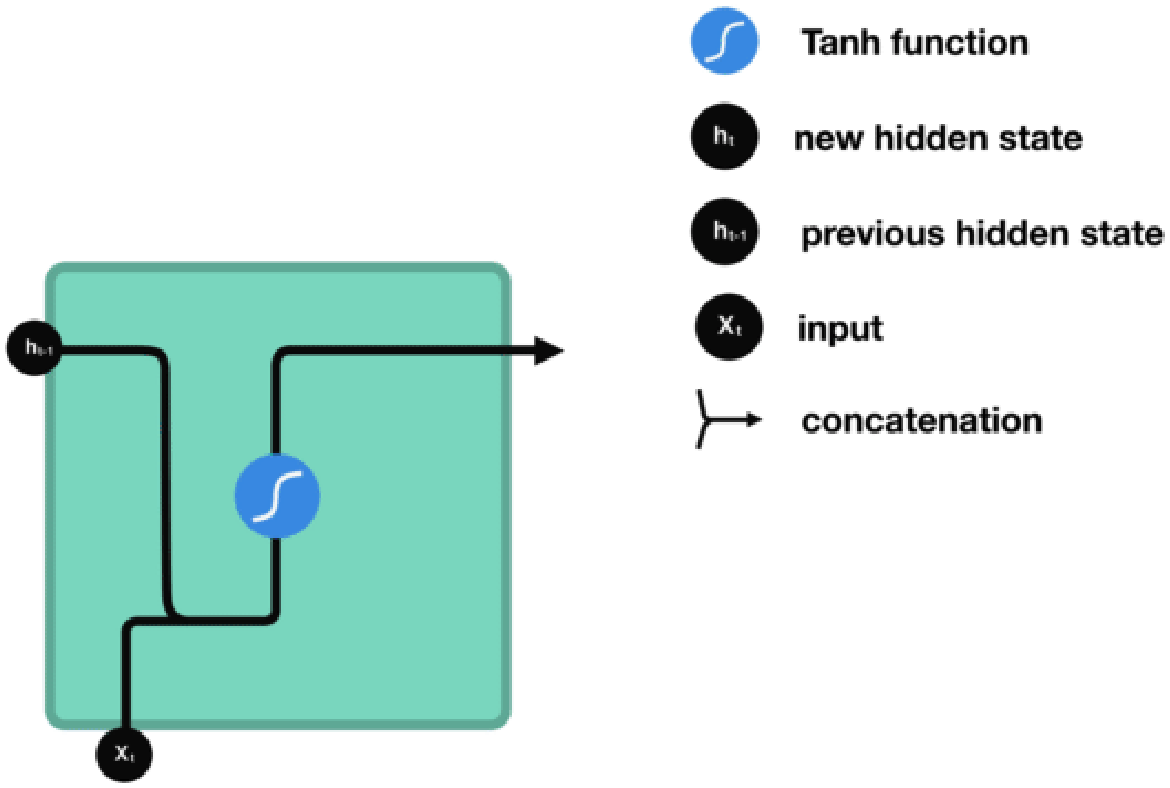
LECTURE 6:

ADVANCED RECURRENT NEURAL NETWORKS

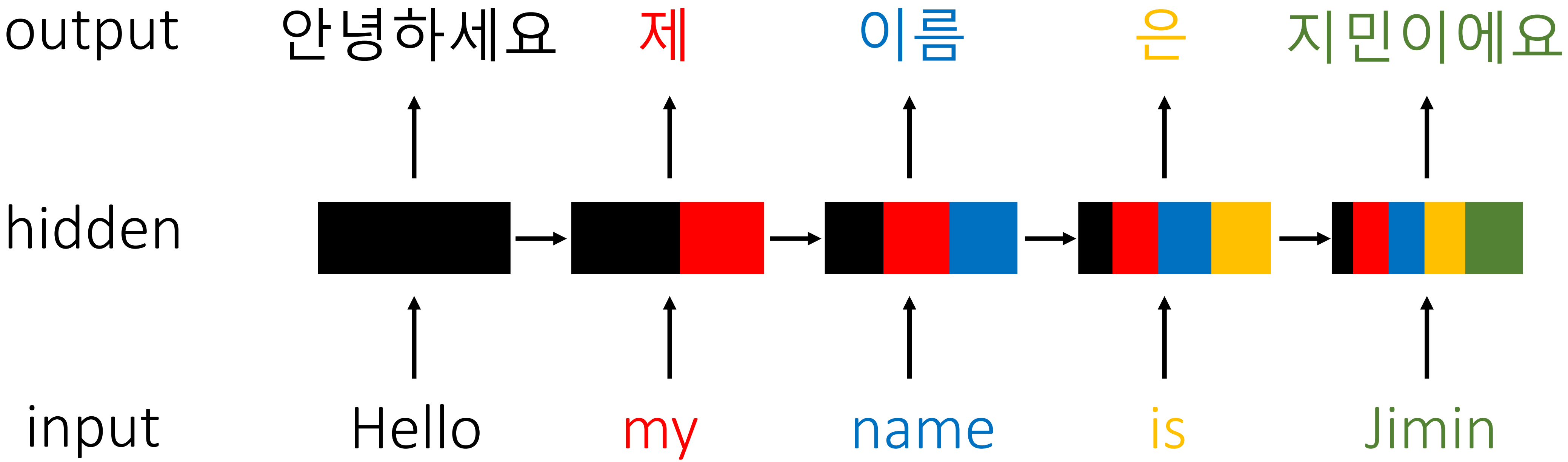
University of Washington, Seattle

Fall 2024

Previously in EEP 596…



Previously in EEP 596...



OUTLINE

Part 1: Gated RNNs

- Need for Gated RNNs
- Long Short-Term Memory (LSTM)
- Gated Recurrent Unit (GRU)

Part 2: Training Gated RNNs

- Mini-batch Gradient in RNNs
- RNN extensions on LSTM/GRU

Part 3: Encoder-Decoder RNNs

- Many to many RNN Recap
- Encoder-Decoder Architecture
- Training Encoder-Decoder RNNs

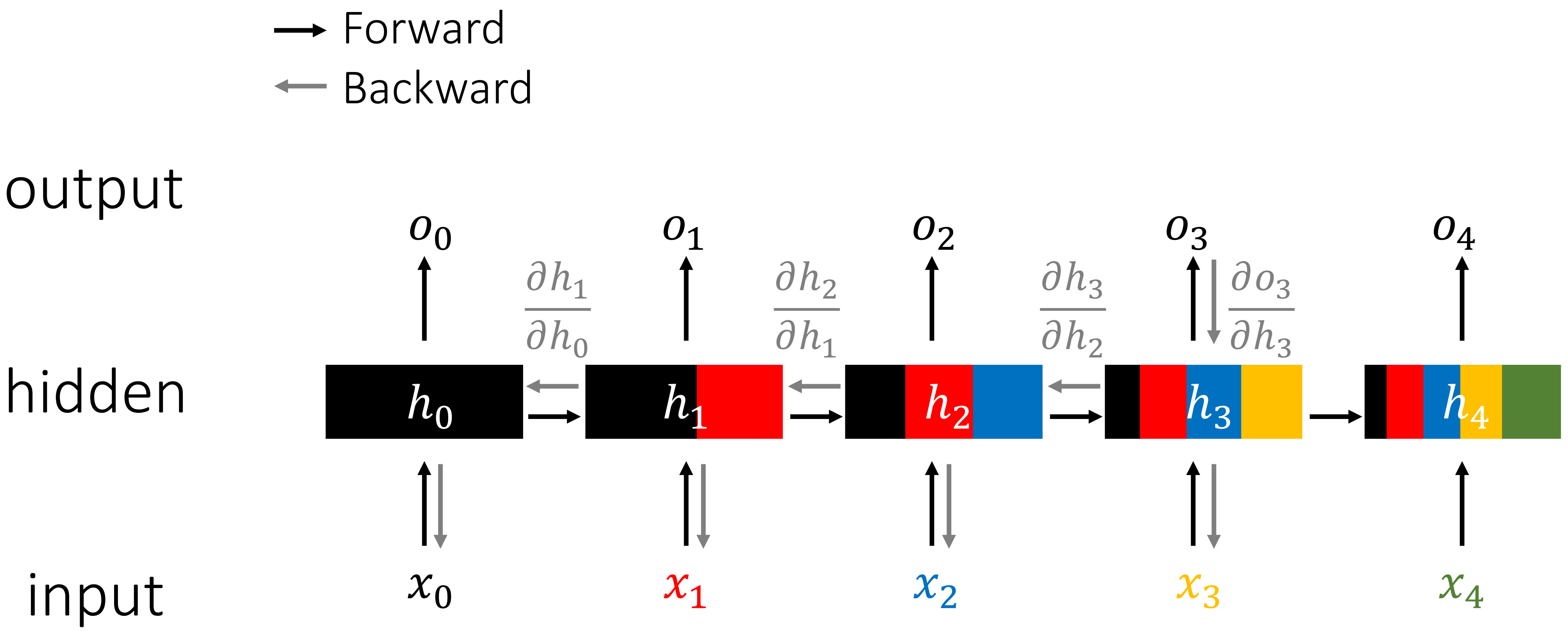
GATED RNNs

Need for Gated RNNs

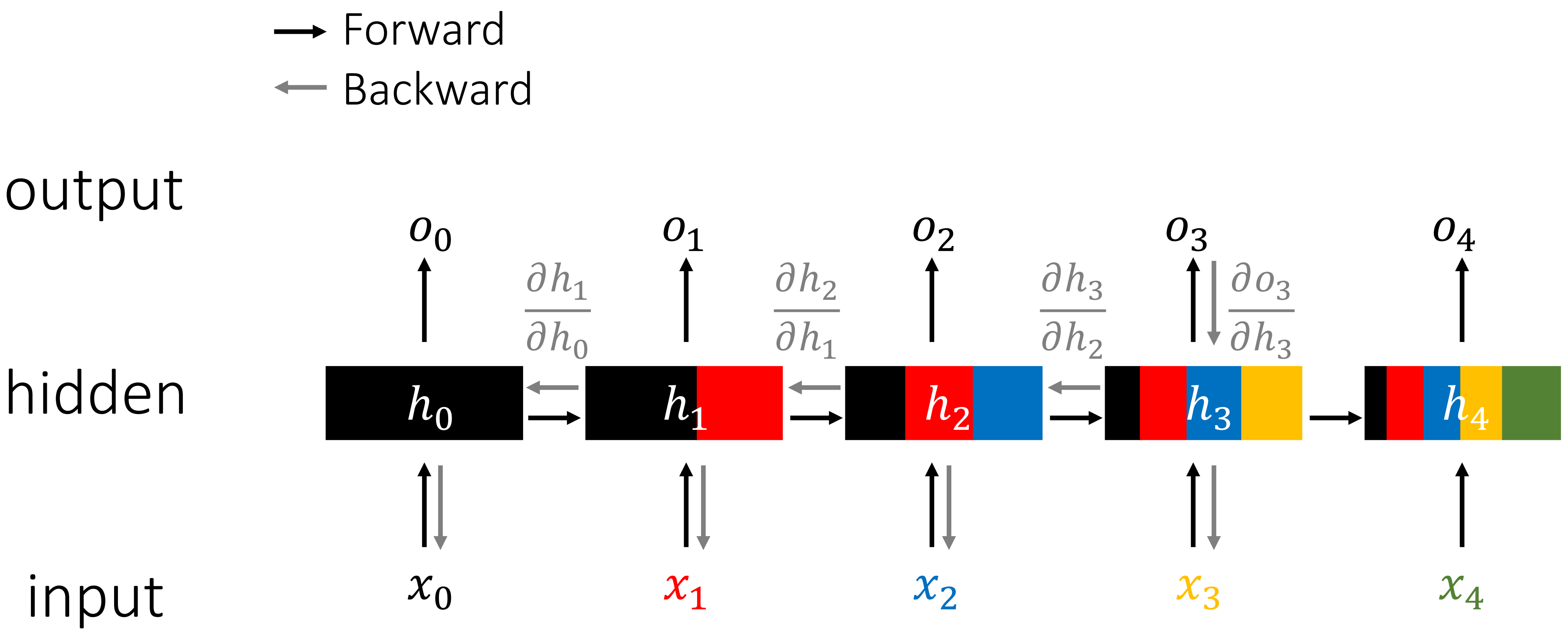
Long Short-Term Memory (LSTM)

