Introduction to Deep Learning

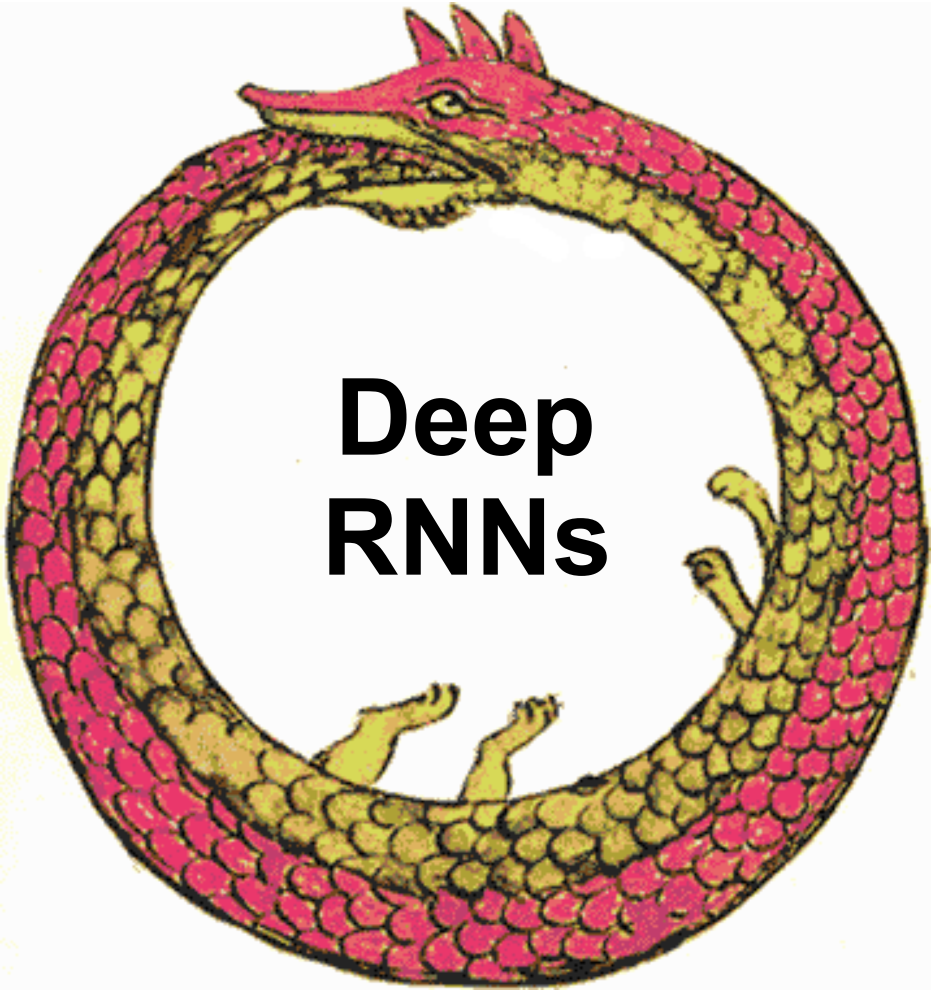
20. Advanced Recurrent Networks

STAT 157, Spring 2019, UC Berkeley

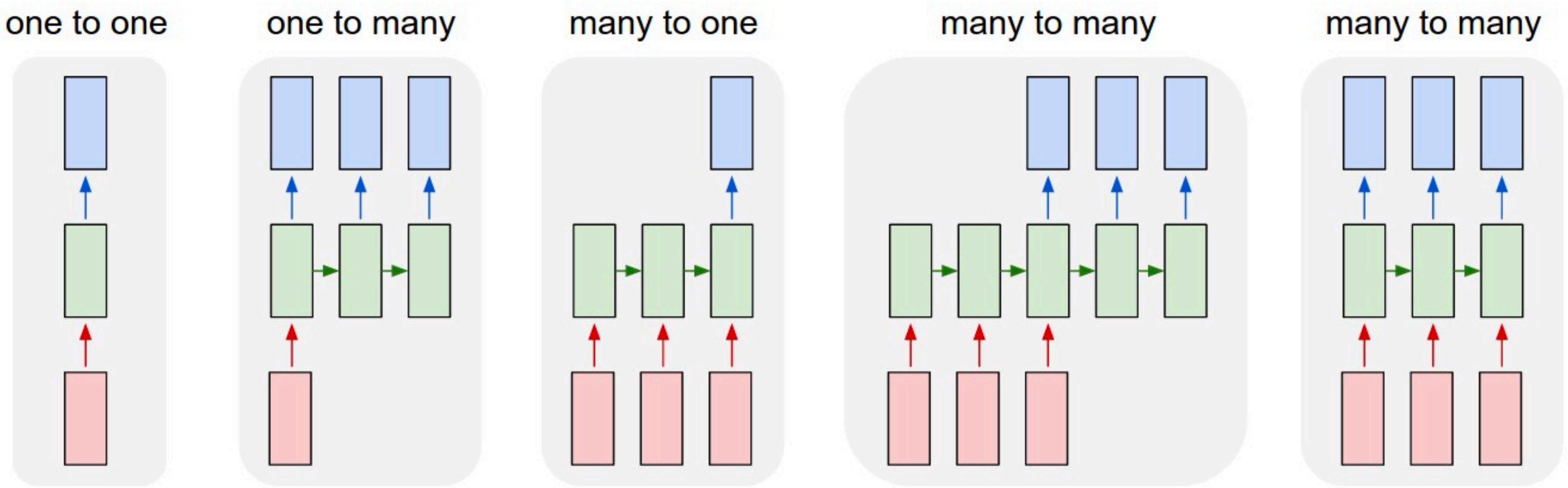
Alex Smola and Mu Li

courses.d2l.ai/berkeley-stat-157

Deep RNNs



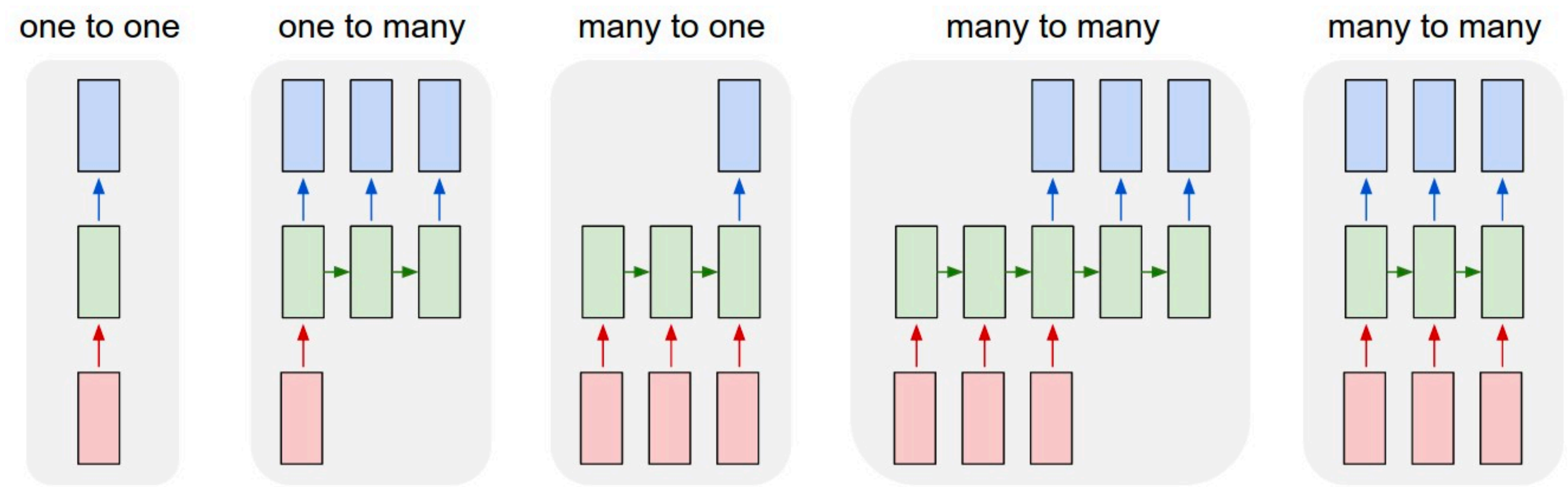
Using RNNs



- Encode sequence
- Decode sequence
- Do both

(image courtesy of karpathy.github.io)

Using RNNs



Poetry Generation

Sentiment Analysis

Document Classification

Question Answering

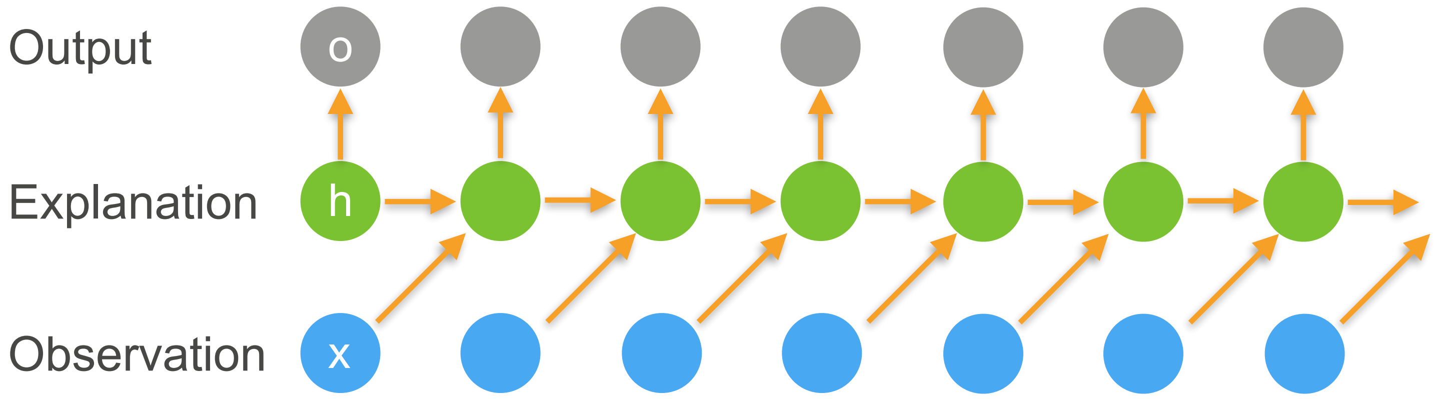
Machine Translation

Named Entity Tagging

courses.d2l.ai/berkeley-stat-157

(image courtesy of karpathy.github.io)

