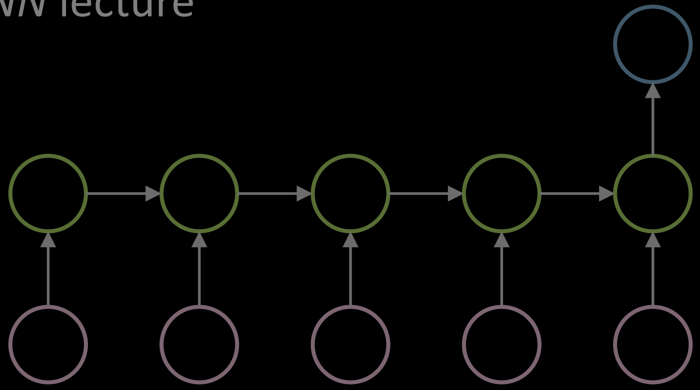
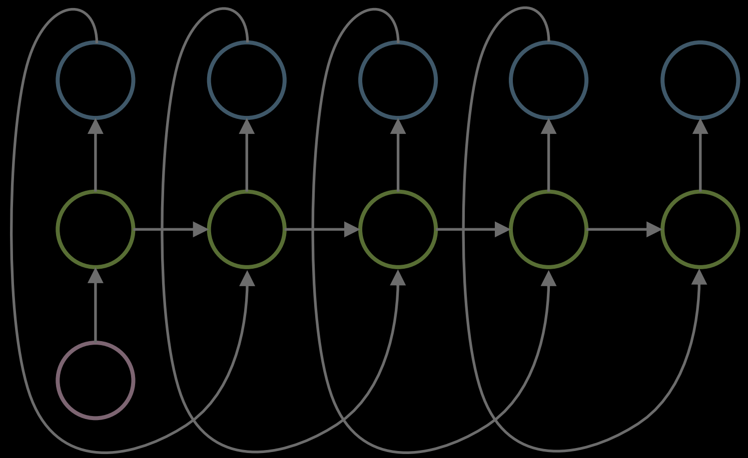
Attention (self/cross, hard/soft)

Dealing with sets

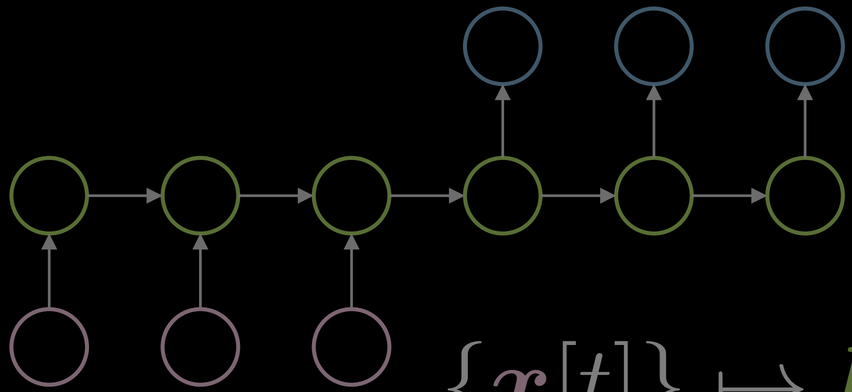
from the *RNN* lecture



$\{\boldsymbol{x}[t]\} \mapsto \hat{\boldsymbol{y}}[T] \quad \text{seq} \mapsto \text{vec}$

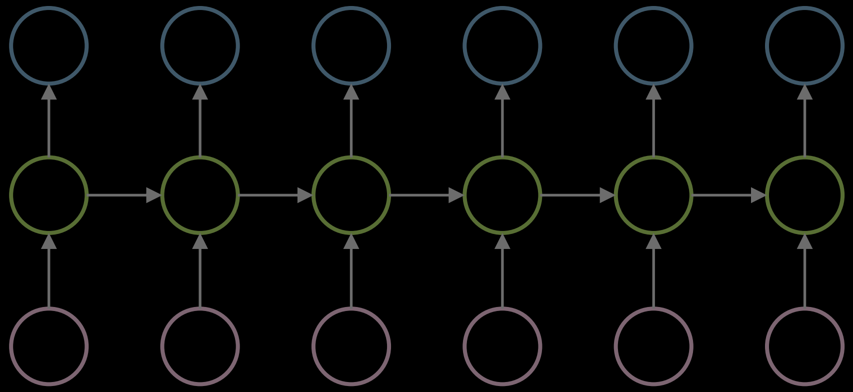


$\boldsymbol{x}[1] \mapsto \{\hat{\boldsymbol{y}}[t]\} \quad \text{vec} \mapsto \text{seq}$



