Neural Network Architectures

Qing-Yuan Jiang

The school of Computer Science and Engineering, NJUST

2026.10.29

Review of Previous Class

- What is an artificial neuron, and what do its weights and bias represent?
- Why do multilayer neural networks need nonlinear activation functions?
- What are the main properties and limitations of sigmoid, tanh, and ReLU?
- How do hidden layers transform inputs into useful representations?
- Why can a single perceptron implement AND but not XOR?
- How can a multilayer network solve the XOR problem?
- What do the forward and backward passes compute?
- How does backpropagation use the chain rule to compute gradients?

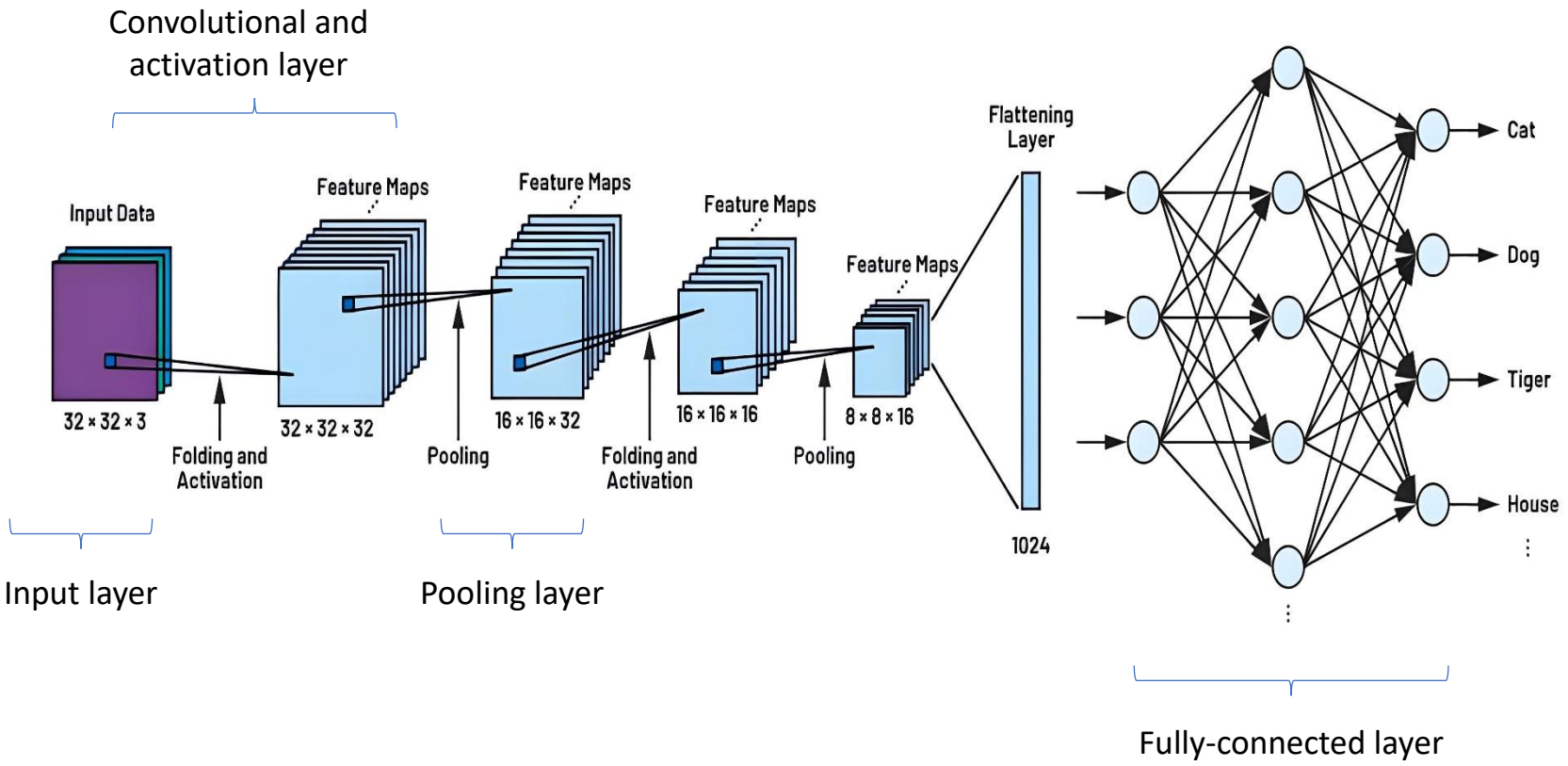
Convolutional Neural Network

Convolutional Neural Networks

Components of Convolutional Neural Networks

- **Input layer:**
  - The raw input of the convolutional network, which can be a raw or preprocessed pixel matrix.
- **Convolutional layer:**
  - Parameter sharing, local connectivity;
  - Utilizes translation invariance to extract local features from global feature maps.
- **Activation layer:**
  - Performs non-linear mapping on the output of the convolutional layer.
- **Pooling layer:**
  - Further filters features, which can effectively reduce the number of parameters required by subsequent network layers.
- **Fully Connected Layer:**
  - Flattens multi-dimensional features into low-dimensional features; usually, low-dimensional features correspond to the learning objectives of the task (categories or regression values).