Recall - Recurrent Neural Networks



- Hidden State update

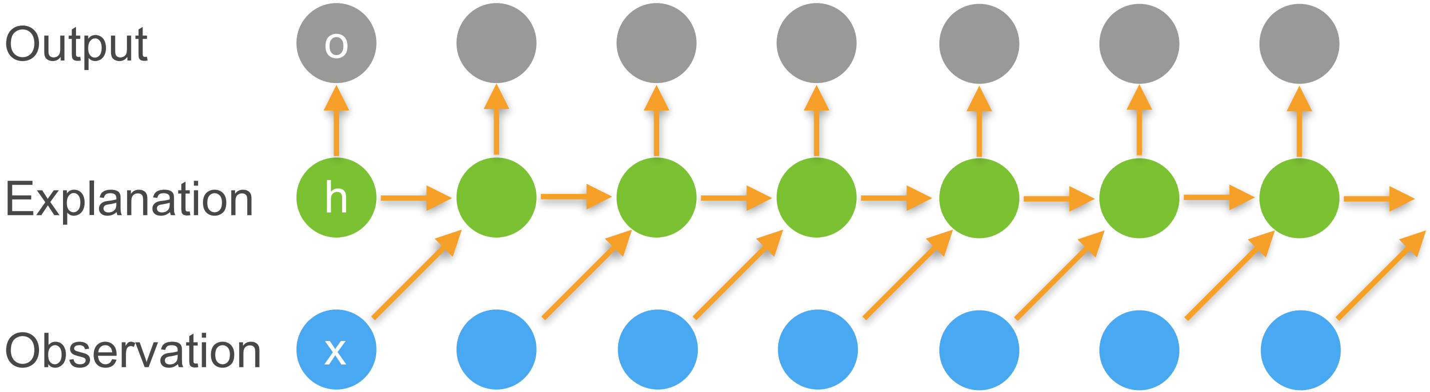
$$\mathbf{h}_t = \phi(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{hx}\mathbf{x}_{t-1} + \mathbf{b}_h)$$

- Observation update

$$\mathbf{o}_t = \phi(\mathbf{W}_{ho}\mathbf{h}_t + \mathbf{b}_o)$$

How to make more nonlinear?

Plan A - Nonlinearity in the units



- Hidden State update

$$\mathbf{h}_t = \phi(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{hx}\mathbf{x}_{t-1} + \mathbf{b}_h)$$

- Observation update

$$\mathbf{o}_t = \phi(\mathbf{W}_{ho}\mathbf{h}_t + \mathbf{b}_o)$$

Replace with MLP?

