RNN / Enc-Dec / Attention / Transformer

Neural-LM [2003]

1. Bengio et al., “A neural probabilistic language model“, NIPS 2000
2. Bengio et al., “A neural probabilistic language model”, JMLR 2003.

RNN-LM [2010]

T. Mikolov, M. Karafi´at, L. Burget, J. Cernock´y, and S. Khudanpur. “Recurrent neural network based language model”, In INTERSPEECH, pages 1045–1048, 2010.

Seq2seq learning (Encoder-Decoder) [2014]

Sutskever, I., Vinyals, O., & Le, Q. V., “Sequence to sequence learning with neural networks”, Advances in Neural Information Processing Systems (pp. 3104–3112), 2014

Attention [2015]

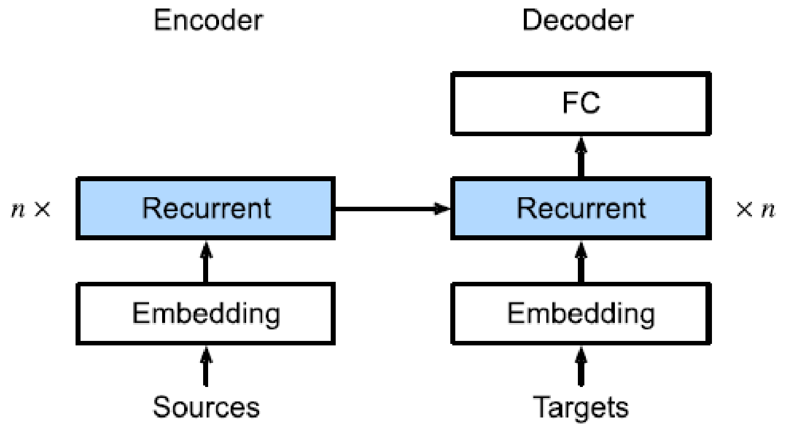
Bahdanau, D., Cho, K., & Bengio, Y., “Neural machine translation by jointly learning to align and translate”, Proc. ICLR 2015

Transformer [2017]

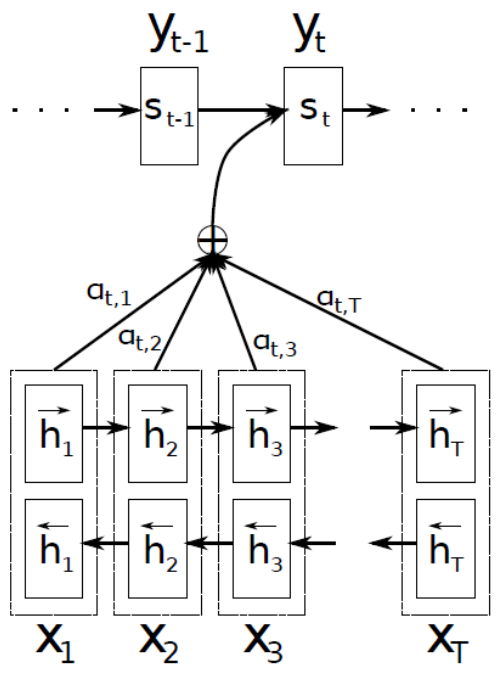
A. Vaswani et al., “Attention is all you need”, In NIPS, pages 6000–6010, 2017.

GPT [2018]

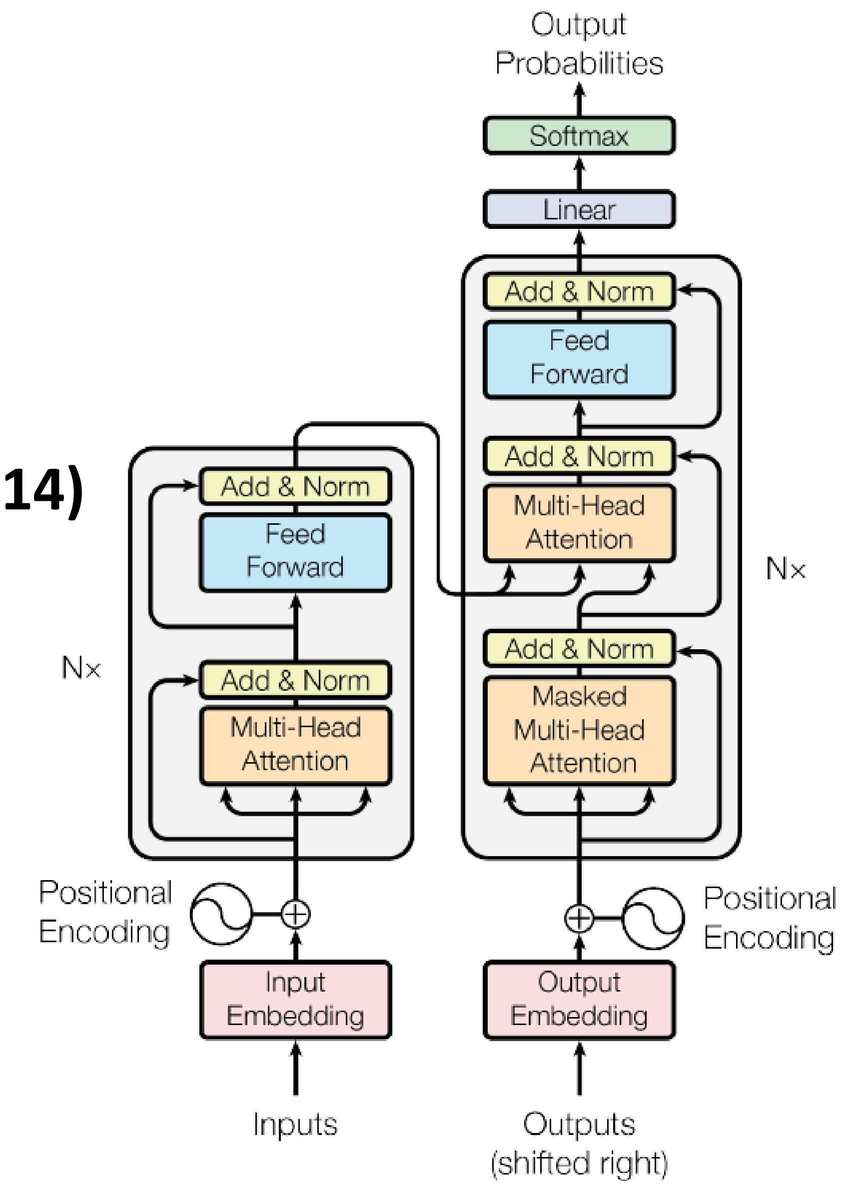
A. Radford, K. Narasimhan, T. Salimans and I. Sutskever, “Improving Language Understanding by Generative Pre-Training”, 2018 (arXiv)



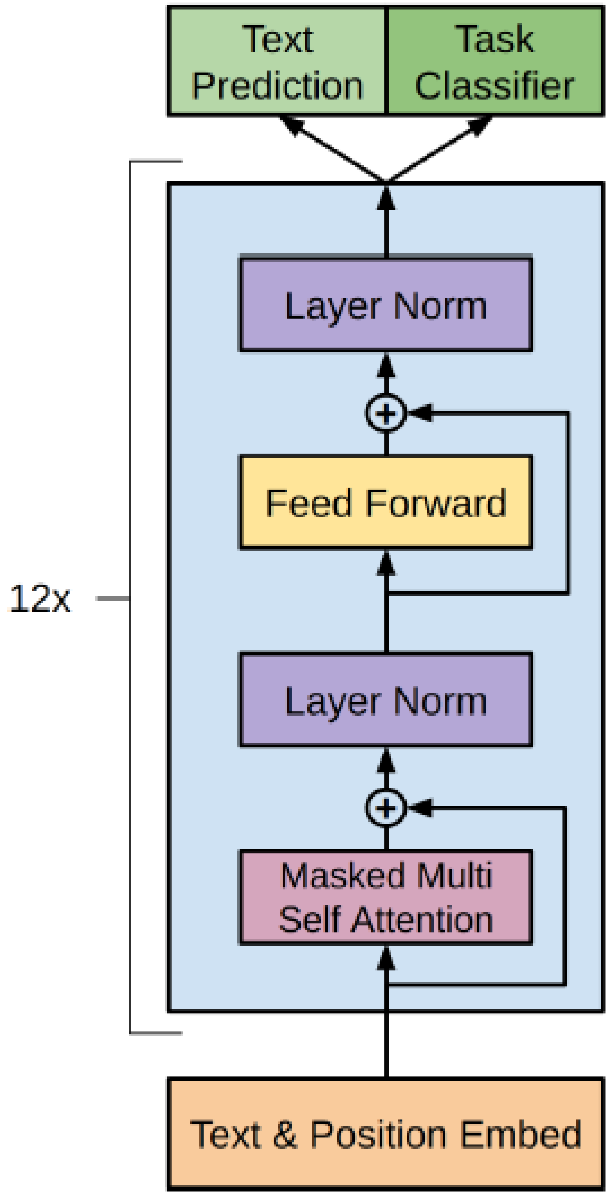
RNN Encoder-Decoder Model (2014)



