Machine Learning I
MATH80629A

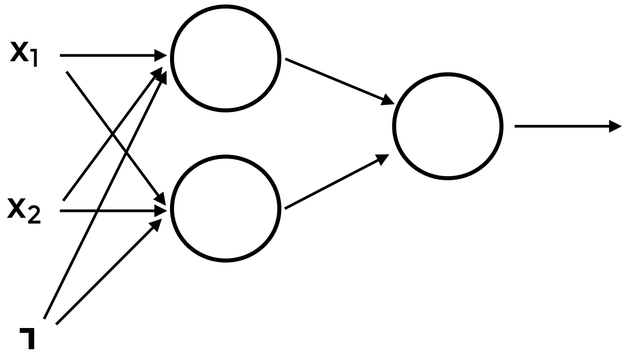
Apprentissage Automatique I
MATH80629

Recurrent and Convolutional Networks
— Week #6

Neural Networks for Modelling Sequential Data

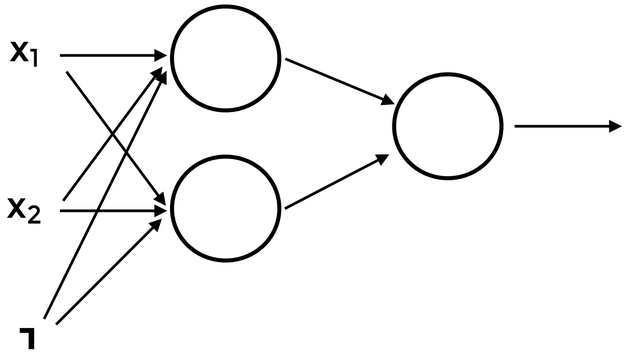
Neural Network models (architectures)

- Feed-forward neural networks are standard
  - Input & Output: Fixed-length
  - Data is processed in parallel



Neural Network models (architectures)

- Feed-forward neural networks are standard
  - Input & Output: Fixed-length
  - Data is processed in parallel
- We can “specialize” neural networks
  - Different data have different characteristics



Sequential Data

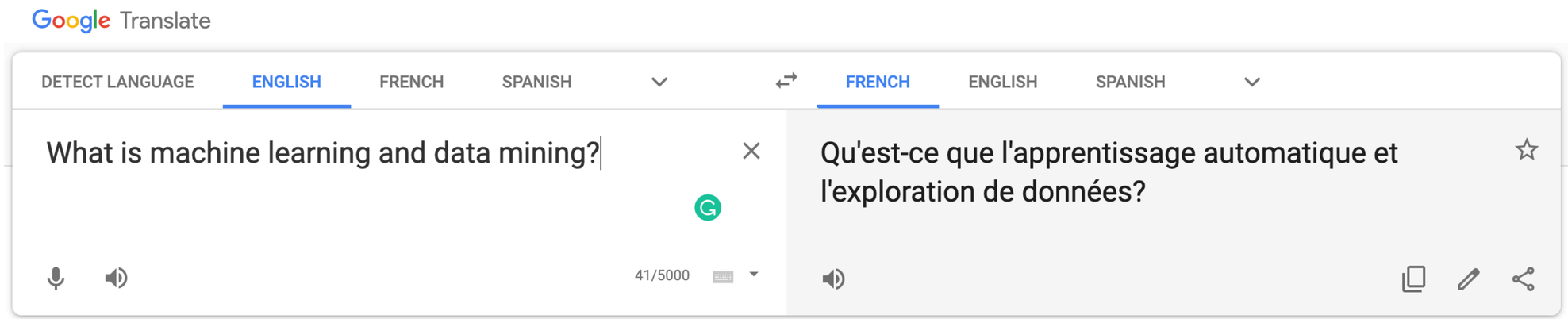
- Data with temporal coherence
- Not fixed length
- For example:
  - Text (document)
  - Speech
  - Stock prices
  - Sensor measurements through time
  - Videos

Text Classification

```
From: bcash@crchh410.NoSubdomain.NoDomain (Brian Cash)
Subject: Re: free moral agency
Nntp-Posting-Host: crchh410
Organization: BNR, Inc.
Lines: 17

In article <735295730.25282@minster.york.ac.uk>, cjhs@minster.york.ac.uk writes:
|> : Are you saying that their was a physical Adam and Eve, and that all
|> : humans are direct decendents of only these two human beings.?  Then who
|> : were Cain and Able's wives?  Couldn't be their sisters, because A&E
|> : didn't have daughters.  Were they non-humans?
|>
|> Genesis 5:4
|>
|> and the days of Adam after he begat Seth were eight hundred years, and
|> he begat sons and daughters:
|>
|> Felicitations -- Chris Ho-Stuart

Yeah, but these were not the wives.  The wives came from Nod, apparently
a land being developed by another set of gods.

Brian /-|-\
```

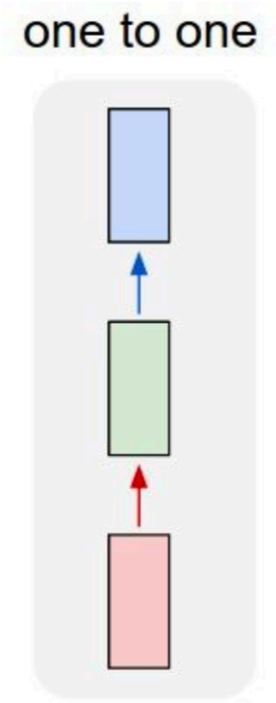
SPAM / HAM

Translation



: A Layer

Task classification



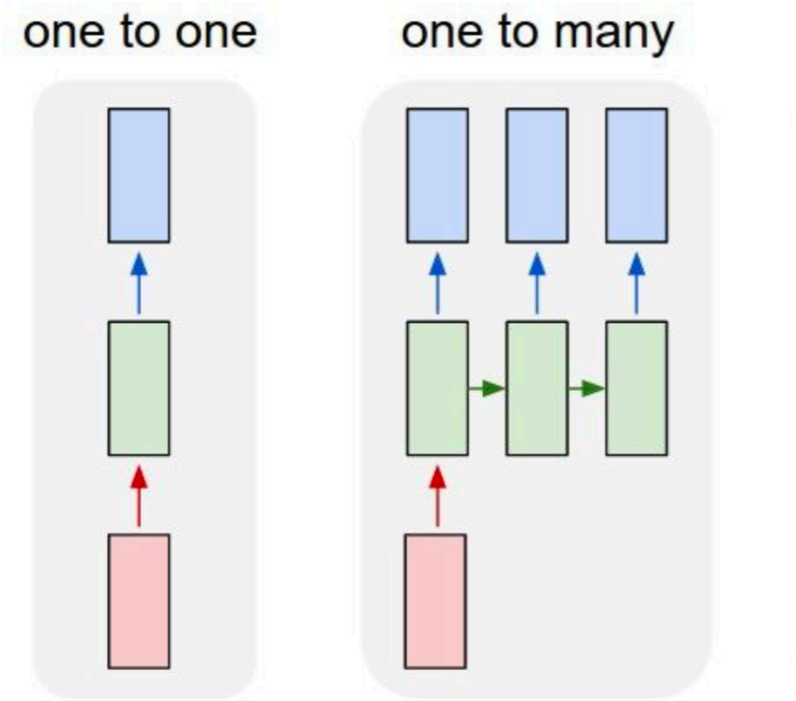
Object Classification

Image -> Class

[Image from http://vision.stanford.edu/teaching/cs231n/slides/2016/winter1516_lecture10.pdf]

