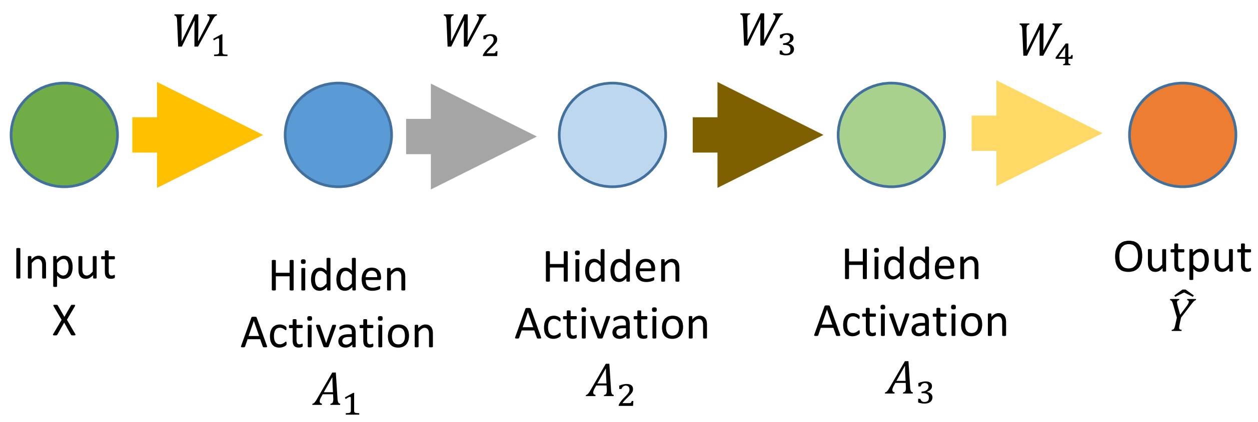
# Gradients of Deep Networks

Chris Cremer

March 29 2017

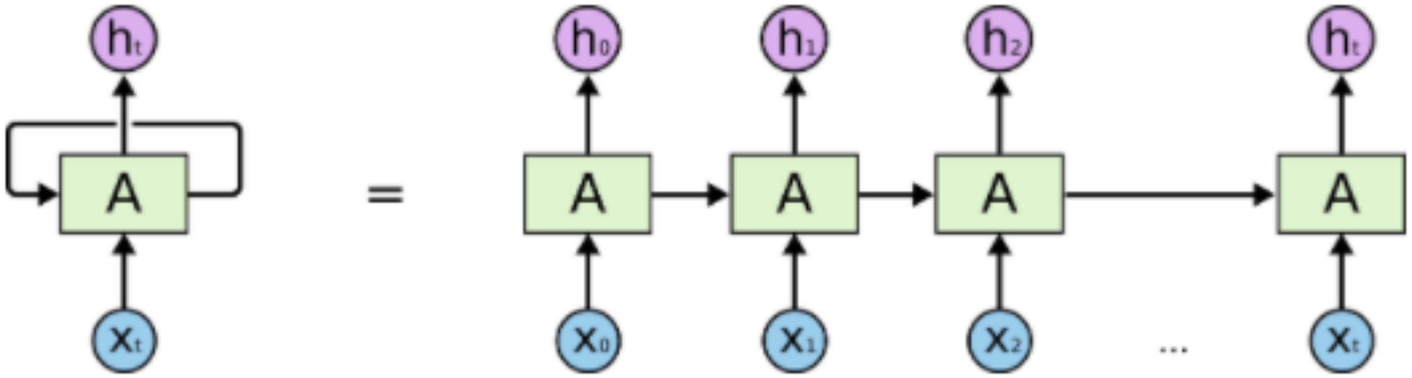
# Neural Net




$$A_t = f(W_t \cdot A_{t-1})$$

Where $f$ = non-linear activation function (sigmoid, tanh, ReLu, Softplus, Maxout, …)

