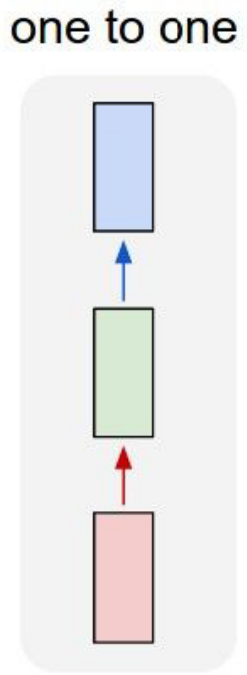
Recurrent Neural Networks

What is RNN?

- Recurrent Neural Network (RNN) is a type of artificial neural network designed for processing sequences of data.

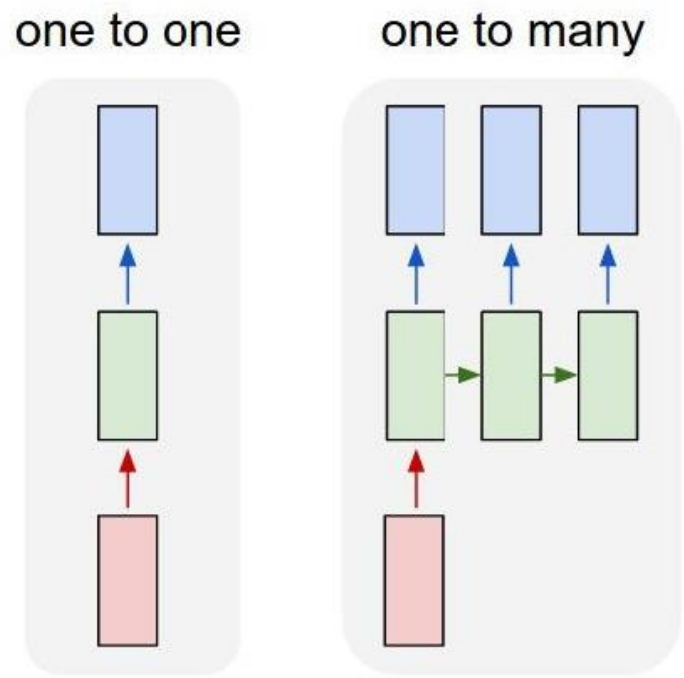
- Unlike traditional feedforward neural networks, where information flows in one direction (from input to output), RNNs have connections that form directed cycles, allowing information to persist.

Recurrent Neural Networks: Process Sequences



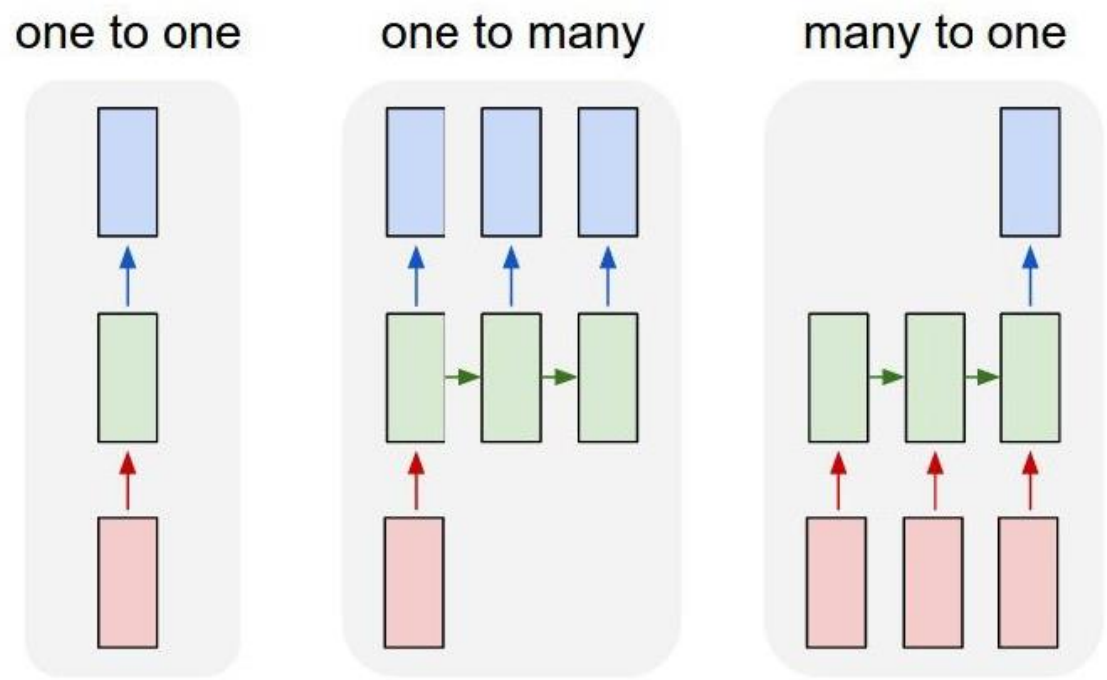
Vanilla Neural Networks

Recurrent Neural Networks: Process Sequences



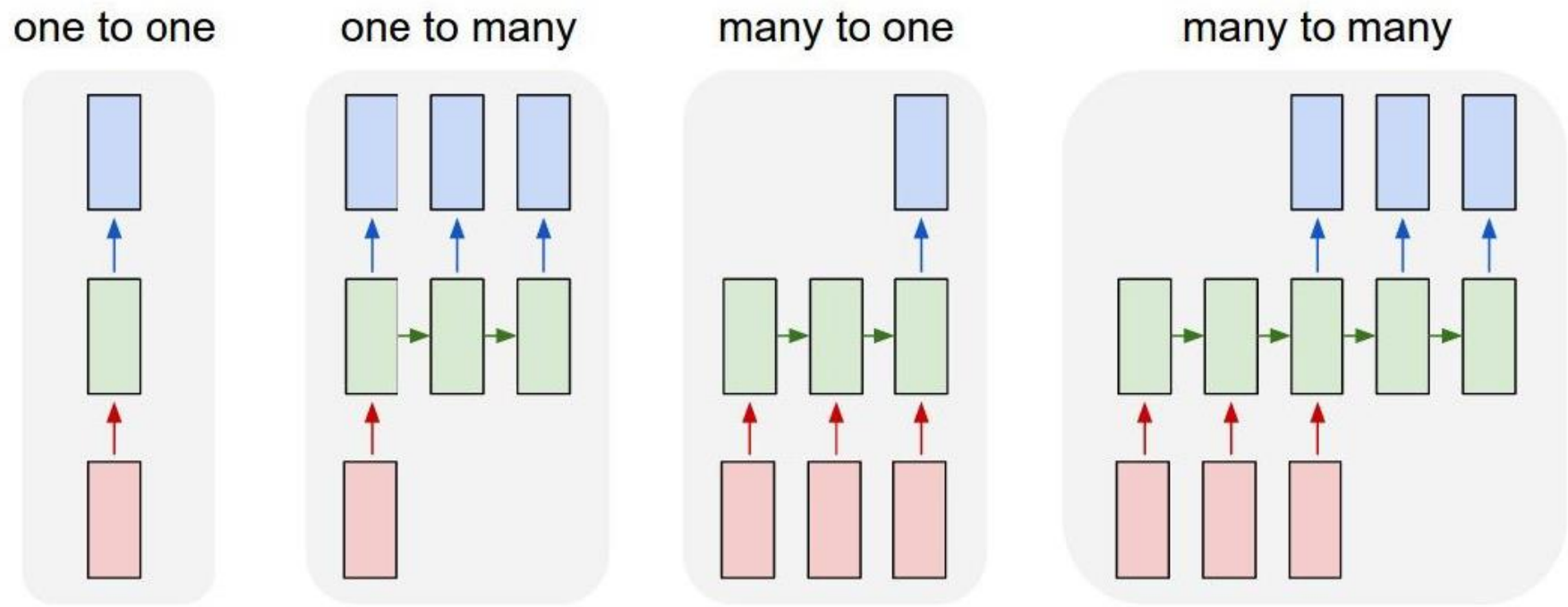
e.g. **Image Captioning**
image -> sequence of words

Recurrent Neural Networks: Process Sequences



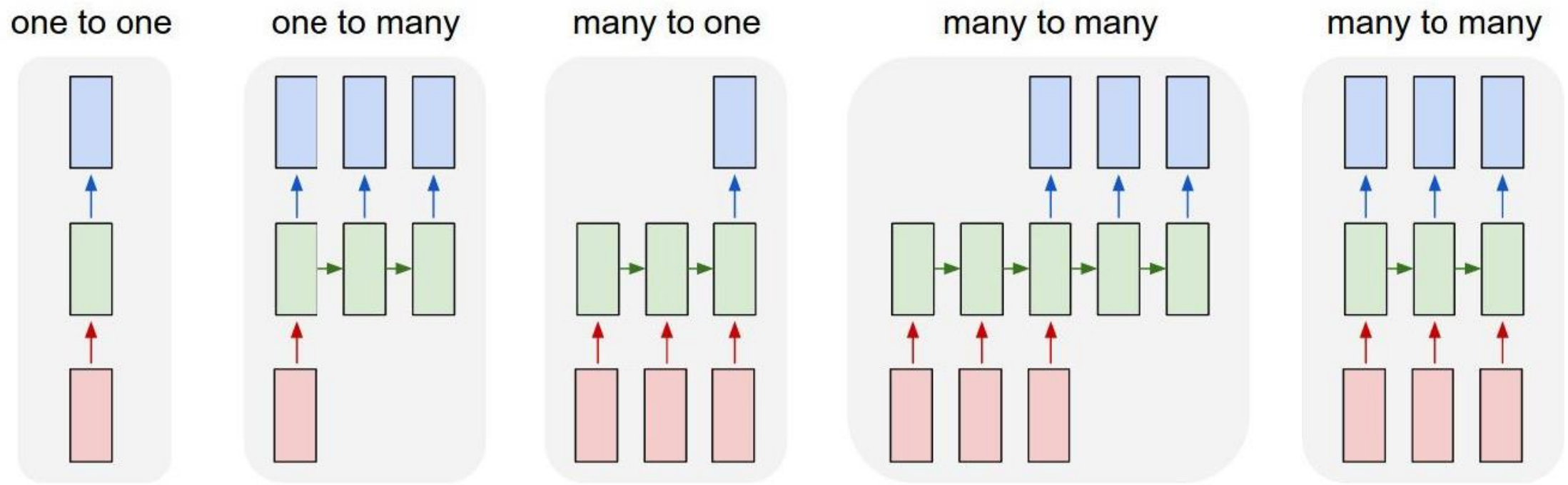
e.g. **action prediction**
sequence of video frames -> action class

Recurrent Neural Networks: Process Sequences



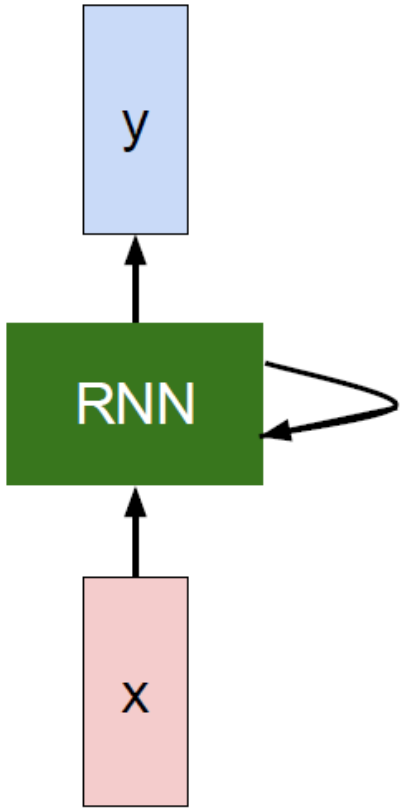
e.g. **Video Captioning**
Sequence of video frames -> caption

Recurrent Neural Networks: Process Sequences

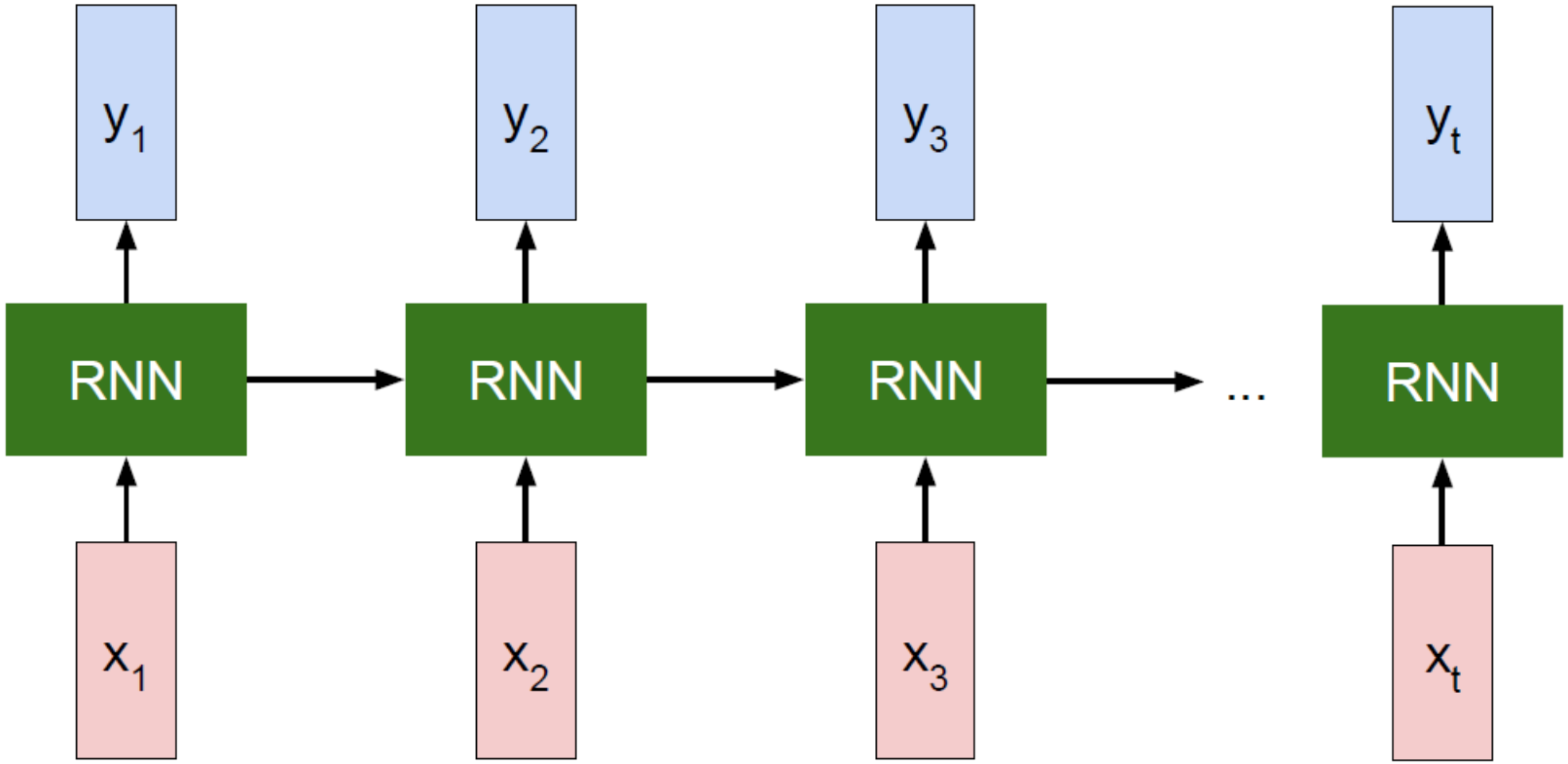


e.g. **Video classification on frame level**

RNN



Unrolled RNN



RNN hidden state

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

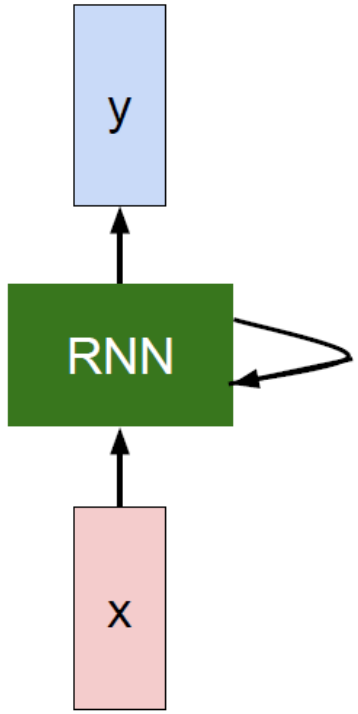
$$h_t = f_W(h_{t-1}, x_t)$$

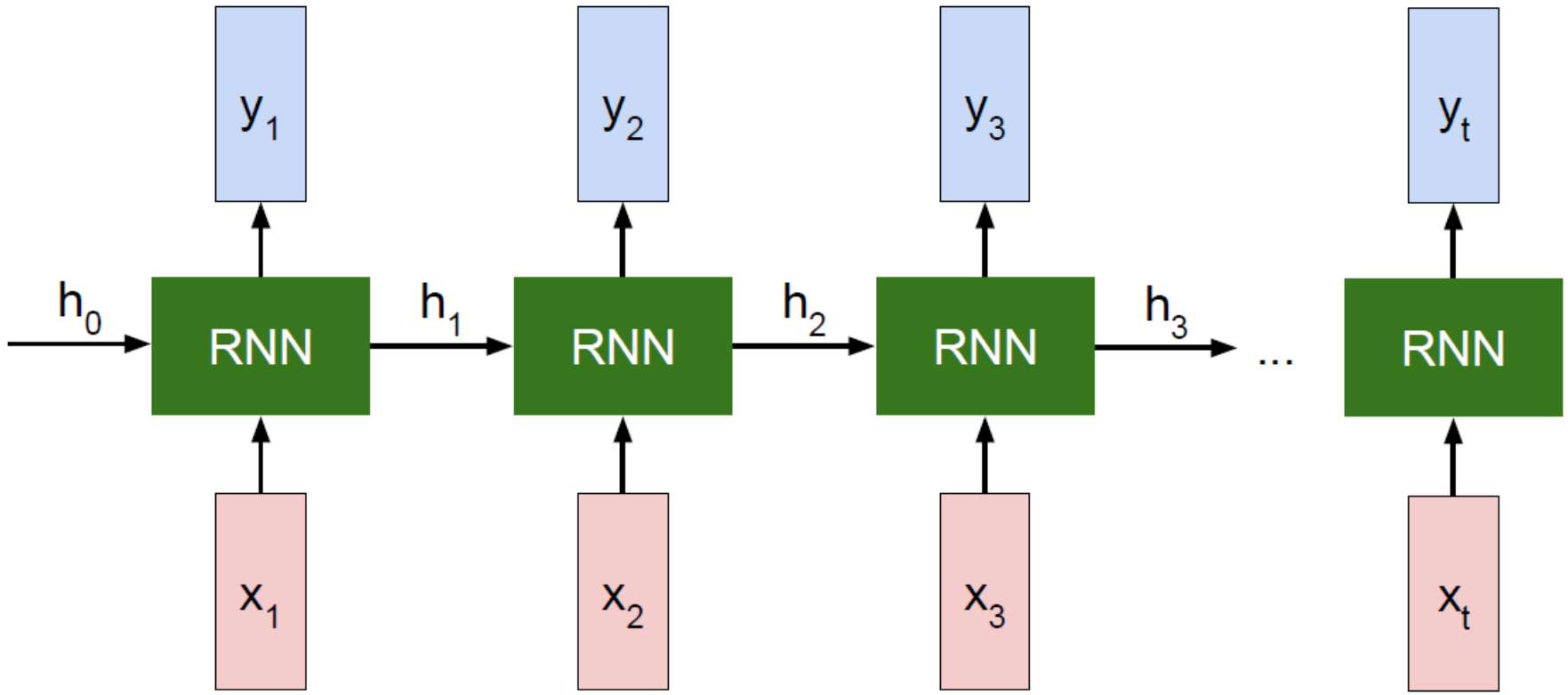
new state

old state

input vector at some time step

some function with parameters W



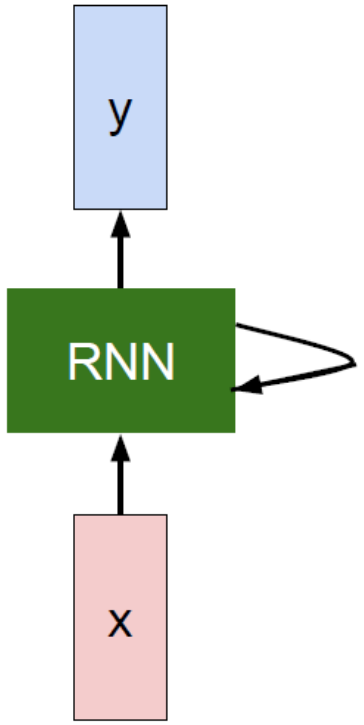

y1
y2
y3
yt
h0
RNN
h1
RNN
h2
RNN
h3
...
RNN
x1
x2
x3
xt

Vanilla Recurrent Neural Network

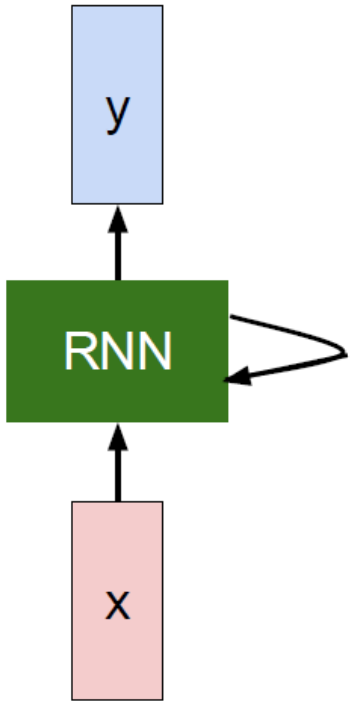
We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

$$h_t = f_W(h_{t-1}, x_t)$$

Notice: the same function and the same set of parameters are used at every time step.