: A Layer

Task classification



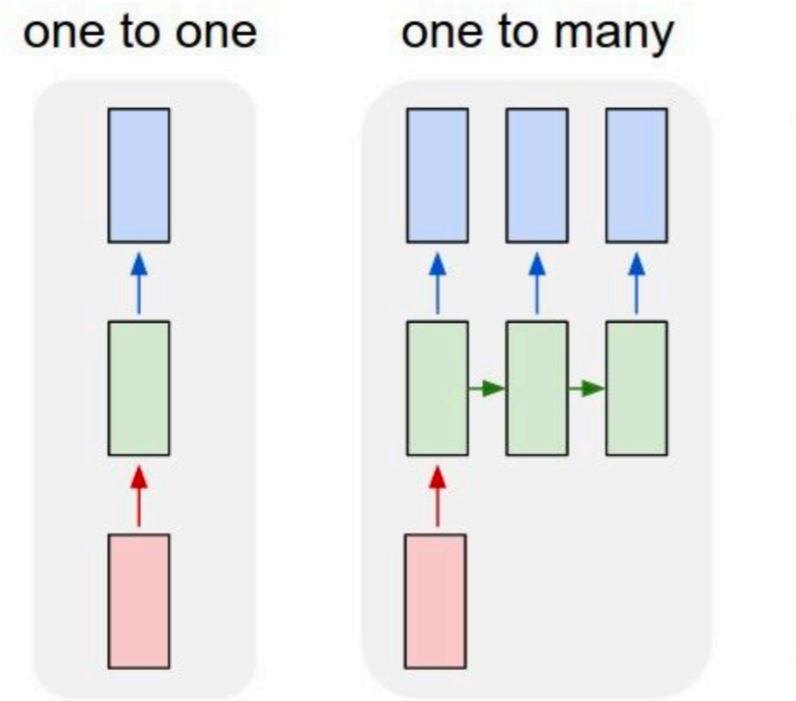
Object Classification

Image -> Class

[Image from http://vision.stanford.edu/teaching/cs231n/slides/2016/winter1516_lecture10.pdf]

: A Layer

Task classification



Object Classification

Image -> Class

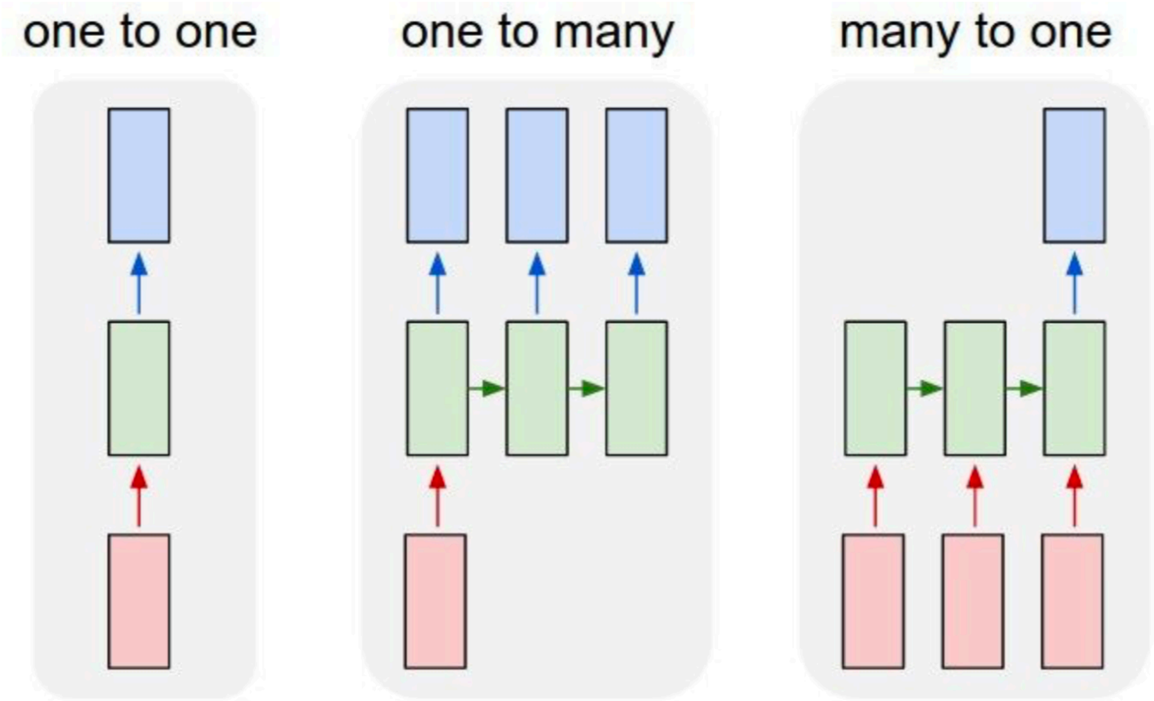
Image Captioning

Image -> Caption

[Image from http://vision.stanford.edu/teaching/cs231n/slides/2016/winter1516_lecture10.pdf]

: A Layer

Task classification



Object Classification

Image Captioning

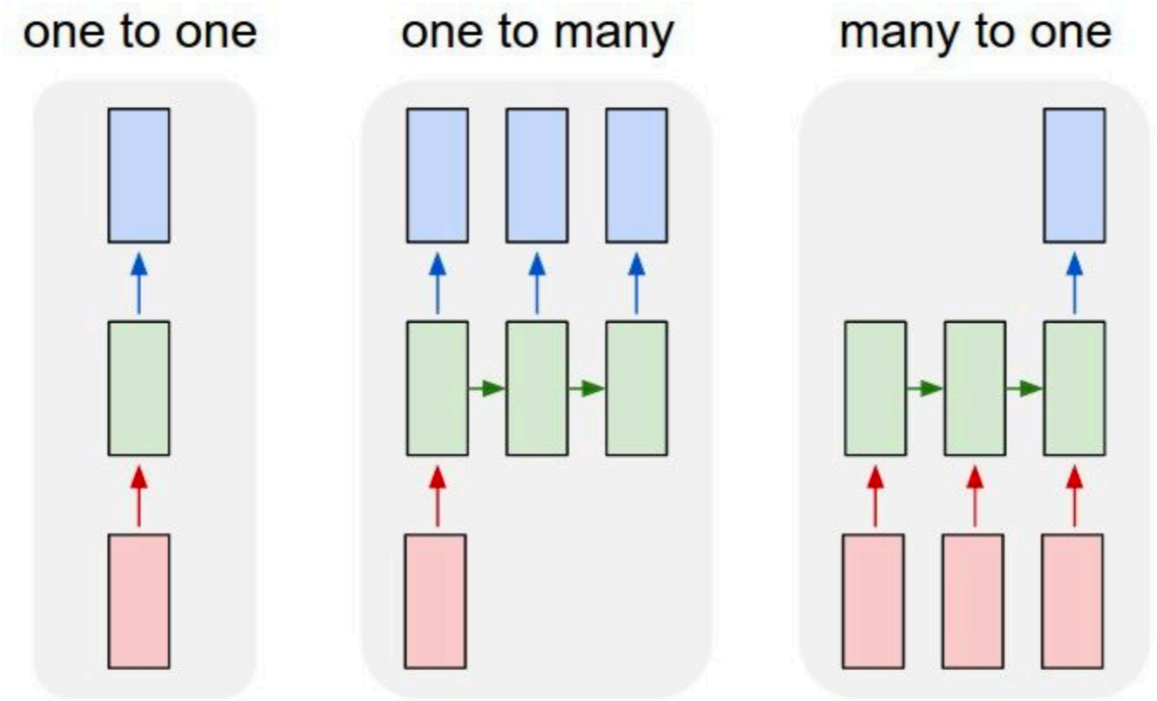
Image -> Class

Image -> Caption

[Image from http://vision.stanford.edu/teaching/cs231n/slides/2016/winter1516_lecture10.pdf]

