Foundations of Deep Learning

ALF

Alfredo Canziani

@alfcnz

Attention (self/cross, hard/soft)

Dealing with sets

$$\boldsymbol{h} = \boldsymbol{X}\boldsymbol{a}$$

Self-attention (I)

$$\{\boldsymbol{x}_i\}_{i=1}^{t} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots \boldsymbol{x}_t\} \rightsquigarrow \boldsymbol{X} \in \mathbb{R}^{n\times t}, \quad \boldsymbol{x}_i \in \mathbb{R}^n$$

$$\boldsymbol{h} = \alpha_1 \boldsymbol{x}_1 + \alpha_2 \boldsymbol{x}_2 + \cdots + \alpha_t \boldsymbol{x}_t = \boldsymbol{X}\boldsymbol{a} \in \mathbb{R}^n$$

$$\alpha_i > 0$$

$$\boldsymbol{X} \doteq \begin{bmatrix} | & | & & | \\ \boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_t \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n\times t}$$

soft attention: $\|\boldsymbol{a}\|_1 = 1$

hard attention: $\|\boldsymbol{a}\|_0 = 1$

$$\boldsymbol{h} = \boldsymbol{X}\boldsymbol{a}$$

Self-attention (II)

$$\boldsymbol{a} = [\mathrm{soft}](\arg)\max_{\beta}(\boldsymbol{X}^{\top}\boldsymbol{x}) \in \mathbb{R}^{t}$$

$$\{\boldsymbol{x}_i\}_{i=1}^{t} \rightsquigarrow \{\boldsymbol{a}_i\}_{i=1}^{t} \rightsquigarrow \boldsymbol{A} \in \mathbb{R}^{t \times t}$$

$$\{\boldsymbol{a}_i\}_{i=1}^{t} \rightsquigarrow \{\boldsymbol{h}_i\}_{i=1}^{t} \rightsquigarrow \boldsymbol{H} \in \mathbb{R}^{n \times t}$$

$$\boldsymbol{H} = \boldsymbol{X}\boldsymbol{A} \in \mathbb{R}^{n \times t}$$

$[\,\cdot\,]$: optional

Key-value store

- Paradigm for
  - storing (saving)
  - retrieving (querying)
  - managing

  an associative array (dictionary / hash table)

Queries, keys, and values

$$\boldsymbol{q} = \boldsymbol{W}_{\boldsymbol{q}} \boldsymbol{x}, \quad \boldsymbol{k} = \boldsymbol{W}_{\boldsymbol{k}} \boldsymbol{x}, \quad \boldsymbol{v} = \boldsymbol{W}_{\boldsymbol{v}} \boldsymbol{x} \qquad \beta = \frac{1}{\sqrt{d}}$$

$$\boldsymbol{q}, \boldsymbol{k} \in \mathbb{R}^{d'}, \quad \boldsymbol{v} \in \mathbb{R}^{d''}, \quad d' = d'' \overset{\downarrow}{=} d$$

$$\{\boldsymbol{x}_i\}_{i=1}^t \rightsquigarrow \{\boldsymbol{q}_i\}_{i=1}^t, \{\boldsymbol{k}_i\}_{i=1}^t, \{\boldsymbol{v}_i\}_{i=1}^t \rightsquigarrow \boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{d \times t}$$

$$\boldsymbol{a} = [\text{soft}](\arg)\max_\beta(\boldsymbol{K}^\top \boldsymbol{q}) \in \mathbb{R}^t \qquad \boldsymbol{h} = \boldsymbol{V}\boldsymbol{a} \in \mathbb{R}^d$$

$$\{\boldsymbol{q}_i\}_{i=1}^t \rightsquigarrow \{\boldsymbol{a}_i\}_{i=1}^t \rightsquigarrow \boldsymbol{A} \in \mathbb{R}^{t \times t} \qquad \boldsymbol{H} = \boldsymbol{V}\boldsymbol{A} \in \mathbb{R}^{d \times t}$$

