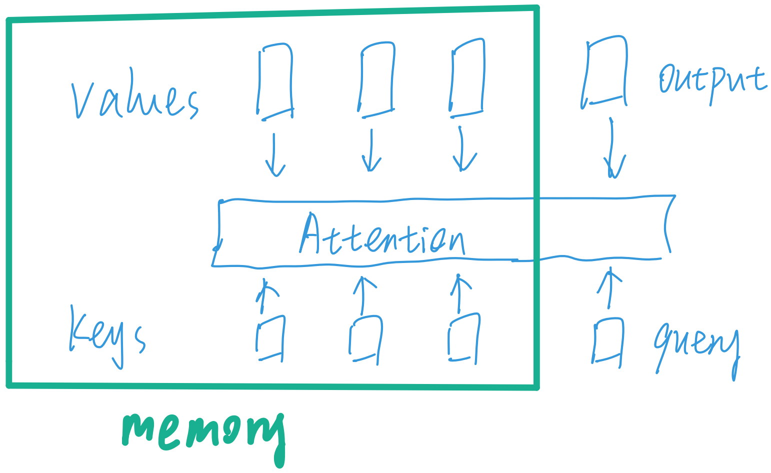
Attention Layer.



Memory: key-value pairs

Output: close to values whose keys similar to query

- Hard assignment is not differentiable due to the max operator.

$\therefore$ need to use soft assignment.

Assume query $q \in R^{d_q}$ and memory $(k_1, v_1) \cdots (k_n, v_n)$ $\begin{cases} k_n \in R^{d_k} \\ v_n \in R^{d_v} \end{cases}$

Compute n scores $a_1, \cdots a_n$ with $a_i = \alpha(q, k_i)$

attention weights: $b_1 \ldots b_n = \text{softmax}(a_1, \cdots a_n)$

output: $o = \sum_{i=1}^{n} b_i v_i$

Dot product attention.

$q, k_i \in R^d$, $\alpha(q, k) = \frac{1}{\sqrt{d}} \langle q, k \rangle$

m queries: $Q \in R^{m \times d}$ and n keys $K \in R^{n \times d}$: $\alpha(Q, K) = \frac{1}{\sqrt{d}} QK^T$

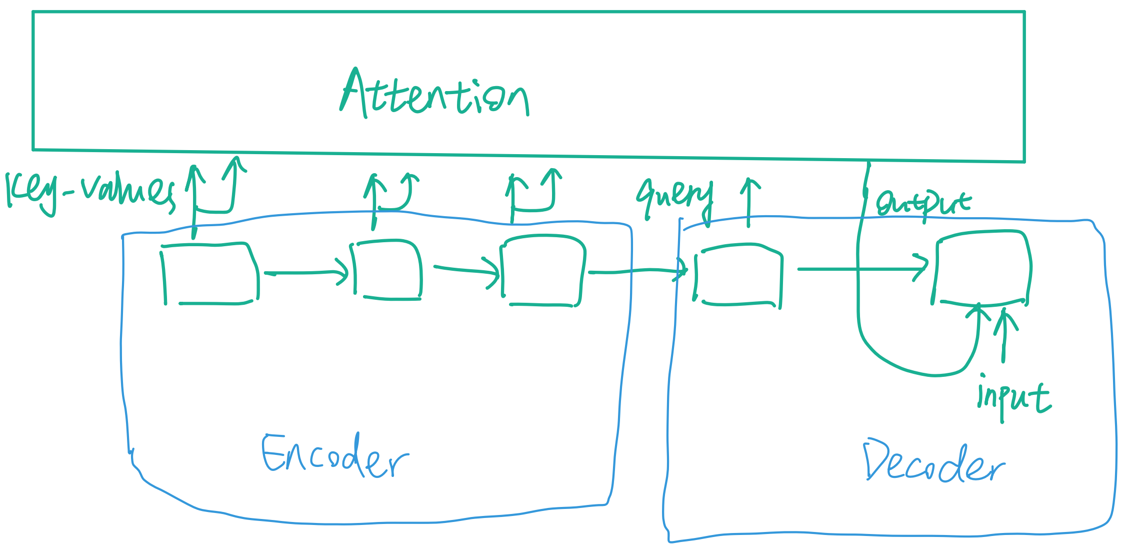
MLP Attention.