$\{\boldsymbol{x}[t]\} \mapsto \boldsymbol{h} \mapsto \{\hat{\boldsymbol{y}}[t]\}$

$\text{seq} \mapsto \text{vec} \mapsto \text{seq}$



$\{\boldsymbol{x}[t]\} \mapsto \{\hat{\boldsymbol{y}}[t]\} \quad \text{seq} \mapsto \text{seq}$

Use cases

- img $\mapsto$ set: image to bounding box (BB) (DETER)
- set $\mapsto$ set: point clouds to BB, surrounding vehicle trajectory pred.
- seq $\mapsto$ seq: translation, conditional image generation (DALL-e)
- seq $\mapsto$ set: event location and duration
- img $\mapsto$ vec: visual image transformer (VIT)
- seq $\mapsto$ vec: movies review

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

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)

$$\boldsymbol{a} = \text{softargmax}_{\beta}(\boldsymbol{X}^{\top}\boldsymbol{x}) \in \mathbb{R}^{t}$$

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

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

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

Key-value store

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

  an associative array (dictionary / hash table)

Self attention

Queries, keys, and values

$$\{\boldsymbol{q}_i\}_{i=1}^t \rightsquigarrow \boldsymbol{Q} \in \mathbb{R}^{d' \times t}$$

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

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

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

$$\boldsymbol{a} = \operatorname{softargmax}_\beta(\boldsymbol{K}^\top \boldsymbol{q}) \in \mathbb{R}^t \qquad \boldsymbol{h} = \boldsymbol{V}\boldsymbol{a} \in \mathbb{R}^{d''}$$

$$\{\boldsymbol{q}_i\}_{i=1}^t \rightsquigarrow \{\boldsymbol{a}_i\}_{i=1}^t \rightsquigarrow \boldsymbol{A} \in \mathbb{R}^{t \times t} \qquad \boldsymbol{H} = \boldsymbol{V}\boldsymbol{A} \in \mathbb{R}^{d'' \times t}$$

Queries, keys, and values

$$\{\boldsymbol{q}_i\}_{i=1}^{t} \rightsquigarrow \boldsymbol{Q} \in \mathbb{R}^{d' \times t}$$

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

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

$$\{\boldsymbol{\xi}_j\}_{j=1}^{\tau} \rightsquigarrow \{\boldsymbol{k}_j\}_{j=1}^{\tau}, \{\boldsymbol{v}_j\}_{j=1}^{\tau} \rightsquigarrow \boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{\{d', d''\} \times \tau}$$

$$\boldsymbol{a} = \operatorname{softargmax}_{\beta}(\boldsymbol{K}^{\top} \boldsymbol{q}) \in \mathbb{R}^{\tau} \qquad \boldsymbol{h} = \boldsymbol{V} \boldsymbol{a} \in \mathbb{R}^{d''}$$

$$\{\boldsymbol{q}_i\}_{i=1}^{t} \rightsquigarrow \{\boldsymbol{a}_i\}_{i=1}^{t} \rightsquigarrow \boldsymbol{A} \in \mathbb{R}^{\tau \times t} \qquad \boldsymbol{H} = \boldsymbol{V}\boldsymbol{A} \in \mathbb{R}^{d'' \times t}$$