: A Layer

Task classification



Object Classification

Image Captioning

Sentiment Analysis

Image -> Class

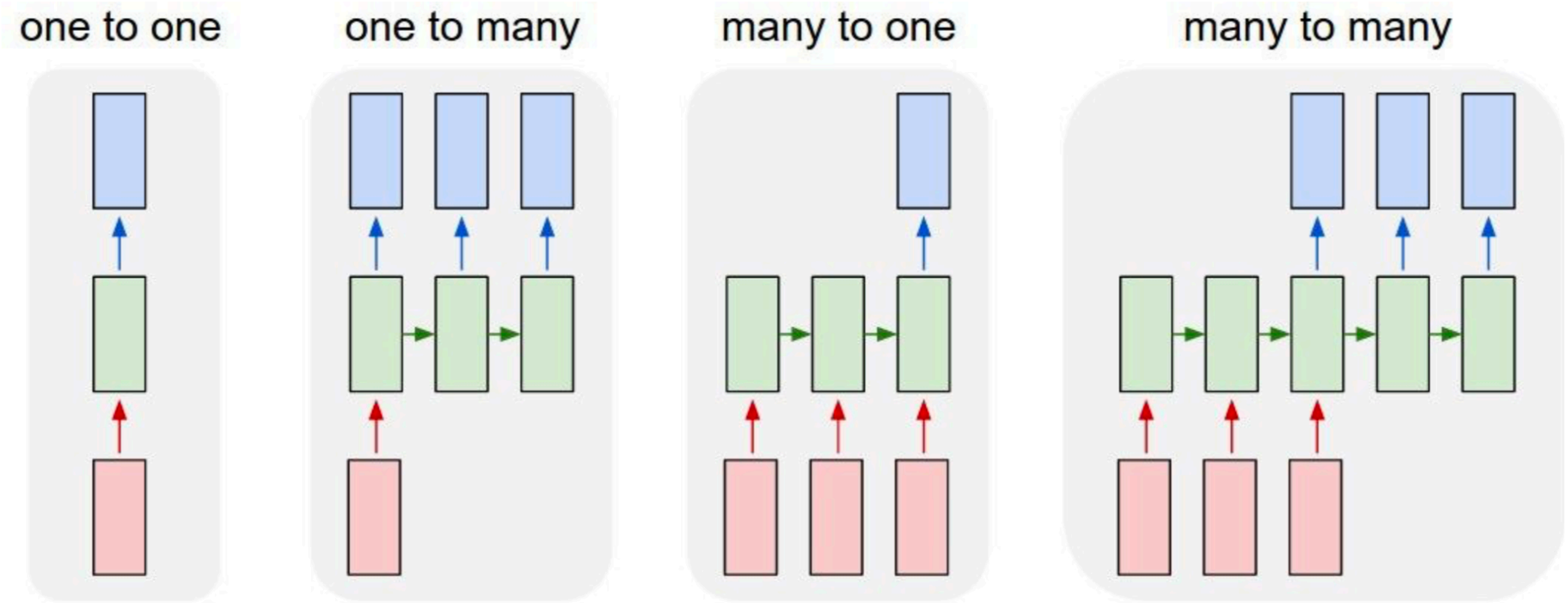
Image -> Caption

Text -> Sentiment

[Image from http://vision.stanford.edu/teaching/cs231n/slides/2016/winter1516_lecture10.pdf]

: A Layer

Task classification



Object Classification

Image Captioning

Sentiment Analysis

Image -> Class

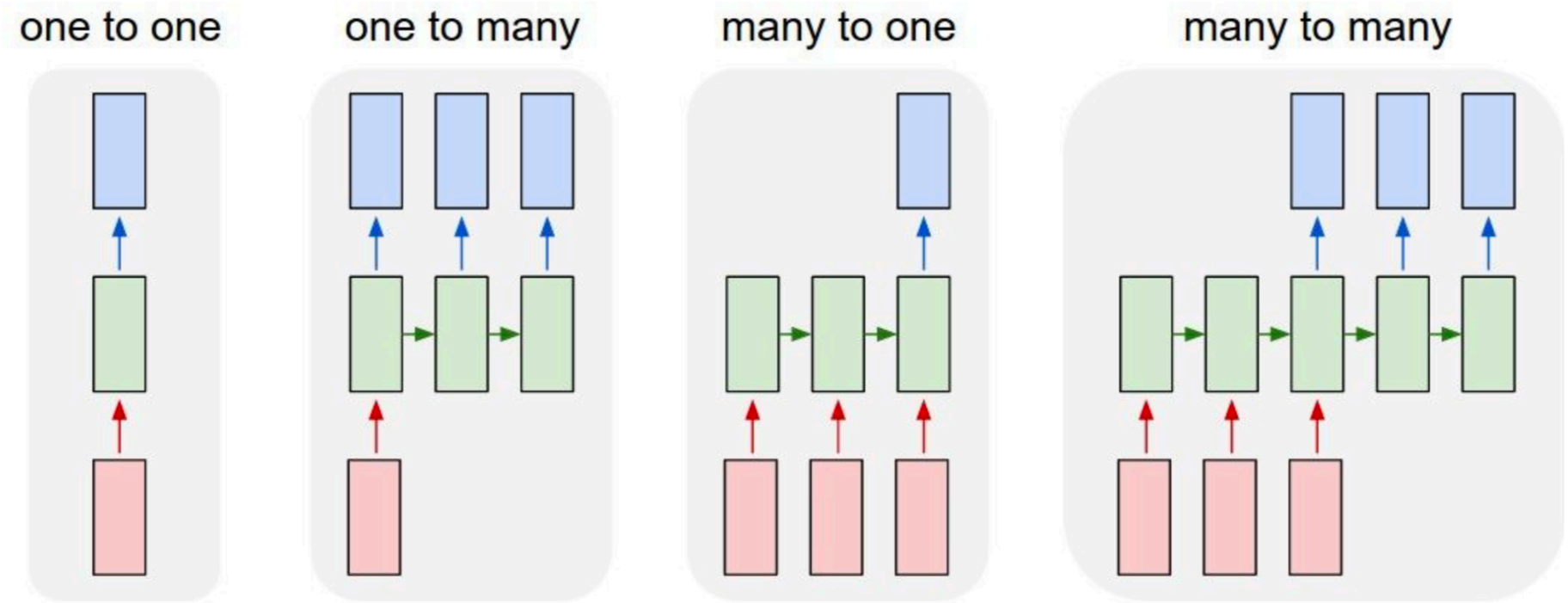
Image -> Caption

Text -> Sentiment

[Image from http://vision.stanford.edu/teaching/cs231n/slides/2016/winter1516_lecture10.pdf]

