# Deep Learning 3

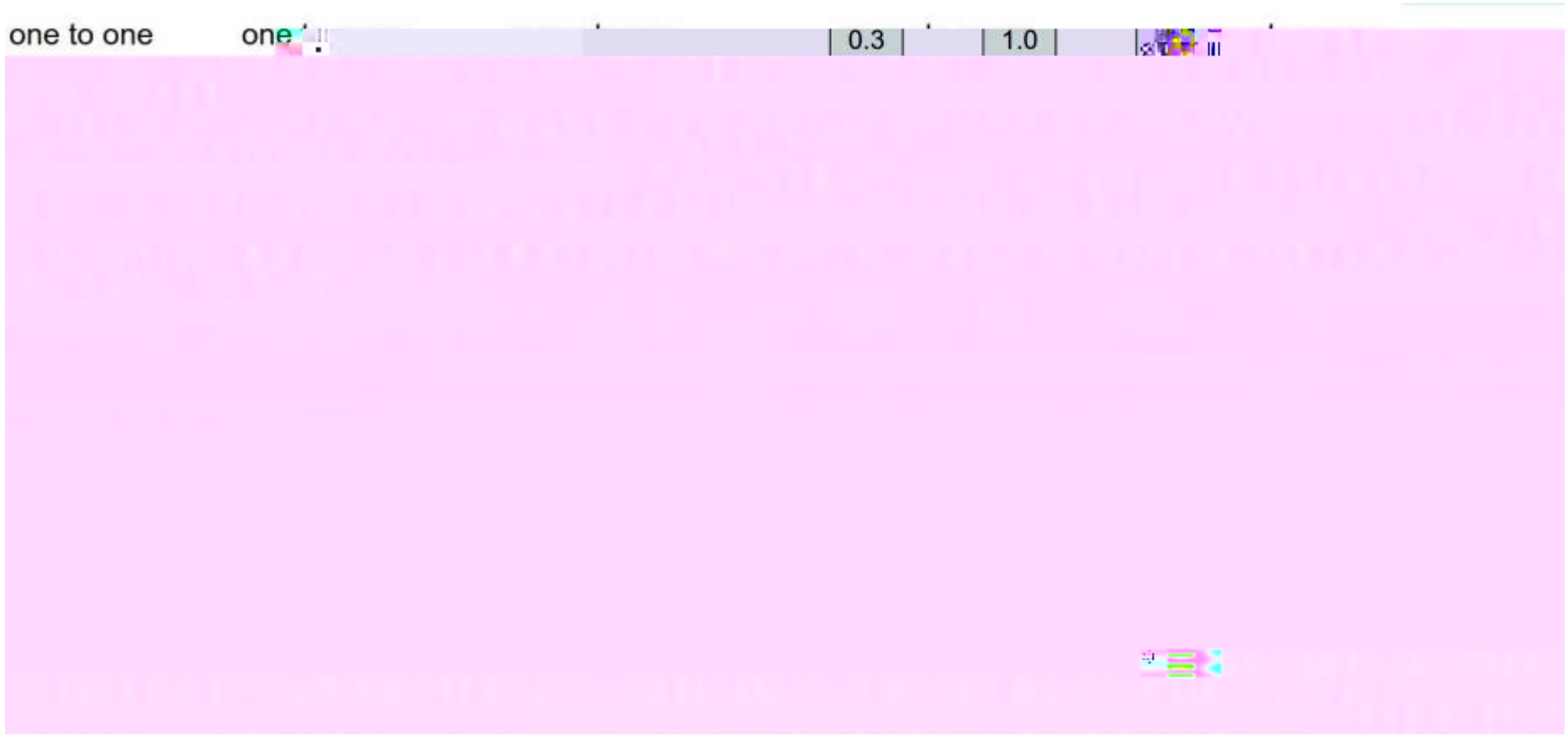
Jian Li

IIIS, Tsinghua

# Recurrent Neural Networks (RNN)

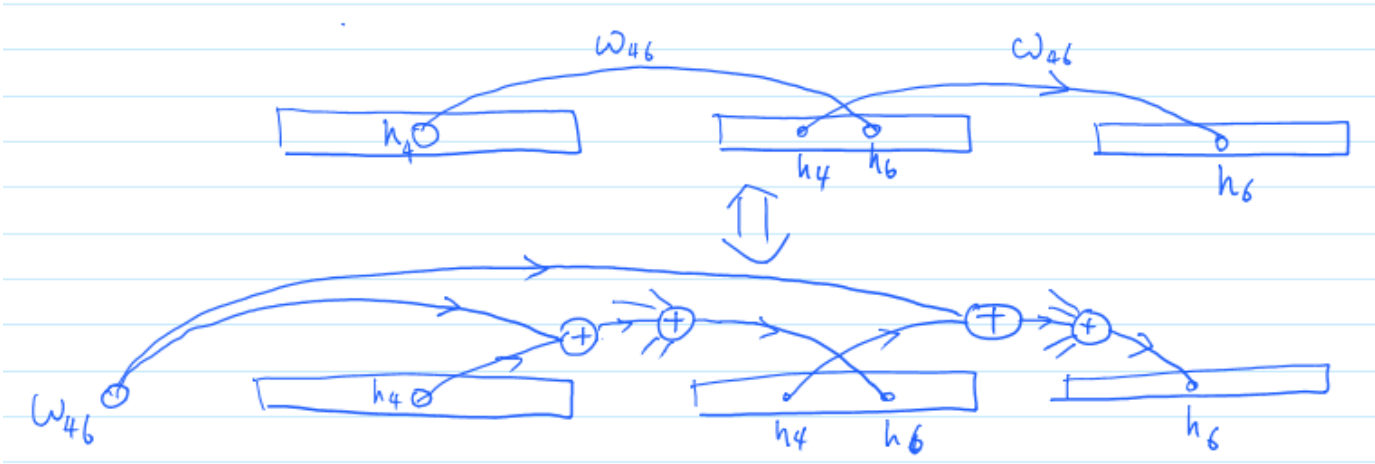
CNN: parameter sharing in space
RNN: parameter sharing in time (suitable for sequences, in particular sequences with variable lengths)