$$d' = d'' \overset{\downarrow}{\doteq} d$$

Implementation

from the *RNN* lecture

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

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

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

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

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

$$\begin{bmatrix}\boldsymbol{q}^1\\ \boldsymbol{q}^2\\ \vdots\\ \boldsymbol{q}^h\end{bmatrix} = \begin{bmatrix}\boldsymbol{W}_{\boldsymbol{q}}^1\\ \boldsymbol{W}_{\boldsymbol{q}}^2\\ \vdots\\ \boldsymbol{W}_{\boldsymbol{q}}^h\end{bmatrix}\boldsymbol{x} \quad \begin{bmatrix}\boldsymbol{k}^1\\ \boldsymbol{k}^2\\ \vdots\\ \boldsymbol{k}^h\end{bmatrix} = \begin{bmatrix}\boldsymbol{W}_{\boldsymbol{k}}^1\\ \boldsymbol{W}_{\boldsymbol{k}}^2\\ \vdots\\ \boldsymbol{W}_{\boldsymbol{k}}^h\end{bmatrix}\boldsymbol{x} \quad \begin{bmatrix}\boldsymbol{v}^1\\ \boldsymbol{v}^2\\ \vdots\\ \boldsymbol{v}^h\end{bmatrix} = \begin{bmatrix}\boldsymbol{W}_{\boldsymbol{v}}^1\\ \boldsymbol{W}_{\boldsymbol{v}}^2\\ \vdots\\ \boldsymbol{W}_{\boldsymbol{v}}^h\end{bmatrix}\boldsymbol{x}$$