Gated Recurrent Unit (GRU)

Recap: Backpropagation in RNNs

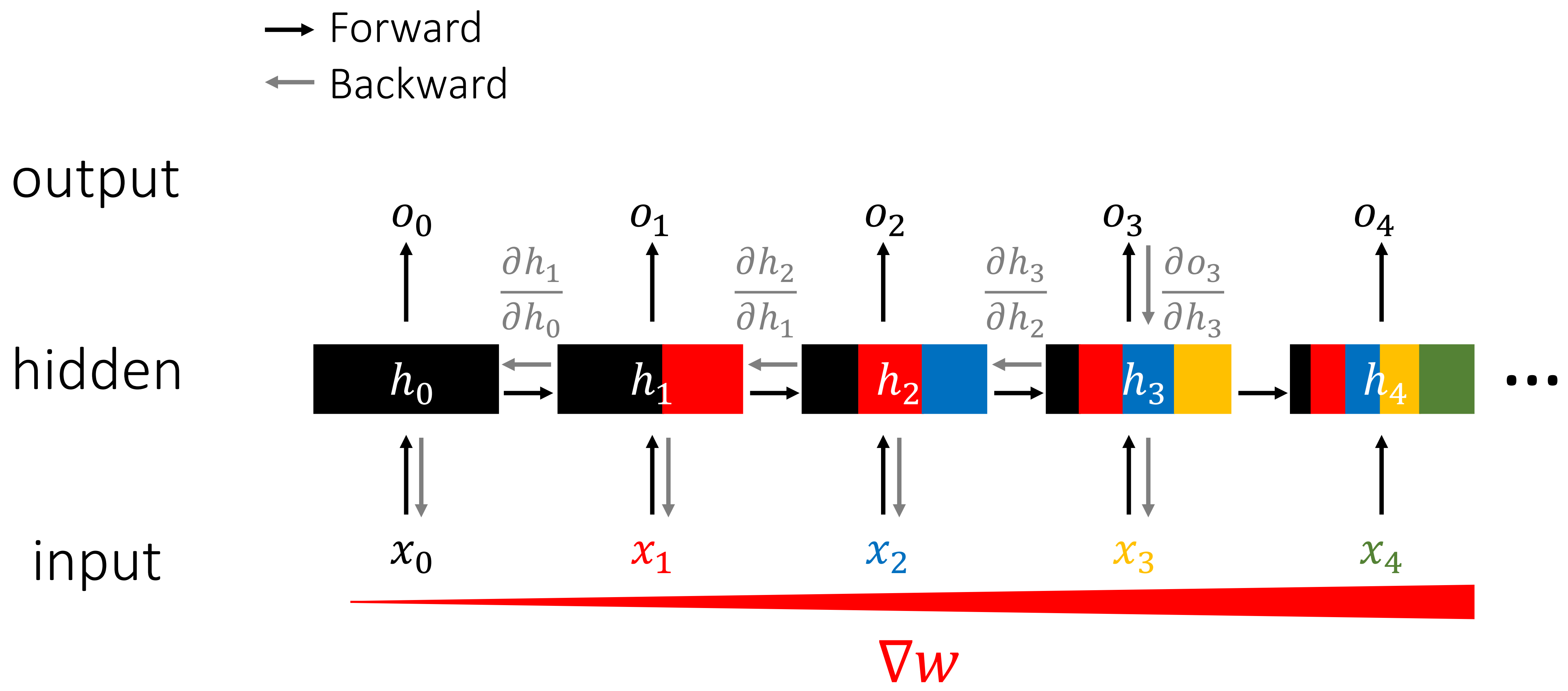


Recap: Backpropagation in RNNs

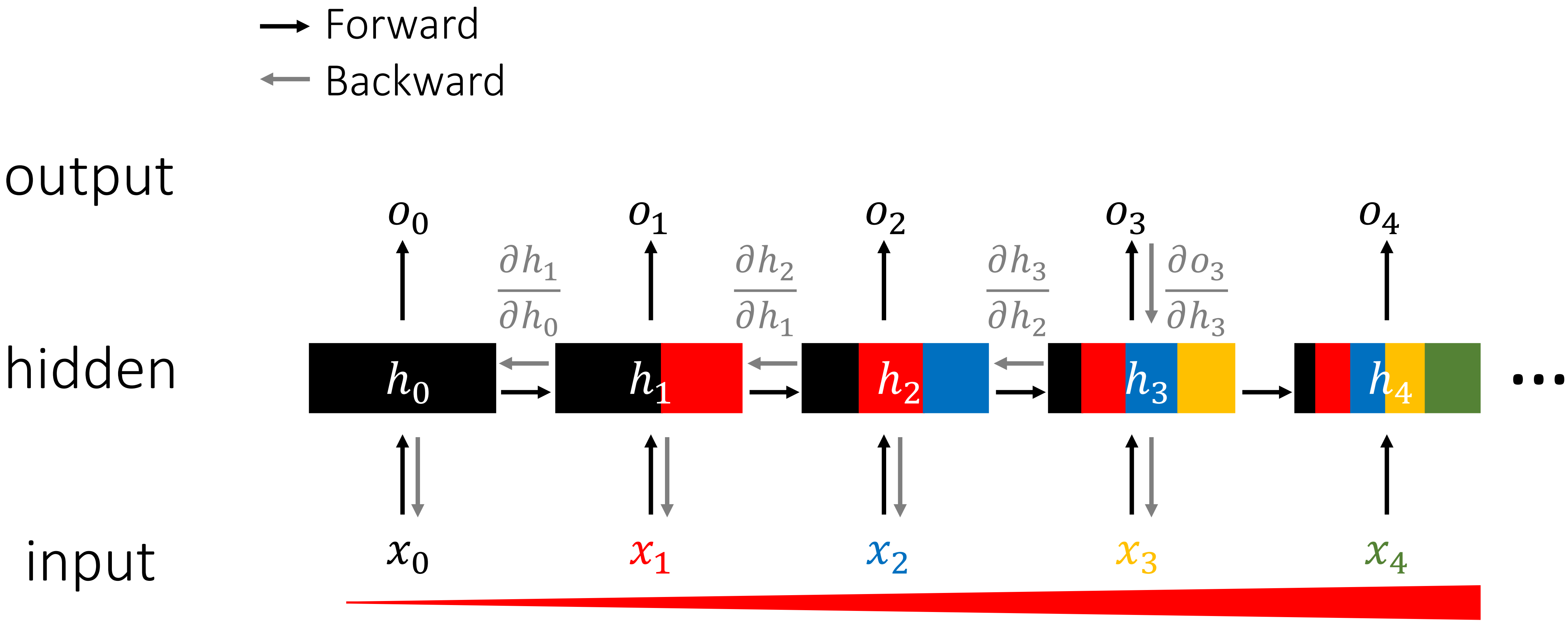


Backpropagation is performed backward in time

Vanishing and Exploding Gradients



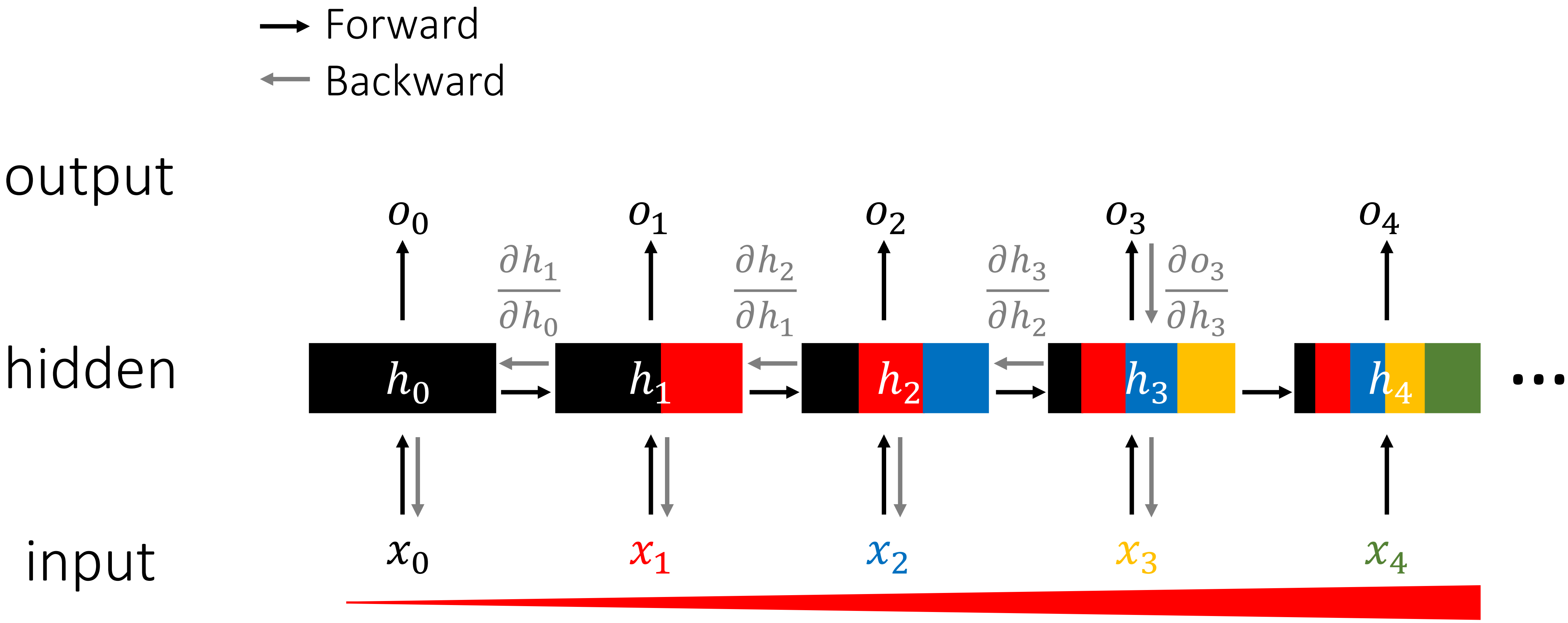
Vanishing and Exploding Gradients



Longer input sequence →

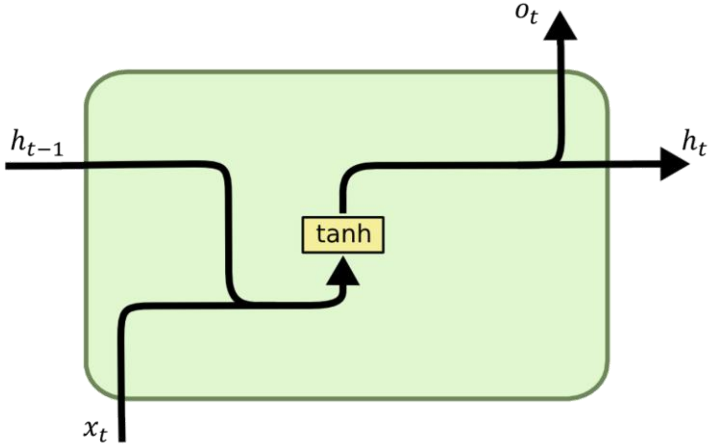
higher risk of Vanishing/Exploding Gradients!

