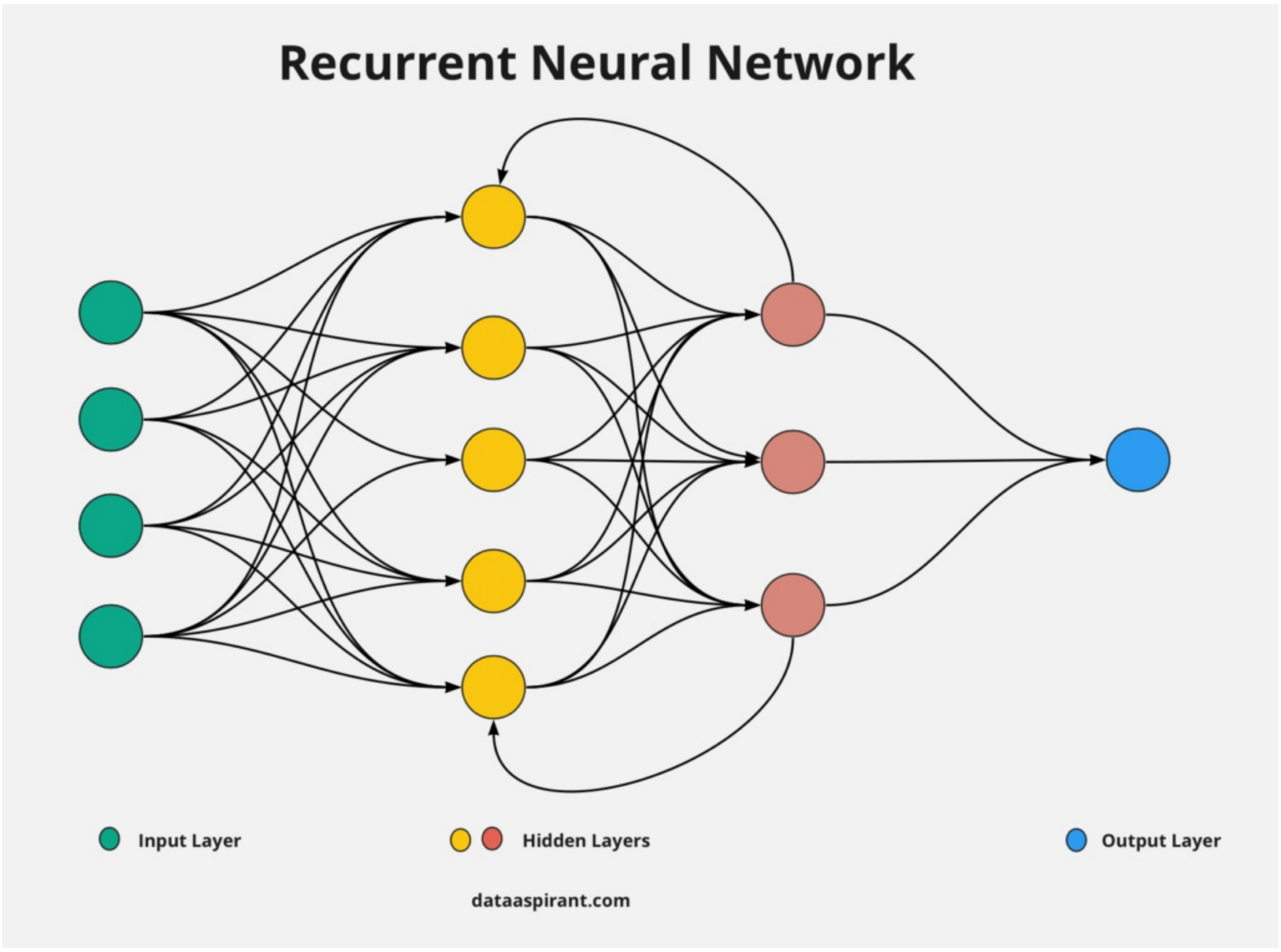
Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM)

Understanding Recurrent Neural Networks (RNN)

Recurrent Neural Networks are a specialized class of neural networks designed to handle sequential and time-series data by maintaining an internal memory that allows them to remember previous inputs. Unlike traditional feedforward neural networks where information flows in one direction, RNNs have connections that loop back, enabling them to pass information from one step in the sequence to the next.



