Recurrent neural networks

Last time

- Transfer learning
- Weight initialization
- Dropout
- Prevent over-training

This time

- Sequential modeling

- How can we model sequential relations using deep neural networks?
- Recurrent Neural Networks (RNNs)

- How can we optimize gradient flow over the sequential dimension ?
- Long Short Term Memory (LSTM)

Till now

- CNNs to do
- Image classification; Object detection/segmentation;
- Regression

- Mostly fixed size input → fixed size output
- How would we do
  - Image captioning [one image to variable number of words]
  - Sentiment classification [variable number of words to one sentiment]
  - Translation [variable number of words to variable number of words]
- Recurrent Neural Networks (RNNs)

Examples of sequence data

- Speech recognition
- Music generation
- Sentiment classification
- DNA sequence analysis
- Machine translation
- Video activity recognition

Recurrent neural networks

one to one | one to many | many to one | many to many | many to many

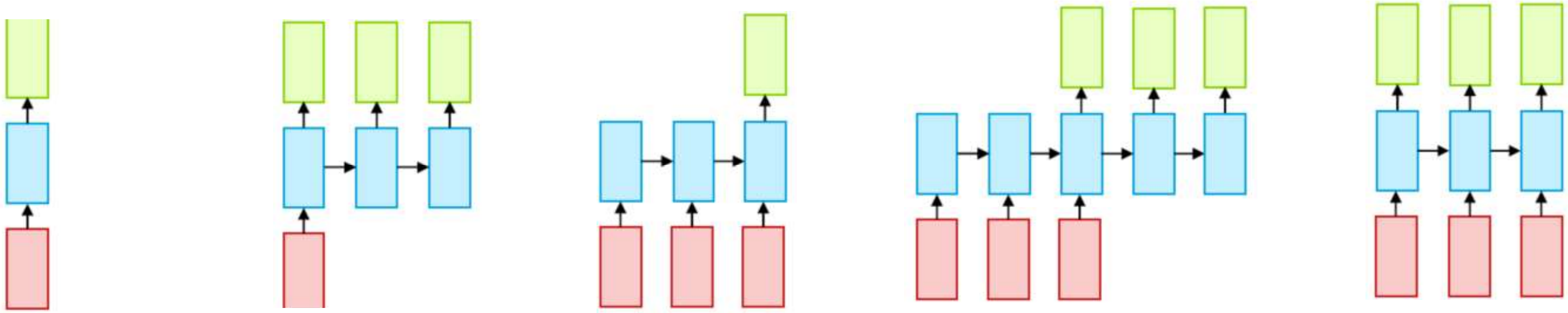


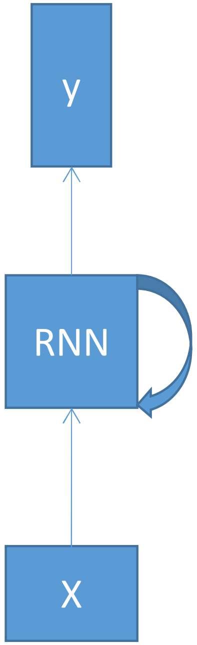
Image captioning

Text sentiment classiciation

Machine translation
many to many
Video

Video frame classification

Recurrent neural networks



- Recurrence relation
- For a certain time □

Function with parameter □ | Input

Note: parameters W fixed over time

$h_t = f_w(h_{t-1}, x_t)$

New state | Old state

- Straightforward option:
- Two linear relations:
  - old state → new state
  - Input → new state

