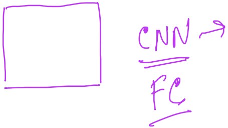
Introduction to Deep Learning

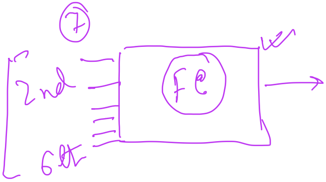
Arijit Mondal
Dept. of Computer Science & Engineering
Indian Institute of Technology Patna
arijit@iitp.ac.in



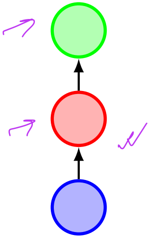
Recurrent Neural Network

Introduction

- Recurrent neural networks are used for processing sequential data in general
  - Convolution neural network is specialized for image
- Capable of processing variable length input
- Shares parameters across different part of the model
  - Example: "I went to IIT in 2017" or "In 2017, I went to IIT"
  - Example: "I grew up in Bengal. .... I can speak fluent ____"
  - For traditional machine learning models require to learn rules for different positions

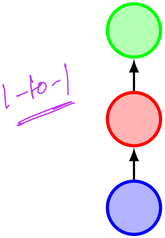


Types of applications



Feed-forward network

Types of applications



Feed-forward network

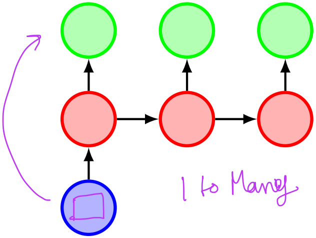
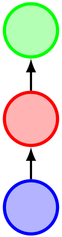


Image captioning

Types of applications



Feed-forward network

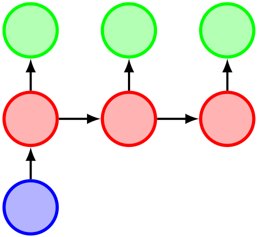
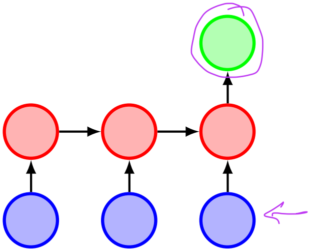
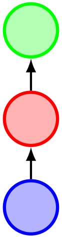


Image captioning



Sentiment analysis

Types of applications



Feed-forward network

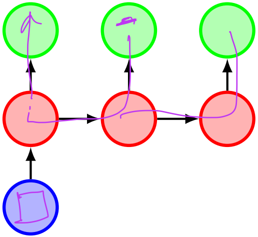
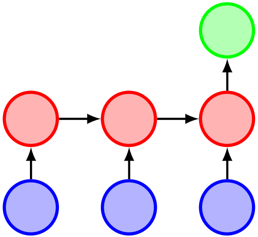
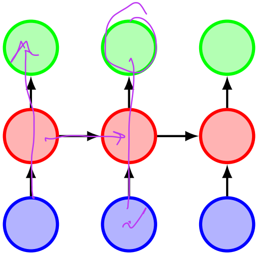


Image captioning

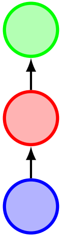


Sentiment analysis



Video frame labeling

Types of applications



Feed-forward network

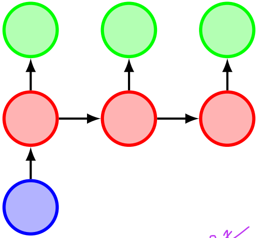
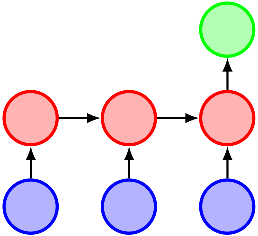
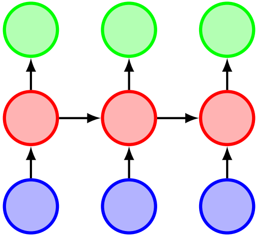