Vanishing and Exploding Gradients

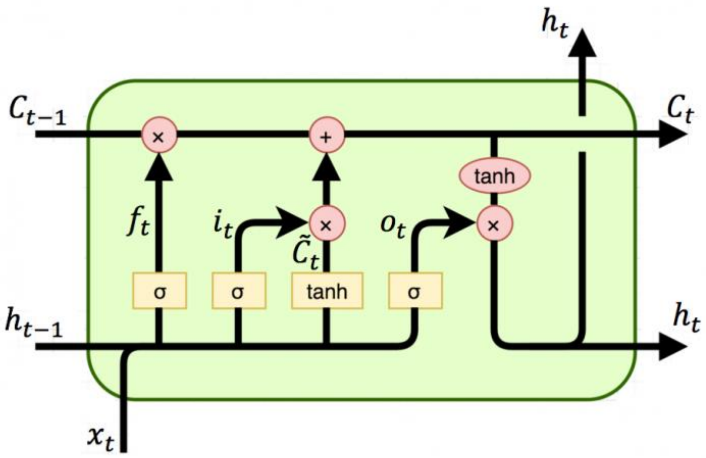


Need for better RNN architecture capable of
processing longer sequence

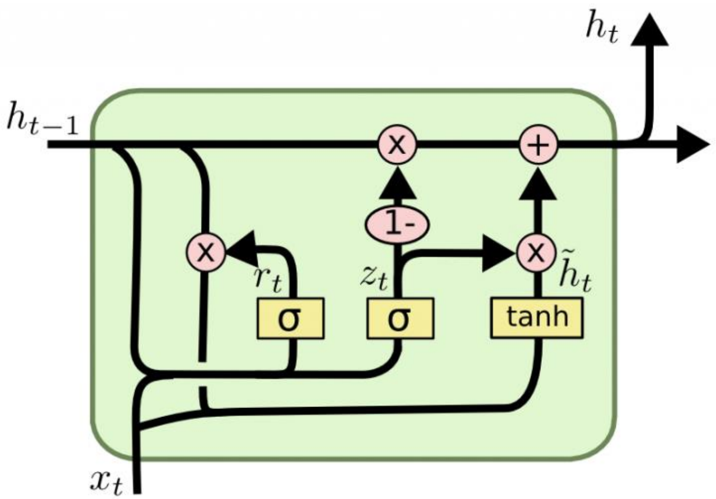
Gated RNNs



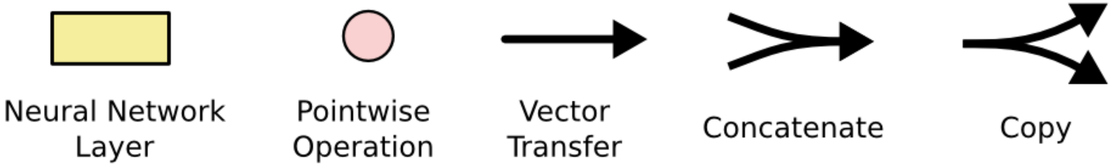
Vanilla RNN



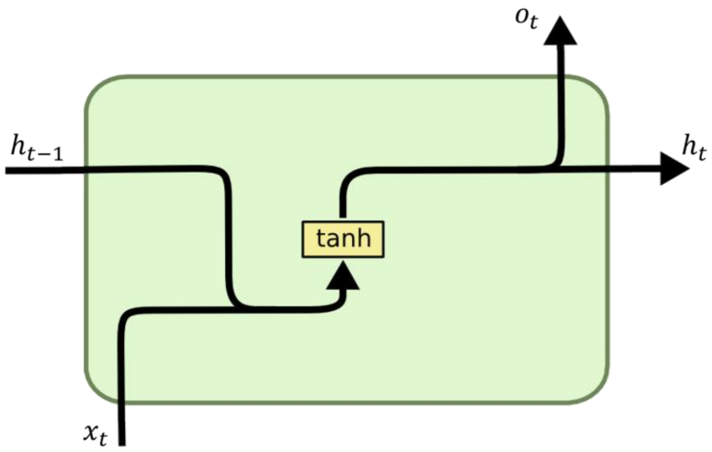
LSTM



GRU

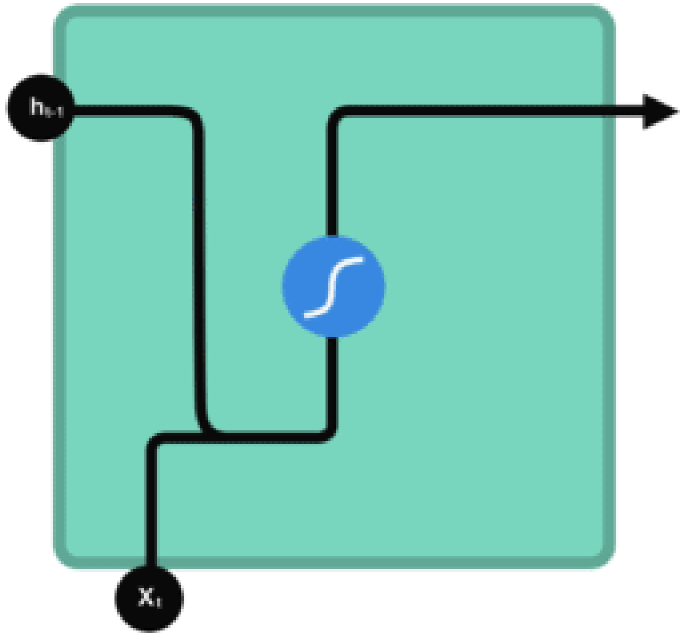


Vanilla RNN



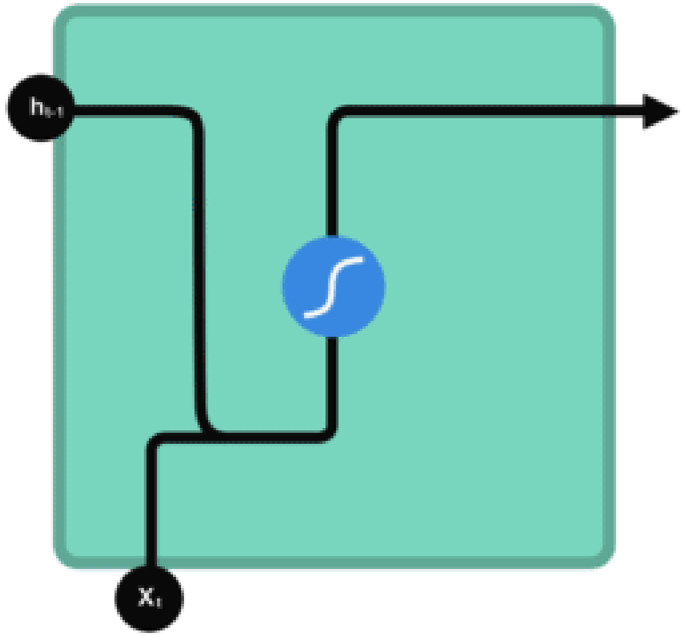
Vanilla RNN

Vanilla RNN



- Tanh function
- h_t new hidden state
- h_{t-1} previous hidden state
- X_t input
- concatenation

Vanilla RNN



Tanh function

h_t new hidden state

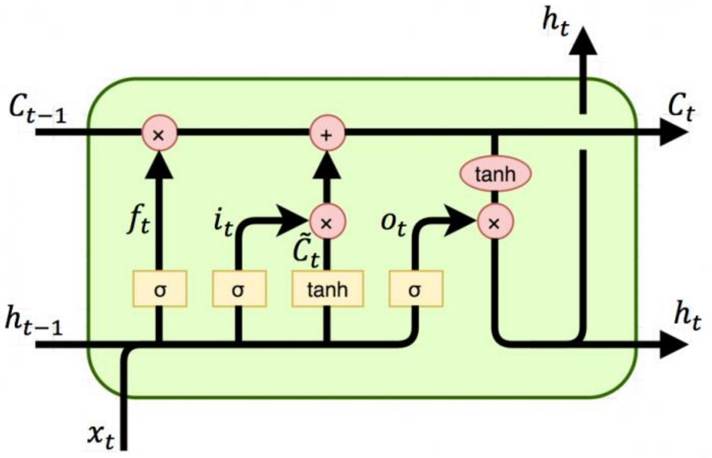
h_{t-1} previous hidden state

X_t input

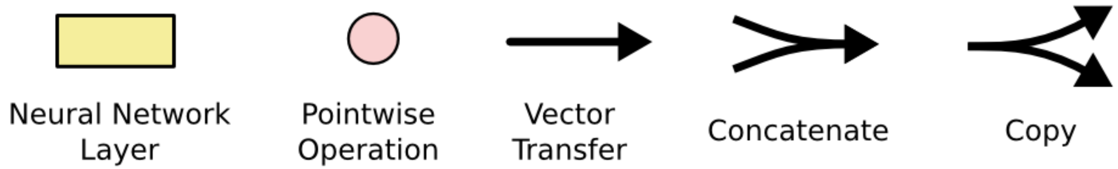
concatenation

$$
\begin{aligned}
\boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} \\
\boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}) \\
\boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)} \\
\hat{\boldsymbol{y}}^{(t)} &= \mathrm{softmax}(\boldsymbol{o}^{(t)})
\end{aligned}
$$

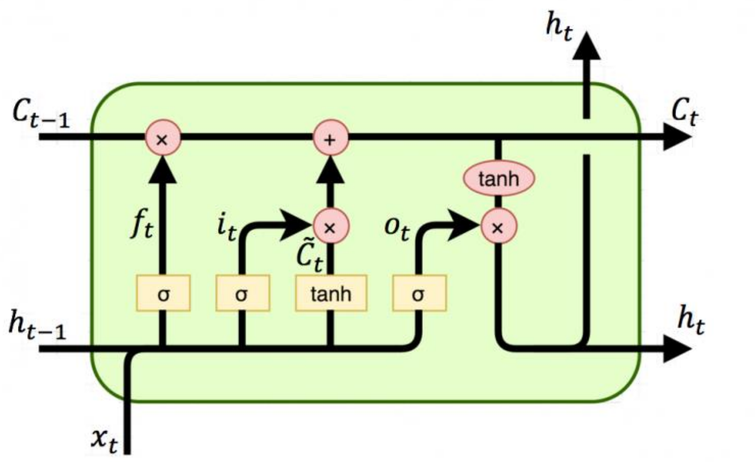
LSTM (Long Short-Term Memory)



LSTM

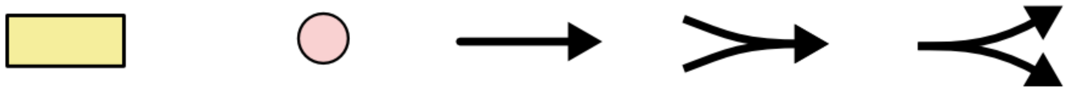


LSTM (Long Short-Term Memory)



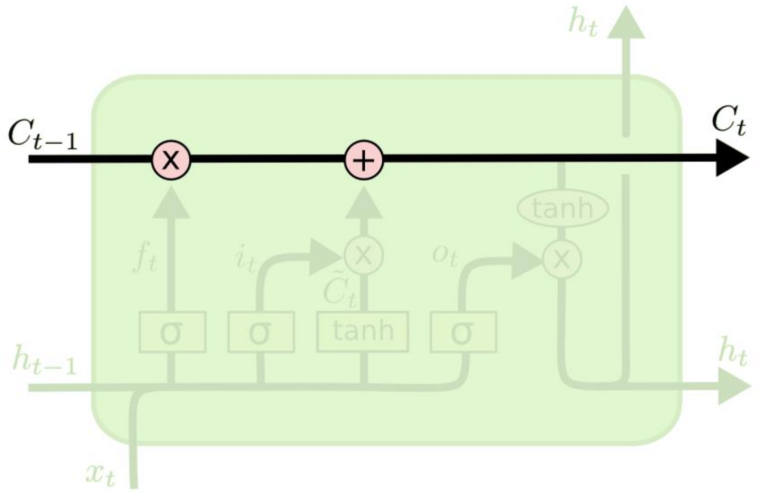
LSTM

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c)$$
$$h_t = o_t \circ \sigma_h(c_t)$$



Neural Network Layer | Pointwise Operation | Vector Transfer | Concatenate | Copy