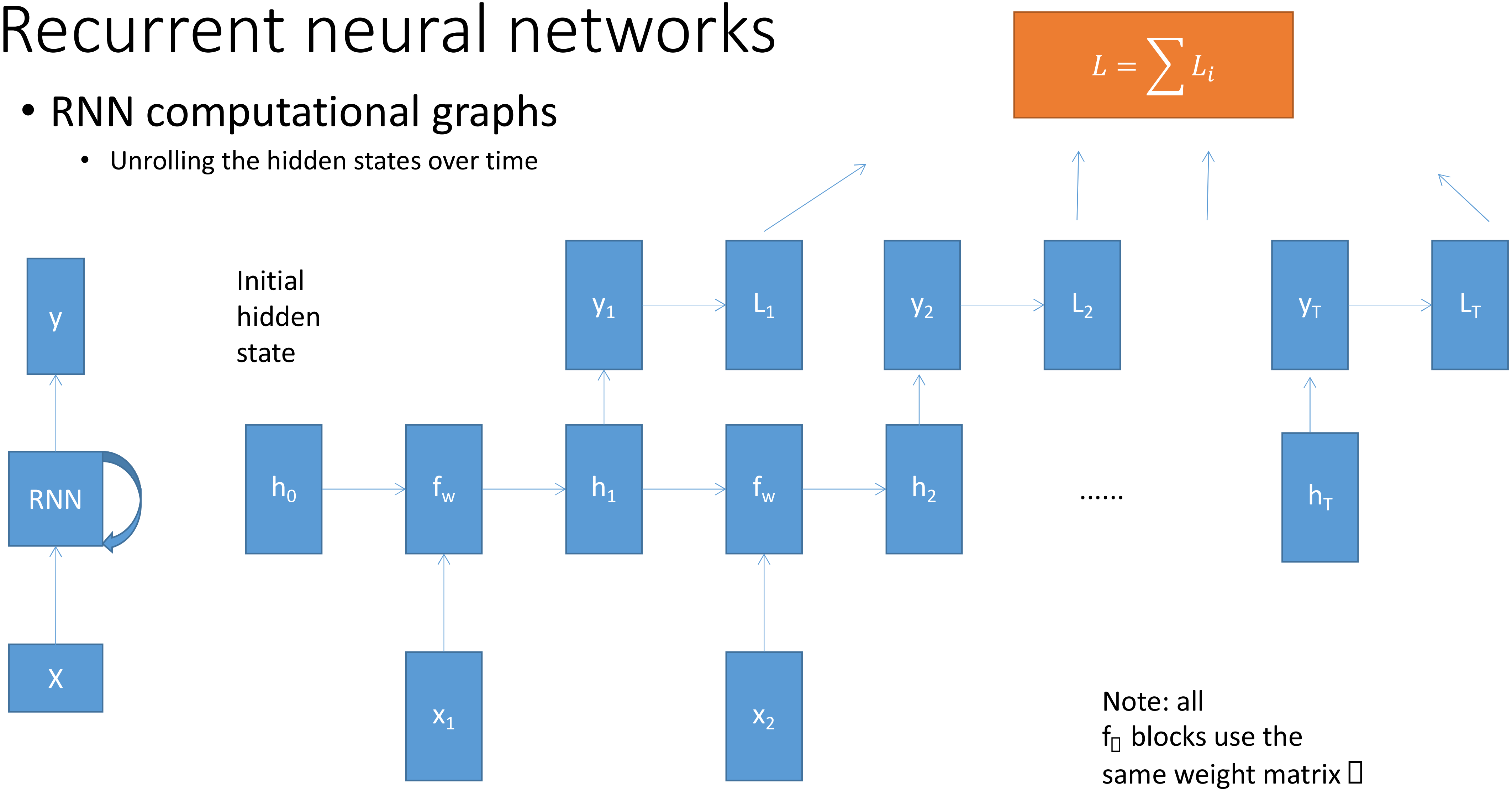
$h_t = \tanh(W_{hh} h_{h-1} + W_{xh} x_t)$