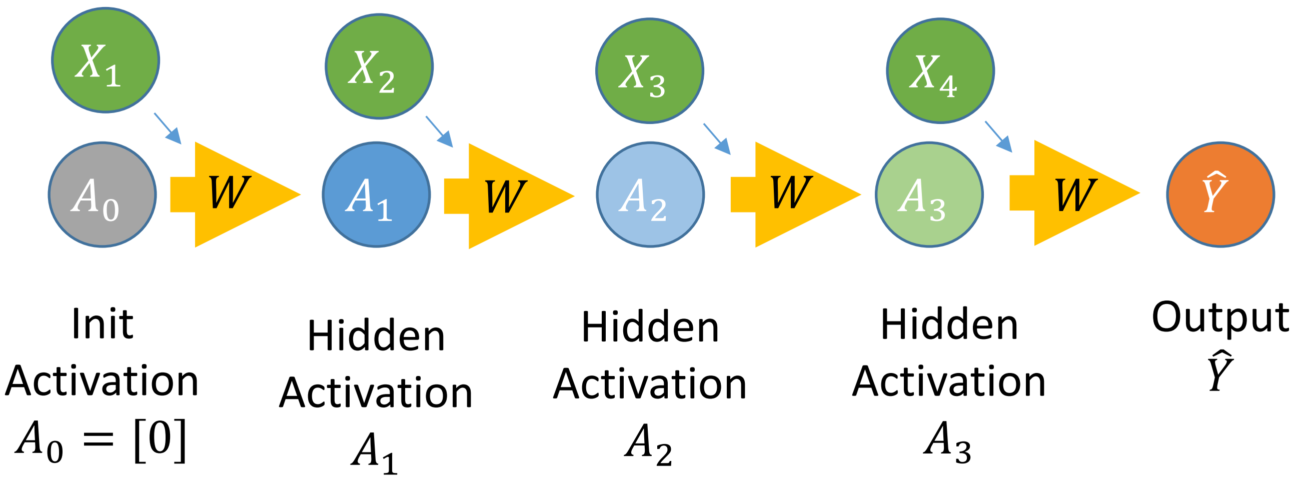
# Recurrent Neural Net




An unrolled recurrent neural network.