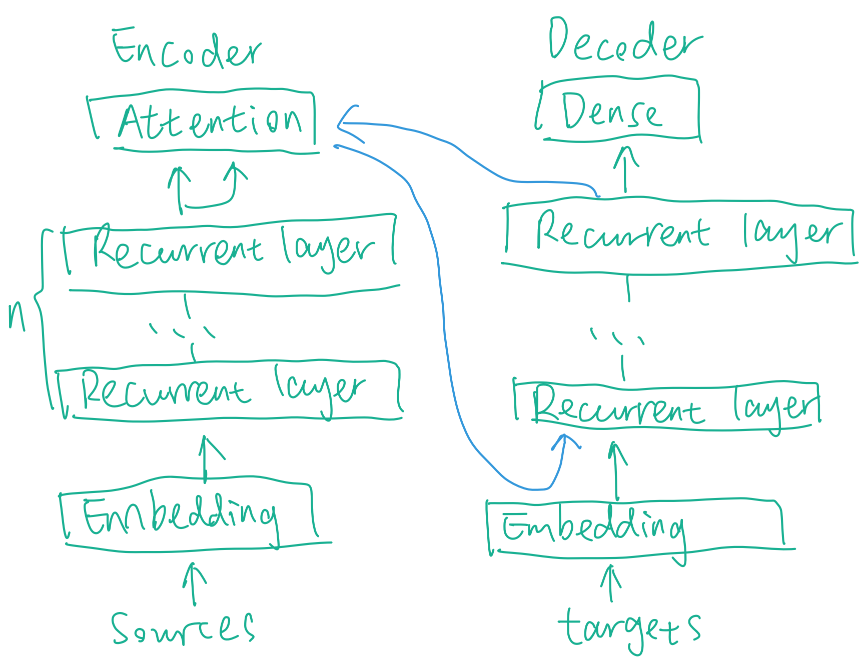
Learnable parameters $W_k \in \mathbb{R}^{h \times d_k}$, $W_q \in \mathbb{R}^{h \times d_q}$, $v \in \mathbb{R}^{h}$.

$$\alpha(k, q) = v^T \tanh(W_k k + W_q q) \in \mathbb{R}$$

i.e. concatenate key + query, feed into MLP, with hidden size h

Seq2Seq with attention

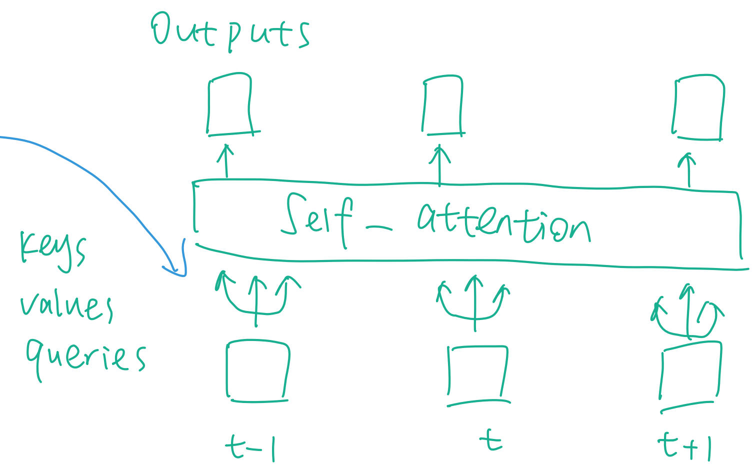




Transformer.

Self attention.