- Linear output relation:
  - New state → output

$y_{\square} = \square$ □ □ □ □

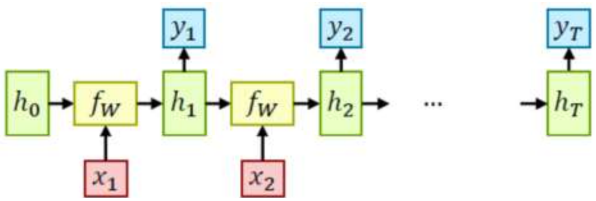
Recurrent neural networks

- RNN computational graphs
  - Unrolling the hidden states over time



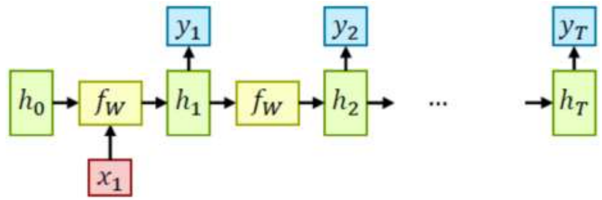
Recurrent neural networks

RNN computational graphs



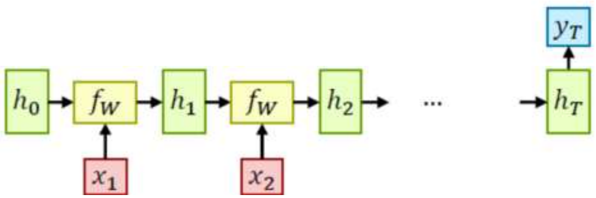
Many
to
many

Use single input to initialize
the hidden state of the model



One
to
many

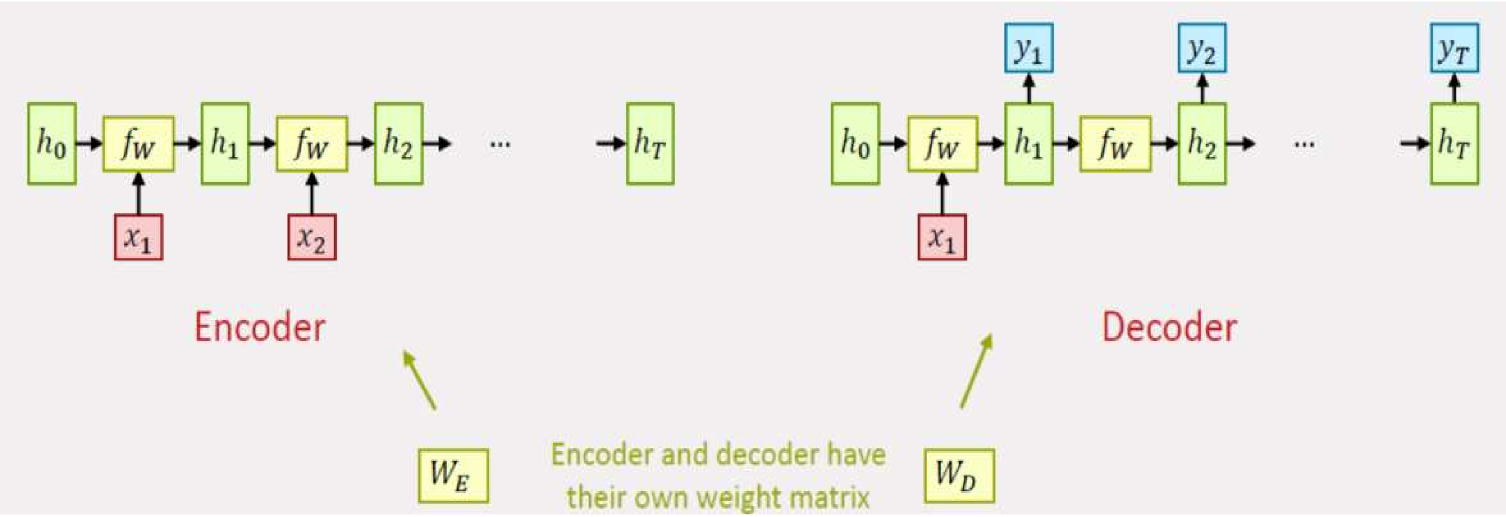
Many
to
one



Final hidden state
summarized all of the context
of the entire sequence

Recurrent neural networks

- RNN computational graphs: Sequence to sequence
- Many to one followed by one to many

