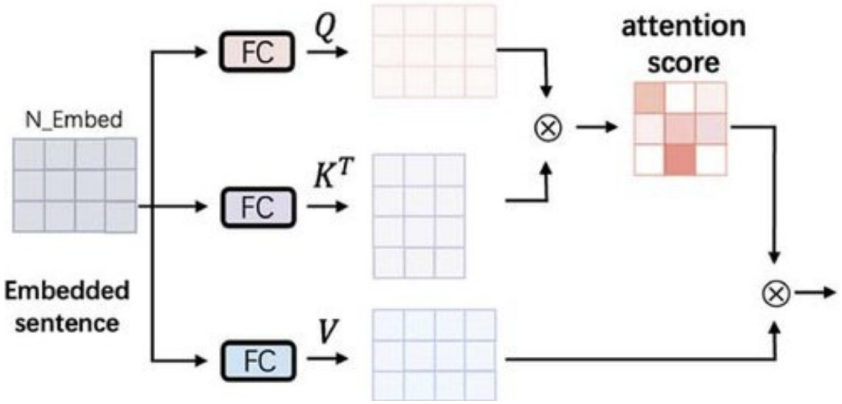


Transformer networks



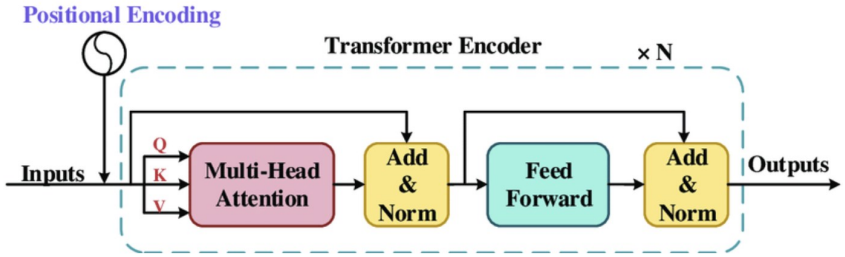
Transformers were introduced as alternative to RNNs

(Self)Attention



Self-attention is “central” operation for Transformers

Transformer encoder



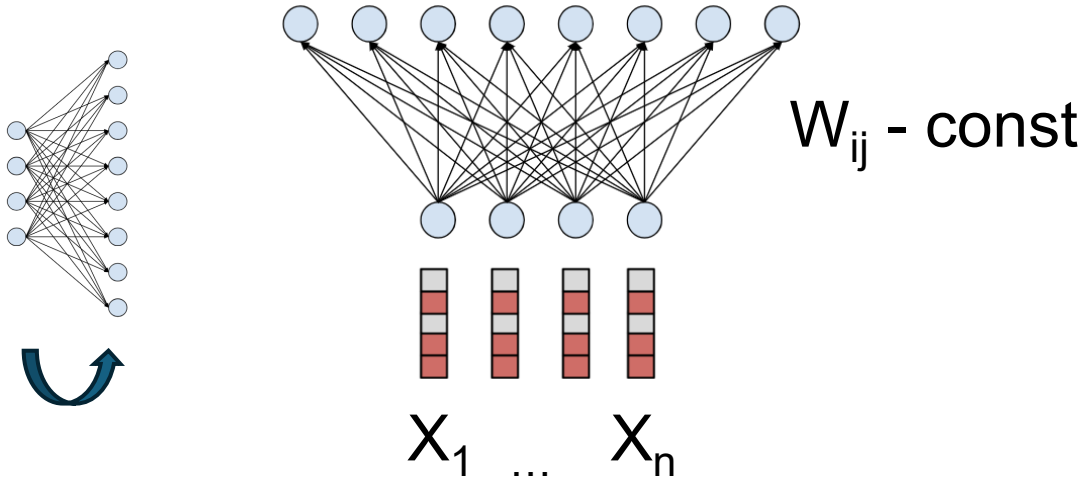
Transformers NN consists of block.
Each block combines:

- Attention
- Layer normalization
- Residual connections
- FC layers

Transformer networks

Intuition behind **attention mechanism**

Fully connected (FC) layer

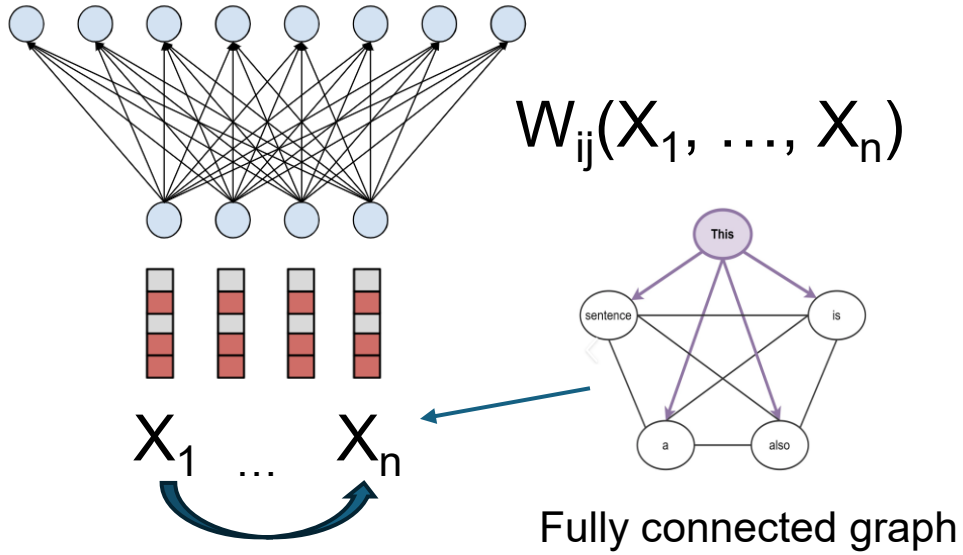


W_{ij} - const

X_1 ... X_n

Weights independent on input

Attention



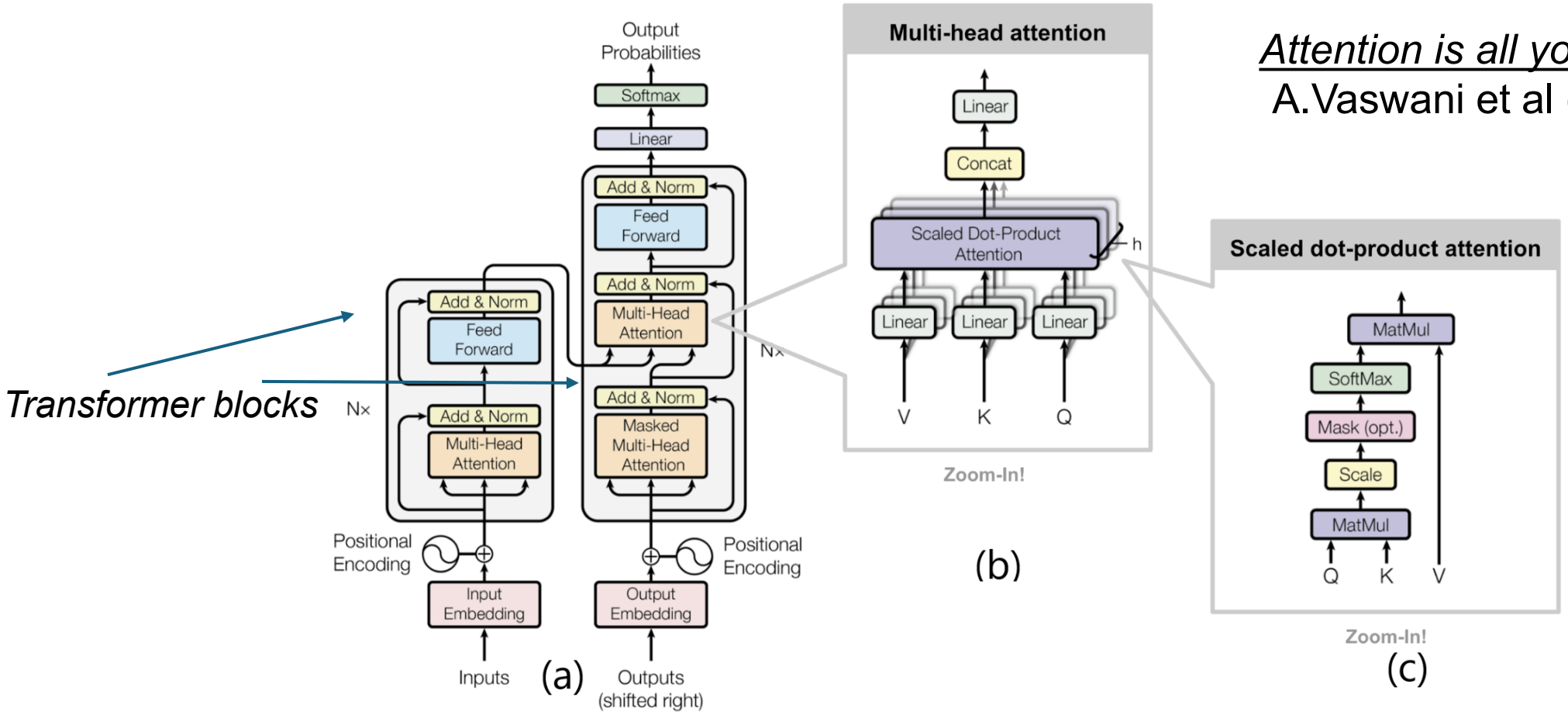
$W_{ij}(X_1, \ldots, X_n)$

X_1 ... X_n

Fully connected graph

Weights dependent on input via attention

Transformer networks

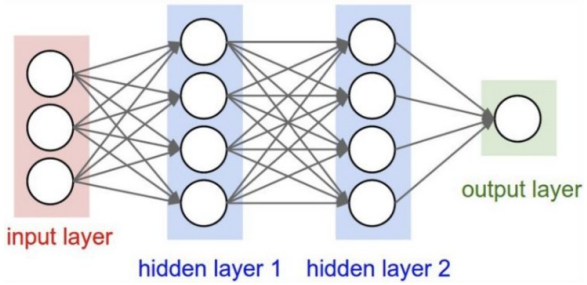


Attention is all you need
A.Vaswani et al (2017)

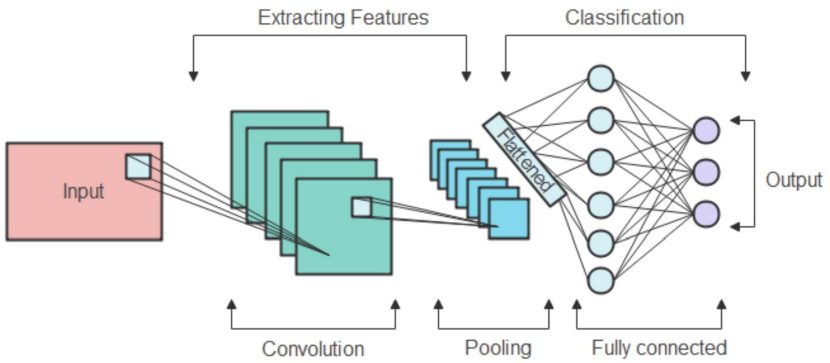
And many more other important components … (see Ivan's talk)