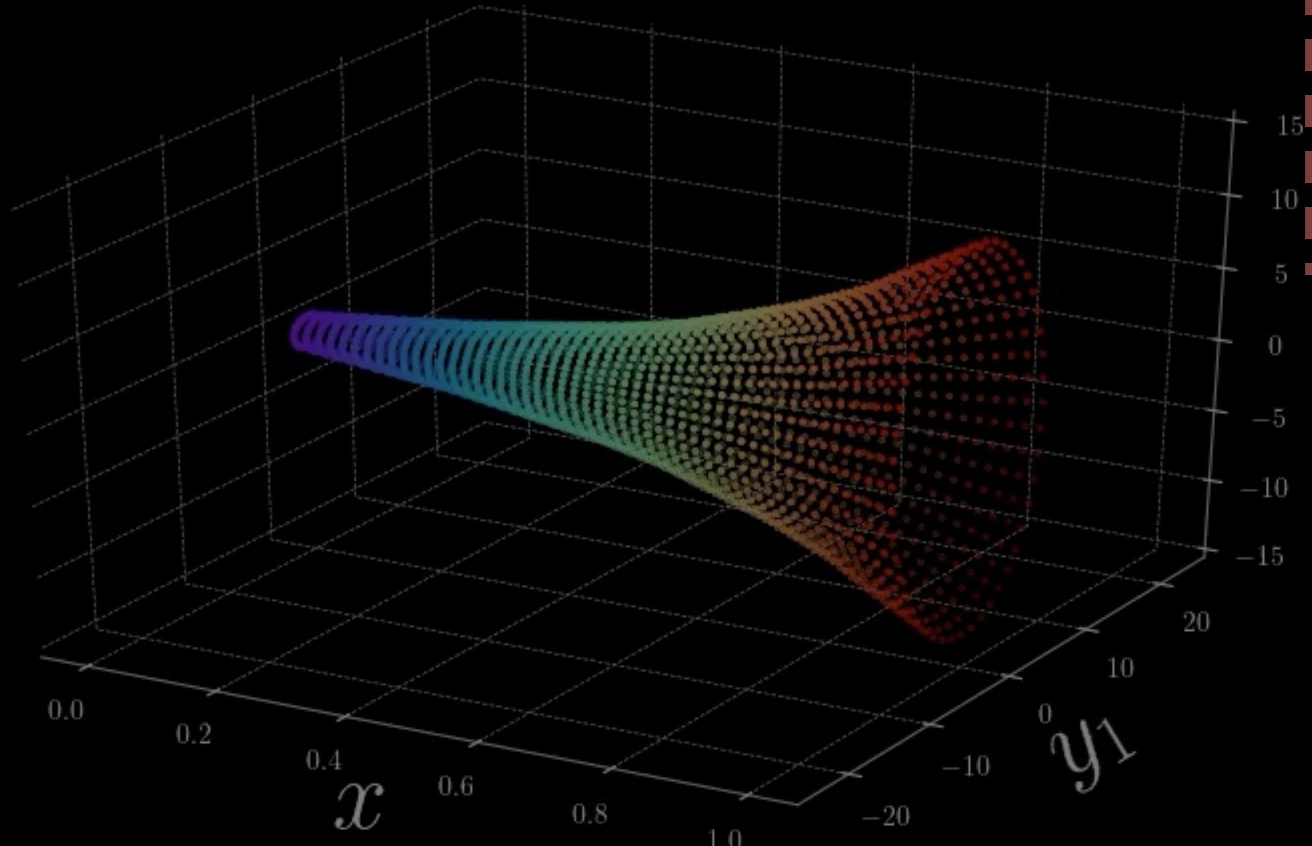
Transformer

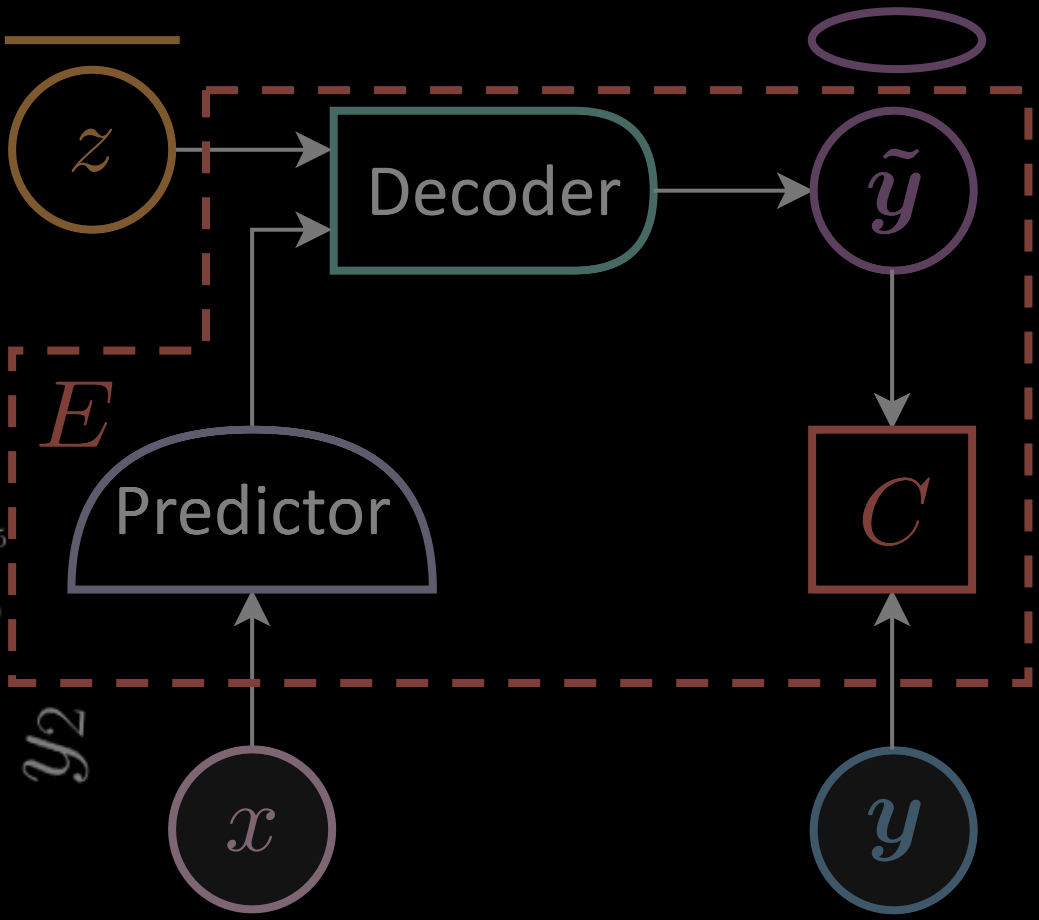
Encoders-predictor-decoder architecture

(for Neural Machine Translation)