Plan A - Nonlinearity in the units

- Keeps the structure of the latent space
- More complex gradients (very costly)
- E.g. Zoph et al, 2018 learned cells with ~40 units (slow and expensive - nobody uses them in practice)

- Hidden State update

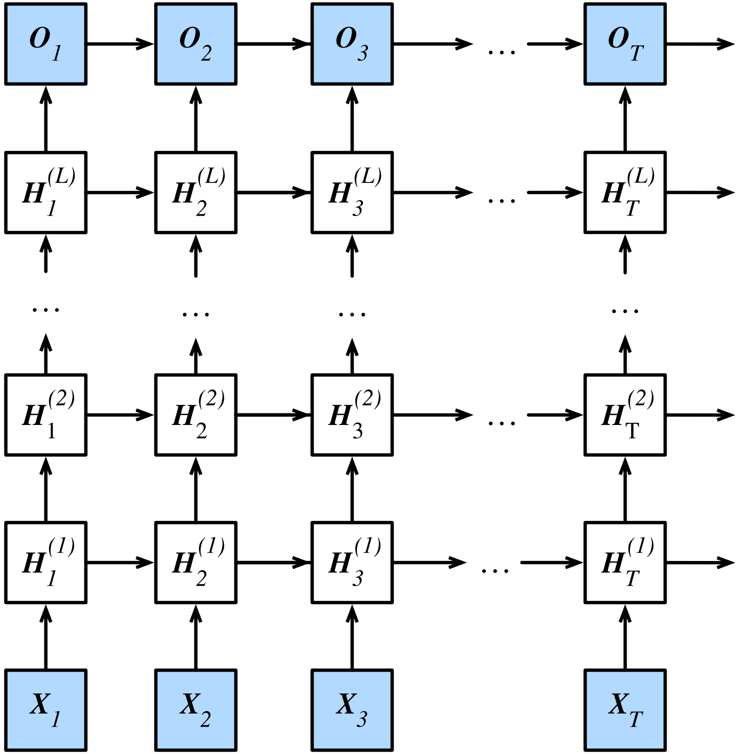
$$\mathbf{h}_t = \phi(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{hx}\mathbf{x}_{t-1} + \mathbf{b}_h)$$

- Observation update

$$\mathbf{o}_t = \phi(\mathbf{W}_{ho}\mathbf{h}_t + \mathbf{b}_o)$$

Replace with MLP?

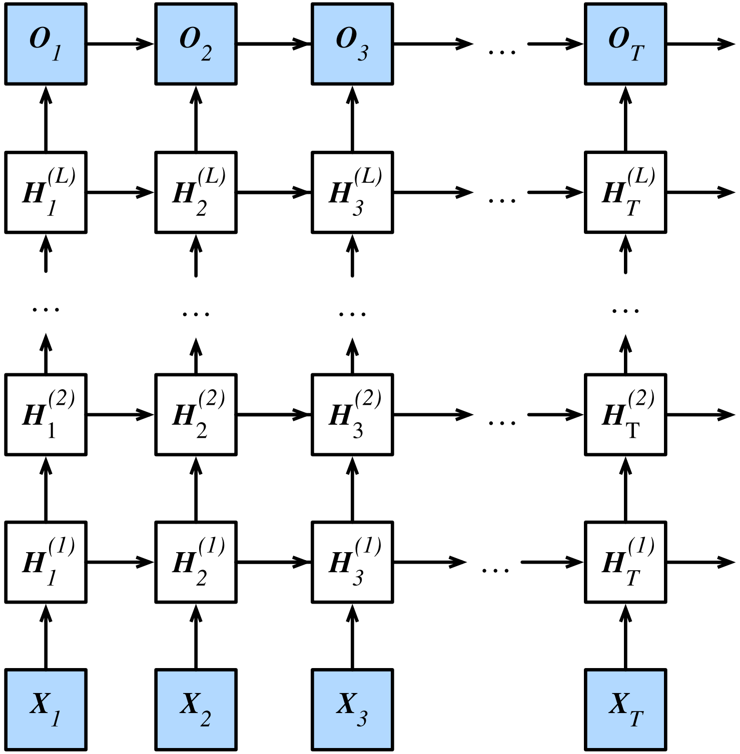
Plan B - We go deeper



- Shallow RNN
  - Input
  - Hidden layer
  - Output
- Deep RNN
  - Input
  - **Hidden layer**
  - **Hidden layer**

  …
  - Output

Plan B - We go deeper



$$\mathbf{H}_t = f(\mathbf{H}_{t-1}, \mathbf{X}_t)$$
$$\mathbf{O}_t = g(\mathbf{H}_t)$$

$$\mathbf{H}_t^1 = f_1(\mathbf{H}_{t-1}^1, \mathbf{X}_t)$$
$$\mathbf{H}_t^j = f_j(\mathbf{H}_{t-1}^j, \mathbf{H}_t^{j-1})$$
$$\mathbf{O}_t = g(\mathbf{H}_t^L)$$

Code …


JOHN COLTRANE BOTH
DIRECTIONS AT ONCE
THE LOST ALBUM
impulse!
impulse!
Bidirectional RNNs

The Future Matters

```
I am _____
I am _____ very hungry,
I am _____ very hungry, I could eat half a pig.
```

The Future Matters

```
I am happy.
I am not    very hungry,
I am very   very hungry, I could eat half a pig.
```

The Future Matters

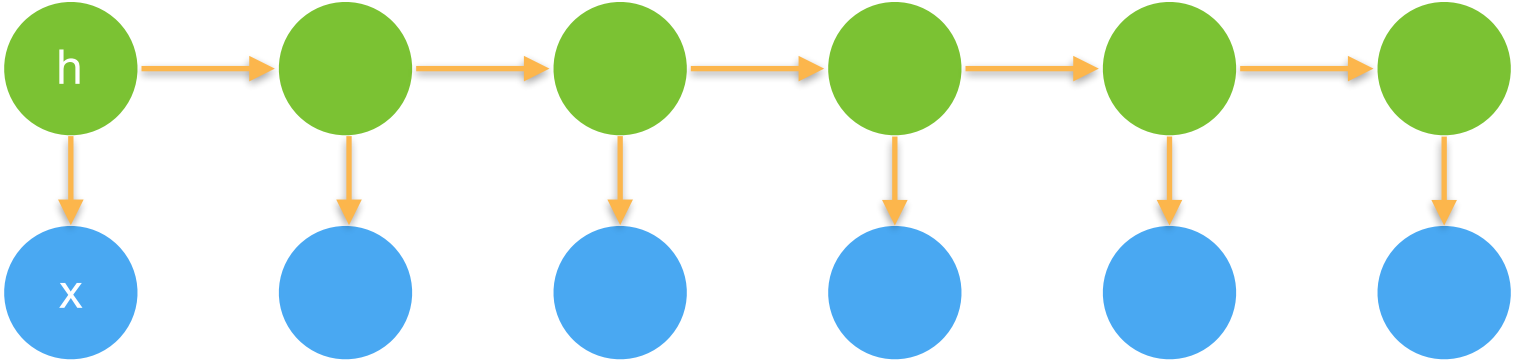
```
I am happy.
I am not    very hungry,
I am very   very hungry, I could eat half a pig.
```