LSTM: Detailed Architecture

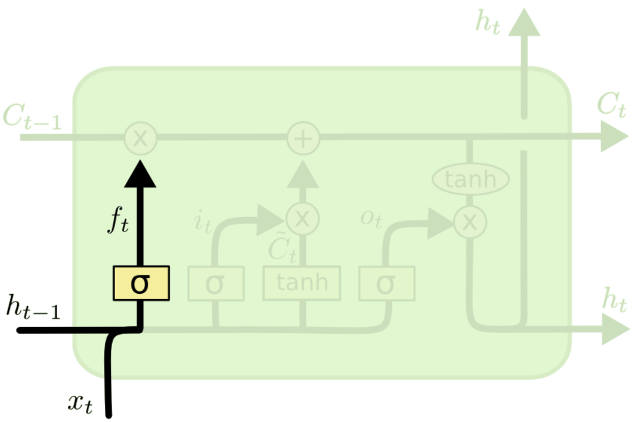


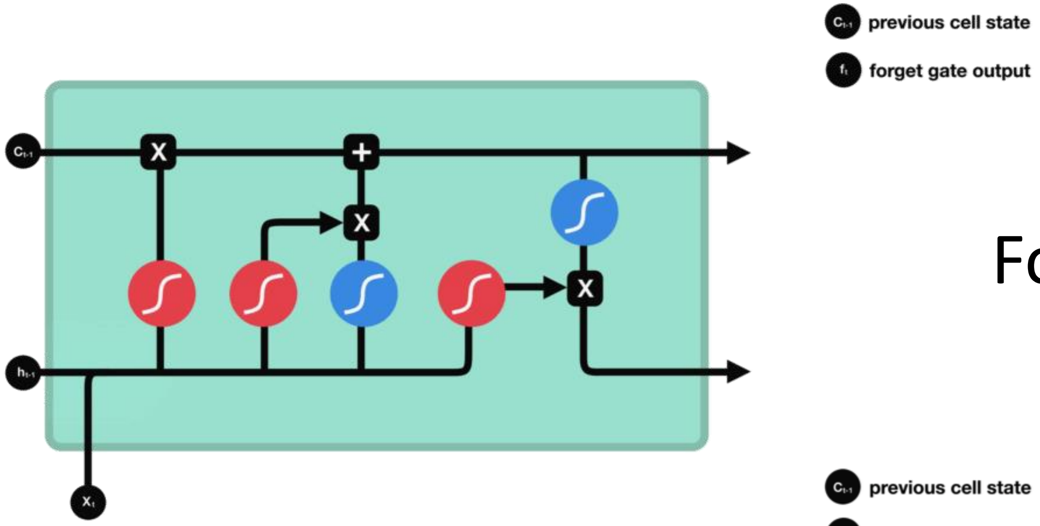
Cell state

- Unique to LSTM
- Long term memory of the model

LSTM: Detailed Architecture

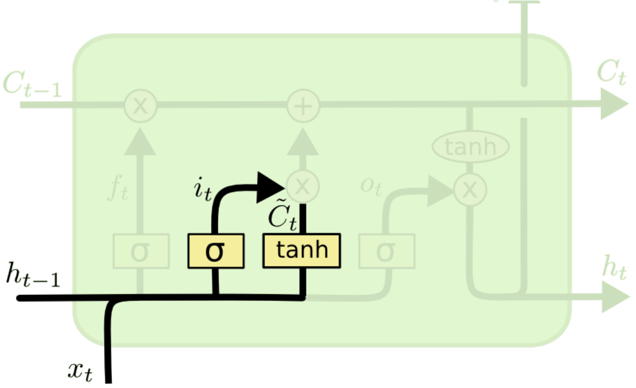
$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$

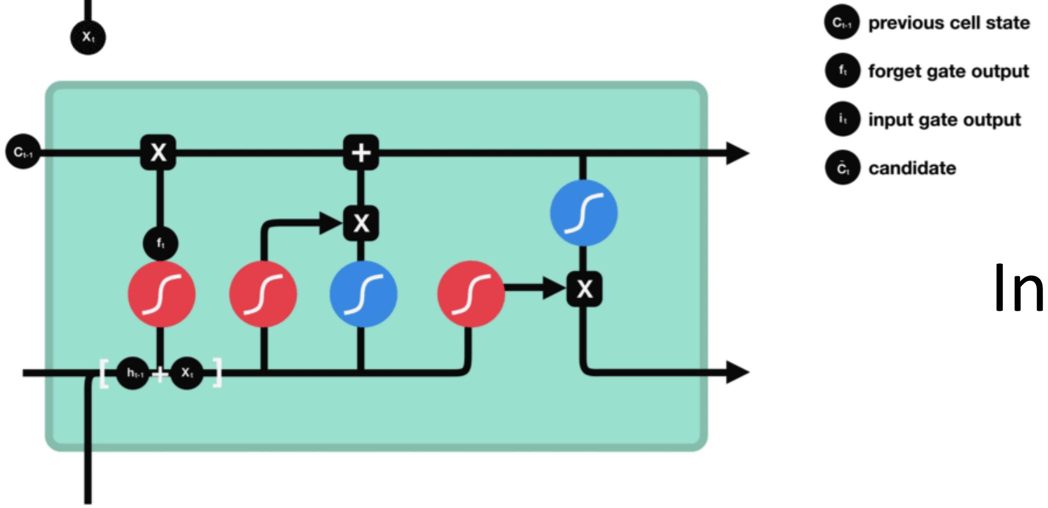




Forget gate layer

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$

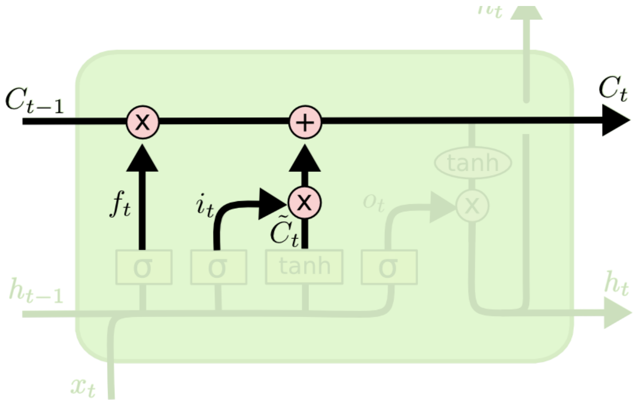


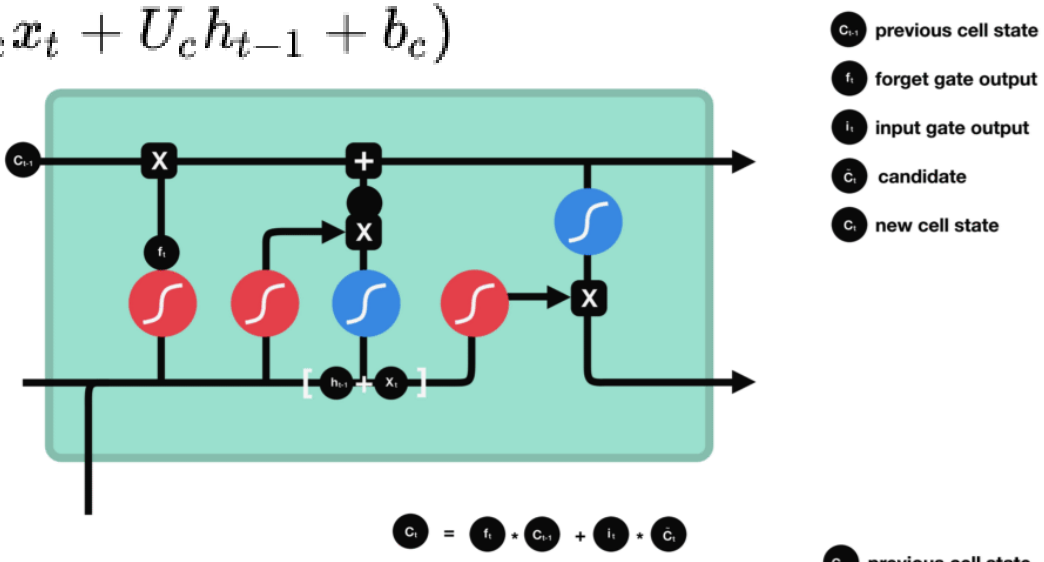


Input gate layer

LSTM: Detailed Architecture

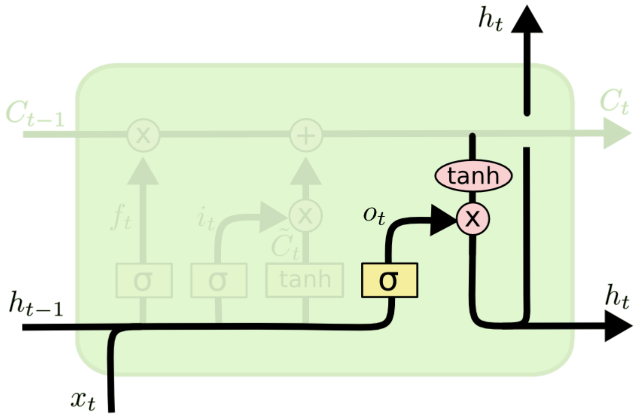
$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c)$$

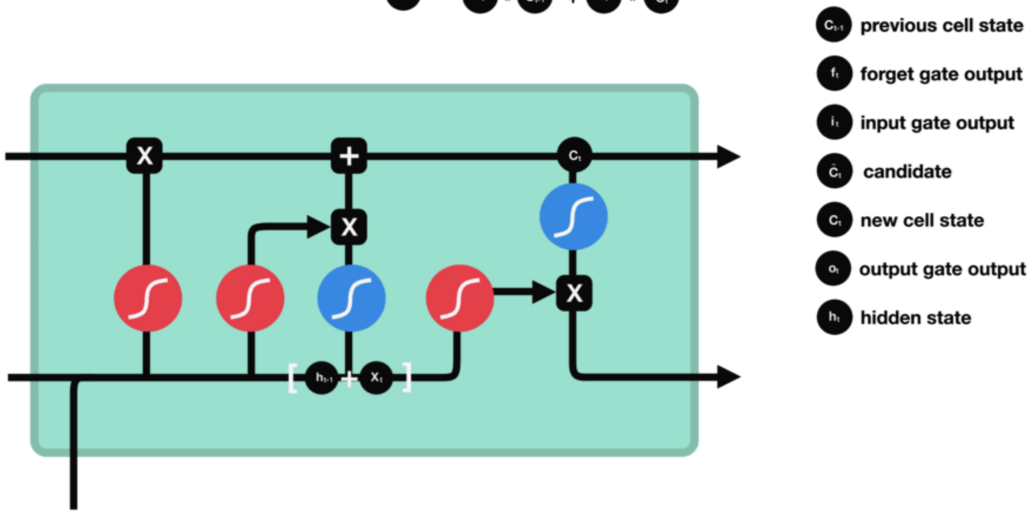




Update cell state

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$





Output gate layer

LSTM: Detailed Architecture

Forget gate

Decides what is relevant to keep from previous steps

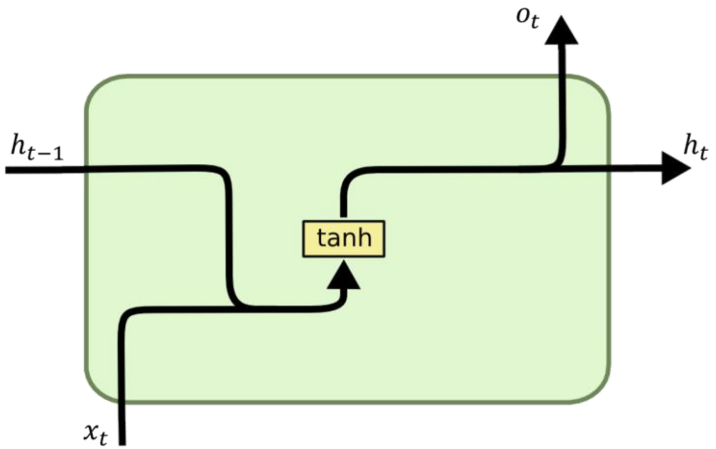
Input gate

Decides what information is relevant to add from the current step

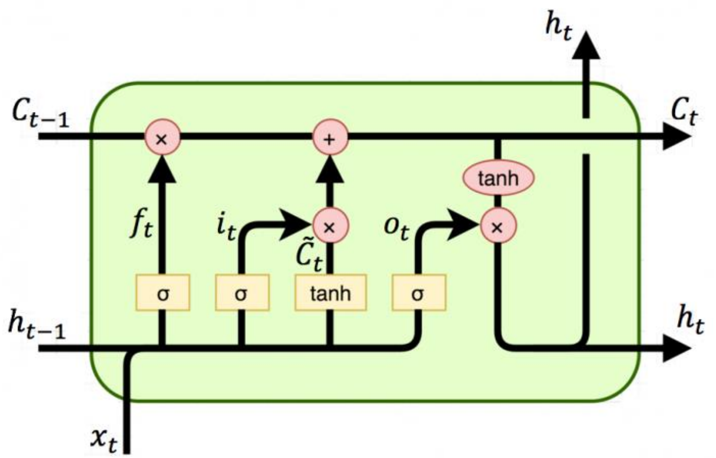
Output Gate

Determines what the next hidden state should be

LSTM (Long Short-Term Memory)