: A Layer

Task classification



Object Classification

Image -> Class

Image Captioning

Image -> Caption

Sentiment Analysis

Text -> Sentiment

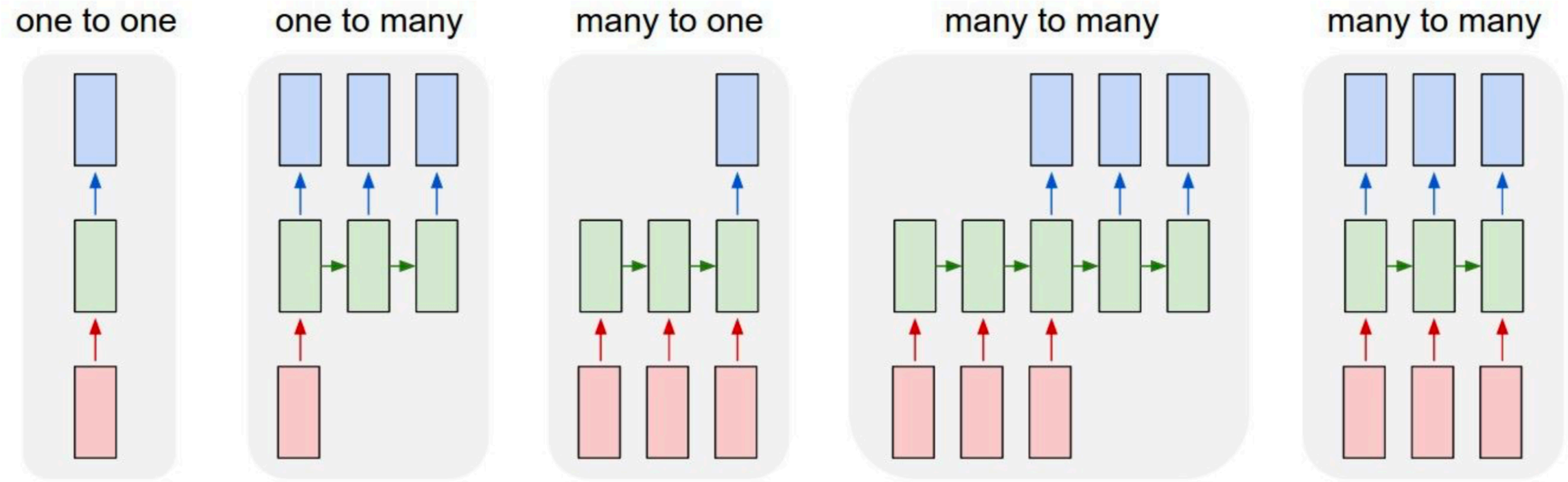
Machine Translation

French -> English

[Image from http://vision.stanford.edu/teaching/cs231n/slides/2016/winter1516_lecture10.pdf]

: A Layer

Task classification



Object Classification

Image -> Class

Image Captioning

Image -> Caption

Sentiment Analysis

Text -> Sentiment

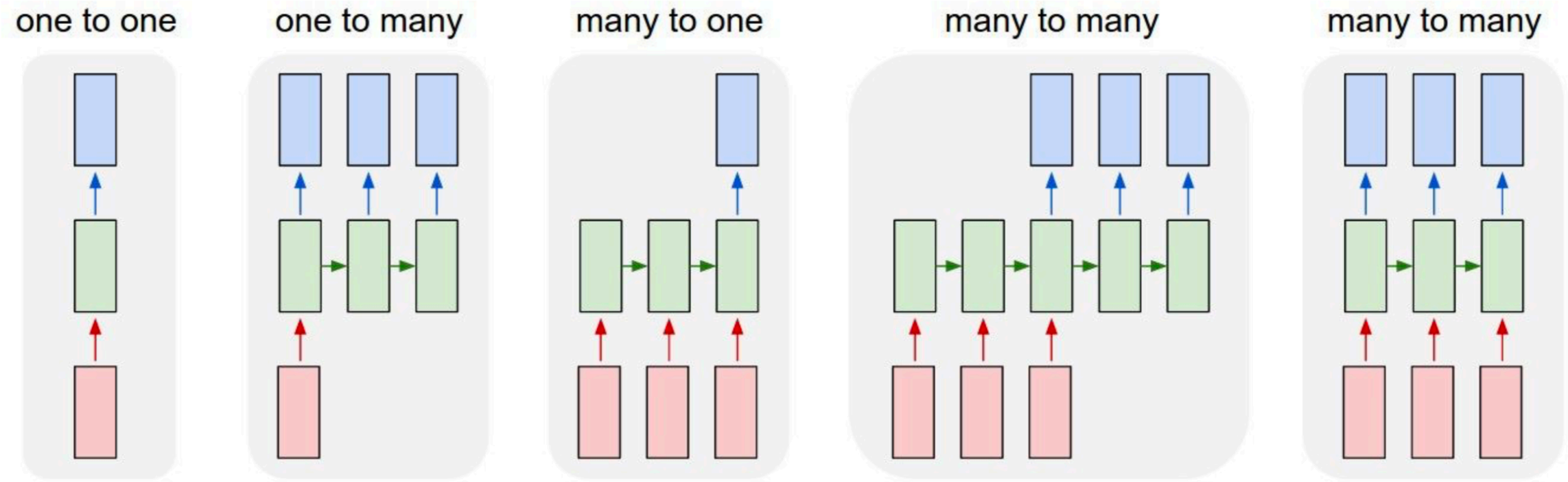
Machine Translation

French -> English

[Image from http://vision.stanford.edu/teaching/cs231n/slides/2016/winter1516_lecture10.pdf]

: A Layer

Task classification



Object Classification

Image -> Class

Image Captioning

Image -> Caption

Sentiment Analysis

Text -> Sentiment

Machine Translation

French -> English

Anomaly Detection

Sensor reading -> Class

[Image from http://vision.stanford.edu/teaching/cs231n/slides/2016/winter1516_lecture10.pdf]

A practical overview of Recurrent Neural Networks (RNNs)

