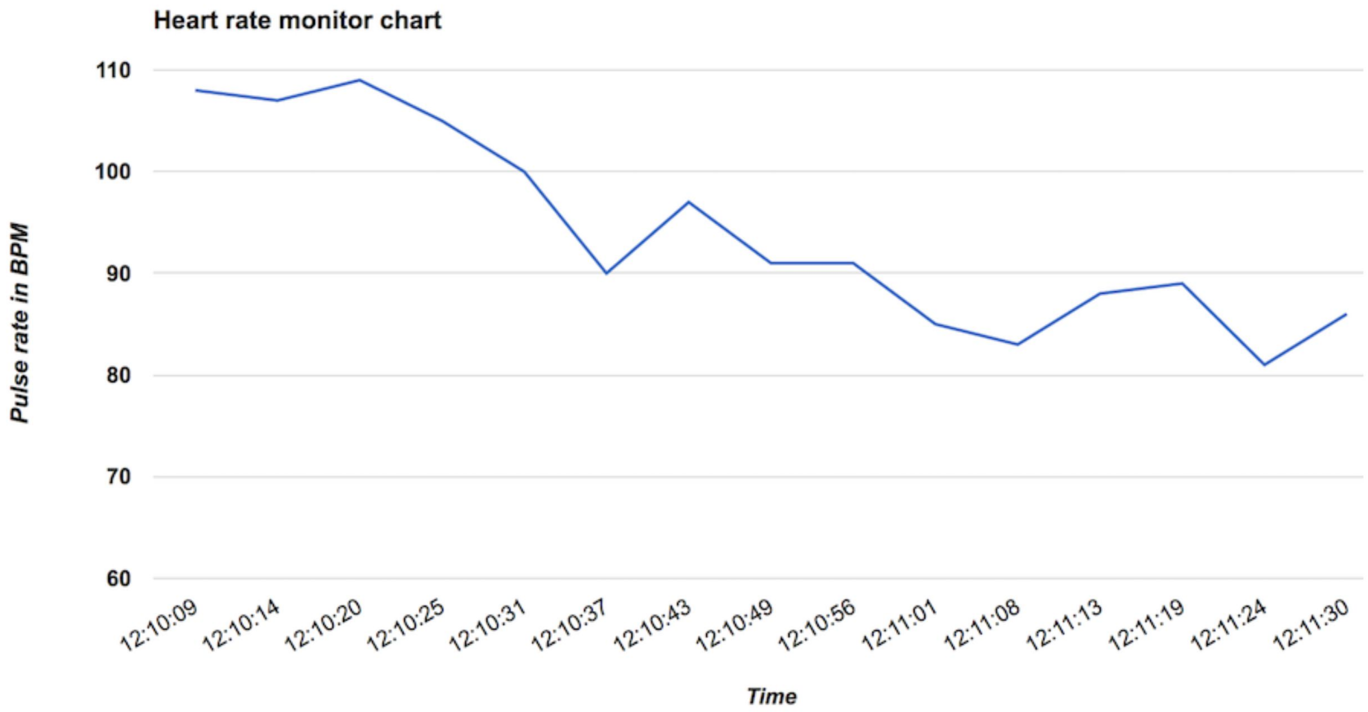
Deep Learning for NLP - Focus on Medical Applications

Recurrent Neural Networks

Working with sequential data?

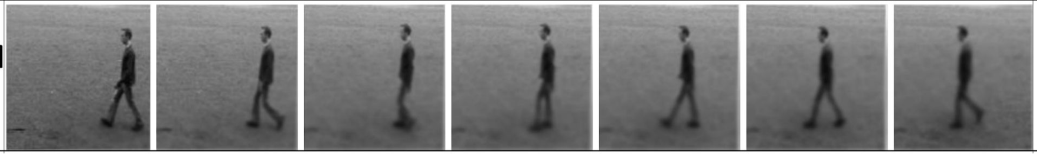
HR = 86 -> What will be the HR in 20 min?

Working with sequential data?



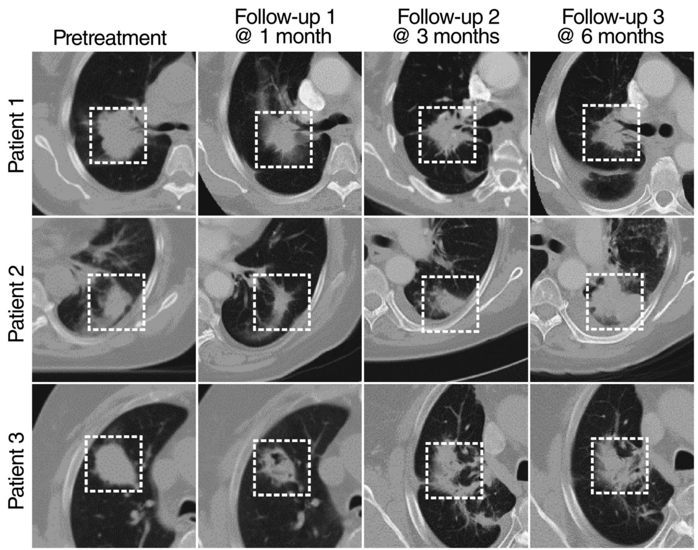
Recurrent Networks | Examples

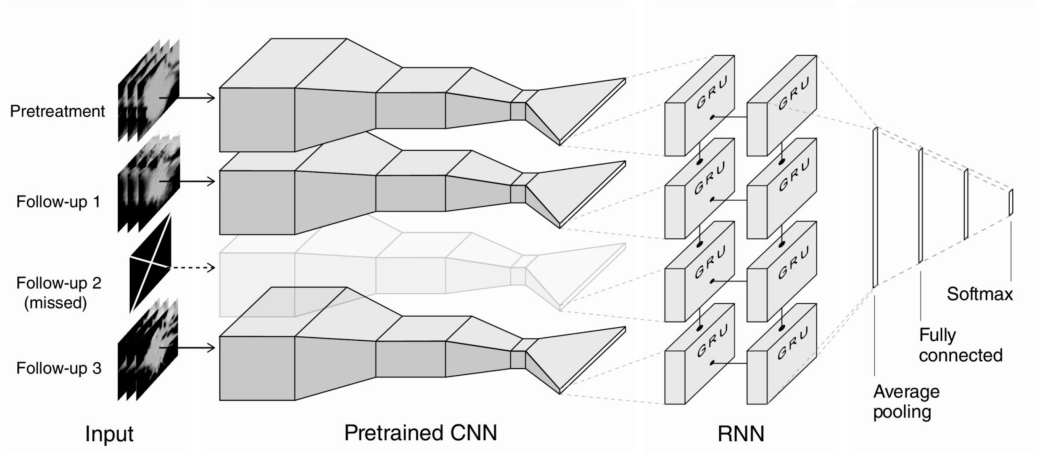
- Hehe Fan et al. - Cubic LSTMs for Video Prediction