Base DNN architectures

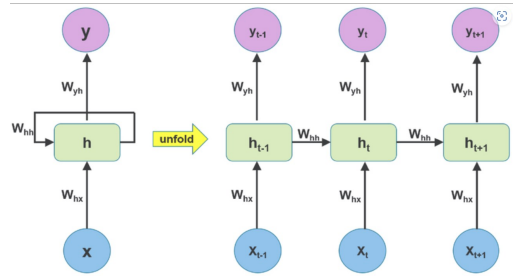
MLP



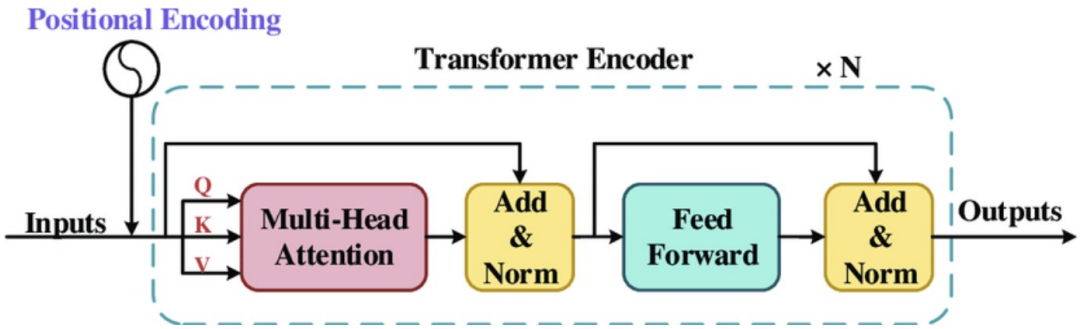
CNN



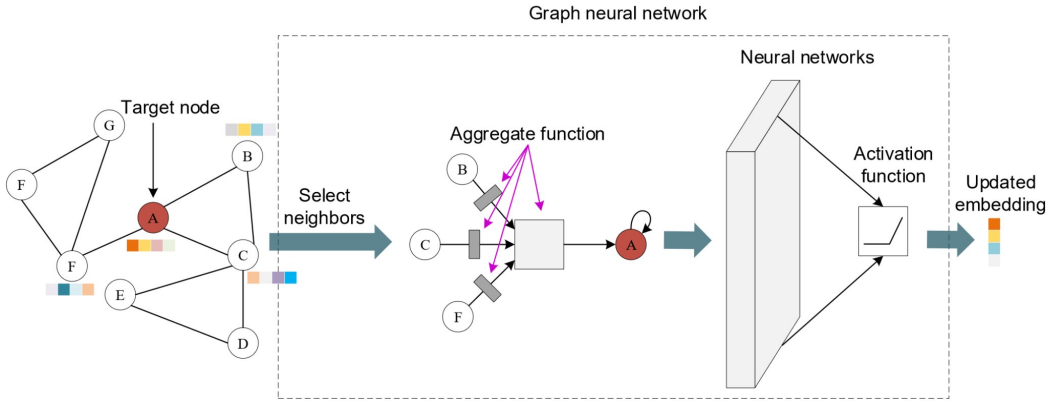
RNN



Transformer



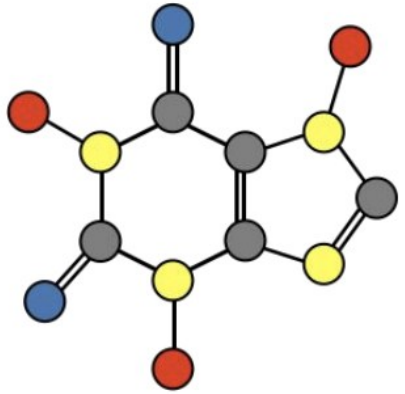
GNN



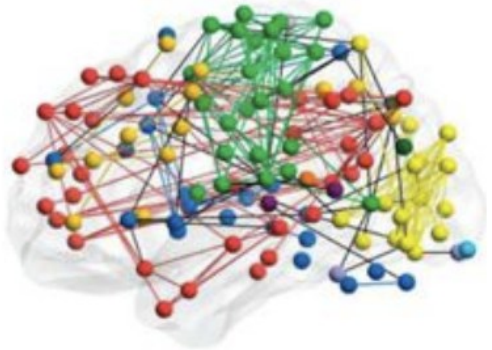
Graph Neural Networks (GNNs)

Data with no regular structure

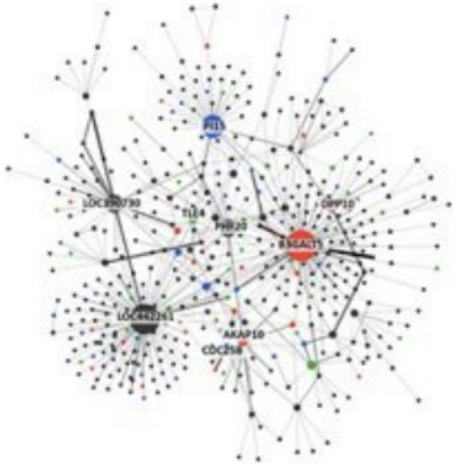
Social Networks



Molecules

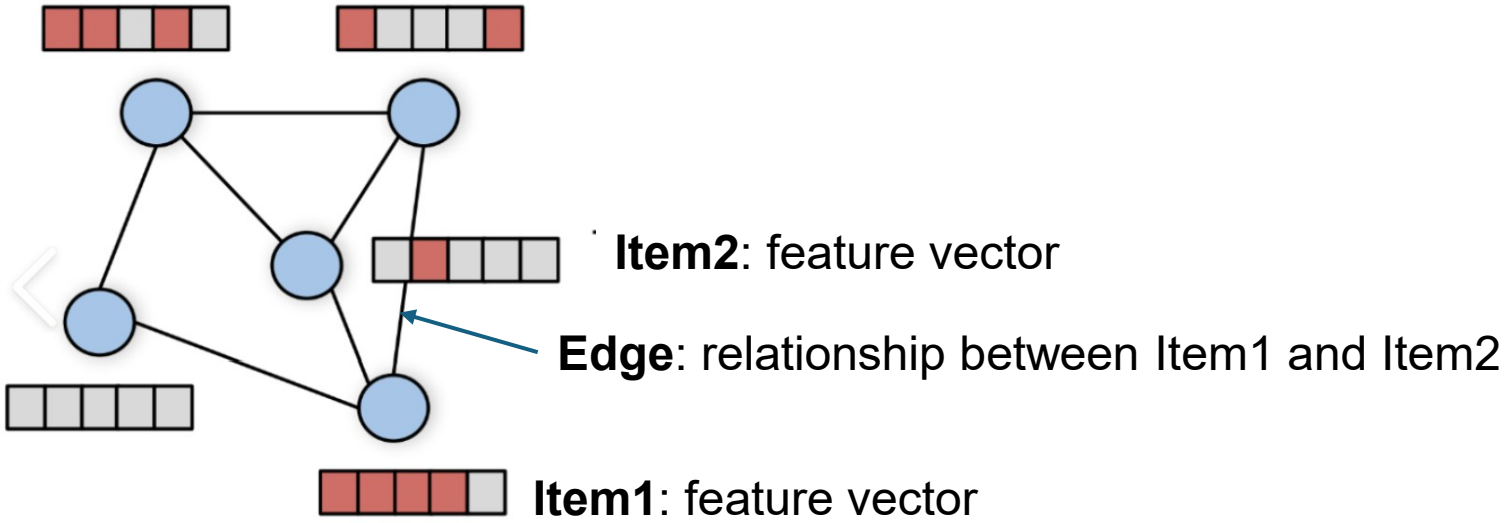


Functional Networks



Interaction Networks

Graph Neural Networks (GNNs)

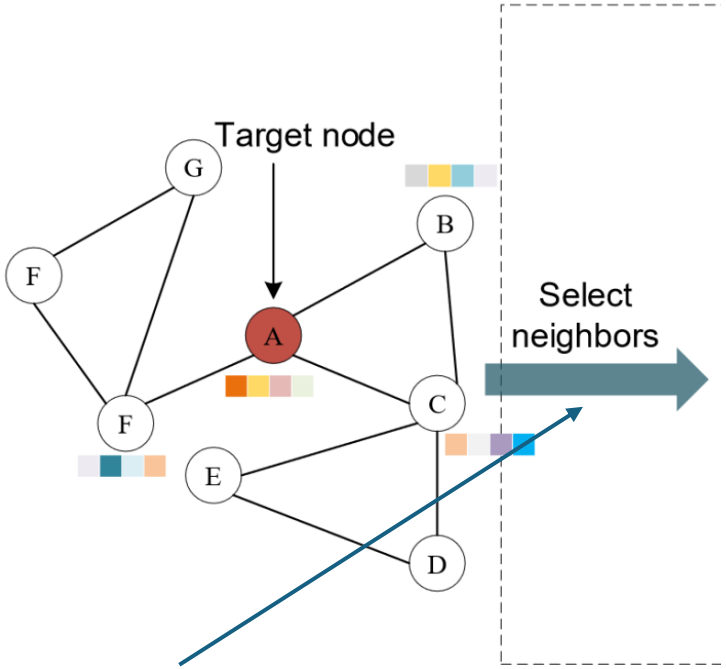


No regular structure: Graph-structured data via

- Nodes – items represented by feature vectors
- Edges – relationships between items