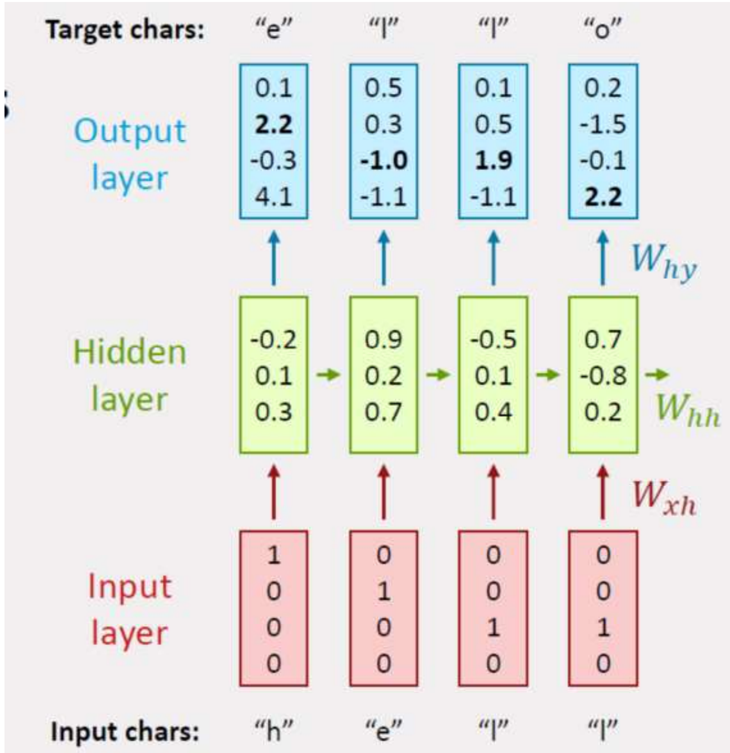


Recurrent neural networks

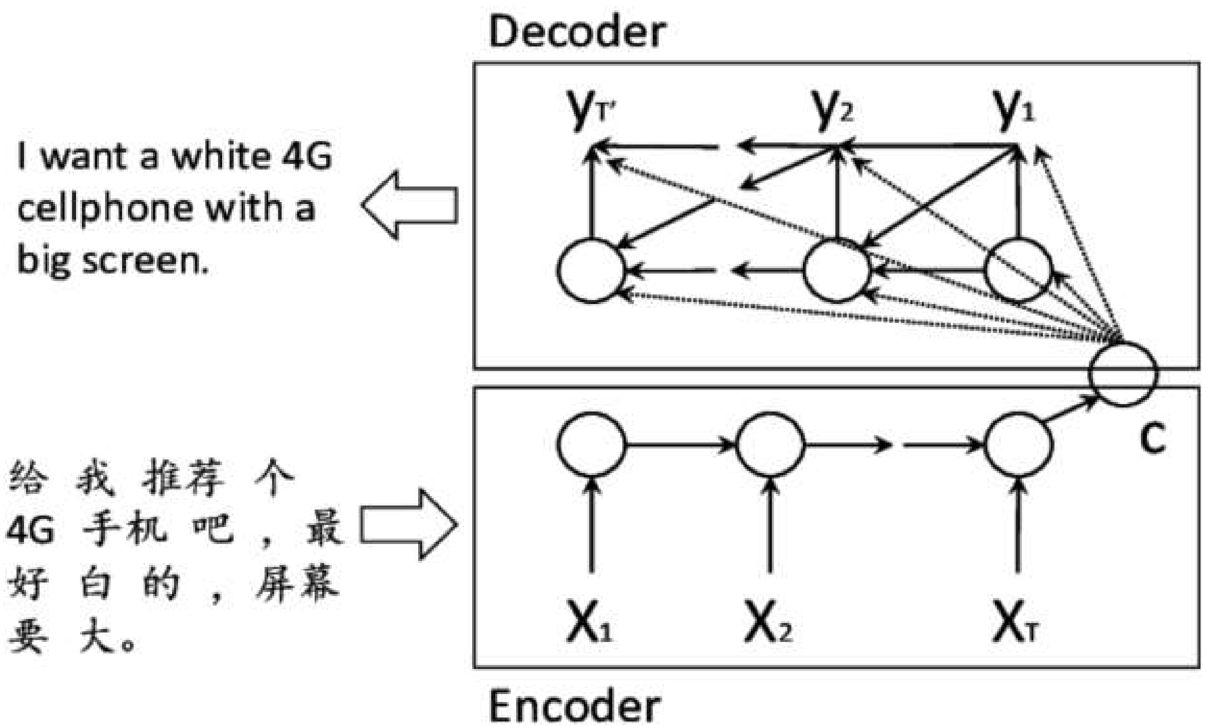
- RNN computational graphs:
  - Sequence to sequence
  - Many to one followed by one to many

Softmax

- Example
  - character level language model
  - Predict next letter
  - Vocabulary: [h, e, l, o]
  - Training sequence: “hello”



Recurrent neural networks- applications



https://www.researchgate.net/figure/An-illustration-of-the-RNN-based-neural-network-model-for-Chinese-to-English-machine_fig1_306093825

Recurrent neural networks- applications

- This License refers to version of the GNU General Public License. Copyright also means copyright-bick,
- Remade me any thing to his sword
- To his salt and most hidden loose to be so for sings, but not in a libutt of his matter than that shall be sure as will be soldye
- As master compary, do not live in traitor.
- Bless thy five wits!