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

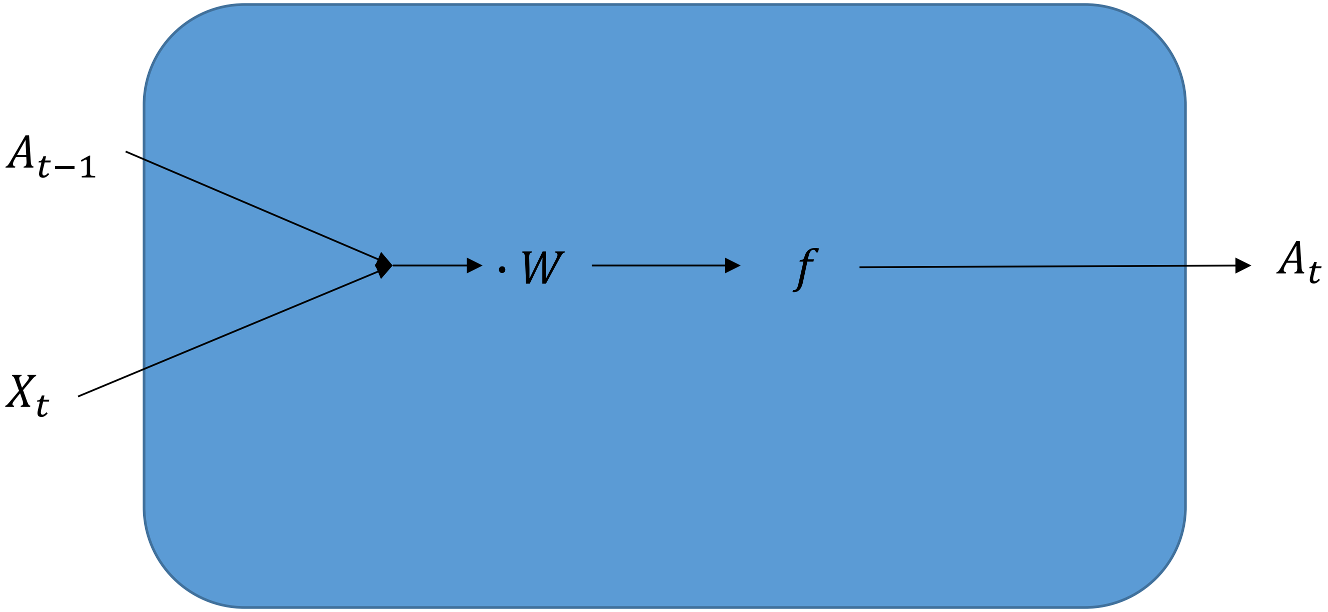
# Recurrent Neural Net



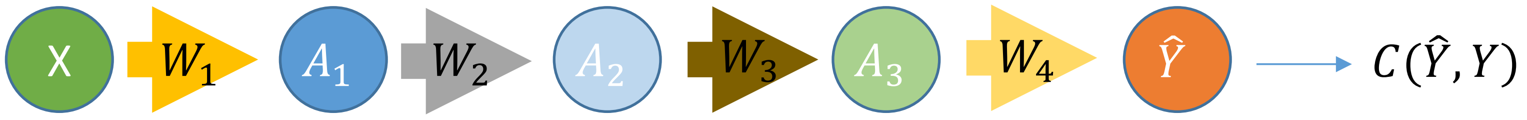

$$A_t = f(W_t \cdot A_{t-1})$$

Where $f$ = non-linear activation function (sigmoid, tanh, ReLu, Softplus, Maxout, …)
Notice that weights are the same

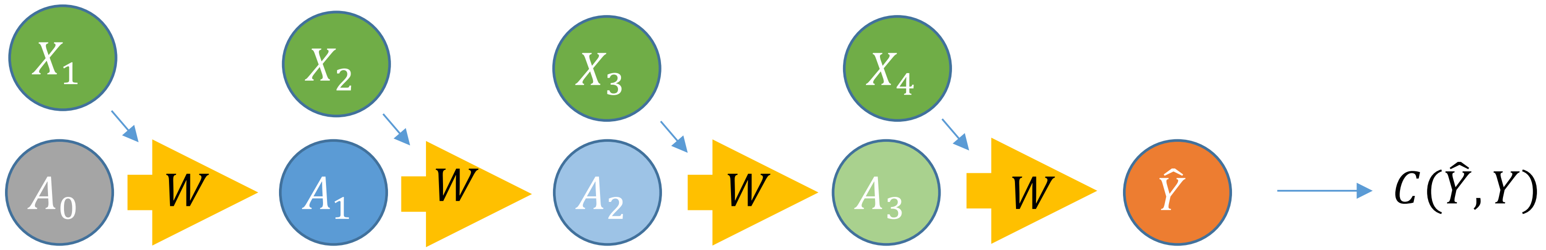
# Recurrent Neural Network – One Timestep



# Gradient Descent



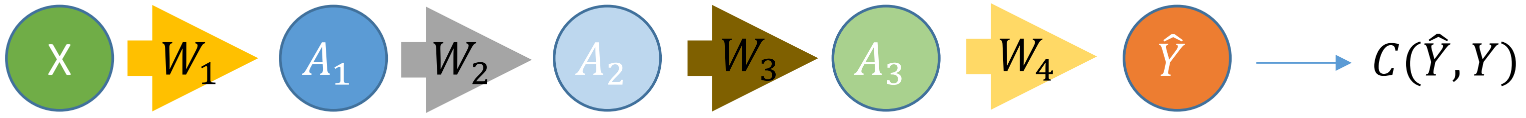

- We want $\frac{\partial C}{W_1}, \frac{\partial C}{W_2}, \frac{\partial C}{W_3}, \ldots$ so that we can do gradient descent: $W_{new} = W_{old} - \alpha \frac{\partial C}{W_{old}}$




Where $C$ is a cost function (Squared error, Cross-Entropy, …)

# Backprop (Chain Rule)

We want $\frac{\partial C}{W_1}, \frac{\partial C}{W_2}, \frac{\partial C}{W_3} \ldots$




$A_t = f(W_t \cdot A_{t-1})$

$f$ = non-linear activation function (sigmoid, tanh, ReLu, Softplus, …)

$C$ = cost function (Squared error, Cross-Entropy, …)

$$\frac{\partial C}{\partial \hat{Y}} = derivative\ of\ cost\ function$$

$$\frac{\partial C}{\partial W_4} = \frac{\partial C}{\partial \hat{Y}} \cdot \frac{\partial \hat{Y}}{\partial W_4} \quad derivative\ of\ activation\ function$$

$$\frac{\partial C}{\partial A_2} = \frac{\partial C}{\partial \hat{Y}} \cdot \frac{\partial \hat{Y}}{\partial A_2} = \frac{\partial C}{\partial \hat{Y}} \cdot \frac{\partial \hat{Y}}{\partial A_3} \cdot \frac{\partial A_3}{\partial A_2}$$

$$\frac{\partial A_t}{\partial A_{t-1}} = \frac{\partial f(W_t \cdot A_{t-1})}{\partial A_{t-1}} = f'(W_t \cdot A_{t-1}) W_t$$