Vanilla Recurrent Neural Network



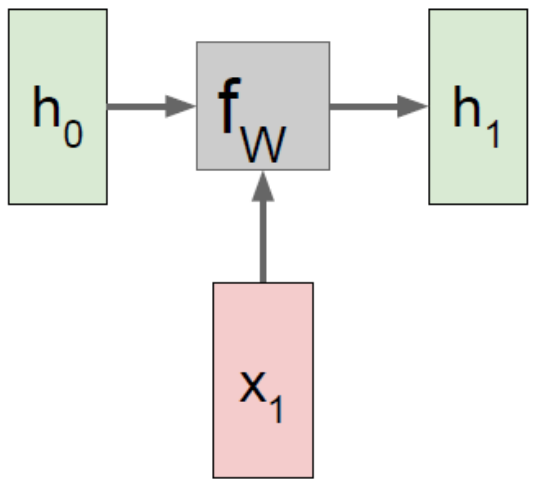
$$h_t = f_W(h_{t-1}, x_t)$$

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

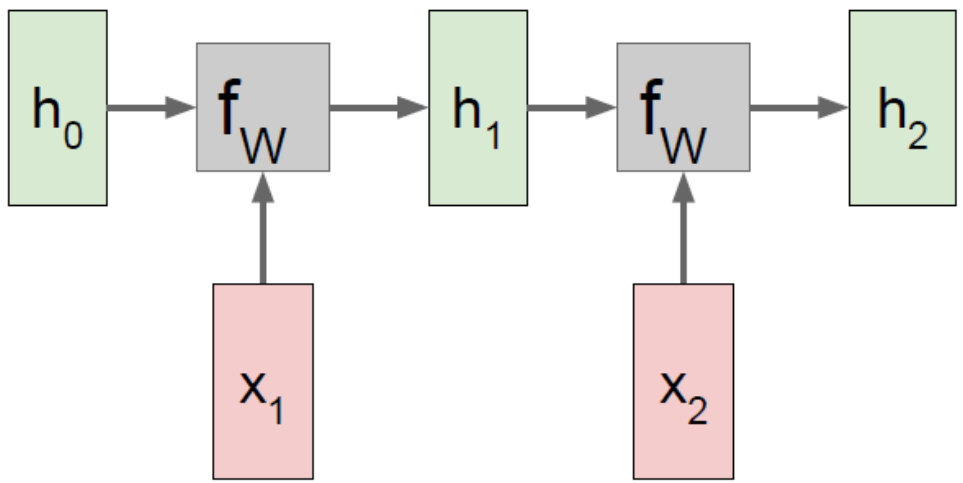
$$y_t = W_{hy}h_t$$

Sometimes called a “Vanilla RNN” or an “Elman RNN” after Prof. Jeffrey Elman

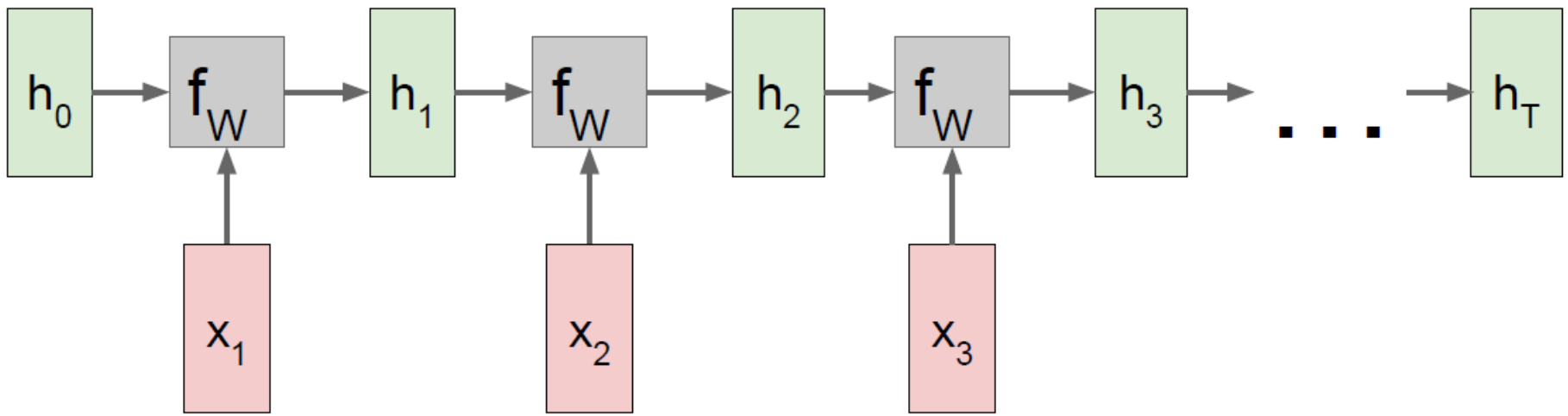
RNN: Computational Graph



RNN: Computational Graph

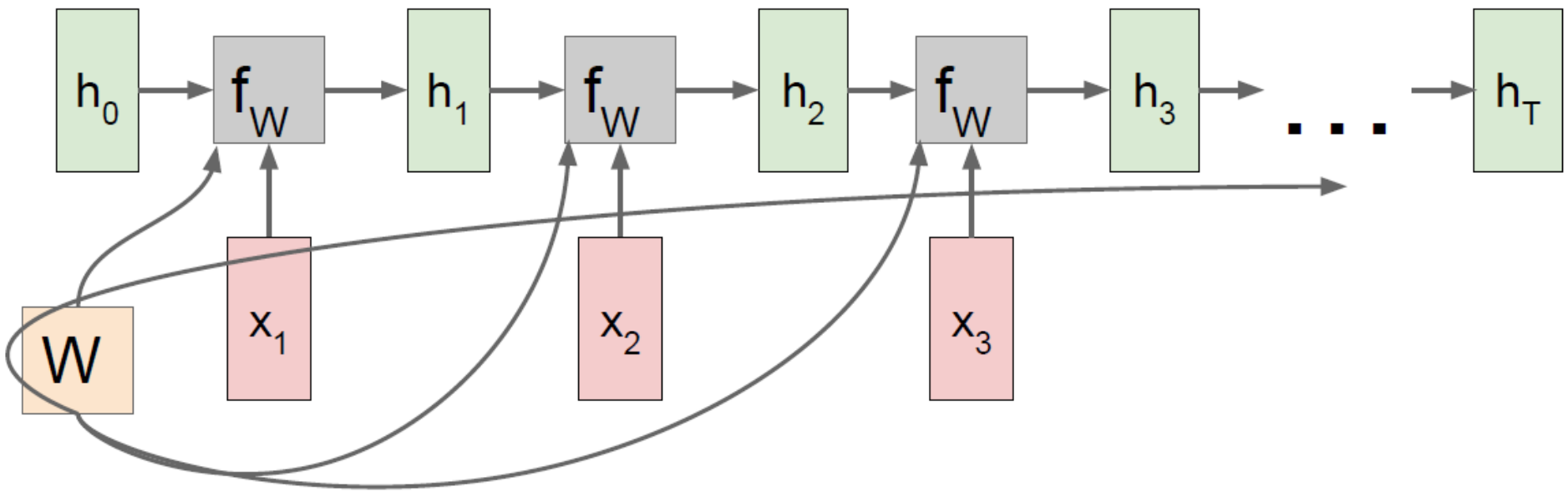


RNN: Computational Graph

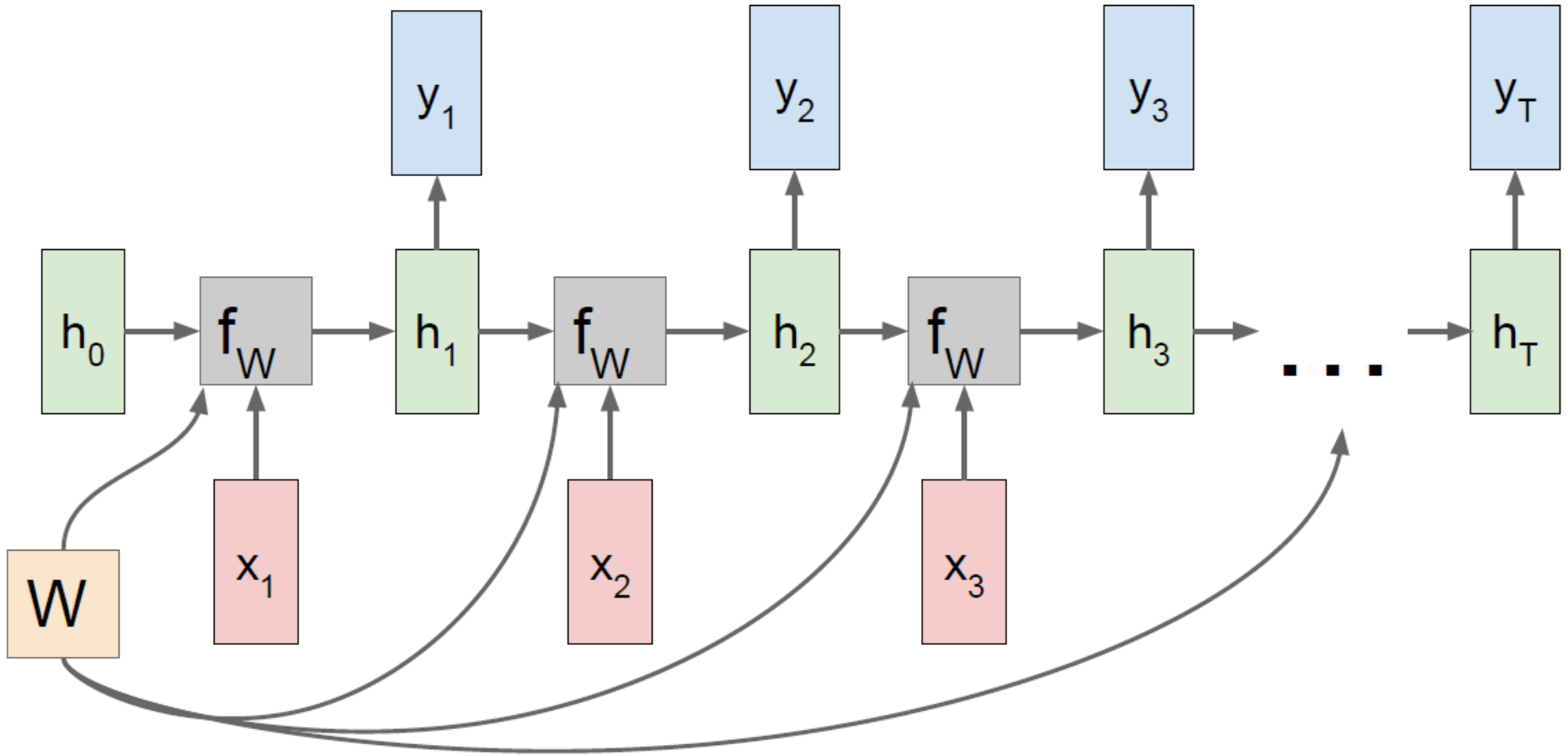


RNN: Computational Graph

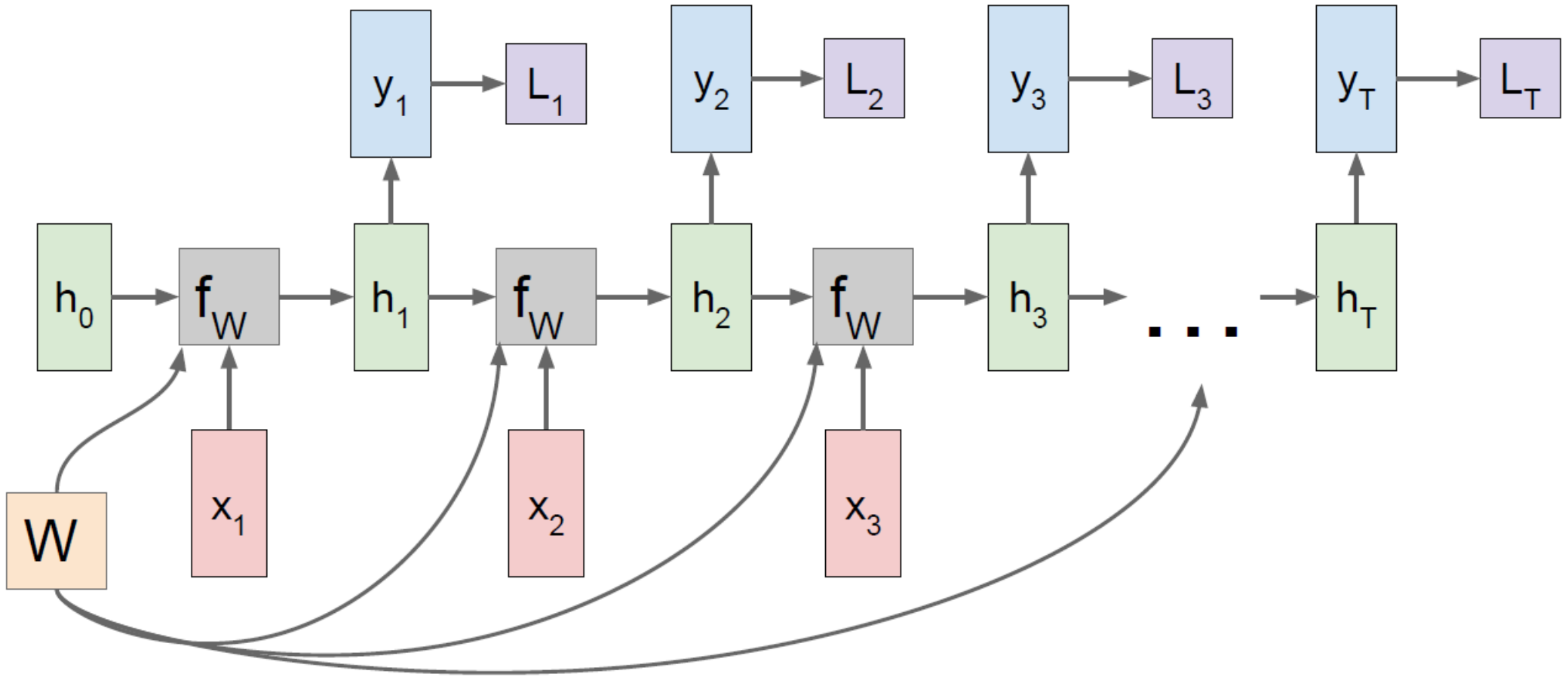
Re-use the same weight matrix at every time-step



