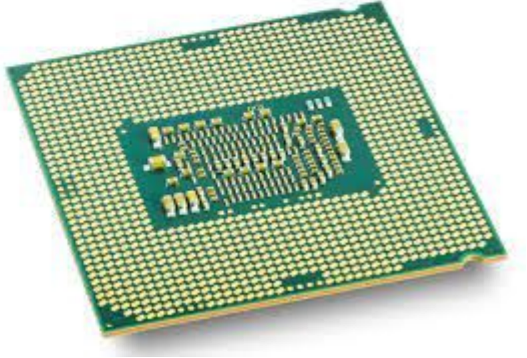
DATA 442: Neural Networks & Deep Learning

Dan Runfola – danr@wm.edu

icss.wm.edu/data442/



CPU

Only a Few Cores
(Counted in Hundreds, at most, and normally in ten or less).

Very fast clock speeds
(4GHz +)

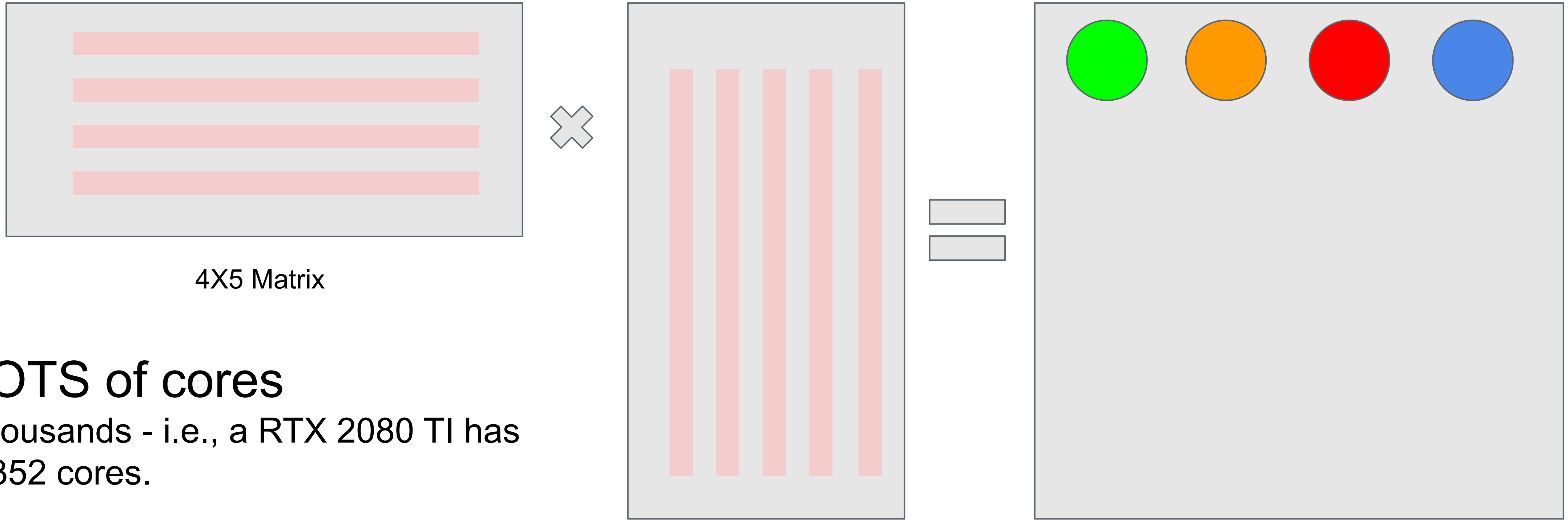
Uses Physical Memory located Elsewhere on Motherboard

GPU

LOTS of cores
Thousands - i.e., a RTX 2080 TI has 4,352 cores.

Slower clock speeds
(1-2 GHz)

Physical memory is integrated with the card.



4X5 Matrix

5x5 Matrix

4x5 Matrix

LOTS of cores

Thousands - i.e., a RTX 2080 TI has 4,352 cores.

Slower clock speeds (1-2 GHz)

Physical memory is integrated with the card.

Low-level GPU Programming

Comparison:
https://arxiv.org/vc/arxiv/papers/1005/1005.2581v1.pdf

Deep Learning Frameworks

Majors Players:

Torch / PyTorch (Facebook)

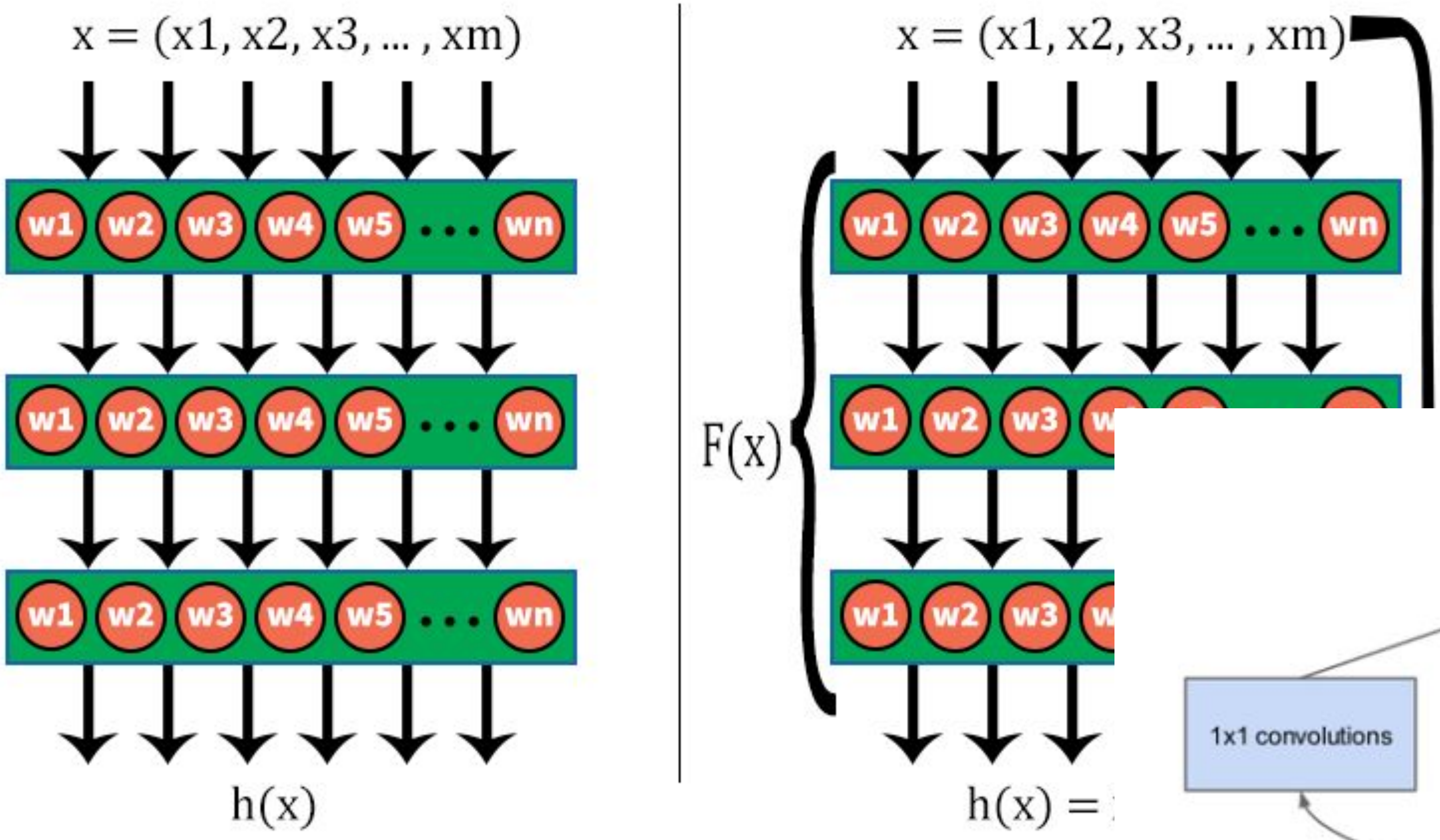
TensorFlow / Keras (Google)

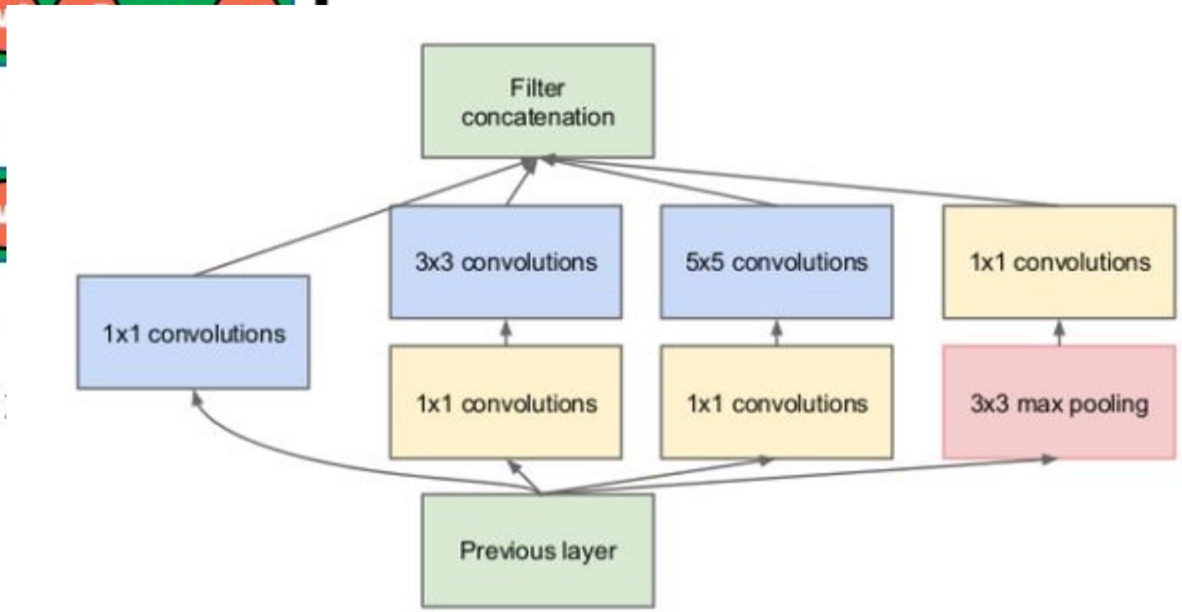
Old / Less Used / Integrated / Non-English:

Caffe (UC Berkeley); Theano (U Montreal); Caffe 2 (Facebook); PaddlePaddle (Baidu); CNTK (MSFT); MXNET (Amazon, MIT, CMU)

https://www.paddlepaddle.org.cn/

Standard DNN | Residual Block





Speed

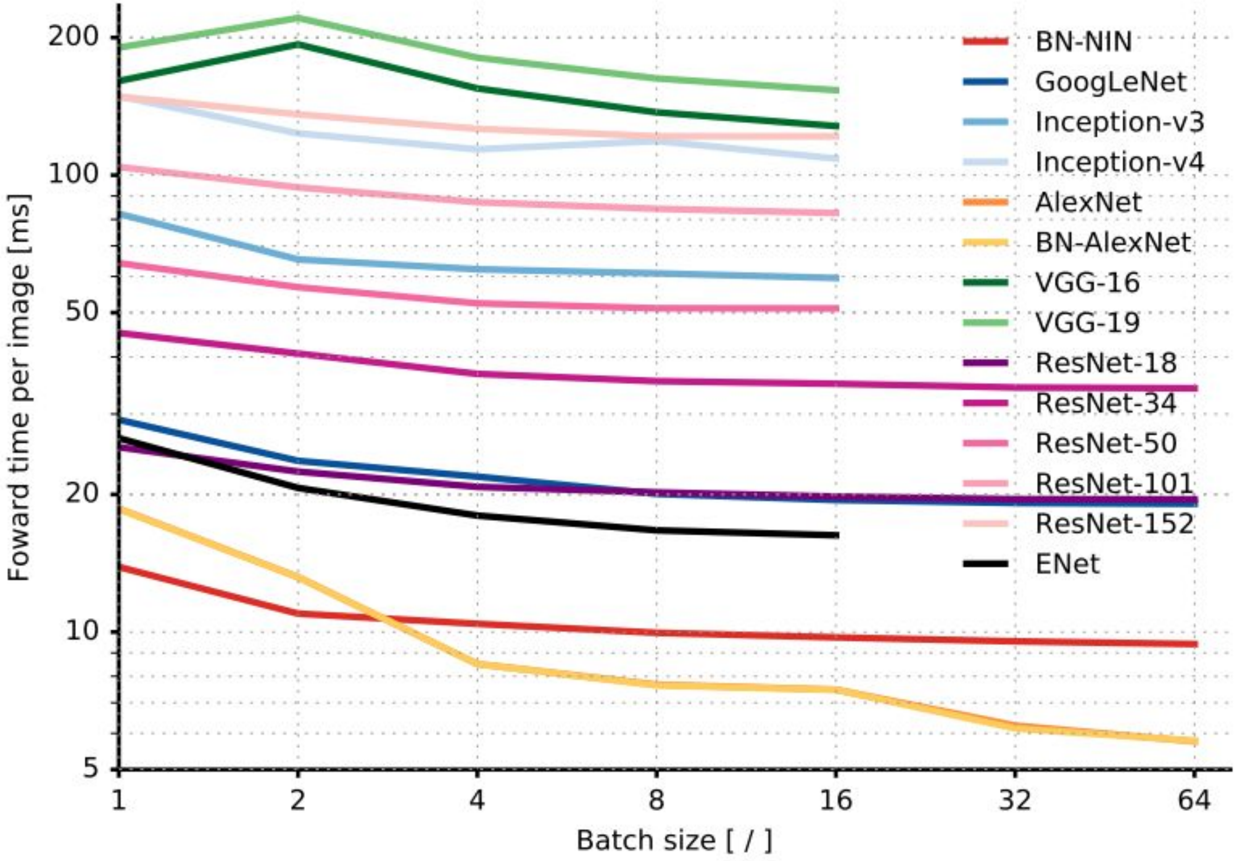
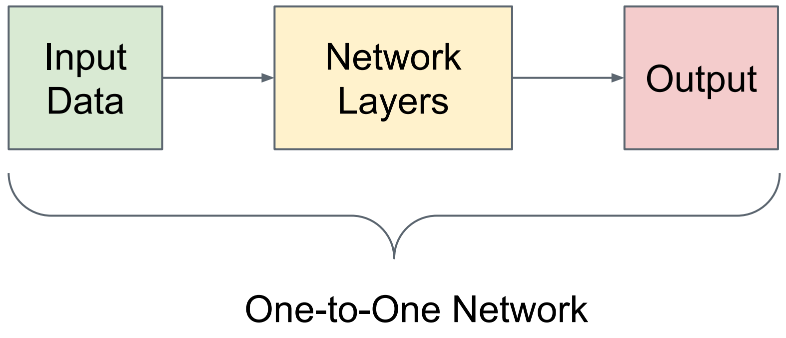


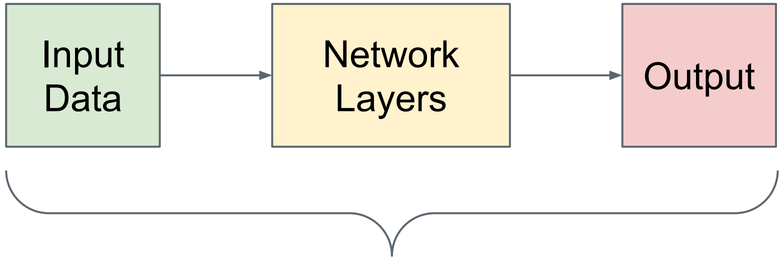
Figure 3: **Inference time *vs.* batch size.** This chart show inference time across different batch sizes with a logarithmic ordinate and logarithmic abscissa. Missing data points are due to lack of enough system memory required to process larger batches. A speed up of 3× is achieved by AlexNet due to better optimisation of its fully connected layers for larger batches.