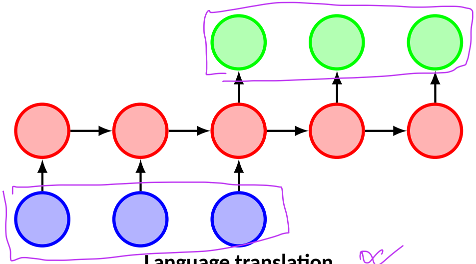
Image captioning



Sentiment analysis



Video frame labeling

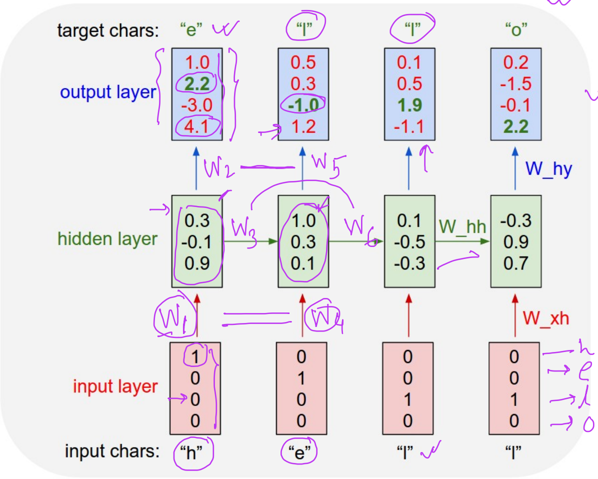


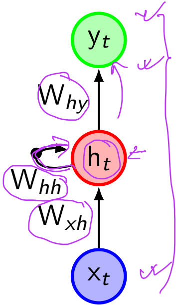
Language translation

Example

1-hot encoding

hello h, e, l, o



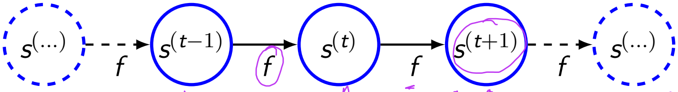


4 times

Image source: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

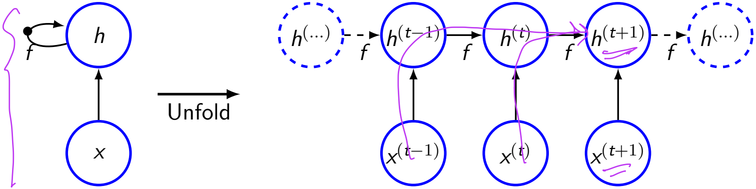
Computational graph

- Formal way to represent the computation
- Unfolding the graph results in sharing of parameters
- Consider a system $\mathbf{s}^{(t)} = f(\mathbf{s}^{(t-1)}, \boldsymbol{\theta})$ where $\mathbf{s}^{(t)}$ denotes the state of the system
  - It is recurrent
  - For finite number of steps, it can be unfolded
  - Example: $s^{(3)} = f(s^{(2)}, \theta) = f(f(s^{(1)}, \theta), \theta)$