from the *EBM* lecture

Trained model manifold





$$z = \left[0 : \frac{\pi}{24} : 2\pi\right[$$

$$x = \left[0 : \frac{1}{50} : 1\right]$$

from the *AE* lecture

Autoencoder

$$\boldsymbol{h} = f(\boldsymbol{W_h y} + \boldsymbol{b_h})$$

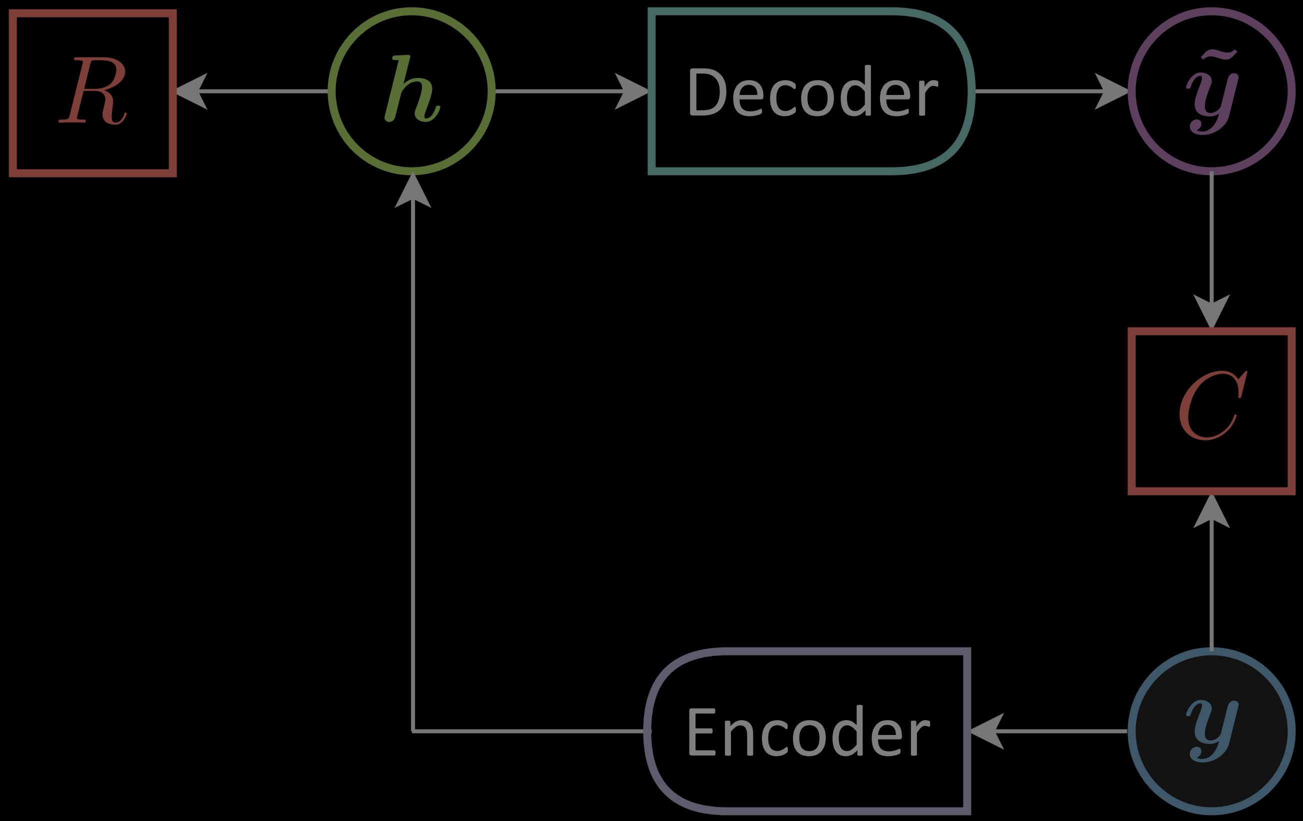
$$\tilde{\boldsymbol{y}} = g(\boldsymbol{W_y h} + \boldsymbol{b_y})$$

$$\boldsymbol{y}, \tilde{\boldsymbol{y}} \in \mathbb{R}^n$$

$$\boldsymbol{h} \in \mathbb{R}^d$$

$$\boldsymbol{W_h} \in \mathbb{R}^{d \times n}$$

$$\boldsymbol{W_y} \in \mathbb{R}^{n \times d}$$



$$\boldsymbol{h} = \mathrm{Enc}(\boldsymbol{y}) \quad \tilde{\boldsymbol{y}} = \mathrm{Dec}(\boldsymbol{h})$$