- Using AI to mix Shakespearean and Modern English

https://www.knime.com/blog/use-deep-learning-to-write-like-shakespeare

What you can do with RNNs

Image captioning

What you can do with RNNs

```
static void do_command(struct seq_file *m, void *v)
{
  int column = 32 << (cmd[2] & 0x80);
  if (state)
    cmd = (int)(int_state ^ (in_8(&ch->ch_flags) & Cmd) ? 2 : 1);
  else
    seq = 1;
  for (i = 0; i < 16; i++) {
    if (k & (1 << 1))
      pipe = (in_use & UMXTHREAD_UNCCA) +
        ((count & 0x00000000fffffff8) & 0x000000f) << 8;
    if (count == 0)
      sub(pid, ppc_md.kexec_handle, 0x20000000);
    pipe_set_bytes(i, 0);
  }
  /* Free our user pages pointer to place camera if all dash */
  subsystem_info = &of_changes[PAGE_SIZE];
  rek_controls(offset, idx, &soffset);
  /* Now we want to deliberately put it to device */
  control_check_polarity(&context, val, 0);
  for (i = 0; i < COUNTER; i++)
    seq_puts(s, "policy ");
}
```

- Produce code

RNN practice

Follow use case：

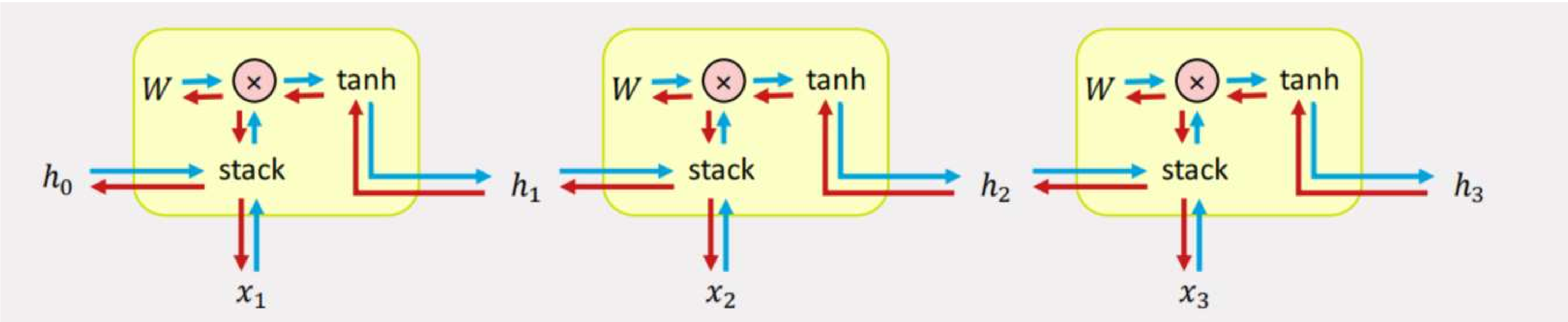
https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn

RNN gradient flow

- What happens during RNN training?

$$
\begin{aligned}
h_t &= \tanh(W_{hh}h_{t-1} + W_{xh}x_t) \\
&= \tanh\left(\begin{pmatrix} W_{hh} & W_{xh} \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right) \\
&= \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right)
\end{aligned}
$$

During backwards pass need to multiply by □



RNN gradient flow

$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial s_k} \frac{\partial s_k}{\partial W}$$