System with inputs

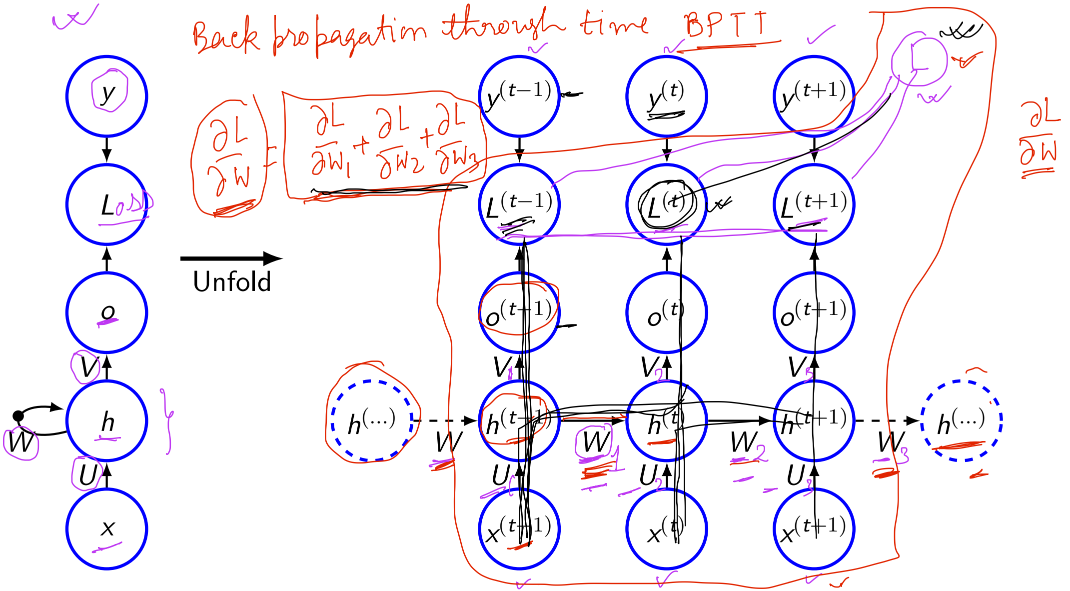
- A system will be represented as $s^{(t)} = f(s^{(t-1)}, x^{(t)}, \boldsymbol{\theta})$
  - A state contains information of whole past sequence
- Usually state is indicated as hidden units such that $h^{(t)} = f(h^{(t-1)}, x^{(t)}, \boldsymbol{\theta})$
- While predicting, network learn $h^{(t)}$ as a kind of lossy summary of past sequence upto t
  - $h^{(t)}$ depends on $(x^{(t)}, x^{(t-1)}, \ldots, x^{(1)})$



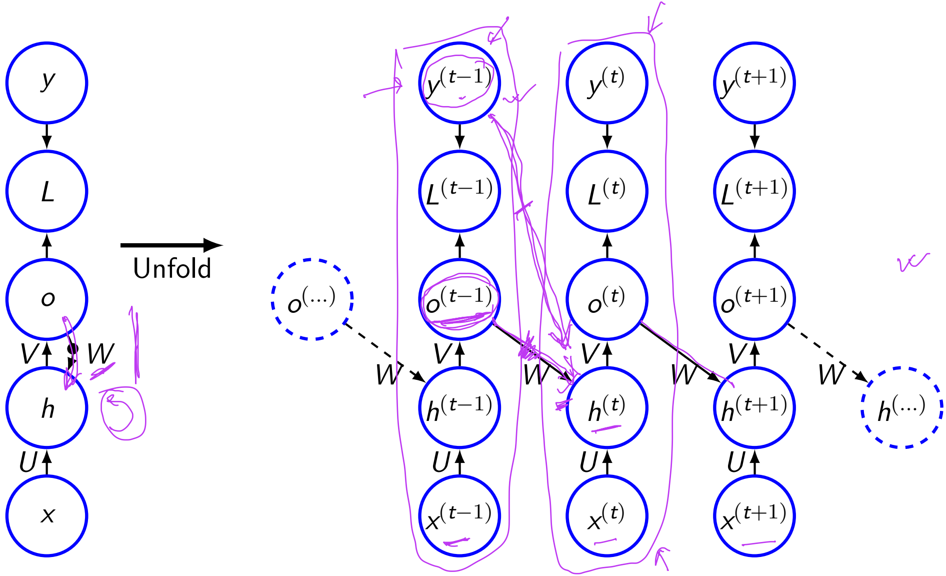
System with inputs (contd.)