# Basics



# BPTT

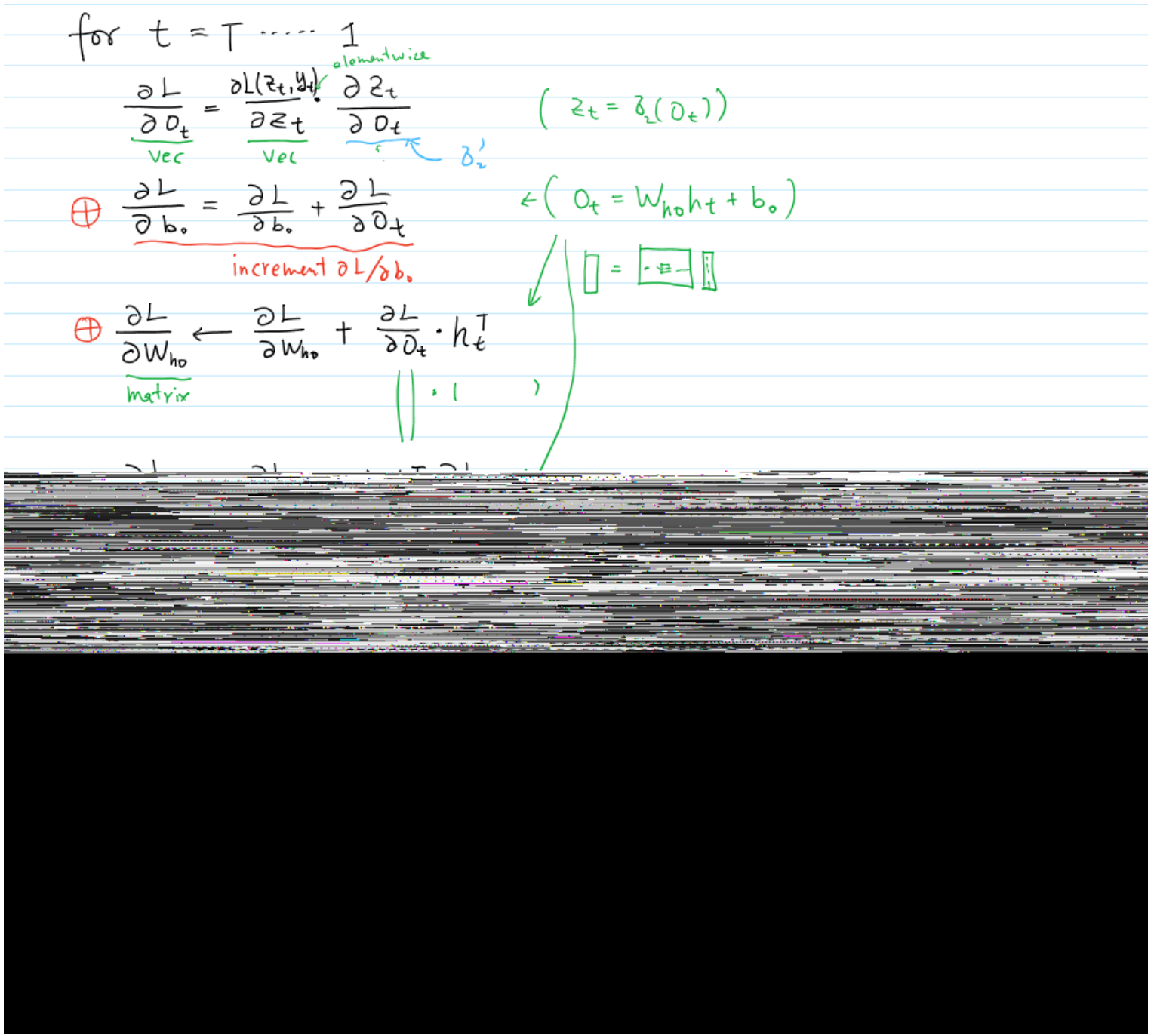
- Backprop thru time t=T, T-1 U Y2,1X U
- The weight variables w are shares across all time steps.
- So in backprop, they need to be imcremented when the grad flows back each time step