- Very different words to fill in, depending on past and **future** context of a word.
- RNNs so far only look at the past
- In interpolation (fill in) we can use the future, too.

Flashback - Graphical Models

- Hidden Markov Model

$p(h_t \mid h_{t-1}, x_{t-1})$ and $p(x_t \mid h_t, x_{t-1})$



- Can model sequence jointly and solve by dynamic programming

Dynamic programming

- Joint probability

$$p(x, h) = p(h_1)p(x_1 \mid h_1)\prod_{i=2}^{T} p(h_t \mid h_{t-1})p(x_t \mid h_t)$$

Dynamic programming

$$p(x) = \sum_h p(h_1)p(x_1 | h_1) \prod_{i=2}^{T} p(h_t | h_{t-1}) p(x_t | h_t)$$

$$= \sum_{h_2,\ldots h_T} \underbrace{\left[\sum_{h_1} p(h_1)p(x_1 | h_1)p(h_2 | h_1) \right]}_{=:\pi_2(h_2)} p(x_2 | h_2) \prod_{i=2}^{T} p(h_t | h_{t-1}) p(x_t | h_t)$$

$$= \sum_{h_3,\ldots h_T} \underbrace{\left[\sum_{h_2} \pi_2(h_2)p(x_2 | h_2)p(h_3 | h_2) \right]}_{=:\pi_3(h_3)} p(x_3 | h_3) \prod_{i=3}^{T} p(h_t | h_{t-1}) p(x_t | h_t)$$

Dynamic programming

- Joint probability

$$p(x, h) = p(h_1)p(x_1 \mid h_1)\prod_{i=2}^{T} p(h_t \mid h_{t-1})p(x_t \mid h_t)$$

- Forward pass

$$\pi_{t+1}(h_{t+1}) = \sum_{h_t} \pi_t(h_t)p(x_t \mid h_t)p(h_{t+1} \mid h_t)$$

Dynamic programming

$$p(x) = \sum_h \prod_{i=1}^{T-1} p(h_t | h_{t-1}) p(x_t | h_t) \cdot p(h_T | h_{T-1}) p(x_T | h_T)$$

$$= \sum_{h_1, \ldots h_{T-1}} \prod_{i=1}^{T-1} p(h_t | h_{t-1}) p(x_t | h_t) \cdot \underbrace{\left[\sum_{h_T} p(h_T | h_{T-1}) p(x_T | h_T) \right]}_{=:\rho_{T-1}(h_{T-1})}$$

$$= \sum_{h_1, \ldots h_{T-2}} \prod_{i=1}^{T-2} p(h_t | h_{t-1}) p(x_t | h_t) \cdot \underbrace{\left[\sum_{h_{T-1}} p(h_{T-1} | h_{T-2}) p(x_{T-1} | h_{T-1}) \right]}_{=:\rho_{T-2}(h_{T-2})}$$

Dynamic programming

- Joint probability

$$p(x,h) = p(h_1)p(x_1 \mid h_1)\prod_{i=2}^{T} p(h_t \mid h_{t-1})p(x_t \mid h_t)$$

- Forward pass

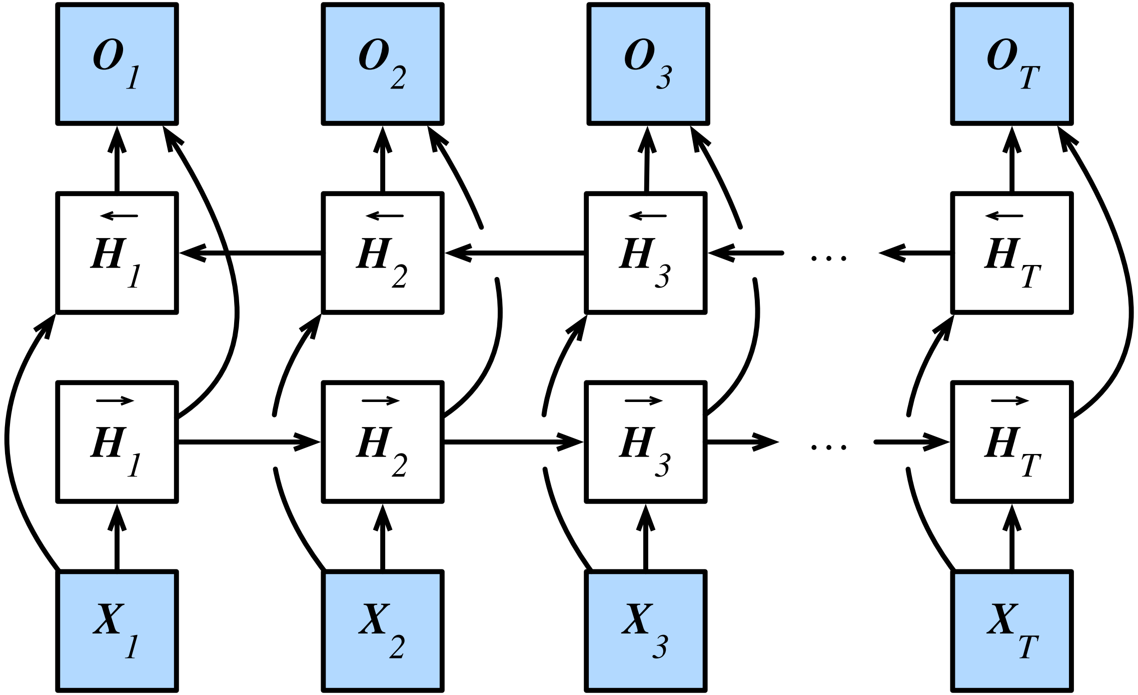
$$\pi_{t+1}(h_{t+1}) = \sum_{h_t} \pi_t(h_t)p(x_t \mid h_t)p(h_{t+1} \mid h_1)$$

- Backward pass

$$p(x_j \mid x_{-j}) \propto \sum_{h_j} \pi_j(h_j)\rho_j(h_j)p(x_j \mid h_j)$$

Can we do this with RNNs, too?