https://arxiv.org/pdf/1605.07678.pdf

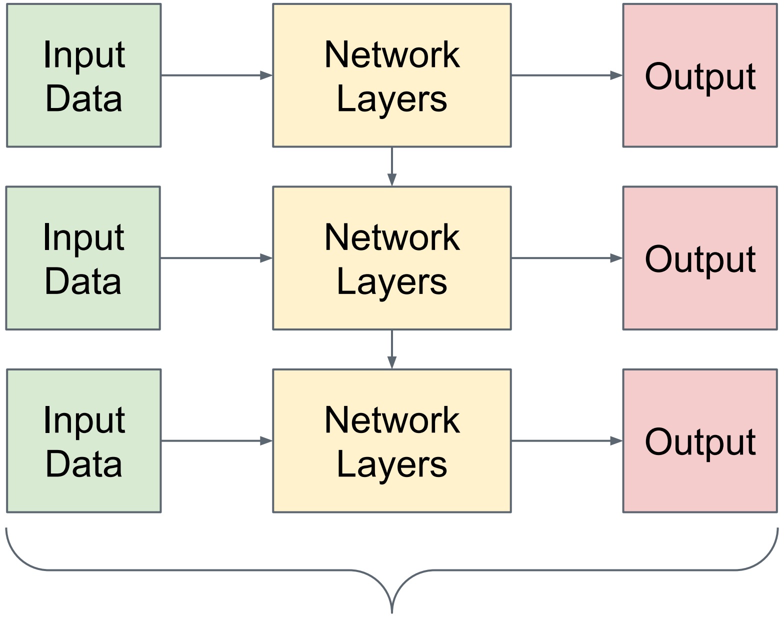
Recurrent Neural Networks



Recurrent Neural Networks

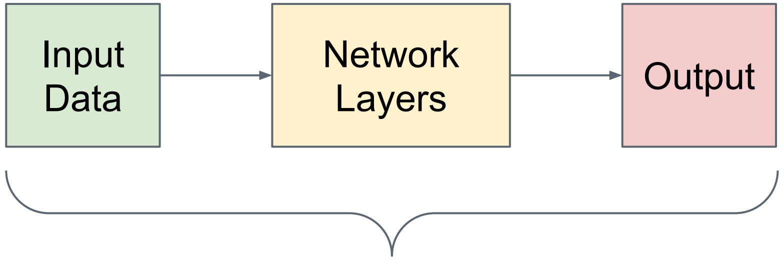


One-to-One Network

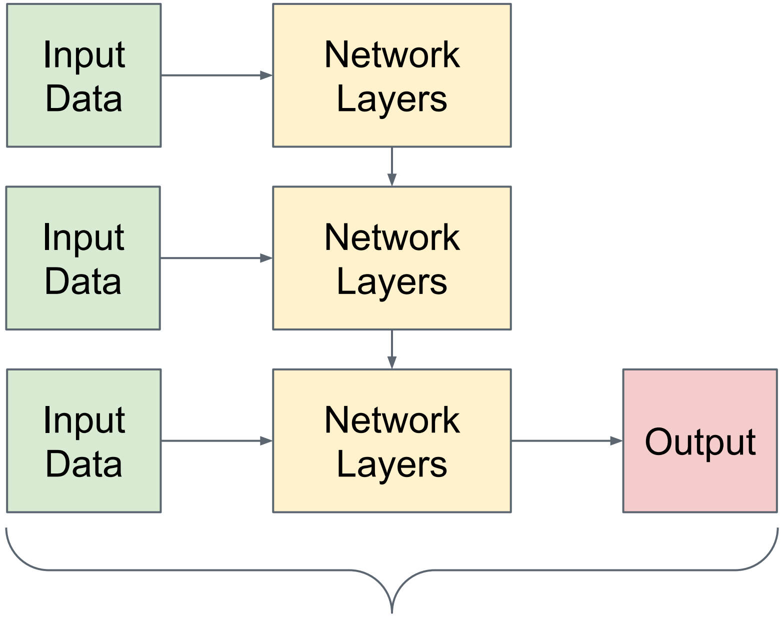


Many-to-Many Network

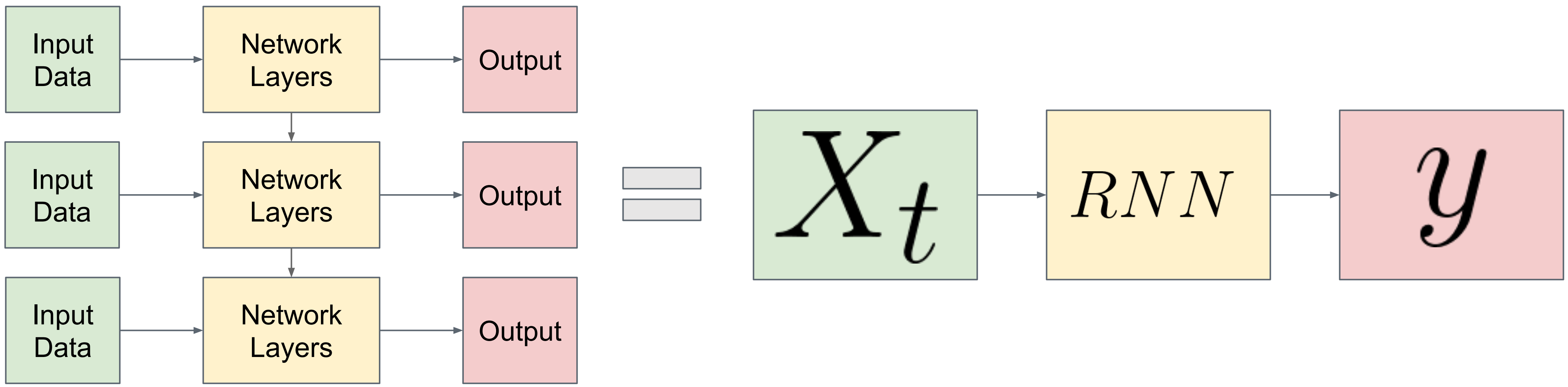
Recurrent Neural Networks

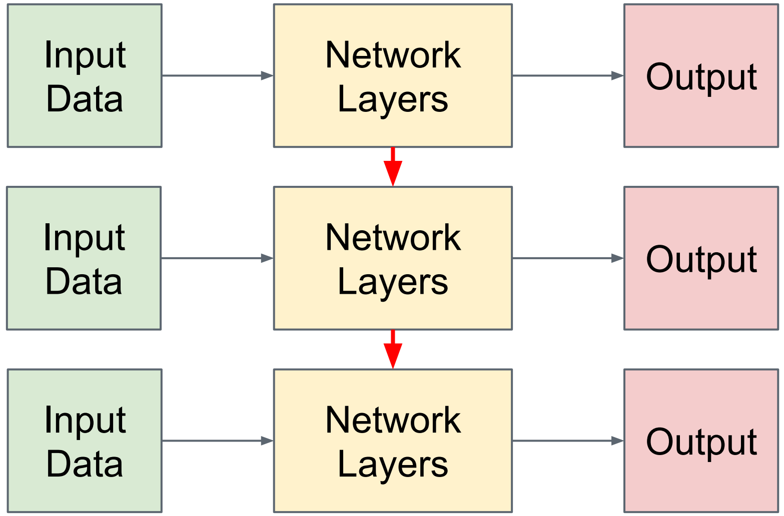


One-to-One Network

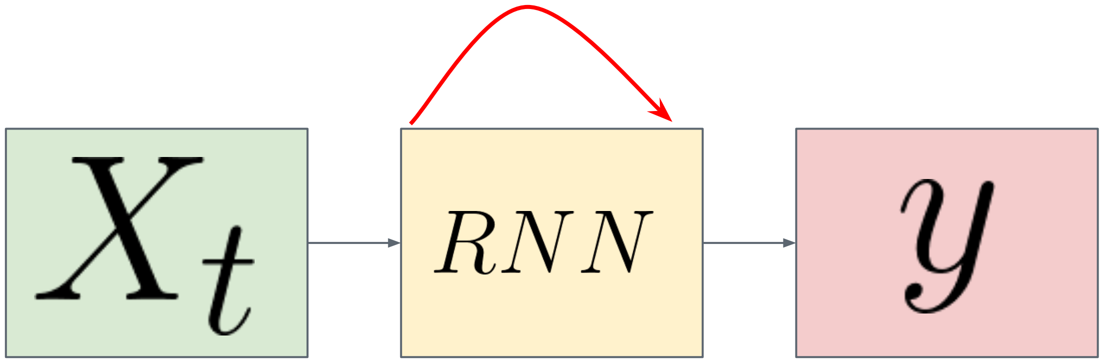


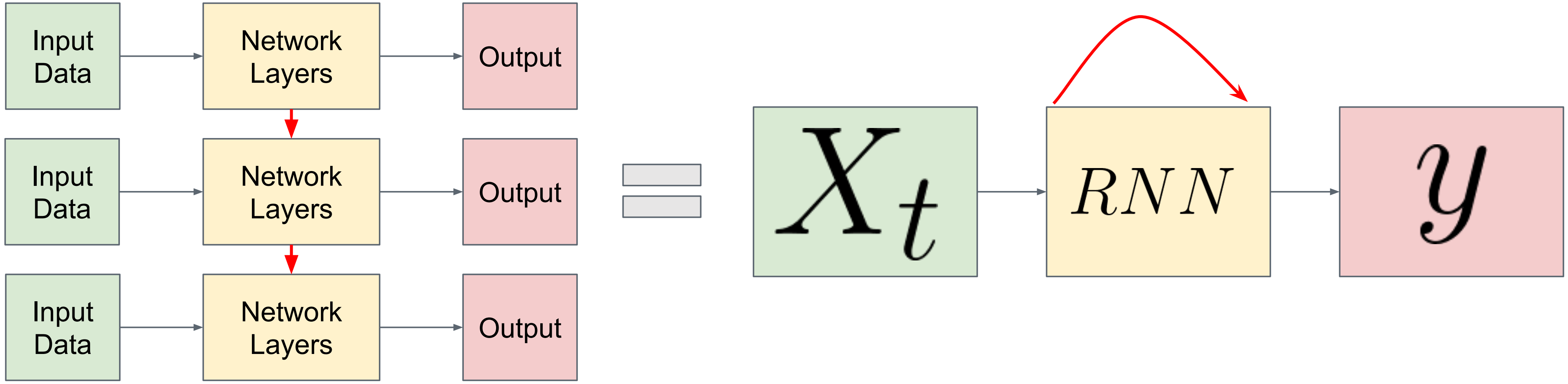
Many-to-One Network


Input Data
Network Layers
Output
Input Data
Network Layers
Output
Input Data
Network Layers
Output
X_t
RNN
y

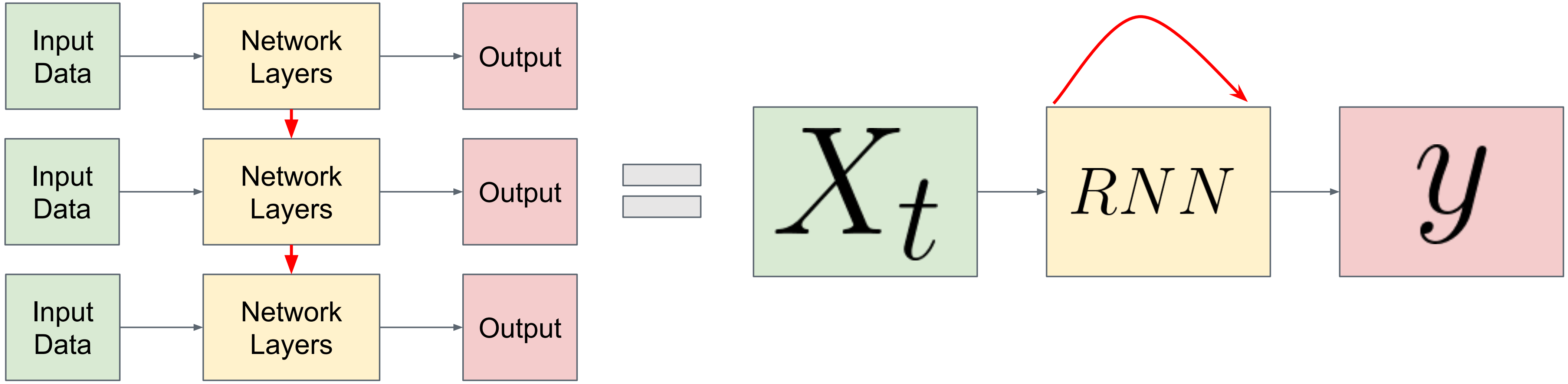

Input Data
Network Layers
Output
Input Data
Network Layers
Output
Input Data
Network Layers
Output

=

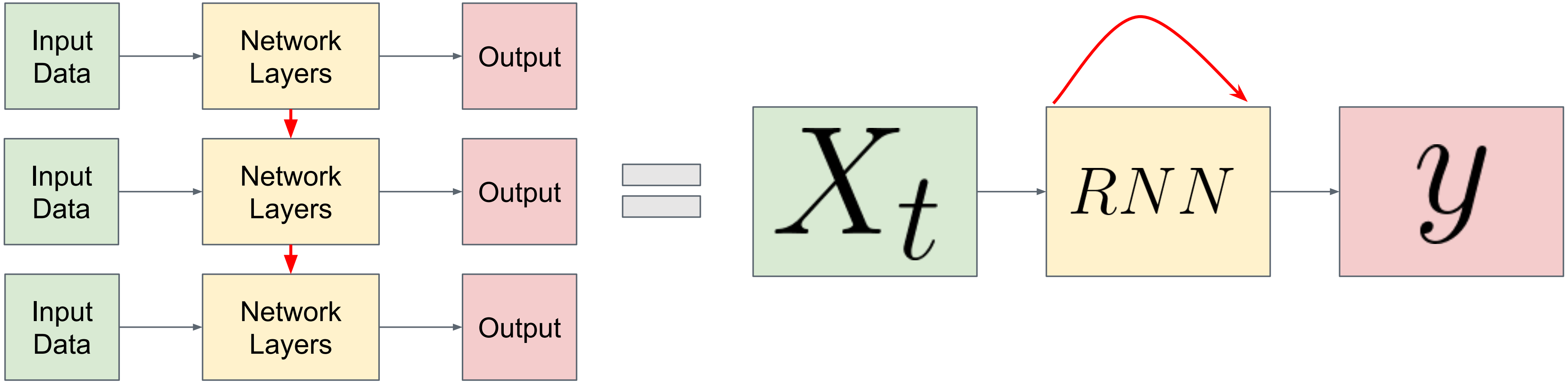

X_t
RNN
y



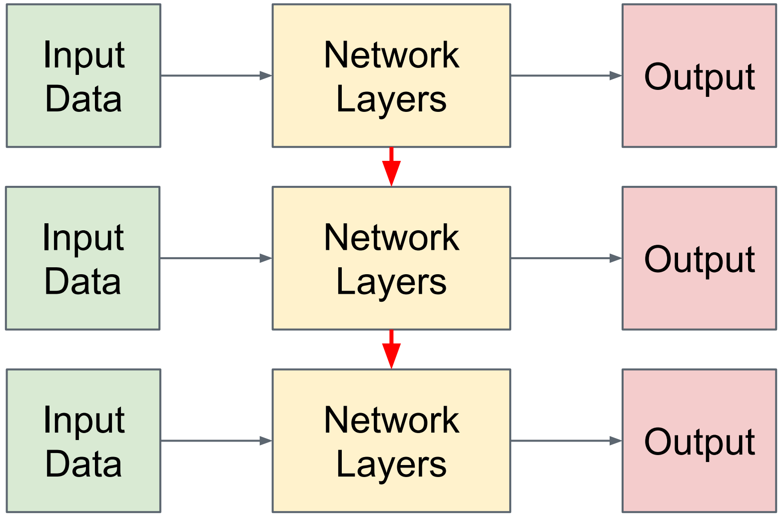
https://www.youtube.com/watch?v=_7SVJGTjUMI



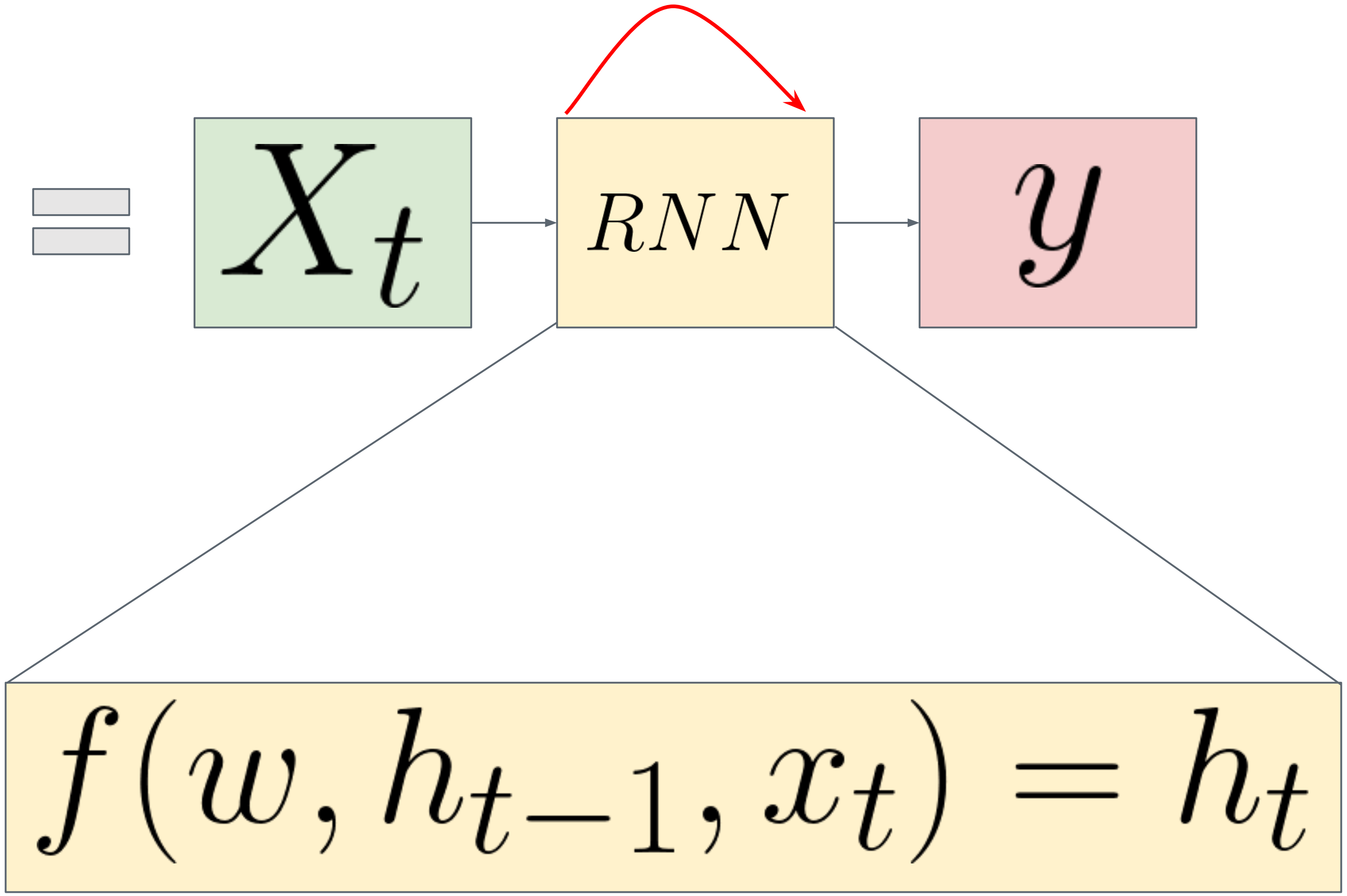
https://www.youtube.com/watch?v=_7SVJGTjUMI