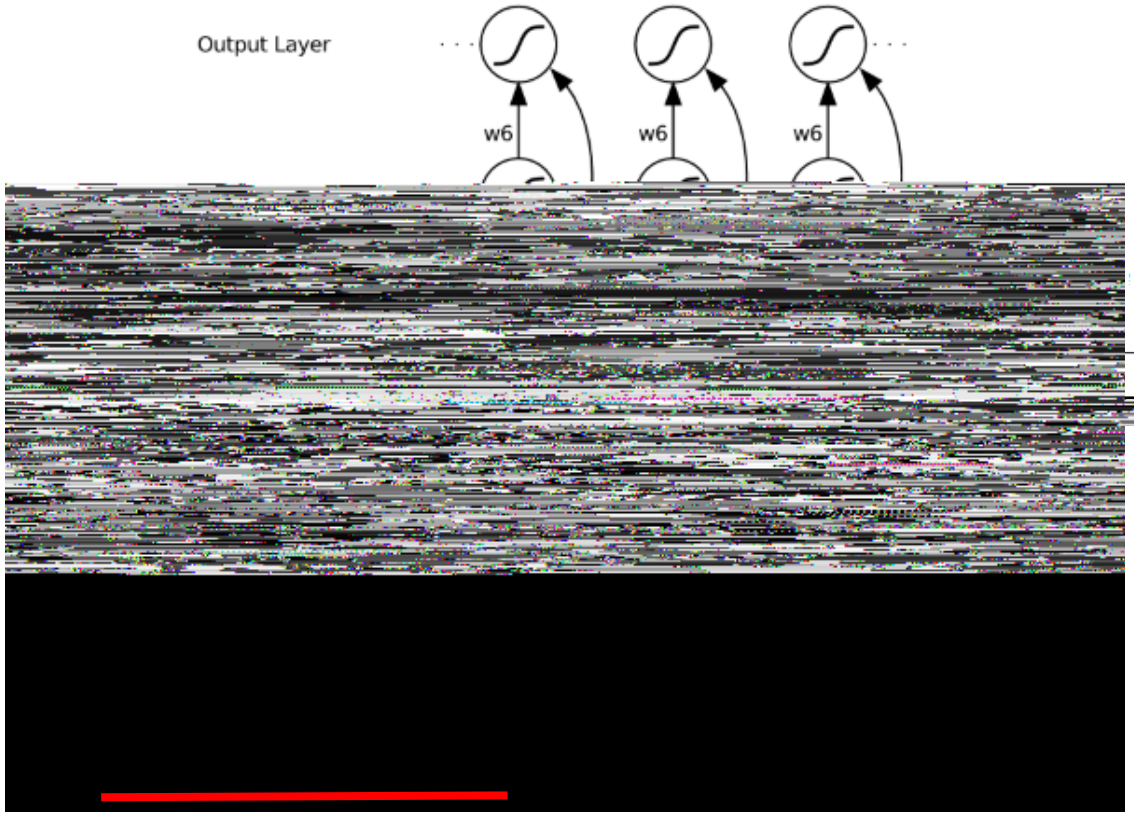
# BPTT

‡Backprop thru time

t=T,T- í U Y X U î U í



# Bi-directional RNN




**for** $t = 1$ to $T$ **do**
Forward pass for the forward hidden layer, storing activations at each timestep
**for** $t = T$ to 1 **do**
Forward pass for the backward hidden layer, storing activations at each timestep
**for** all $t$, in any order **do**
hidden layers

**Algorithm 3.1:** BRNN Forward Pass

**for** all $t$, in any order **do**
Backward pass for the output layer, storing $\delta$ terms at each timestep
**for** $t = T$ to 1 **do**
BPTT backward pass for the forward hidden layer, using the stored $\delta$ terms from the output layer
**for** $t = 1$ to $T$ **do**
BPTT backward pass for the backward hidden layer, using the stored $\delta$ terms from the output layer

**Algorithm 3.2:** BRNN Backward Pass

Figure from Graves. Supervised Sequence Labelling with Recurrent Neural Networks

# Gradient Vanishing/Exploding problem

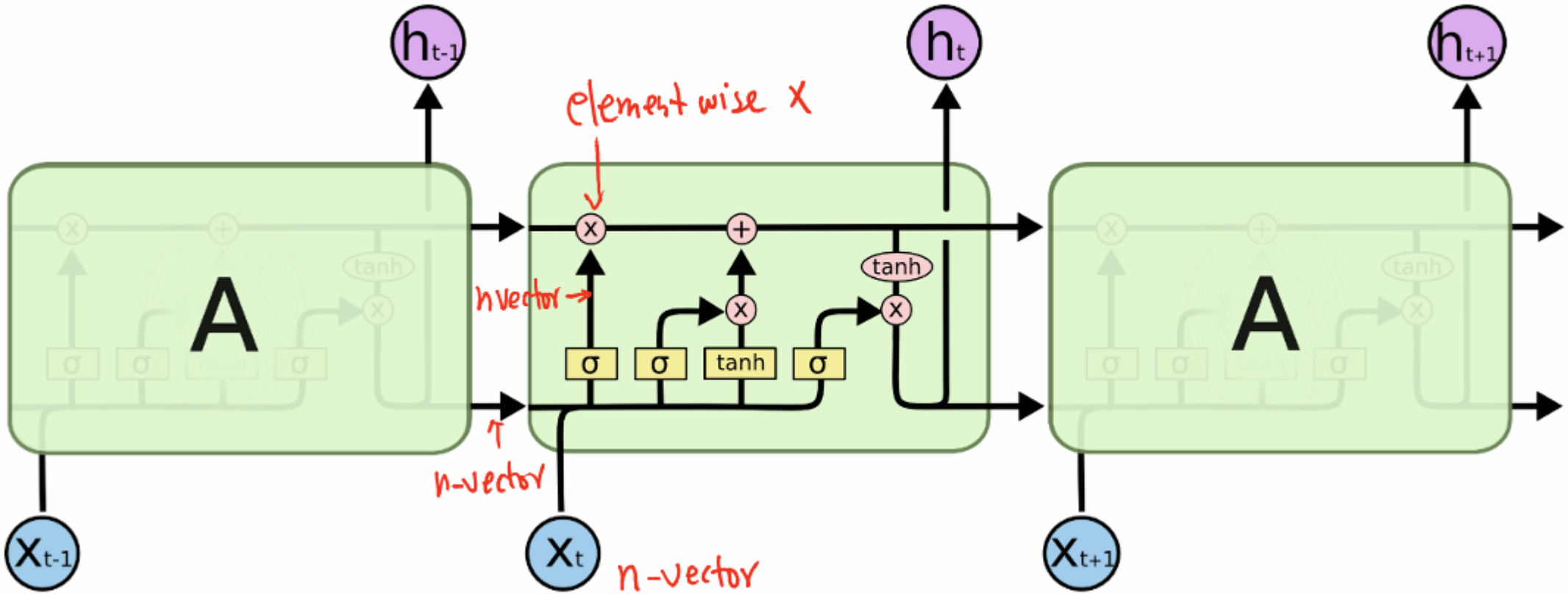
# Gradient Vanishing/Exploding problem

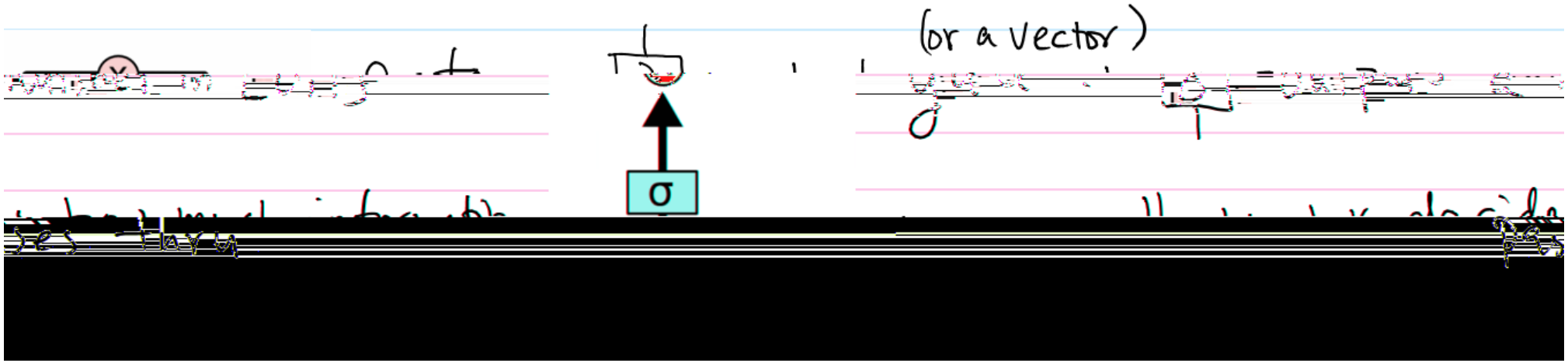
- Analyzing the norms of the Jacobians, yields:

$$\left\| \frac{\partial h_j}{\partial h_{j-1}} \right\| \leq \|W^T\| \|\text{diag}[f'(h_{j-1})]\| < \beta$$