Bahdanau Attention (2015)



Transformer Model (2017)

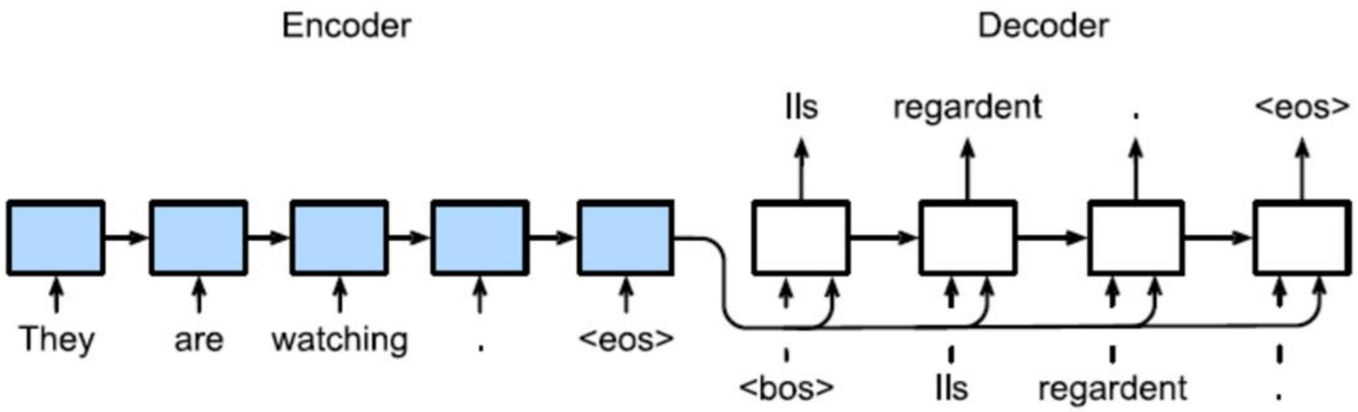


Decoder-only GPT (2018)

Encoder-Decoder Architecture: Sequence-to-Sequence Learning

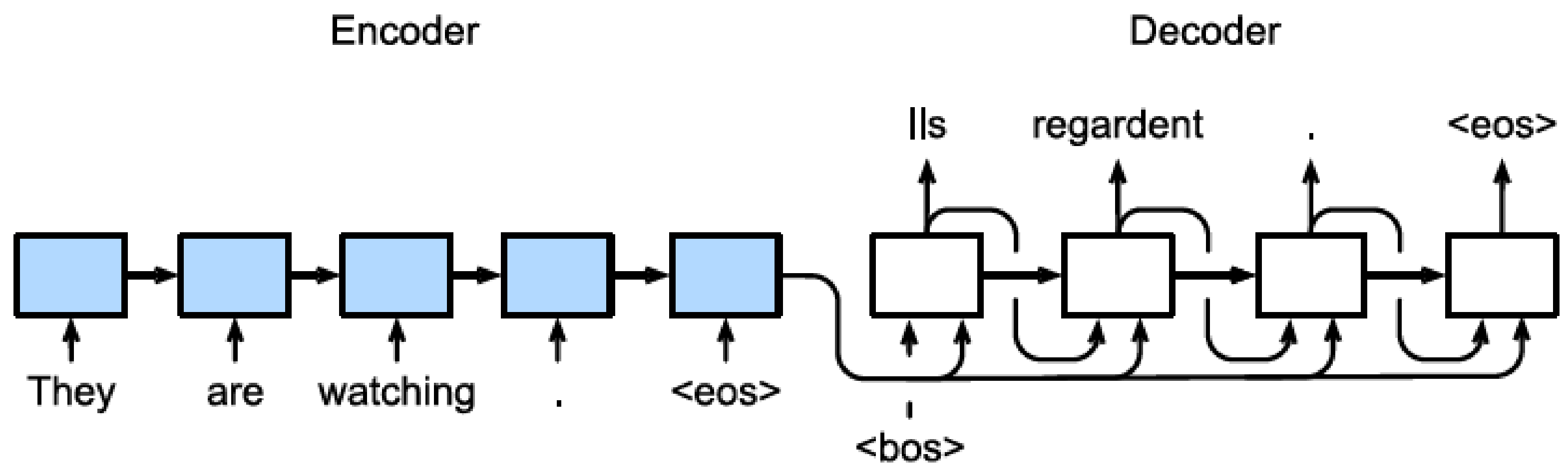
Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. Advances in Neural Information Processing Systems (pp. 3104–3112).

TRAINING: TEACHER FORCING



Sequence-to-sequence learning with an RNN encoder and an RNN decoder.

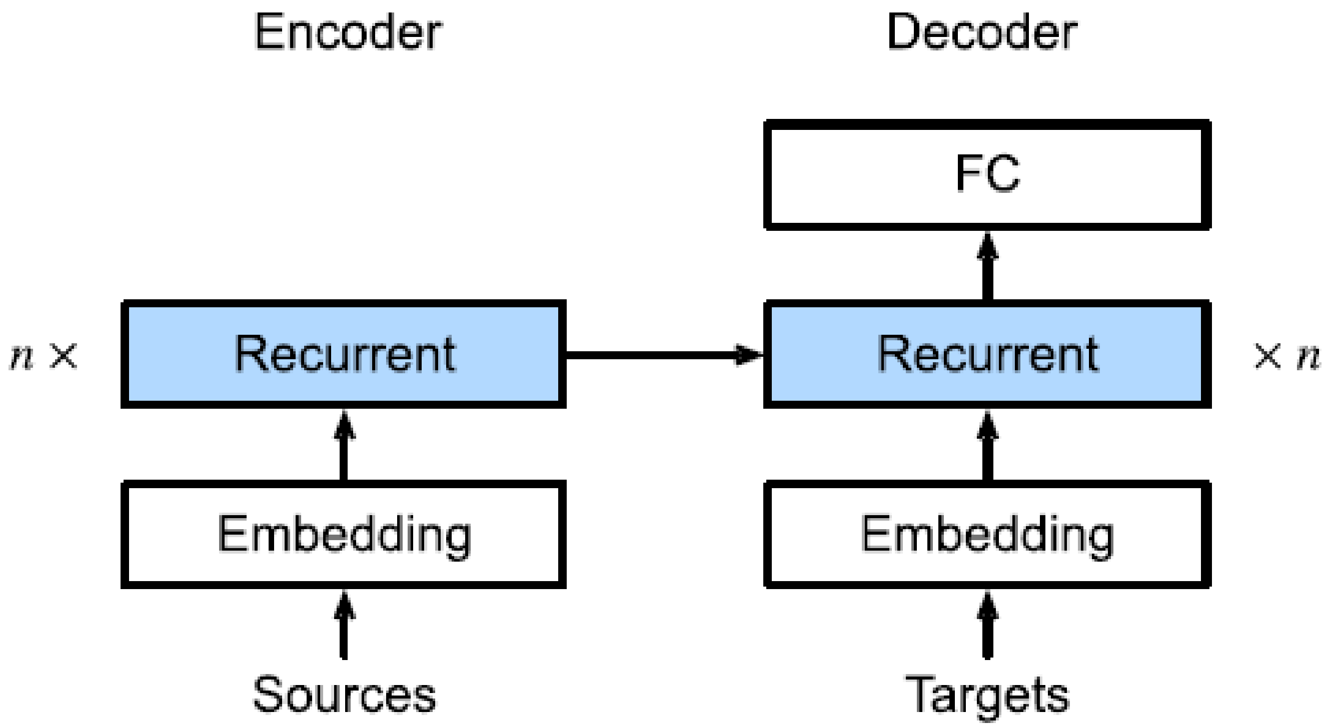
TEST: PREDICTION



Predicting the output sequence token by token using an RNN encoder–decoder.