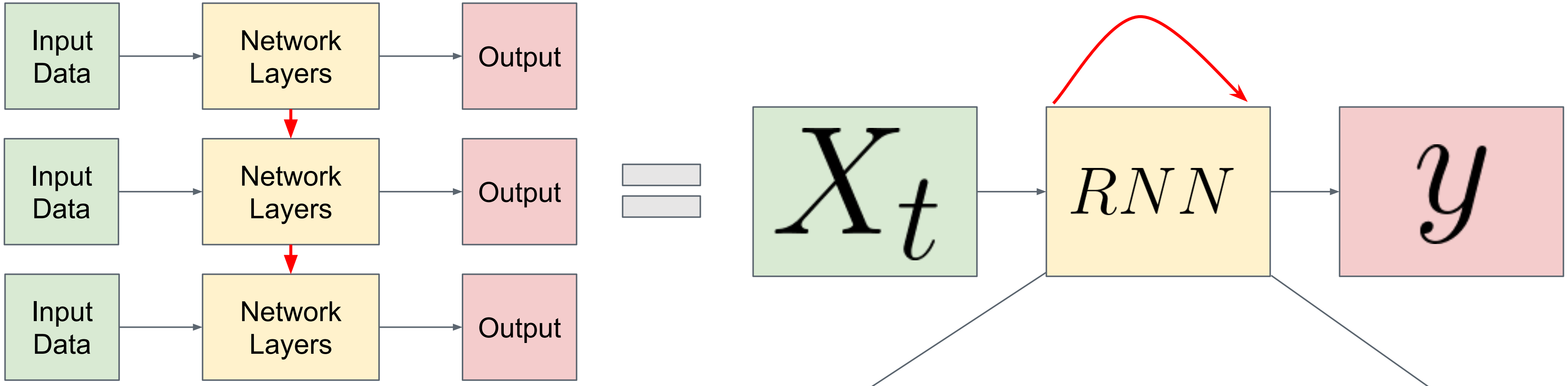
https://www.youtube.com/watch?v=_7SVJGTjUMI



=



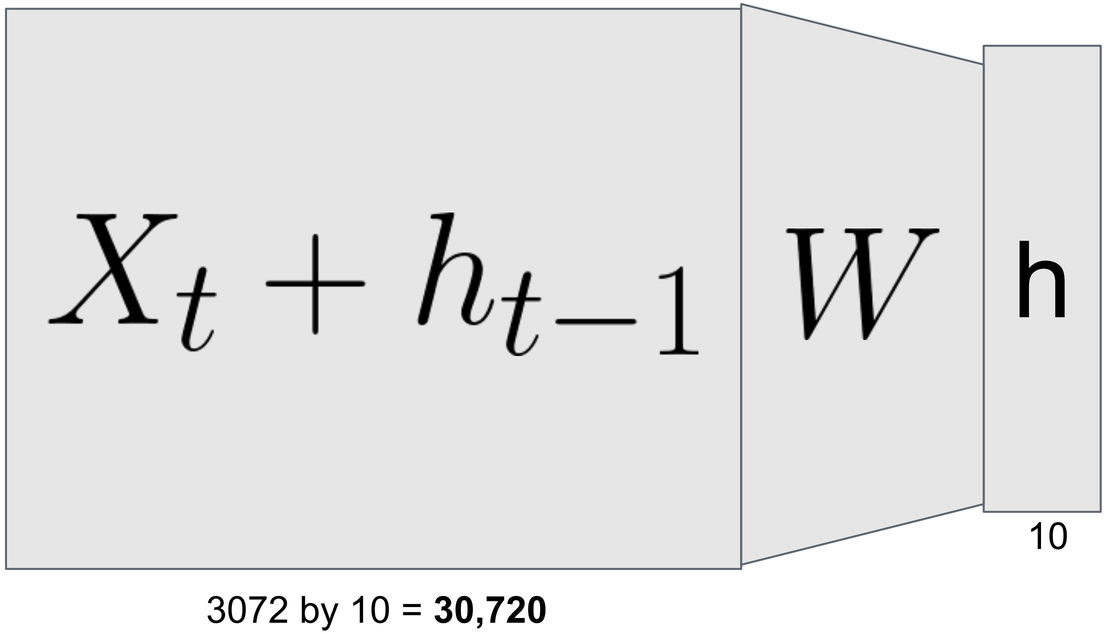
https://www.youtube.com/watch?v=_7SVJGTjUMI