# Long Short Term Memory (LSTM)

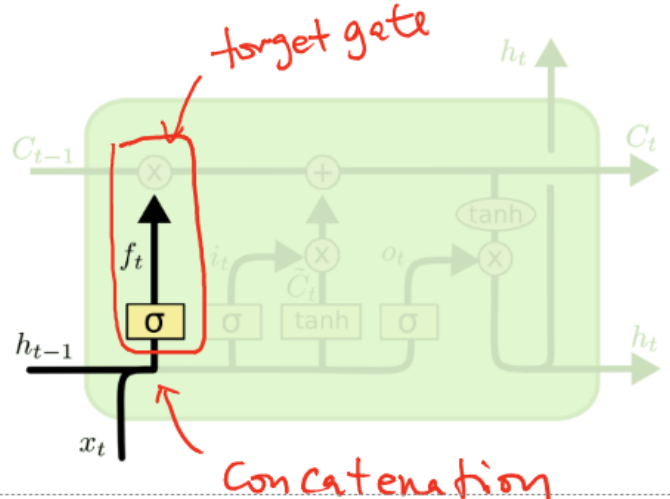
# Overview


$h_{t-1}$
$h_t$
$h_{t+1}$
element wise X
A
A
tanh
n vector
σ
σ
tanh
σ
n-vector
$X_{t-1}$
$X_t$
n-vector
$X_{t+1}$