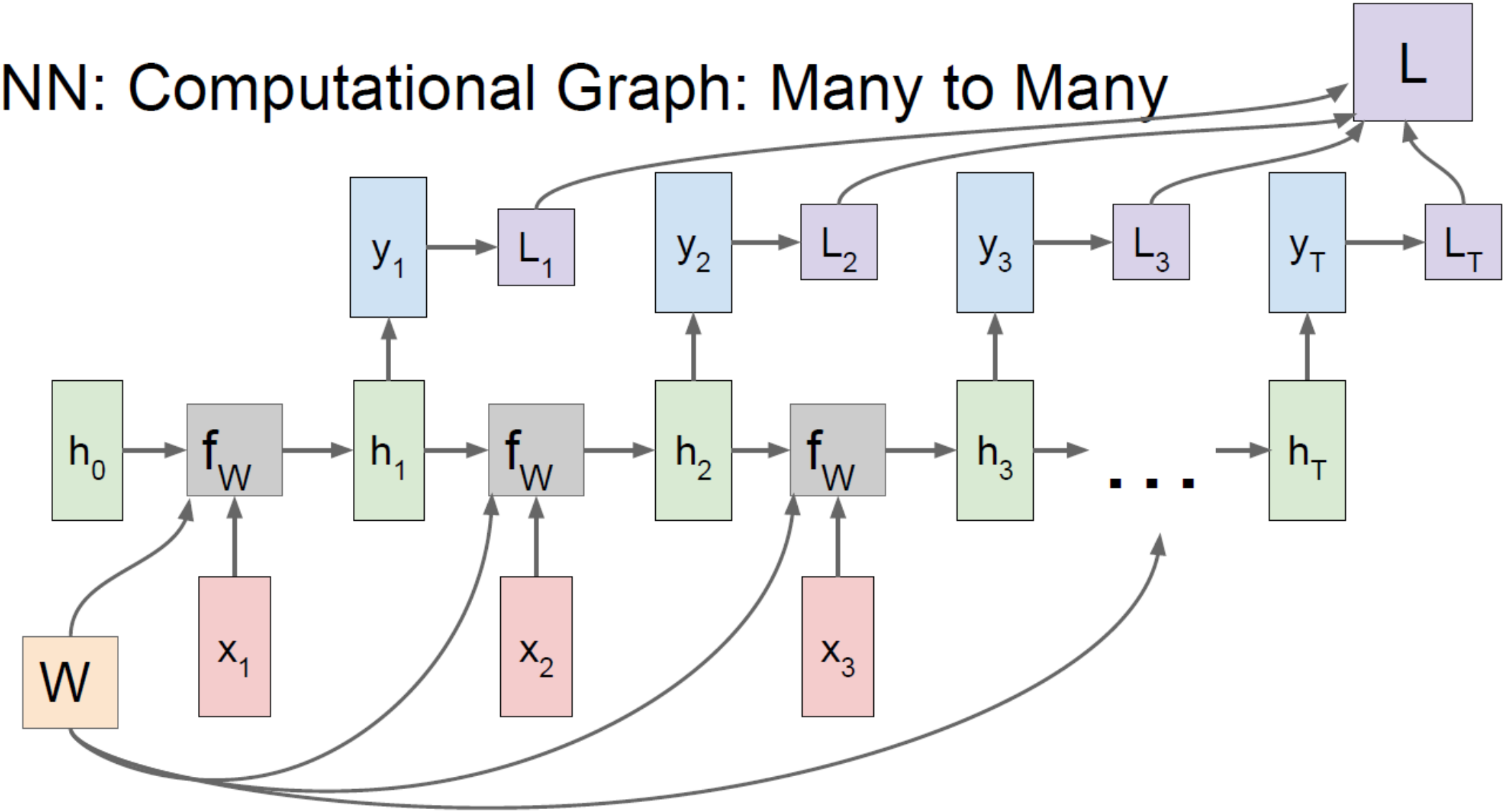
RNN: Computational Graph: Many to Many



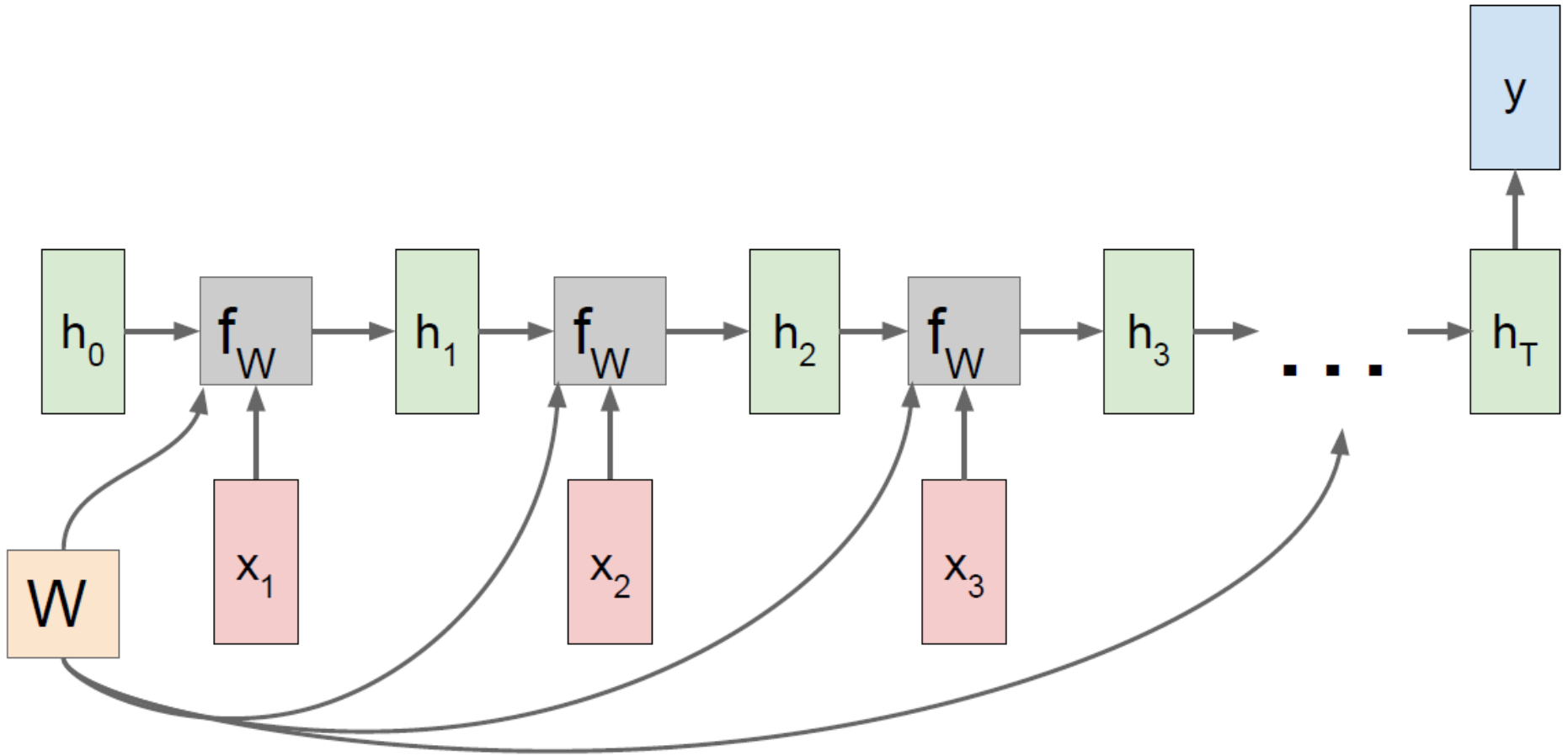
RNN: Computational Graph: Many to Many



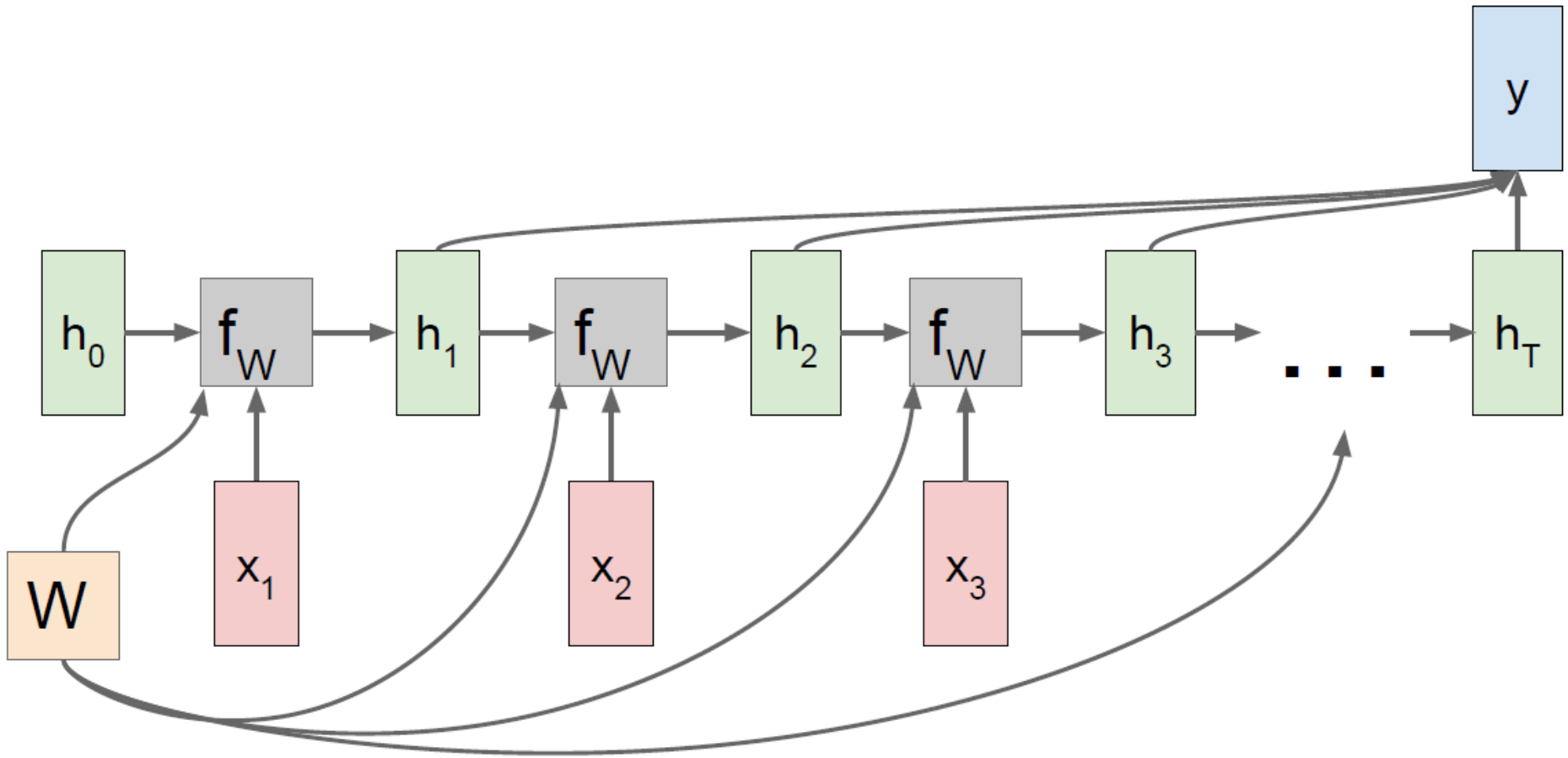
RNN: Computational Graph: Many to Many



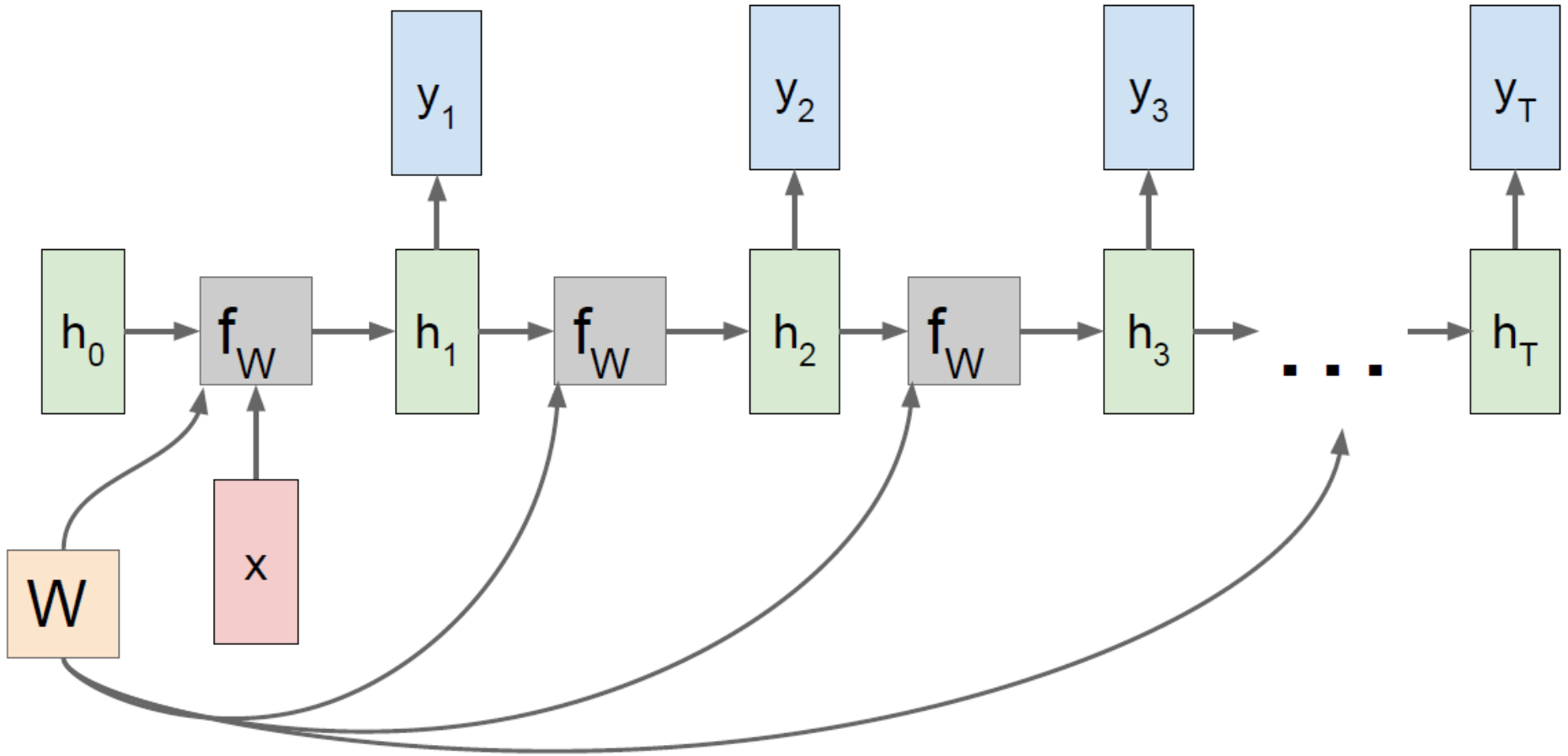
RNN: Computational Graph: Many to One



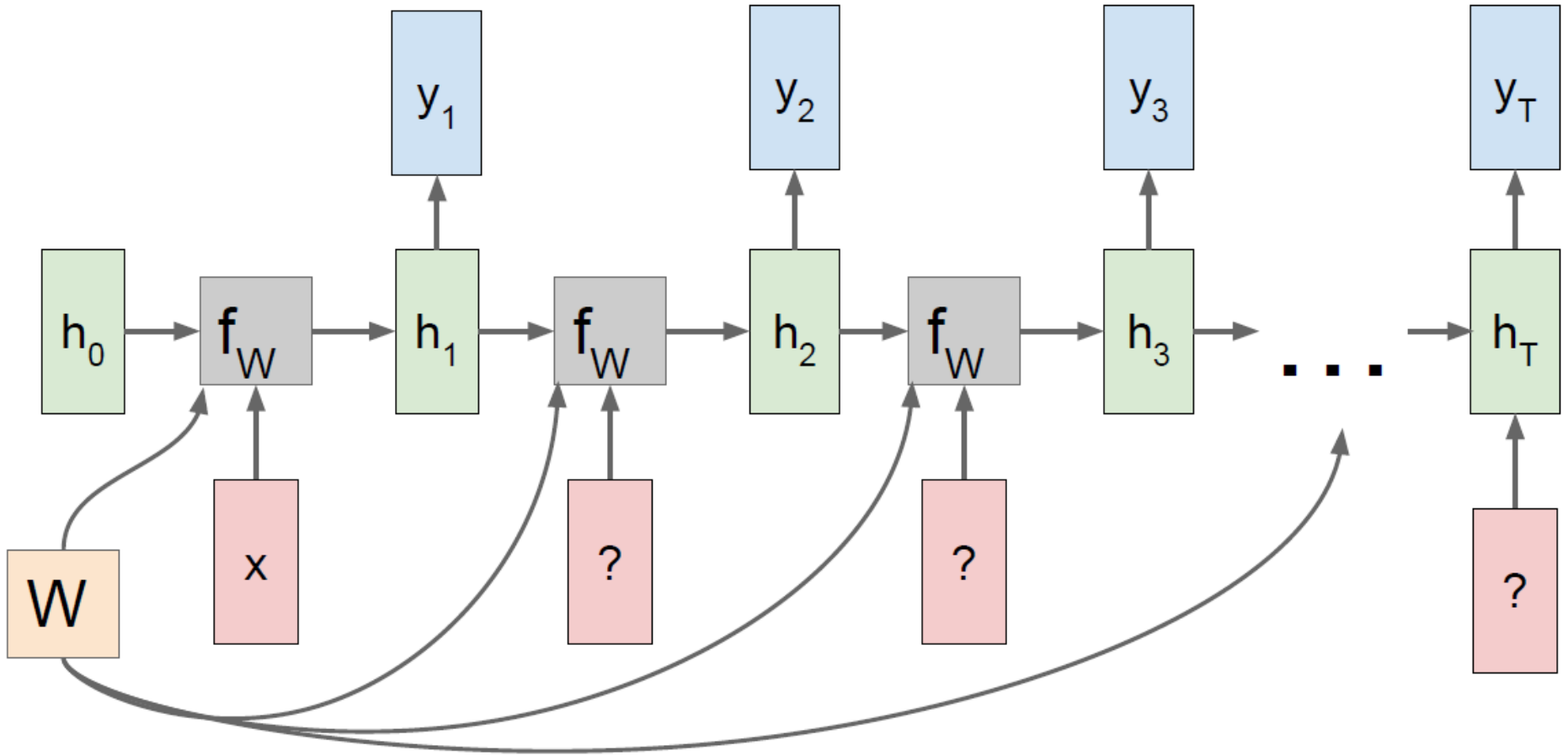
RNN: Computational Graph: Many to One



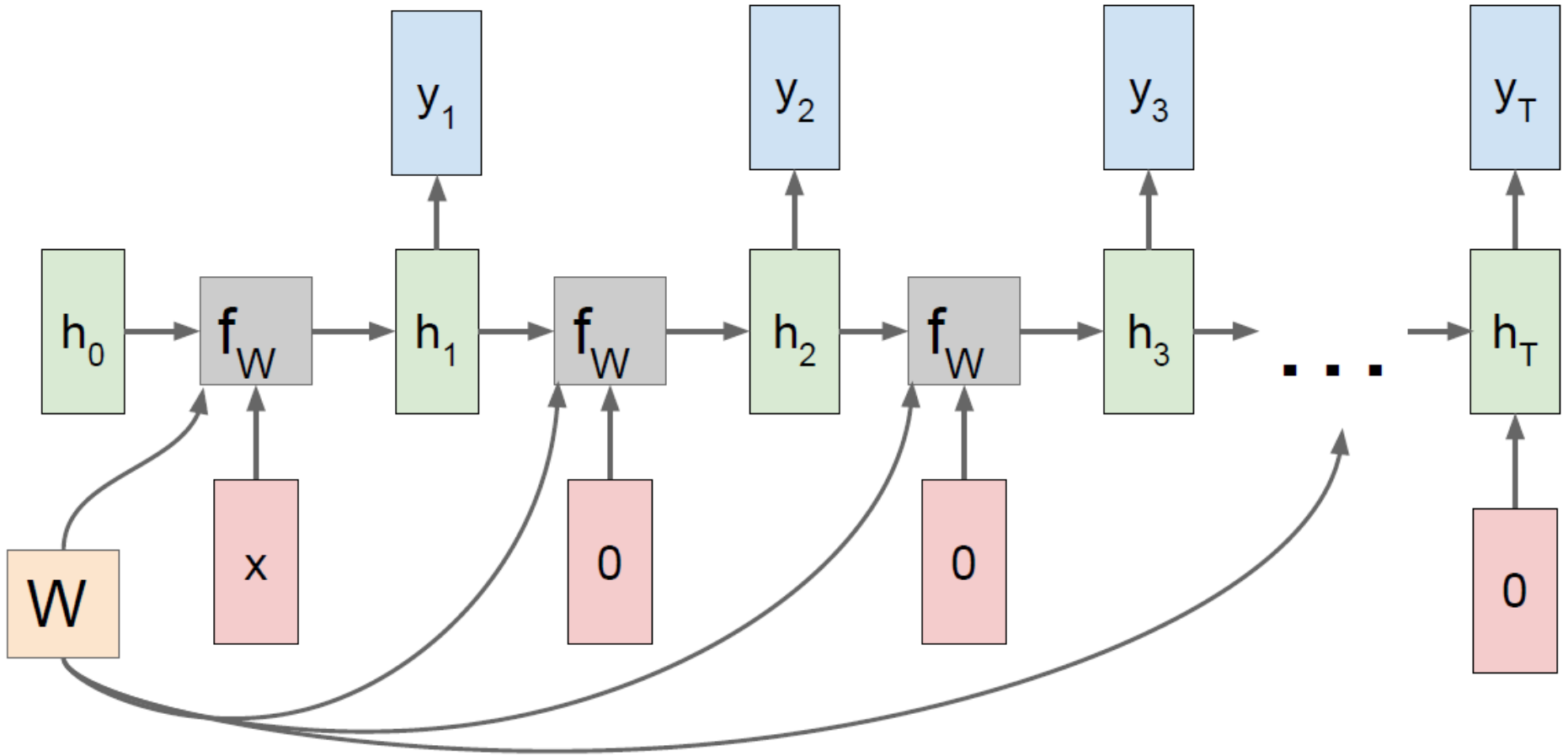
RNN: Computational Graph: One to Many



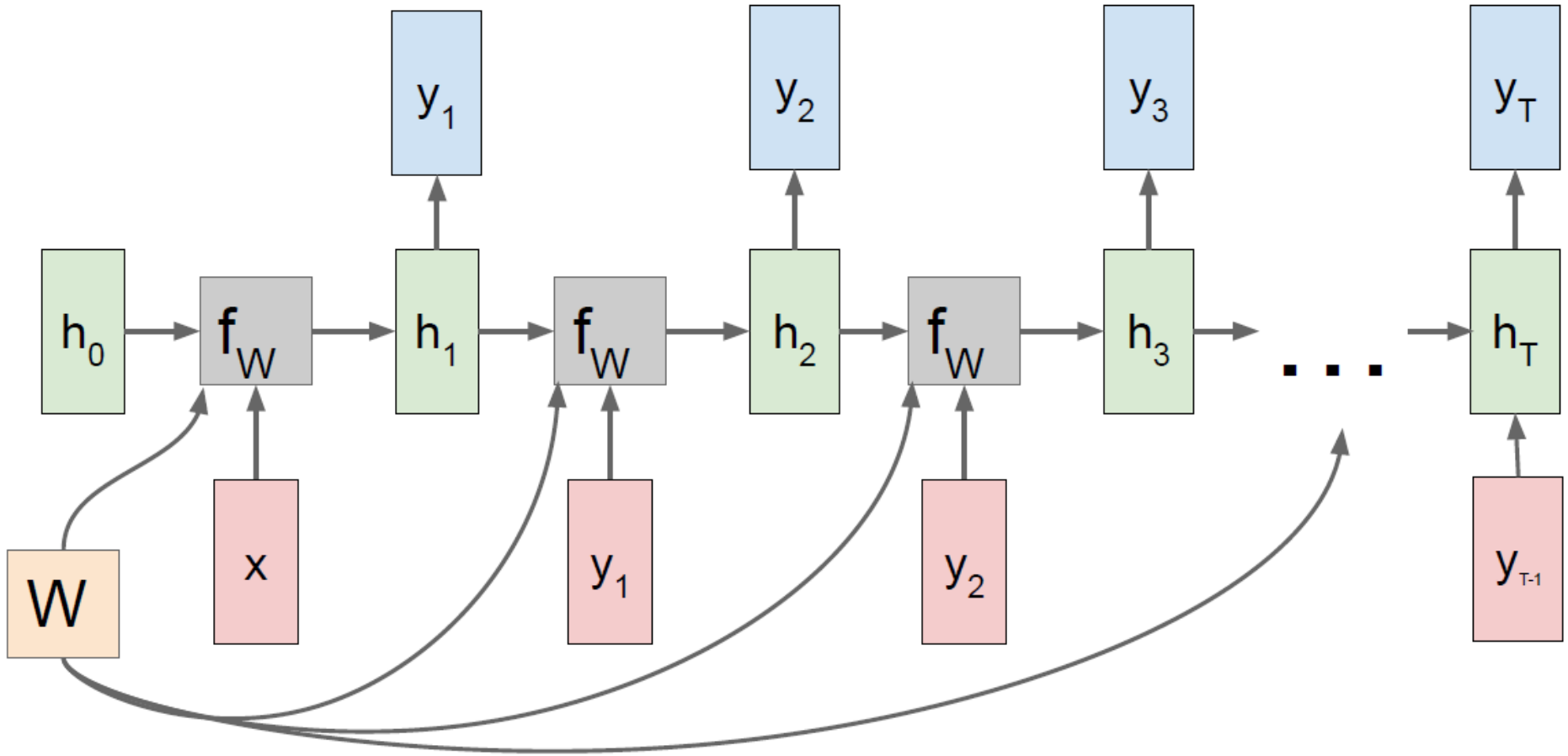
RNN: Computational Graph: One to Many



RNN: Computational Graph: One to Many

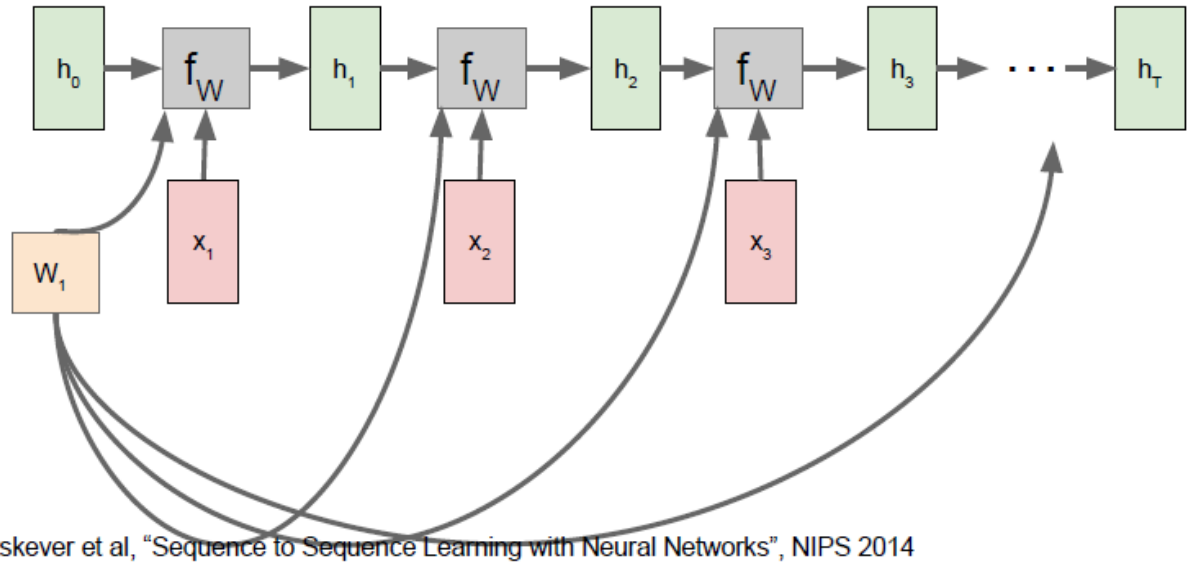


RNN: Computational Graph: One to Many



Sequence to Sequence: Many-to-one + one-to-many

Many to one: Encode input sequence in a single vector

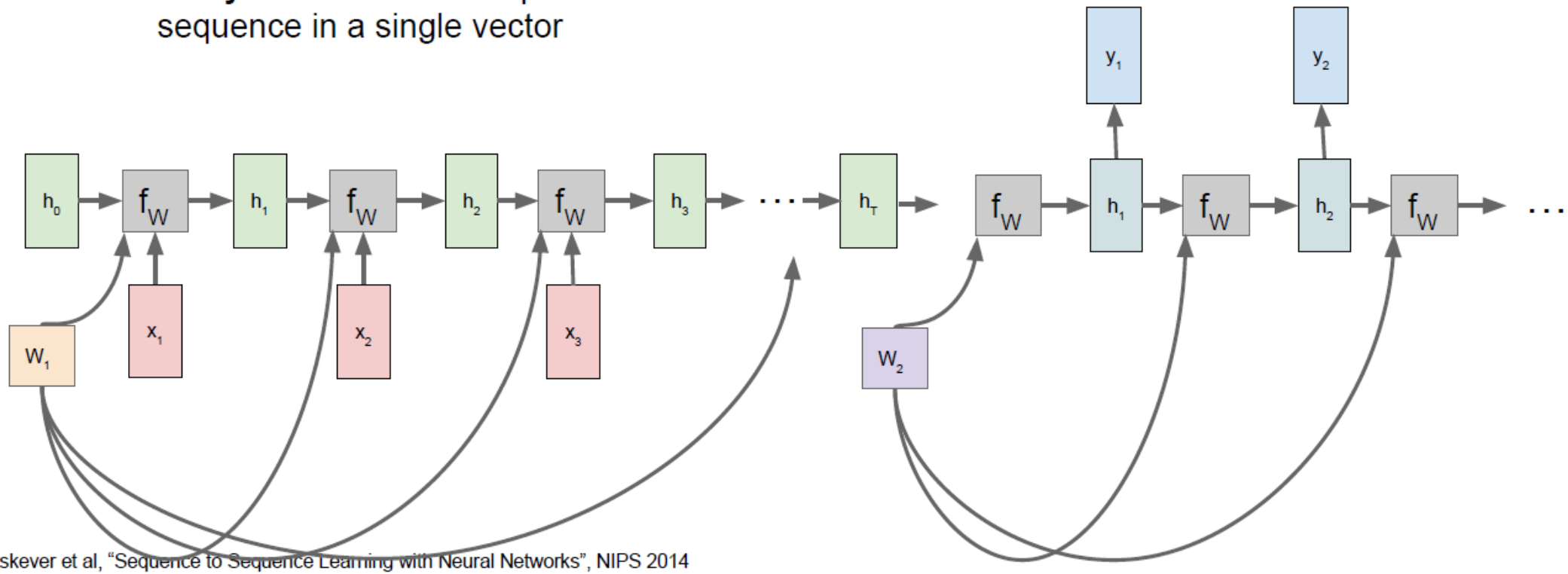


Sutskever et al, "Sequence to Sequence Learning with Neural Networks", NIPS 2014

Sequence to Sequence: Many-to-one + one-to-many

Many to one: Encode input sequence in a single vector

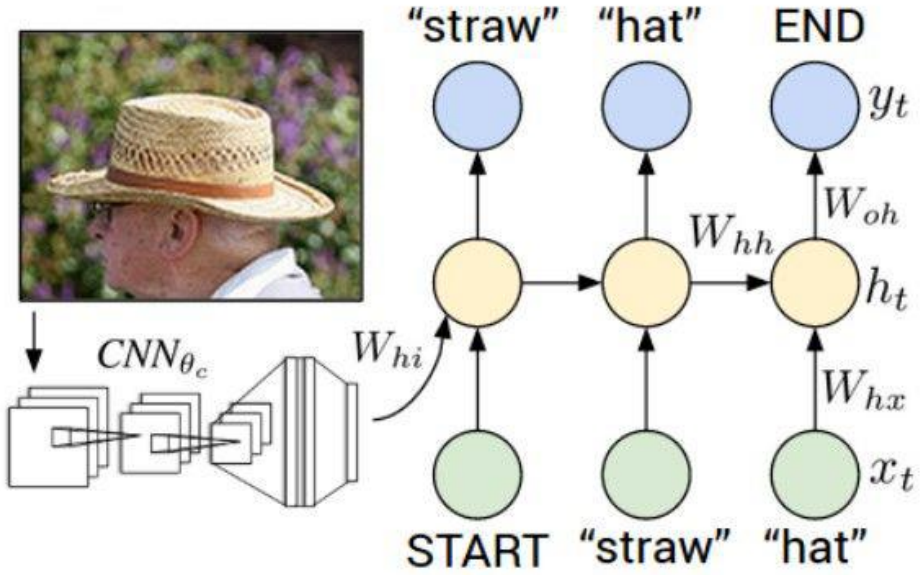
One to many: Produce output sequence from single input vector



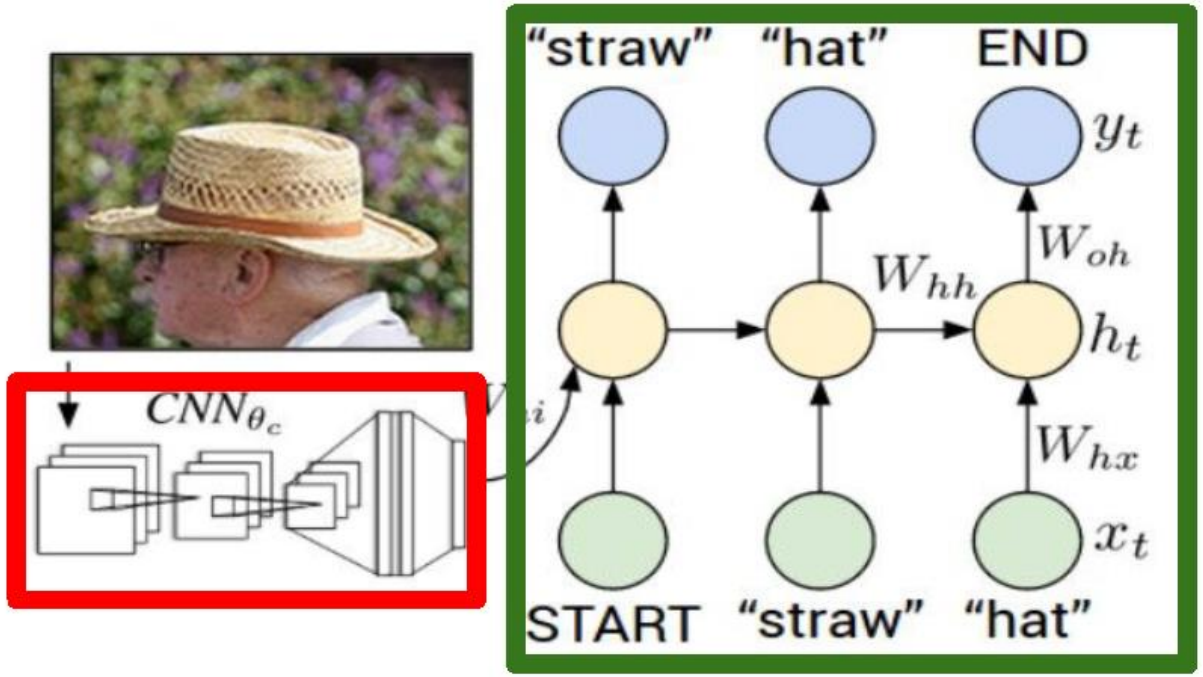
Sutskever et al, "Sequence to Sequence Learning with Neural Networks", NIPS 2014

Image Captioning