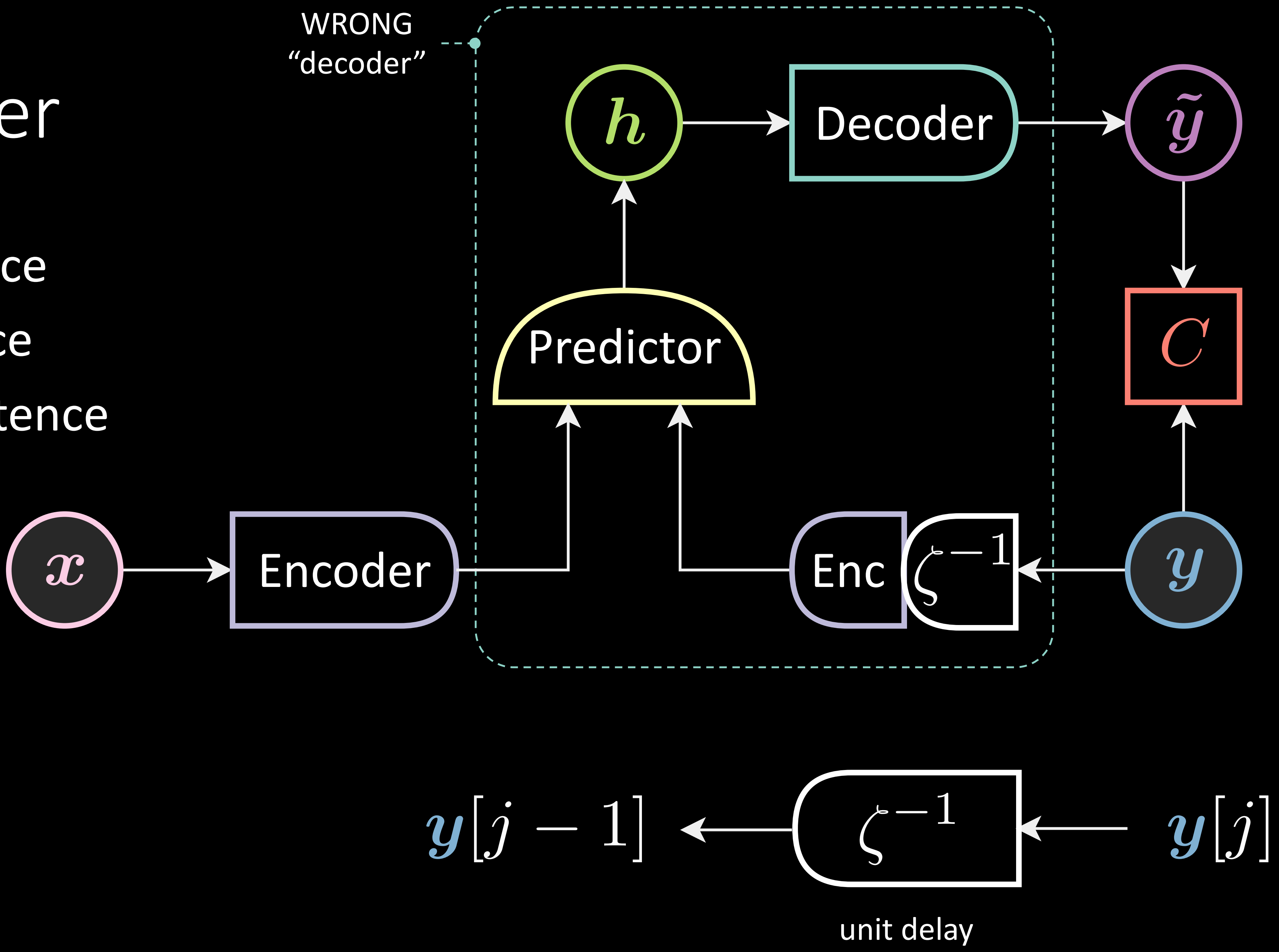
$$F(\boldsymbol{y}) = C(\boldsymbol{y}, \tilde{\boldsymbol{y}}) + R(\boldsymbol{h})$$