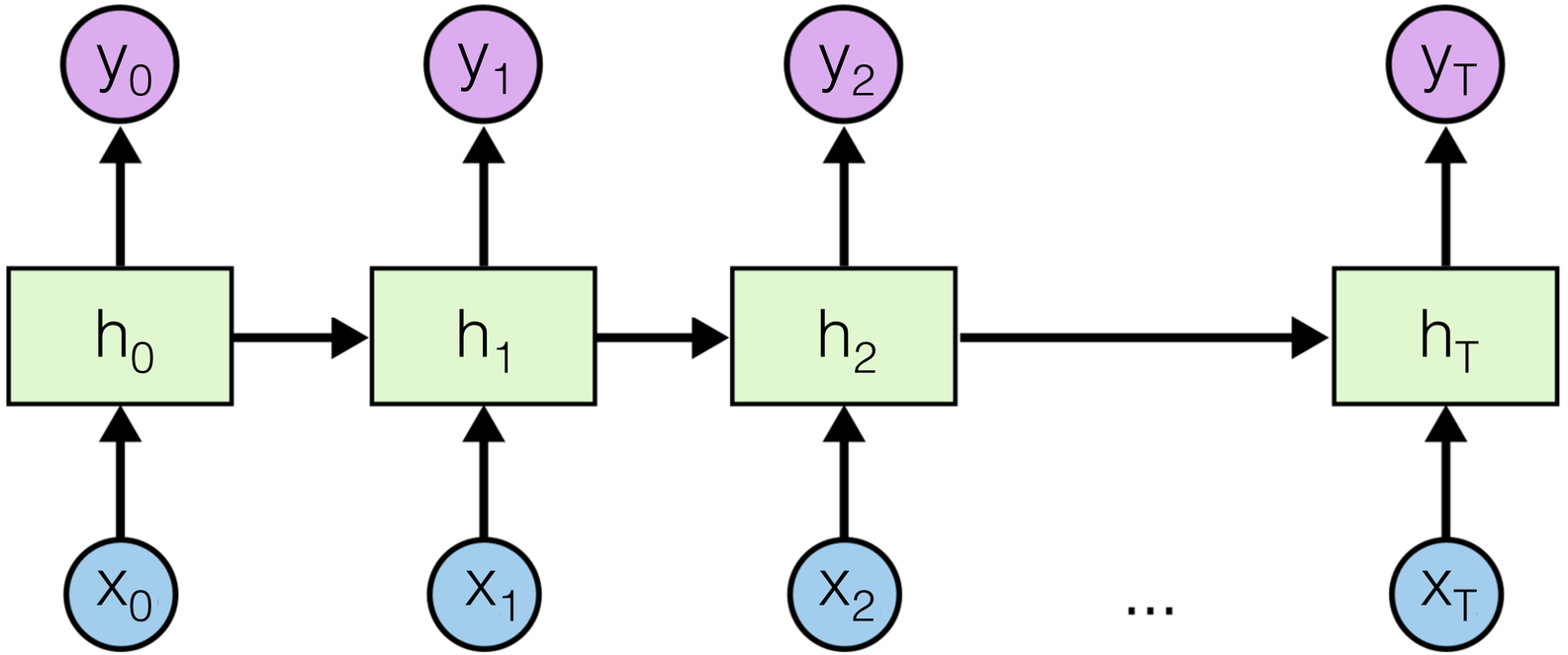
Intuition behind Recurrent Neural Networks

- Model for sequential data
- Imagine reading a book
  - You read the text word by word. Each word is added to your memory
  - After each page you memory contains some representation of all the words you have read so far

Basic idea

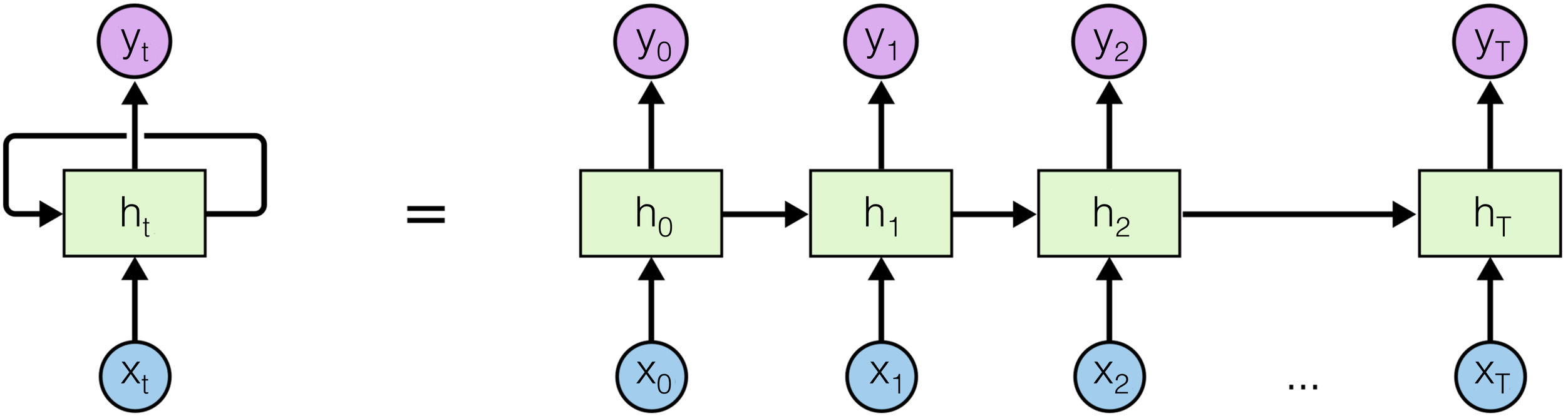
A many-to-many task

- **x**: inputs
- **y**: outputs
- **h**: hidden layers
- subscripts index time



[http://colah.github.io/]

Unroll through time

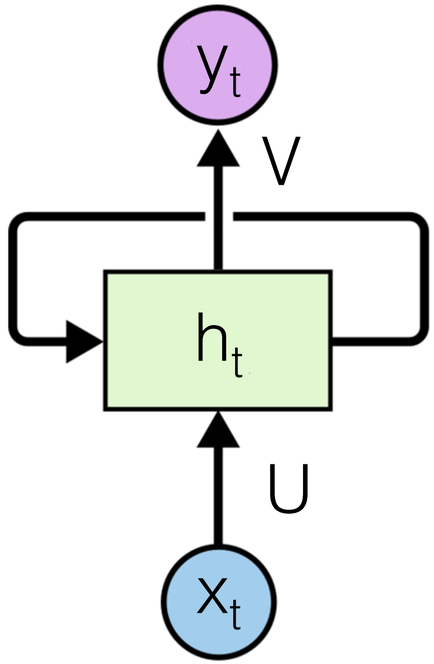


http://colah.github.io/

Basic idea (with parameters)

Process through time (t)

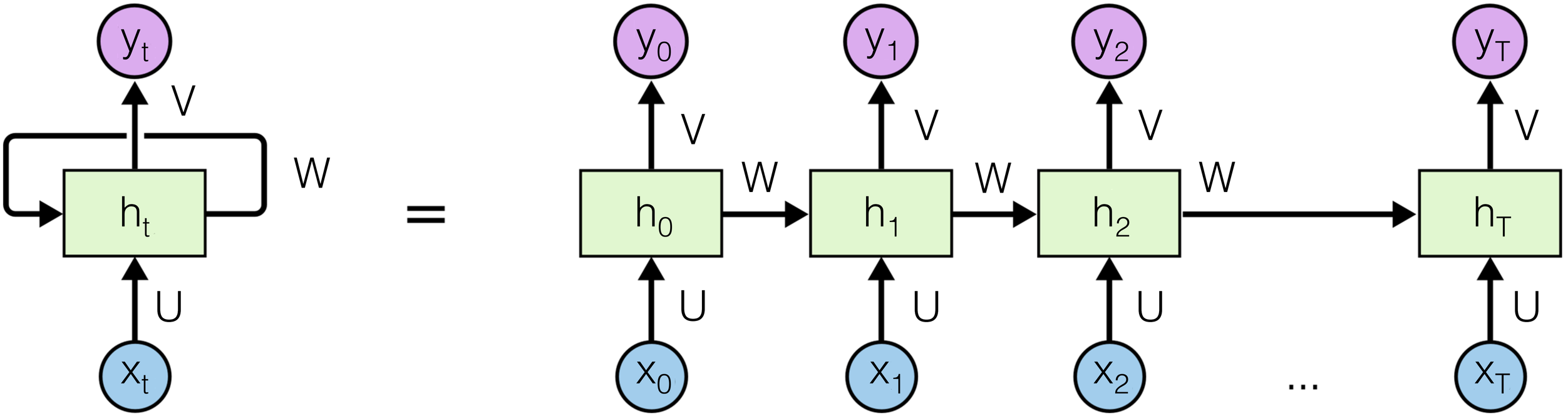
- **U, V, W: parameters**
  - **Shared through time**
- **Simplest parametrization**