- Image Captioning is the task of generating a textual description for an image combining Computer Vision and Natural Language Processing.



Recurrent Neural Network



Convolutional Neural Network

image

conv-64

conv-64

maxpool

conv-128

conv-128

maxpool

conv-256

conv-256

maxpool

conv-512

conv-512

maxpool

conv-512

conv-512

maxpool

FC-4096

FC-4096

FC-1000

softmax

test image

image

conv-64

conv-64

maxpool

conv-128

conv-128

maxpool

conv-256

conv-256

maxpool

conv-512

conv-512

maxpool

conv-512

conv-512

maxpool

FC-4096

FC-4096

FC-1000

softmax

X

test image

image

conv-64

conv-64

maxpool

conv-128

conv-128

maxpool

conv-256

conv-256

maxpool

conv-512

conv-512

maxpool

conv-512

conv-512

maxpool

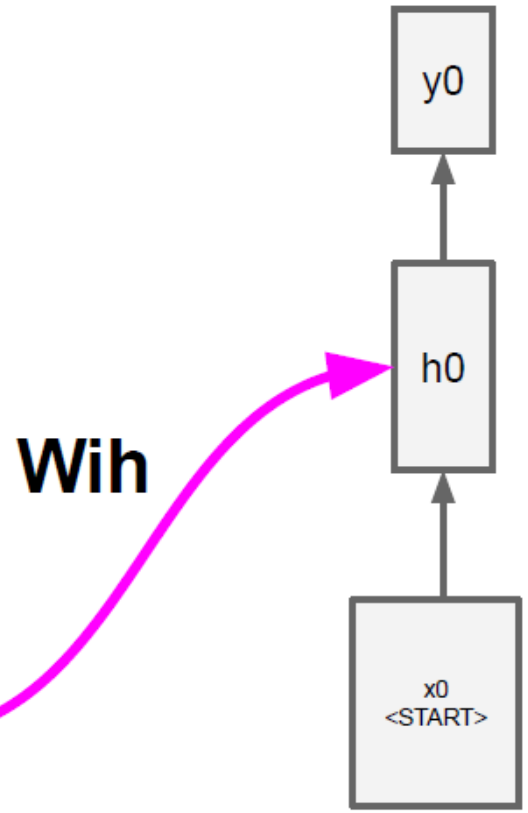
FC-4096

FC-4096

test image

x0
<START>

test image



Wih

before:

$h = \tanh(W_{xh} * x + W_{hh} * h)$

now:

$h = \tanh(W_{xh} * x + W_{hh} * h + W_{ih} * v)$

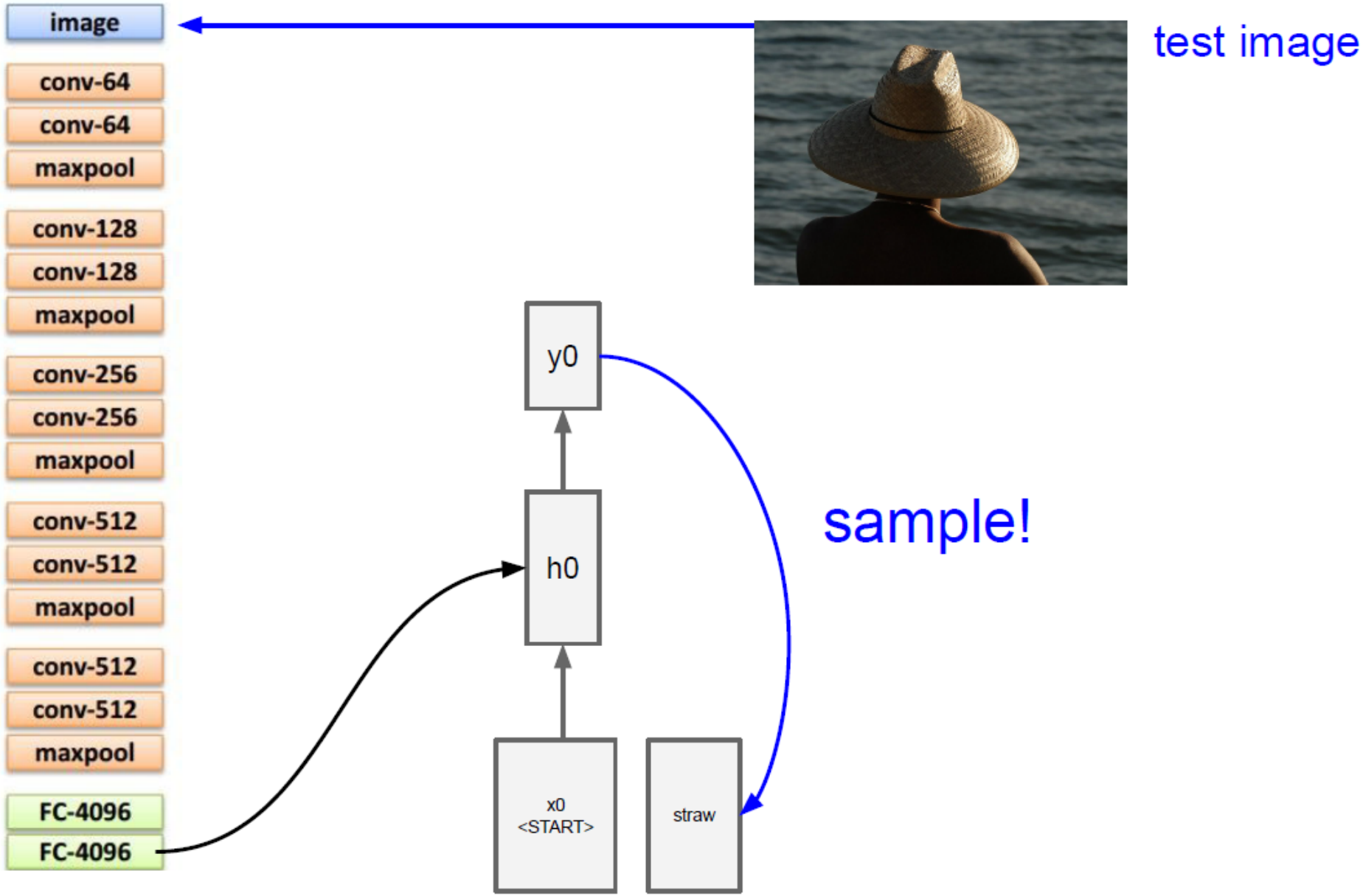


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
y0
h0
x0
<START>
straw
sample!

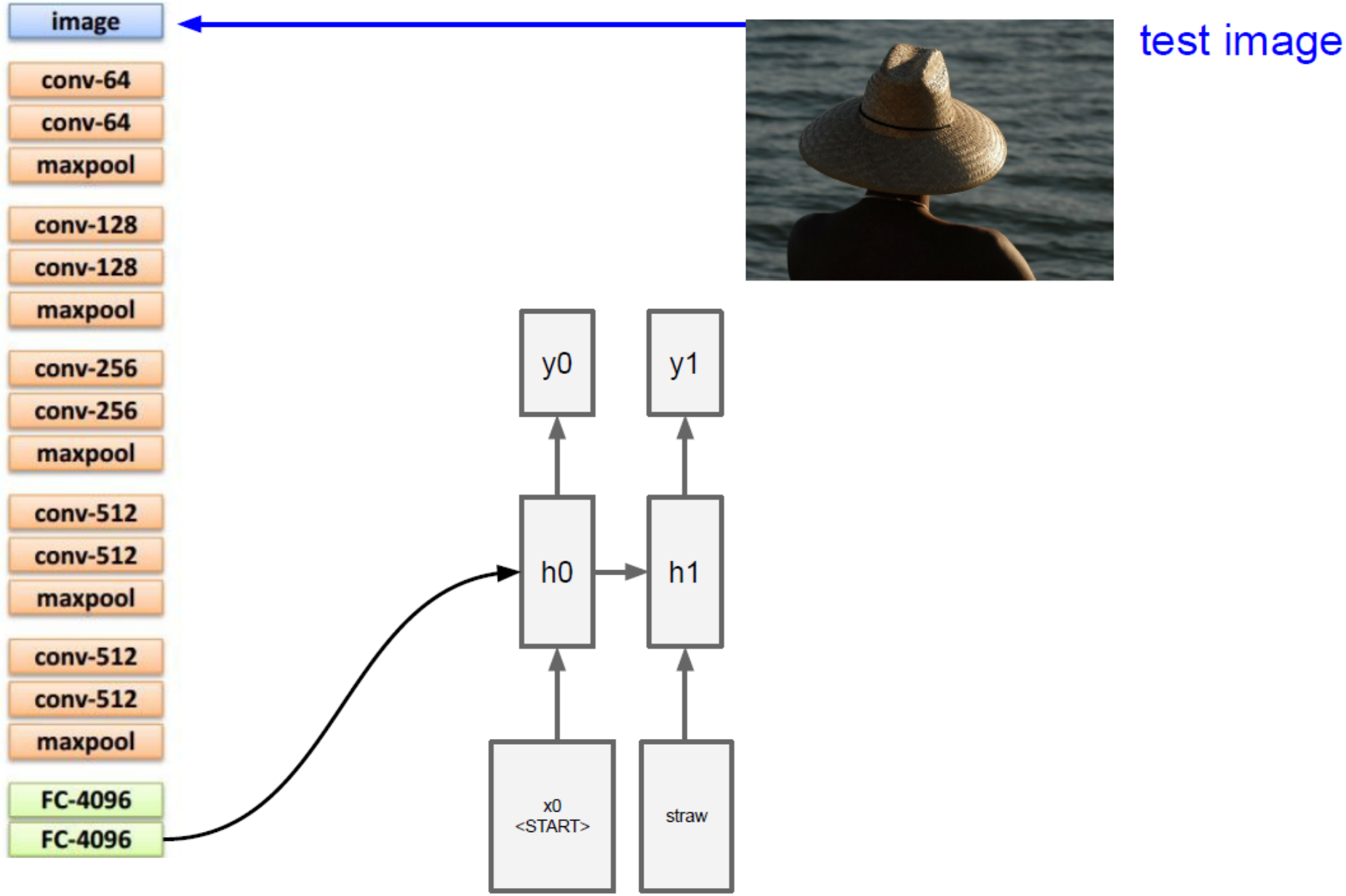


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
y0
y1
h0
h1
x0
<START>
straw

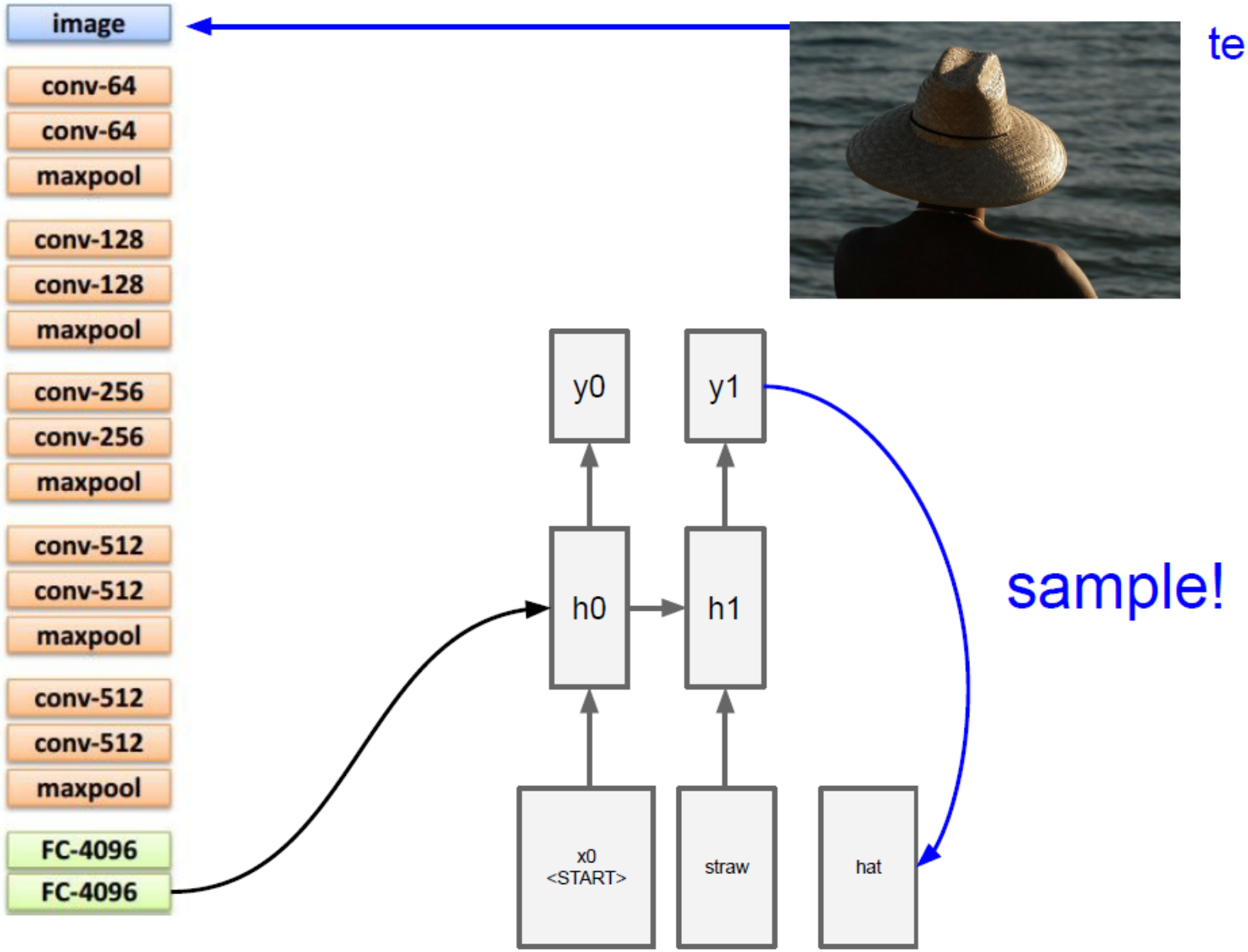


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
y0
y1
h0
h1
x0
<START>
straw
hat
sample!