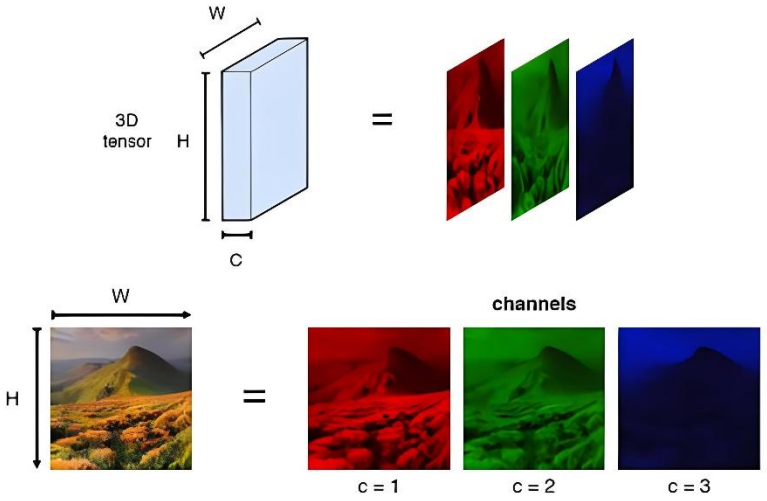
The Arch. of CNN



Convolutional Neural Networks

Input Layer:

- The Input Layer is typically the **raw data** or preprocessed data fed into the convolutional neural network.
- Taking an image classification task as an example, the input image generally contains three **RGB channels** and is a 3D pixel value matrix of H × W × 3, where H and W represent the height and width, respectively.



Convolutional Neural Networks

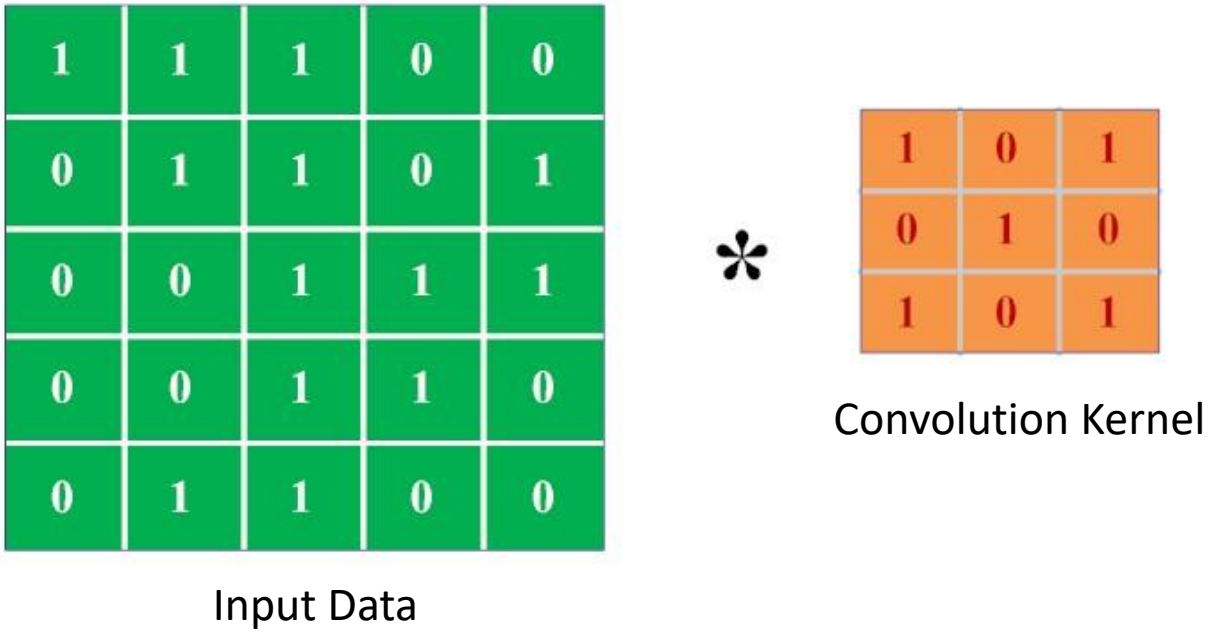
Convolutional Layer:

- In principle, the convolution operation is actually a mathematical operation of element-wise multiplication and summation on two pixel matrices.
- One of the matrices is the input data matrix, and the other matrix is the convolution kernel (filter or feature matrix). The computed result represents the specific local features extracted from the original image.

Convolutional Neural Networks

Convolutional Layer:

- Example of Convolution Operation:
  - Assume the data matrix and convolution kernel are as follows:



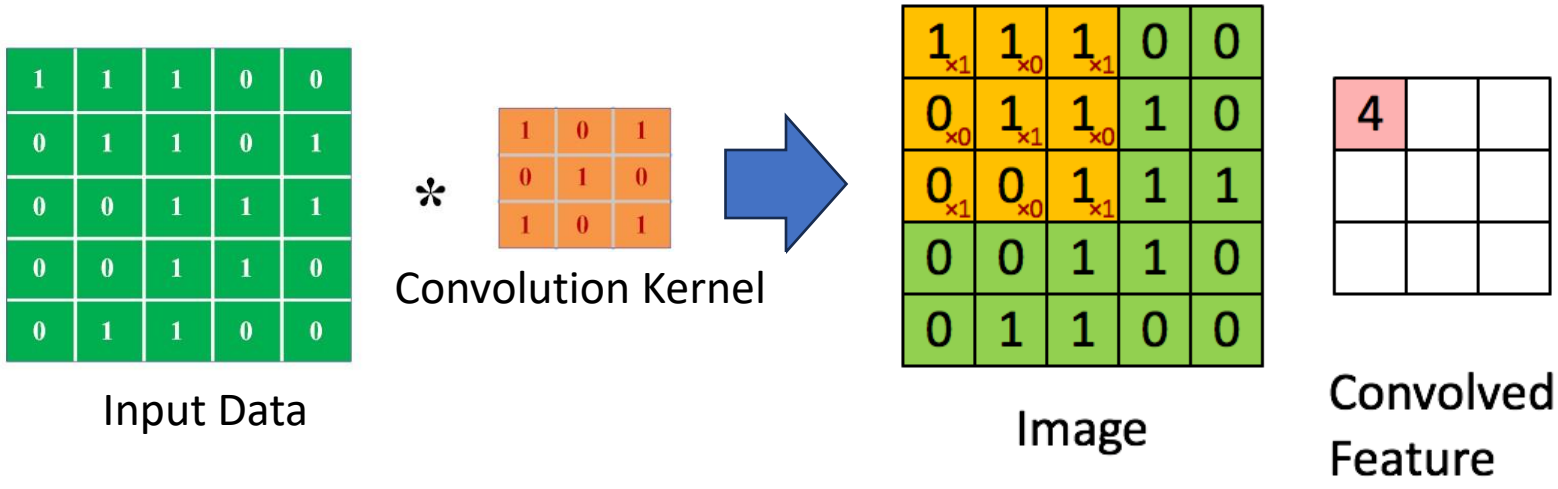
Input Data

Convolution Kernel

Convolutional Neural Networks

Convolutional Layer:

- Example of Convolution Operation:
  - Convolution Process:

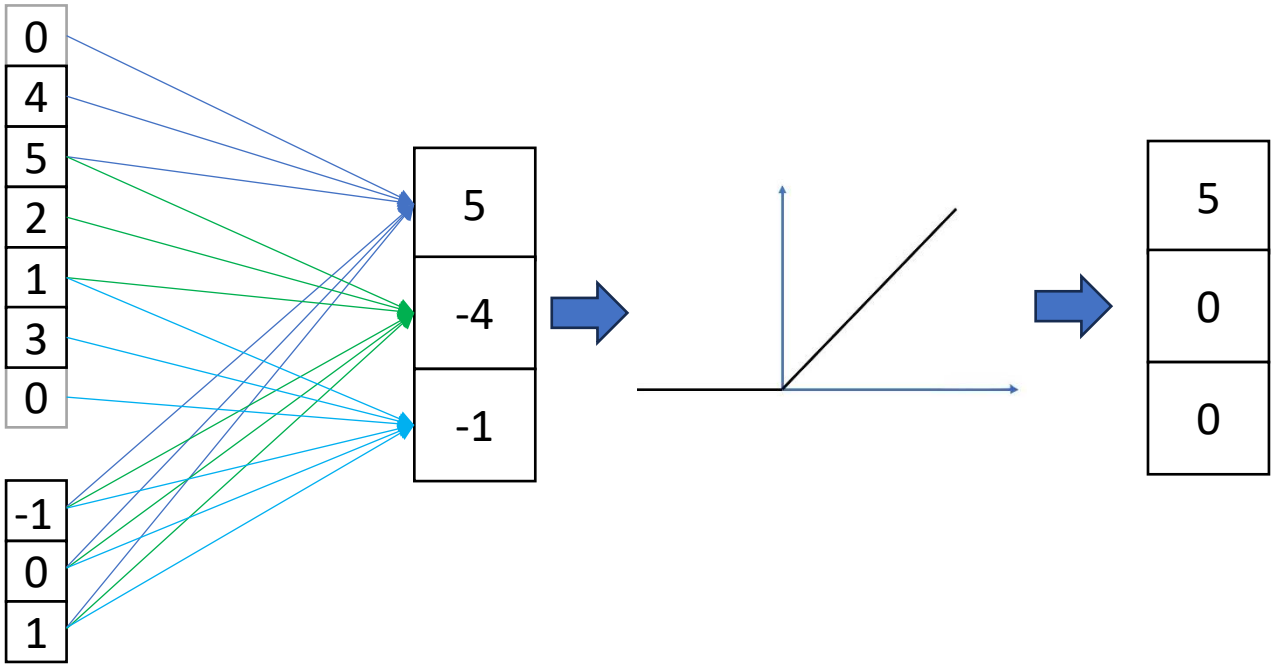


Convolutional Neural Networks

- **Convolutional Layer:**
  - Concepts and Parameters in the Convolutional Layer
    - **Weight sharing:** The parameters of the convolution kernel remain unchanged during a single scan.
    - **Local connectivity:** Local information extraction.
    - **Step size or stride:** The distance the convolution kernel moves each time.
    - **Padding:** The width of zeros filled around the original image.