Time	t = 12	t = 15	t = 17	t = 21	t = 25	t = 27	t = 30	Accuracy
Ground truth								PSNR
DrNet								20.3
MCnet								26.1
CubicLSTM (Ours)								**28.2**

Recurrent Networks | Examples

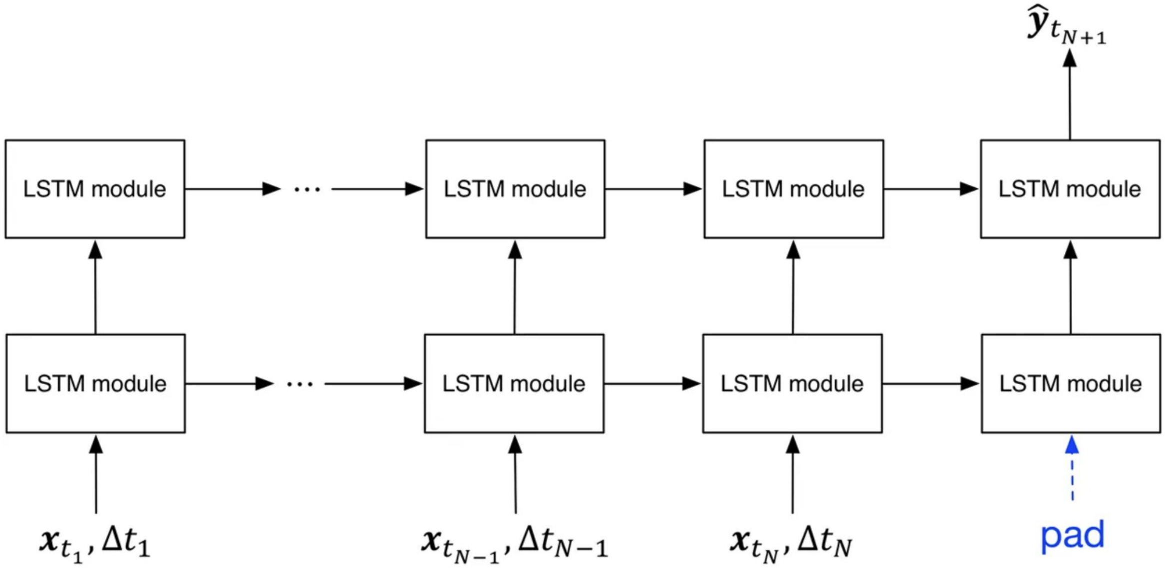
- Yiwen Xu et al. - Deep Learning Predicts Lung Cancer Treatment Response from Serial Medical Imaging





Recurrent Networks | Examples

- Tingyan Wang et al. - Predictive Modeling of the Progression of Alzheimer's Disease with Recurrent Neural Networks



The architecture of the proposed RNN model for AD stage prediction.

Recurrent Networks | Examples

- Heart Failure
  - G. Maragatham et al. - LSTM Model for Prediction of Heart Failure in Big Data
- Mortality (ICU)
  - Yao Zhu et al - Predicting ICU Mortality by Supervised Bidirectional LSTM Networks
- Other
  - Discharge Time
  - Adverse Drug Reaction
  - Kidney Failure
  - Seizure detection
- Spell checking (Language Modeling)
  - Pravallika Etoori et al. - Automatic Spelling Correction for Resource-Scarce Languages using Deep Learning

Recurrent Networks | Examples | Language Modeling

I was running yesterday.

I - was - running - yesterday.

I - w - a - s - r - u - n -n - i - n - g - y - e- s- t - e - r - d - a - y - .

Time

Recurrent Networks | Examples | Language Modeling

I was running yesterday.

yesterday.

I - was - running - ?

I - w - a - s - r - u - n - n - i - n - g - y - e- s- t - e - r - d - a - y - .

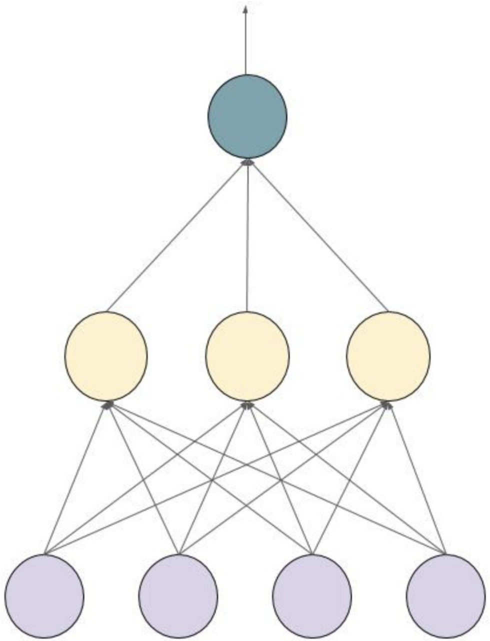
Time

Recurrent Networks | When to use them?

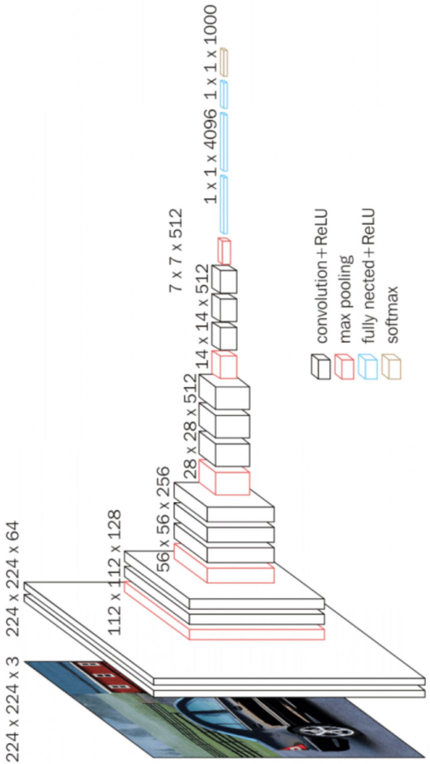
- Anything that in anyway changes over time
  - Sound
  - Text
  - Images (Video)
  - Pretty much anything related to a patient
    - Diseases
    - Symptoms
    - Measurements