Graph Neural Networks (GNNs)

Layer operations

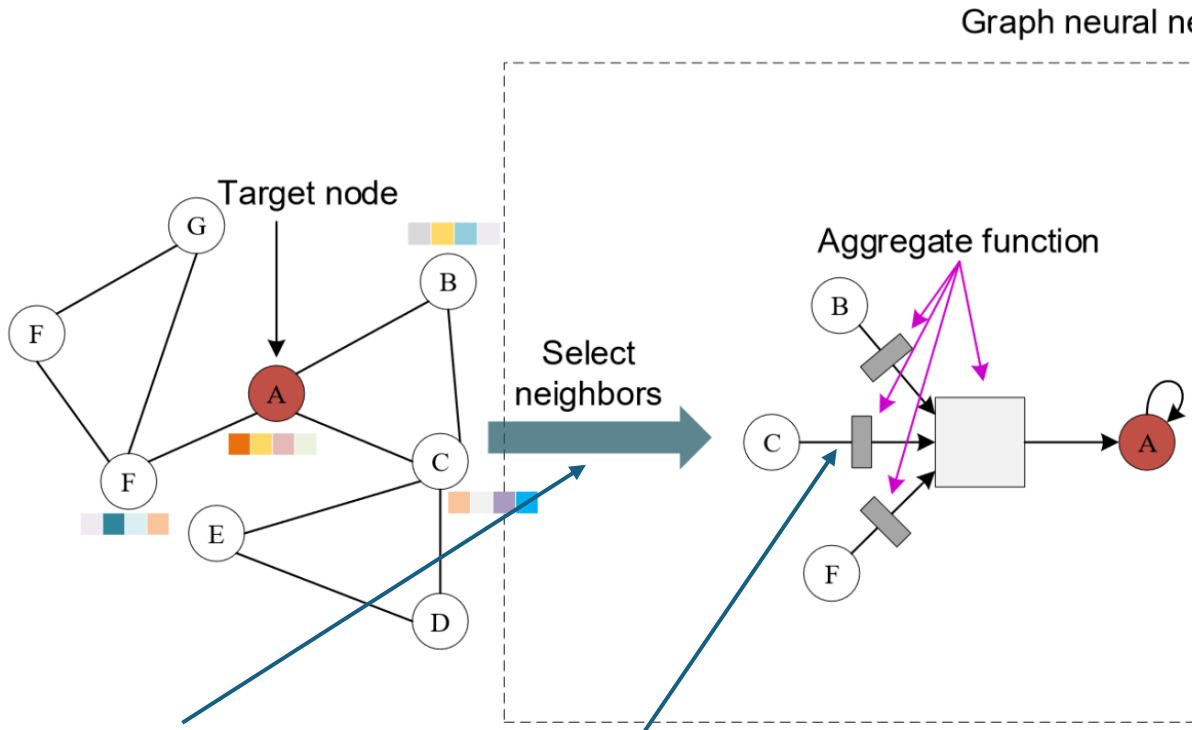


Neighbor Selection (e.g. k-Nearest Neighbors)

Nodes gather features from neighbors

Graph Neural Networks (GNNs)

Layer operations

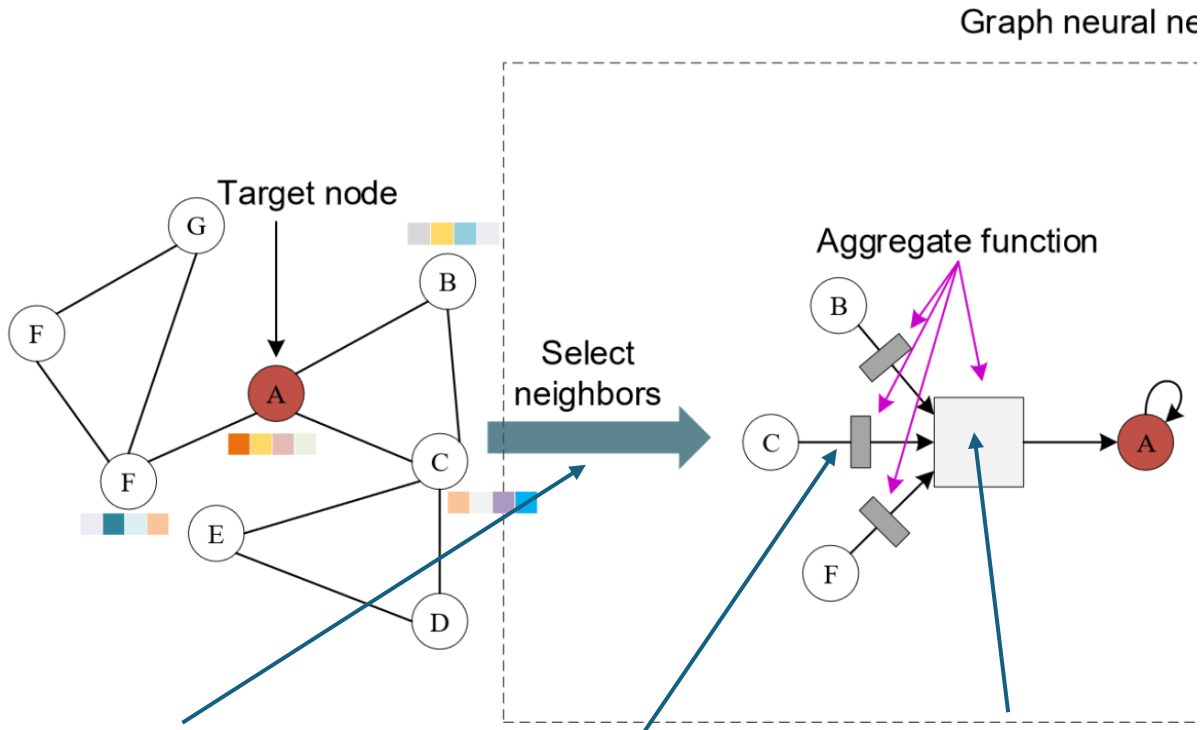


Neighbor Selection
Nodes gather features from neighbors

Message Passing
Nodes exchange information

Graph Neural Networks (GNNs)