Vanilla RNN



LSTM

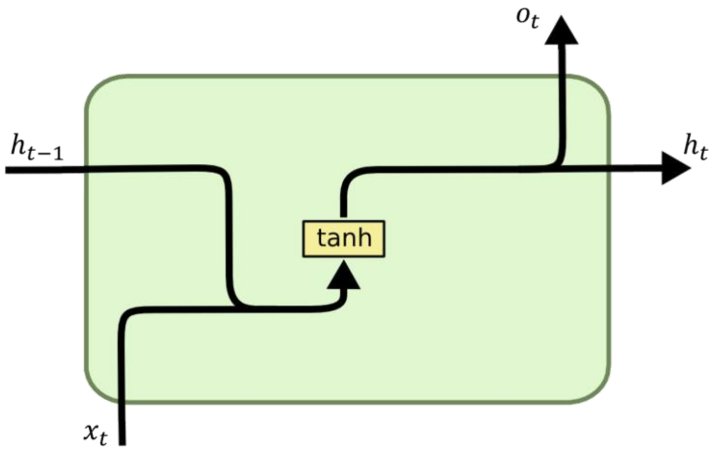
$$h_t = \sigma(w h_{t-1}).$$

$$\frac{\partial h_{t'}}{\partial h_t} = \prod_{k=1}^{t'-t} w \sigma'(w h_{t'-k})$$

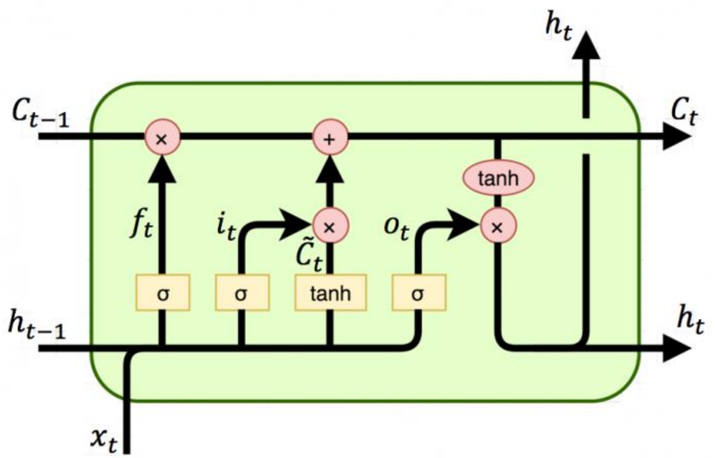
$$= \underbrace{w^{t'-t}}_{!!!} \prod_{k=1}^{t'-t} \sigma'(w h_{t'-k})$$

$$\frac{\partial c_{t'}}{\partial c_t} = \prod_{k=1}^{t'-t} \sigma(v_{t+k}).$$

LSTM (Long Short-Term Memory)



Vanilla RNN



LSTM

$$h_t = \sigma(wh_{t-1}).$$

$$\frac{\partial h_{t'}}{\partial h_t} = \prod_{k=1}^{t'-t} w\sigma'(wh_{t'-k})$$

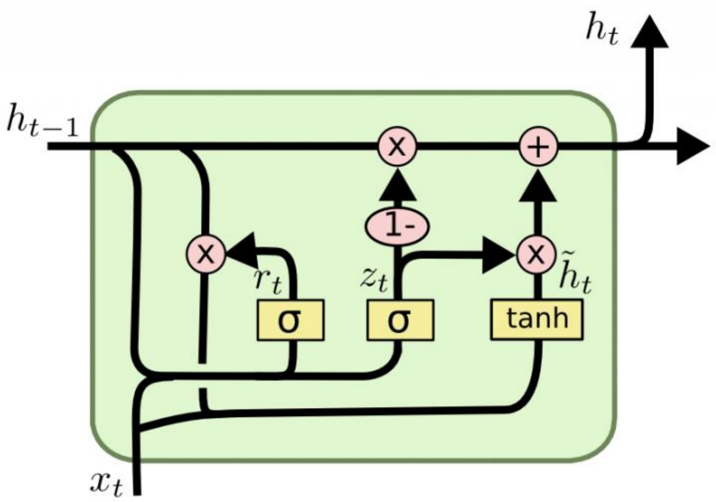
$$= \underbrace{w^{t'-t}}_{!!!} \prod_{k=1}^{t'-t} \sigma'(wh_{t'-k})$$

$$\frac{\partial c_{t'}}{\partial c_t} = \prod_{k=1}^{t'-t} \sigma(v_{t+k}).$$

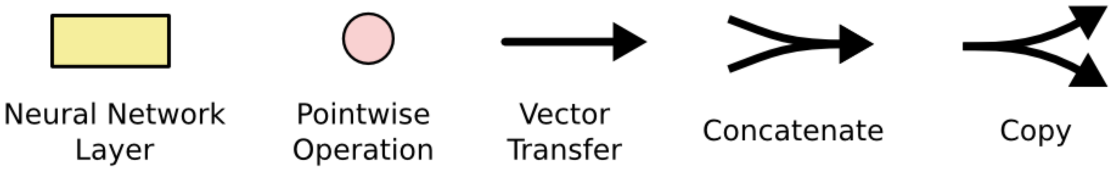
Gradient decays or grow exponentially if $w \neq 1$

No exponential decay or growth term

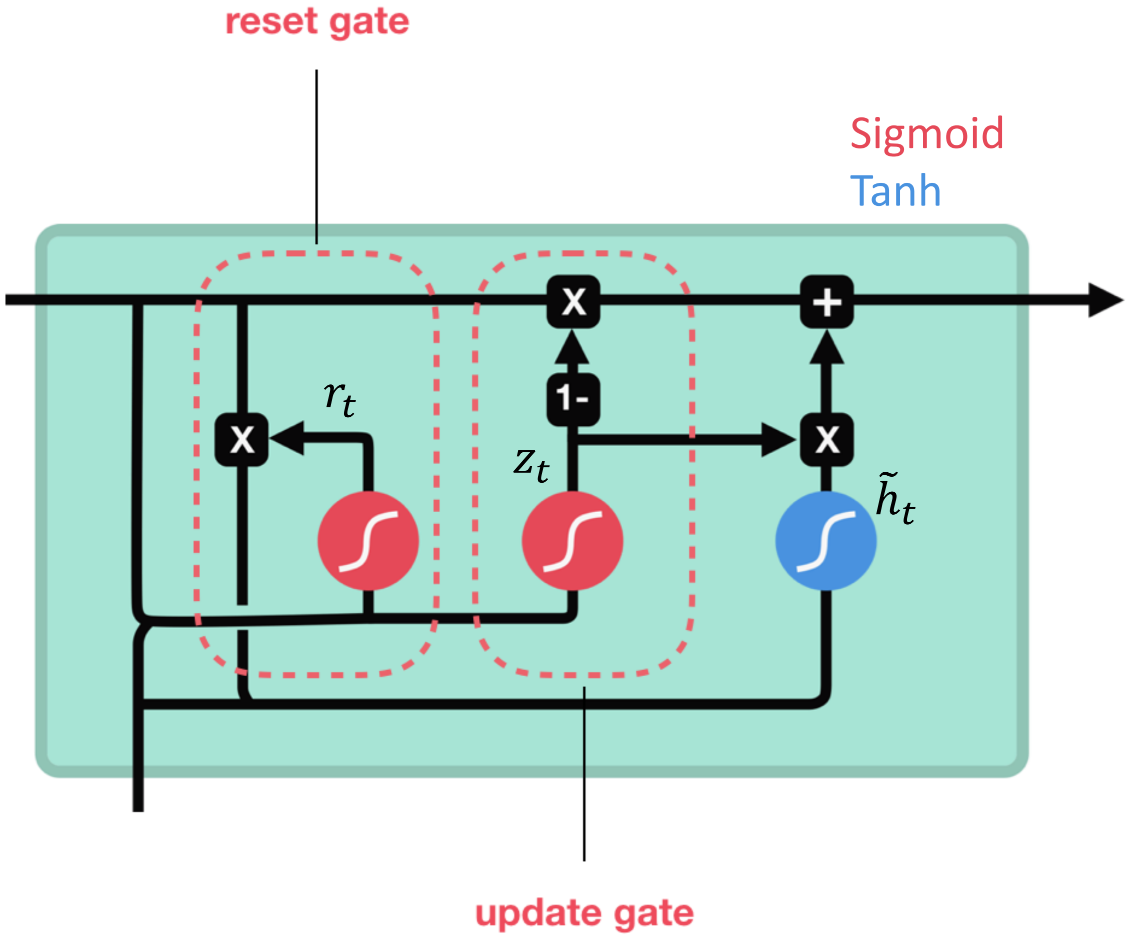
Gated RNNs



GRU



GRU: Detailed Architecture



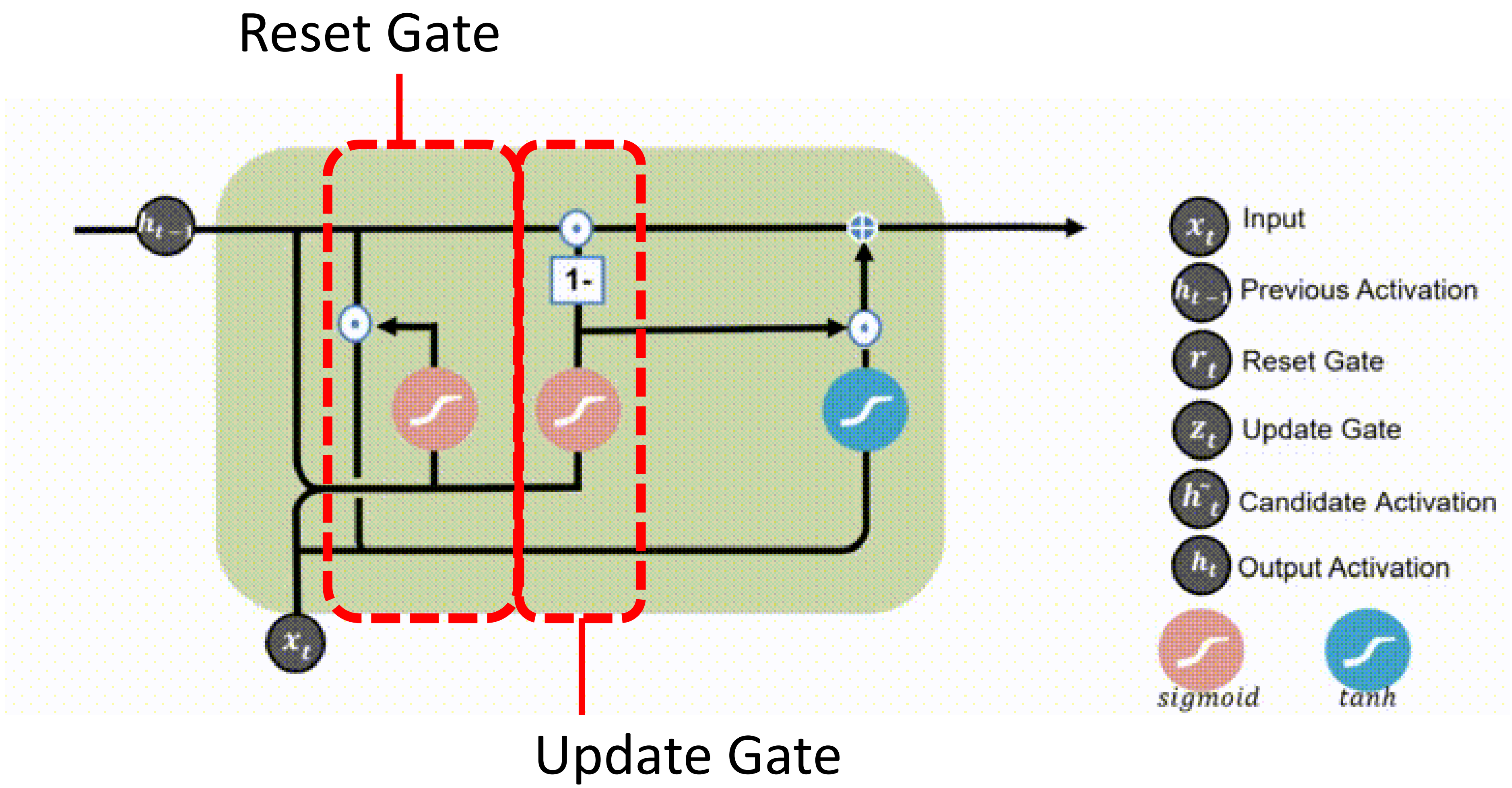
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Information Flow in GRU