In the next couple of slides RNN cell representation images are taken from: MIT 6.S191

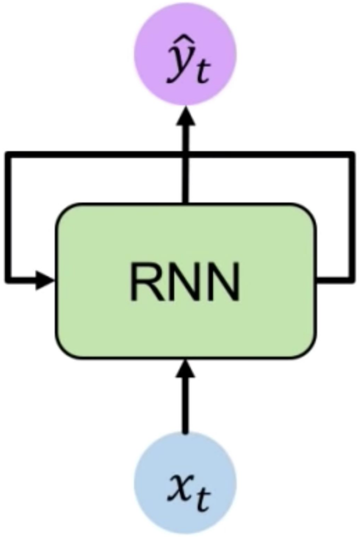
Feedforward vs CNN vs Recurrent Networks



Standard Feedforward fully connected Neural Network



VGG16 - Convolutional Network



Recurrent Neural Network - Abstract

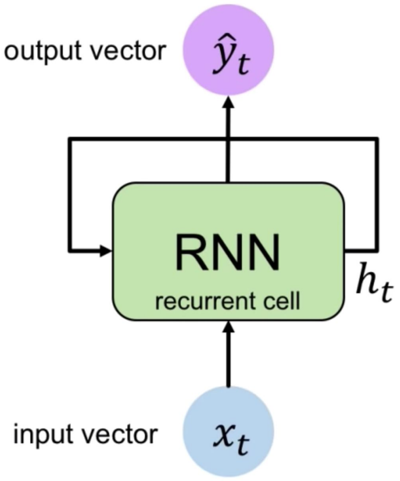
Feedforward Networks

yesterday.

I - was - running - ?

- No weight sharing
  - Each portion of the network has to learn all words
- The length of the sentence is not variable
  - We need to <PAD> the sentences
  - (Theoretically this will not be needed for RNNs but it will still be done)
- No notion of before/after

Recurrent Neural Networks



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

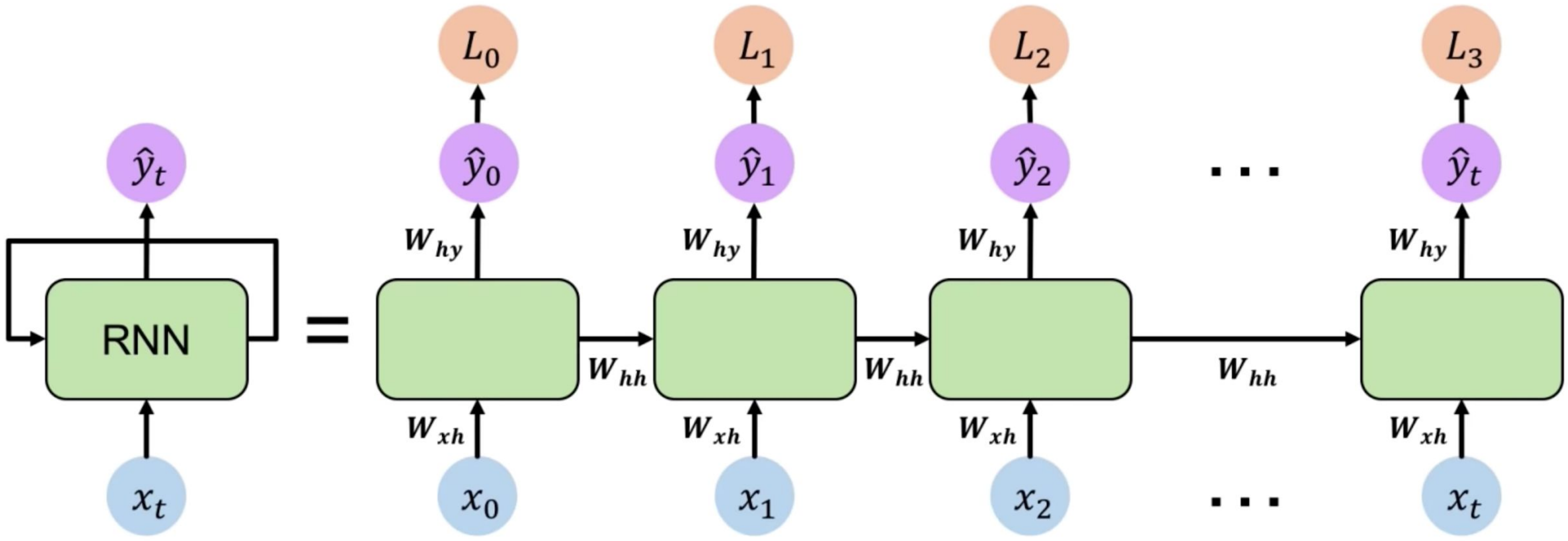
$h_t = \mathrm{sigmoid}(W_{h,h} h_{t-1} + W_{x,h} x_t)$