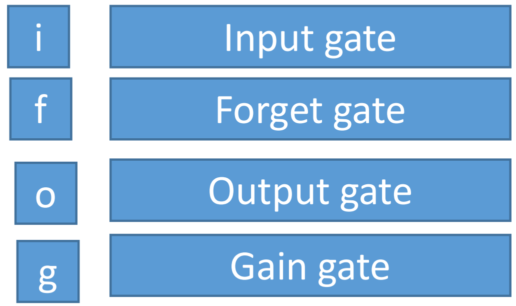
the gradient of the error with respect to the input weight W can be expressed as

$$\frac{\partial s_3}{\partial s_1} = \frac{\partial s_3}{\partial s_2} \frac{\partial s_2}{\partial s_1}$$

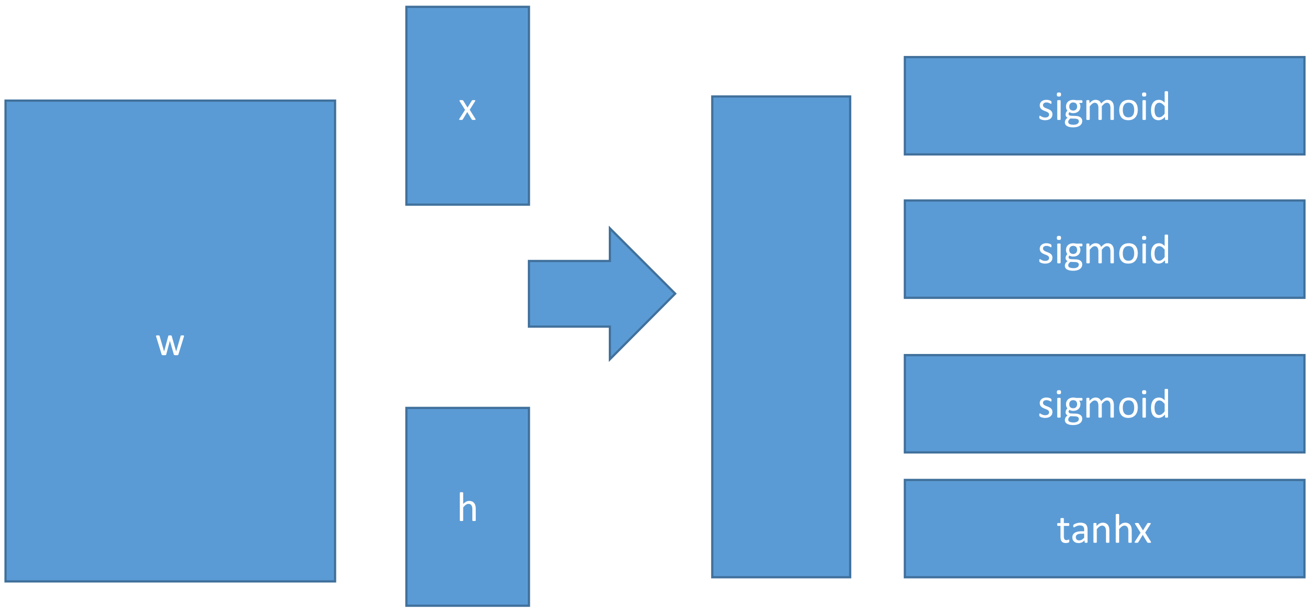
We note that this expression includes the derivative of the current hidden state s3 with respect to each of the k other hidden states — which in turn must be evaluated via the chain rule

$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \left(\prod_{j=k+1}^{3} \frac{\partial s_j}{\partial s_{j-1}} \right) \frac{\partial s_k}{\partial W}$$

Long Short Term Memory (LSTM)



- Introduction of a “cell state”