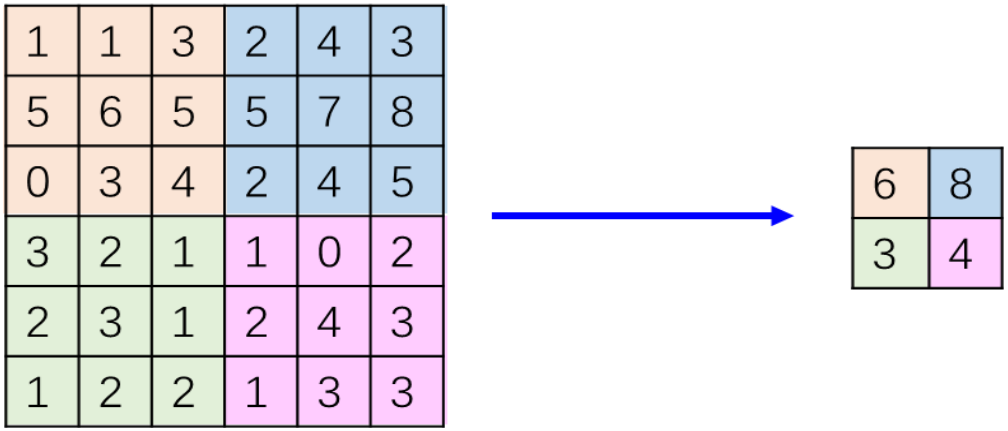
Convolutional Neural Networks

- **Activation Layer:**
  - Example:



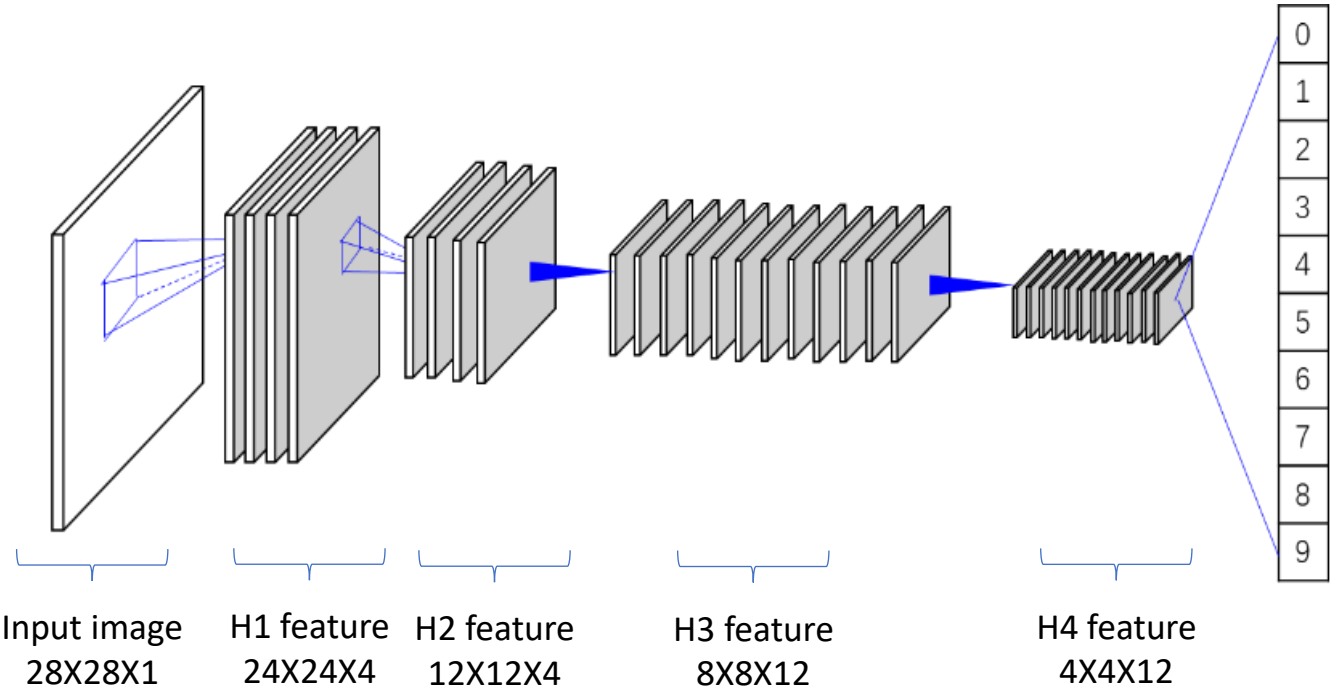
Convolutional Neural Networks

- Pooling Layer:
  - Pooling or aggregation, categorized by operation type, is usually divided into: Max Pooling, Average Pooling, and Sum Pooling. They respectively extract the maximum, average, and sum of the feature values within the receptive field as the output; the most commonly used is Max Pooling.
  - Example of Max Pooling:



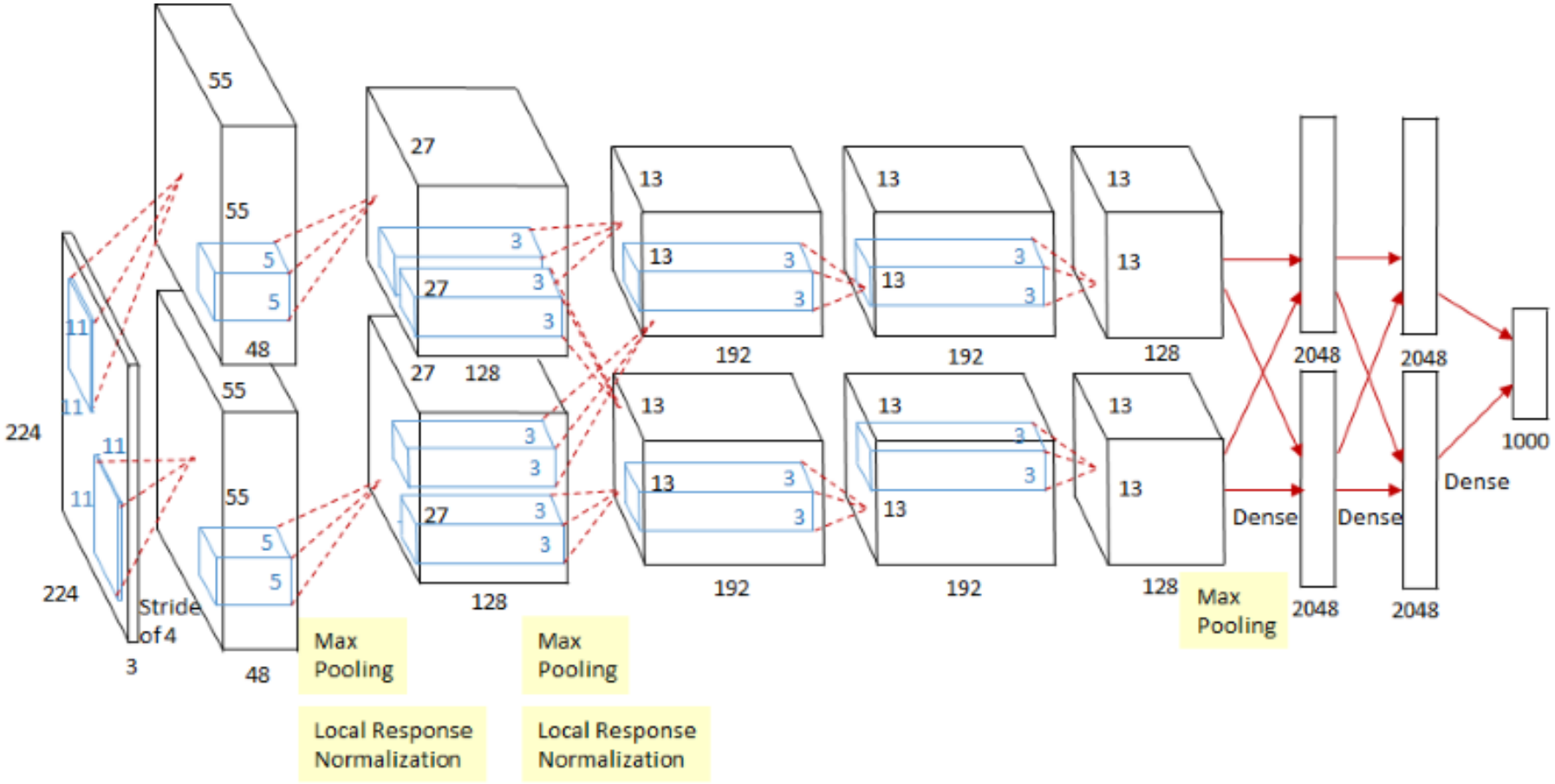
Convolutional Neural Networks

- **LeNet-1:**
  - Designed for handwritten character tasks; later outperformed by SVM.