$C_t$: Cell state

"Forget gate": Control whether to forget the previous cell state $C_{t-1}$



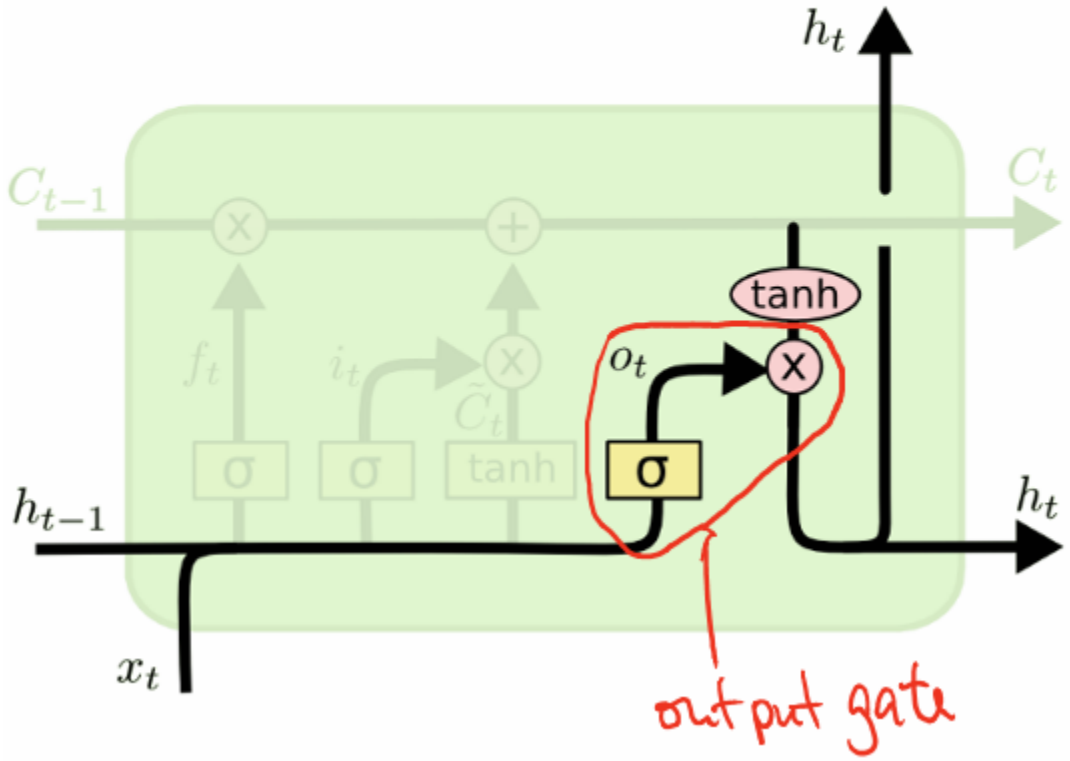

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

# Input gate

input gate

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

# output gate

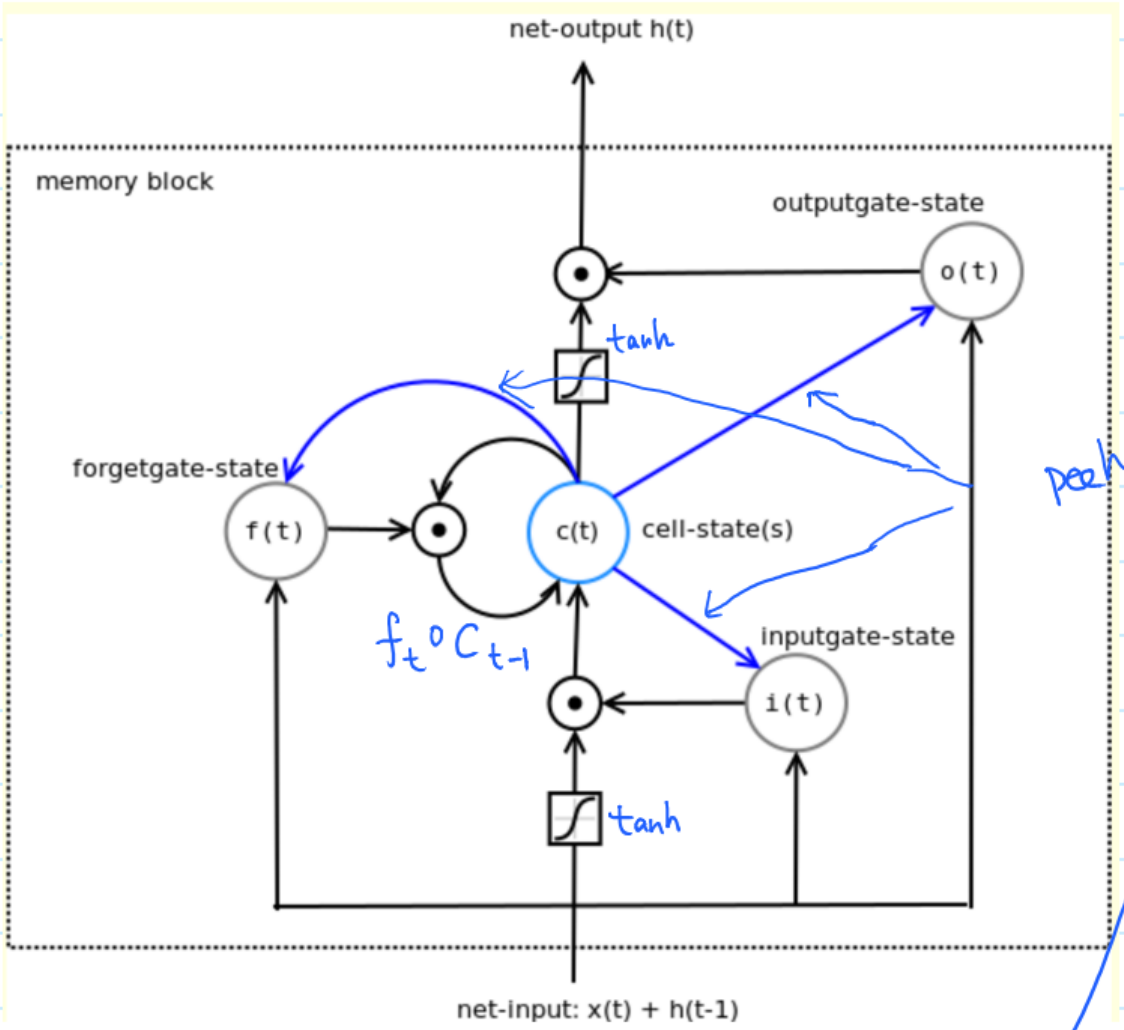



output gate

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