- Predict the output sequence at each step: Predicted token from the previous time step is fed into the decoder as an input.
- One simple strategy: Sample whichever token that has been assigned by the decoder the highest probability when predicting at each step.
- As in training, at the initial time step the beginning-of-sequence (“<bos>”) token is fed into the decoder.
- When the end-of-sequence (“<eos>”) token is predicted, the prediction of the output sequence is complete

Encoder–Decoder for Sequence-to-Sequence Learning



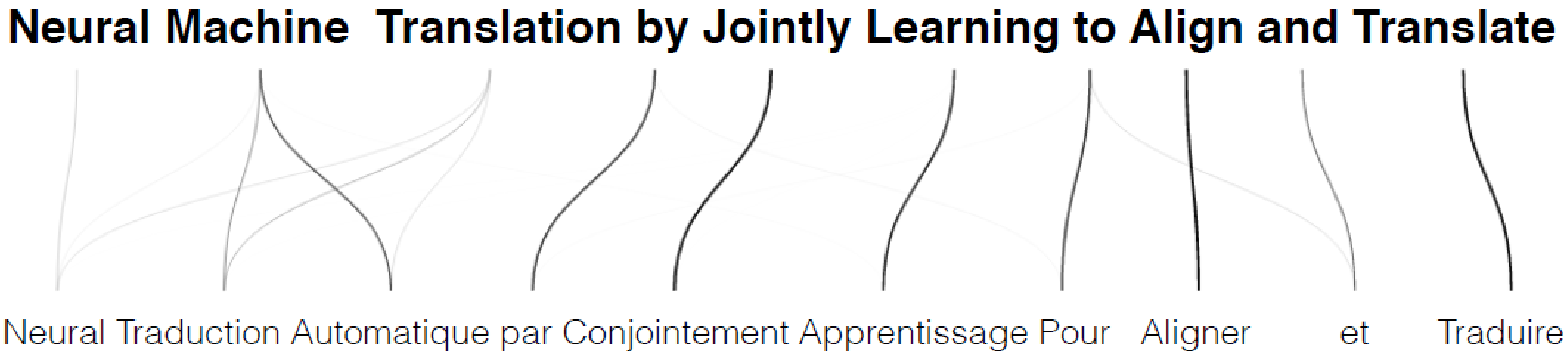
Layers in an RNN encoder–decoder model.

Attention [2015]

Bahdanau, D., Cho, K., & Bengio, Y.,

"Neural machine translation by jointly learning to align and translate",

Proc. ICLR 2015



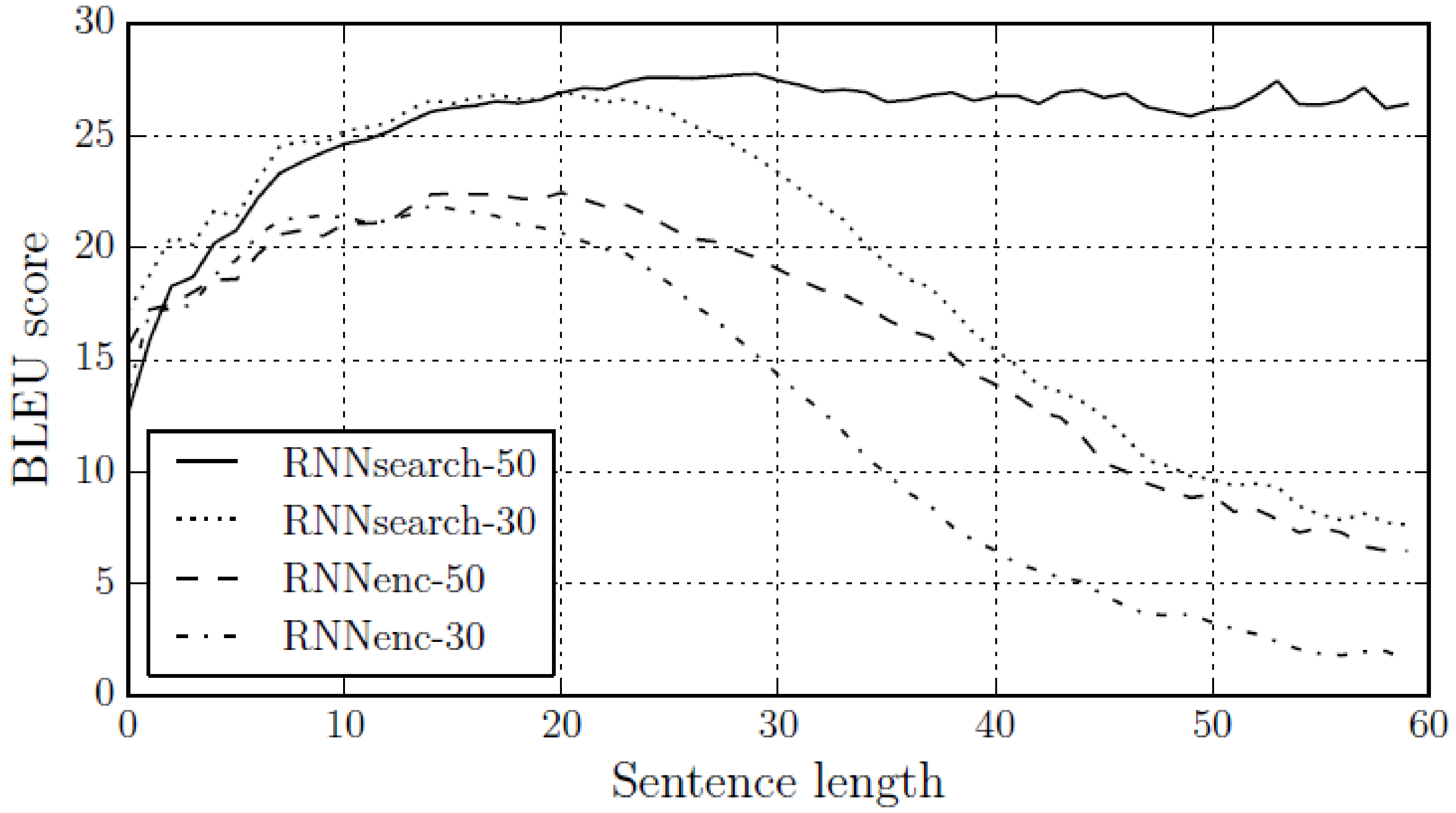
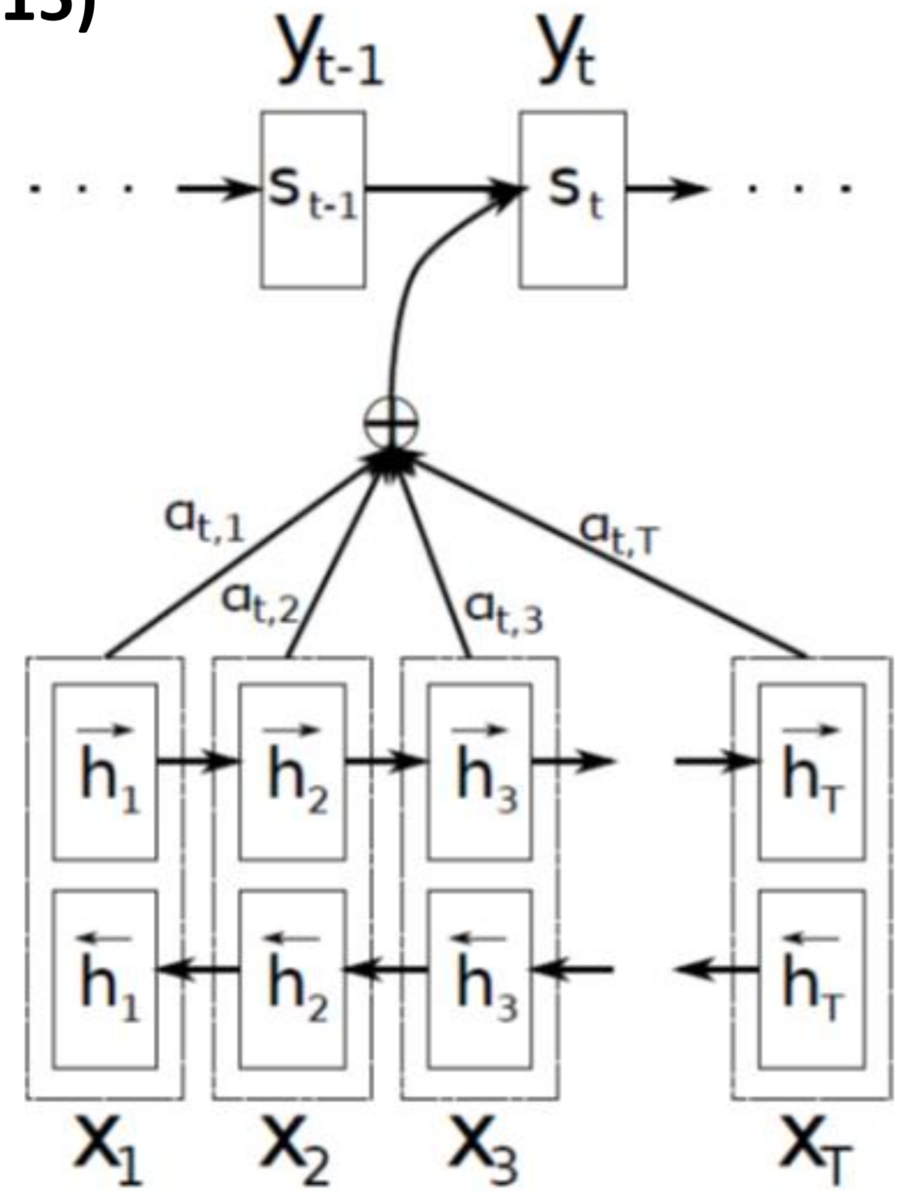