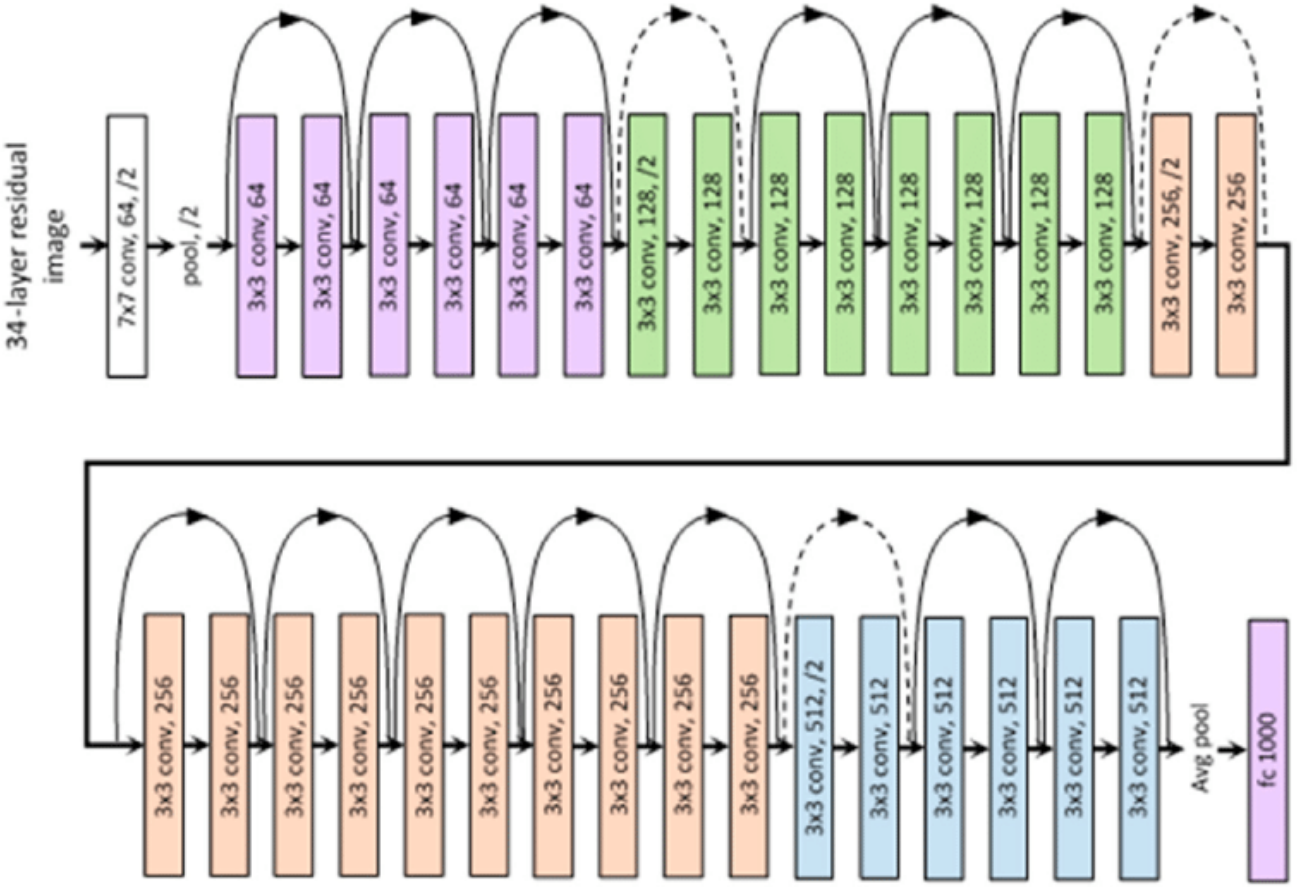
Convolutional Neural Networks

- **AlexNet:**



Convolutional Neural Networks

- **ResNet:**

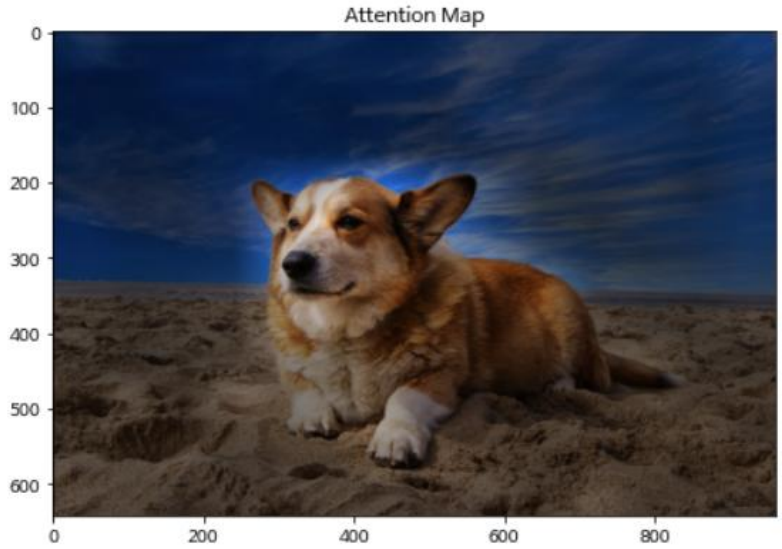


Transformer

Attention

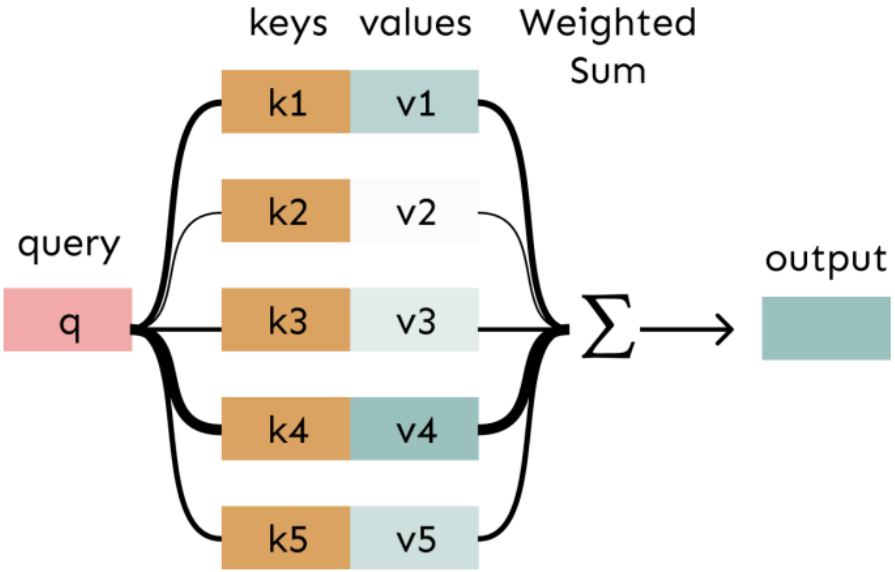
- Attention is a way of expressing the **importance of internal information** within data.





Attention

- Can be viewed as a **similarity-based lookup table**.