Layer operations



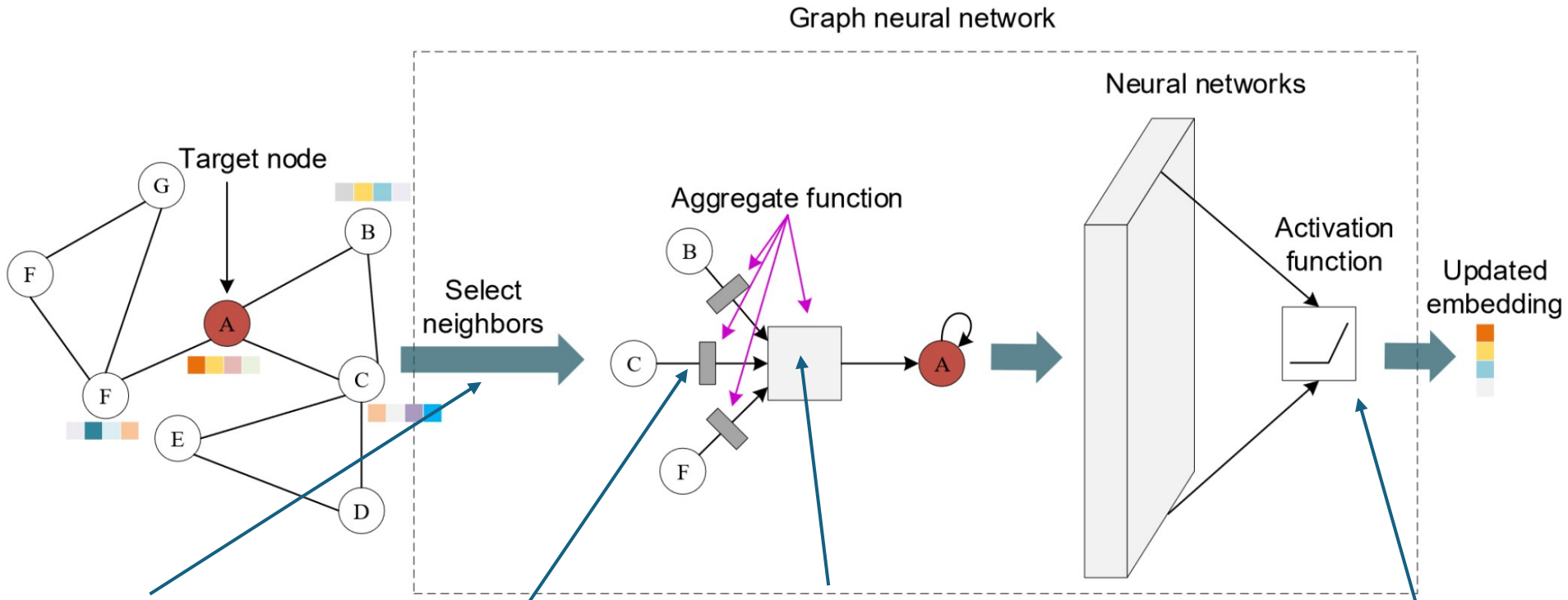
Neighbor Selection
Nodes gather features from neighbors

Aggregation
Combine neighbor messages (sum, mean, max, or attention)

Message Passing
Nodes exchange information

Graph Neural Networks (GNNs)

Layer operations



Neighbor Selection
Nodes gather features from neighbors

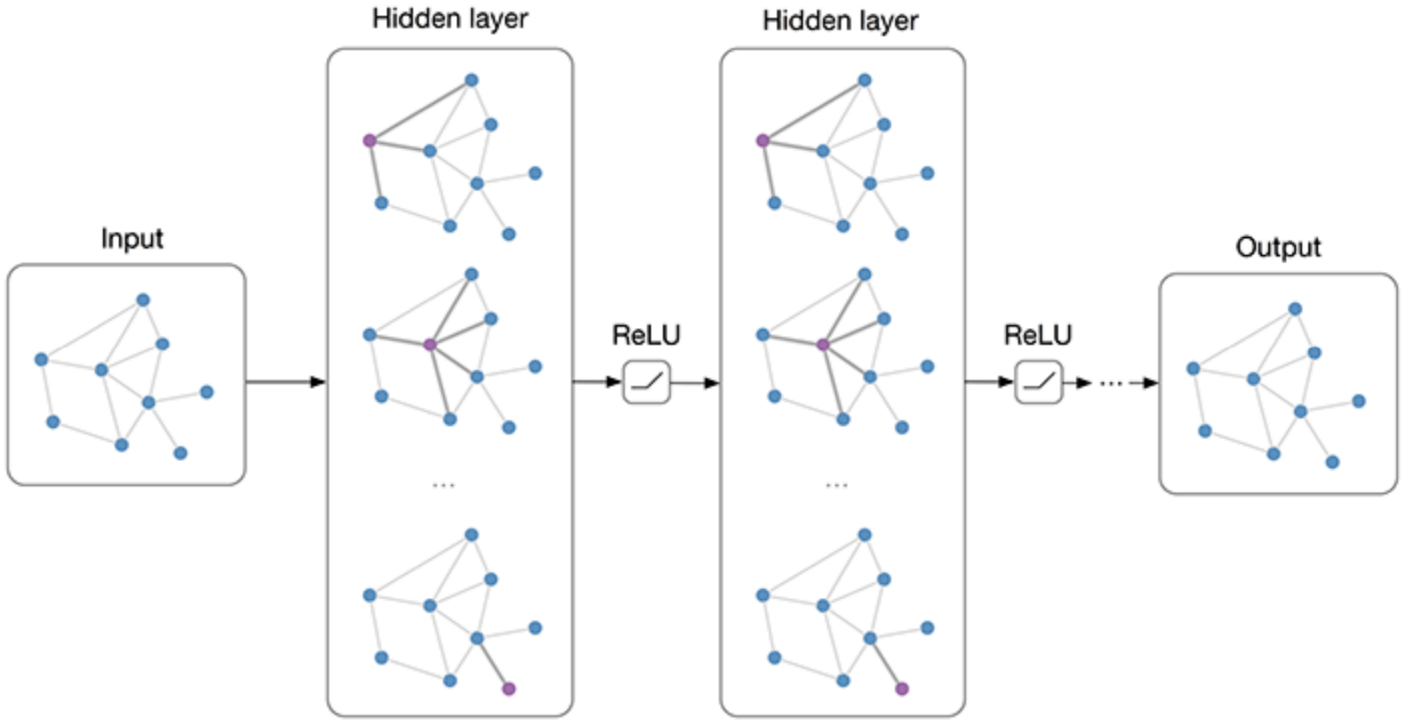
Message Passing
Nodes exchange information