- Unfolded recursion after t steps will be $h^{(t)} = g^{(t)}(x^{(t)}, x^{(t-1)}, \ldots, x^{(1)}) = f(h^{(t-1)}, x^{(t)}, \boldsymbol{\theta})$
- Unfolding process has some advantages
  - Regardless of sequence length, learned model has same input size
  - Uses the same transition function f with the same parameters at every time steps
- Can be trained with fewer examples
- Recurrent graph is succinct
- Unfolded graph illustrates the information flow

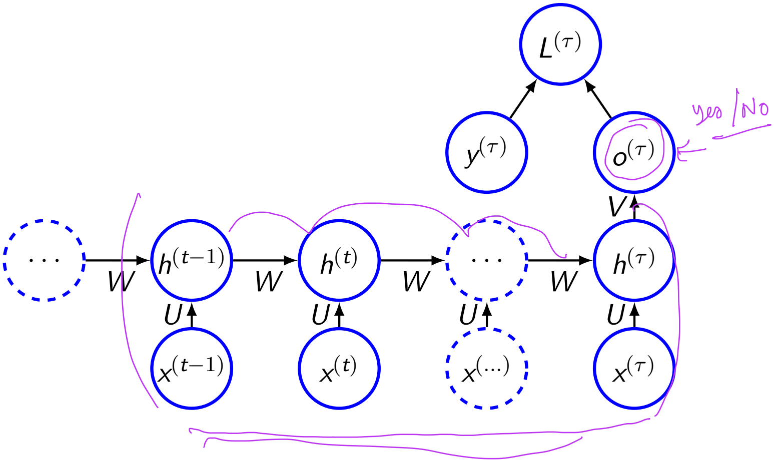
Recurrent connection in hidden units



Output to hidden unit connection

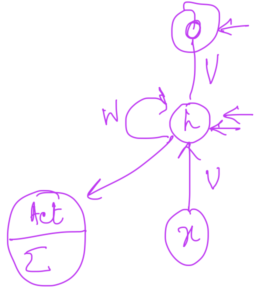


Sequence processing



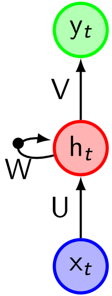
Recurrent neural network

- Function computable by a Turing machine can be computed by such recurrent network of finite size
- $\tanh$ is usually chosen as activation function for hidden units
- Output can be considered as discrete, so o gives unnormalized log probabilities
- Forward propagation begins with initial state h^0
- So we have,
  - $\mathsf{a}^{(t)} = \mathsf{b} + \mathsf{W}\mathsf{h}^{(t-1)} + \mathsf{U}\mathsf{x}^{(t)}$
  - $\mathsf{h}^{(t)} = \tanh(\mathsf{a}^{(t)})$
  - $\mathsf{o}^{(t)} = \mathsf{c} + \mathsf{V}\mathsf{h}^{(t)}$
  - $\hat{\mathsf{y}}^{(t)} = \mathrm{softmax}(\mathsf{o}^{(t)})$
- Input and output have the same length



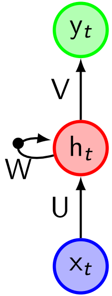
Backpropagation through time

- The network will be unfolded and gradient will be back propagated
- Number of stages need to be decided
- Issue in gradient computation
  - Vanishing gradients
  - Exploding gradients



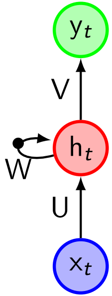
Backpropagation through time

- The network will be unfolded and gradient will be back propagated
- Number of stages need to be decided
- Issue in gradient computation
  - Vanishing gradients
  - Exploding gradients
- Loss function
  - $E_t = \frac{1}{2}\sum_{k=1}^{\text{out}}(\hat{y}_k - y_k)^2,$



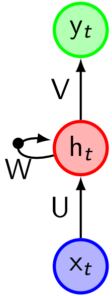
Backpropagation through time

- The network will be unfolded and gradient will be back propagated
- Number of stages need to be decided
- Issue in gradient computation
  - Vanishing gradients
  - Exploding gradients
- Loss function
  - $E_t = \frac{1}{2}\sum_{k=1}^{\text{out}}(\hat{y}_k - y_k)^2$, $E = \frac{1}{2}\sum_{t=1}^{\tau}\sum_{k=1}^{\text{out}}(\hat{y}_{tk} - y_{tk})^2$