$$h_t = o_t * \tanh\left(C_t\right)$$

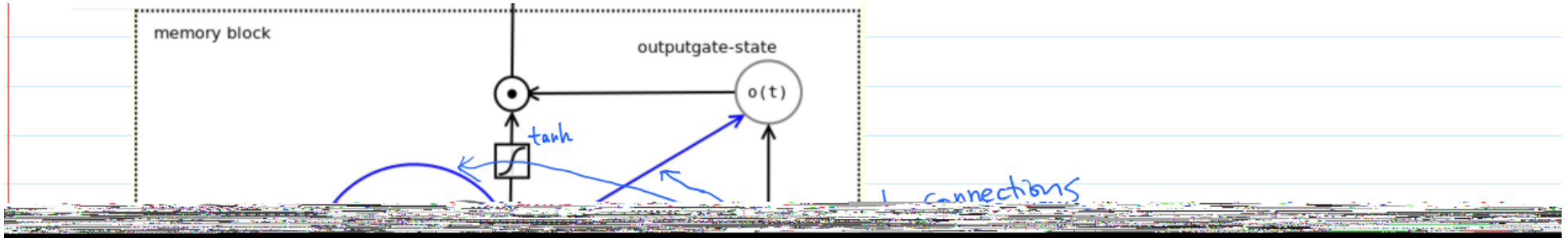
# In many papers, it looks like this

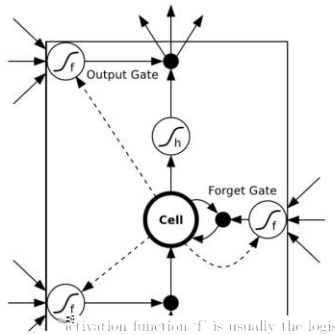



peehole connections

if there is no such peehole, this is equivalent to the previous one

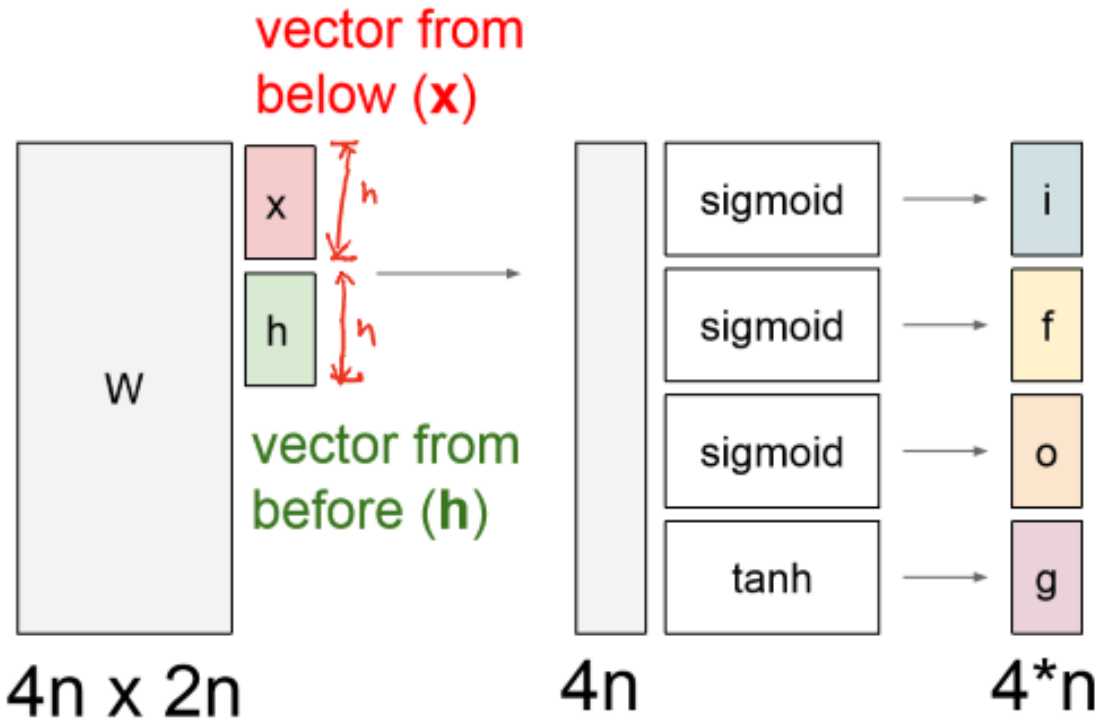
http://christianherta.de/lehre/dataScience/machineLearning/neuralNetworks/LSTM.php