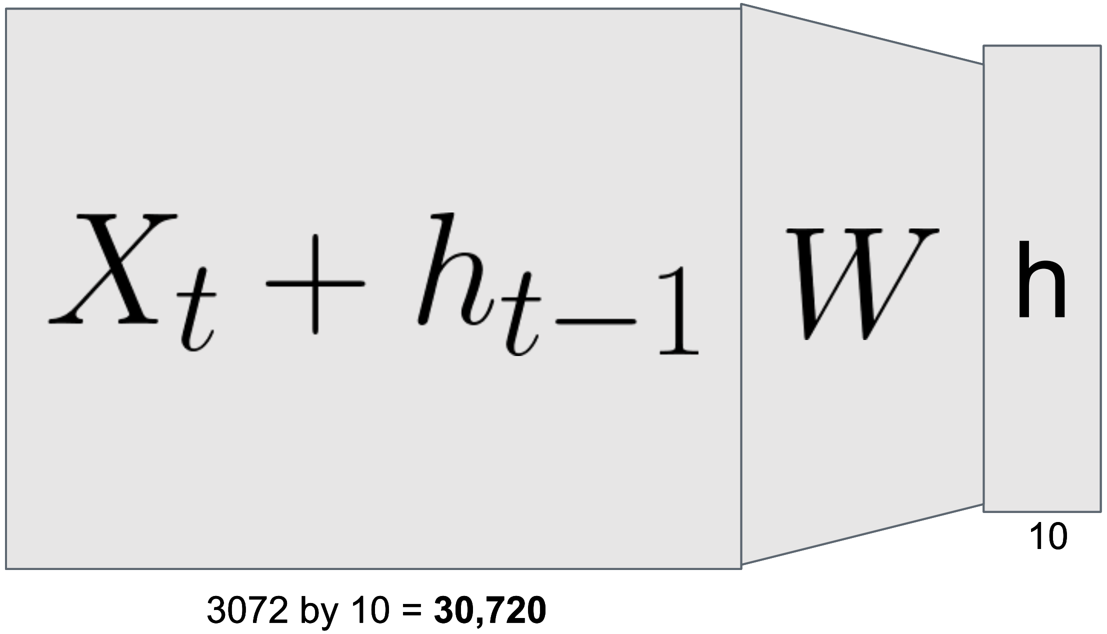
$$f(w, \boxed{h_{t-1}}, x_t) = h_t$$

https://www.youtube.com/watch?v=_7SVJGTjUMI

Unpacking a RNN



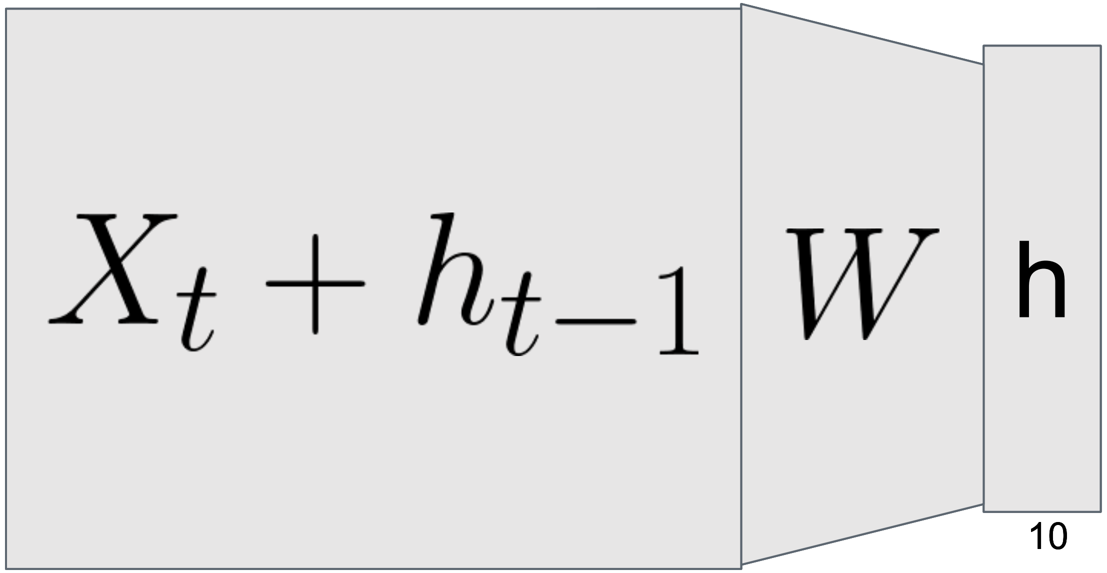
Unpacking a RNN



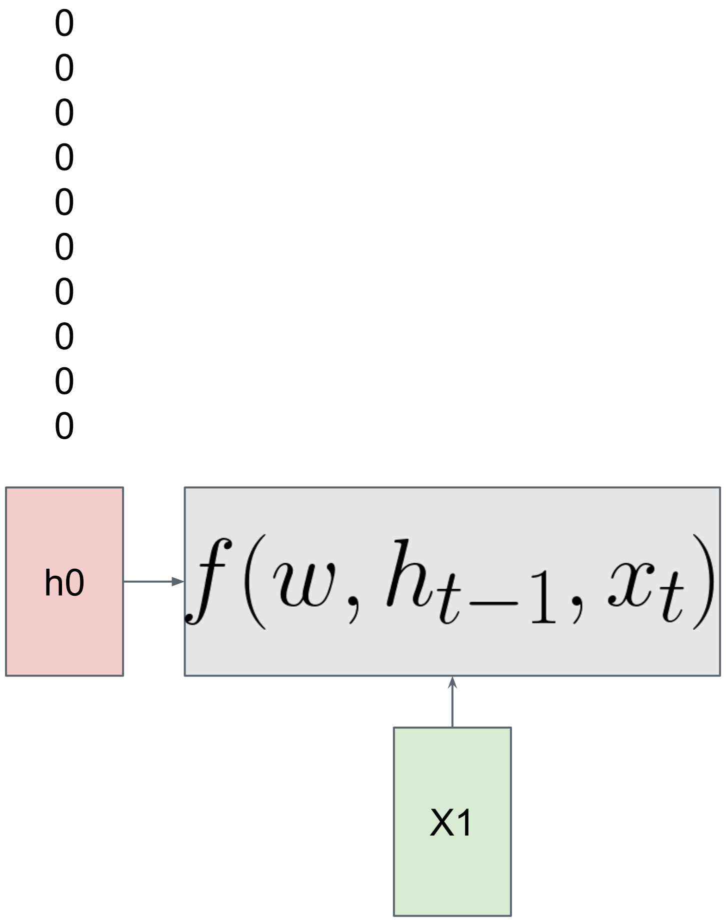
0
0
0
0
0
0
0
0
0
0

h0

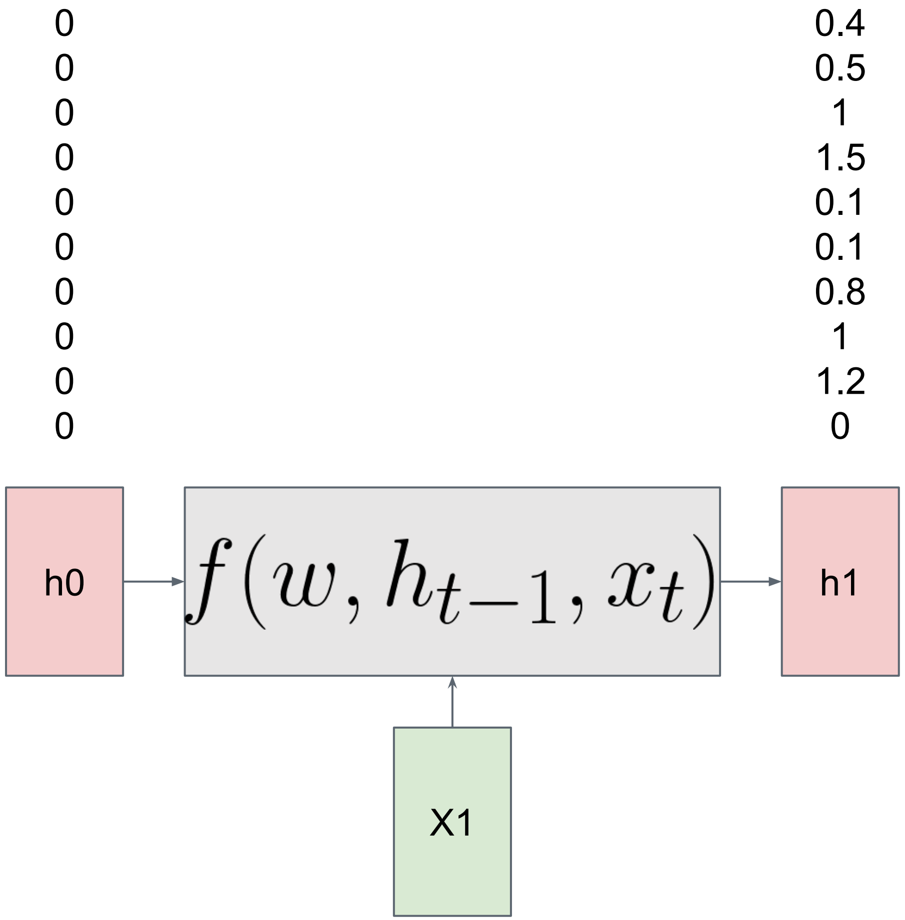