Aggregation
Combine neighbor messages (sum, mean, max, or attention)

Update
Apply nonlinearity to update node features

Graph Neural Networks (GNNs)

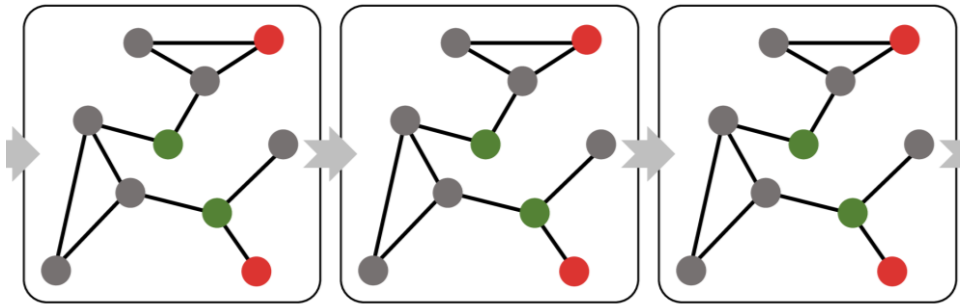
*Full **GNN***



Hidden layer: above (layer) operations for each node
GNN: combination of layers

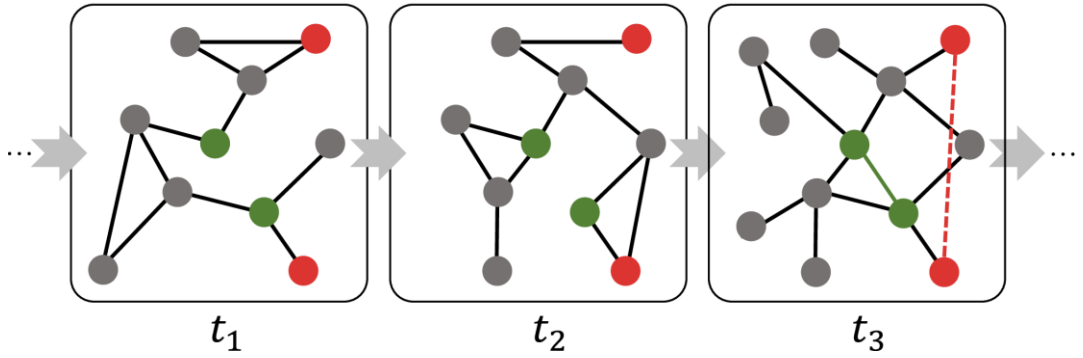
Graph Neural Networks (GNNs)

Static graph



- Structure (nodes and edges) remains **fixed**
- Node representations are updated based on this unchanging graph topology

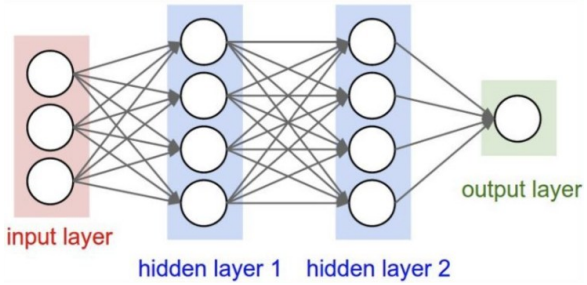
Dynamic graph



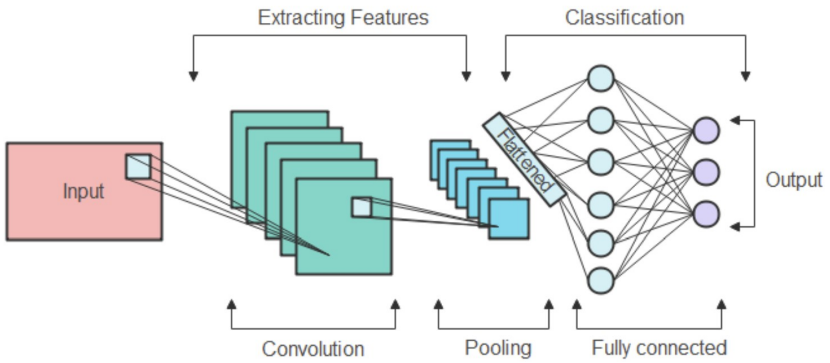
- Structure **evolves over time**, with nodes and edges being added or removed.
- Node representations are updated as the graph topology changes during the process.