Bidirectional RNN



- One RNN forward
- Another one backward
- Combine both hidden states for output generation

```
epoch 600, perplexity 1.016867, time 0.15 sec
 - travellerer cumplph peougunininininin suppepepepepepepepepe
 - time travellererer  fuf this shanatatatatatatatatatatatatatatata
epoch 800, perplexity 1.007069, time 0.15 sec
 - traveller hime of copspepepep smefsffff''''''''''''''''''''
 - time travellerer prefifififididididididididididididididididididi
epoch 1000, perplexity 1.001932, time 0.15 sec
 - travellerererererererererererererererererererererererererer
 - time travellerererererererererererererererererererererererererer
```

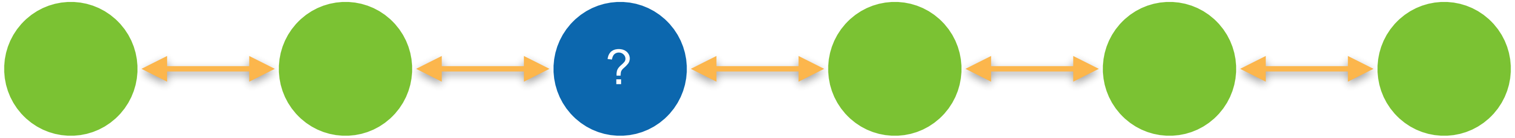
This does not work for sequence generation

```
epoch 600, perplexity 1.016867, time 0.15 sec
 - travellerer cumplph peouguninininin suppepepepepepepepepe
 - time travellererer  fuf this shanatatatatatatatatatatatatata
epoch 800, perplexity 1.007069, time 0.15 sec
 - traveller hime of copspepepep smefsffff''''''''''''''''''
 - time travellerer prefififififidididididididididididididididi
epoch 1000, perplexity 1.001932, time 0.15 sec
 - travellererererererererererererererererererererererererer
 - time travellerererererererererererererererererererererererer
```

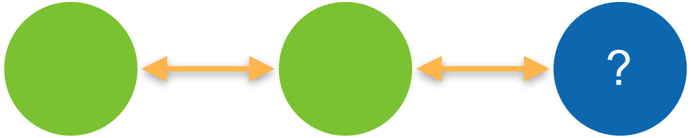
Why?

Reasons

- Training time



- Test time

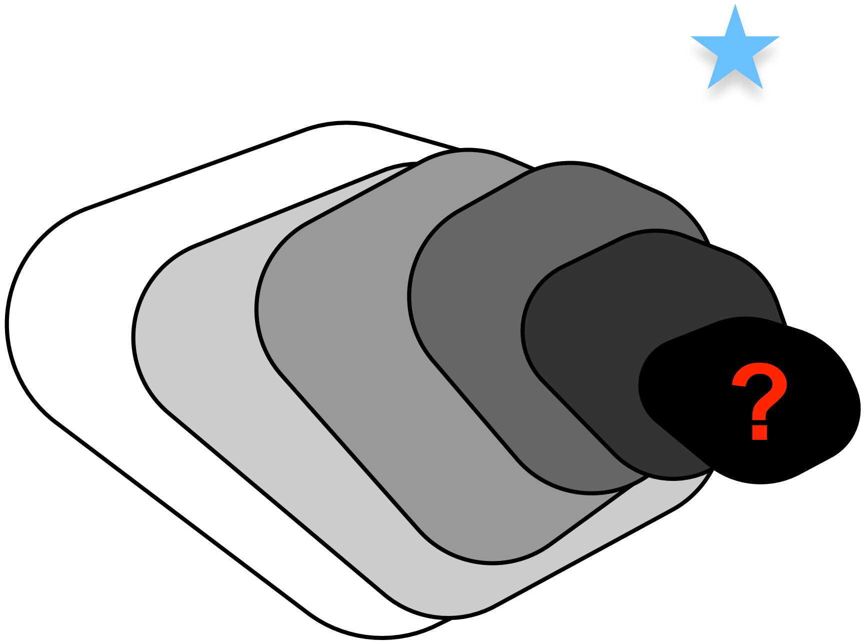


Next lecture

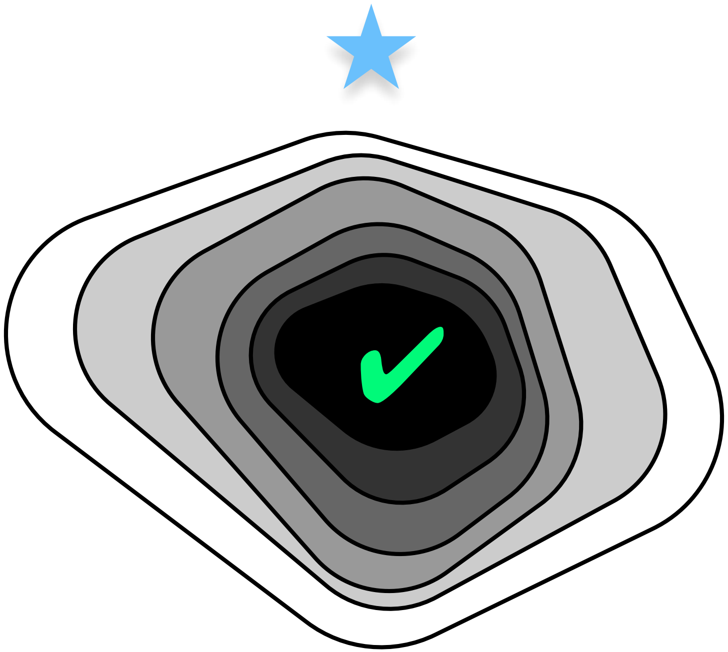
Can still use it to **encode** the sequence

Residual Connections for RNNs

Flashback - Does adding layers improve accuracy?