Step 1: Calculate the similarity between query and key:

$$e_{ij} = \mathbf{q}_i^\top \mathbf{k}_j$$

Step 2: Normalize the similarity:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_k \exp(e_{ik})}$$

Step 3: Compute the weighted sum of values:

$$o_i = \sum_j \alpha_{ij} v_i$$

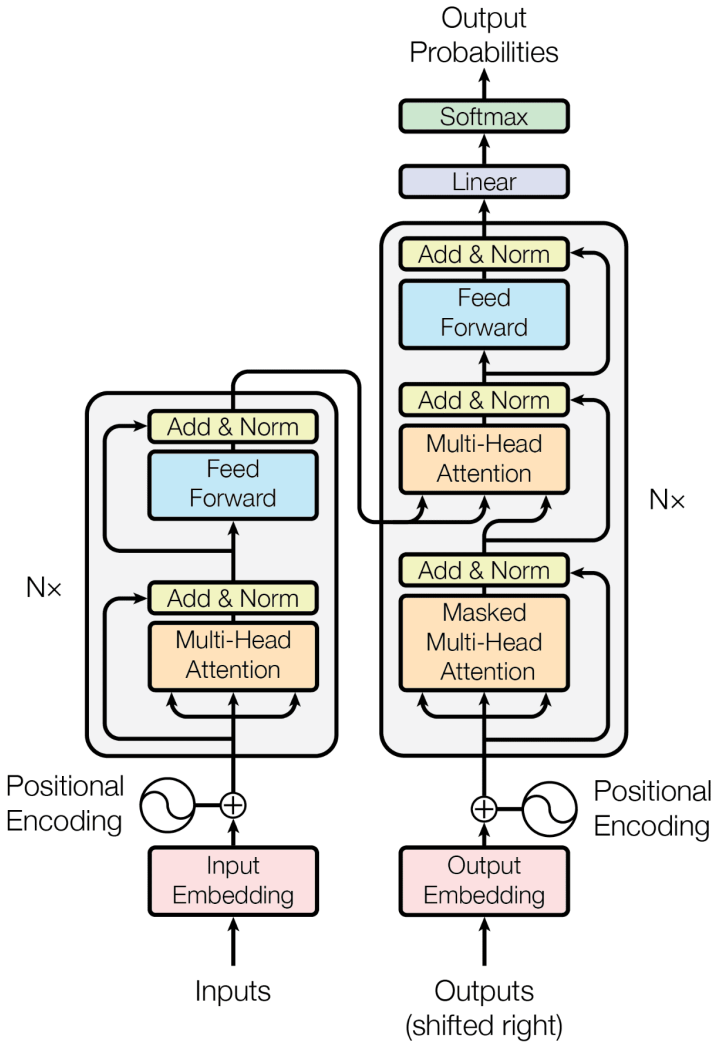
Transformer

Core Module: Attention

- The Transformer completely abandons traditional CNNs and RNNs; its entire networ architecture consists solely of attention mechanisms.

Encoder-Decoder Structure:

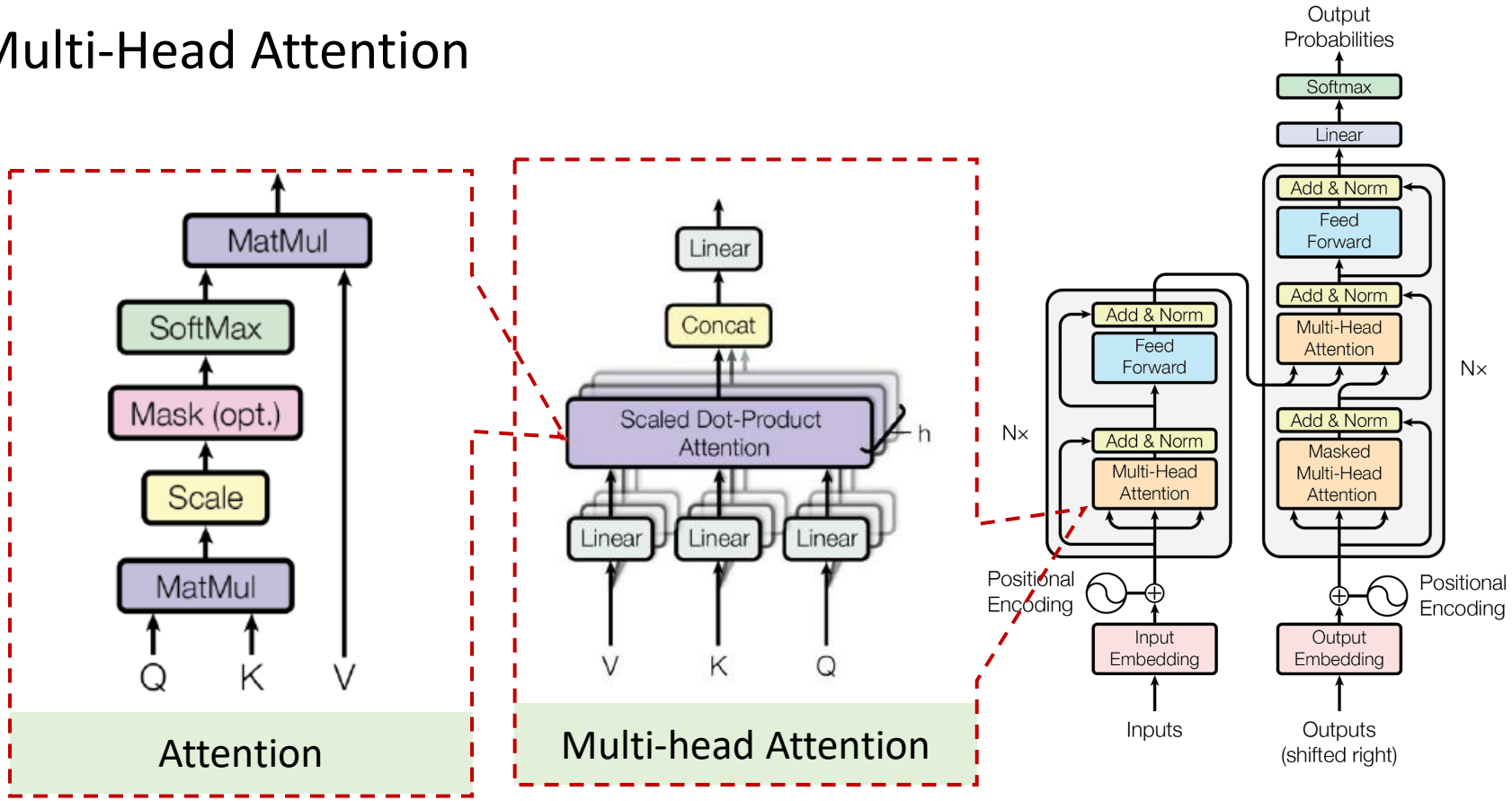
- The encoder component transforms the inpu sequence into hidden layer features.
- The decoder transforms the hidden layer features into an output sequence.



