

# Class Notes: Attention Mechanisms in Neural Networks

## 1. Bahdanau (Additive) Attention

For each decoder hidden state  $s_t$ , the attention mechanism computes a context vector  $c_t$  as a weighted sum of encoder hidden states  $h_s$ .

Keys: Encoder hidden states  $h_s$

Queries: Decoder hidden state  $s_t$

Values: Encoder hidden states  $h_s$

$$e_{ts} = v_a^\top \tanh(W_q s_t + W_k h_s)$$

$$\alpha_{ts} = \frac{\exp(e_{ts})}{\sum_j \exp(e_{tj})}$$

## Attention Mechanisms

### 1. Bahdanau (Additive) Attention

For each decoder hidden state  $s_t$ , attention mechanism computes context vector  $c_t$  as a weighted sum of encoder hidden states  $h_1, \dots$ .

Keys	Queries	Values
$c_t$	query	Encoder hidden states $h_s$
values	values	Encoder hidden states $h_1, \dots$

Scoring function  $e_{ts} = \frac{1}{\sum_j \exp(e_{tj})} \exp(e_{ts})$

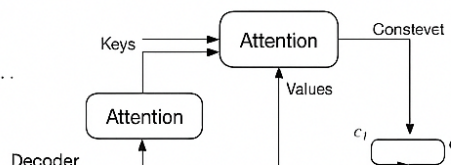


Figure 1: Bahdanau Attention flow diagram.

## 2. Luong (Multiplicative) Attention

The “global” variant uses the decoder hidden state from the previous timestep.

Keys: Encoder hidden states  $h_s$

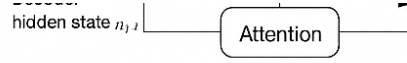
Queries: Decoder hidden state  $s_{t-1}$

Values: Encoder hidden states  $h_s$

$$e_{ts} = h_t^\top h_s \quad \text{or} \quad e_{ts} = h_t^\top W h_s$$

$$\alpha_{ts} = \frac{\exp(e_{ts})}{\sum_j \exp(e_{tj})}$$

$$\text{Attention weights } \alpha_{ts} = \frac{\exp(e_{ts})}{\sum_j \exp(e_{tj})}$$



## 2. Luong (Multiplicative) Attention

The “global” variant of Luong attention uses the decoder hidden state  $u_t$  previous timestep.

Keys	Queries	Values
$c_t$	query	Encoder hidden states $e$
values	values	Encoder hidden states $h, \dots$

$$\text{Scoring function } e_{ts} = h_t s_{t-1}$$

$$\text{Attention weights } \alpha_{ts} = \frac{\exp(e_{ts})}{\sum_j \exp(e_{tj})}$$

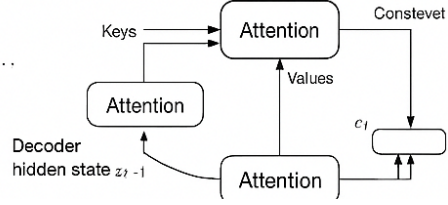


Figure 2: Luong Attention flow diagram.

## 3. Multi-Head Attention

Self-attention where  $Q, K, V$  are projected into  $h$  subspaces, processed in parallel, then concatenated.

$$Q_h = QW_h^Q, \quad K_h = KW_h^K, \quad V_h = VW_h^V$$

$$A_h = \text{softmax} \left( \frac{Q_h K_h^\top}{\sqrt{d_k}} \right), \quad O_h = A_h V_h$$

$$\text{Output} = \text{Concat}(O_1, O_2, \dots, O_h)$$

## 4. Multi-Head Attention

Self-attention. Aggregate sequence into single vector –using self-attention pooling.

Let the inputs be  $(x_1, \dots, x_L)$

$$K_{\eta} = x_1 W_{K\eta}, \quad Q_{\eta} = \frac{x_1 W_{Q\eta}}{\mathcal{Z}_T R_{L\eta}}$$

$$Q_{\eta} = x_1 W_{Q\eta}, \quad V_{\eta} = x_1 W_{V\eta}$$

Output of head  $h$   $\tilde{h} = x_i$

$$sc = \frac{e_{\eta_1}}{g_{\eta}}, \quad \alpha_s = \frac{\exp(qs_s)}{\sum_j \exp(qs_j)}$$

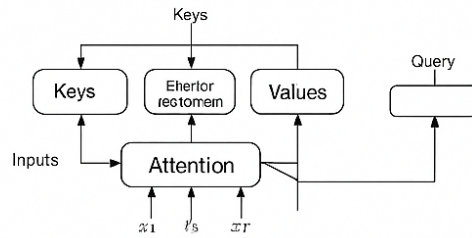


Figure 3: Multi-Head Attention computation from the provided diagram.

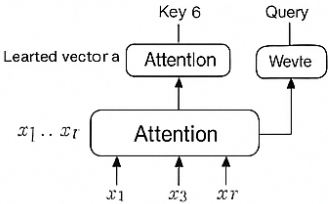
## 4. Self-Attention

Aggregates a sequence into a single vector via scaled dot-product.

#### 4. Self-Attention

Aggregate a sequence  $j$  into a single vector, self-attention pooling.

Keys	queries	Score	Values
$x_1$	$q$	Additive	$\text{Linear}(c) + xr$
Values	learned	Dot product	Scaled dot-prod



Feature	Keys	Score	Output	Score	Twist
Bahdanau $(h_1 \dots x_r)$	$x_1, \dots, w_1 \dots x_r$	Additive	Weighte	Weighted sum	Weighted sum
Luong $(x_1, \dots x_r)$	$x_1, \dots, x_1, x_r$	Dot-pred-act	Multi-head	Weighted sum	Multi-head

Figure 4: Self-Attention flow from the provided diagram.

#### 5. Summary Table

Variant	Q Source	K Source	Scoring	Use Case
Bahdanau	Decoder RNN	Encoder RNN	Additive MLP	Seq2Seq
Luong	Decoder RNN	Encoder RNN	Dot/General	Seq2Seq
Self-Attn	Same sequence	Same sequence	Scaled Dot	Transformers
Multi-Head	Same sequence	Same sequence	Scaled Dot (multi)	Rich relations