image

conv-64
conv-64
maxpool

conv-128
conv-128
maxpool

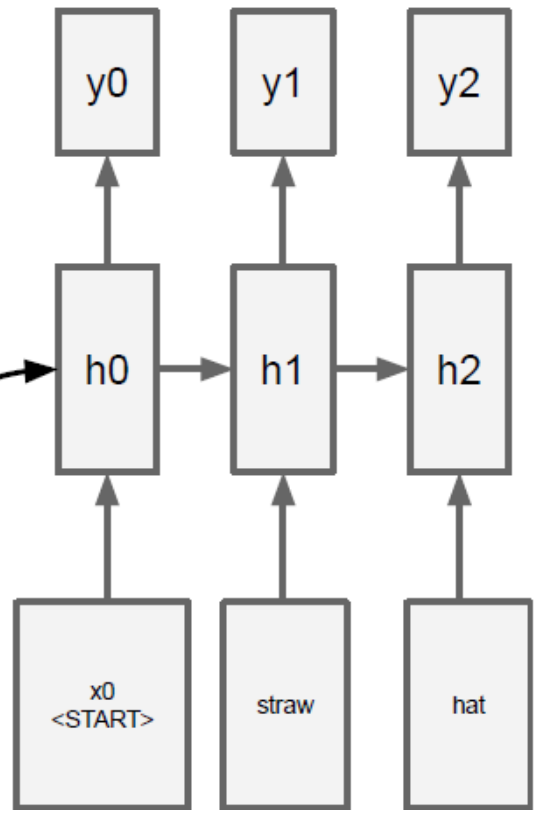
conv-256
conv-256
maxpool

conv-512
conv-512
maxpool

conv-512
conv-512
maxpool

FC-4096
FC-4096

test image



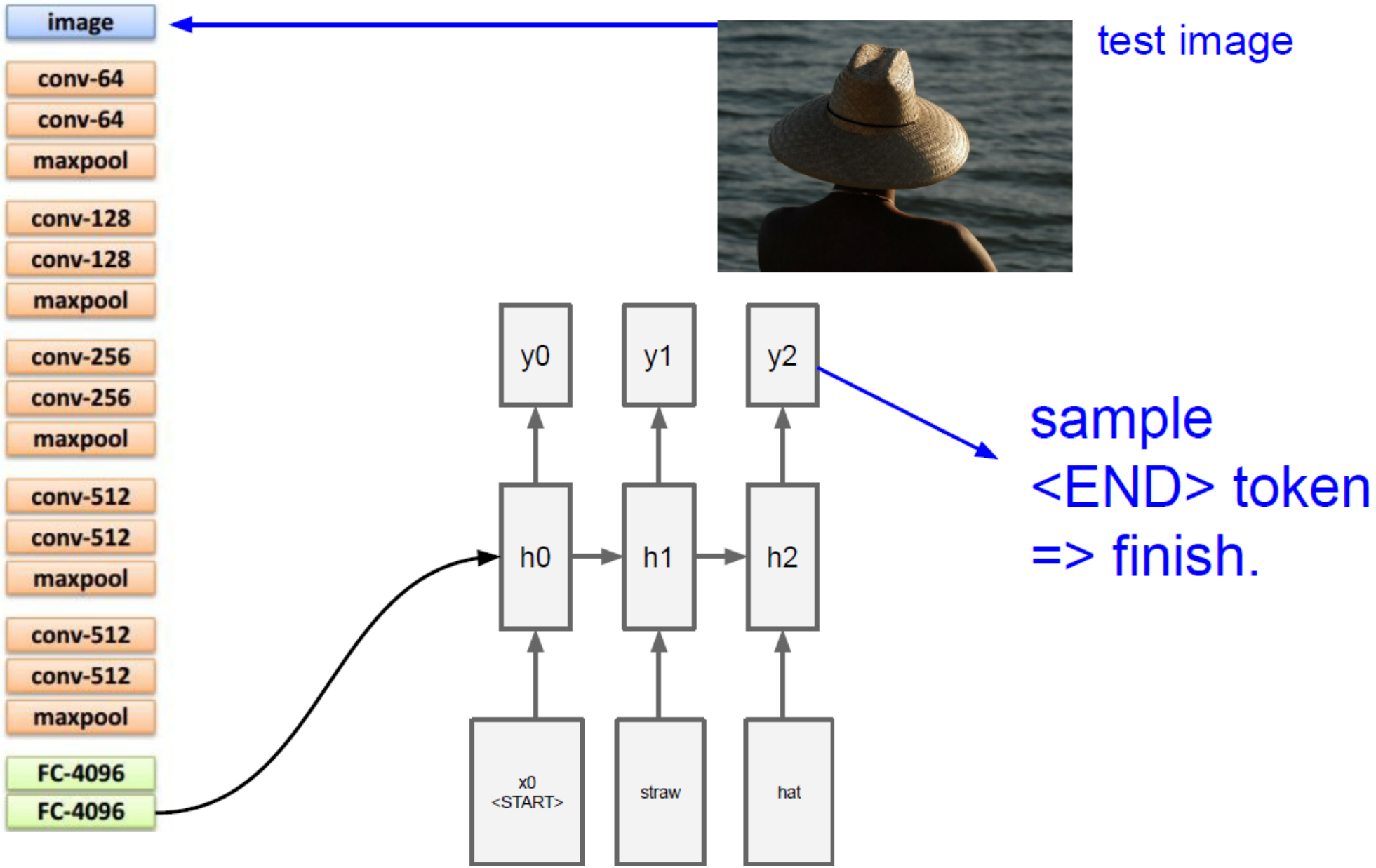


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
y0
y1
y2
h0
h1
h2
x0
<START>
straw
hat
sample
<END> token
=> finish.

Image Captioning: Example Results

Captions generated using neuraltalk2
All images are CC0 Public domain:
cat suitcase, cat tree, dog, bear
surfers, tennis, giraffe, motorcycle

A cat sitting on a suitcase on the floor

A cat is sitting on a tree branch

A dog is running in the grass with a frisbee

A white teddy bear sitting in the grass

Two people walking on the beach with surfboards

A tennis player in action on the court

Two giraffes standing in a grassy field

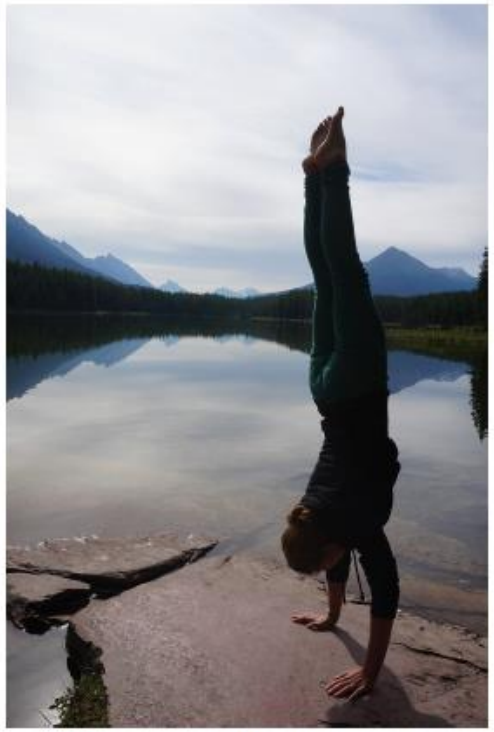
A man riding a dirt bike on a dirt track

Image Captioning: Failure Cases

Captions generated using neuraltalk2
All images are CC0 Public domain: fur coat, handstand, spider web, baseball

A woman is holding a cat in her hand

A person holding a computer mouse on a desk



A woman standing on a beach holding a surfboard

A bird is perched on a tree branch

A man in a baseball uniform throwing a ball

Visual Question Answering (VQA)

Q: What endangered animal is featured on the truck?

A: **A bald eagle.**
A: A sparrow.
A: A humming bird.
A: A raven.

Q: Where will the driver go if turning right?

A: **Onto 24 ¾ Rd.**
A: Onto 25 ¾ Rd.
A: Onto 23 ¾ Rd.
A: Onto Main Street.

Q: When was the picture taken?

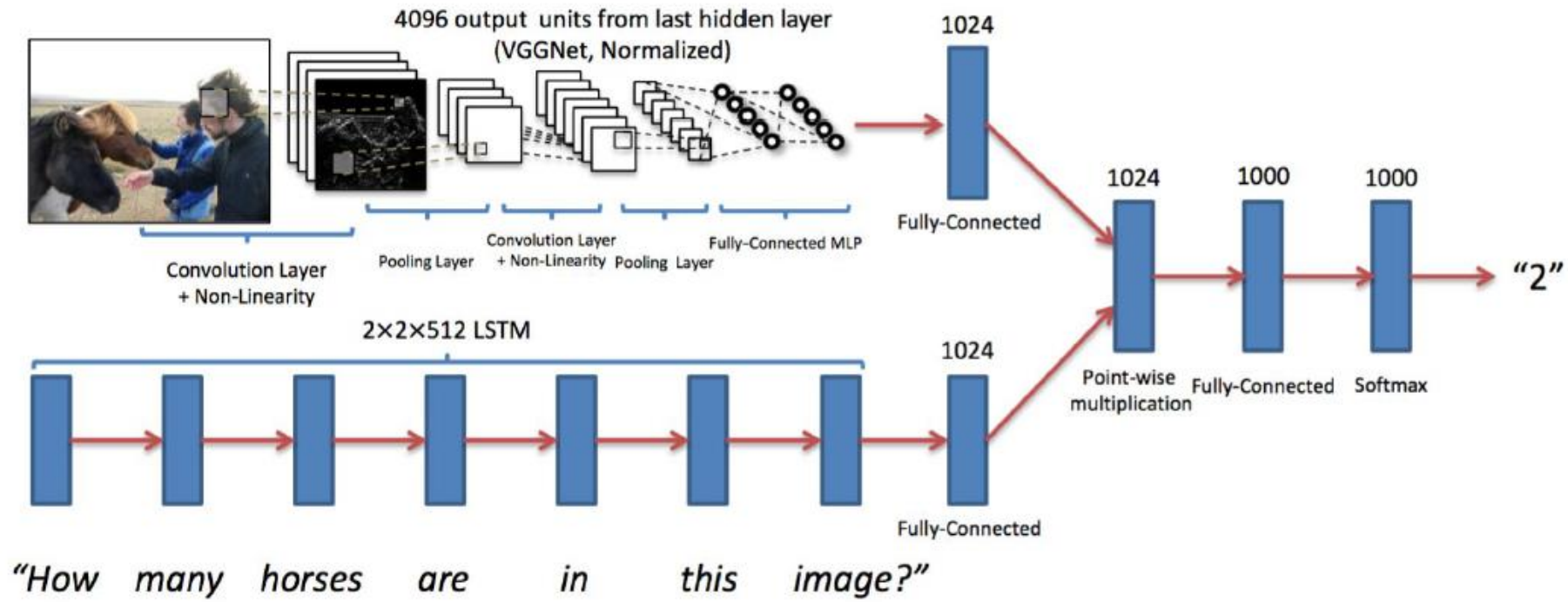
A: **During a wedding.**
A: During a bar mitzvah.
A: During a funeral.
A: During a Sunday church service

Q: Who is under the umbrella?

A: **Two women.**
A: A child.
A: An old man.
A: A husband and a wife.

Agrawal et al, "VQA: Visual Question Answering", ICCV 2015
Zhu et al, "Visual 7W: Grounded Question Answering in Images", CVPR 2016

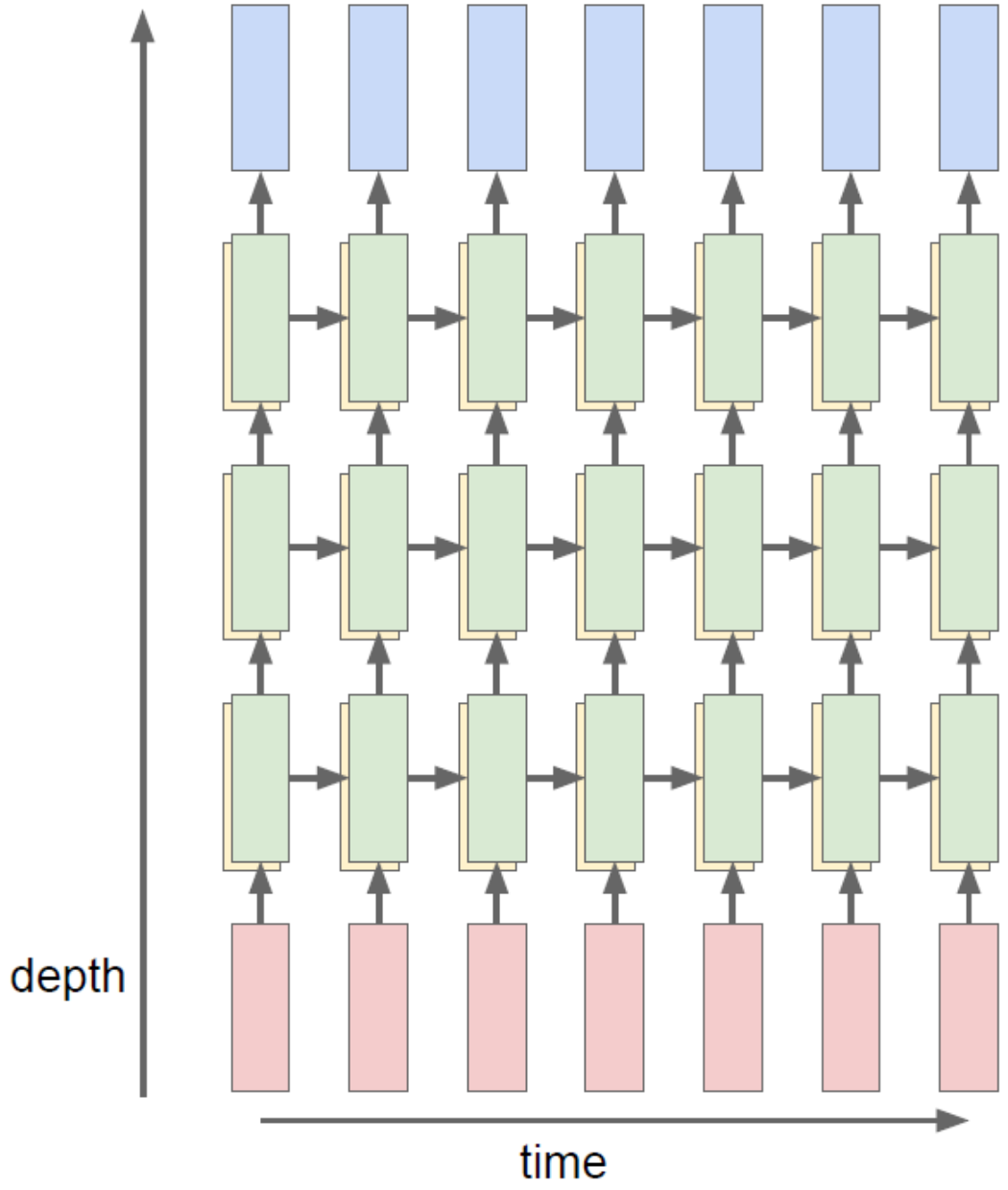
Visual Question Answering (VQA)



grawal et al, "Visual 7W: Grounded Question Answering in Images", CVPR 2015

Multilayer RNN

- RNNs are designed to capture dependencies in sequential data.
- However, when the vanishing gradient problem occurs, the ability of the network to learn and remember long-term dependencies diminishes.
- The gradients associated with earlier time steps become too small for meaningful updates, leading to a loss of information over time.



The Problem of Long-Term Dependencies

- Sometimes, we only need to look at recent information to perform the present task.
- For example, consider a language model trying to predict the next word based on the previous ones.
- If we are trying to predict the last word in *"the clouds are in the sky,"* we don't need any further context – it's pretty obvious the next word is going to be sky.
- In such cases, where the gap between the relevant information and the place that it's needed is small, RNNs can learn to use the past information.

The Problem of Long-Term Dependencies

- But there are also cases where we need more context. Consider trying to predict the last word in the text "*I grew up in France ….. I speak fluent French.*"

- Recent information suggests that the next word is probably the name of a language, but if we want to narrow down which language, we need the context of France, from further back.

- It's entirely possible for the gap between the relevant information and the point where it is needed to become very large.

- Unfortunately, as that gap grows, RNNs become unable to learn to connect the information.

Long Short Term Memory (LSTM)

Vanilla RNN

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

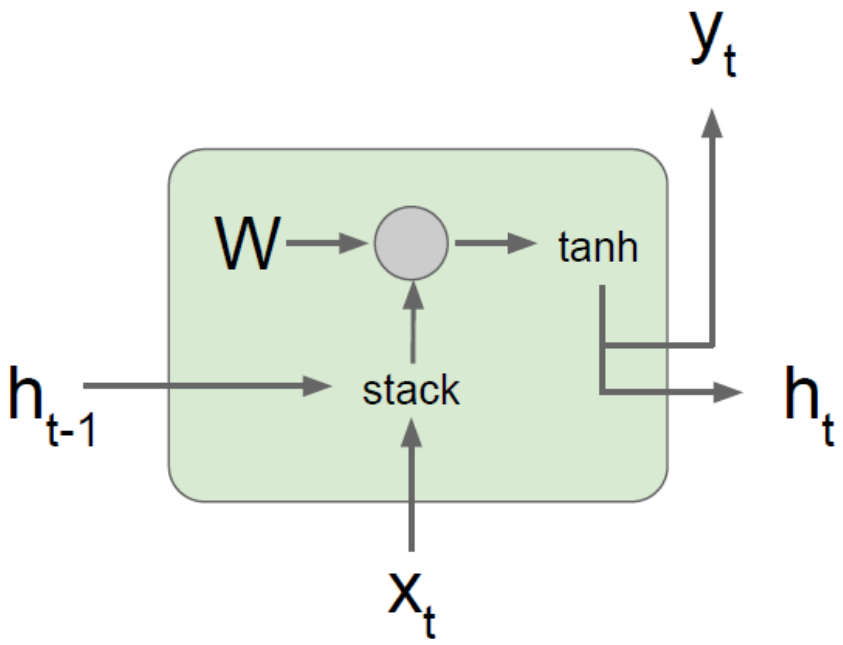
LSTM

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

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

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

Vanilla RNN



$$
\begin{aligned}
h_t &= \tanh(W_{hh}h_{t-1} + W_{xh}x_t) \\
&= \tanh\left(\begin{pmatrix} W_{hh} & W_{hx} \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right) \\
&= \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right)
\end{aligned}
$$

Long Short Term Memory (LSTM)

- Long Short Term Memory networks – usually just called “LSTMs” – are a special kind of RNN, capable of learning long-term dependencies.

- They were introduced by Hochreiter & Schmidhuber (1997), and were refined and popularized by many people in following work.

- At the heart of an LSTM lies a more complex architecture compared to a standard RNN. The key components include the **cell state**, **forget gate**, **input gate**, **output gate**, and various mechanisms designed to handle the flow of information.

Long Short Term Memory (LSTM)

Vanilla RNN

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

LSTM

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

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

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

Long Short Term Memory (LSTM)

Cell State (c or g):

The cell state acts as a memory unit that runs through the entire sequence. It allows the LSTM to maintain information over extended periods, enabling the network to learn and remember long-term dependencies.

Forget Gate (f):

The forget gate decides what information from the previous cell state should be discarded or retained. It takes into account the current input and the previous hidden state, producing values between 0 and 1. A value of 0 means "completely forget," while a value of 1 means "completely remember."

Long Short Term Memory (LSTM)

Input Gate (i):

The input gate determines what new information from the current input should be stored in the cell state. Similar to the forget gate, it processes the input and hidden state, deciding the proportion of new information to be added to the cell state.

Output Gate (o):

The output gate regulates what information from the cell state should be output to the next time step. It decides the hidden state for the current time step based on the input and hidden state, facilitating the flow of relevant information.

Long Short Term Memory (LSTM)

Vanilla RNN

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

LSTM

Four gates

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

Cell state

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

Hidden state

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

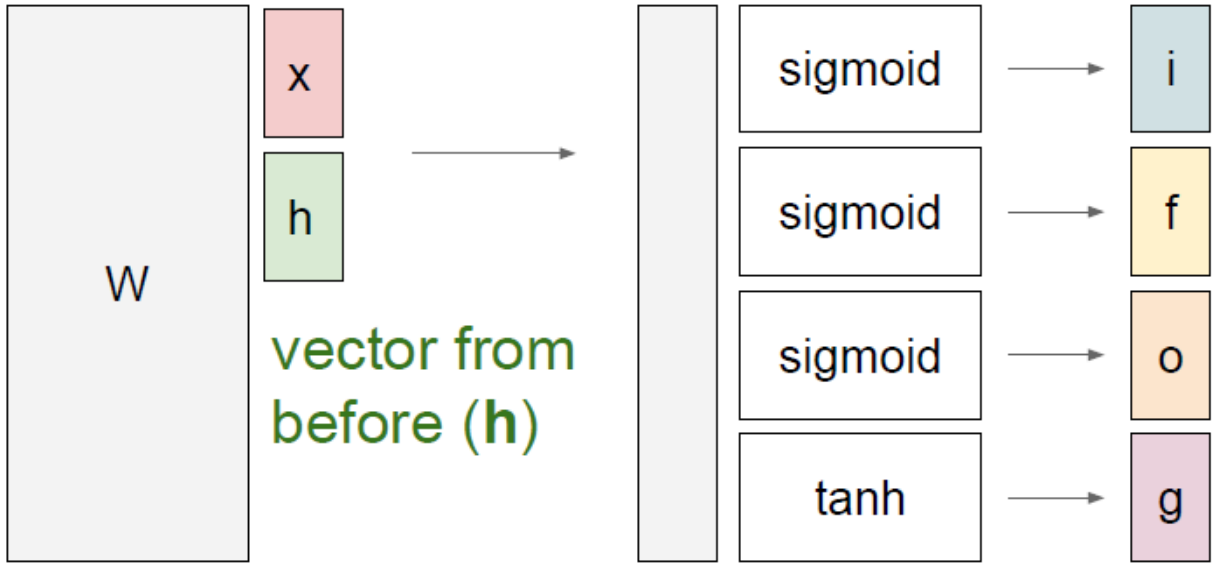
Long Short Term Memory (LSTM)

i: Input gate, whether to write to cell

f: Forget gate, Whether to erase cell

o: Output gate, How much to reveal cell

g: Gate gate! (Cell state), How much to write to cell

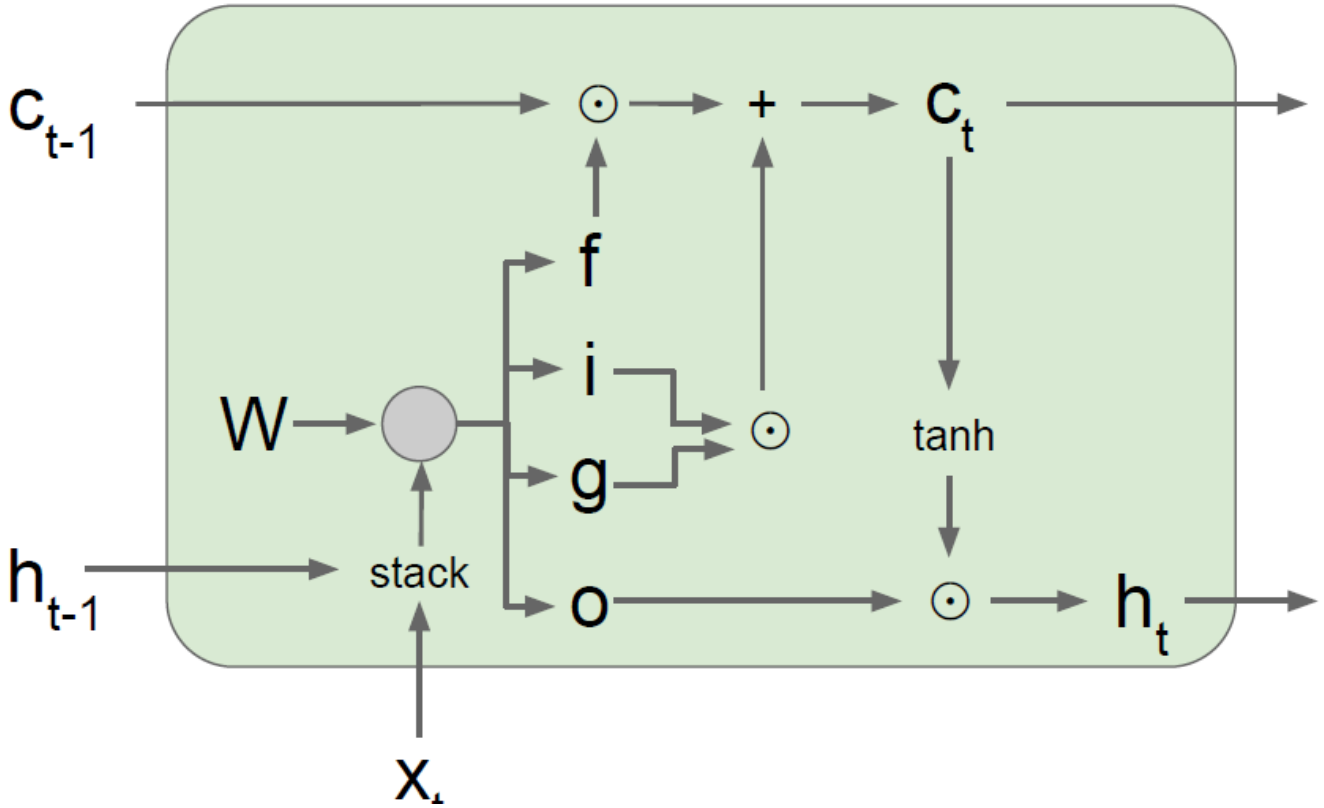


$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

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

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

Long Short Term Memory (LSTM)