generic function classes

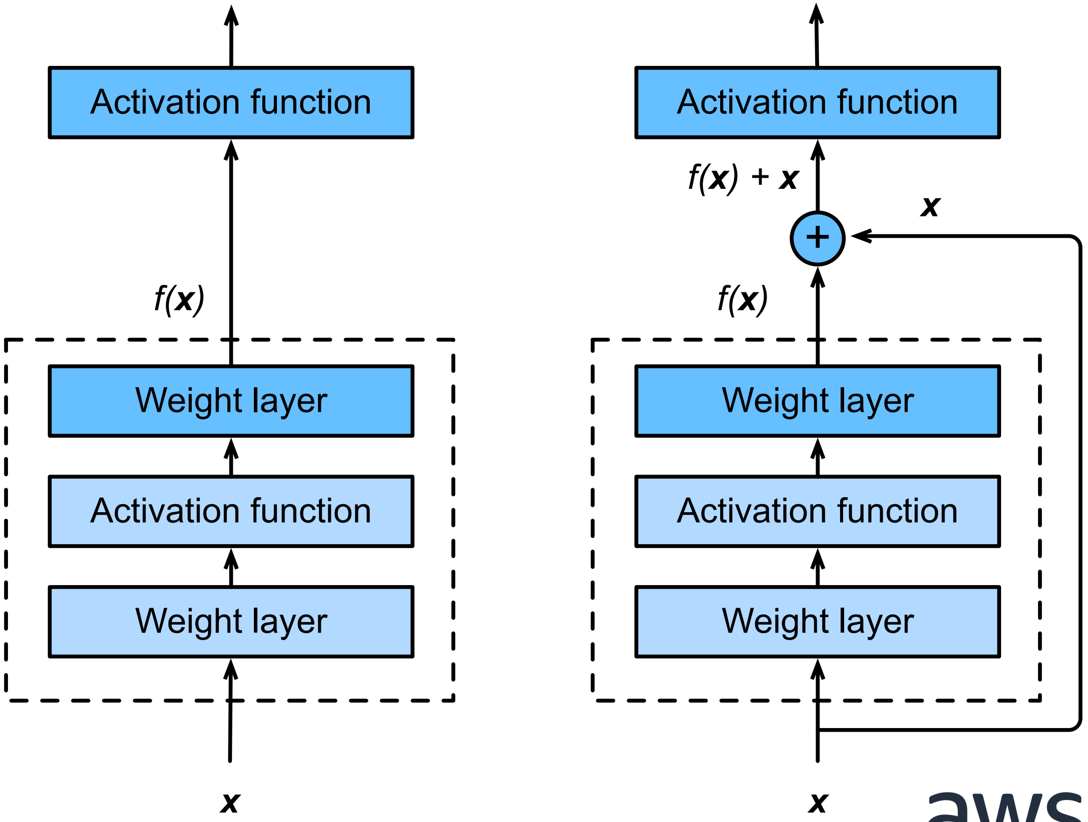


nested function classes

Flashback - Residual Networks

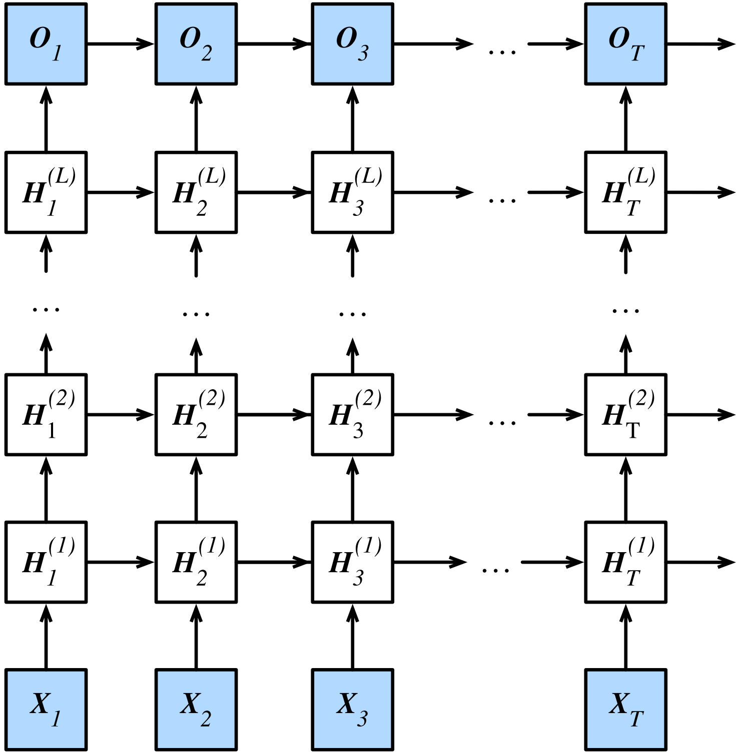
- Adding a layer **changes** function class
- We want to **add to** the function class
- 'Taylor expansion' style parametrization