Cell state $c_t = f \odot c_{t-1} + i \odot g$

Hidden state $h_t = o \odot \tanh(c_t)$

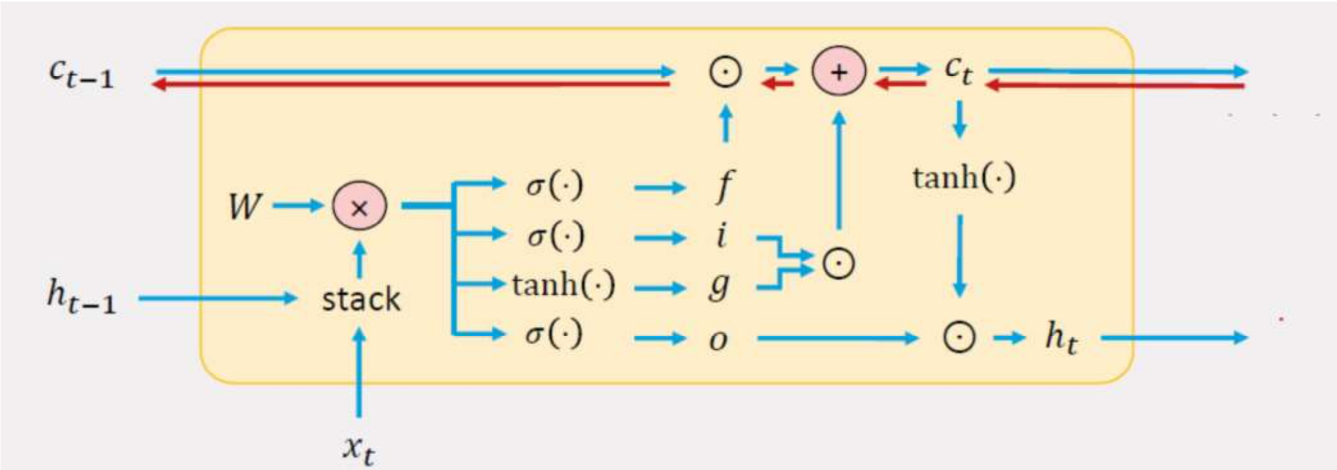
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma(\cdot) \\ \sigma(\cdot) \\ \sigma(\cdot) \\ \tanh(\cdot) \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

Long Short Term Memory (LSTM)

$c_t = f \odot c_{t-1} + i \odot g$

$h_t = o \odot \tanh(c_t)$

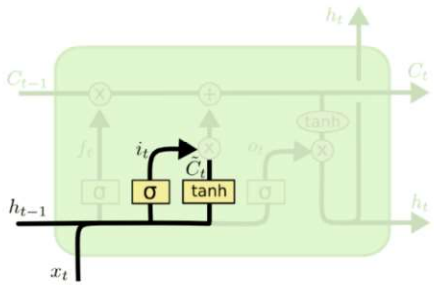
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma(\cdot) \\ \sigma(\cdot) \\ \sigma(\cdot) \\ \tanh(\cdot) \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$



During backprop from c_t to c_{t-1}:

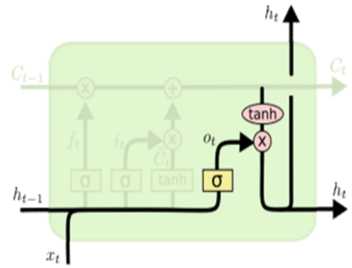
- only elementwise multiplication by f
- no matrix multiplication by W

Calculating New Values to Add to Cell State



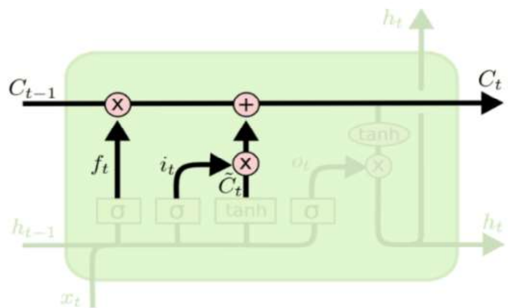
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Calculating Hidden State from Cell State