RNN Architecture and Working Mechanism

The fundamental structure of an RNN consists of an input layer, one or more hidden layers, and an output layer. At each time step t, the network receives an input x_t and produces an output y_t, while also updating its hidden state h_t.

The hidden state is calculated using the formula:

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

where:

- h_t is the current hidden state
- h_{t-1} is the previous hidden state
- x_t is the current input
- W_{hh} and W_{xh} are weight matrices

The output is then computed as:

$$y_t = W_{hy} \cdot h_t$$

This recurrent connection allows the network to maintain a form of memory, making it suitable for tasks like language modeling, speech recognition, and time series prediction.

Training RNNs: Backpropagation Through Time

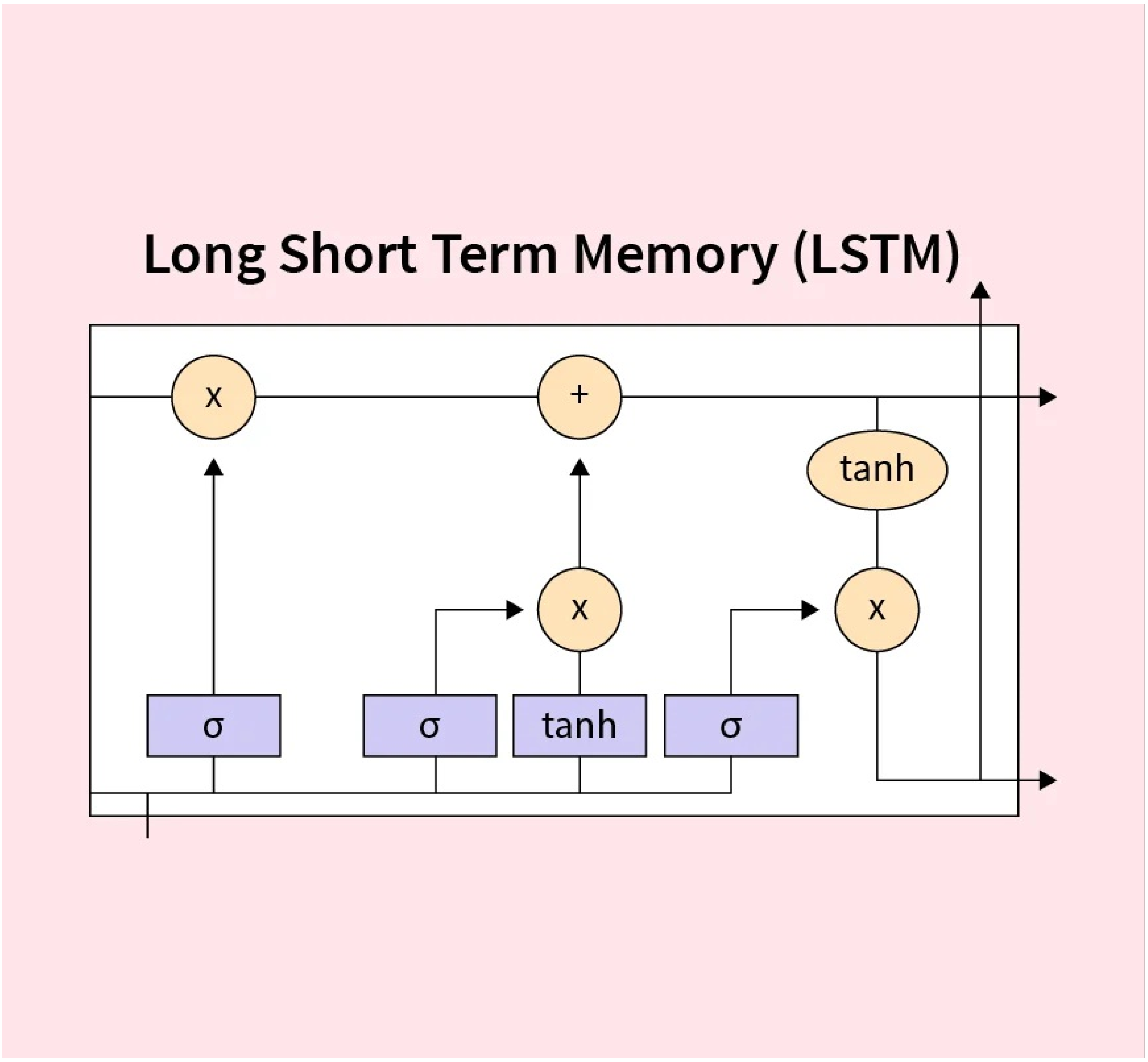
RNNs are trained using **Backpropagation Through Time (BPTT)**, which unrolls the network across time steps and calculates gradients by rolling back through the sequence. This process adjusts weights to minimize prediction error across the entire sequence.