Transformer

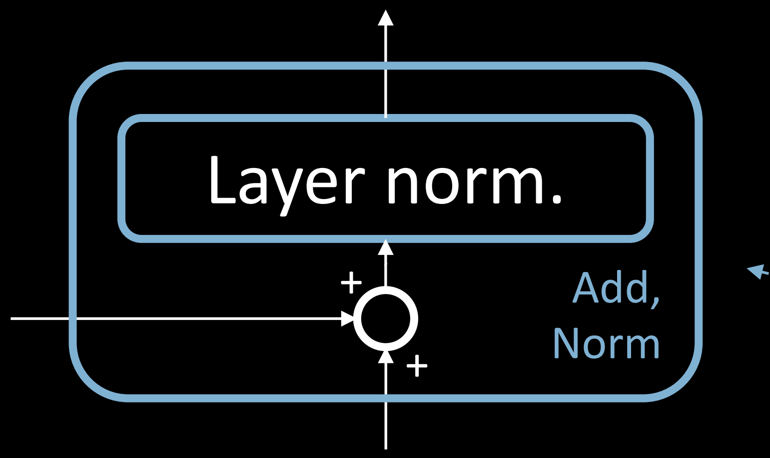
$\boldsymbol{x}$ source sentence

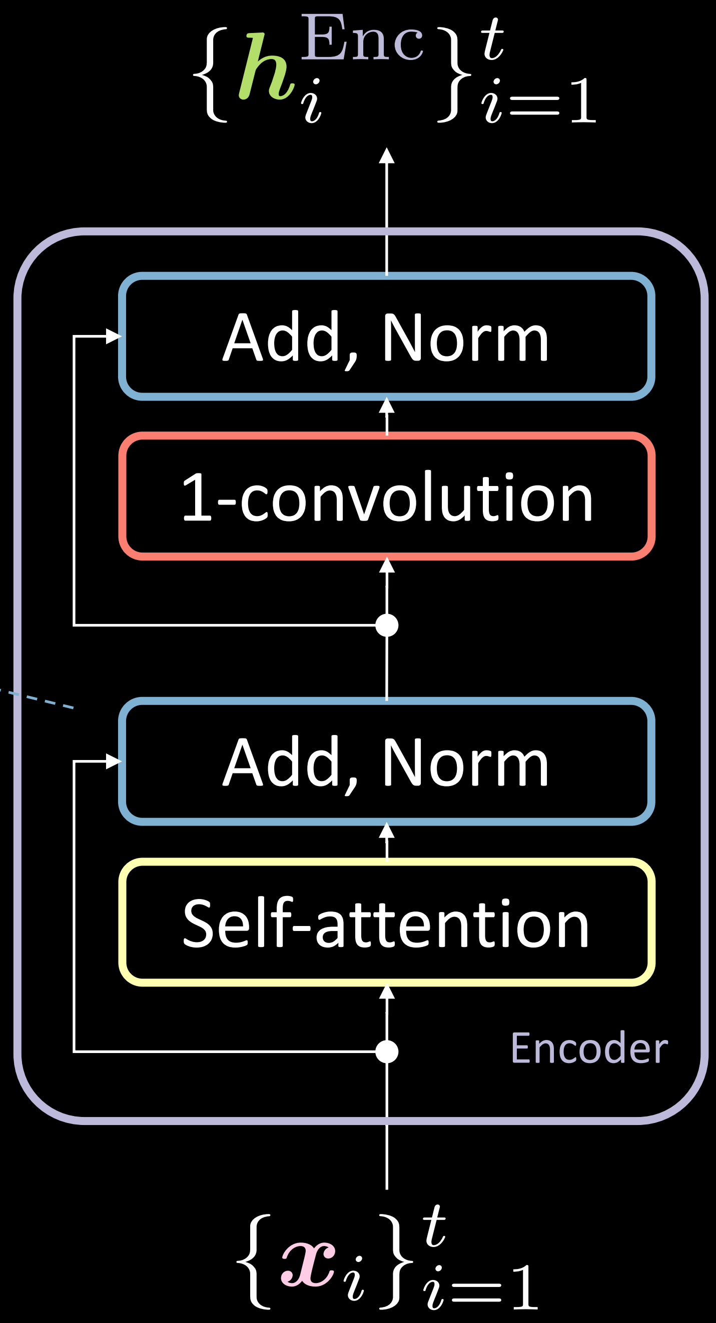
$\boldsymbol{y}$ target sentence