$\hat{y}_t = W_{h,y} h_t$

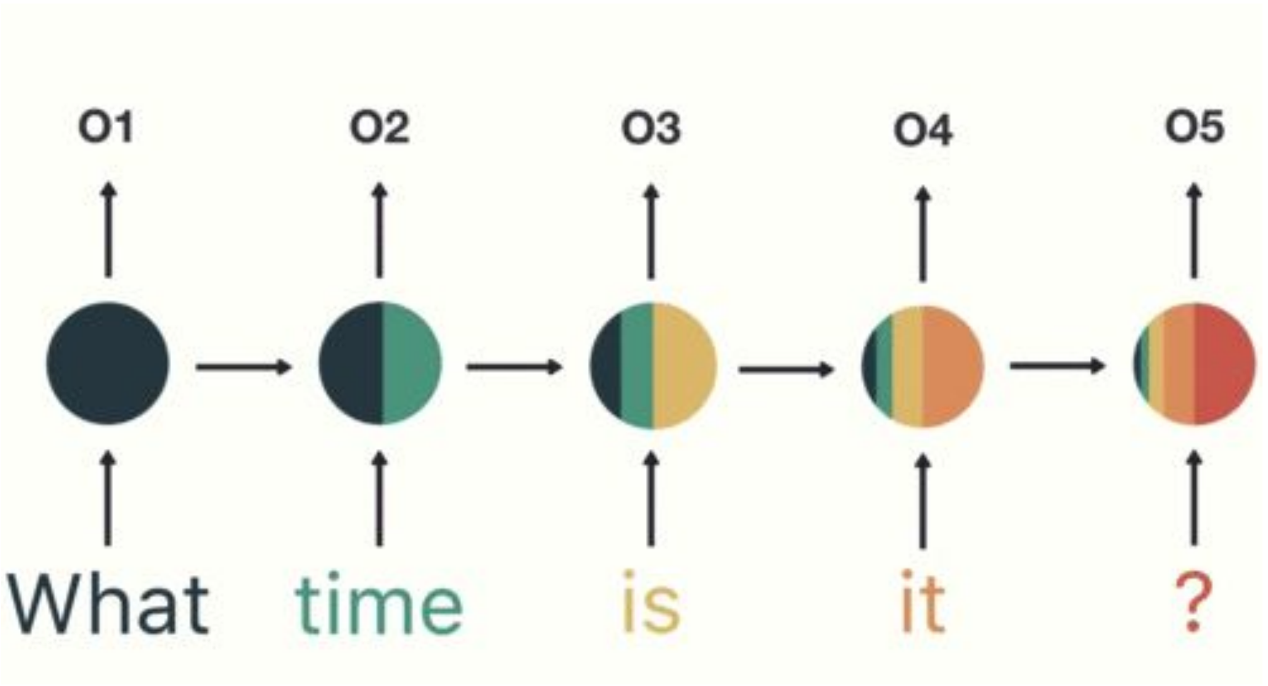
f_w - Nonlinearity (activation function)
h_t - Hidden state of the Recurrent Cell
x_t - input vector
$\hat{y}_t$ - Output Vector

Recurrent Neural Networks

- When unrolled the weights are shared

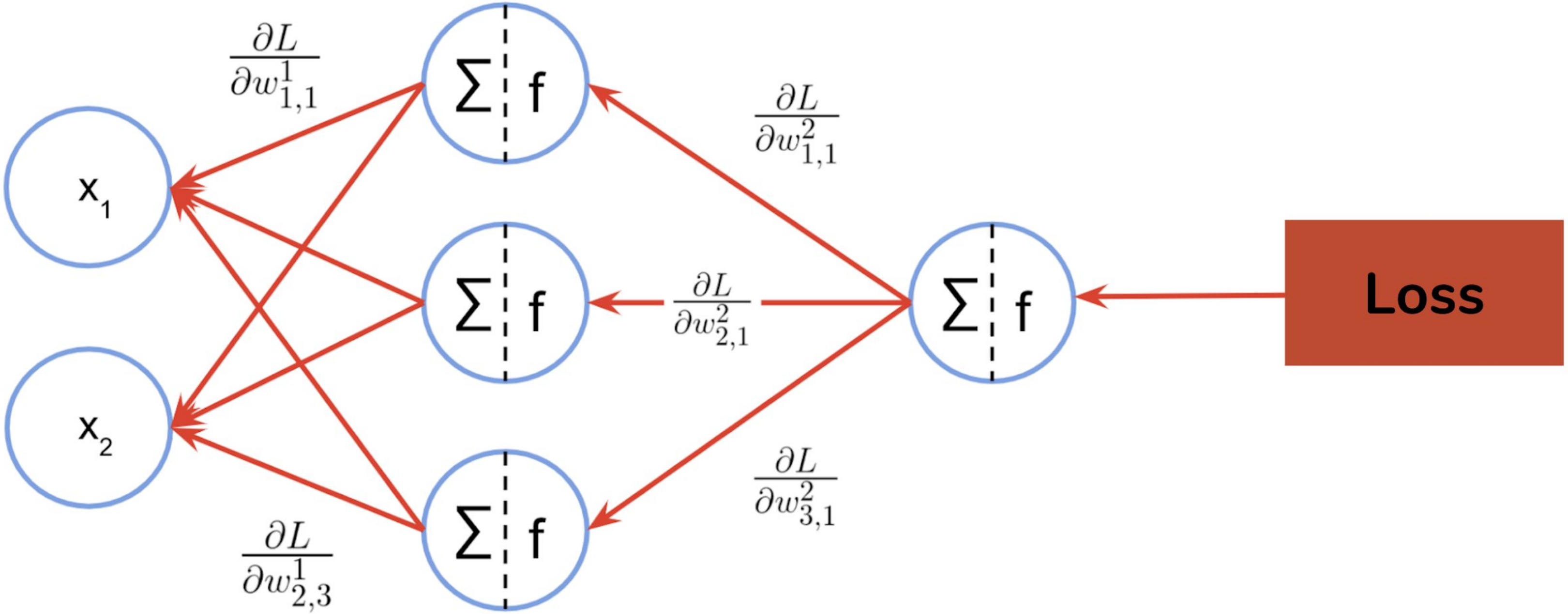


Recurrent Neural Networks



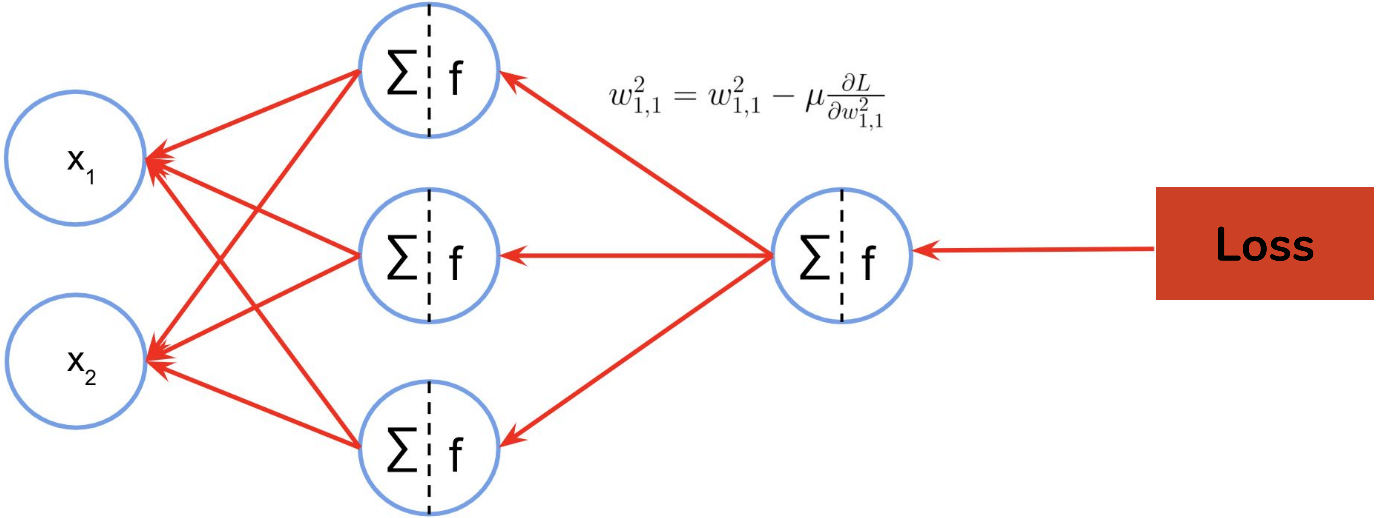
Recurrent Neural Networks | Backpropagation