LSTM: Detailed Architecture

Update gate

How much of the past information needs to be retained

Reset gate

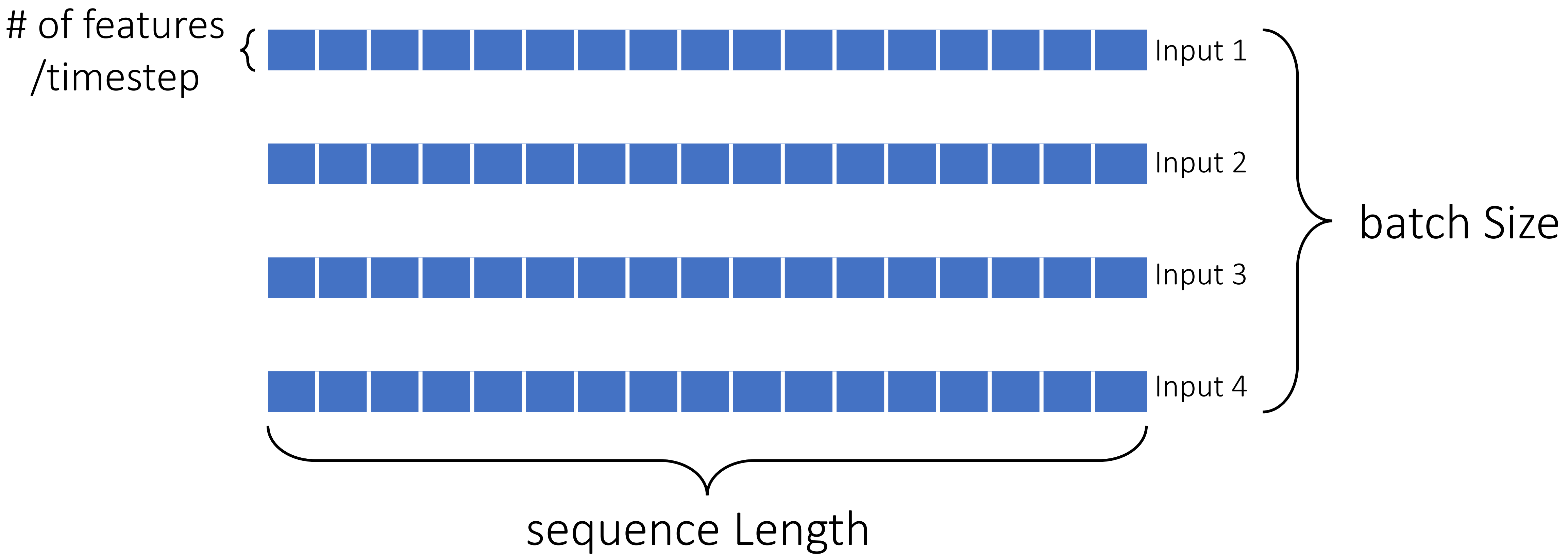
How much of the past information to forget