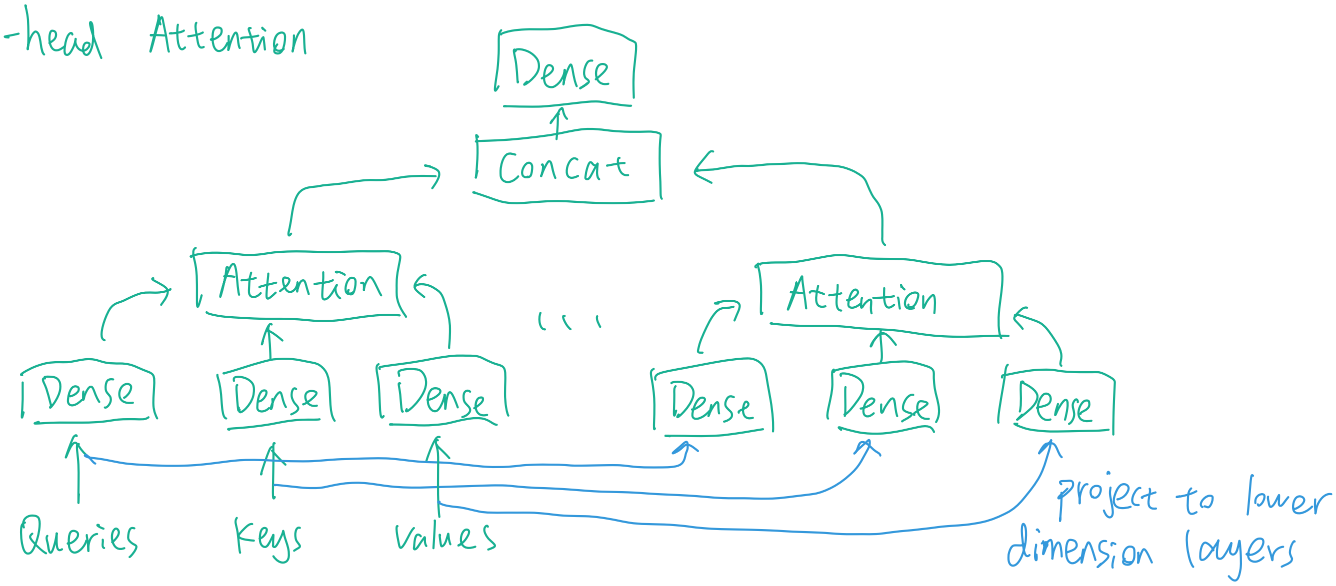
Copy key, value, query.



- Transformer uses attention layer to replace RNN layers.
- To generate n outputs, we can copy each input into a key, value, query

- No sequential information is preserved.
- Runs in parallel

Multi-head Attention



$$W_q^{(i)} \in R^{p_q \times d_q}, \quad W_k^{(i)} \in R^{p_k \times d_k}, \quad W_v^{(i)} \in R^{p_v \times d_v}$$

$$O^{(i)} = \text{attention}(W_q^{(i)} q, W_k^{(i)} k, W_v^{(i)} v)$$

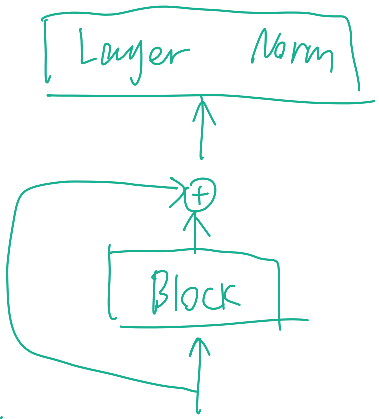
$$O = W_o \begin{bmatrix} O^{(1)} \\ \vdots \\ O^{(h)} \end{bmatrix}$$

Position-wise Feed-Forward Networks

- Reshape input (batch, seq. length, feature size) into (batch × seq. length, feature size).
- Apply a two layer MLP
- Reshape back to 3-D
- Equals to apply two (1,1) conv layers.

Add and Norm

- Layer norm is similar to batch norm.
- But mean and variances are calculated along the last dimension
- X.mean(axis=-1) instead of the first batch dimension in batch norm X.mean(axis=0)



Positional Encoding.

- Assume embedding output $X \in R^{l \times d}$ with shape (seq. length, embed dim)
- Create $P \in R^{l \times d}$ with

$$\begin{cases} P_{i,2j} = \sin(i/10000^{2j/d}) \\ P_{i,2j+1} = \cos(i/10000^{2jd}) \end{cases}$$

- Output $X + P$

Predicting

- Predict at time t
- Inputs of previous times as keys and values
- Input at time t as query, as well as key, value, to predict output.