Figure 2: The BLEU scores of the generated translations on the test set with respect to the lengths of the sentences. The results are on the full test set which includes sentences having unknown words to the models.

Bahdanau Attention (2015)



ATTENTION

Query Q, Key K, Value V => [Q, K, V]

Database D : Tuples [Keys (k), values (v)]

Last name (k)	First name (v)
Zhang	Aston
Lipton	Zachary
Li	$\boxed{\text{Mu}}$
Smola	Alesc
Hu	Rachel.
⋮	⋮

Query

q = Li → Li → $\boxed{\text{Mu}}$ → Returned value

Query 'q' operates on (k, v) pairs.

Exact Match

Query q = Li

=> (Li, Mu)

Approximate Match.

q = Li

=> (Lipton, Zachary)

Attention Mechanism [Bahdanau (2014)]

$$D = \{(k_1, v_1), (k_2, v_2), \dots (k_m, v_m)\}$$

Database of 'm' tuples of (keys, values).

Let q : query.

Attention over D

$$\text{Attention}(q, D) = \sum_{i=1}^{m} \underbrace{\alpha(q, k_i)}_{\in \mathbb{R}} v_i \quad \text{—— (1)}$$

are scalar attention weights

Eqn (1): Attention Pooling.

: Linear Combination of "values" in the db D.

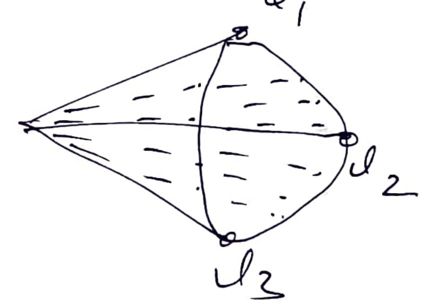
Attention: operation pays "particular attention" to the terms for which ~~weights~~ weight 'α' is large.

Special Cases

$$\text{Attention}(q, D) = \sum_{i=1}^{m} \alpha(q, k_i)\, v_i$$

1. $\alpha(q, k_i) \geq 0$ (non-negative)

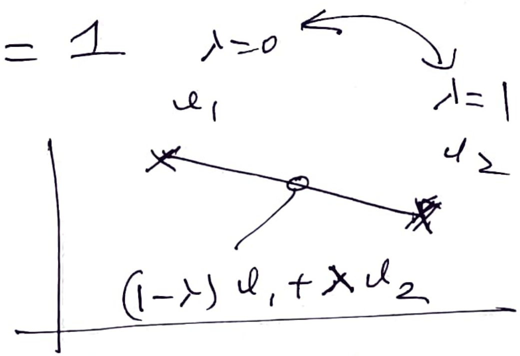
o/p is contained in the "convex-cone" spanned by values of v_i



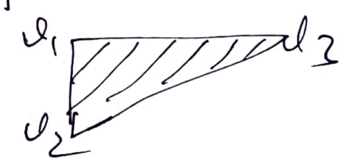
2. $\alpha(q, k_i) \Rightarrow$ Convex Combination

i.e. $\alpha(q, k_i) \geq 0$ & $\sum_{i=1}^{m} \alpha(q, k_i) = 1$

Attention(q, D): Lies on Simplex Surface



$$\boxed{\text{MOST COMMON SETTING IN DEEP LEARNING}}$$



3. Exactly one $\alpha(q, k_i) = 1$

rest - - - $= 0$

$\Rightarrow$ Traditional Database Querying

ie $q \Longrightarrow \underline{\underline{v_i \text{ for which } \alpha(q, k_i) = 1}}$

4. All weights are equal

$$\alpha(q, k_i) = \frac{1}{m} \quad \forall i$$

$\Rightarrow$ Averaging the entire Database D.

$\Rightarrow$ $\boxed{\text{AVERAGE POOLING IN DEEP LEARNING}}$

Common mechanism to ensure Case (2) | Convex Combination (5)

ie $\sum_{i=1}^{m} \alpha(q, k_i) = 1$

is to define

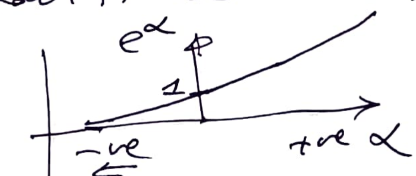
$$\alpha(q, k_i) = \frac{\alpha(q, k_i)}{\sum_j \alpha(q, k_j)}$$