- Do a forward pass then go back and calculate gradients based on loss



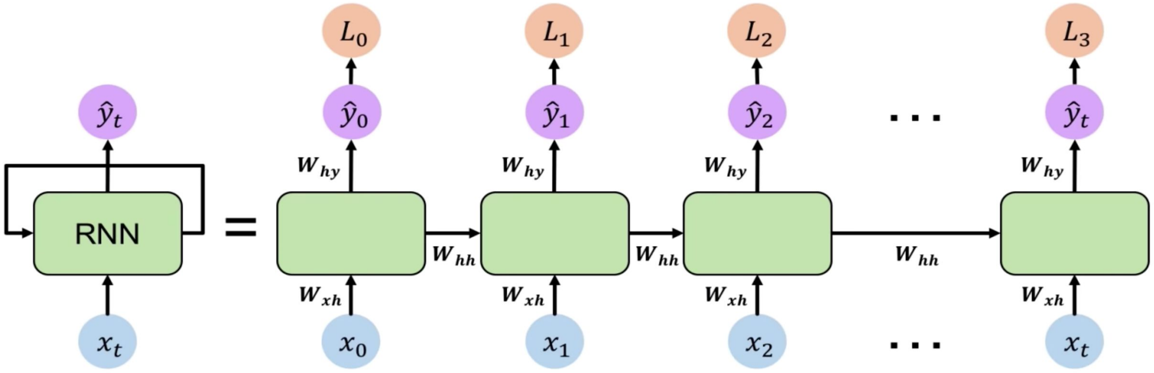
Recurrent Neural Networks | Backpropagation

- Do a forward pass then go back and calculate gradients based on loss



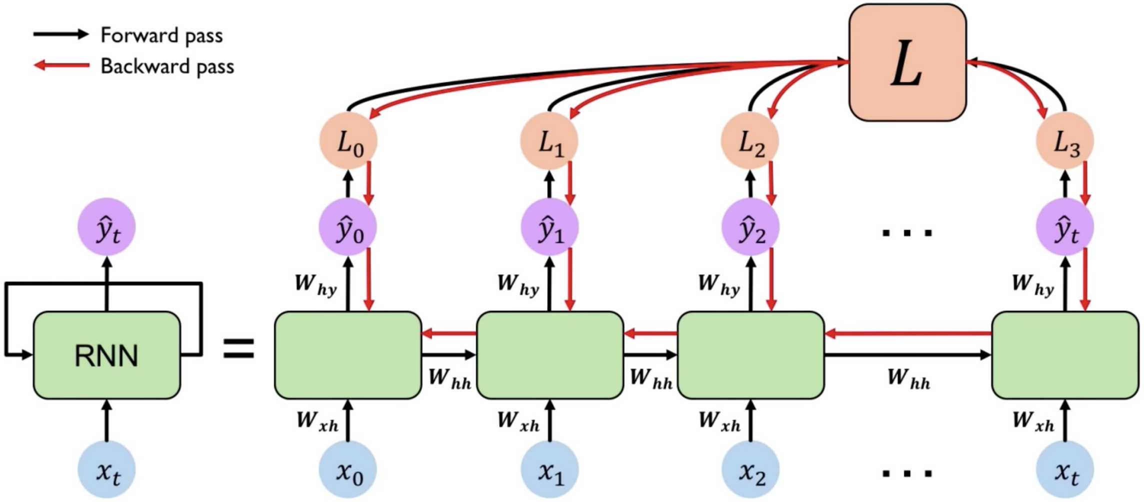
Recurrent Neural Networks | Backpropagation through time (BPTT)

- Forward pass - calculate outputs across time
- We can not backpropagate based on one single instance of the network



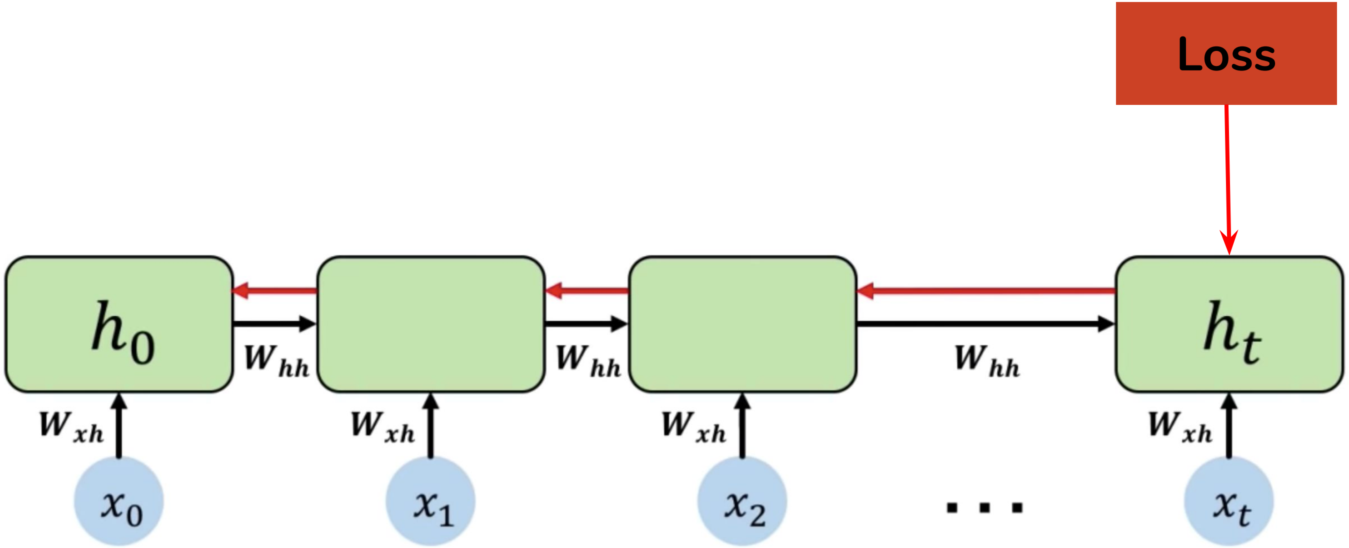
Recurrent Neural Networks | Backpropagation through time (BPTT)