$$\boldsymbol{h_t} = \tanh(\boldsymbol{U x_t} + \boldsymbol{W h_{t-1}})$$
$$\boldsymbol{y_t} = \boldsymbol{f}(\boldsymbol{V h_t})$$

[http://colah.github.io/]

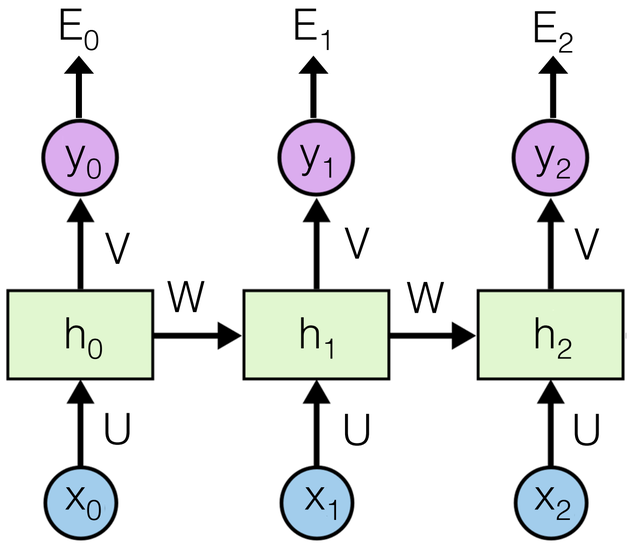
Unroll through time



http://colah.github.io/

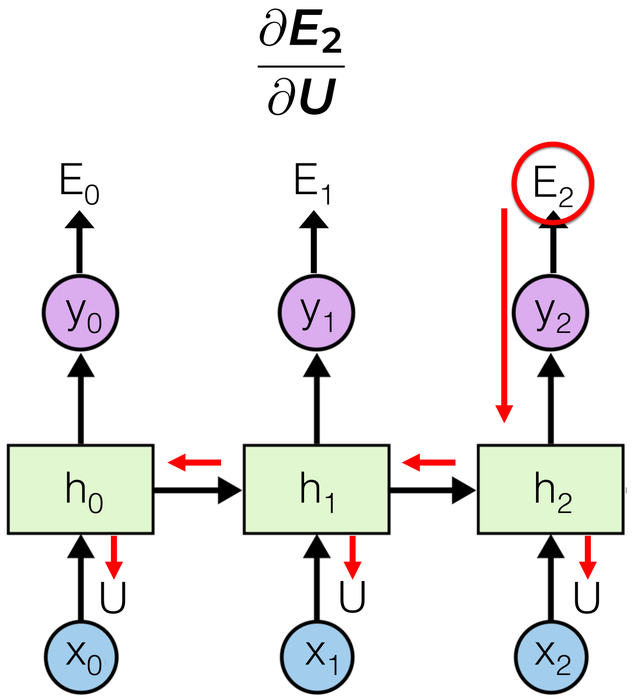
Training RNNs

- **Gradient descent from the loss** $E = \sum_t (y_t - \hat{y}_t)^2$
- **Following the structure the gradient is back propagated through time**

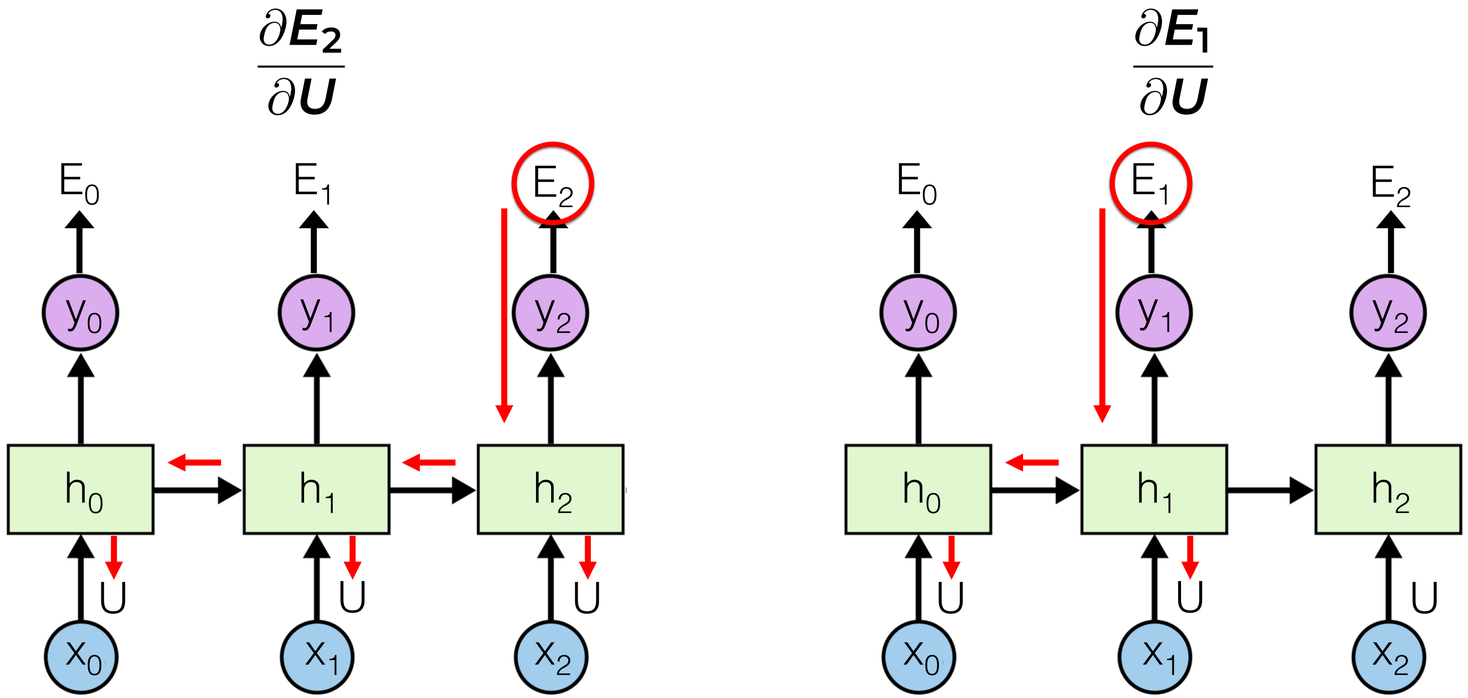


For example, the gradient for U $\frac{\partial E}{\partial U} = \sum_t \frac{\partial E_t}{\partial U}$