Backpropagation through time

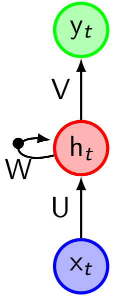
- The network will be unfolded and gradient will be back propagated
- Number of stages need to be decided
- Issue in gradient computation
  - Vanishing gradients
  - Exploding gradients
- Loss function
  - $E_t = \frac{1}{2}\sum_{k=1}^{\text{out}}(\hat{y}_k - y_k)^2$, $E = \frac{1}{2}\sum_{t=1}^{\tau}\sum_{k=1}^{\text{out}}(\hat{y}_{tk} - y_{tk})^2$
  - $E = -\sum_{t=1}^{\tau}\sum_{k=1}^{\text{out}}[\hat{y}_{tk}\ln y_{tk} + (1-\hat{y}_{tk})\ln(1-y_{tk})]$



Backpropagation through time

- Basic equations

$$
\begin{aligned}
h_t &= Ux_t + W\phi(h_{t-1}) \\
y_t &= V\phi(h_t)
\end{aligned}
$$



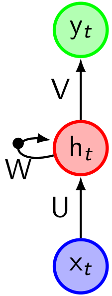
Backpropagation through time

- Basic equations

$$\begin{aligned} h_t &= Ux_t + W\phi(h_{t-1}) \\ y_t &= V\phi(h_t) \end{aligned}$$

- Gradient

$$\frac{\partial E}{\partial W}$$



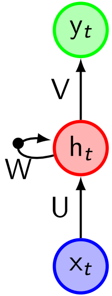
Backpropagation through time

- Basic equations

$$\begin{aligned} h_t &= Ux_t + W\phi(h_{t-1}) \\ y_t &= V\phi(h_t) \end{aligned}$$

- Gradient

$$\frac{\partial E}{\partial W} = \sum_{t=1}^{\tau} \frac{\partial E_t}{\partial W}$$



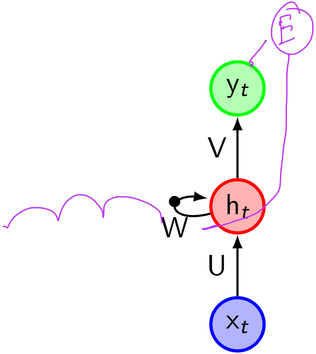
Backpropagation through time

- Basic equations

$$h_t = Ux_t + W\phi(h_{t-1})$$
$$y_t = V\phi(h_t)$$

- Gradient

$$\frac{\partial E}{\partial W} = \sum_{t=1}^{\tau} \frac{\partial E_t}{\partial W} = \sum_{t=1}^{\tau}\sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t}$$



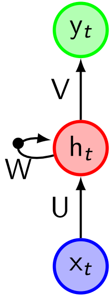
Backpropagation through time

- Basic equations

$$\begin{aligned} h_t &= Ux_t + W\phi(h_{t-1}) \\ y_t &= V\phi(h_t) \end{aligned}$$

- Gradient

$$\frac{\partial E}{\partial W} = \sum_{t=1}^{\tau} \frac{\partial E_t}{\partial W} = \sum_{t=1}^{\tau} \sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t}$$



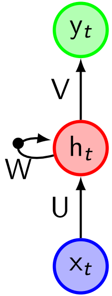
Backpropagation through time

- Basic equations

$$h_t = Ux_t + W\phi(h_{t-1})$$
$$y_t = V\phi(h_t)$$

- Gradient

$$\frac{\partial E}{\partial W} = \sum_{t=1}^{\tau} \frac{\partial E_t}{\partial W} = \sum_{t=1}^{\tau} \sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial h_k}$$