- Forward pass - calculate outputs across time
- We can not backpropagate based on one single instance of the network



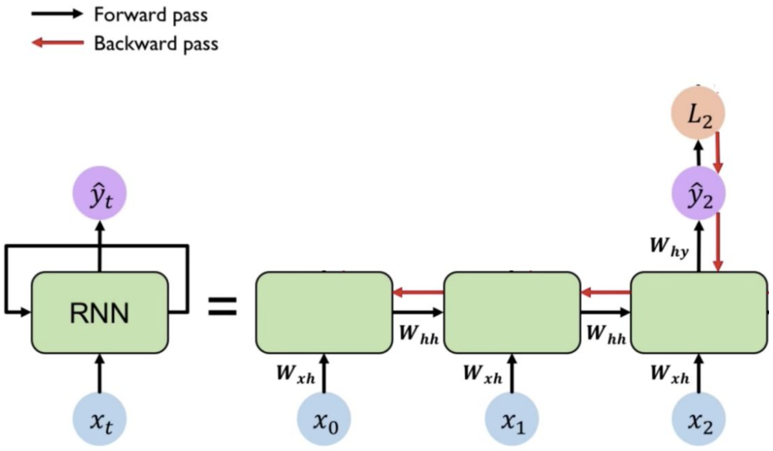
Recurrent Neural Networks | Backpropagation through time (BPTT)

- Forward pass - calculate outputs across time
- We can not backpropagate based on one single instance of the network



Recurrent Neural Networks | Backpropagation through time (BPTT)

- Calculating weight updates
- Derivation of the loss with respect to $W^{0}_{h,h}$
- Use chain rule



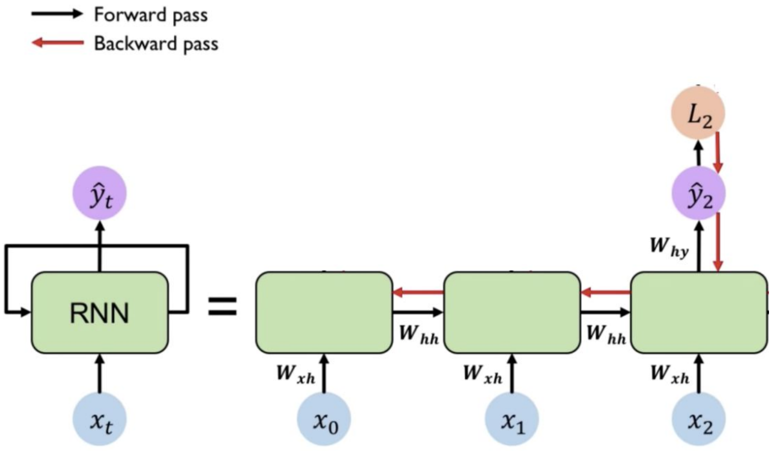
$z_t = W_{h,h}h_{t-1} + W_{x,h}x_t$

$h_t = \text{sigmoid}(z_t)$

$\hat{y}_t = W_{h,y}h_t$

Recurrent Neural Networks | Backpropagation through time (BPTT)

- Calculating weight updates
- Derivation of the loss with respect to $W^0_{h,h}$
- Use chain rule



$$\frac{\partial L_2}{\partial W^0_{h,h}} = \frac{\partial L_2}{\partial \hat{y}^2} \frac{\partial \hat{y}^2}{\partial h^2_t} \frac{\partial h^2_t}{\partial z^2_t} \frac{\partial z^2_t}{\partial h^1_t} \frac{\partial h^1_t}{\partial z^1_t} \frac{\partial z^1_t}{\partial h^0_t} \frac{\partial h^0_t}{\partial z^0_t} \frac{\partial z^0_t}{\partial W^0_{h,h}}$$

$z_t = W_{h,h} h_{t-1} + W_{x,h} x_t$

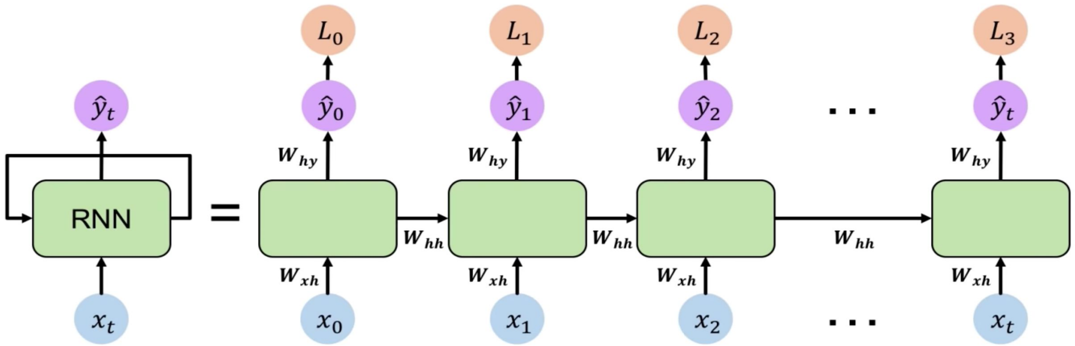
$h_t = \text{sigmoid}(z_t)$

$\hat{y}_t = W_{h,y} h_t$

Recurrent Neural Networks | Backpropagation through time (BPTT)