normalized

and to ensure also $\alpha(q, k_i) \geq 0$

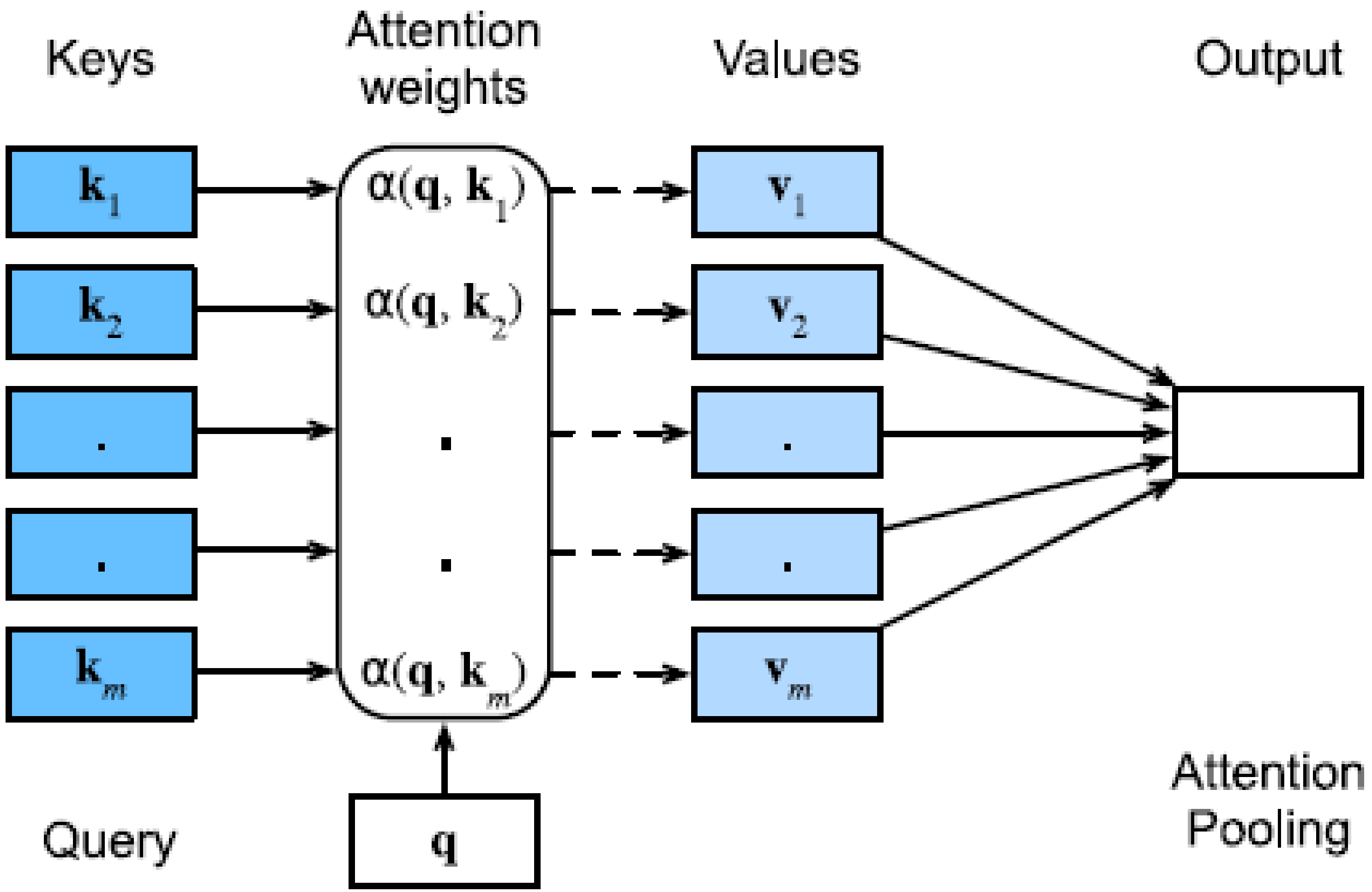
→ Let $\alpha(q, k_i)$ be "any" function, but render it as Case (2)

$$\alpha(q, k_i) = \frac{\exp[\alpha(q, k_i)]}{\sum_j \exp[\alpha(q, k_j)]}$$



- Differentiable
- Gradient never vanishes.

- SOFTMAX operation used for Multinomial models.



The attention mechanism computes a linear combination over values $\mathbf{v}_i$ via attention pooling, where weights are derived according to the compatibility between a query $\mathbf{q}$ and keys $\mathbf{k}_i$.

Interpretation of Case ②: α : Non negative
: Sums up to 1

Large weight α_s as a way for the model to "select" v_i's of "RELEVANCE" to query 'q'.