Base DNN architectures

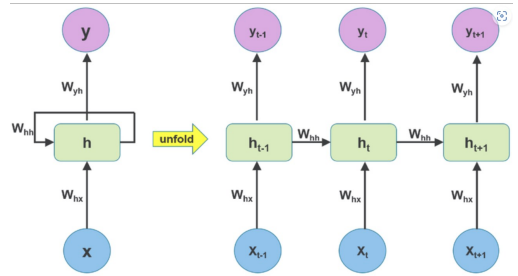
MLP



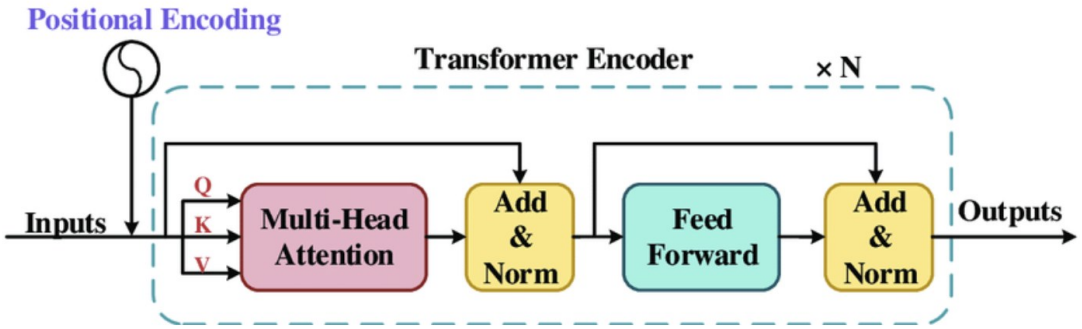
CNN



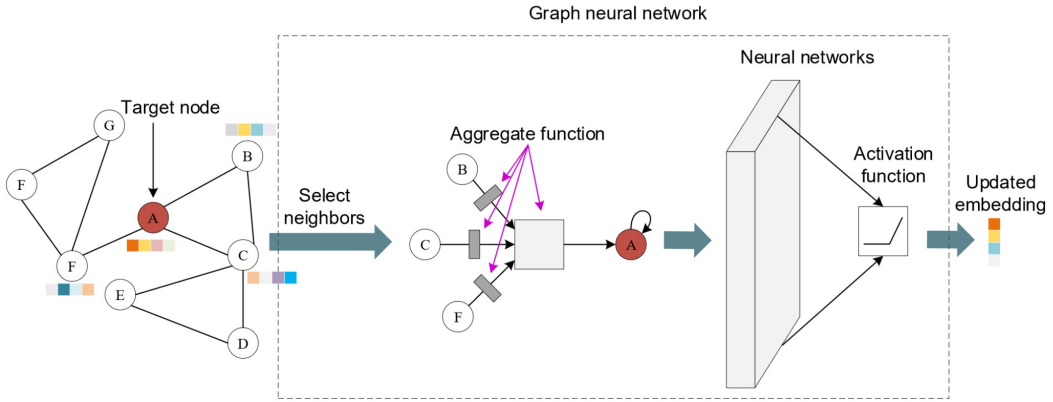
RNN



Transformer



GNN

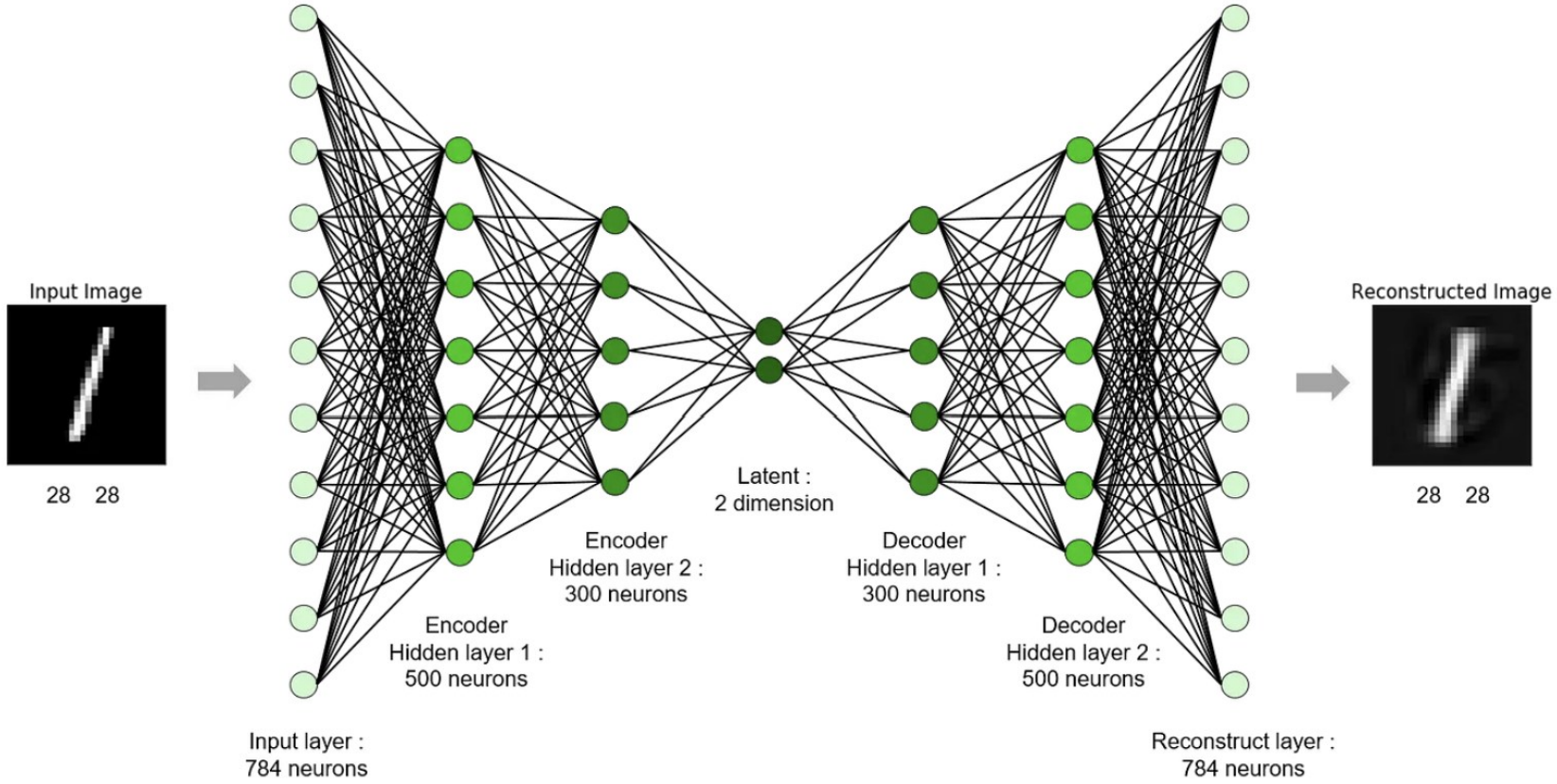


Generative Models

*Models that **generate new data** by learning the distribution of existing data:*

- Autoencoders (AE) & Variational Autoencoders (VAE)
- Generative Adversarial Networks (GANs)
- Normalizing Flows
- Diffusion Models