<u>Example</u>

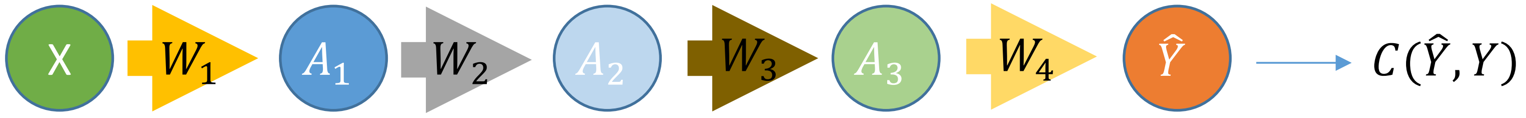
$$C(\hat{Y}, Y) = (\hat{Y} - Y)^2$$

$$\frac{\partial C}{\partial \hat{Y}} = 2(\hat{Y} - Y)$$

$$A_t = \sigma(W_t \cdot A_{t-1})$$

$$\frac{\partial A_t}{\partial A_{t-1}} = A_t(1 - A_t)$$

# Vanishing/Exploding Gradient




$$\frac{\partial A_t}{\partial A_{t-1}} = \frac{\partial f(W_t \cdot A_{t-1})}{\partial A_{t-1}} = f'(W_t \cdot A_{t-1})W_t$$

$$\frac{\partial C}{\partial W_1} = \frac{\partial C}{\partial \hat{Y}} \cdot \frac{\partial \hat{Y}}{\partial A_3} \cdot \frac{\partial A_3}{\partial A_2} \cdot \frac{\partial A_3}{\partial A_2} = \frac{\partial C}{\partial \hat{Y}} \cdot (f'(W_t \cdot A_{t-1})W_t)^T$$

$T = number\ of\ layers =$ number of timesteps
For NNs, t goes from T to 0
For RNNs, W is the same for every t

$(f'(W_t \cdot A_{t-1})W_t)^T$

- if $f'(W_t \cdot A_{t-1})W_t > 1$ → Gradient Explodes
- if $f'(W_t \cdot A_{t-1})W_t < 1$ → Gradient Vanishes

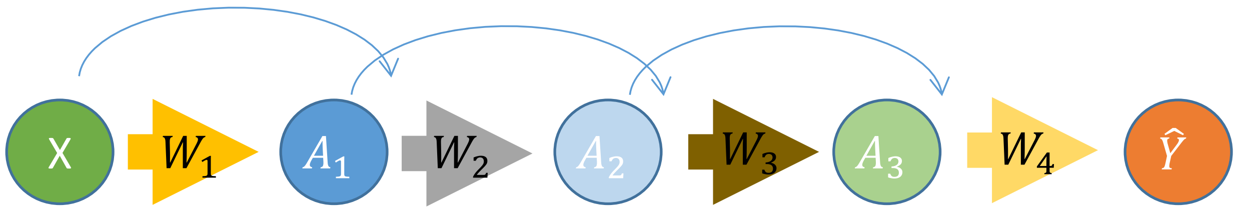
# Resnet/HighwayNet/GRU/LSTM

- NNs:
  - ResNet (2015)
  - Highway Net (2015)

- RNNs:
  - LSTM (1997)
  - GRU (2014)

K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259, 2014.

# Residual Network (ResNet)




$$A_t = f(W_t \cdot A_{t-1}) + A_{t-1}$$

Idea:
- If layer is useless (ie lose information), can skip it
- Easier for network to have zero weights, than be identity

# ResNet Gradient

$$A_t = f(W_t \cdot A_{t-1}) + A_{t-1}$$

$$\frac{\partial A_t}{\partial A_{t-1}} = \frac{\partial f(W_t \cdot A_{t-1}) + A_{t-1}}{\partial A_{t-1}} = f'(W_t \cdot A_{t-1})W_t + 1$$