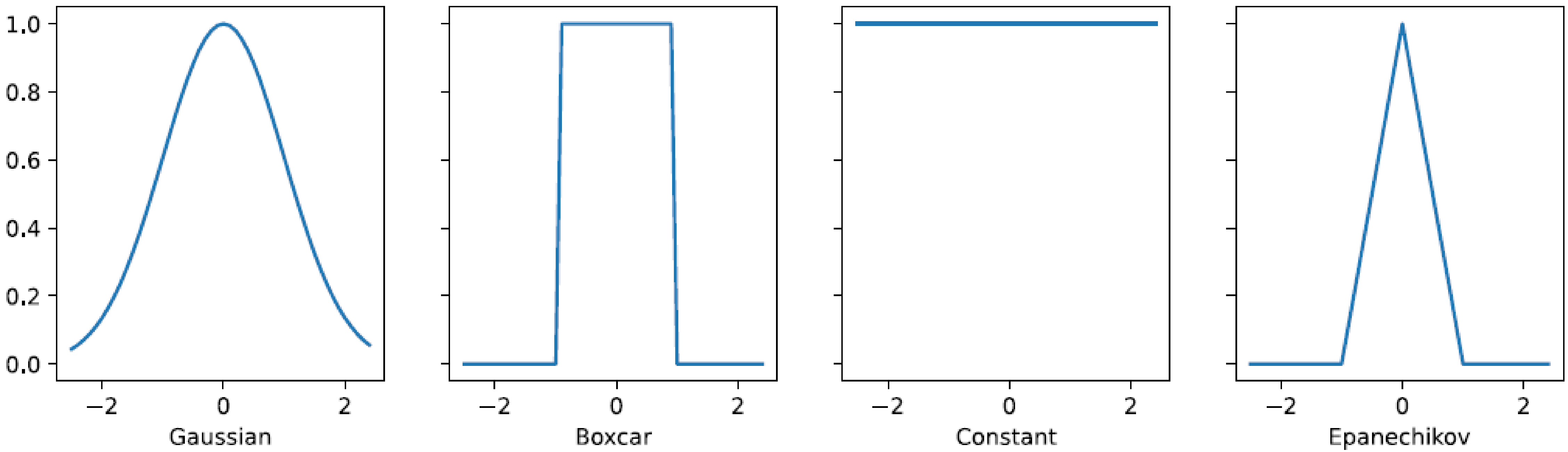
Some Common kernels

$$\alpha(q,k) = \exp\left(-\frac{1}{2}\|q-k\|^2\right) \quad \text{Gaussian}$$

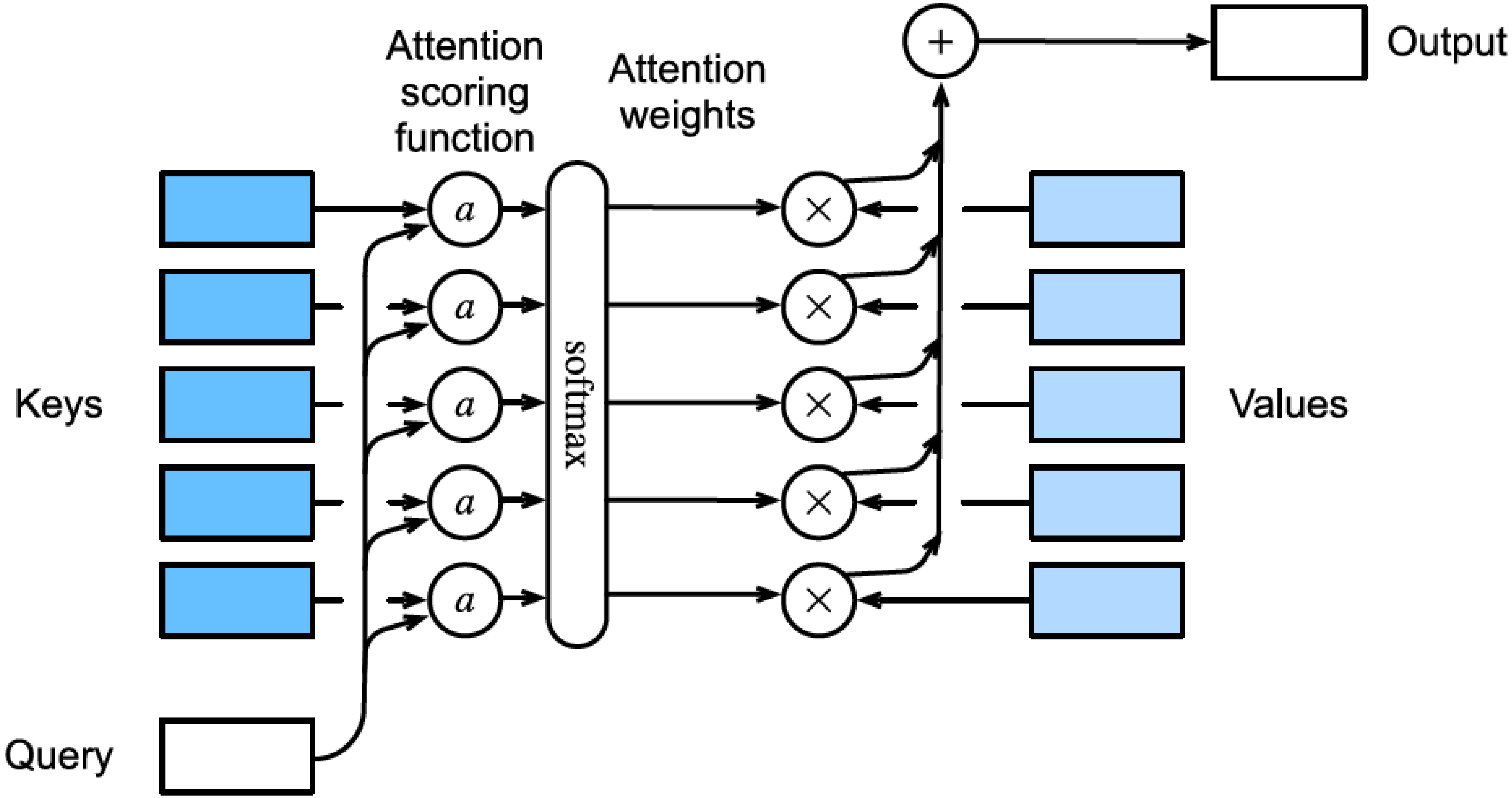
$$\alpha(q,k) = 1 \text{ if } \|q-k\| \leq 1 \quad \text{Box Car}$$

$$\alpha(q,k) = \max(0, 1-\|q-k\|) \quad \text{Epanechikov.}$$

Choice of kernels: Related to KDE / Parzen Windows


1.0
0.8
0.6
0.4
0.2
0.0
−2
0
2
Gaussian
Boxcar
Constant
Epanechikov

ATTENTION SCORING FUNCTION

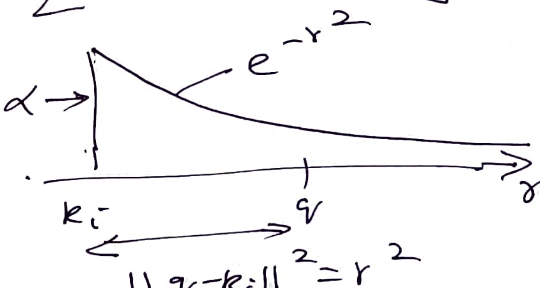


Computing the output of attention pooling as a weighted average of values, where weights are computed with the attention scoring function a and the softmax operation.

DOT-PRODUCT ATTENTION

Gaussian Kernel $\alpha(q, k_i) = \exp\left[-\frac{1}{2}\|q - k_i\|^2\right]$

$r = \|q - k_i\| \Rightarrow \alpha = e^{-r^2} \Rightarrow$



w/o exponentiation

$$\alpha(q, k_i) = -\frac{1}{2}\|q - k_i\|^2$$

$$= q^T k_i - \frac{1}{2}\|k_i\|^2 - \frac{1}{2}\|q\|^2$$

from $\|q - k_i\|^2 = \|q\|^2 + \|k_i\|^2 - 2q^T k_i$

9

$$\alpha(q, k_i) = q^T k_i - \frac{1}{2}\|k_i\|^2 - \frac{1}{2}\|q\|^2$$

Only term that matters

⇒ Batch ~~&~~ Layer Normalization.
⇒ Lead to activations that have "Constant" norms $\|k_i\|$ in DL Context
∴ Can be dropped

- Depends only on "q"
- Can be dropped (identical for all (q, k_i))
- Show that "normalization" as in "softmax" Cancels this term in Numerator & Denominator.

$$\therefore \boxed{\alpha(q, k_i) = q^T k_i}$$

$q \in \mathbb{R}^d$

$k_i \in \mathbb{R}^d$.

$$\alpha(q, k_i) = q^T k_i \qquad q \in \mathbb{R}^d, \; k_i \in \mathbb{R}^d.$$

To ensure Variance of the dot-product remains 1 regardless of vector lengths (dim d)

$\Rightarrow$ Use "Scaled Dot Product Attention" Scoring fn.

ie $$\alpha(q, k_i) = \frac{q^T k_i}{\sqrt{d}}.$$

First Commonly used Attn. Fn. in TRANSFORMERS (Vaswani, 2017)

$$\alpha(q, k_i) = \frac{q^T k_i}{\sqrt{d}}$$

& Normalize attention weights by softmax operation.

$$\alpha(q, k_i) = \text{Softmax}\left[\alpha(q, k_i)\right]$$

$$= \frac{\exp\left(q^T k_i / \sqrt{d}\right)}{\sum_{j=1}^{m} \exp\left(q^T k_j / \sqrt{d}\right)}$$

If $q \in \mathbb{R}^d$ & $k_i \in \mathbb{R}^{d'}$ ie q & k_i are not in same dim.

then, use $\underline{q^T M k}$

- Matrix of suitable dim (e.g $d \times d'$)
- Linear Projection by a Fully Connected (FC) layer.

n queries
m key-value pairs } $q \& k_i$: dim d $\in \mathbb{R}^d$
values v_i : dim v $\in \mathbb{R}^v$

ie for ~~...~~ batch of n queries
Then the Scaled Dot Prod. Attn of Queries

Queries
$Q \in \mathbb{R}^{n \times d}$

$$Q \rightarrow \begin{array}{c|c} & d \\ 1 & \text{———} \\ 2 & \text{———} \\ \vdots & \\ n & \text{———} \end{array}$$