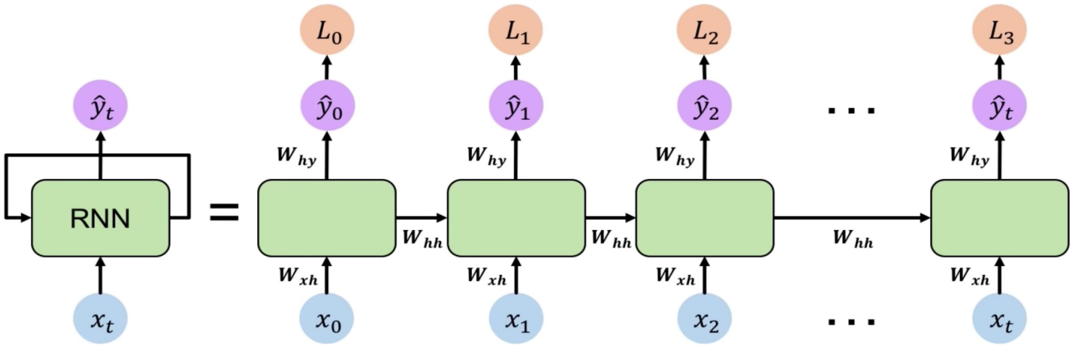
- Exploding Gradient
  - Many values > 1
  - Gradient clipping
- Vanishing Gradient
  - Many Values < 1
  - Not so easy to solve

$$\frac{\partial L_2}{\partial W_{h,h}^0} = \frac{\partial L_2}{\partial \hat{y}^2} \frac{\partial \hat{y}^2}{\partial h_t^2} \frac{\partial h_t^2}{\partial z_t^2} \frac{\partial z_t^2}{\partial h_t^1} \frac{\partial h_t^1}{\partial z_t^1} \frac{\partial z_t^1}{\partial h_t^0} \frac{\partial h_t^0}{\partial z_t^0} \frac{\partial z_t^0}{\partial W_{h,h}^0}$$



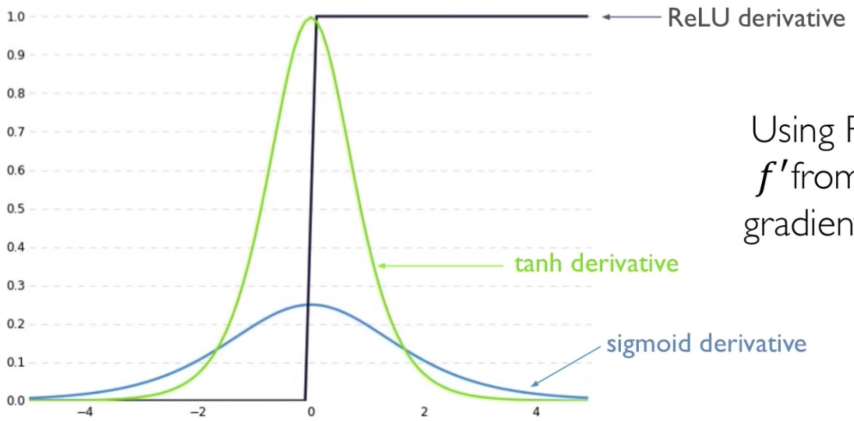
Recurrent Neural Networks | Text Classification (Colab)

- The last output contains the information from the whole input sequence
- Sentence/Text Classification
  - Ignore all outputs apart from the last one
  - Put a FC network on top of the last output
  - Do classification



Recurrent Neural Networks | Vanishing Gradient

- If a lot of the weights are < 1
  - After only a couple of steps back the gradient is 0
  - Biases the network to detect short term dependencies
  - Long dependencies are completely ignored
- Tricks
  - Use a ReLU activation function instead of Sigmoid
  - Initialize with identity matrix
  - Use gated cells

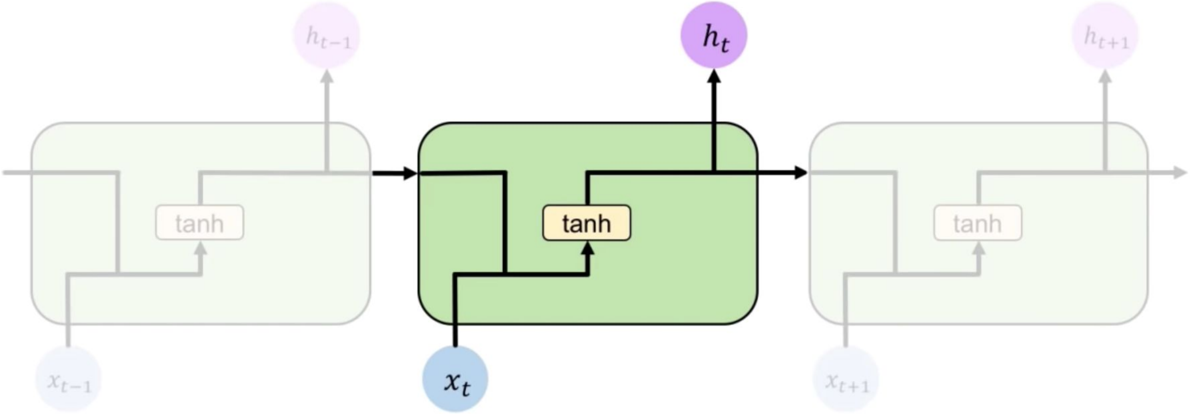


Using ReLU prevents f' from shrinking the gradients when $x > 0$

Recurrent Neural Networks | Vanishing Gradient

- Long Short Term Memory (LSTM)