$$f(x) = x + g(x)$$



He et al., 2015

Flashback - Deep RNNs

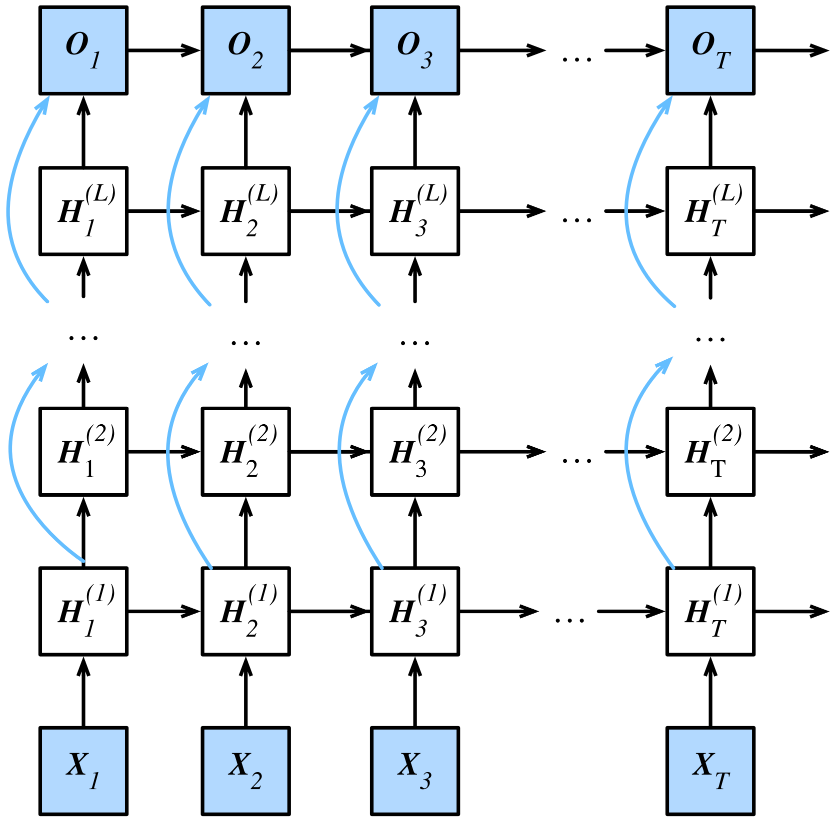


- Deep RNN
  - Input
  - **Hidden layer**
  - **Hidden layer**

    …
  - Output

Drumroll …

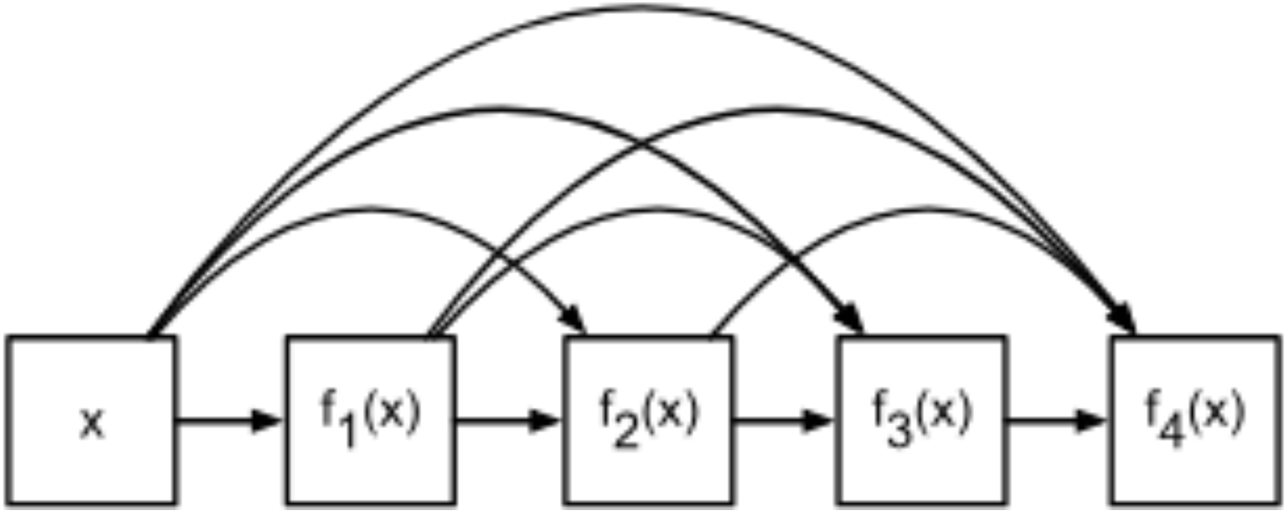
Residual RNNs



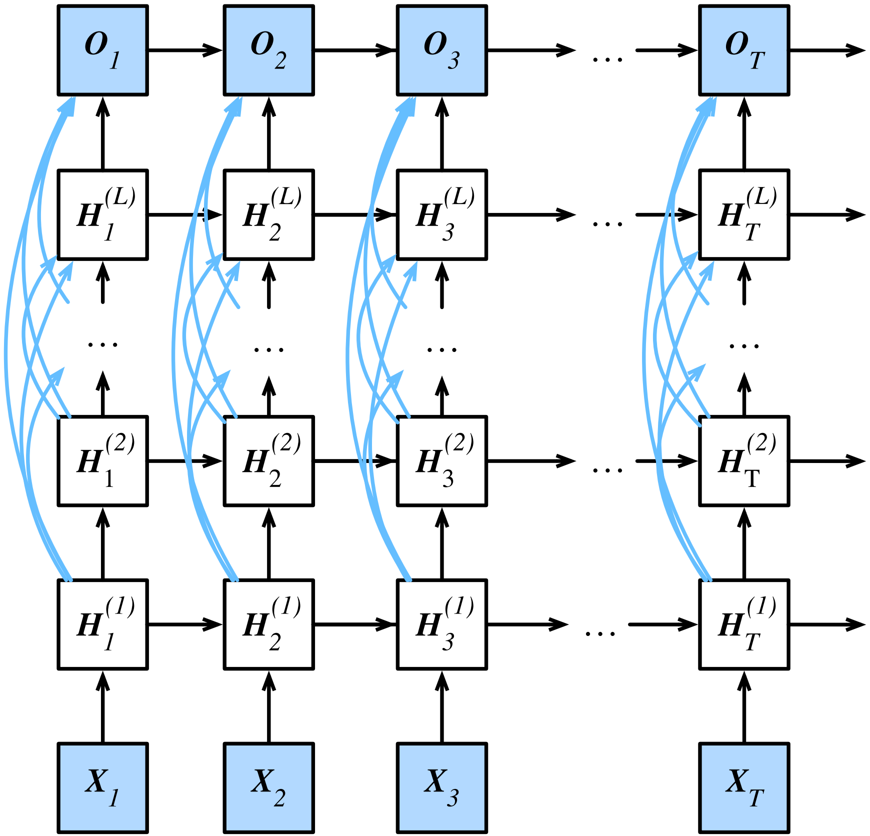
$$\bar{\mathbf{H}}_t^{(2i)} = \mathbf{H}_t^{(2i)} + \mathbf{H}_t^{(2i)-1}$$

- Input of every second layer is also added to its output (residual connection)
- Variants
  - Simple addition
  - Nonlinearity before addition
  - Could also concatenate

What about DenseNet?