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

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

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

Attention

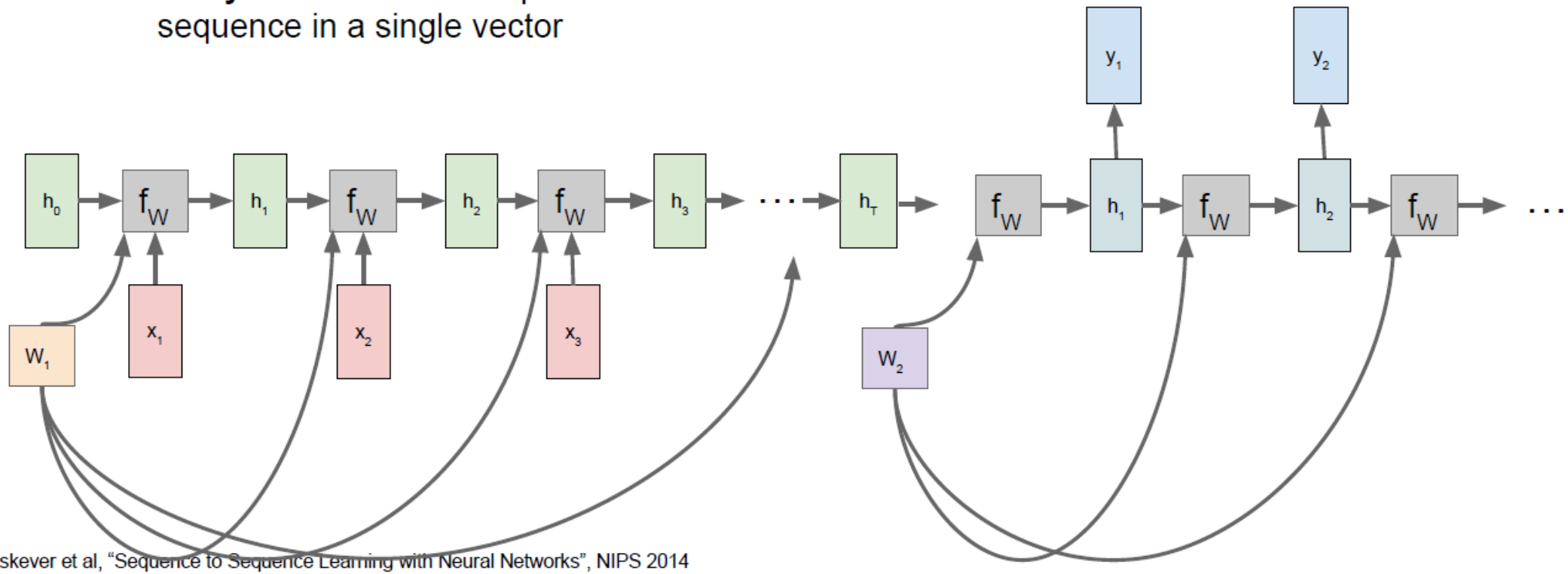
- Attention is a mechanism used in neural networks, particularly in natural language processing (NLP) and computer vision tasks, to focus on specific parts of input data while making predictions.

- The attention mechanism allows the model to dynamically weigh different parts of the input, giving more emphasis to relevant information.

Sequence to Sequence: Many-to-one + one-to-many

Many to one: Encode input sequence in a single vector

One to many: Produce output sequence from single input vector



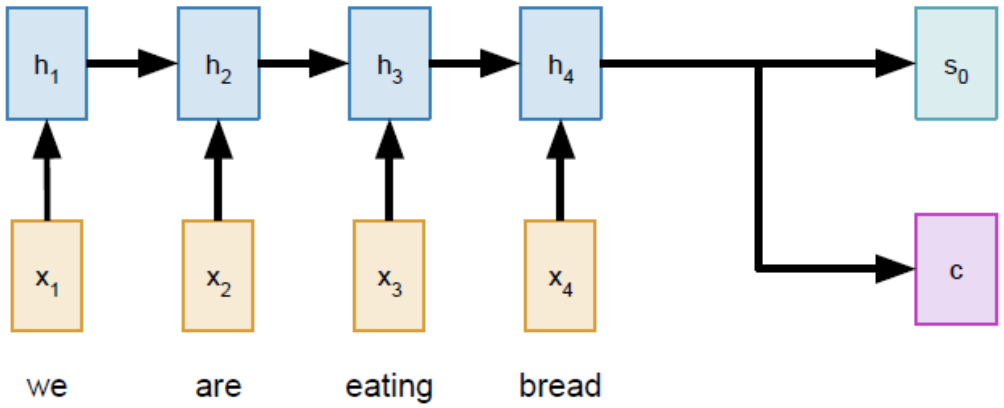
Sutskever et al, "Sequence to Sequence Learning with Neural Networks", NIPS 2014

Sequence to Sequence with RNNs

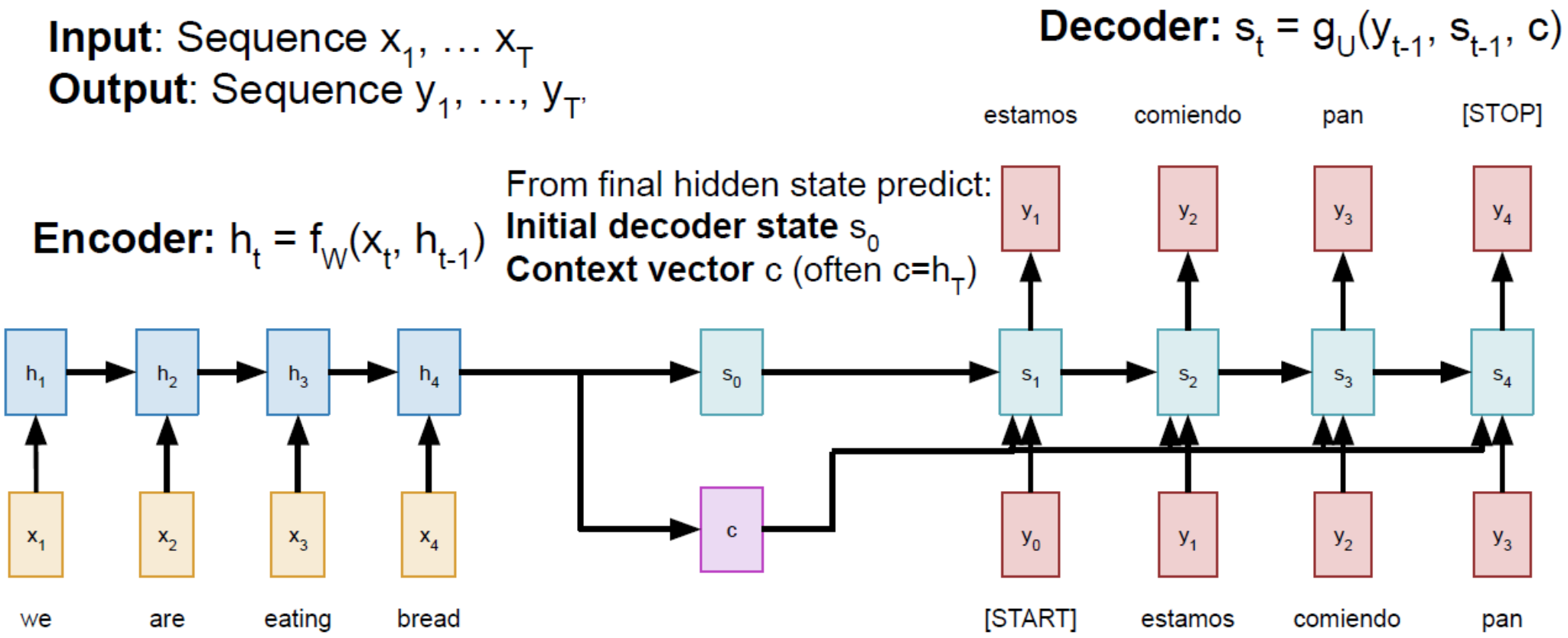
Input: Sequence $x_1, \ldots x_T$
Output: Sequence $y_1, \ldots, y_{T'}$

Encoder: $h_t = f_W(x_t, h_{t-1})$

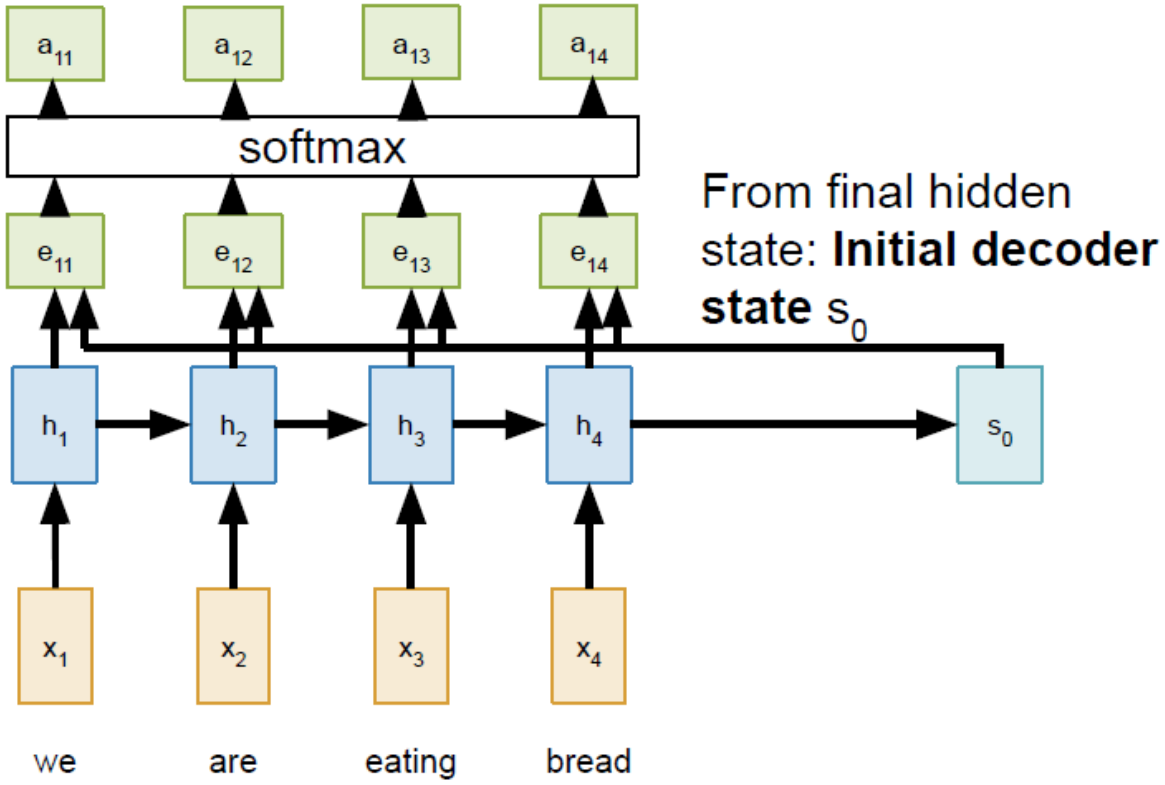
From final hidden state predict:
Initial decoder state s_0
Context vector c (often $c=h_T$)



Sequence to Sequence with RNNs



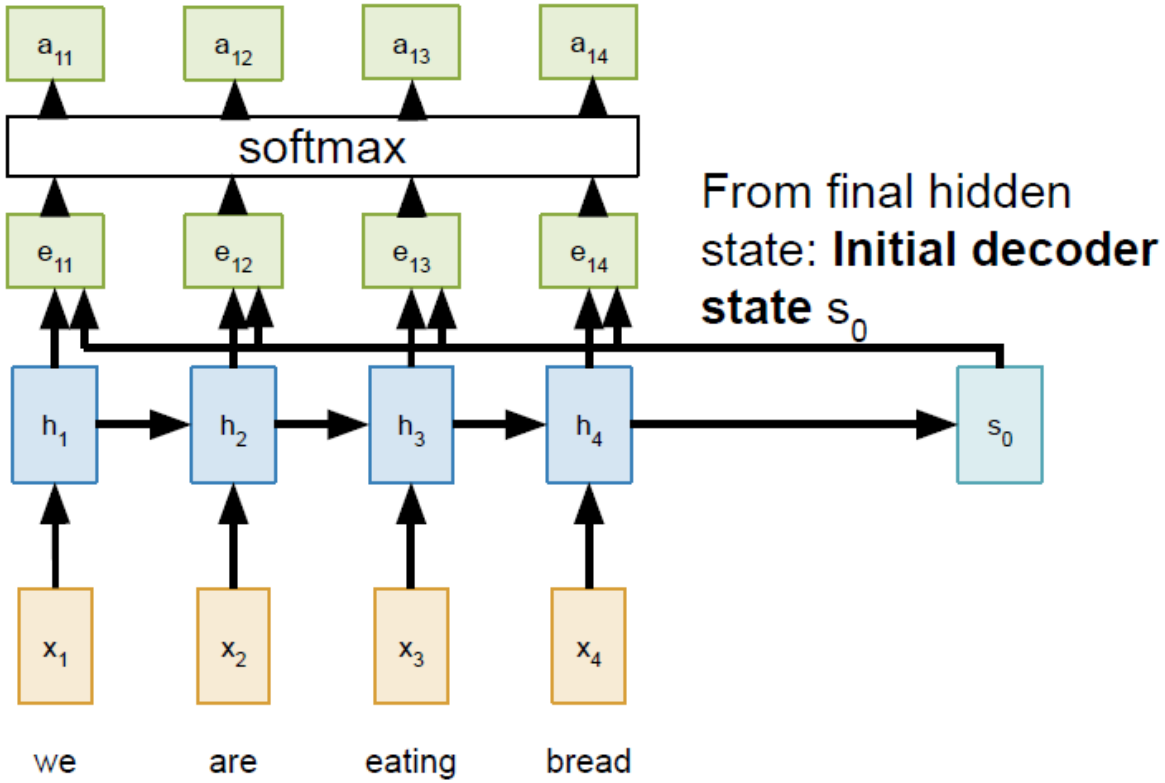
Sequence to Sequence with RNNs and attention



Alignment scores

$e_{t,i} = f_{att}(s_{t\_1}, h_i)$

Sequence to Sequence with RNNs and attention



Alignment scores

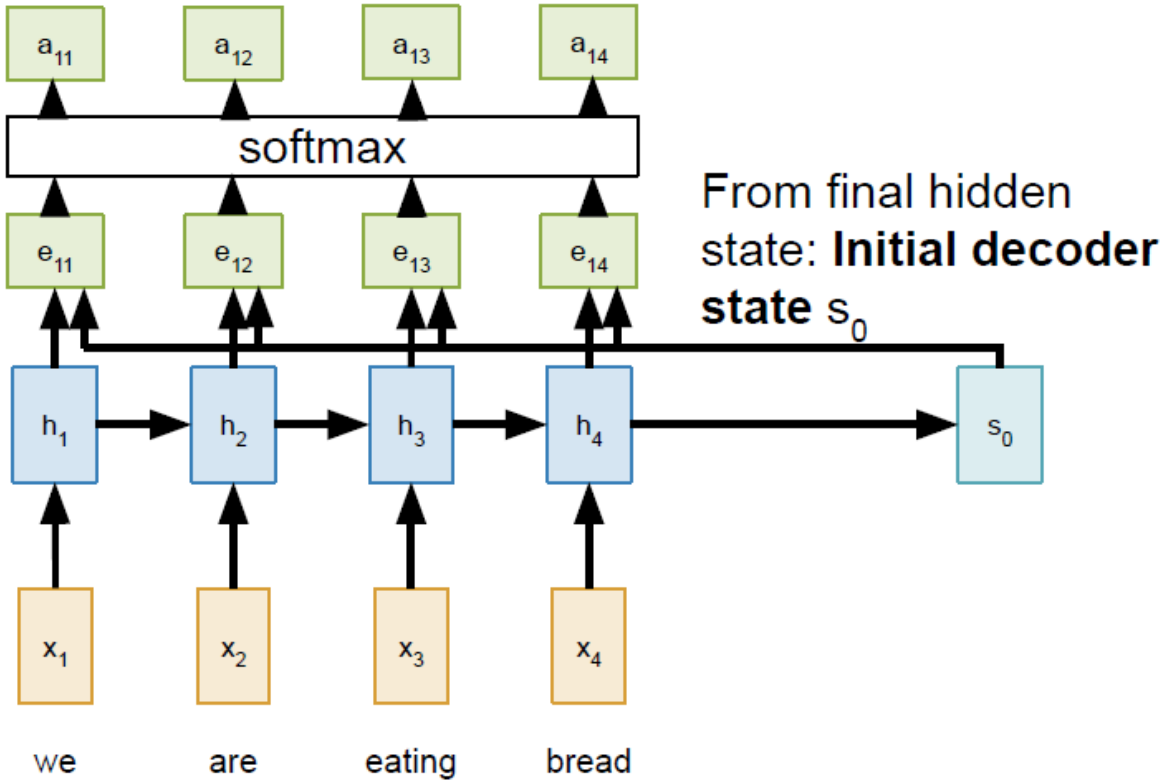
$$e_{t,i} = f_{att}(s_{t\_1}, h_i)$$

Weights

Normalize alignment scores

$$a_{t,i} = softmax(e_{t,i})$$

Sequence to Sequence with RNNs and attention



Alignment scores

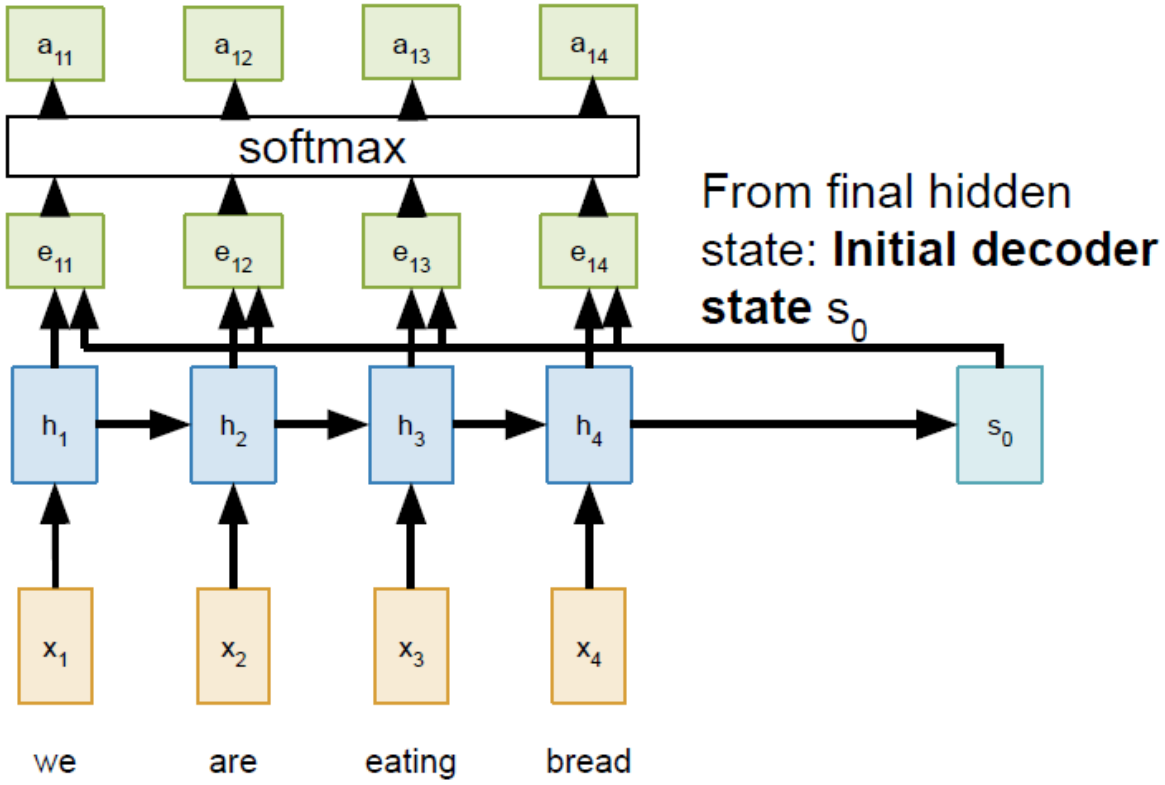
$$e_{t,i} = f_{att}(s_{t\_1}, h_i)$$

Weights

Normalize alignment scores

$$a_{t,i} = softmax(e_{t,i})$$

Sequence to Sequence with RNNs and attention



Alignment scores

$$e_{t,i} = f_{att}(s_{t\_1}, h_i)$$

Weights

Normalize alignment scores

$$a_{t,i} = softmax(e_{t,i})$$

Context Vector

$$c_t = \sum_{i=1}^{T} a_{t,i}\, h_i$$

Sequence to Sequence with RNNs and attention

- **Alignment scores**: The alignment model takes the encoded hidden states h_i, and the previous decoder output s_{t_1}, to compute a score $e_{t,i}$, that indicates how well the elements of the input sequence align with the current output at the position