$\tilde{\boldsymbol{y}}$ predicted sentence

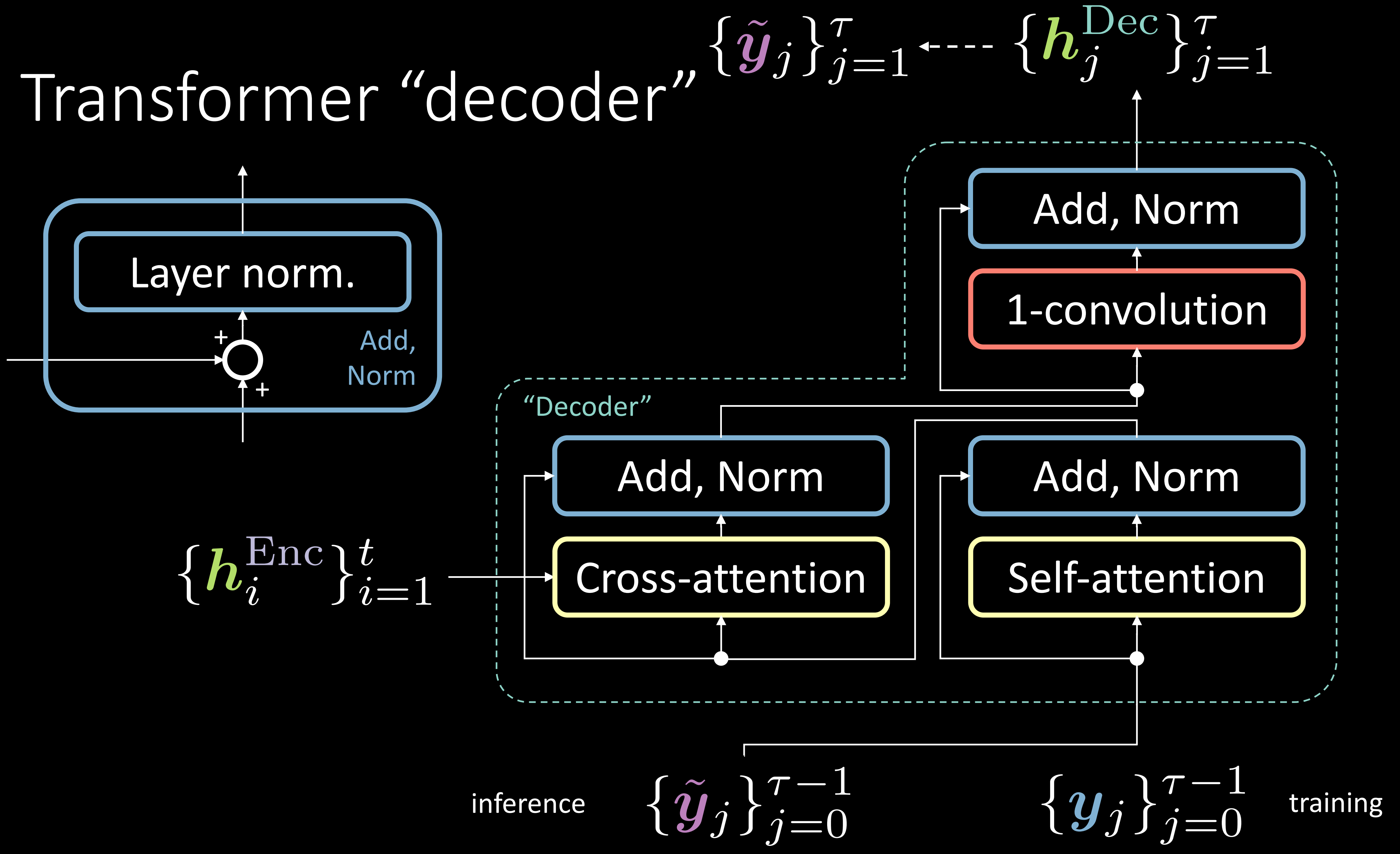


Transformer encoder





Transformer “decoder”



Transformer “decoder”