Limitations of Standard RNNs

While RNNs excel at capturing short-term dependencies, they struggle with long sequences due to the **vanishing gradient problem**. During backpropagation, gradients can become extremely small (vanish) or large (explode) as they propagate through many time steps, making it difficult for the network to learn long-term dependencies.

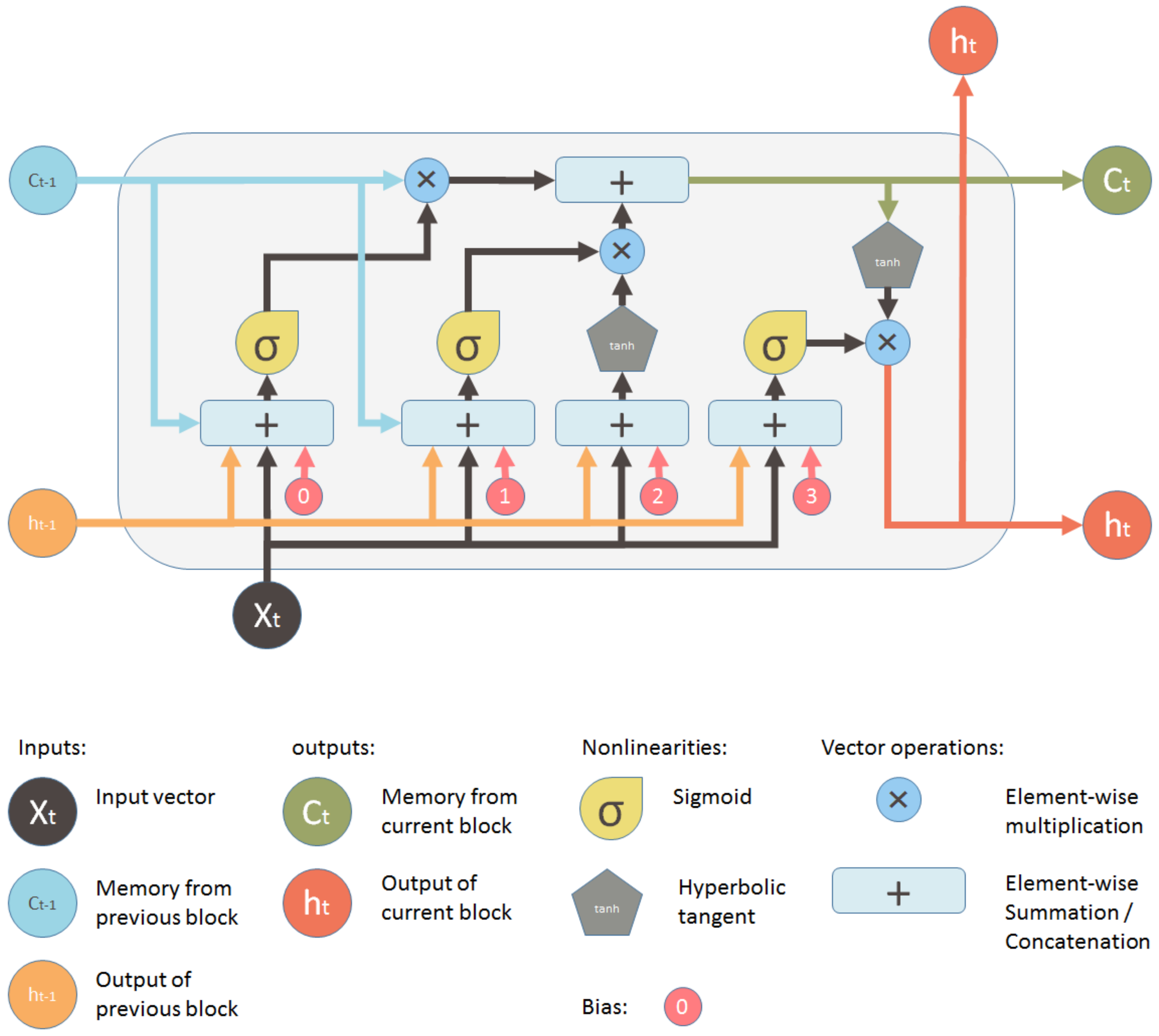
Long Short-Term Memory (LSTM) Networks

LSTM networks are an advanced variant of RNNs specifically engineered to overcome the vanishing gradient problem and effectively capture long-term dependencies in sequential data.



LSTM Architecture: The Cell State and Gates

The key innovation in LSTMs is the introduction of a **cell state** that acts as a conveyor belt, allowing information to flow through the network with minimal modifications. This cell state is regulated by three specialized gates that control information flow:



1. Forget Gate

The forget gate determines what information from the previous cell state should be discarded. It processes the previous hidden state h_{t-1} and current input x_t through a sigmoid activation function:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

The output ranges from 0 (completely forget) to 1 (completely retain). Values close to 0 indicate that the information should be forgotten, while values near 1 mean the information should be kept for future use.

2. Input Gate

The input gate controls what new information should be added to the cell state. This process involves two steps:

First, a sigmoid layer decides which values to update:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

Second, a tanh layer creates a vector of candidate values C_t that could be added to the cell state:

$$C_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

These two components work together to determine what new information enters the memory.

3. Cell State Update

The cell state is updated by combining the forget and input gate operations:

$$C_t = f_t \odot C_{t-1} + i_t \odot C_t$$

where ⊙ represents element-wise multiplication. This equation first multiplies the old cell state by the forget gate output (removing unwanted information), then adds the new candidate information scaled by the input gate.