Unpacking a RNN



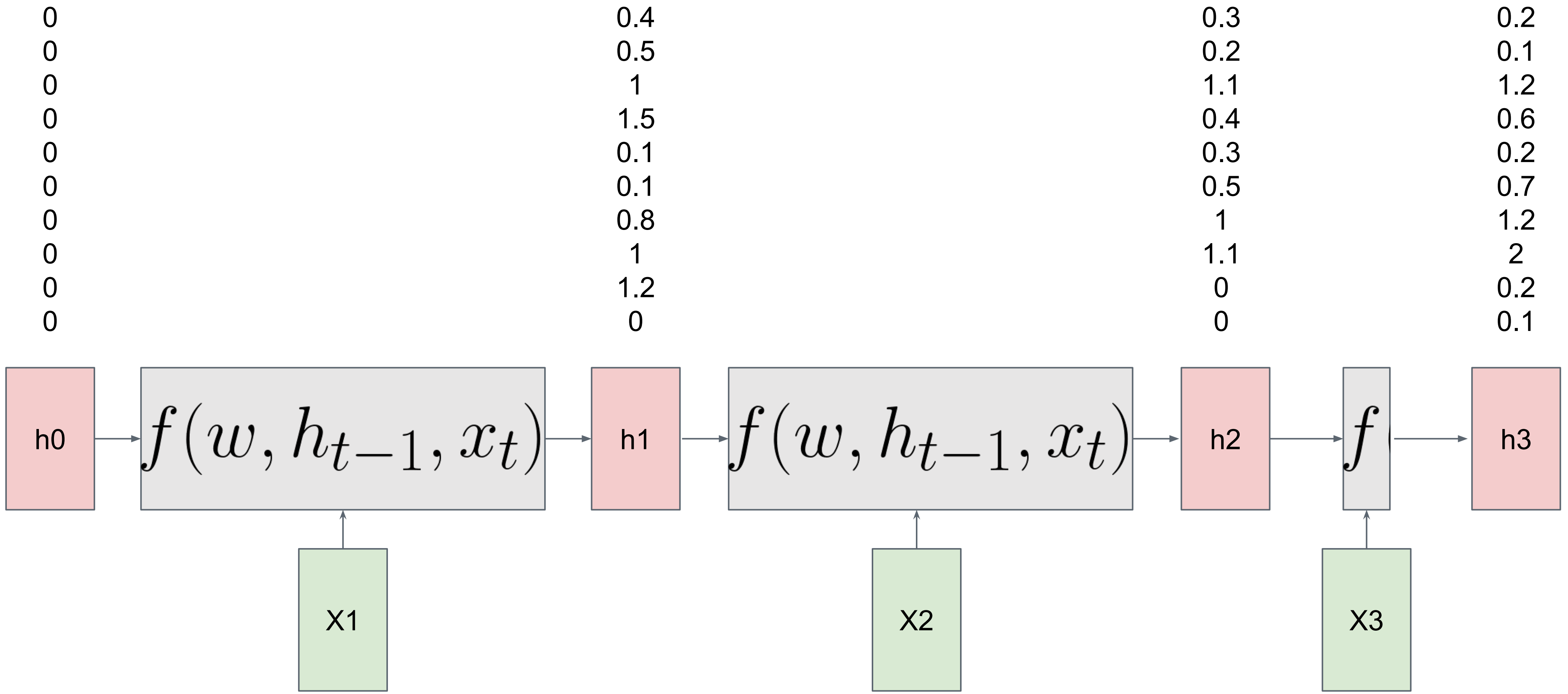
3072 by 10 = **30,720**



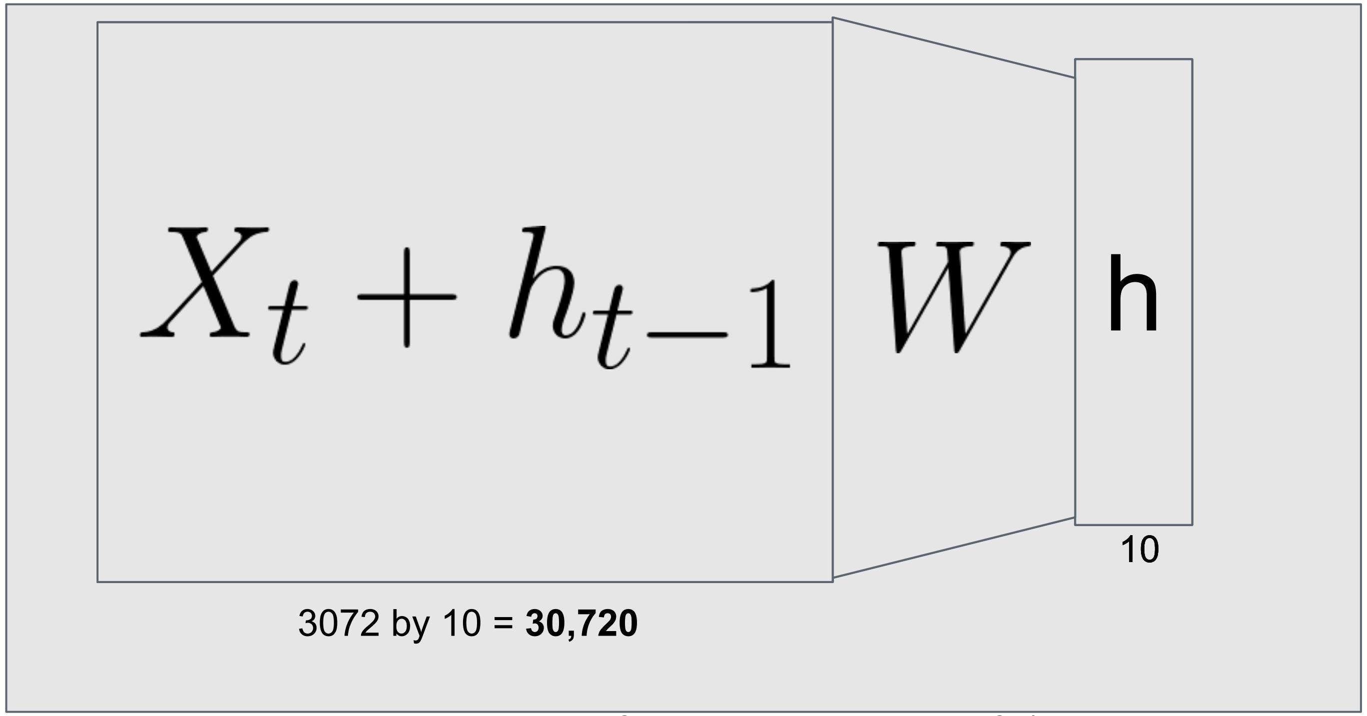
Unpacking a RNN

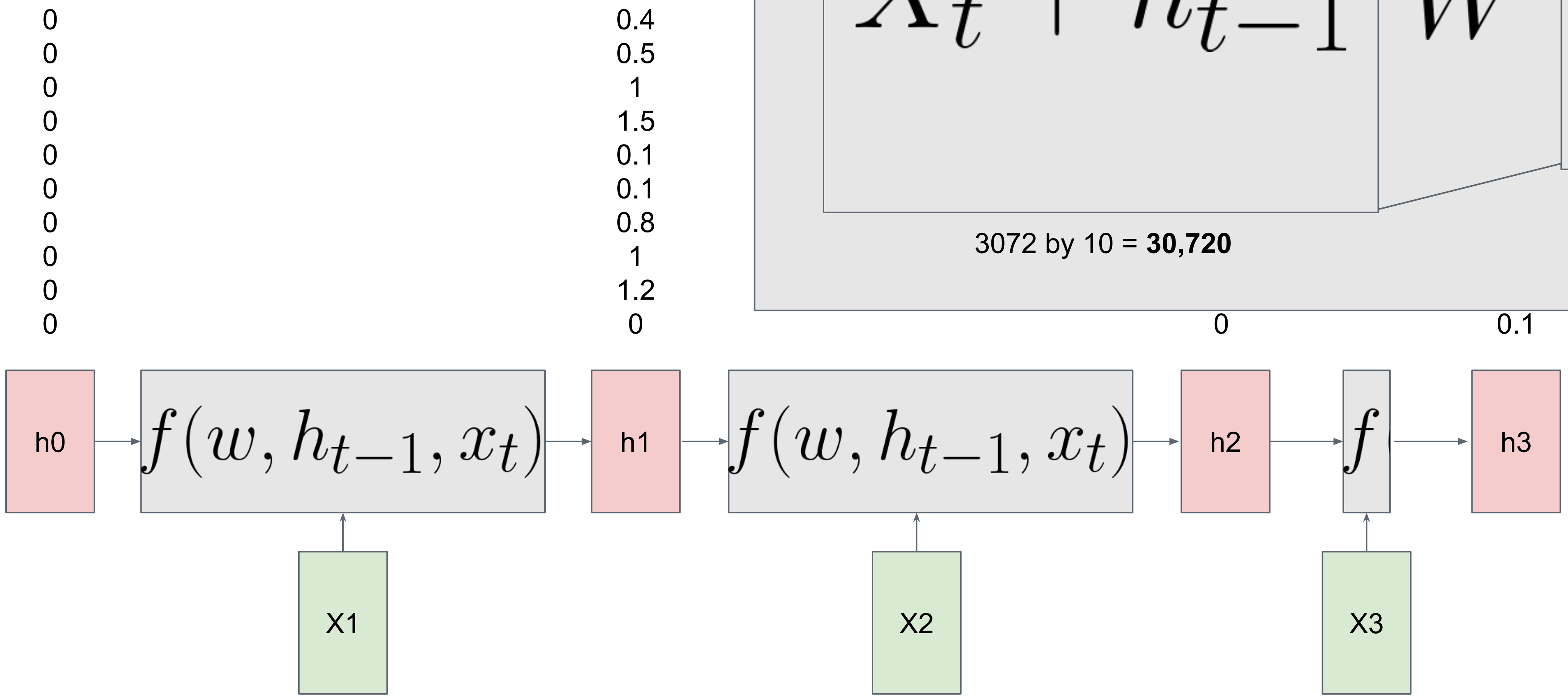


Unpacking a RNN

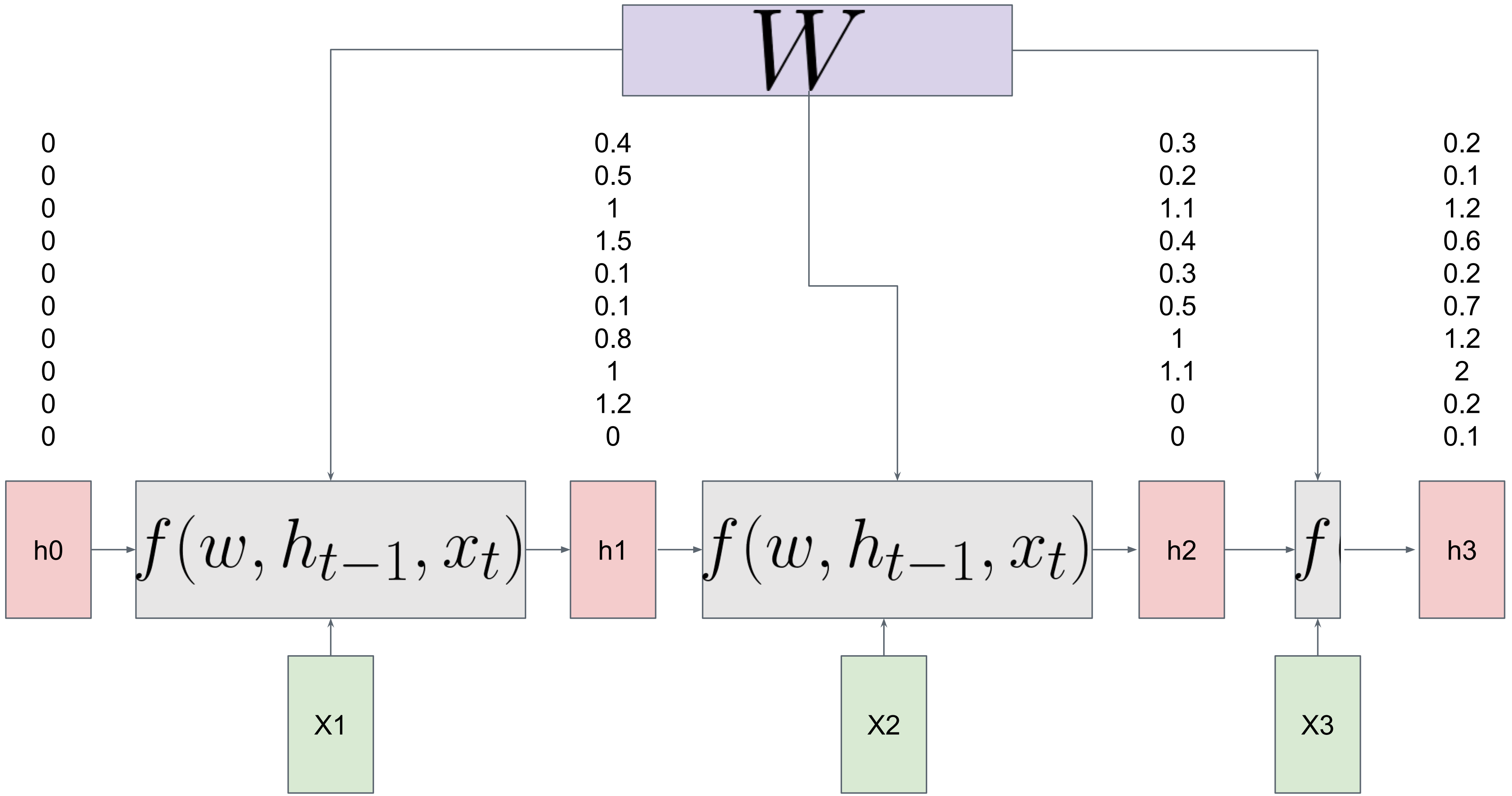


Unpacking a RNN

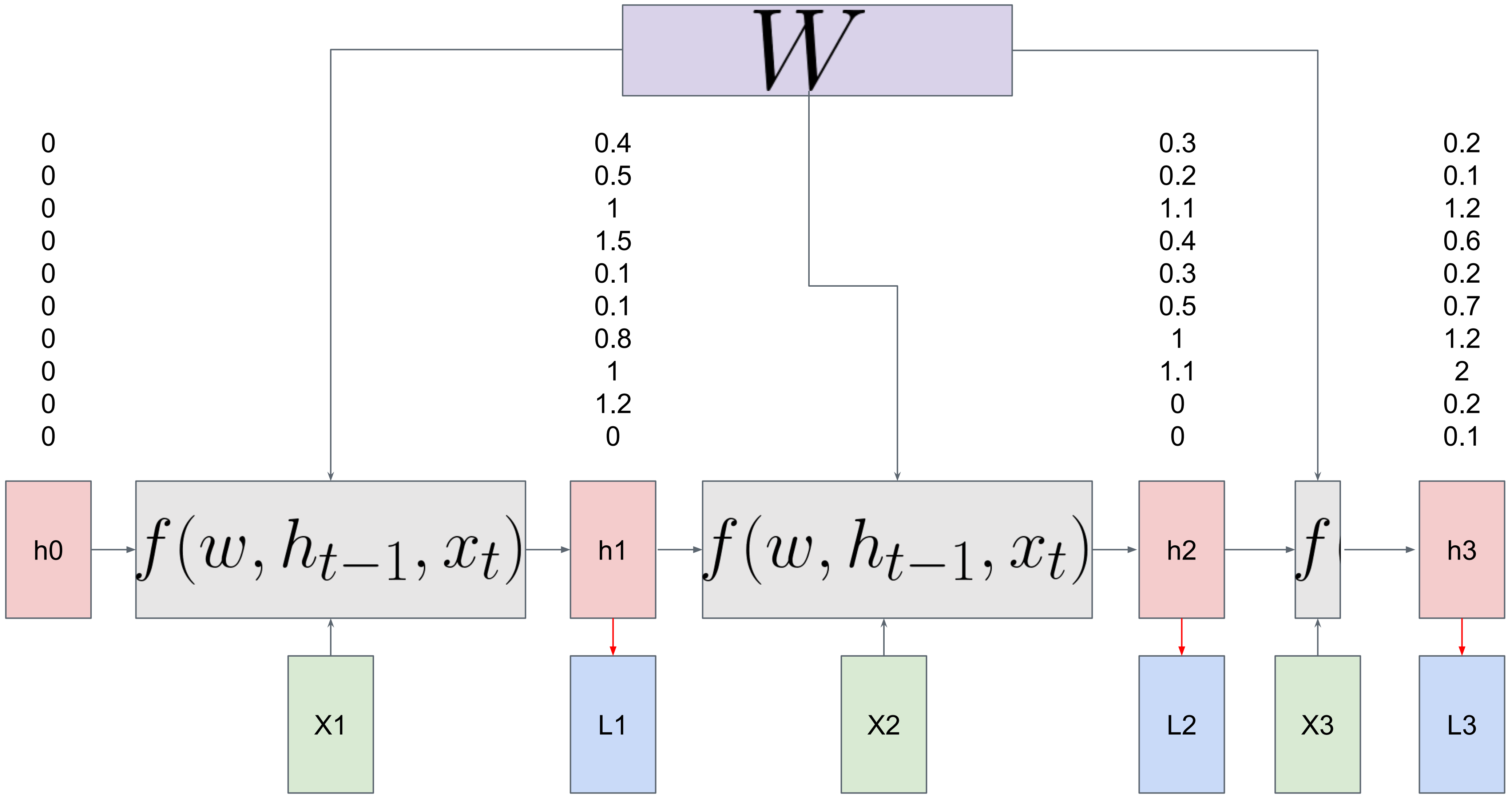