TRAINING GATED RNNs

Mini-batch Gradient in RNNs

RNN Extensions in LSTM/GRU

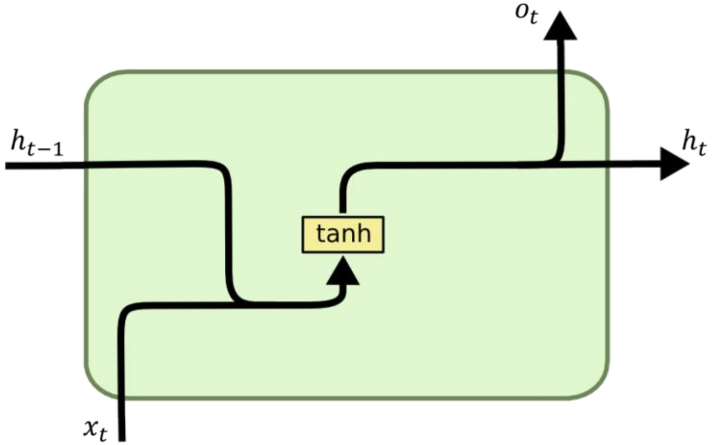
Mini-batch Gradient in RNNs



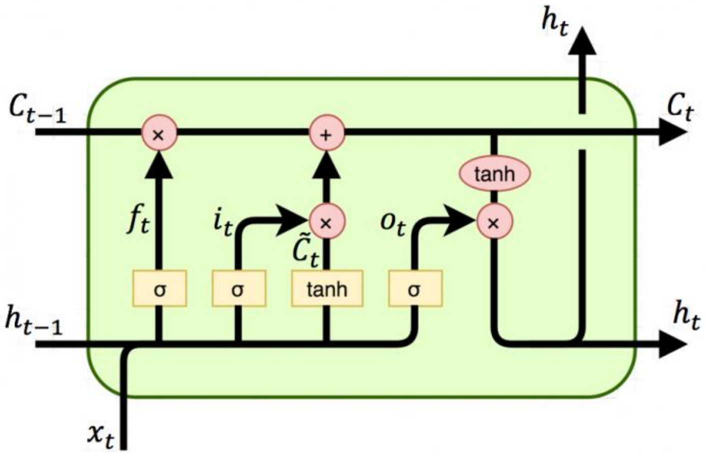
Mini-batch = set of sequences

Each timestep can be associated with an array

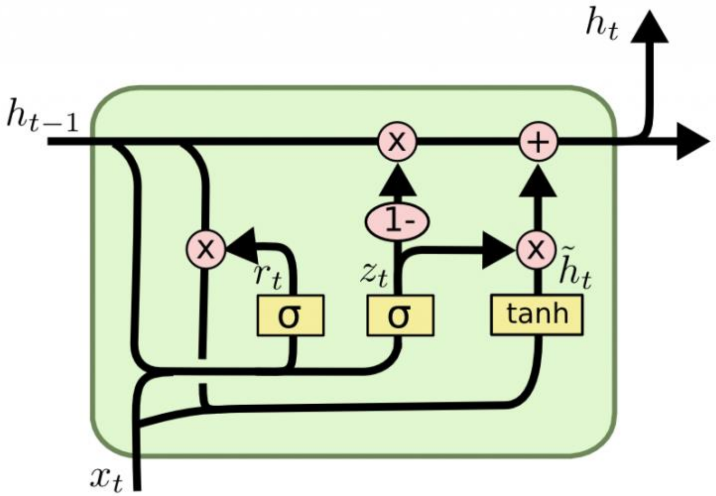
Gated RNNs



Vanilla RNN



LSTM

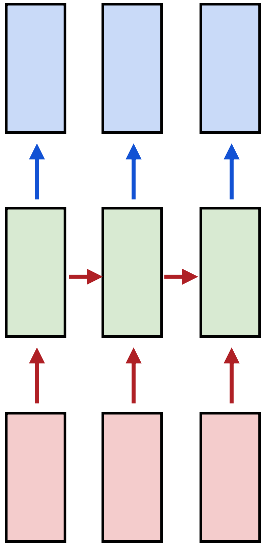


GRU

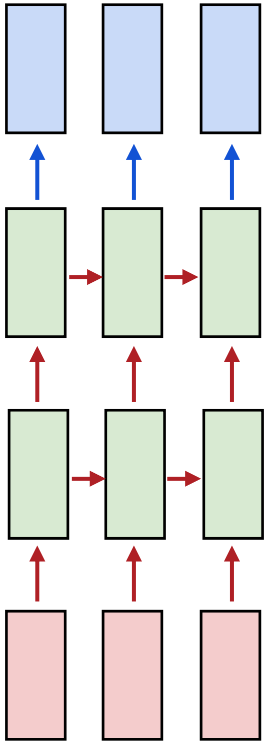
Inputs = x_t

Outputs = $f(h(t))$

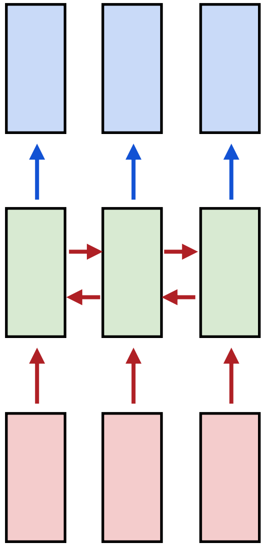
RNN Extensions in LSTM/GRU



Regular RNN



Deep RNN



Bi-directional RNN

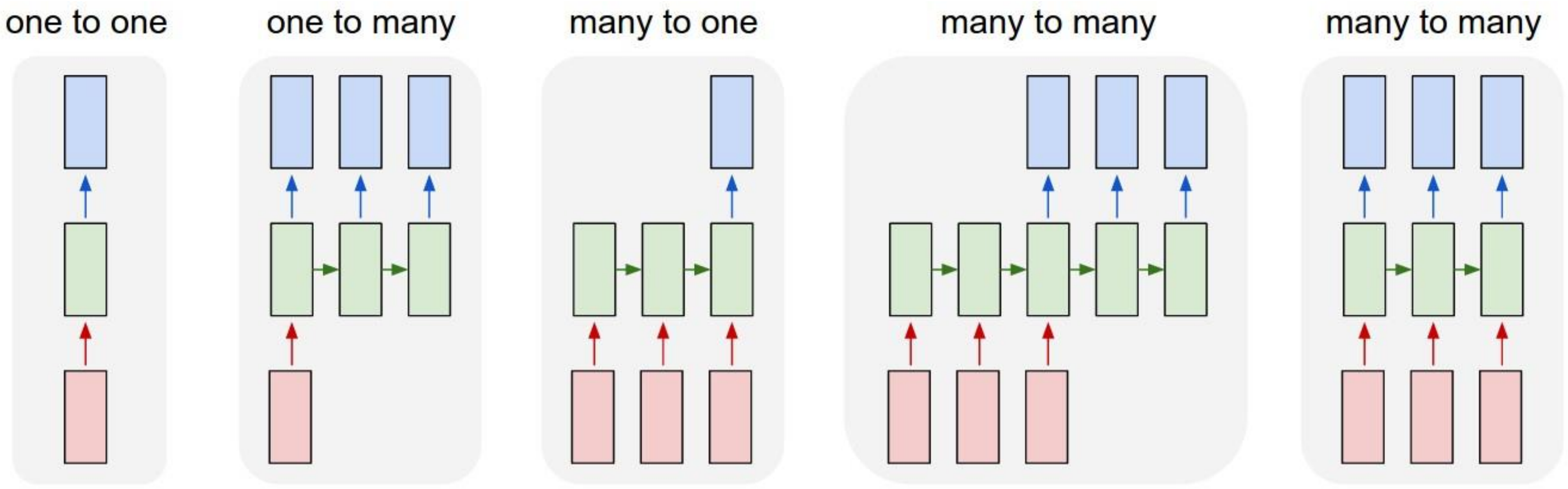
ENCODER-DECODER RNNs

Many-to-Many RNN Recap

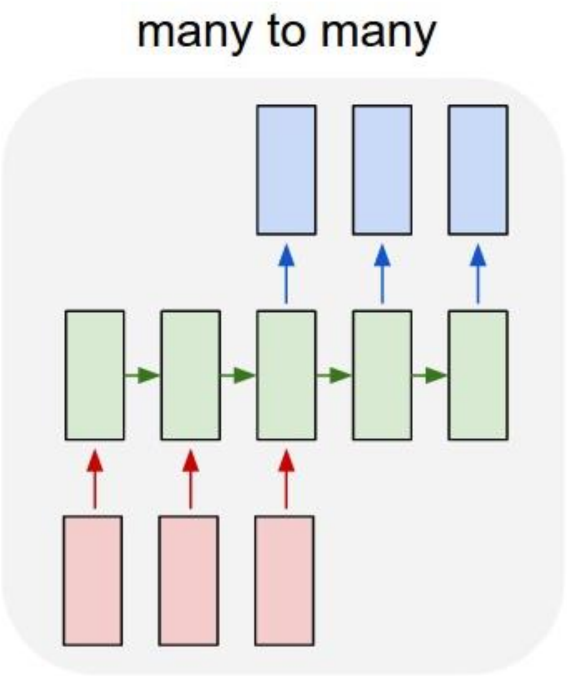
Encoder-Decoder Architecture

Training Encoder-Decoder RNNs

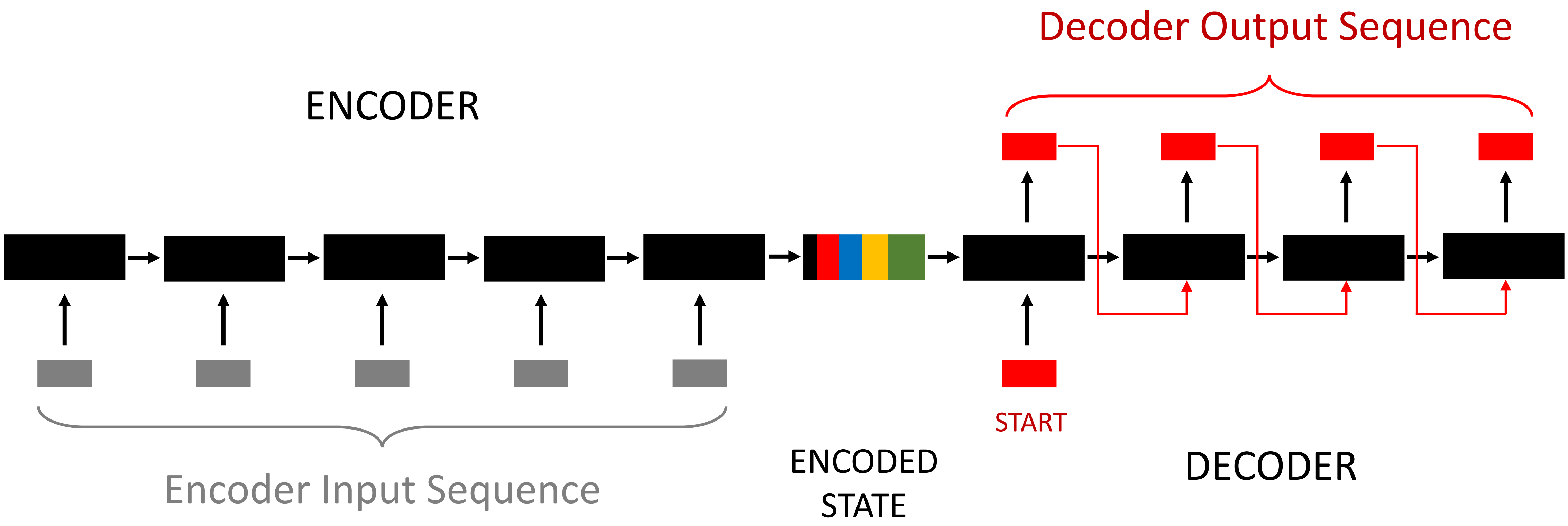
RNN Configurations



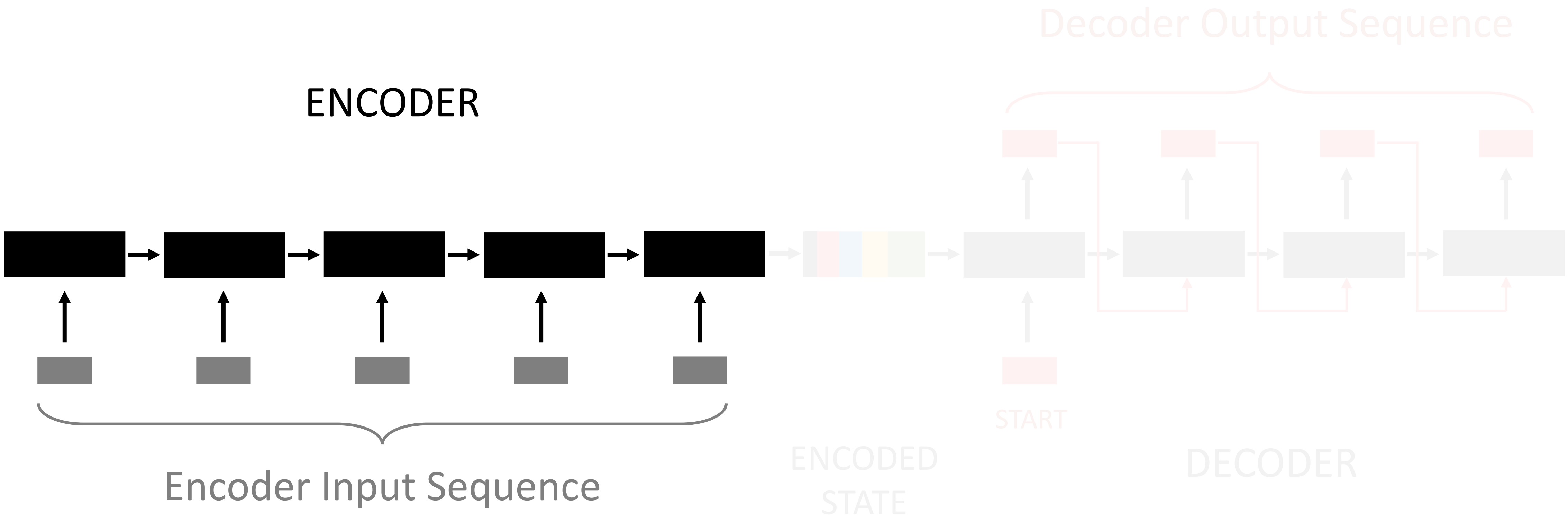
Many-to-Many



Encoder-Decoder Architecture

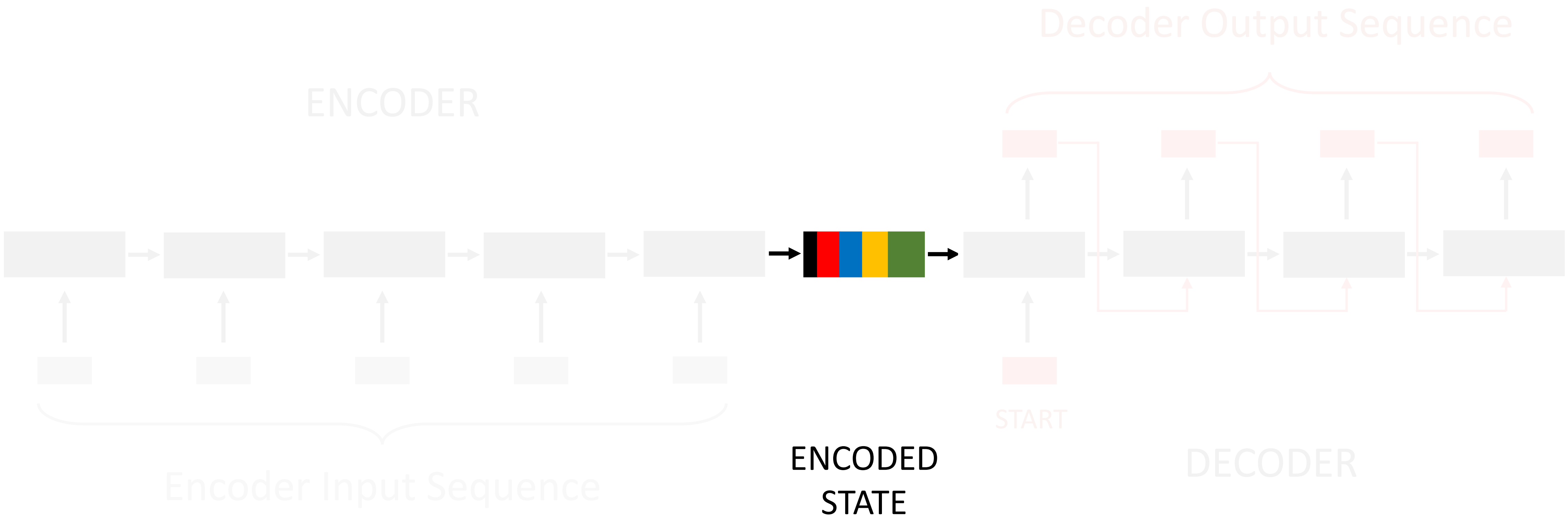


Encoder

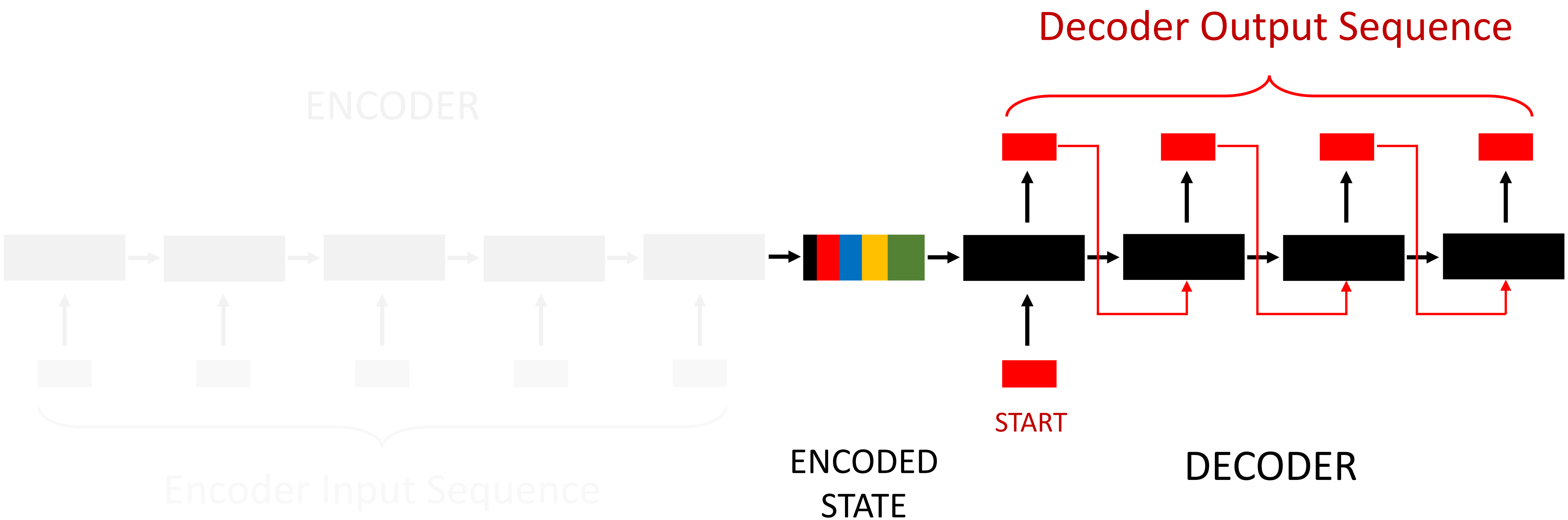


ENCODER
Encoder Input Sequence
Decoder Output Sequence
START
ENCODED
STATE
DECODER