4. Output Gate

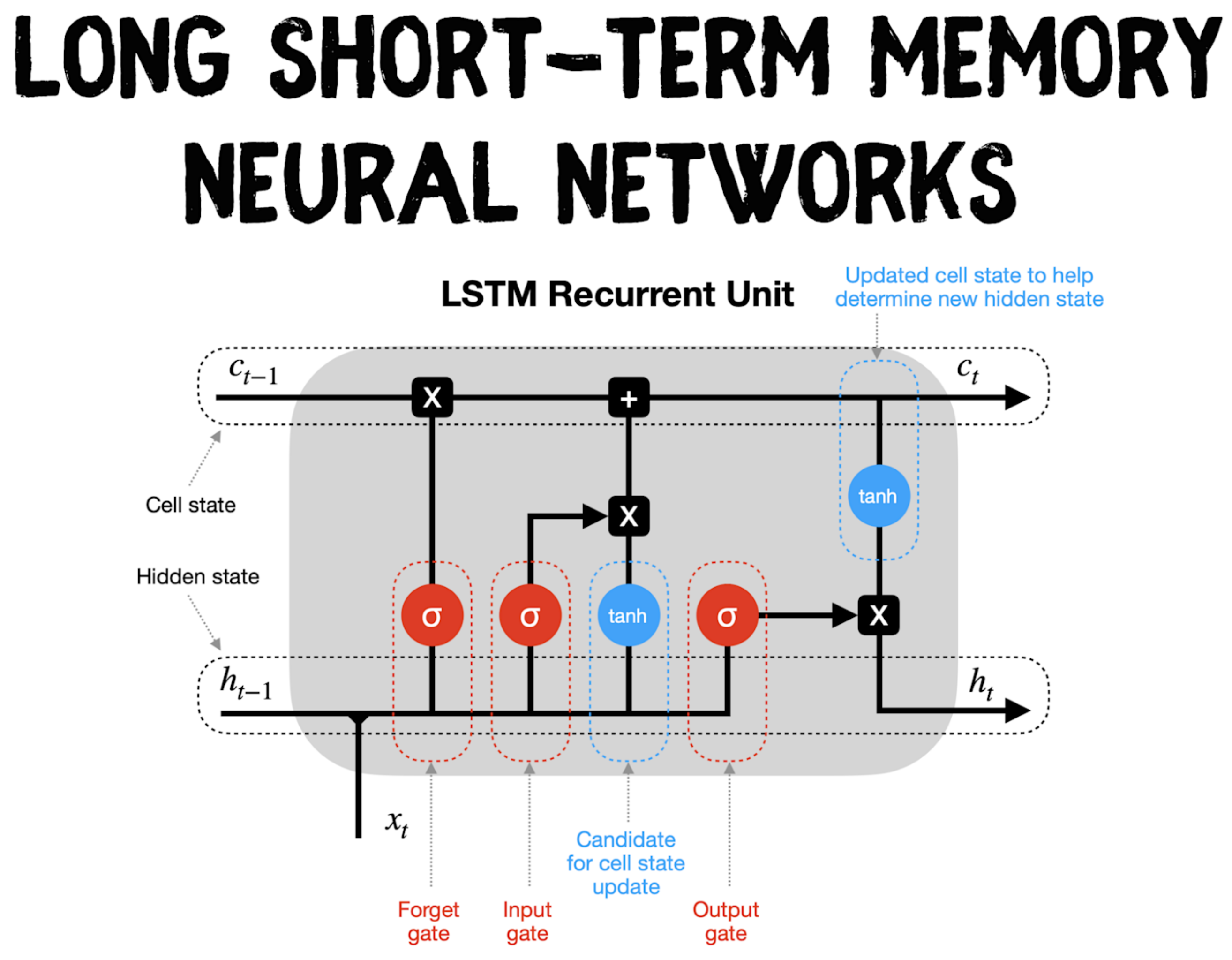
The output gate determines what information from the cell state should be passed as the hidden state to the next time step:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

The hidden state is then calculated by:

$$h_t = o_t \odot \tanh(C_t)$$

The cell state passes through a tanh function to squash values between -1 and 1, then gets filtered by the output gate.



LONG SHORT-TERM MEMORY NEURAL NETWORKS
LSTM Recurrent Unit
Updated cell state to help determine new hidden state
c_{t-1}
c_t
Cell state
Hidden state
h_{t-1}
h_t
x_t
X
+
X
X
σ
σ
tanh
σ
tanh
Candidate for cell state update
Forget gate
Input gate
Output gate

How LSTM Solves the Vanishing Gradient Problem

The cell state in LSTM provides a direct pathway for gradients to flow backward through time without significant attenuation. By allowing information to pass through with only minor linear interactions, LSTMs maintain gradient flow across many time steps, enabling the network to learn dependencies spanning hundreds of time steps.

The gating mechanism uses element-wise multiplication with sigmoid outputs (values between 0 and 1), which provides a differentiable way to control information flow suitable for backpropagation. This analog storage system allows the network to selectively retain or discard information, making training much more stable than standard RNNs.

Applications and Advantages

LSTMs have become the standard choice for sequential modeling tasks including:

- **Natural Language Processing**: Machine translation, text generation, sentiment analysis
- **Speech Recognition**: Converting spoken language to text[^5][^6]
- **Time Series Forecasting**: Stock price prediction, weather forecasting
- **Video Analysis**: Action recognition, video captioning

⁂