- Rnn's use cycles to transmit info back to themselves, like a sort of loop

Formulas and notation:

n = # of samples in a batch

d = # of inputs in each sample

h = # of hidden units (dimensions of hidden state)

- **Hidden state at a time step t** : $H_t \in \mathbb{R}^{n \times h}$
- **Input at a time step t** : $X_t \in \mathbb{R}^{n \times d}$
- **weight matrix** : $W_{xh} \in \mathbb{R}^{d \times h}$
- **hidden-state-to-hidden-state matrix** : $W_{hh} \in \mathbb{R}^{h \times h}$
- **bias parameter** : $b_h \in \mathbb{R}^{1 \times h}$
- **activation function** : ϕ

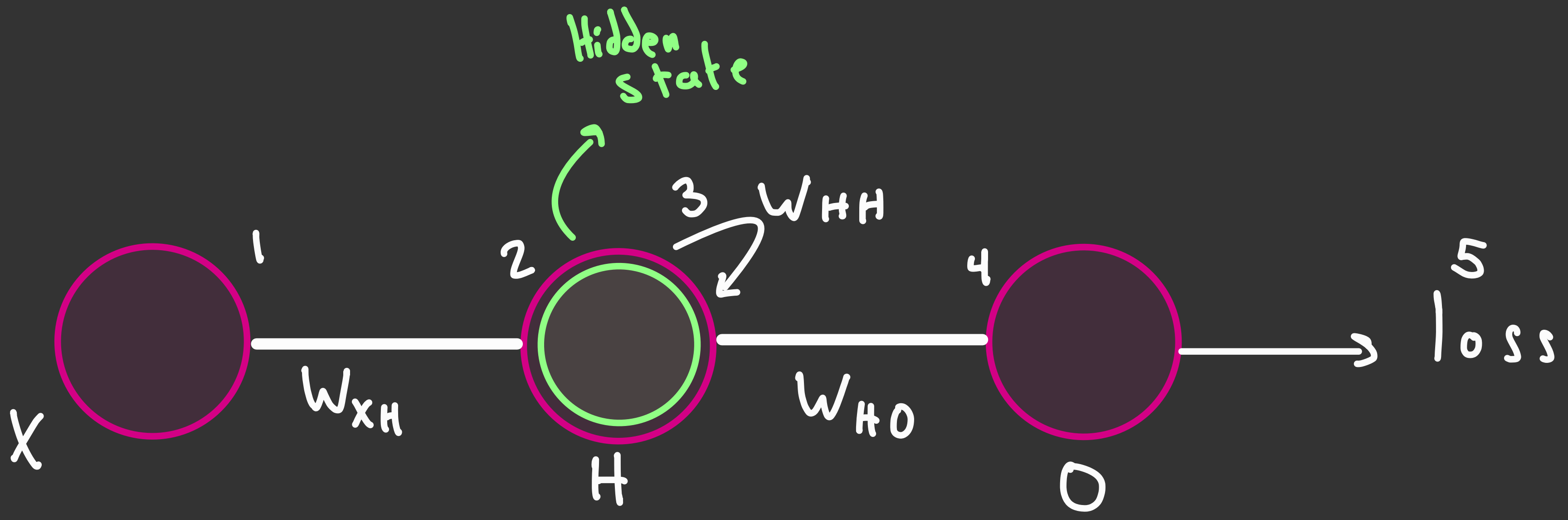
- **Equation 1 (hidden variable)**

$$H_t = \phi_h (X_t W_{xh} + H_{t-1} W_{hh} + b_h)$$

- **Equation 2 (output variable)**

$$O_t - \phi_o (H_t W_o + b_o)$$

Simple representation

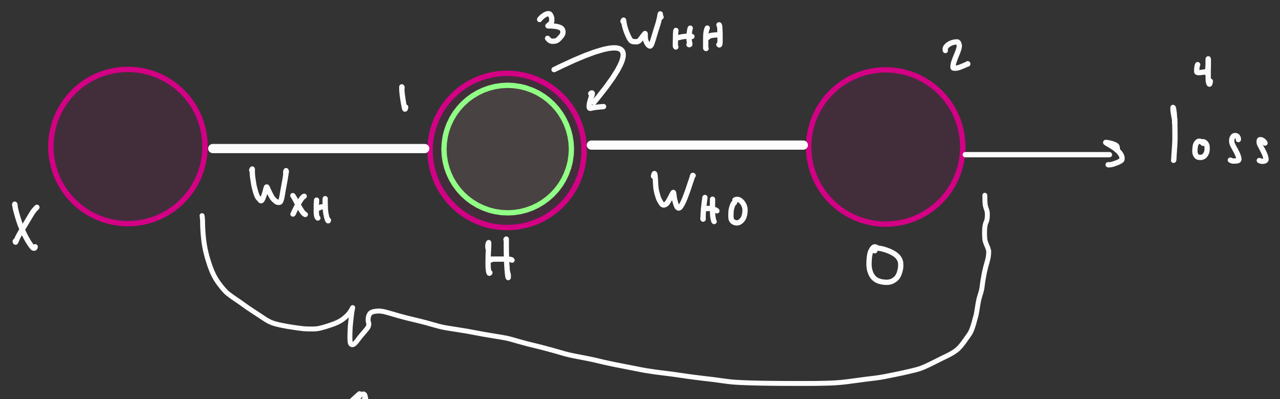


Practical example: predict next number

input seq: 1, 3, 5, 7, 9

tanh activation function

SGD optimizer



Input processing ↷

Input 1 (1)