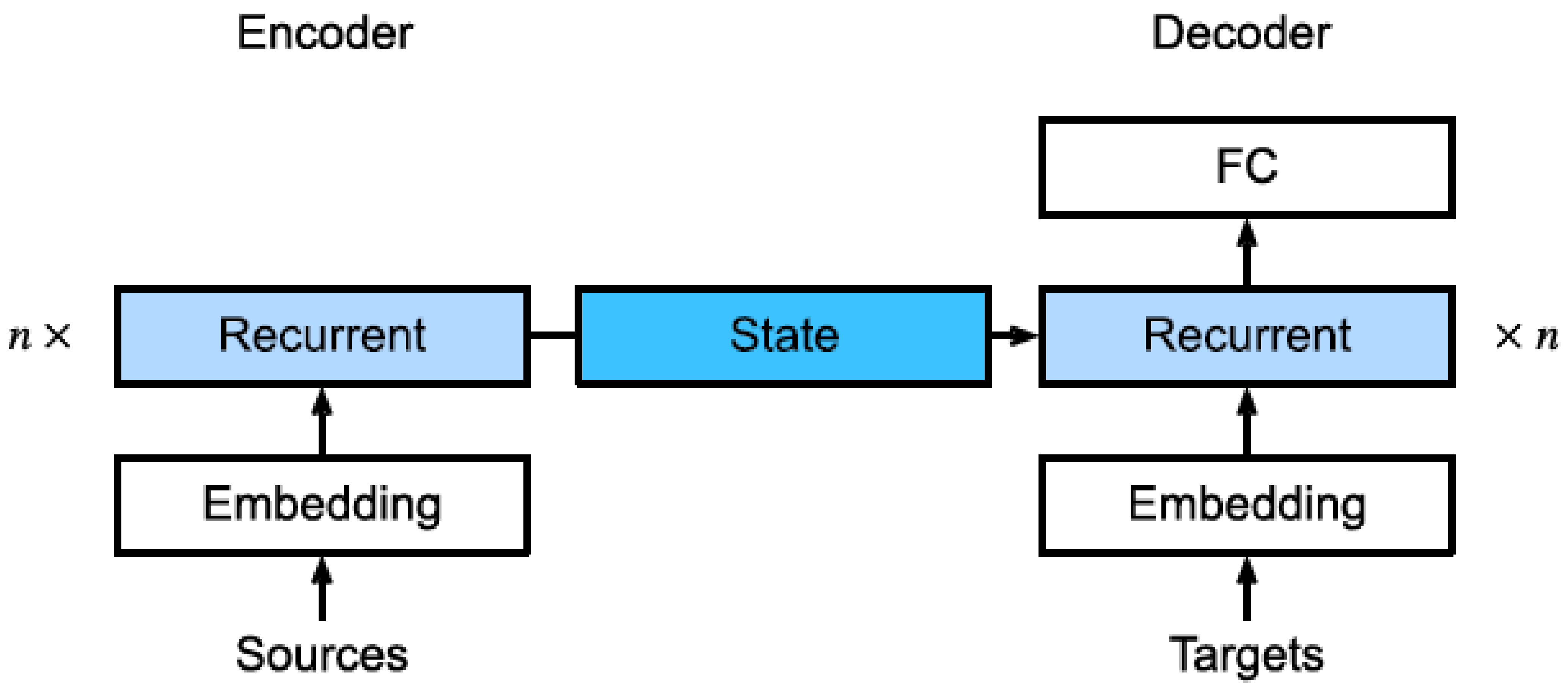
Keys
$K \in \mathbb{R}^{m \times d}$

$$\begin{array}{c|c} & d \\ 1 & \text{———} \\ 2 & \text{———} \\ \vdots & \\ m & \text{———} \end{array}$$

Values
$V \in \mathbb{R}^{m \times v}$

$$\begin{array}{c|c} & v \\ 1 & \text{———} \\ 2 & \text{———} \\ \vdots & \\ m & \text{———} \end{array}$$

$$\text{Scaled DPA} \Rightarrow \alpha(Q, K) \Rightarrow \text{Softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V \in \mathbb{R}^{n \times v}$$

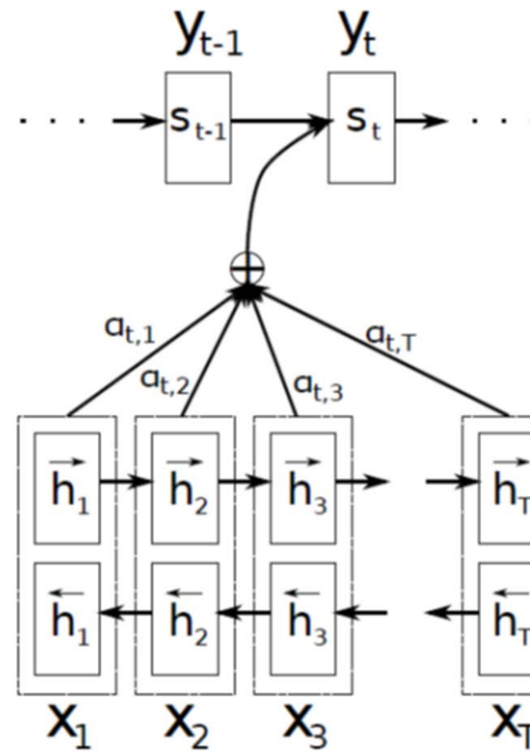


Sequence-to-sequence model. The state, as generated by the encoder, is the only piece of information shared between the encoder and the decoder.

Bahdanau Attention (2014)

- "Learning to Align"
  - Predict a token in o/p @ decoder
  - If not all the i/p tokens are relevant

→ The model aligns ["attends"] only to parts of the i/p sequence that are considered "relevant" to the current prediction.



Bahdanau A.M

$\Rightarrow$ Most influential idea in past decade in DL.

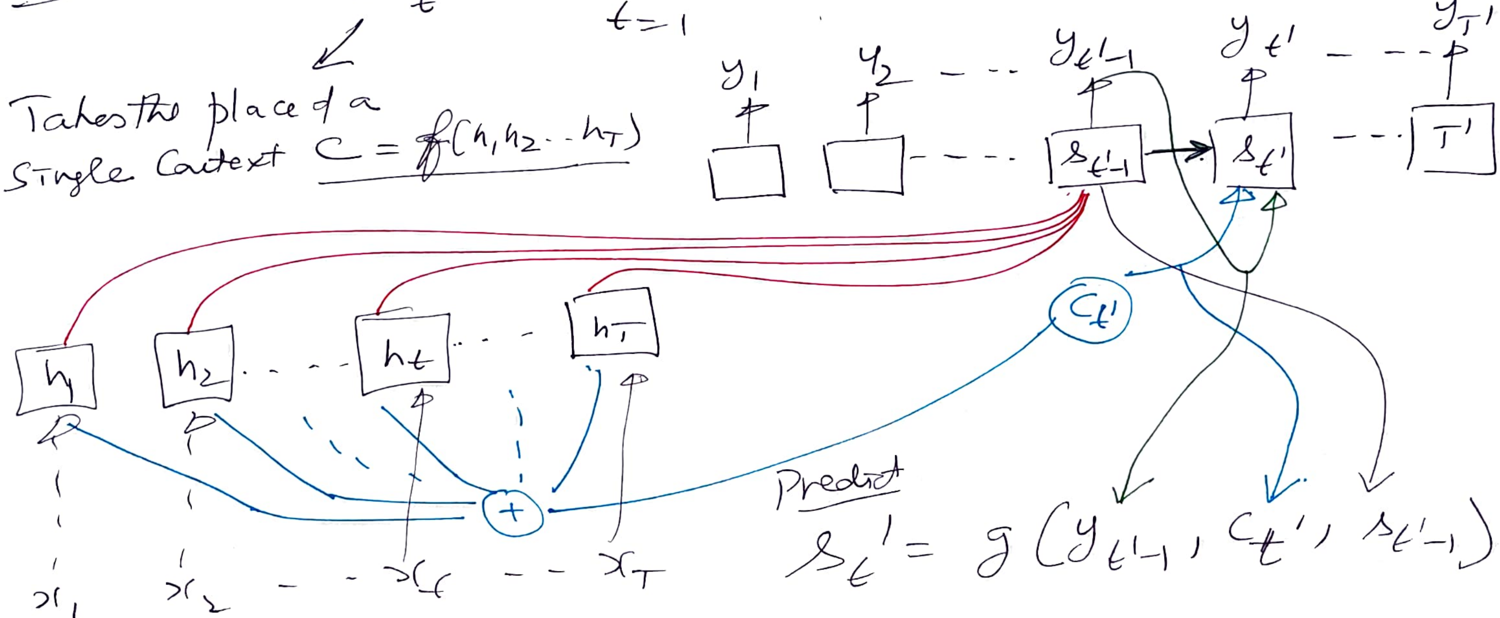
$\Rightarrow$ Giving rise to TRANSFORMERS [Vaswani, 2017]