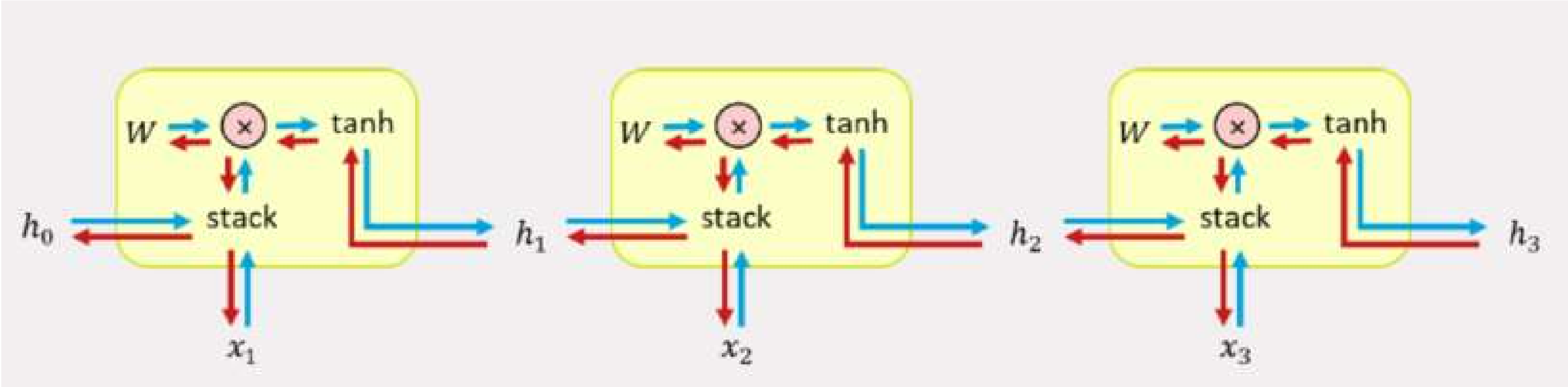
$$o_t = \sigma\left(W_o \ [h_{t-1}, x_t] \ + \ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Updating Cell State

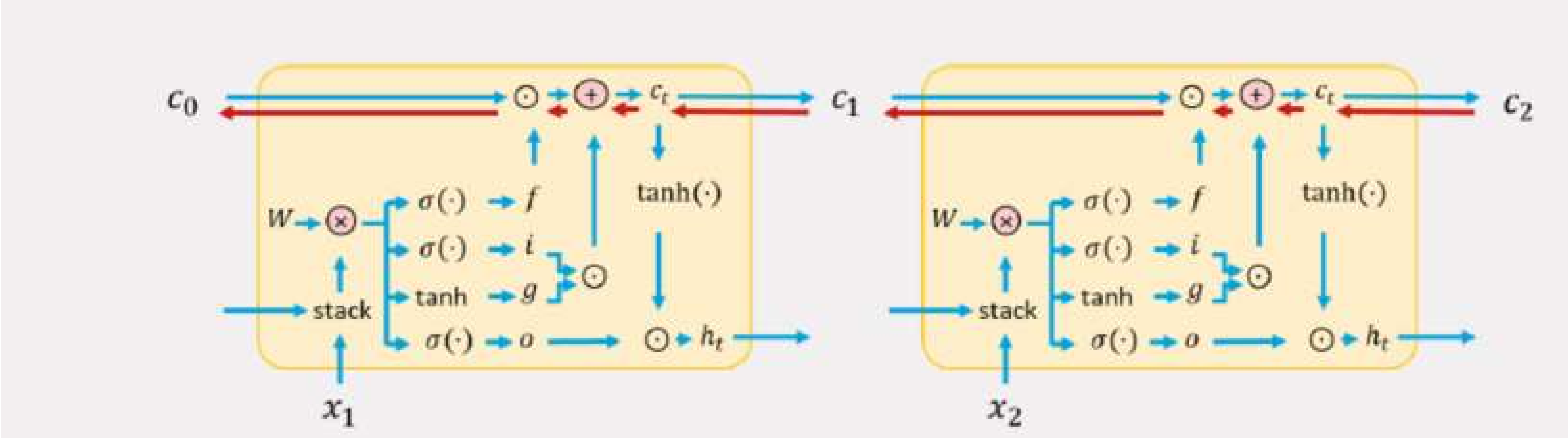


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Vanilla RNN



LSTM



Temporal modeling summary

- Recurrent Neural Networks (RNNs)
  - Add recurrent relation in the network to model sequential relations
  - Unroll the network over time and apply standard backpropagation to train
  - In practice: vanilla RNNs don’t work that well due to vanishing/exploding gradients

- Long Short Term Memory (LSTM)
  - Introduce an additional cell state, governing the input and output of the hidden state
  - Input --, output --, forget --, “gating“ gates
  - Uninterupted gradient flow

Case study

- LSTM case study
    - See Canvas

- Cifar case study

https://ermlab.com/en/blog/nlp/cifar-10-classification-using-keras-tutorial/