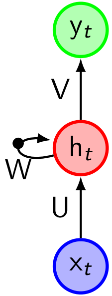
Backpropagation through time

- Basic equations

$$h_t = Ux_t + W\phi(h_{t-1})$$
$$y_t = V\phi(h_t)$$

- Gradient

$$\frac{\partial E}{\partial W} = \sum_{t=1}^{\tau} \frac{\partial E_t}{\partial W} = \sum_{t=1}^{\tau}\sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t}\frac{\partial y_t}{\partial h_t}\frac{\partial h_t}{\partial h_k}\frac{\partial h_k}{\partial W}$$



Backpropagation through time

- Basic equations

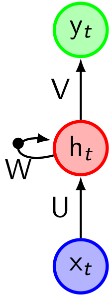
$$\begin{aligned} \mathsf{h}_t &= \mathsf{U}\mathsf{x}_t + \mathsf{W}\phi(\mathsf{h}_{t-1}) \\ \mathsf{y}_t &= \mathsf{V}\phi(\mathsf{h}_t) \end{aligned}$$

- Gradient

$$\frac{\partial E}{\partial \mathsf{W}} = \sum_{t=1}^{\tau} \frac{\partial E_t}{\partial \mathsf{W}} = \sum_{t=1}^{\tau} \sum_{k=1}^{t} \frac{\partial E_t}{\partial \mathsf{y}_t} \frac{\partial \mathsf{y}_t}{\partial \mathsf{h}_t} \frac{\partial \mathsf{h}_t}{\partial \mathsf{h}_k} \frac{\partial \mathsf{h}_k}{\partial \mathsf{W}}$$

- Now we have,

$$\frac{\partial \mathsf{h}_t}{\partial \mathsf{h}_k}$$



Backpropagation through time

- Basic equations

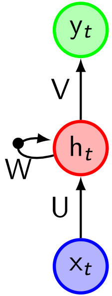
$$h_t = Ux_t + W\phi(h_{t-1})$$
$$y_t = V\phi(h_t)$$

- Gradient

$$\frac{\partial E}{\partial W} = \sum_{t=1}^{\tau} \frac{\partial E_t}{\partial W} = \sum_{t=1}^{\tau} \sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W}$$

- Now we have,

$$\frac{\partial h_t}{\partial h_k} = \prod_{i=k+1}^{t} \frac{\partial h_i}{\partial h_{i-1}}$$



Backpropagation through time

- Basic equations

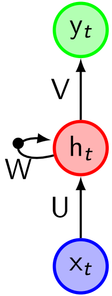
$$h_t = Ux_t + W\phi(h_{t-1})$$
$$y_t = V\phi(h_t)$$

- Gradient

$$\frac{\partial E}{\partial W} = \sum_{t=1}^{\tau} \frac{\partial E_t}{\partial W} = \sum_{t=1}^{\tau}\sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t}\frac{\partial y_t}{\partial h_t}\frac{\partial h_t}{\partial h_k}\frac{\partial h_k}{\partial W}$$

- Now we have,

$$\frac{\partial h_t}{\partial h_k} = \prod_{i=k+1}^{t} \frac{\partial h_i}{\partial h_{i-1}} = \prod_{i=k+1}^{t} W^T \mathrm{diag}[\phi'(h_{i-1})]$$



Backpropagation through time

- Issues in gradient

$$\left\| \frac{\partial h_i}{\partial h_{i-1}} \right\| \leq \|W^T\| \, \|\text{diag}[\phi'(h_{i-1})]\| \leq \lambda_W \lambda_\phi$$

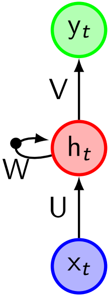
$$\left\| \frac{\partial h_t}{\partial h_k} \right\| \leq (\lambda_W \lambda_\phi)^{t-k}$$