$$(f'(W_t \cdot A_{t-1})W_t + 1)^T = (Z + 1)^T$$

$$where\ Z = f'(W_t \cdot A_{t-1})W_t$$

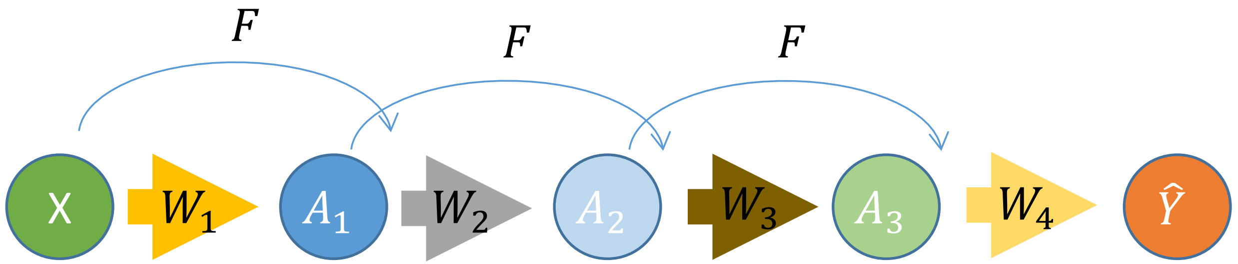
$$(Z + 1)^2 = Z^2 + 2Z + 1$$

$$(Z + 1)^3 = Z^3 + 3Z^2 + 3Z + 1$$

$$(Z + 1)^4 = Z^4 + 4Z^3 + 6Z^2 + 4Z + 1$$

- Vanishing gradient problem: gradient persists through layers
- Exploding gradient problem: weight decay, weight norm, layer norm, batch norm, …

# Highway Network




$$A_t = f(W_t \cdot A_{t-1}) \cdot \mathrm{B} + A_{t-1} \cdot (1 - B)$$

$$B = \sigma(W_{t2} \cdot A_{t-1})$$

$\sigma$ = sigmoid since output (0,1)

# Highway Net Gradient

$$A_t = f(W_t \cdot A_{t-1}) \cdot \mathrm{B} + A_{t-1} \cdot (1 - B) \qquad B = \sigma(W_{t2} \cdot A_{t-1})$$

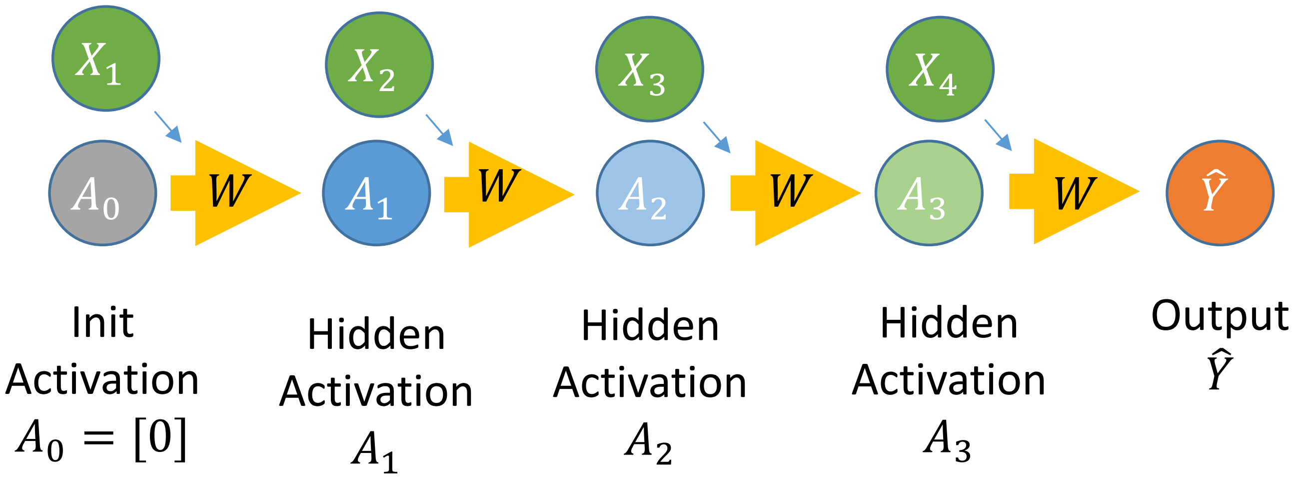
$$= f(W_t \cdot A_{t-1}) \cdot \sigma(W_{t2} \cdot A_{t-1}) + A_{t-1} \cdot (1 - \sigma(W_{t2} \cdot A_{t-1})\,)$$