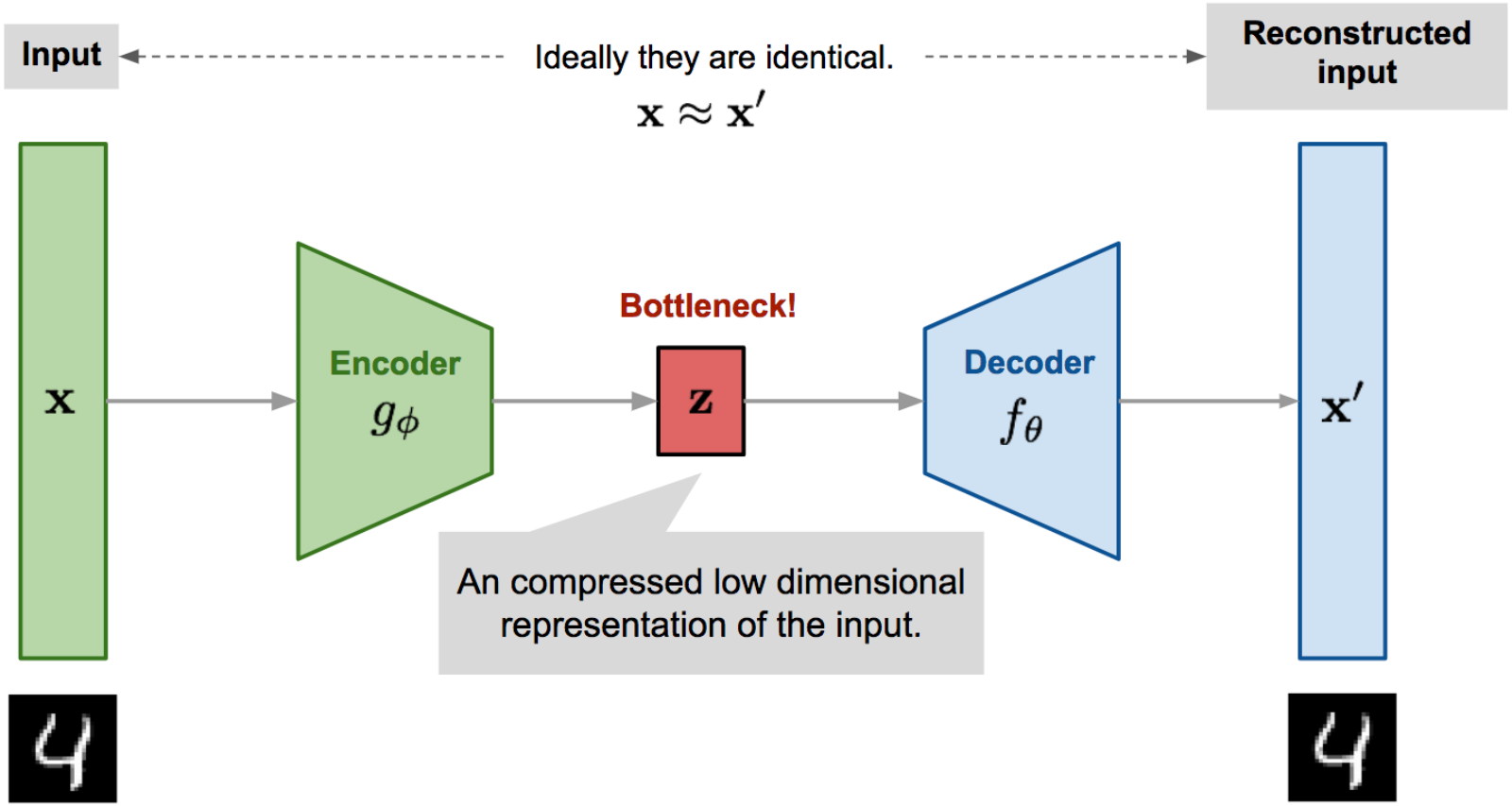
Autoencoder (AE)



Autoencoder

Encodes the input to low-dimensional latent space and **decodes** to original shape

Autoencoder (AE)

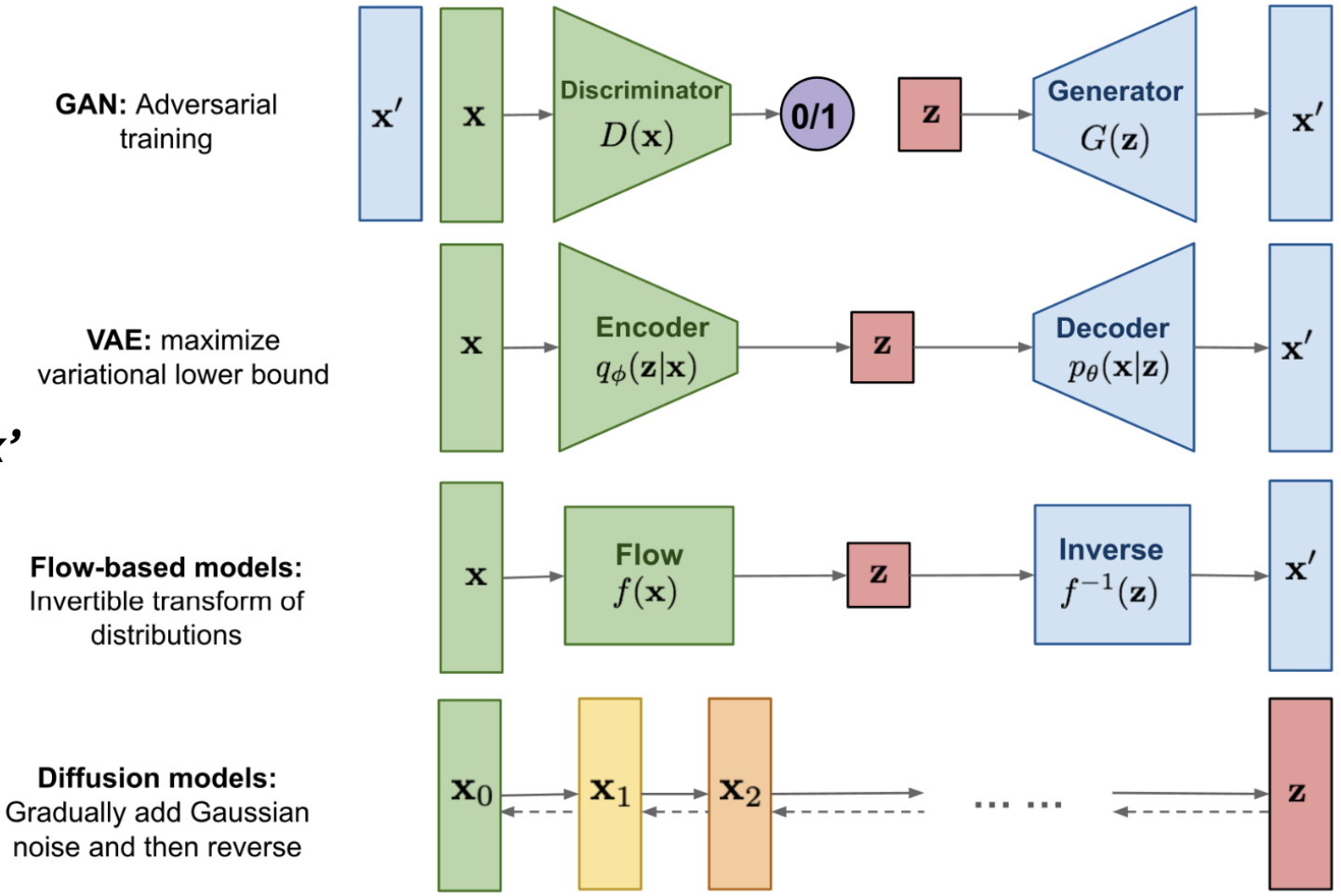


Encodes the input to low-dimensional latent space and **decodes** to original shape

Generative Models

How to generate new data?

- ***Sample latent variable z:***
  - Low-dimensional, compressed representation
  - Simple probability distribution (often a standard Gaussian distribution)
- ***Apply learned transformation to map z → x'***
  - Generator (GAN)
  - Decoder (VAE)
  - Inverse flow (Normalizing flow)
  - Remove Gaussian noise (Diffusion)
- *The transformation is **learned during training** to capture the **underlying data distribution***



There is not one library to rule them all

Neural networks/ Deep learning

Google DeepMind

ML algorithms / optimization

+

theano