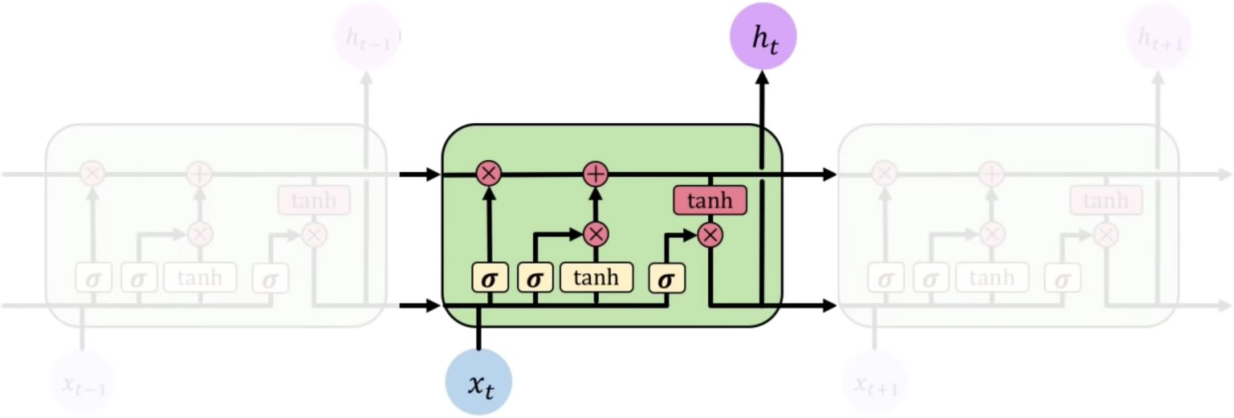
Standard RNN



Recurrent Neural Networks | LSTMs

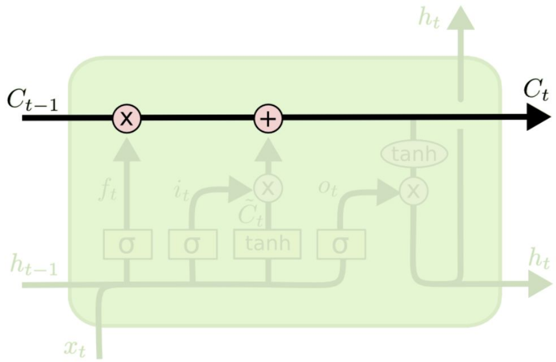
- Long Short Term Memory (LSTM)
  - Use gates to not forget
  - A bit more messy

LSTM



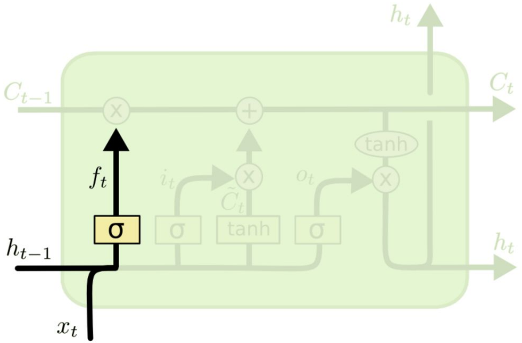
Recurrent Neural Networks | LSTMs

- Cell state
  - The memory of the cell
  - What it knows until now
  - The gates allow information to go into the cell state



Recurrent Neural Networks | LSTMs

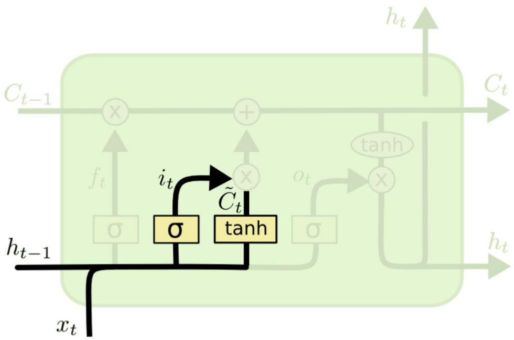
- Forget gate
  - What information should be forgotten and what kept
  - Receives
    - Old output
    - New input



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

Recurrent Neural Networks | LSTMs

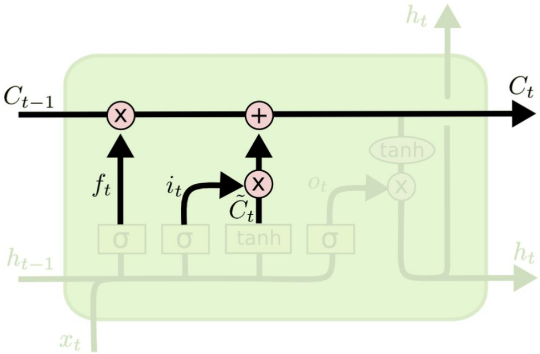
- Update gate
  - What new information to add
  - Receives
    - Old output
    - New input
- Creates a vector of new candidate values



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

Recurrent Neural Networks | LSTMs

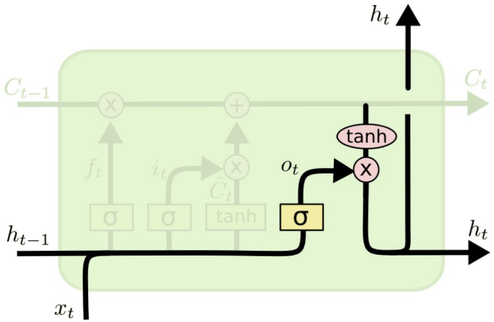
- Updating the cell state
  - Multiply old state by the forget gate
  - Multiply the new candidates by the update gate
  - Sum everything



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

Recurrent Neural Networks | LSTMs

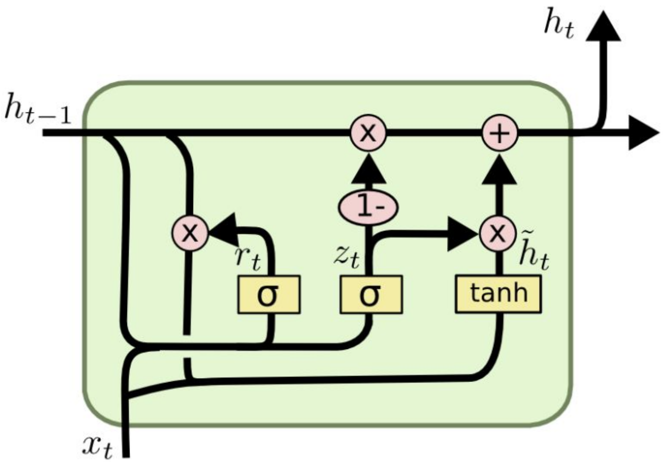
- Calculating the output
  - Cell state
  - Output gate
- Doesn’t output the whole cell state



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

Recurrent Neural Networks | GRU

- Gated Recurrent Unit
  - Combines update and forget gate
  - Simpler model



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

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

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

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

What we have done

- Differences between FC, CNN, RNN
- How to train an RNN
  - Exploding gradients
  - Vanishing gradients
- What can RNNs be used for
  - Any sequential data
- Different cell types
  - LSTM
  - GRU

Next Time

- Language Modeling
  - New dataset
  - Playing with different text generators
- Multilayer RNNs
  - Stacking RNNs
- Advanced dropout
  - Temporal dropout
  - Inter-layer dropout
- Transfer learning
  - From LM to text classification