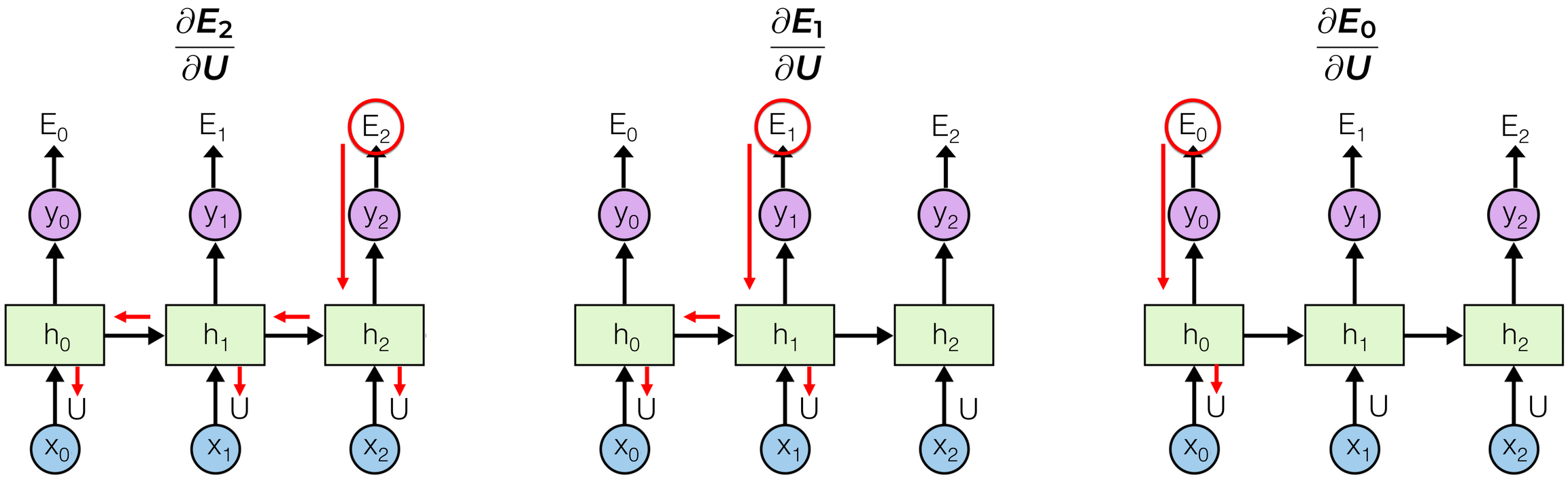
For example, the gradient for U $\frac{\partial E}{\partial U} = \sum_t \frac{\partial E_t}{\partial U}$



For example, the gradient for U $\frac{\partial E}{\partial U} = \sum_t \frac{\partial E_t}{\partial U}$

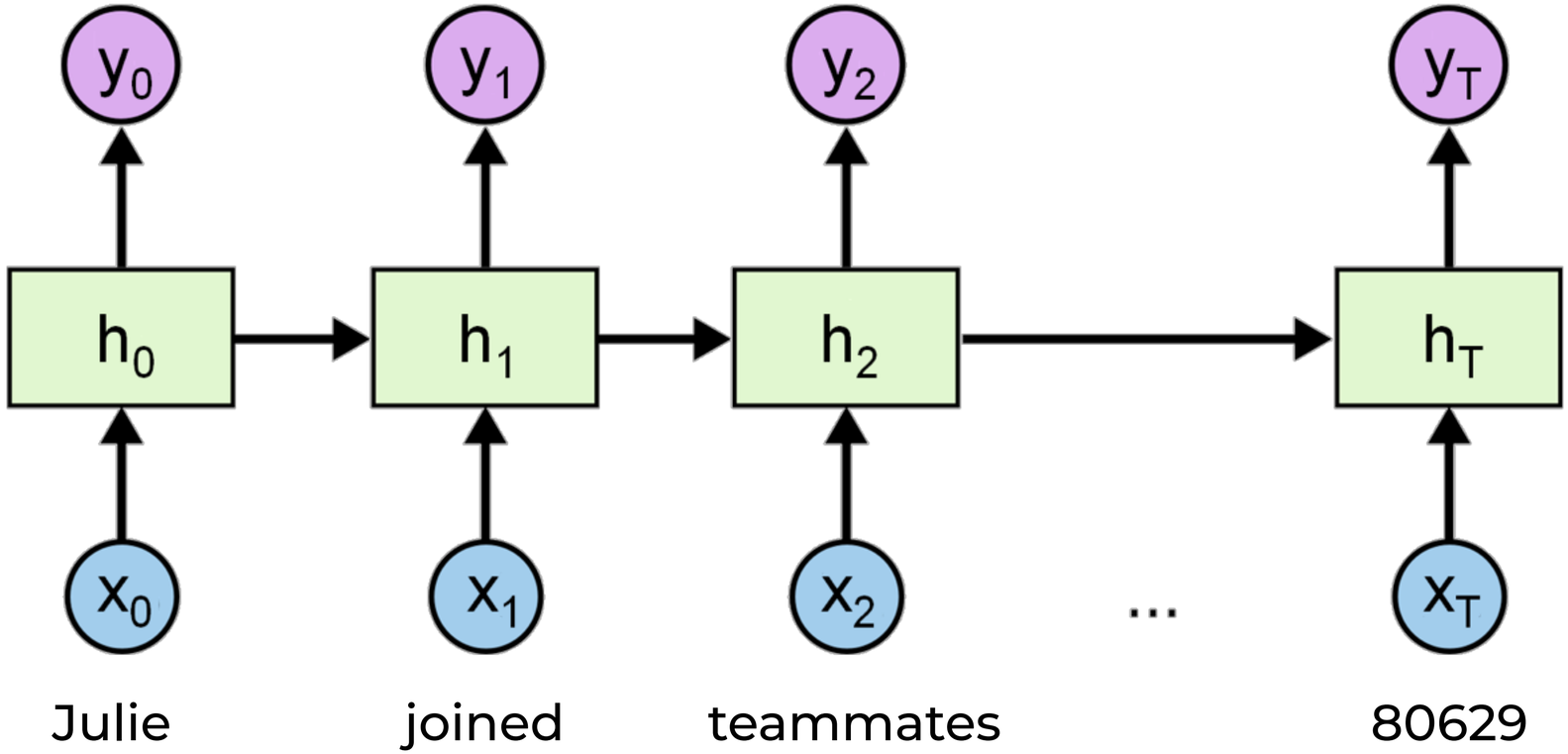


For example, the gradient for U $\frac{\partial E}{\partial U} = \sum_t \frac{\partial E_t}{\partial U}$