BAM Model

QUERY KEY VALUE

$$c_{t'} = \sum_{t=1}^{T} \alpha(s_{t'-1}, h_t)\, h_t$$

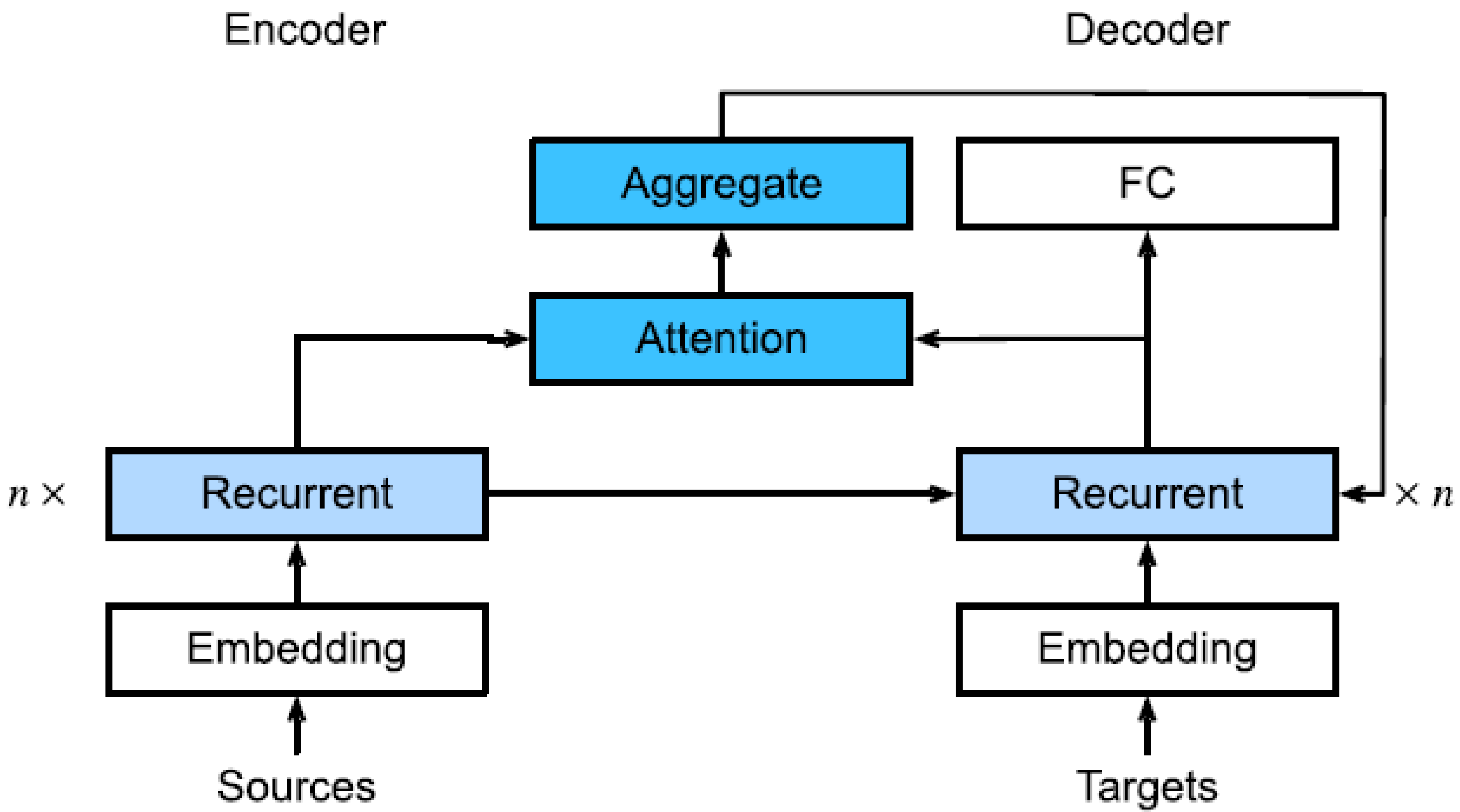
Takes the place of a single Context $C = f(h_1, h_2 \ldots h_T)$



Predict

$$s_{t'} = g(y_{t'-1}, c_{t'}, s_{t'-1})$$

Bahdanau Attention (2015)

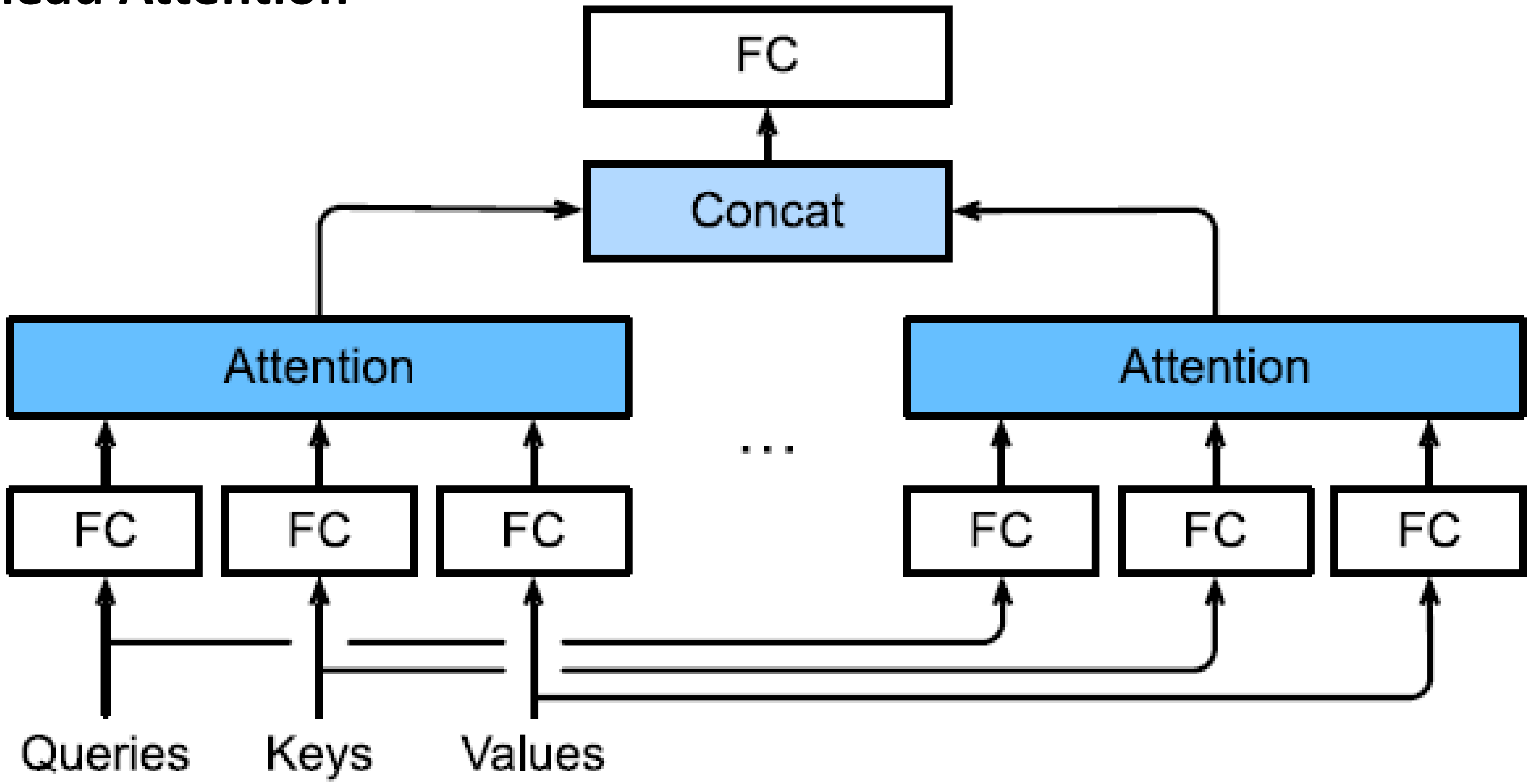


Layers in an RNN encoder–decoder model with the Bahdanau attention mechanism.

MULTI-HEAD ATTENTION

- Capture short range & long-range dependencies within a sequence
- Given a same set Q, K, V
  Q = $s_{t'-1}$, K = h_t, V = h_t
- $Q, K, V \Rightarrow$ Transformed with "h" independently learned linear projection [FC layers]
- These "h" projected Q, K, Vs are fed into Attention Pooling in Parallel.
  - Each of "h" attn pooling o/p = "head"

Multi-head Attention



Multi-head attention, where multiple heads are concatenated then linearly transformed.

SELF-ATTENTION (SA)

- Every token (in an i/p seq)
  attends to every other token.
  [Unlike where Decoder steps attend to
  Encoder steps] as in B.A.M.

Thank you !!