activation function 'f' is usually the logistic sigmoid, so that the gate activations are between 0 (gate closed) and 1 (gate open). The cell input and output activation functions ('g' and 'h') are usually tanh or logistic sigmoid, though in some cases 'h' is the identity function. The weighted 'peephole' connections from the cell to the gates are shown with dashed lines. All other connections within the block are unweighted (or equivalently, have a fixed weight of 1.0). The only outputs from the block to the rest of the network emanate from the output gate multiplication.

# More compact

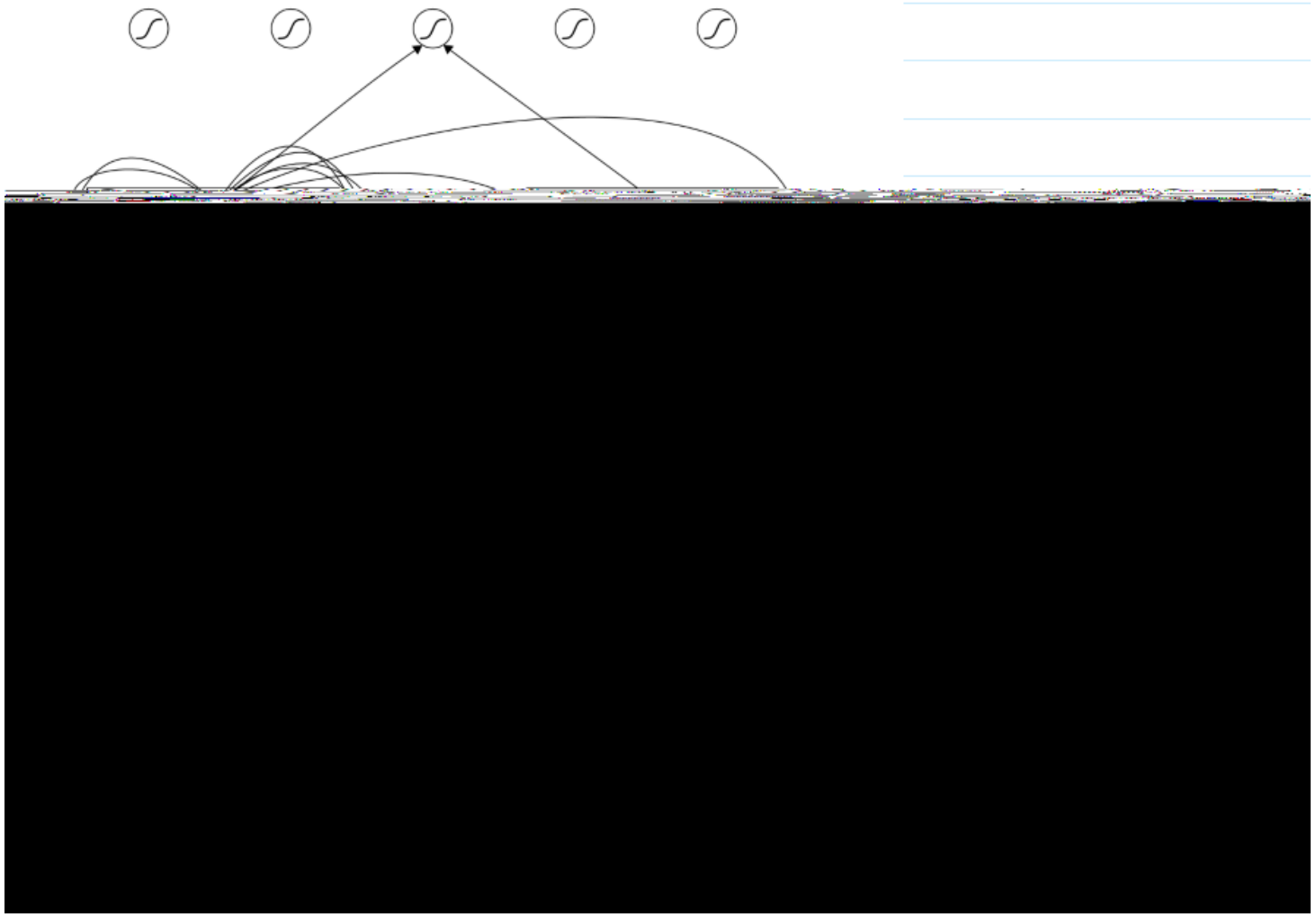



$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}$$

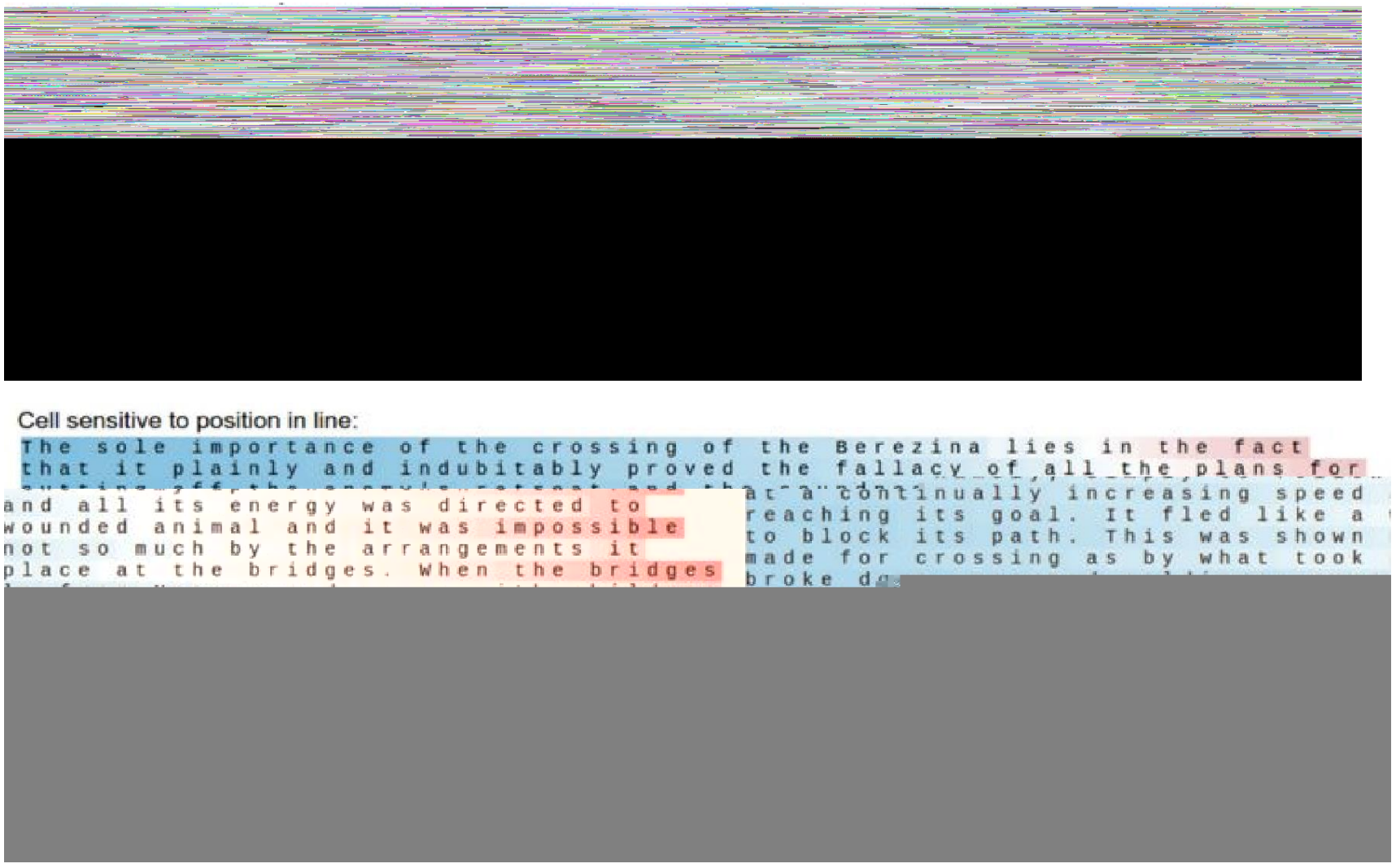
$$c_t^l = f \odot c_{t-1}^l + i \odot g$$