- **Weights**: The weights $a_{t,i}$, are computed by applying a softmax operation to the previously computed alignment scores

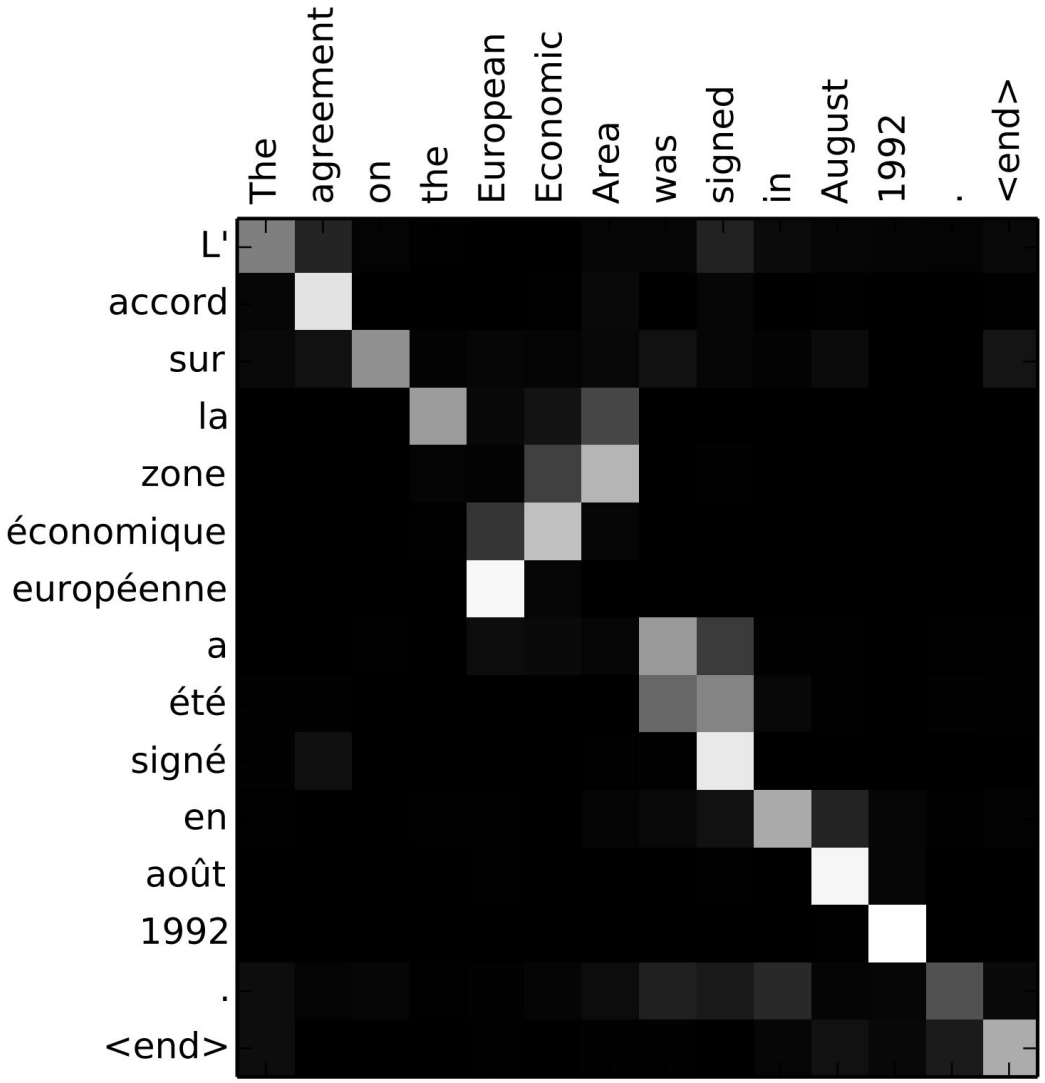
- **Context vector**: A unique context vector c_t, is fed into the decoder at each time step. It is computed by a weighted sum of all encoder hidden states

Visualize attention weights (Translation)

- **Example**: English to French
- Translation

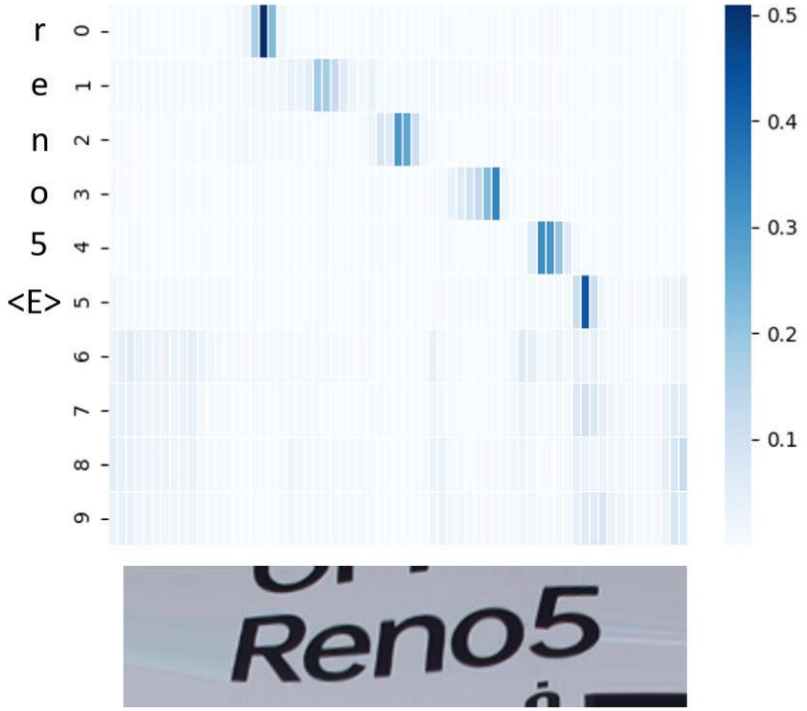
- **Input**: “The agreement on the European Economic Area was signed in August 1992.”

- **Output**: “L’accord sur la zone économique européenne a été signé en août 1992.”



Visualize attention weights (Text Recognition)

- Attention weights for Arabic and English examples using a Bilingual model.



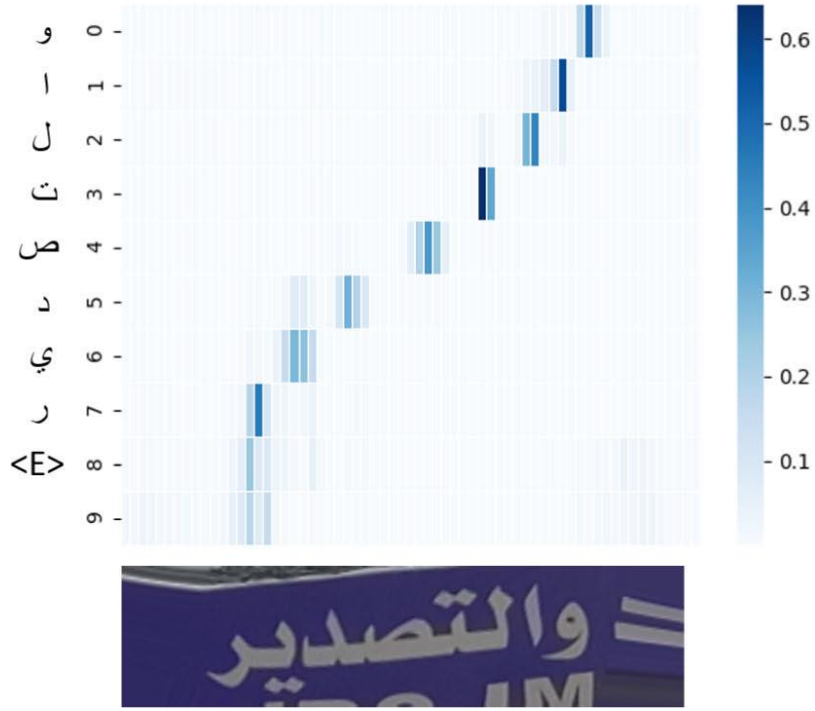


Image Captioning with Attention

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

Which problem is associated with traditional RNNs and is mitigated by Long Short-Term Memory (LSTM) networks?

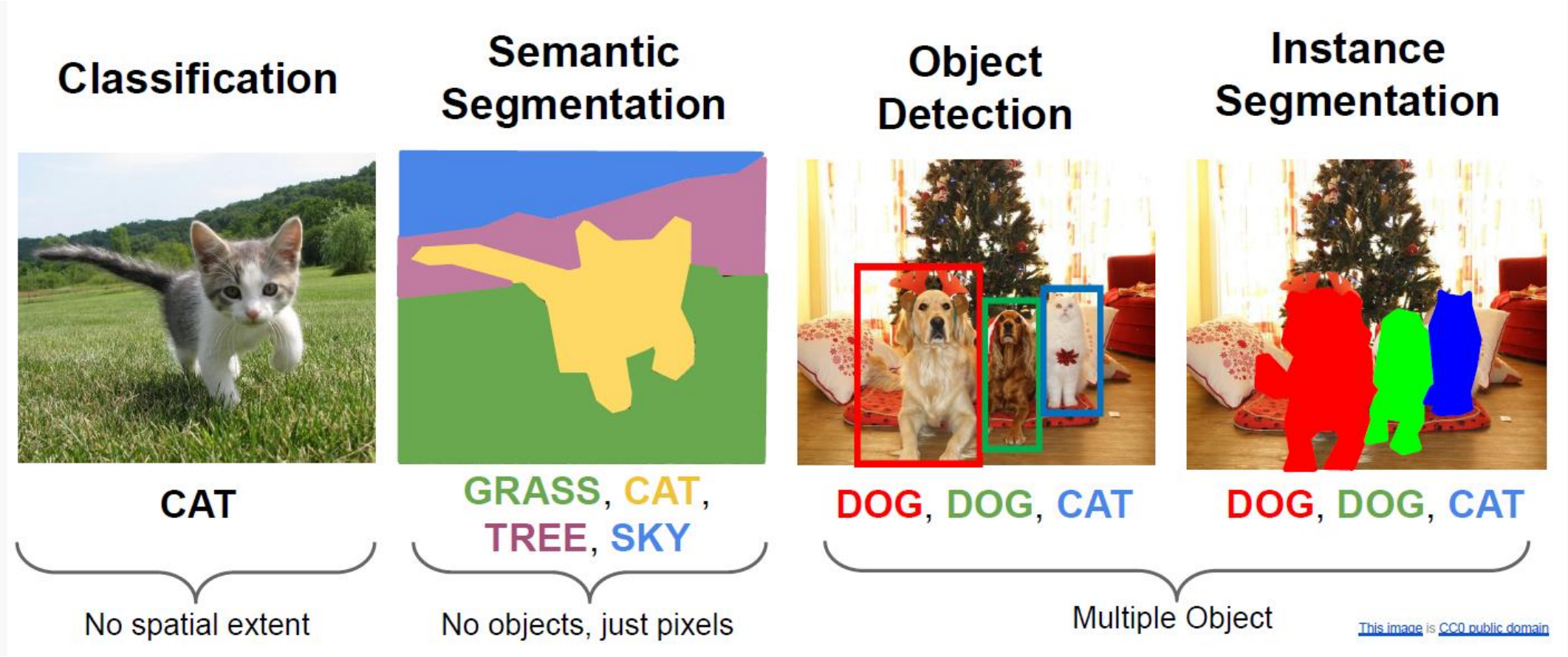
A) Overfitting
B) Vanishing gradient
C) Underfitting
D) Exploding gradient

What is the function of the forget gate in an LSTM network?

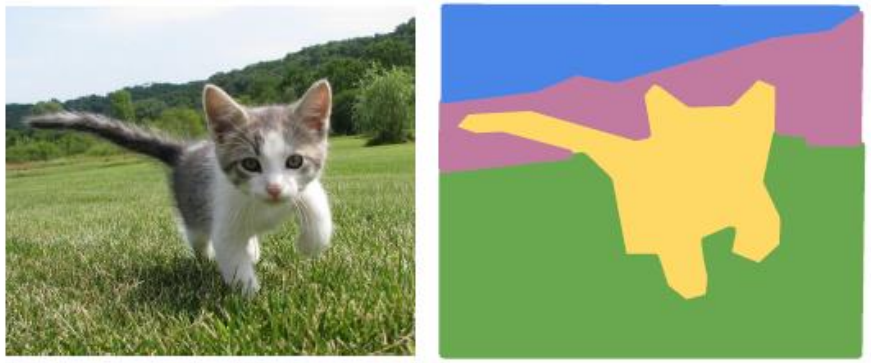
A) Determines new information to be stored
B) Regulates output information
C) Decides what information to discard from the cell state
D) Updates candidate cell state

Object Detection and Image Segmentation

Computer Vision Tasks

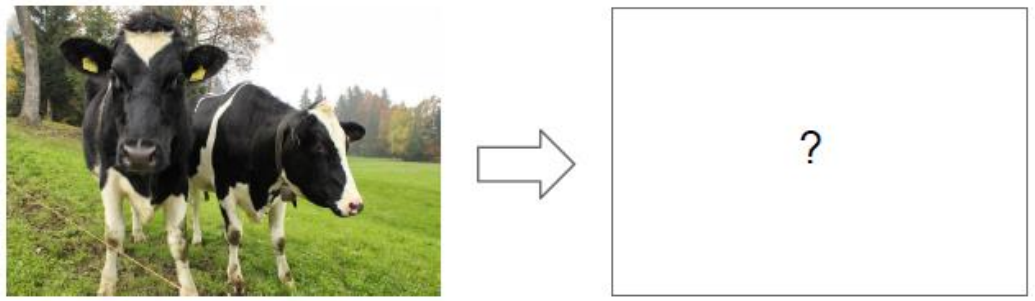


Semantic Segmentation



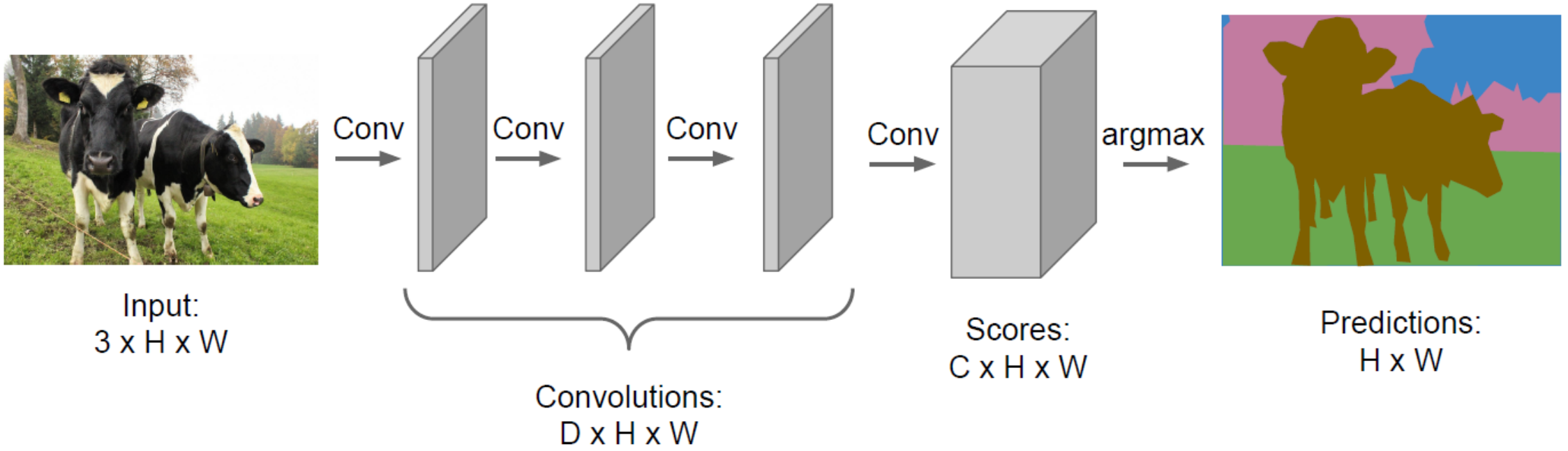
GRASS, **CAT**,
TREE, **SKY**, **...**

Paired training data: for each training image, each pixel is labeled with a semantic category.

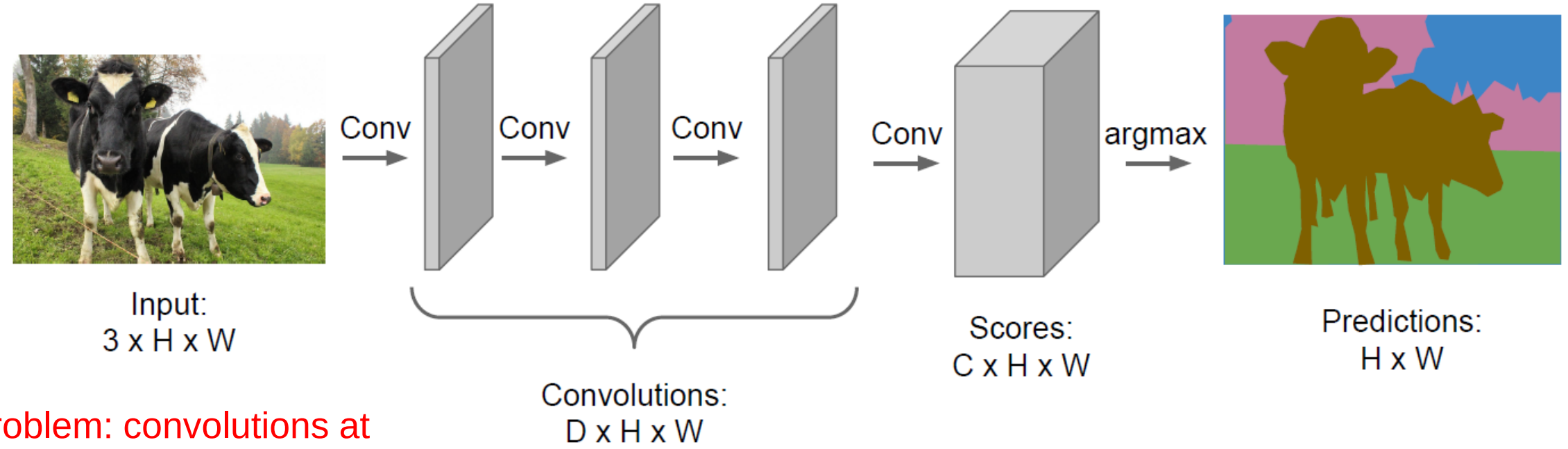


At test time, classify each pixel of a new image.

Semantic Segmentation Idea: Fully Convolutional



Semantic Segmentation Idea: Fully Convolutional

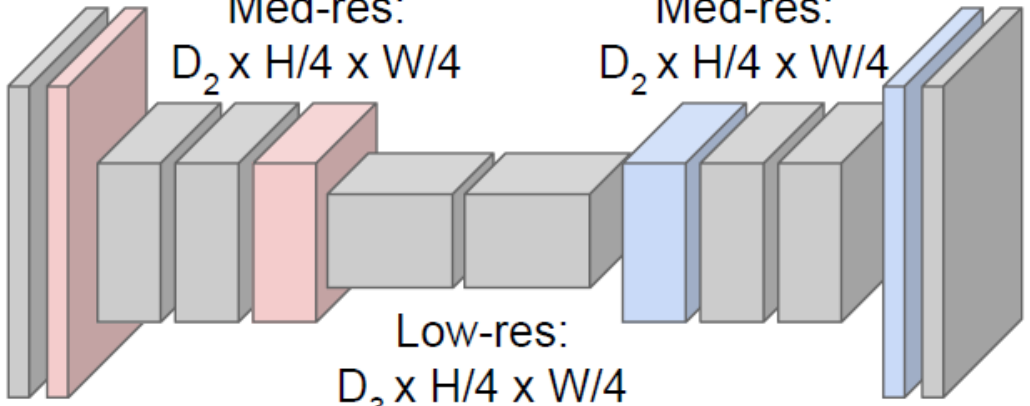


Problem: convolutions at original image resolution will be very expensive ...