RNN with DenseNet Connections



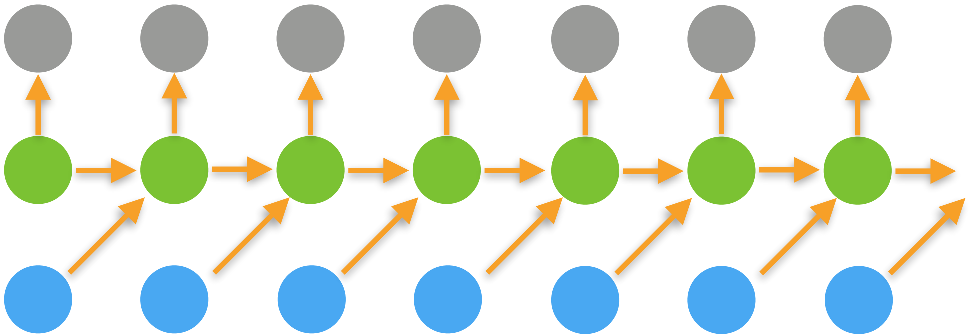
$$\bar{\mathbf{H}}_t^{(t)} = [\mathbf{H}_t^{(t)}, \bar{\mathbf{H}}_t^{t-1}]$$

- Concatenate outputs of previous layers as input top the next layer
- Occasionally add transition layers to reduce dimensionality

Regularization In RNNs

Overfitting

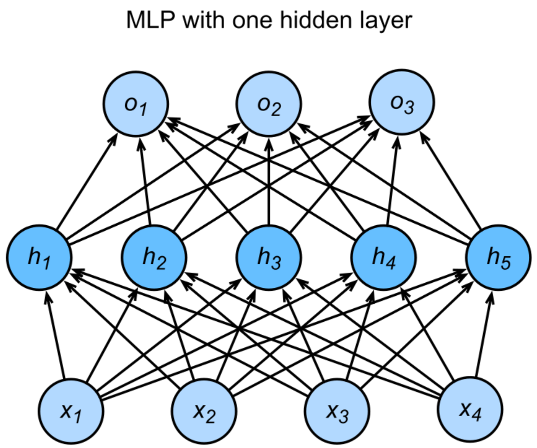
- RNNs overfit just like any other model
- Sequential dependence is more difficult to control
  - Capacity in depth can be controlled, e.g. by dropout
  - For sequential part need to decide how to deal with variable inputs, e.g. input might be skipped)
  - If we use dropout we might miss relevant aspects in the coordinates.

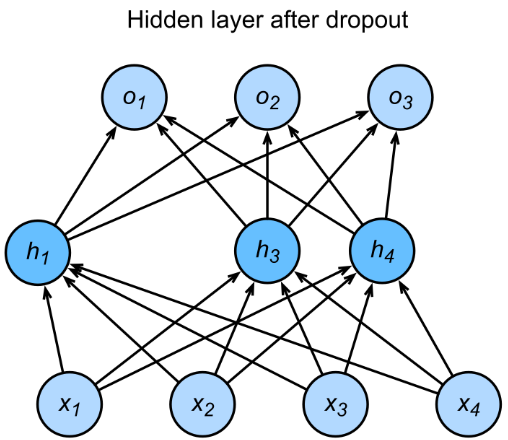


Flashback - Applying Dropout

- Often apply dropout on the output of hidden fully-connected layers

$$\mathbf{h} = \sigma(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)$$
$$\mathbf{h}' = \text{dropout}(\mathbf{h})$$
$$\mathbf{o} = \mathbf{W}_2\mathbf{h}' + \mathbf{b}_2$$
$$\mathbf{y} = \text{softmax}(o)$$

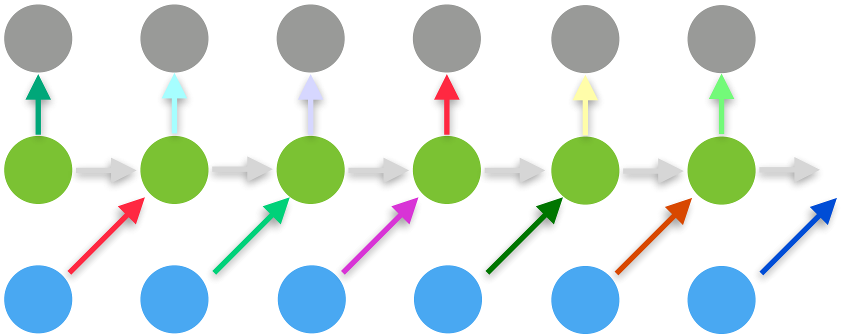


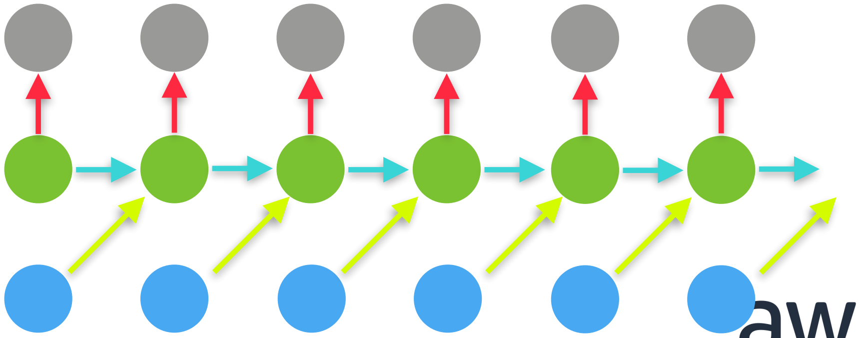


- At **inference** time dropout is inactive, i.e. $\mathbf{h}' = \text{dropout}(\mathbf{h})$

Variational Dropout (Gal & Ghahramani, 2015)

- Regular Dropout
  - Use only per time slice
  - Different mask per slice
- Same mask across all time windows
- Use also for temporal aspect



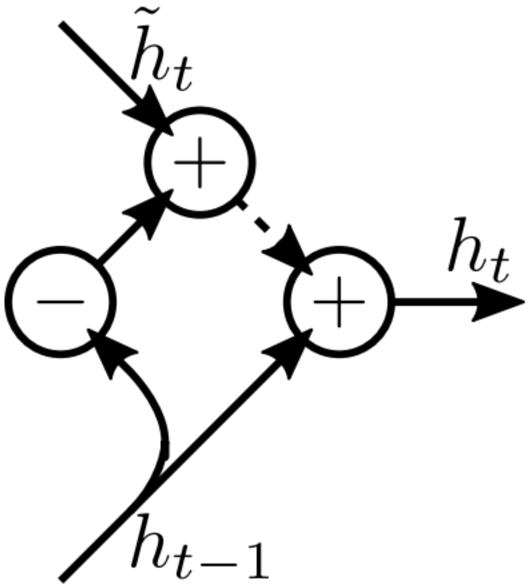


Zoneout (Krueger et al., 2016)

- Robustness against skipping observations in sequence
- Robustness of state representation relative to hidden state updates

- Skip hidden state update and keep the same as previously during training

$$h_t = h_{t-1}$$



Many more tricks

- Parameter averaging (Merity et al., 2017)
  Train RNN and average weights over run
- Stochastic Weight Averaging (Wilson et al., 2018)
  Same approach but keep on changing learning rate
- Fraternal Dropout (Zolna et al., 2017)
  Dropout while minimizing variation between outputs to increase robustness to parametrization

…