Implementation

$$\begin{bmatrix} q \\ k \\ v \end{bmatrix} = \begin{bmatrix} W_q \\ W_k \\ W_v \end{bmatrix} x \in \mathbb{R}^{3d}$$

from the RNN lecture

$$h[t] = g\left(W_h \begin{bmatrix} x[t] \\ h[t-1] \end{bmatrix} + b_h\right)$$

$$h[0] \doteq \mathbf{0}, W_h \doteq \begin{bmatrix} W_{hx} & W_{hh} \end{bmatrix}$$

considering h heads we get a vector in $\mathbb{R}^{3hd}$

using a $W_h \in \mathbb{R}^{d \times hd}$ to go back to $\mathbb{R}^d$

$$\begin{bmatrix} q^1 \\ q^2 \\ \vdots \\ q^h \end{bmatrix} = \begin{bmatrix} W_q^1 \\ W_q^2 \\ \vdots \\ W_q^h \end{bmatrix} x \quad \begin{bmatrix} k^1 \\ k^2 \\ \vdots \\ k^h \end{bmatrix} = \begin{bmatrix} W_k^1 \\ W_k^2 \\ \vdots \\ W_k^h \end{bmatrix} x \quad \begin{bmatrix} v^1 \\ v^2 \\ \vdots \\ v^h \end{bmatrix} = \begin{bmatrix} W_v^1 \\ W_v^2 \\ \vdots \\ W_v^h \end{bmatrix} x$$

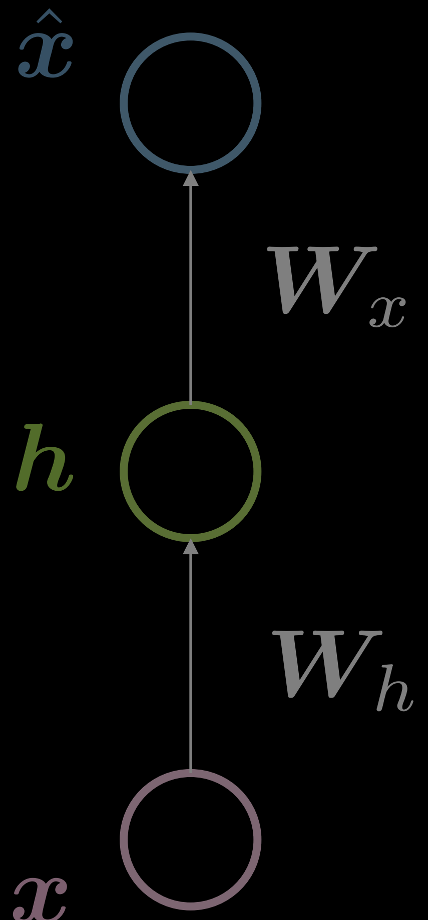
Transformer

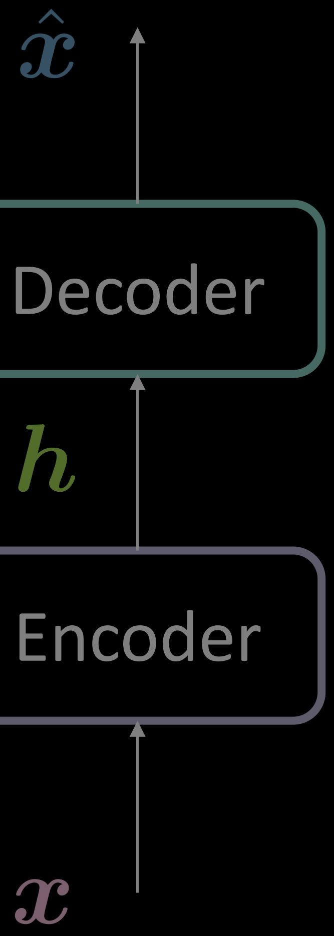
Encoder-decoder architecture

(for Neural Machine Translation)