1 - Calculate hidden state H , combine H (initialized as 0)

$H_1 = \tanh(1 \cdot W_{xh} + H_0 \cdot W_{hh} + b_h)$

2 - Calculate output

$O_1 = H_1 \cdot W_{ho} + b_o$

3 - Store hidden state H to be used for next input

Input 2 (3)

1 - Calculate hidden state H , combine H

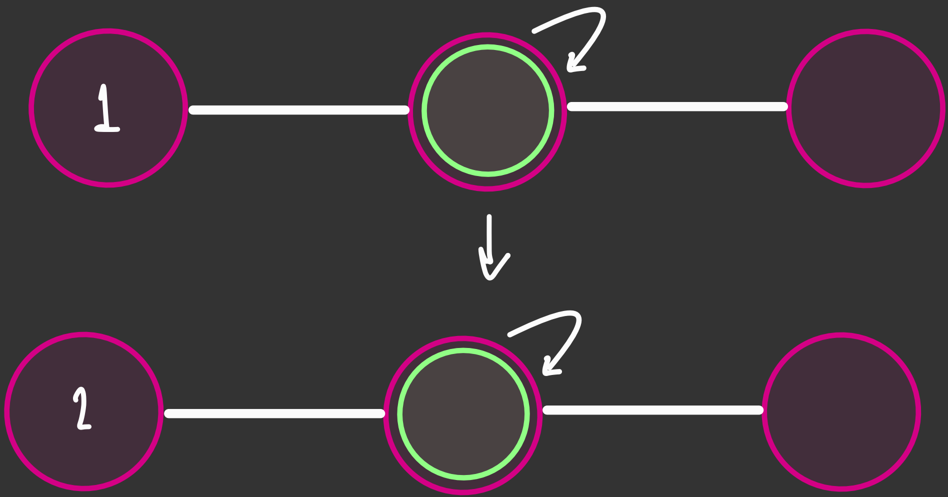
$H_2 = \tanh(3 \cdot W_{xh} + H_1 \cdot W_{hh} + b_h)$

2 - Calculate output

$O_2 = H_2 \cdot W_{ho} + b_o$

3 - Store hidden state H to be used for next input

What's happening in this stage?



The network processes each input individually, and at each time step, what is in the hidden state gets updated so it contains information about all inputs.

The generated outputs will be stored but for our use case, we will concentrate on the final output.

This is dependent on usage and another context-depending task like translation would use all outputs.

Input 3 (5)

1 - Calculate hidden state H , combine H

$H_3 = \tanh(5 \cdot W_{xh} + H_2 \cdot W_{hh} + b_h)$

2 - Calculate output

$O_3 = H_3 \cdot W_{ho} + b_o$

3 - Store hidden state H to be used for next input

Input 4 (7)

1 - Calculate hidden state H , combine H

$H_4 = \tanh(7 \cdot W_{xh} + H_3 \cdot W_{hh} + b_h)$

2 - Calculate output

$O_4 = H_4 \cdot W_{ho} + b_o$

3 - Store hidden state H to be used for next input

Input 5 (9)

1 - Calculate hidden state H , combine H

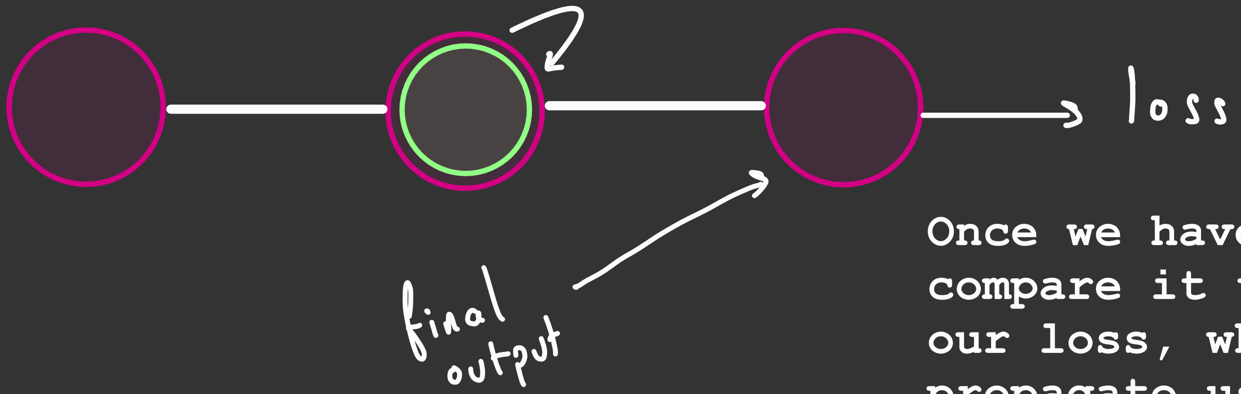
$H_5 = \tanh(9 \cdot W_{xh} + H_4 \cdot W_{hh} + b_h)$