Encoded State

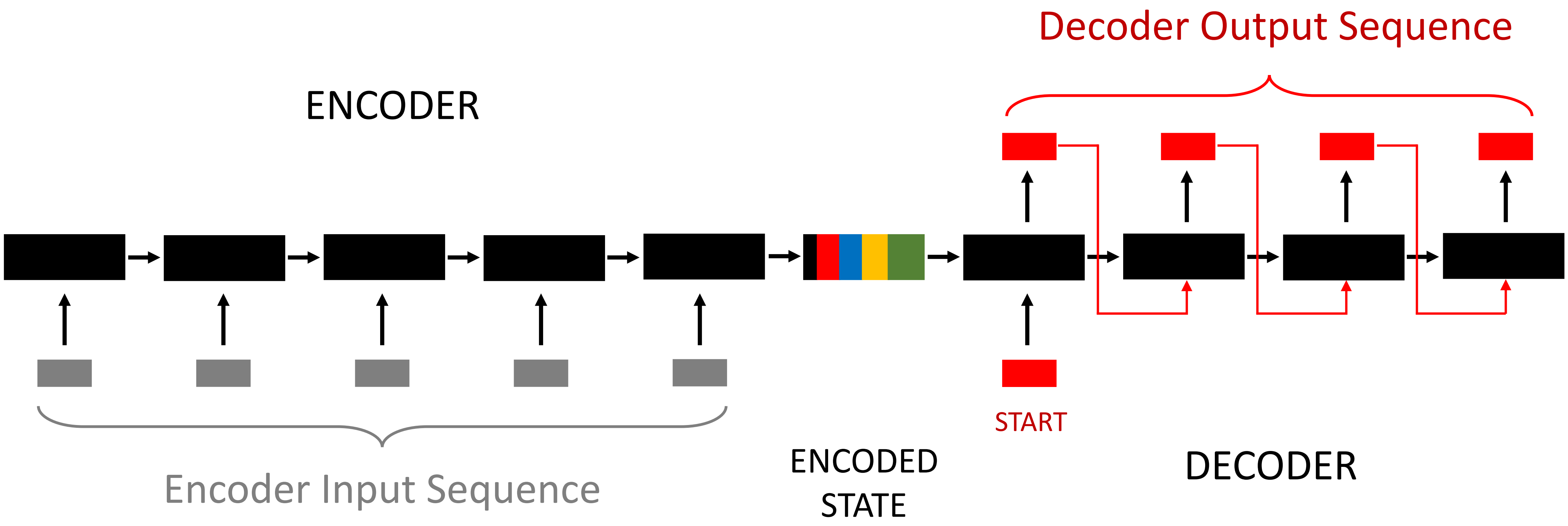


Decoder



Decoder Output Sequence
ENCODER
START
ENCODED
STATE
DECODER
Encoder Input Sequence

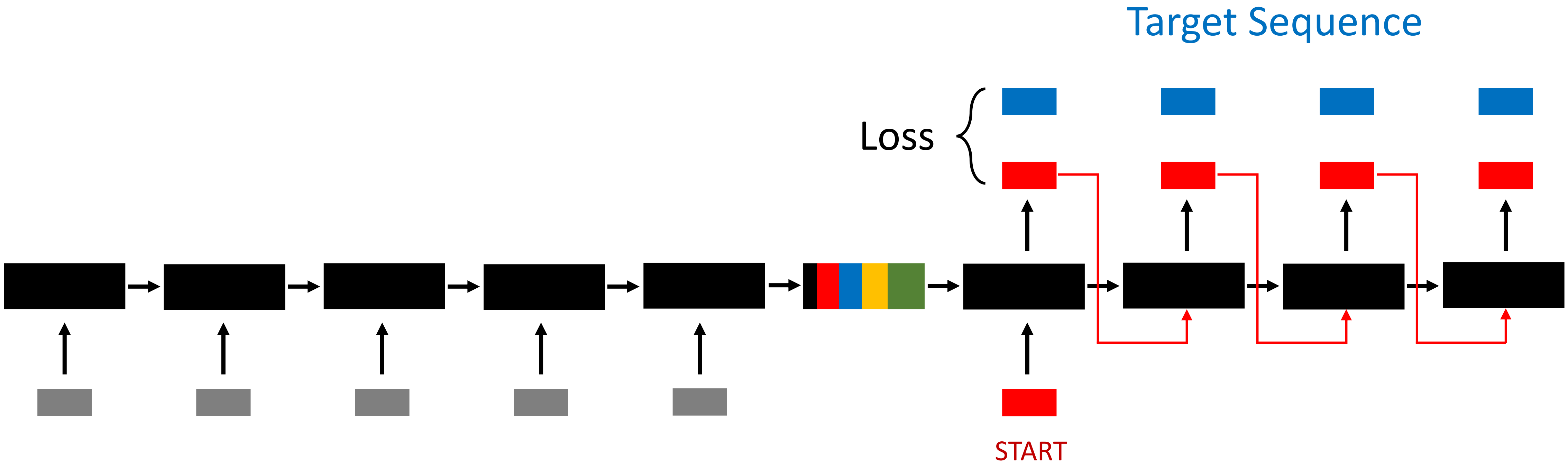
Encoder-Decoder Architecture



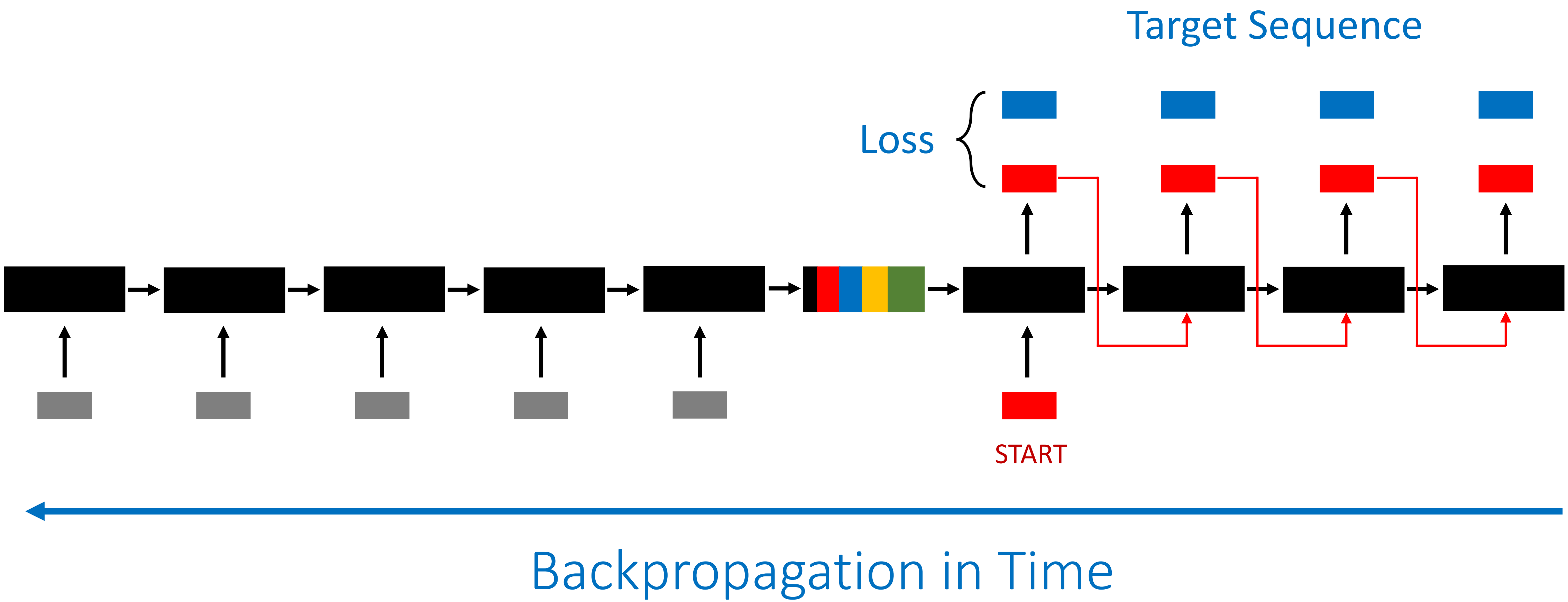
Input sequence length to Encoder (Tx) can be different from the output sequence length of Decoder (Ty)

TRAINING ENCODER-DECODER

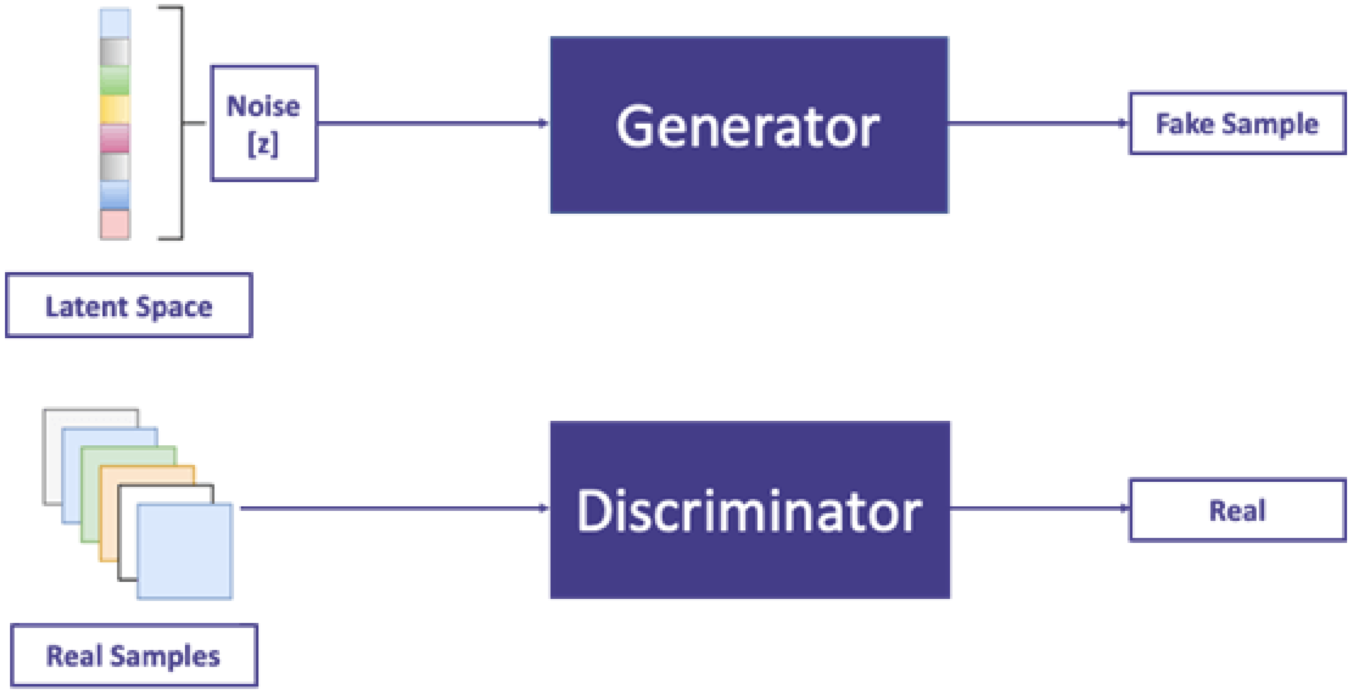
Training Encoder-Decoder



Training Encoder-Decoder



Next episode in EEP 596…



Generative Adversarial Networks