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

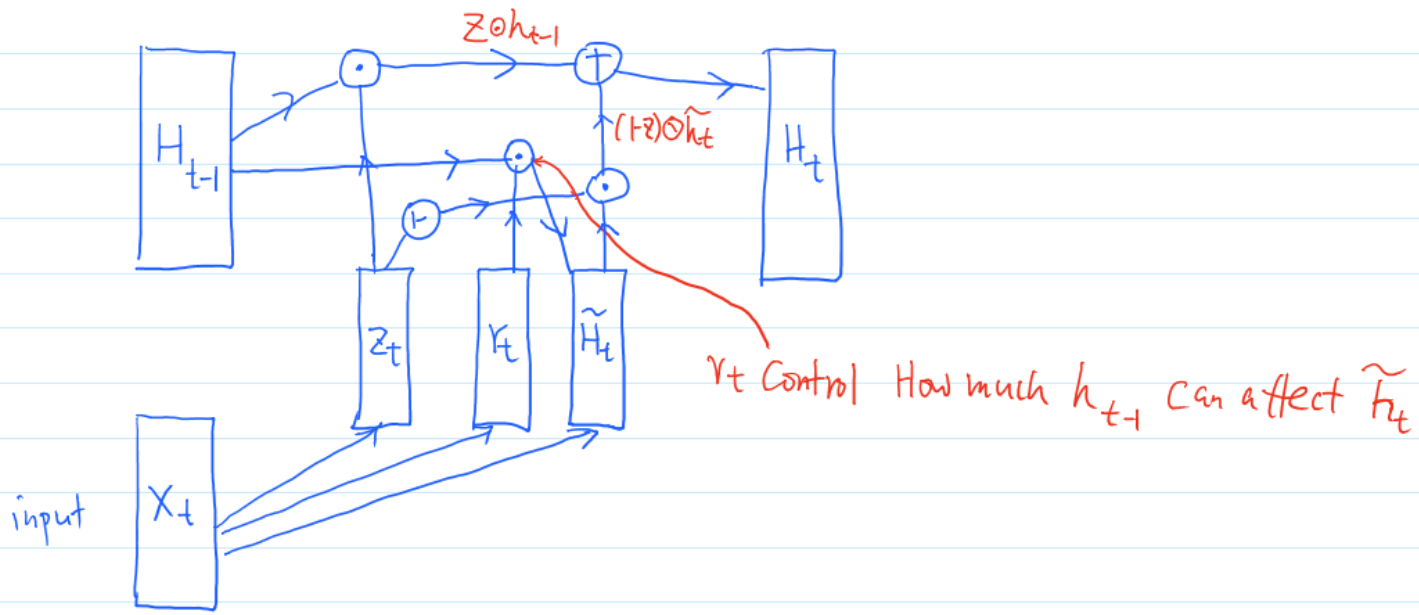
# An LSTM Network



# Visualizing LSTM



# GRU (Gated Recurrent Unit)




$$r_t = \text{sigm}(W_{xr} x_t + W_{hr} h_{t-1} + b_r)$$

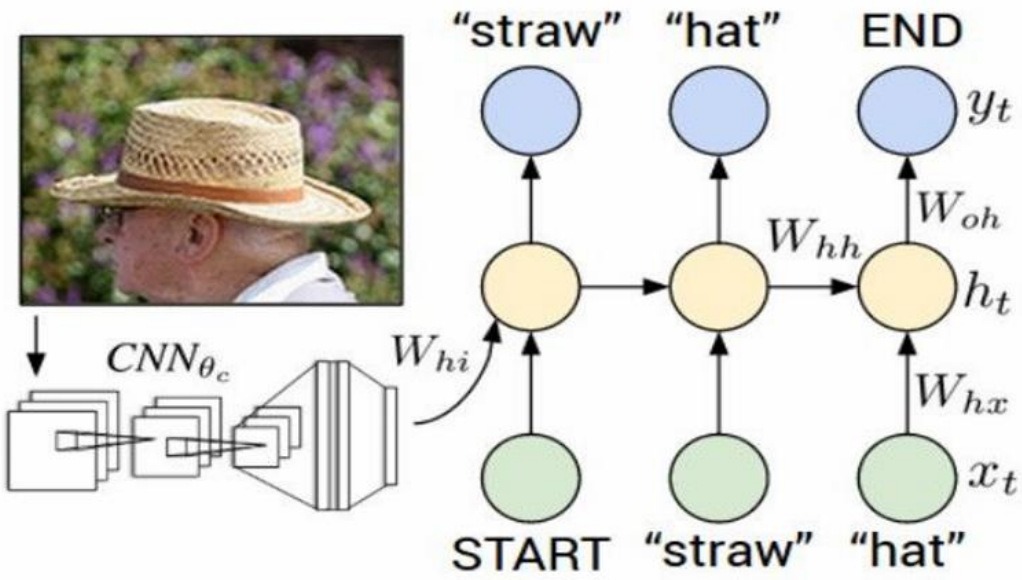
$$z_t = \text{sigm}(W_{xz} x_t + W_{hz} h_{t-1} + b_z)$$

$$\tilde{h}_t = \tanh(W_{xh} x_t + W_{hh}(r_t \odot h_{t-1}) + b_h)$$

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

# Application t Image Captioning

Deep Visual-Semantic Alignments for Generating Image Descriptions, Karpathy, Li




training such RNN is no different from the usual one

training data: CNN +

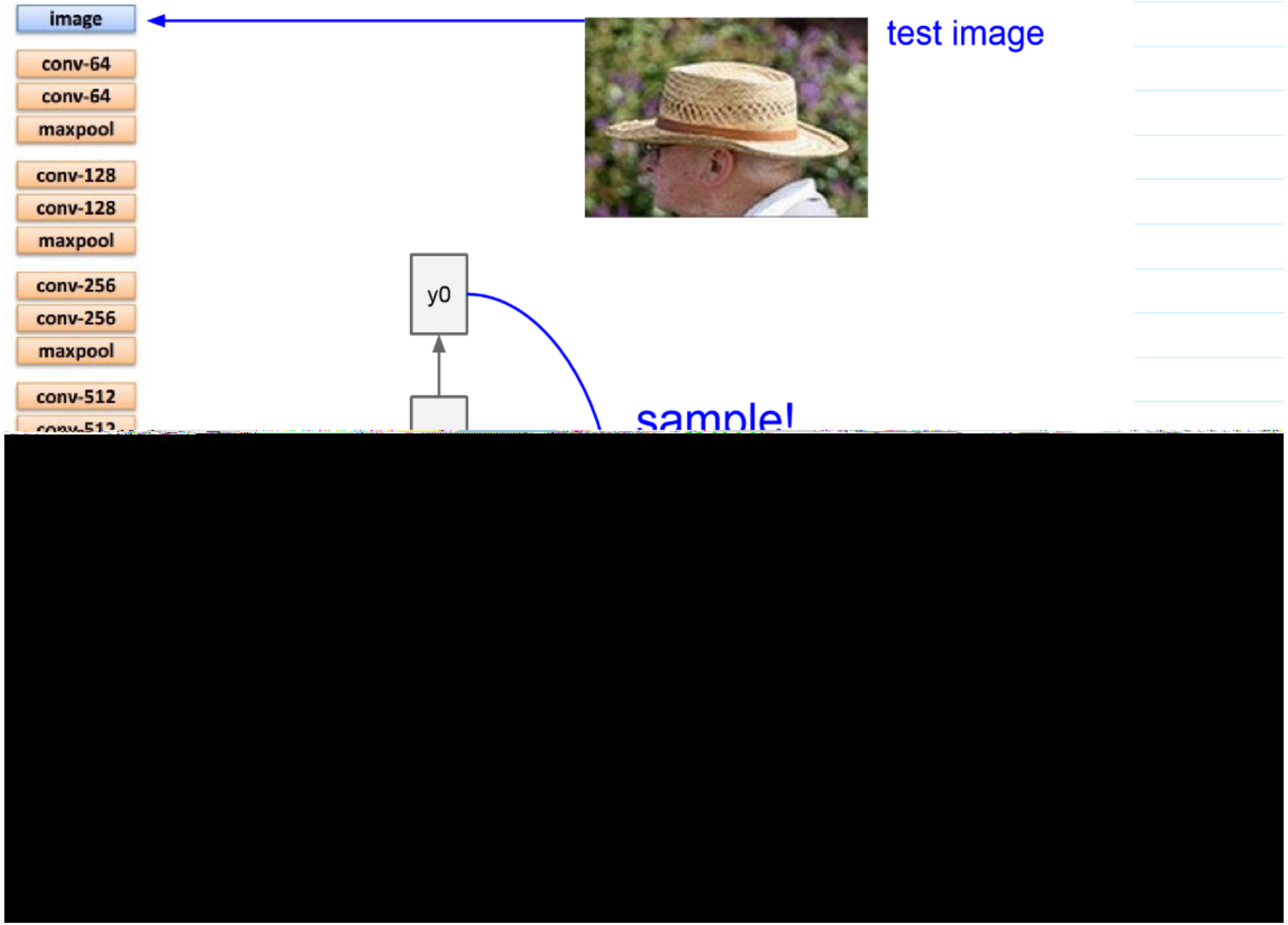


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
test image
y0
sample!

‡Awesome RNN: a lot of useful references

‡https://github.com/kjw0612/awesome-rnn

‡Some slides borrowed from cs231n, cs224d at Stanford

http://cs231n.stanford.edu/syllabus.html

http://cs224d.stanford.edu/index.html