2 - Calculate output

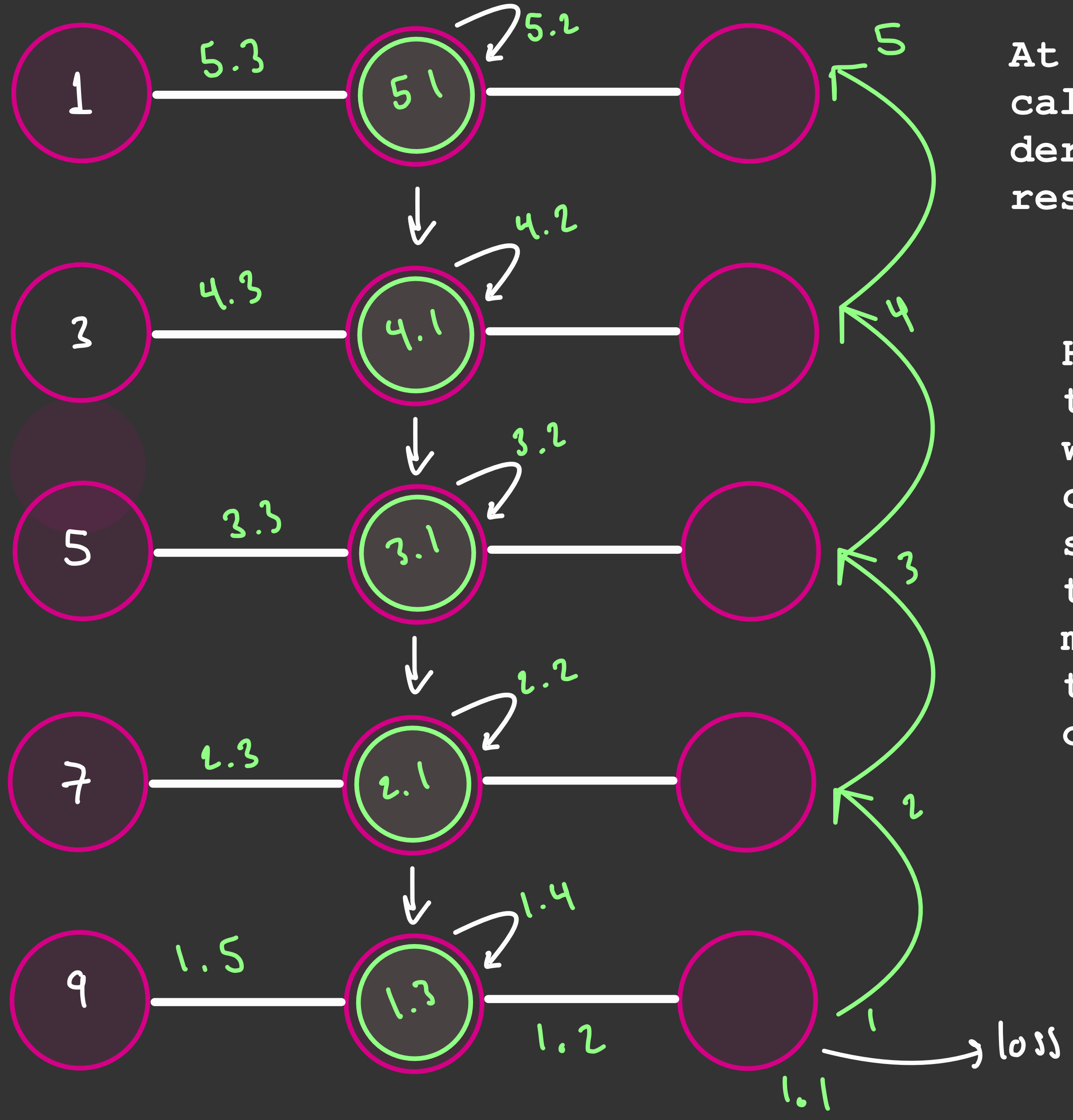
$O_5 = H_5 \cdot W_{ho} + b_o$

3 - Store hidden state H to be used for next input

What happens after all inputs are processed?



Once we have generated a final output, we compare it to the target, and calculate our loss, which we will then back propagate using BPTT (calculation at each time step)



At each time step t, BPTT calculates gradients (partial derivatives) of the loss with respect to each weight in the RNN

Here, we once again depend on the task to determine in we will use all outputs in the calculations. In this case, since we only concentrate on the last output, we won't which means we also won't calculate the gradient of W_{ho} for all other time steps.

#MATH PROCESS#

Chain rule and whatnot, perchance

Last, we will use our optimizer in order to update the weights and biases.