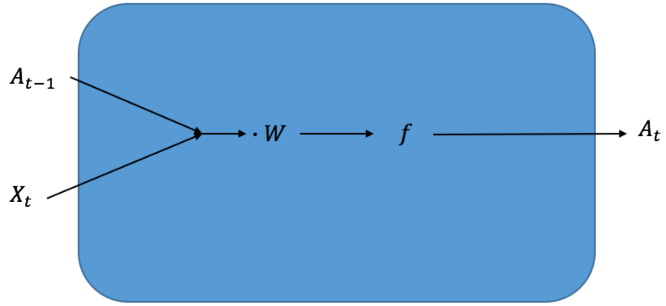
$$= f(W_t \cdot A_{t-1}) \cdot \sigma(W_{t2} \cdot A_{t-1}) + A_{t-1} - \sigma(W_{t2} \cdot A_{t-1}) \cdot A_{t-1}$$

$$\frac{\partial A_t}{\partial A_{t-1}} = \underline{\qquad\qquad} \quad \underline{1} \quad \underline{\qquad\qquad}$$

- Vanishing gradient problem: gradient persists through layers
- Exploding gradient problem: weight decay, weight norm, layer norm, batch norm, …

# Back to RNNs




$$A_t = f(W_t \cdot A_{t-1})$$

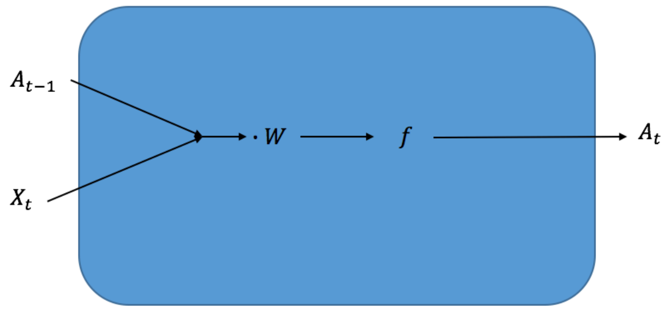
$$\frac{\partial A_t}{\partial A_{t-1}} = \frac{\partial f(W_t \cdot A_{t-1})}{\partial A_{t-1}} = f'(W_t \cdot A_{t-1})W_t$$

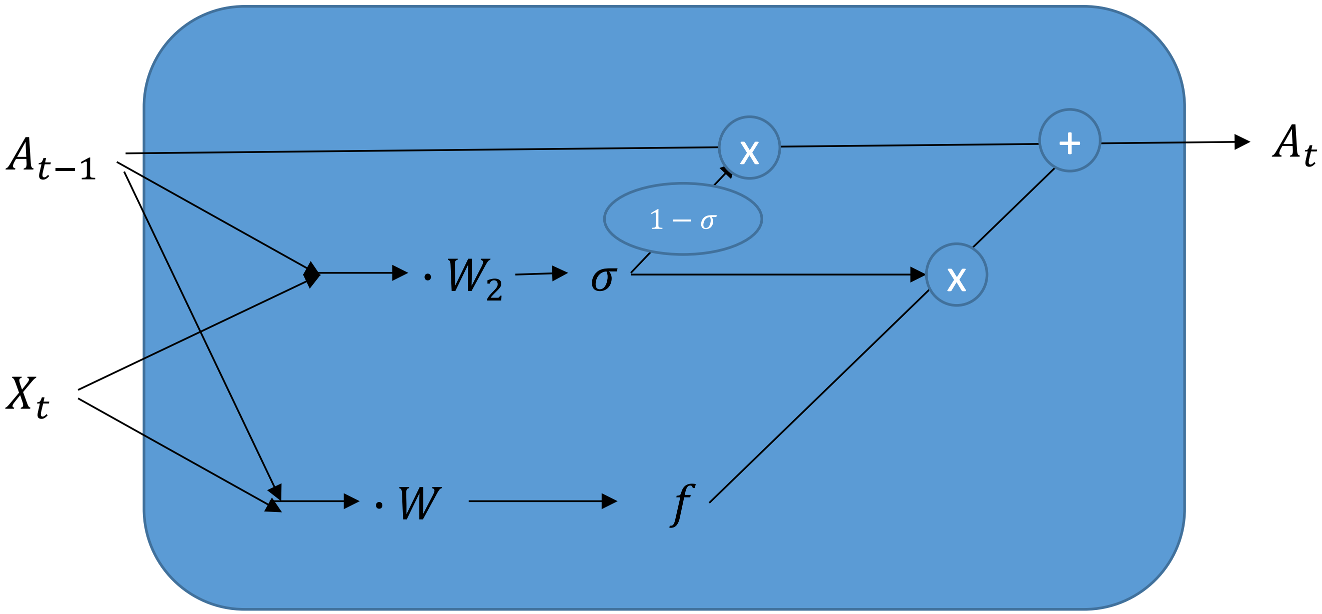
→ Vanishing/Exploding Gradient

Where $f$ = non-linear activation function (sigmoid, tanh, ReLu, Softplus, Maxout, …)
Note: weights are the same