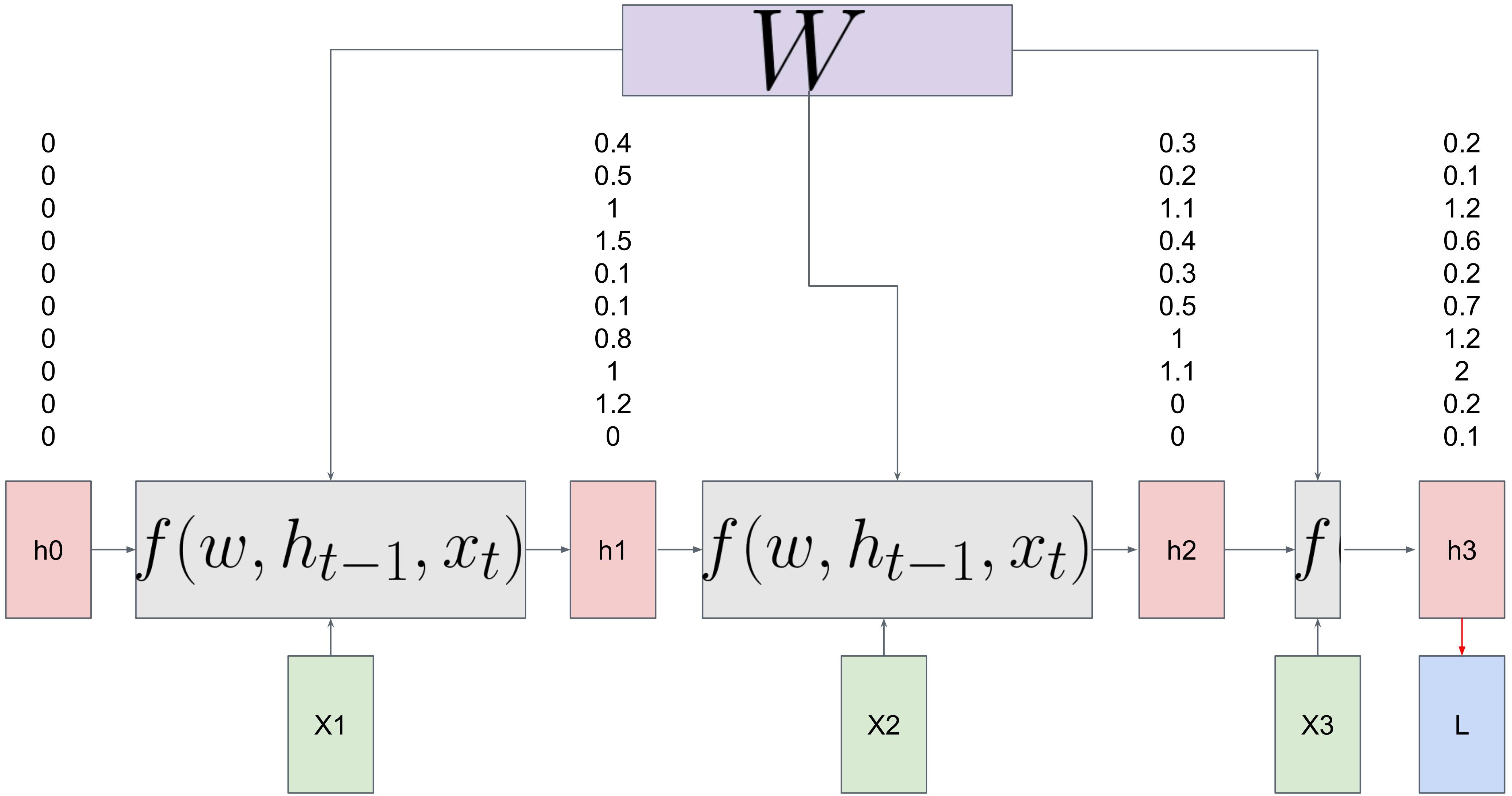
Unpacking a RNN



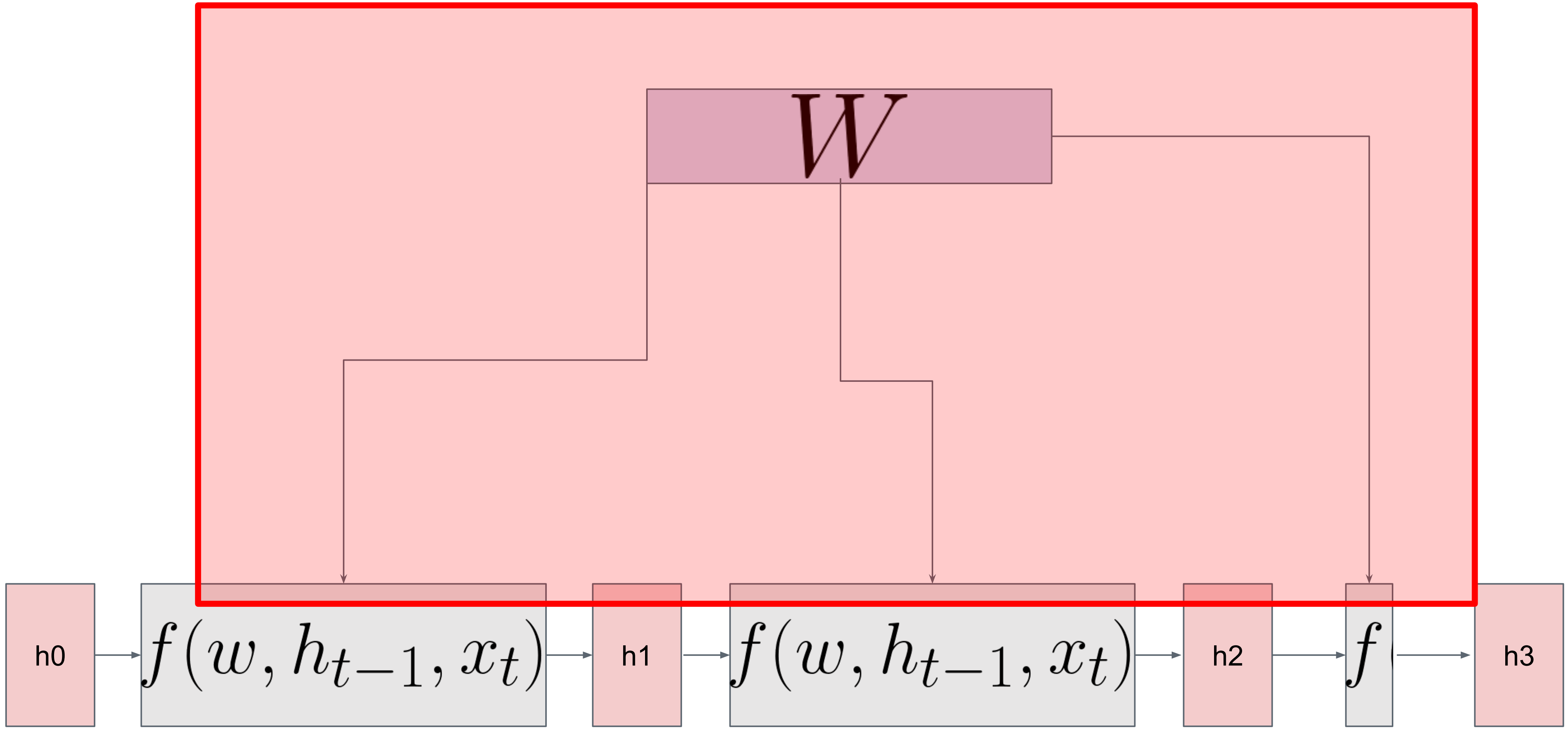
Unpacking a RNN - Many to Many



Unpacking a RNN - Many to One



Flaw in a Vanilla RNN



Long short term memory network

LSTM

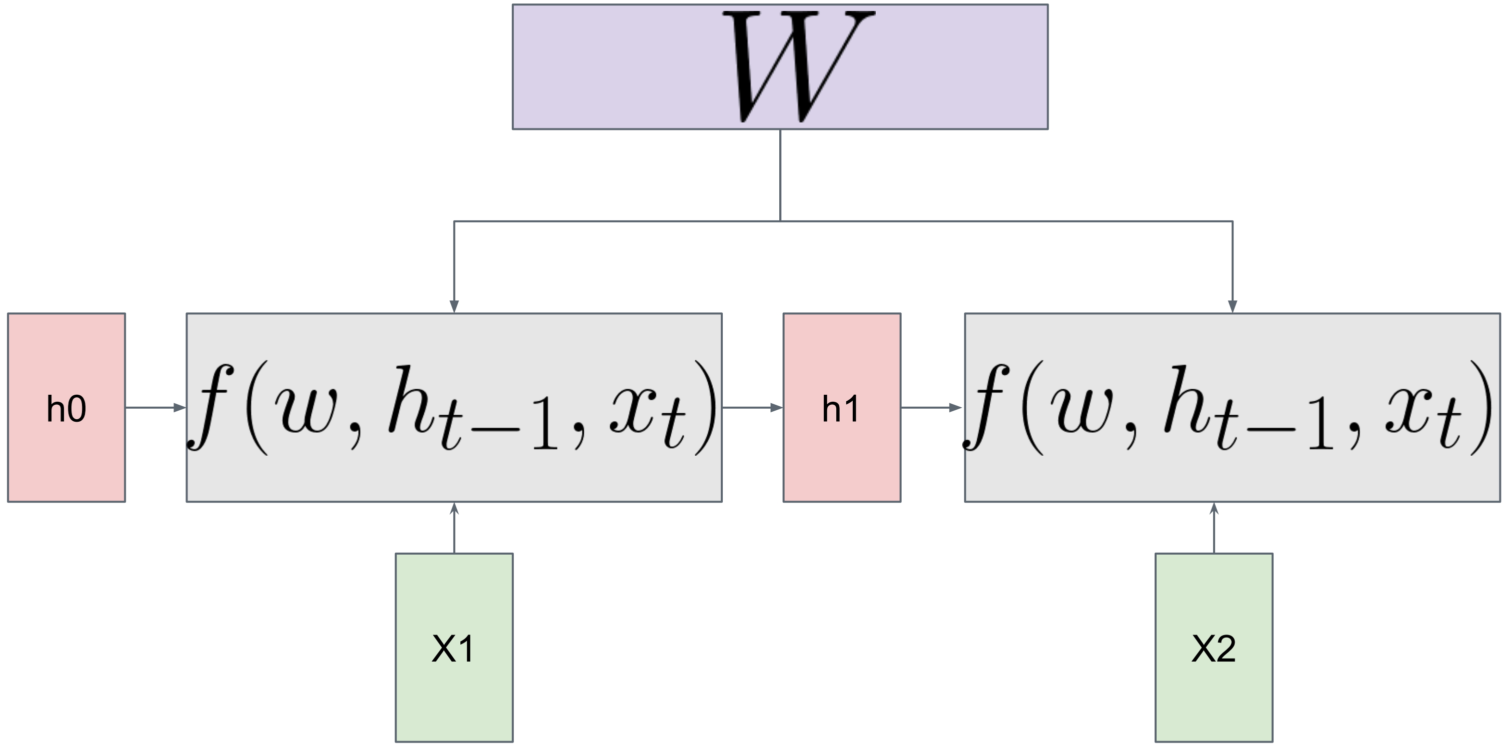
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

