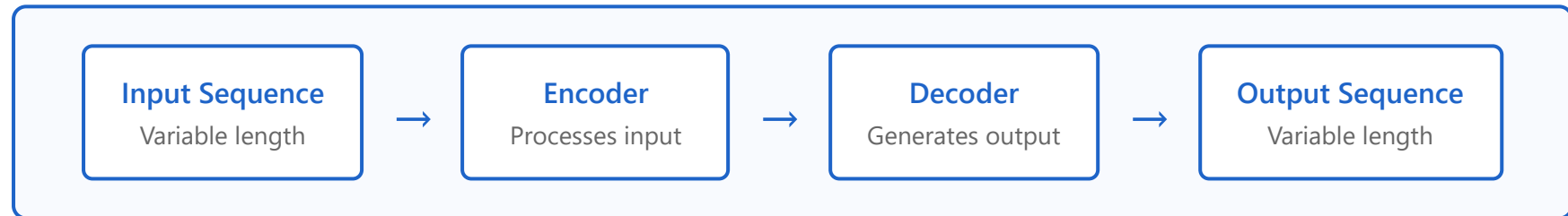


Sequence-to-Sequence Models

Maps input sequence to output sequence
Different lengths allowed (input \neq output)



Key Feature

- Handles variable-length sequences
- End-to-end learning



Applications

- Machine Translation
- Text Summarization
- Conversation Systems

Bottleneck Challenge

Entire input compressed to fixed vector → Attention mechanism addresses this



Step-by-Step Process

1

Input Encoding

Encoder RNN processes input sequence token by token, updating hidden state

2

Context Vector

Final encoder hidden state becomes context vector (fixed representation)

3

Decoder Initialization

Context vector initializes decoder RNN's hidden state

4

Sequential Generation

Decoder generates output tokens one at a time, using previous output as next input

5

Termination

Process continues until special end-of-sequence token is generated



Numerical Example with 3D Vectors

Weight Matrices (Encoder)

W (input weight, 3×3): $\begin{bmatrix} 0.5 & 0.2 & 0.3 \\ 0.1 & 0.6 & 0.4 \\ 0.3 & 0.2 & 0.7 \end{bmatrix}$

U (hidden weight, 3×3): $\begin{bmatrix} 0.4 & 0.3 & 0.2 \\ 0.2 & 0.5 & 0.3 \\ 0.3 & 0.2 & 0.6 \end{bmatrix}$

Input Sequence

$x_1 = [0.5, 0.3, 0.8] \rightarrow x_2 = [0.2, 0.7, 0.4]$

Encoder Step 1

$h_0 = [0, 0, 0]$ (initial hidden state)

$W \cdot x_1 = \begin{bmatrix} 0.5 & 0.2 & 0.3 \\ 0.1 & 0.6 & 0.4 \\ 0.3 & 0.2 & 0.7 \end{bmatrix} \cdot [0.5, 0.3, 0.8] = [0.55, 0.47, 0.77]$

$U \cdot h_0 = [0, 0, 0]$

$h_1 = \tanh([0.55, 0.47, 0.77]) \approx [0.50, 0.44, 0.65]$



Encoder Step 2

$$\mathbf{W} \cdot \mathbf{x}_2 = [[0.5, 0.2, 0.3], [0.1, 0.6, 0.4], [0.3, 0.2, 0.7]] \cdot [0.2, 0.7, 0.4] = [0.38, 0.58, 0.48]$$

$$\mathbf{U} \cdot \mathbf{h}_1 = [[0.4, 0.3, 0.2], [0.2, 0.5, 0.3], [0.3, 0.2, 0.6]] \cdot [0.50, 0.44, 0.65] = [0.46, 0.42, 0.61]$$

$$\mathbf{h}_2 = \tanh([0.38 + 0.46, 0.58 + 0.42, 0.48 + 0.61]) = \tanh([0.84, 1.00, 1.09]) \approx [0.69, 0.76, 0.80]$$

$$\text{Context Vector } \mathbf{c} = \mathbf{h}_2 = [0.69, 0.76, 0.80]$$

Weight Matrices (Decoder)

$$\mathbf{W}' \text{ (input weight, } 3 \times 3\text{): } [[0.6, 0.3, 0.2], [0.2, 0.5, 0.4], [0.4, 0.3, 0.5]]$$

$$\mathbf{U}' \text{ (hidden weight, } 3 \times 3\text{): } [[0.5, 0.2, 0.3], [0.3, 0.6, 0.2], [0.2, 0.3, 0.7]]$$

\mathbf{V} (output weight, vocab_size \times 3): maps hidden to output vocabulary

Decoder Step 1

$$\mathbf{s}_0 = [0.69, 0.76, 0.80] \text{ (initialized with context)}$$

$\mathbf{V} \cdot \mathbf{s}_0 \rightarrow \text{softmax} \rightarrow y_1$ (output token, e.g., word embedding)

$$\text{Assume } y_1 = [1, 0, 0] \text{ (one-hot encoded)}$$

$$\mathbf{s}_1 = \tanh(\mathbf{W}' \cdot y_1 + \mathbf{U}' \cdot \mathbf{s}_0) = \tanh([0.6, 0.2, 0.4] + [0.82, 0.84, 0.91]) \approx [0.77, 0.72, 0.80]$$

Decoder Step 2

$\mathbf{V} \cdot \mathbf{s}_1 \rightarrow \text{softmax} \rightarrow y_2$ (next output token)

$$\mathbf{s}_2 = \tanh(\mathbf{W}' \cdot y_2 + \mathbf{U}' \cdot \mathbf{s}_1) \rightarrow \text{continue...}$$

Process continues until **<END>** token is generated