# Gated Recurrent Unit

RNN



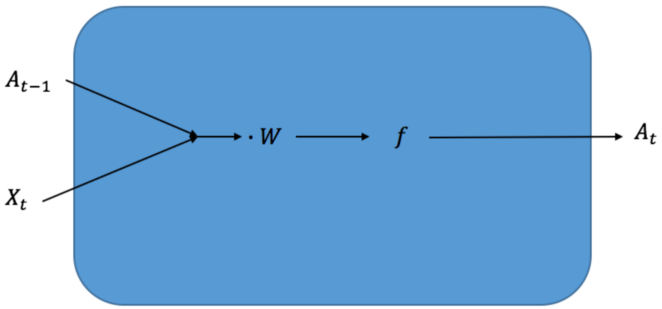

$$A_t = f(W \cdot A_{t-1}) \cdot \mathrm{B} + A_{t-1} \cdot (1 - B)$$

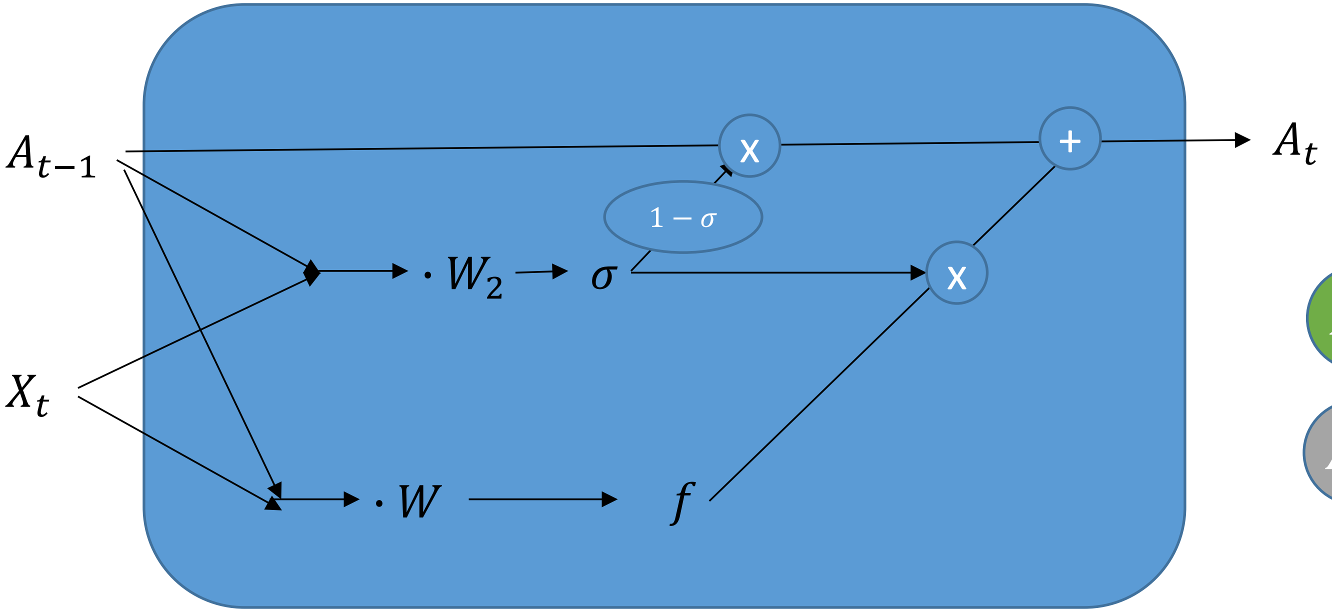
$$B = \sigma(W_2 \cdot A_{t-1})$$

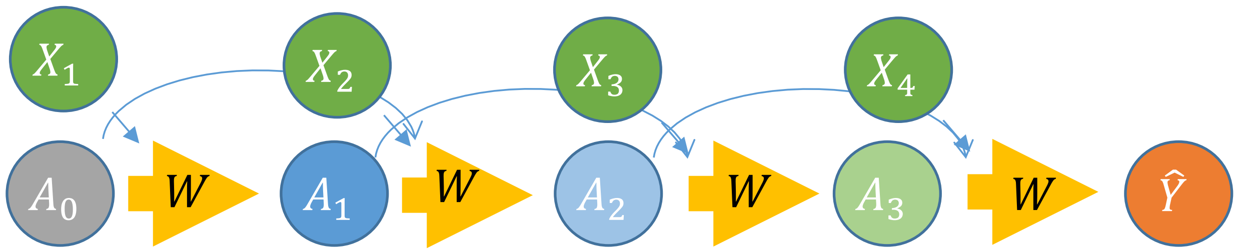
$\sigma$= sigmoid since output (0,1)