Transformer

Multi-head attention layer: Modeling semantic relationships between tokens at arbitrary distances

- Scaled Dot-Product Attention
- Multi-Head Attention



Transformer

Multi-Head Attention

- Scaled Dot-Product Attention:
  - Project input $\mathbf{X}$ into $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ matrix:

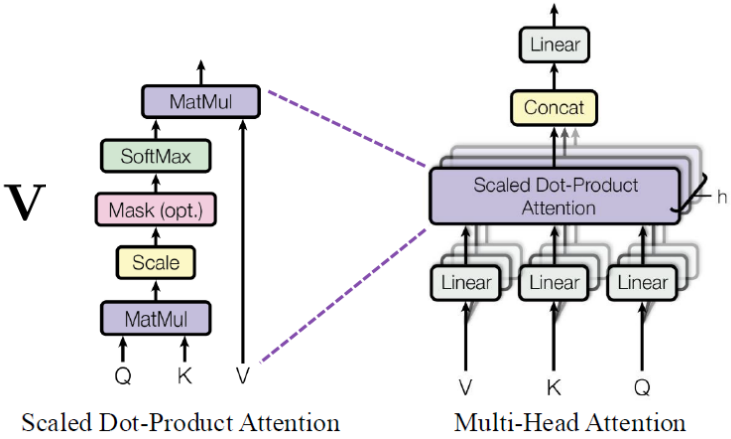
$$\mathbf{Q} = \mathbf{X}\mathbf{W}^Q, \mathbf{K} = \mathbf{X}\mathbf{W}^K, \mathbf{V} = \mathbf{X}\mathbf{W}^V$$

  - Attention Calculation:

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

- Multi-Head Attention:
  - Multi-head concatenation:

$$\text{MHA} = \text{Concat}(\text{head}_1, \text{head}_2, \cdots, \text{head}_n)\mathbf{W}$$

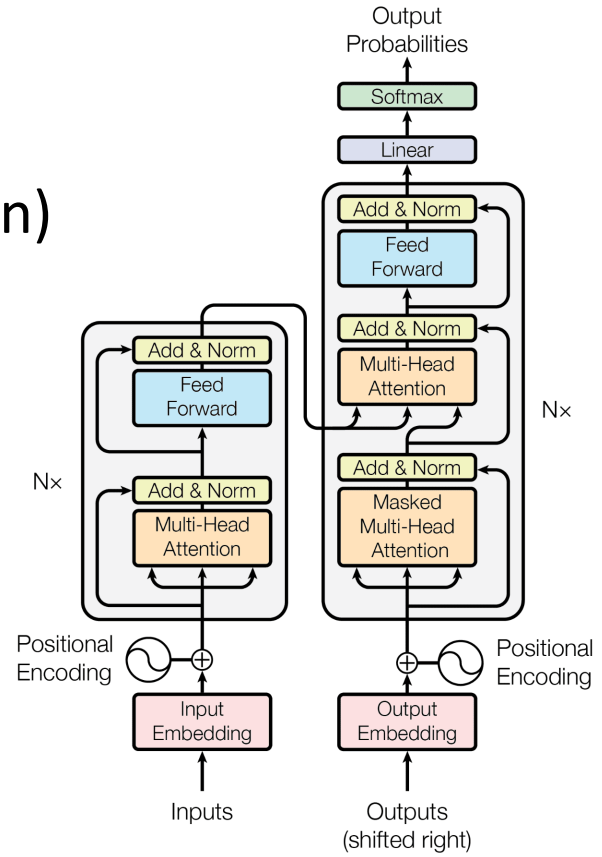


Transformer

Input Encoding

$$\mathbf{x}_t = \mathbf{v}_t + \mathbf{p}_t$$

- $\mathbf{v}_t$: word embedding (semantic information)
  - Tokens are mapped into word vectors via the embedding module
- $\mathbf{p}_t$: Positional Encoding (Positional Information)



Thanks