Semantic Segmentation Idea: Fully Convolutional

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Input:
3 x H x W



High-res:
D_1 x H/2 x W/2

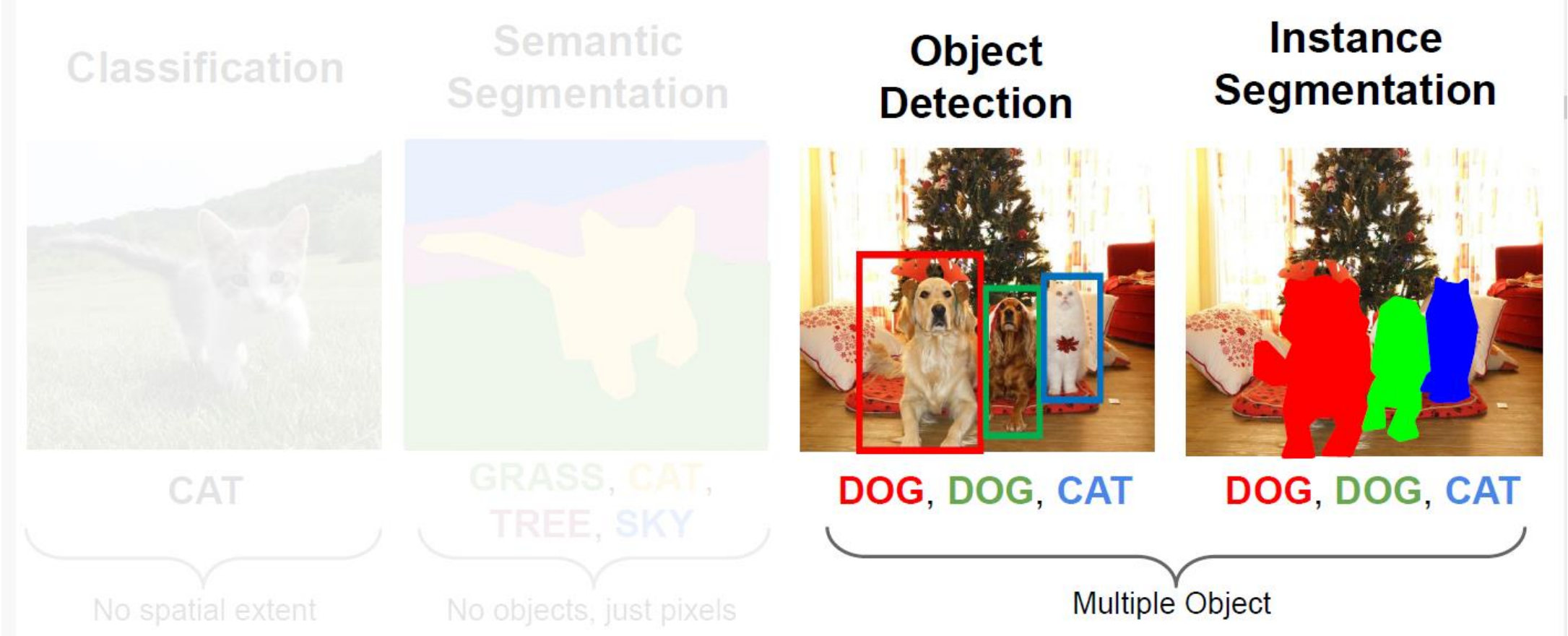
High-res:
D_1 x H/2 x W/2

C x H x W

Predictions:
H x W

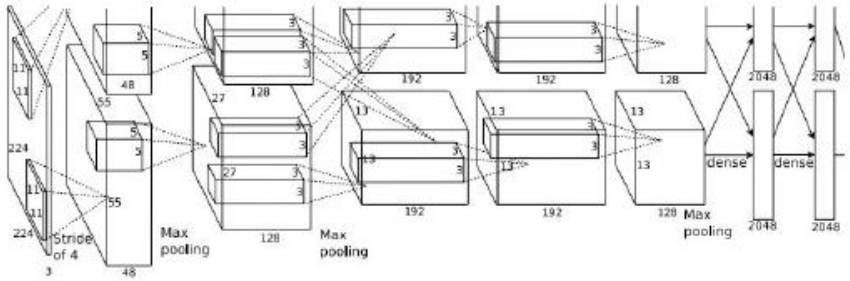
Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

Object Detection



Object Detection

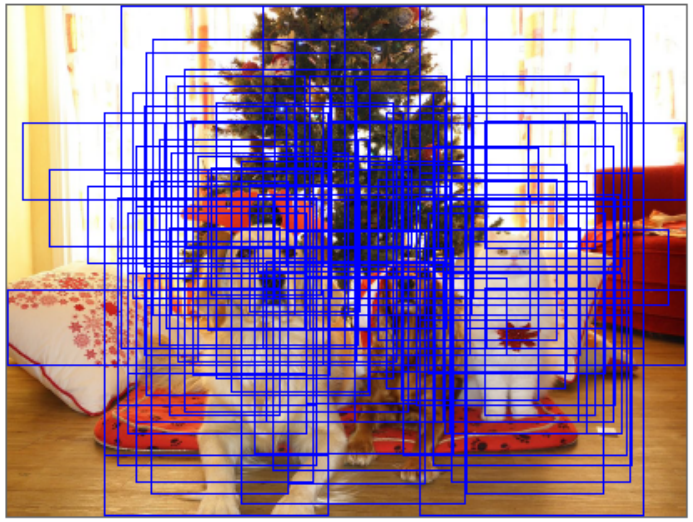
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



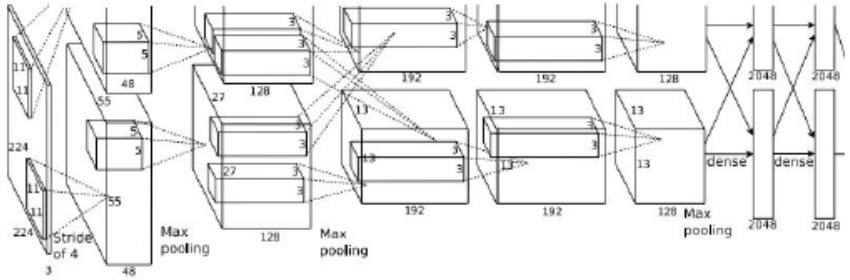
Dog? NO
Cat? YES
Background? NO

Q: What's the problem with this approach?

Object Detection



Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

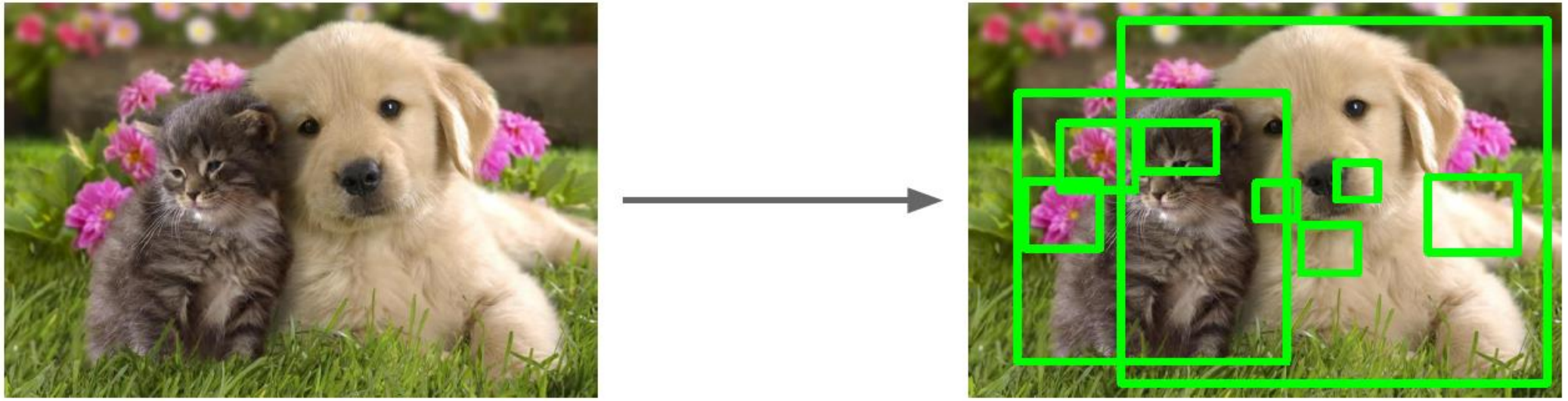


Dog? NO
Cat? YES
Background? NO

Problem: Need to apply CNN to huge number of locations, scales, and aspect ratios, very computationally expensive!

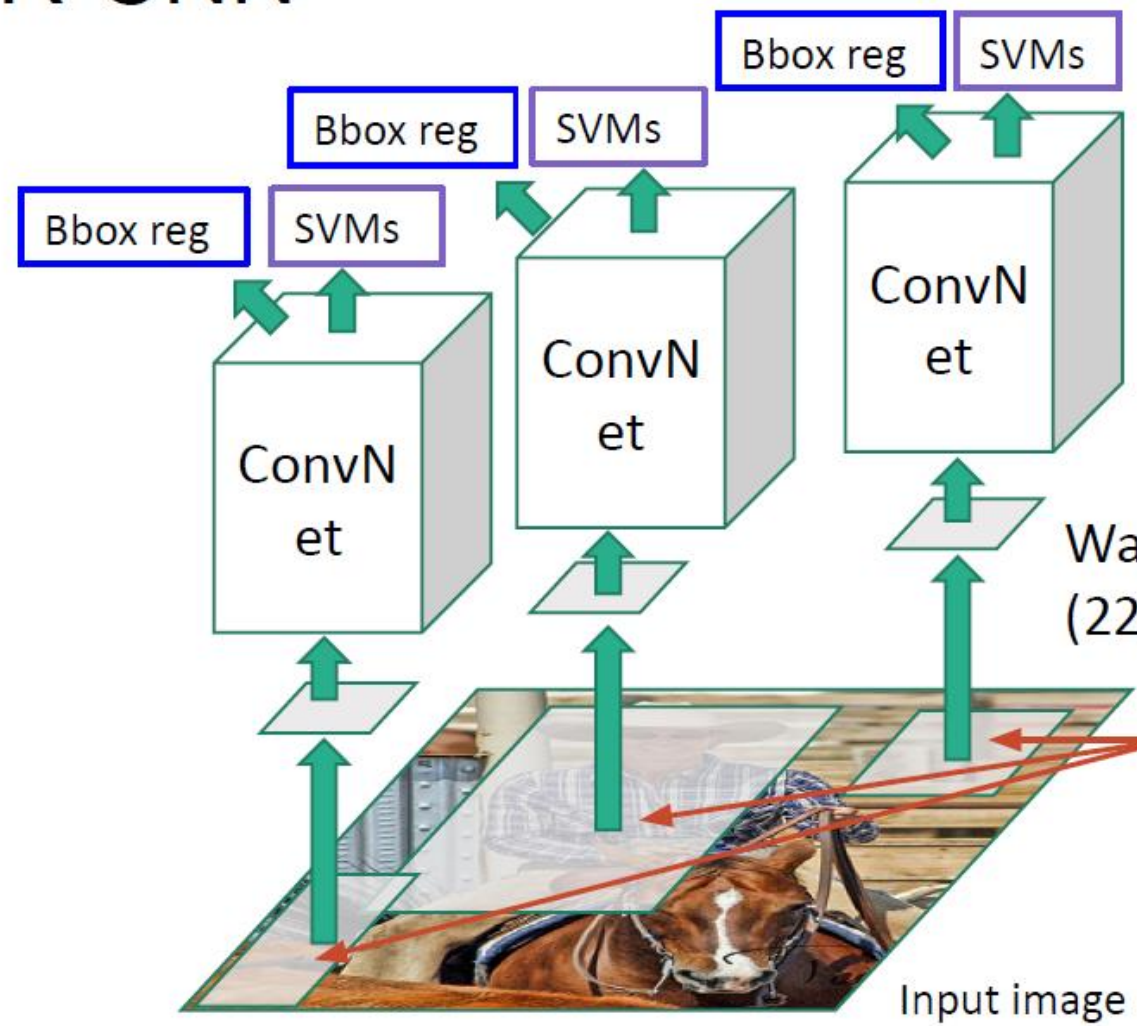
Region Proposals: Selective Search

- Find “blobby” image regions that are likely to contain objects
- Relatively fast to run; e.g. Selective Search gives 2000 region proposals in a few seconds on CPU



R-CNN

Predict "corrections" to the RoI: 4 numbers: (dx, dy, dw, dh)



Classify regions with SVMs

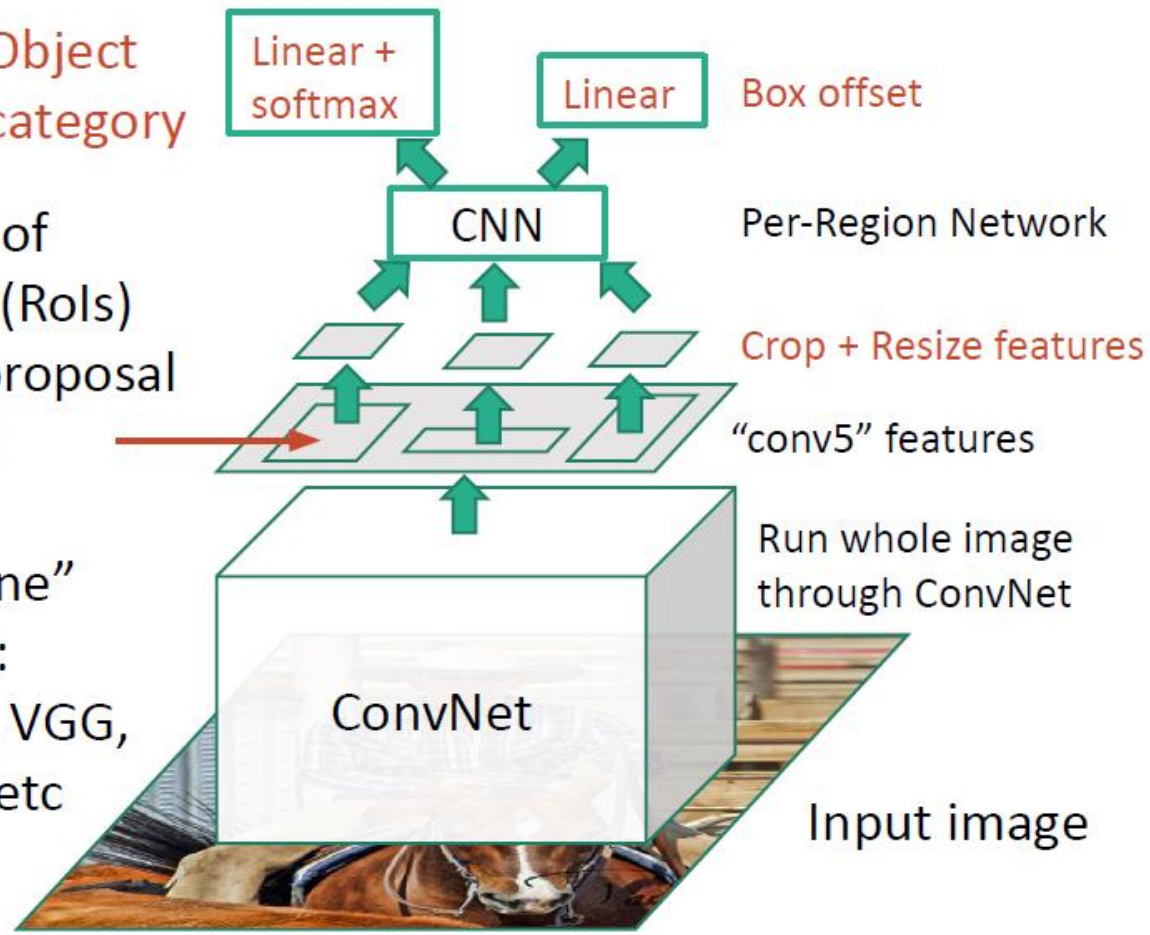
Forward each region through ConvNet (ImageNet-pretranied)

Warped image regions (224x224 pixels)

Regions of Interest (RoI) from a proposal method (~2k)

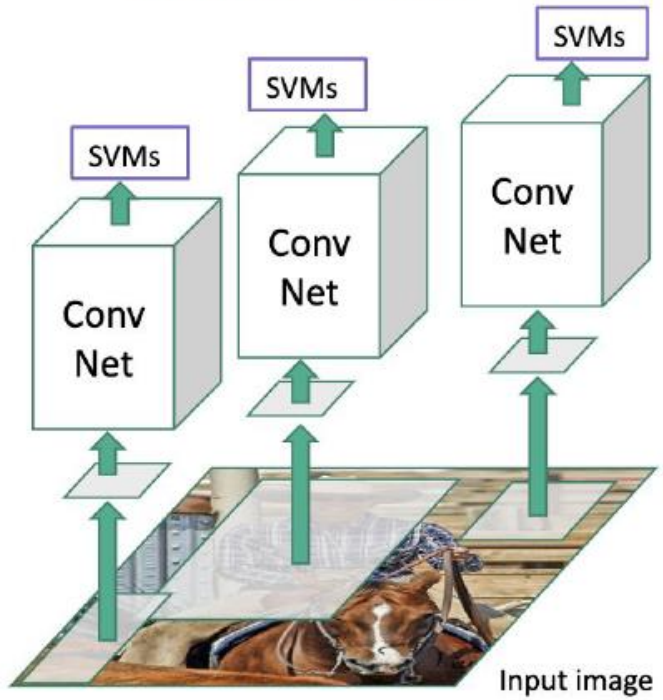
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

Fast R-CNN



Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

"Slow" R-CNN

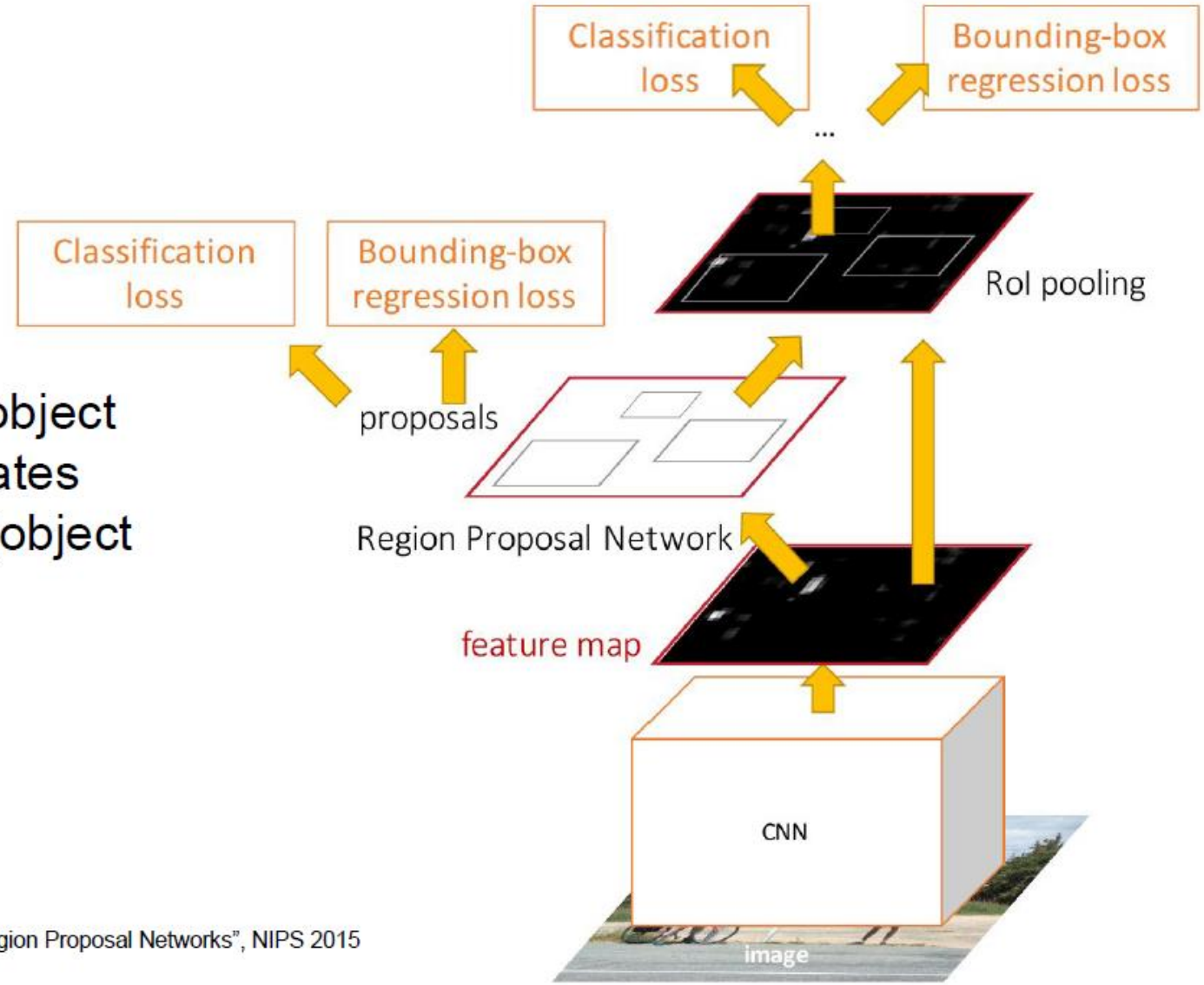


Faster R-CNN:

Make CNN do proposals!

Jointly train with 4 losses:

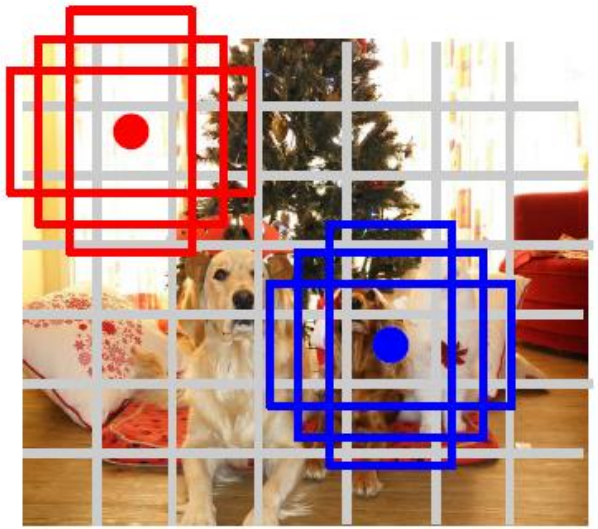
1. RPN classify object / not object
2. RPN regress box coordinates
3. Final classification score (object classes)
4. Final box coordinates



Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015
Figure copyright 2015, Ross Girshick; reproduced with permission

Single-Stage Object Detectors: YOLO / SSD / RetinaNet

Input image
3 x H x W



Divide image into grid
7 x 7

Image a set of **base boxes** centered at each grid cell
Here B = 3

Within each grid cell:

- Regress from each of the B base boxes to a final box with 5 numbers: (dx, dy, dh, dw, confidence)
- Predict scores for each of C classes (including background as a class)
- Looks a lot like RPN, but category-specific!

Output:
7 x 7 x (5 * B + C)

Redmon et al, "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016
Liu et al, "SSD: Single-Shot MultiBox Detector", ECCV 2016
Lin et al, "Focal Loss for Dense Object Detection", ICCV 2017

What is the goal of semantic segmentation in computer vision?

A) Detecting objects and their boundaries

B) Classifying objects in an image

C) Assigning a label to each pixel in an image

D) Estimating depth information

What is the Region Proposal Network (RPN) used for in object detection?

A) Proposing candidate regions for objects
B) Extracting features from objects
C) Classifying objects
D) Regressing bounding box coordinates