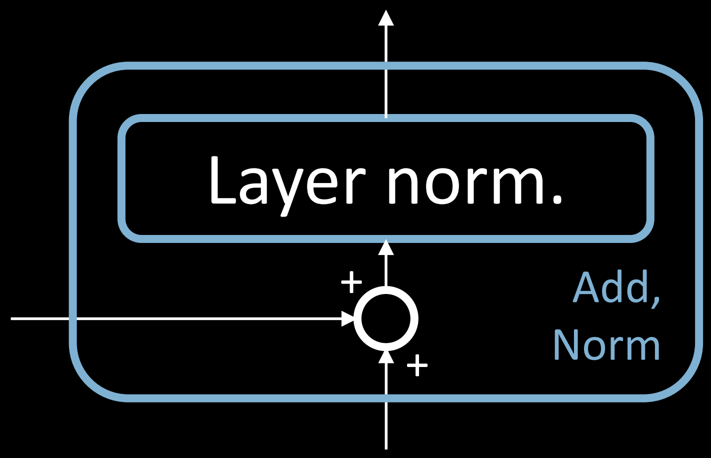
from the AE lecture

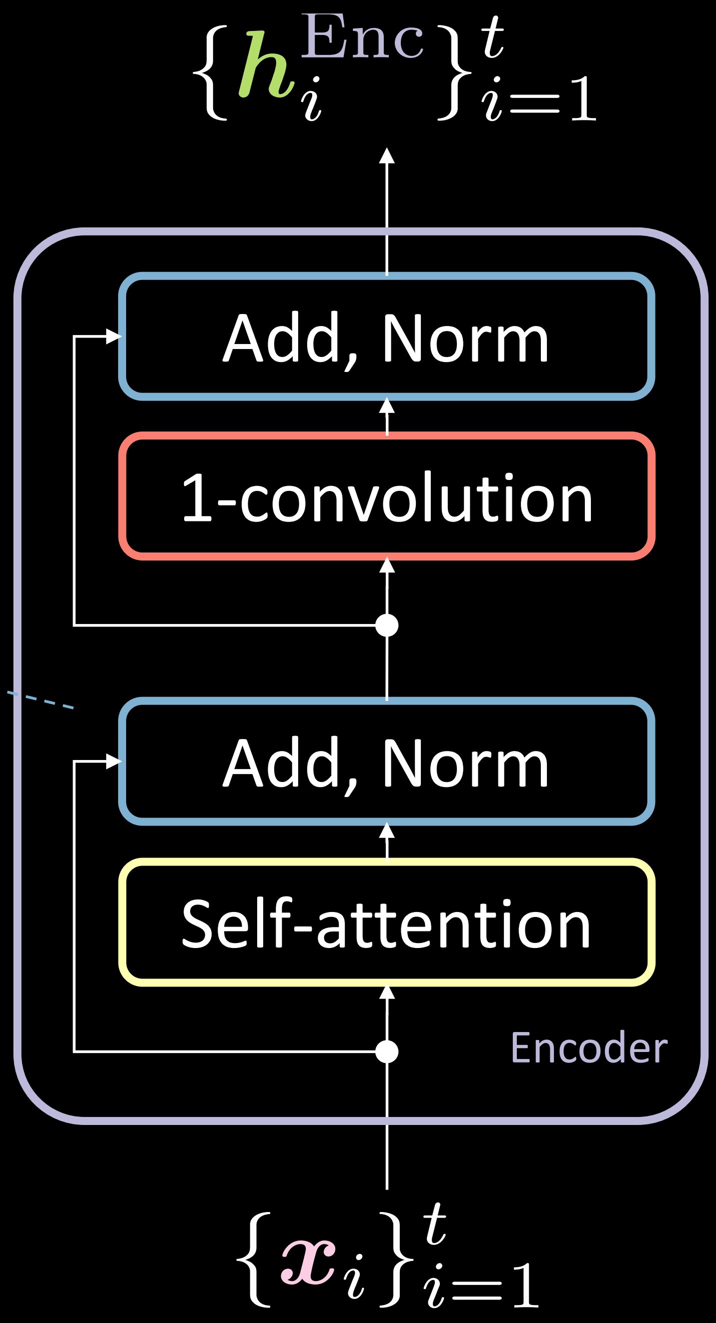
Auto-encoder (recap)





Transformer encoder





Transformer decoder