# Another view of GRUs

RNN



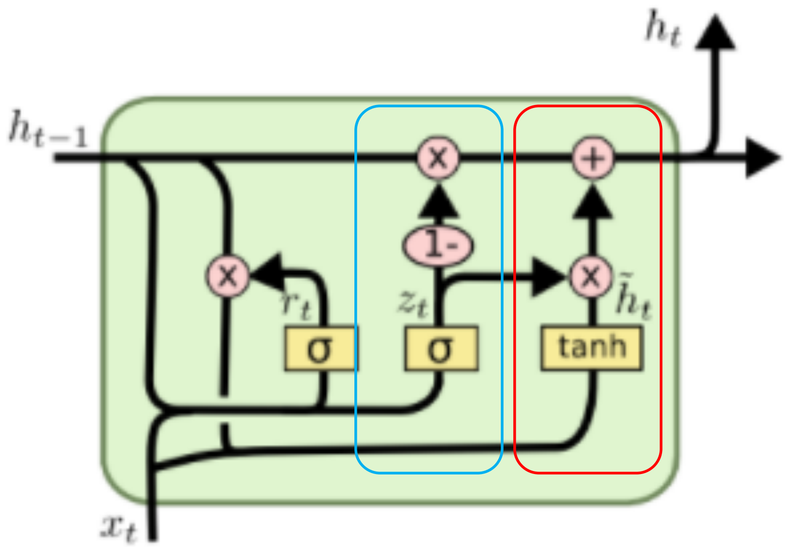




$$A_t = f(W \cdot A_{t-1}) \cdot \mathrm{B} + A_{t-1} \cdot (1 - B)$$

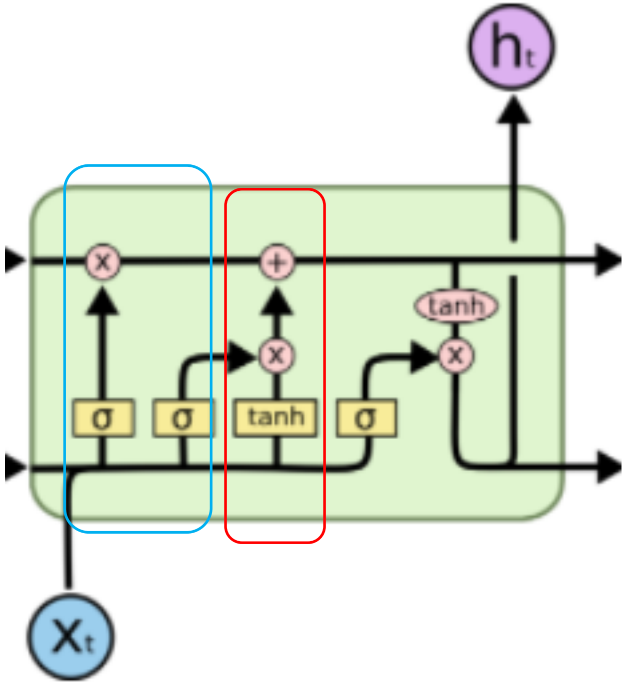
$$B = \sigma(W_2 \cdot A_{t-1})$$

$\sigma$= sigmoid since output (0,1)

# GRU/LSTM: More Gates




GRU




LSTM

# Memory Concerns

- If T=10000, you need to keep 10000 activations/states in memory

# Deep Network Gradients Conclusion

- The models we saw all use the same idea
- One of the earlier uses of skip connections was in the Nonlinear AutoRegressive with eXogenous inputs method (NARX; Lin et al., 1996), where they improved the RNN’s ability to infer finite state machines.

    - Ilya Sustkever PhD thesis 2013

NNs:

ResNet (2015)
Highway Net (2015)

Neither ResNet or Highway reference GRUs/LSTMs

RNNs:

LSTM (1997)
GRU (2014)

# Thanks