$\lambda_W \lambda_\phi > 1 \rightarrow$ clip

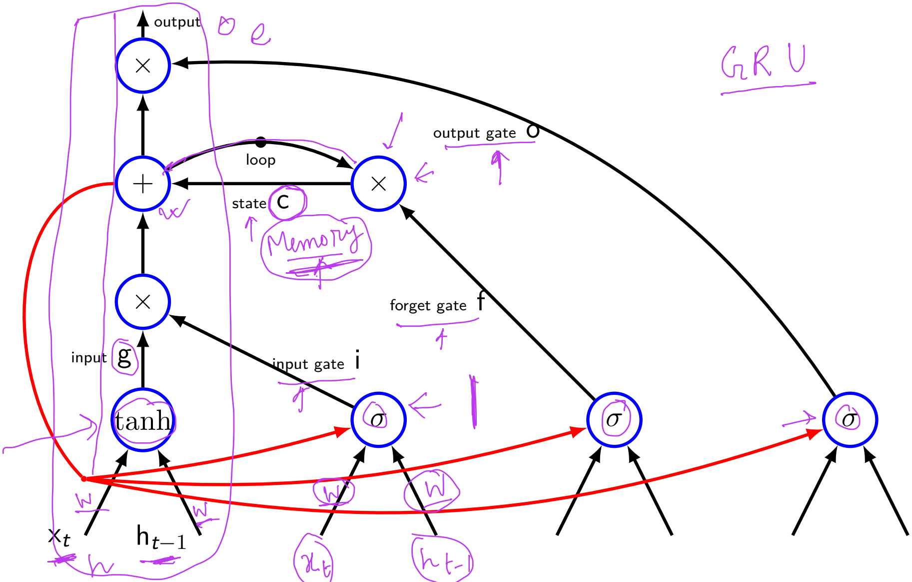
$\lambda_W \lambda_\phi < 1$

$\frac{\partial E}{\partial W} \rightarrow 1000$

$\frac{\partial E}{\partial W} = 1$



LSTM


output
loop
state c
output gate o
forget gate f
input g
input gate i
tanh
σ
σ
σ
x_t
h_{t-1}
Memory
GRU

LSTM

- Mathematical relation

$$i_t = \sigma(\boldsymbol{\theta}_{xi} x_t + \boldsymbol{\theta}_{hi} h_{t-1} + b_i)$$

$$f_t = \sigma(\boldsymbol{\theta}_{xf} x_t + \boldsymbol{\theta}_{hf} h_{t-1} + b_f)$$

$$o_t = \sigma(\boldsymbol{\theta}_{xo} x_t + \boldsymbol{\theta}_{ho} h_{t-1} + b_o)$$

$$g_t = \tanh(\boldsymbol{\theta}_{xg} x_t + \boldsymbol{\theta}_{hg} h_{t-1} + b_g)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$

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

LSTM

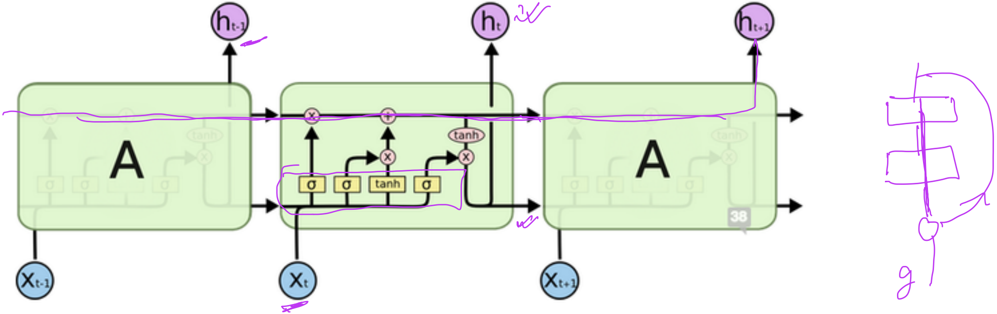


Image source:colah.github.io