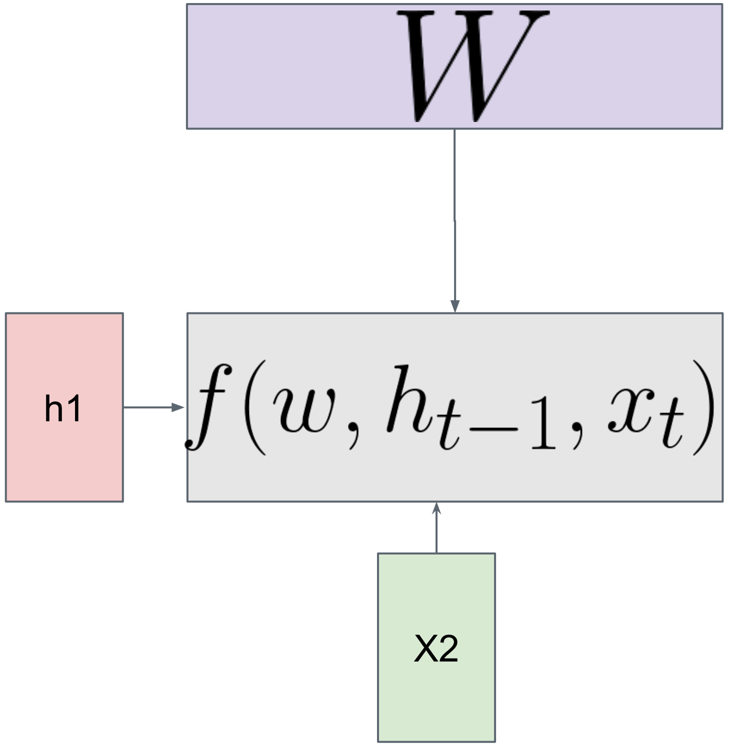
$$c_t = f \odot c_{t-1} + i \odot g$$

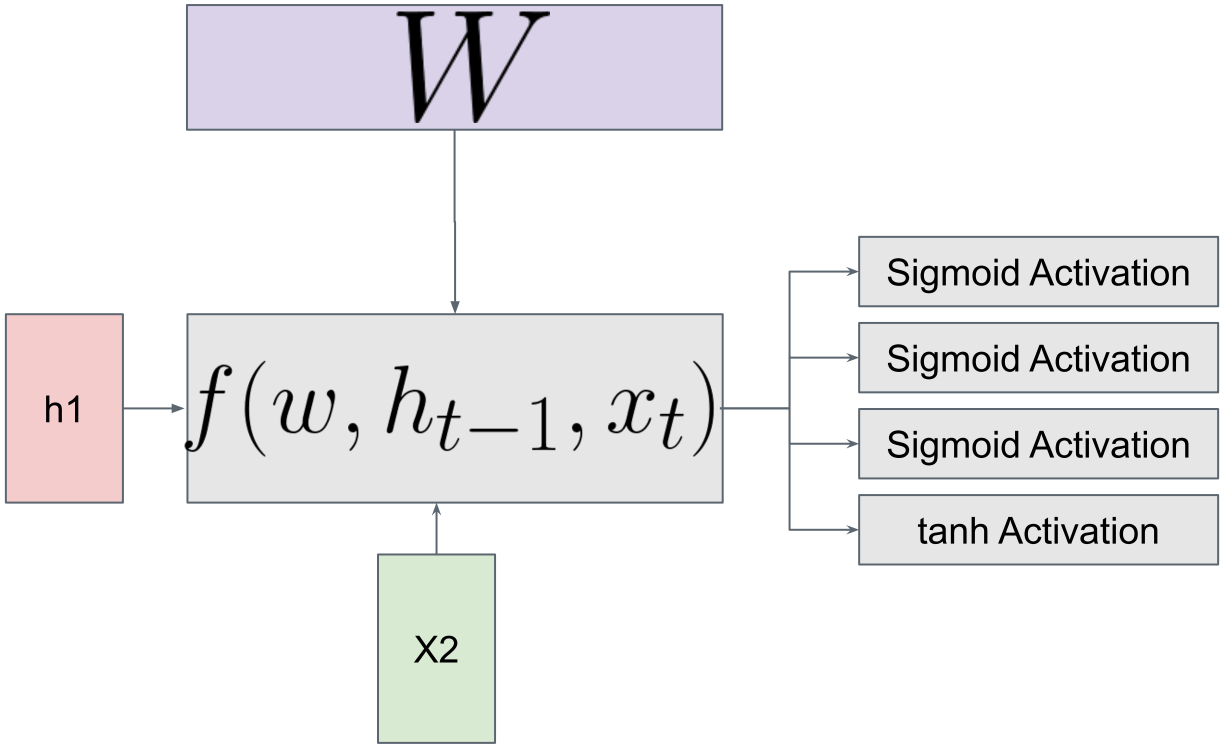
$$h_t = o \odot \tanh(c_t)$$

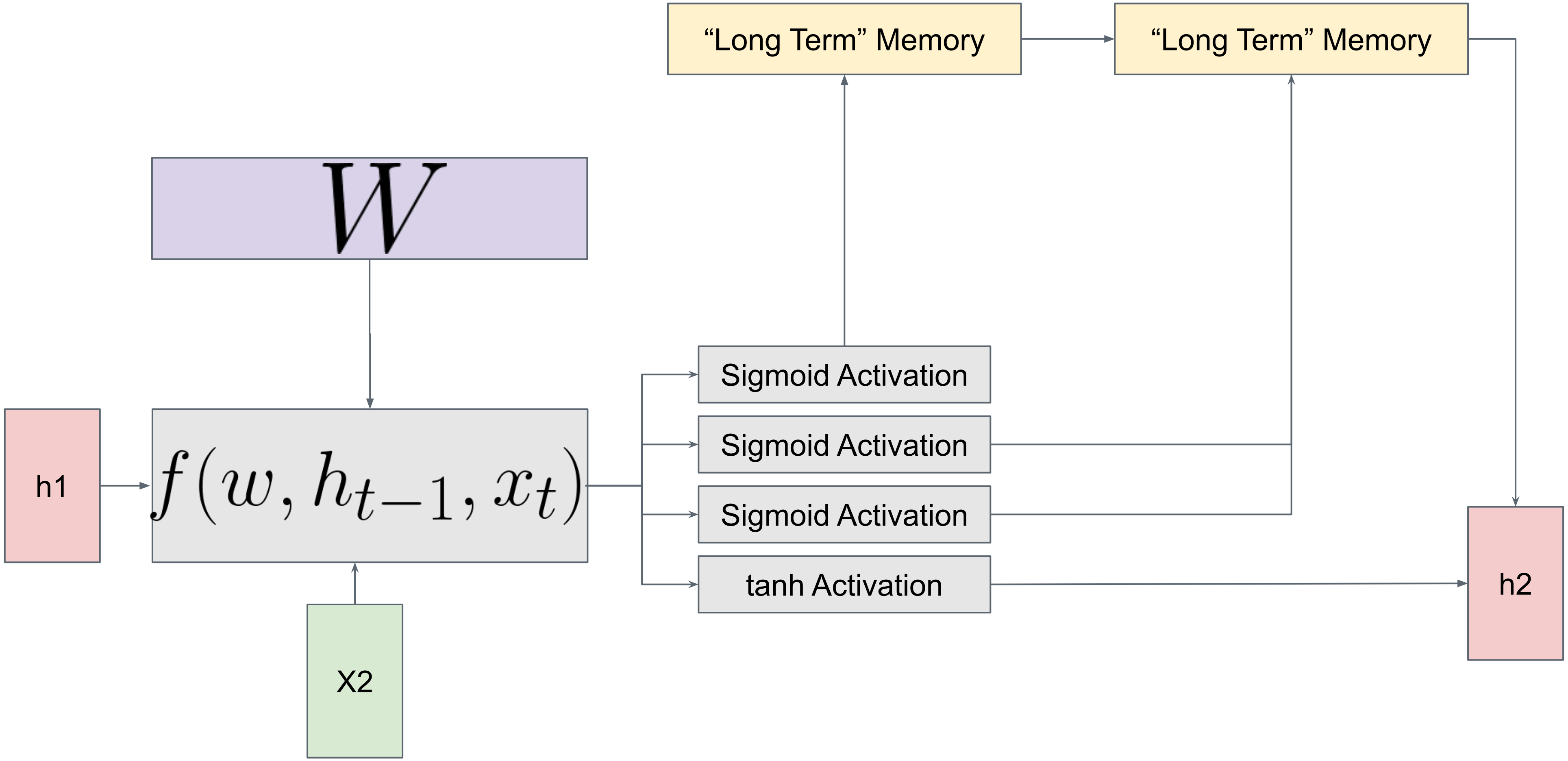
Simple RNN

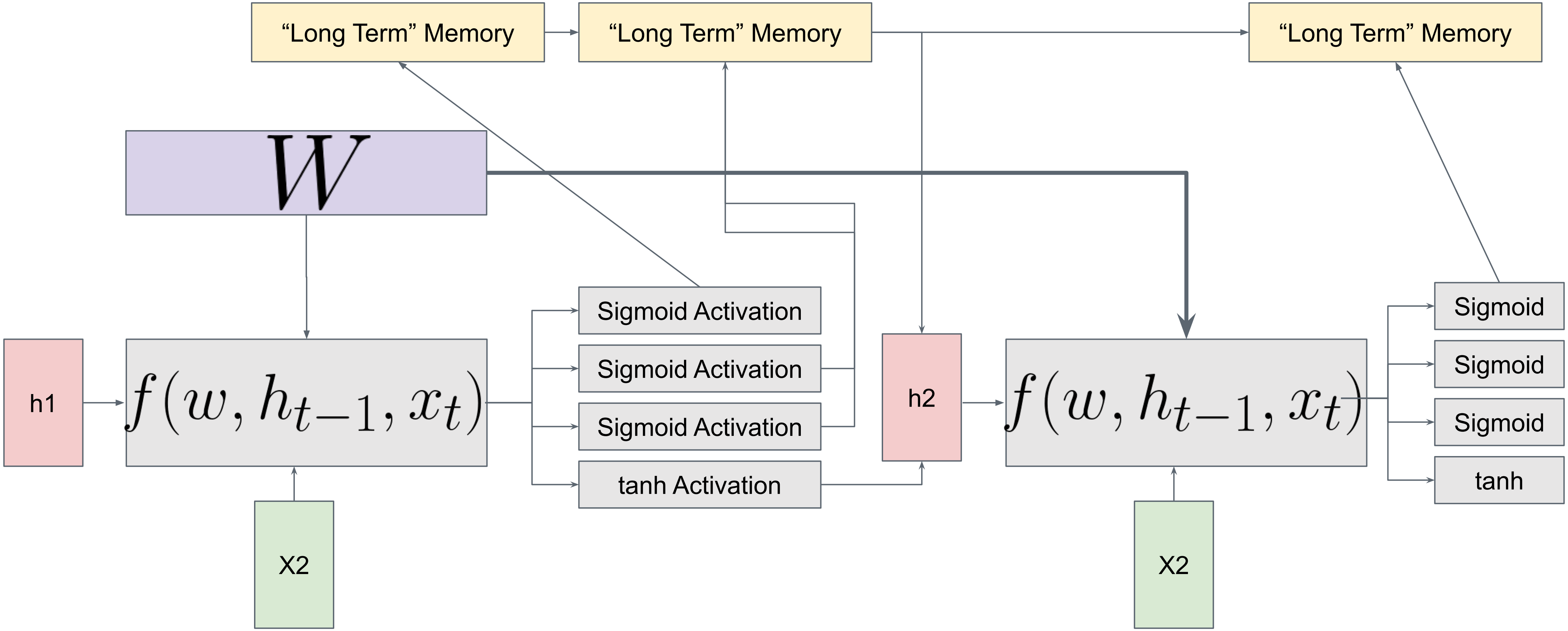
$$h_t = \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$


W
h0
$f(w, h_{t-1}, x_t)$
h1
$f(w, h_{t-1}, x_t)$
X1
X2

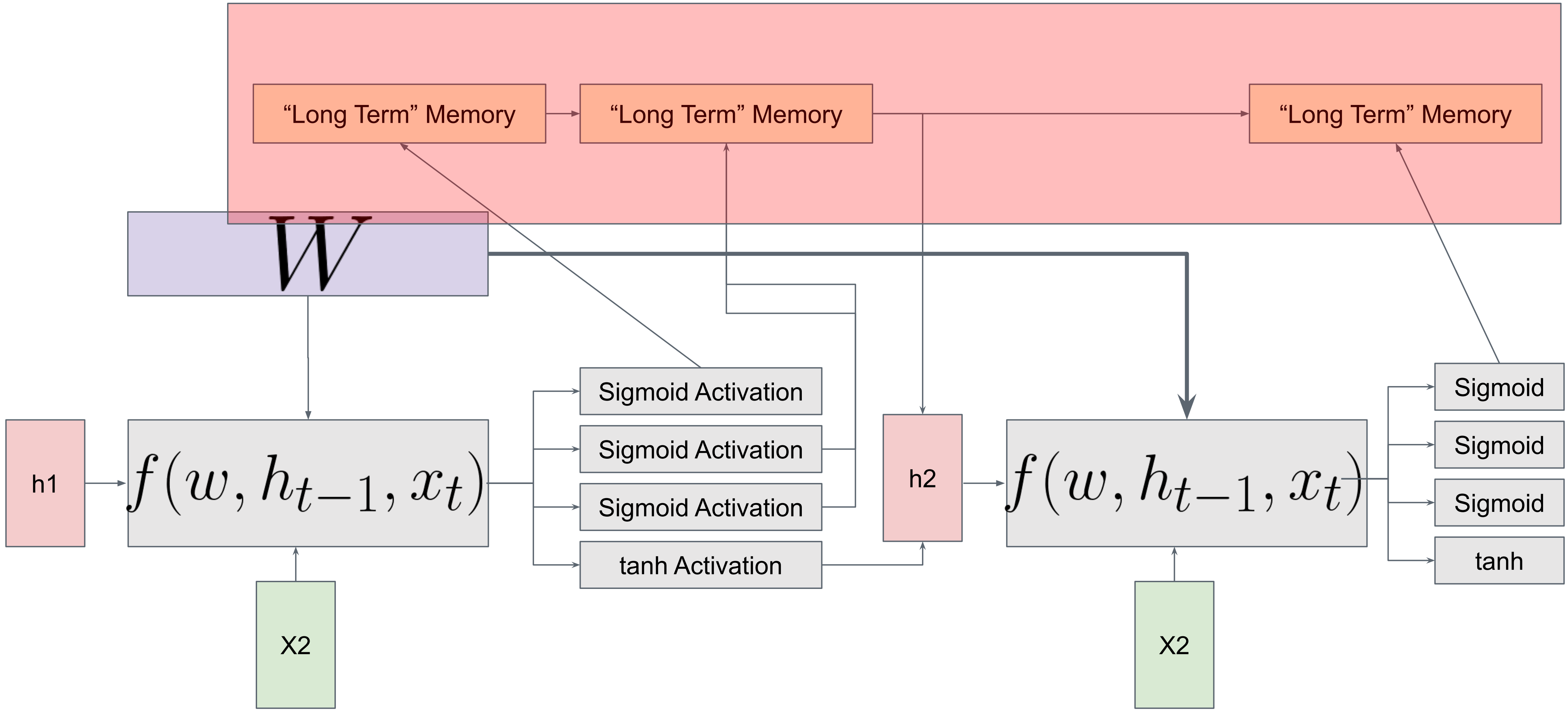

W
h1
$f(w, h_{t-1}, x_t)$
X2


W
h1
$f(w, h_{t-1}, x_t)$
X2
Sigmoid Activation
Sigmoid Activation
Sigmoid Activation
tanh Activation


"Long Term" Memory
"Long Term" Memory
W
h1
$f(w, h_{t-1}, x_t)$
Sigmoid Activation
Sigmoid Activation
Sigmoid Activation
tanh Activation
X2
h2



"Long Term" Memory
"Long Term" Memory
"Long Term" Memory
W
h1
$f(w, h_{t-1}, x_t)$
Sigmoid Activation
Sigmoid Activation
Sigmoid Activation
tanh Activation
h2
$f(w, h_{t-1}, x_t)$
Sigmoid
Sigmoid
Sigmoid
tanh
X2
X2


"Long Term" Memory
"Long Term" Memory
"Long Term" Memory
W
Sigmoid Activation
Sigmoid Activation
Sigmoid Activation
tanh Activation
h1
$f(w, h_{t-1}, x_t)$
X2
h2
$f(w, h_{t-1}, x_t)$
X2
Sigmoid
Sigmoid
Sigmoid
tanh

Summary

- Basic introduction to RNNs
    - RNNs are really great for data that occurs over multiple steps.
    - Normal RNNs are vulnerable to vanishing/exploding gradients.
- Unpacking each step of a RNN
- Contrast Many-To-One and Many-To-Many RNNs
- Introduce LSTM
    - Solves vanishing gradient; still vulnerable to exploding.