Limitations

- **Long-term dependencies are difficult to learn**

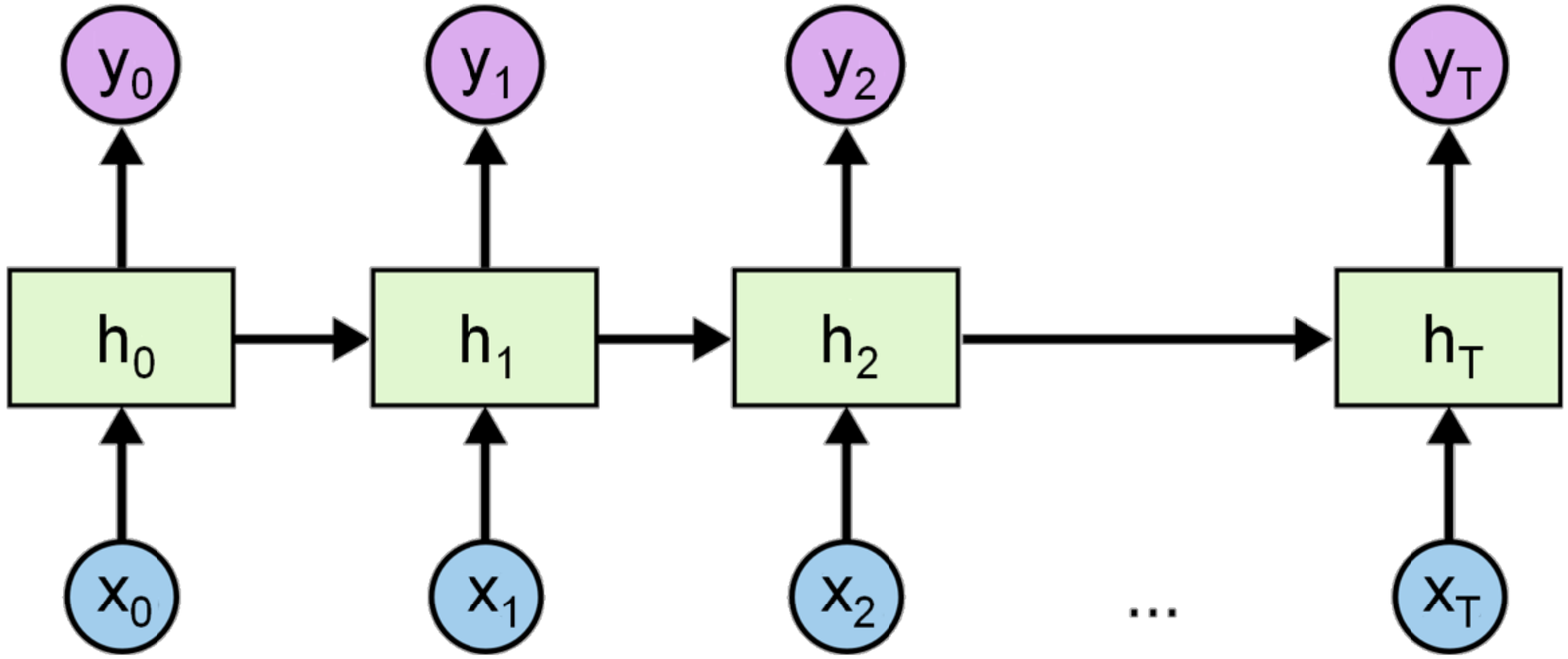


http://colah.github.io/

Limitations

- **Long-term dependencies are difficult to learn**

Q: Who joined the class?

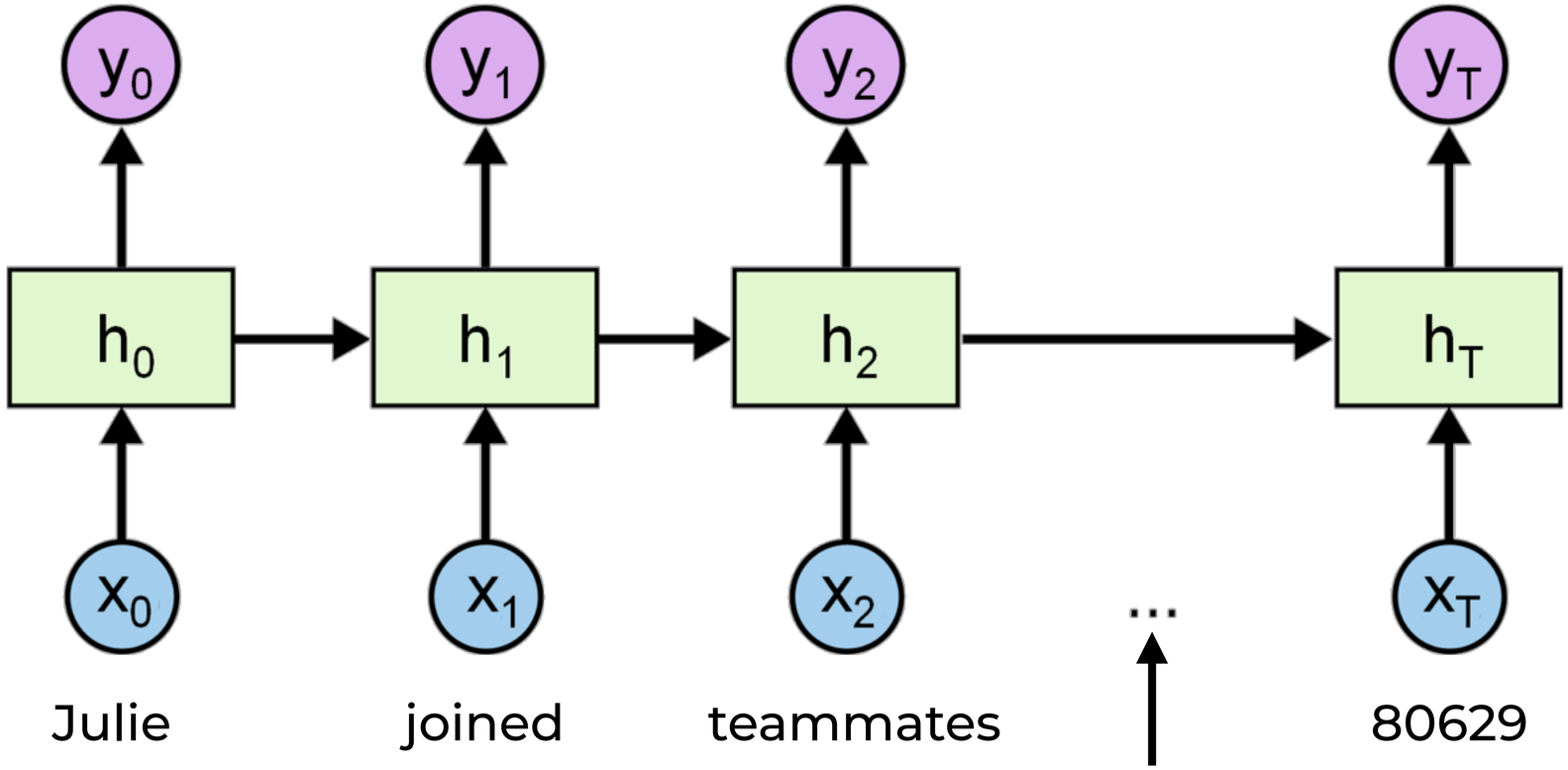


http://colah.github.io/

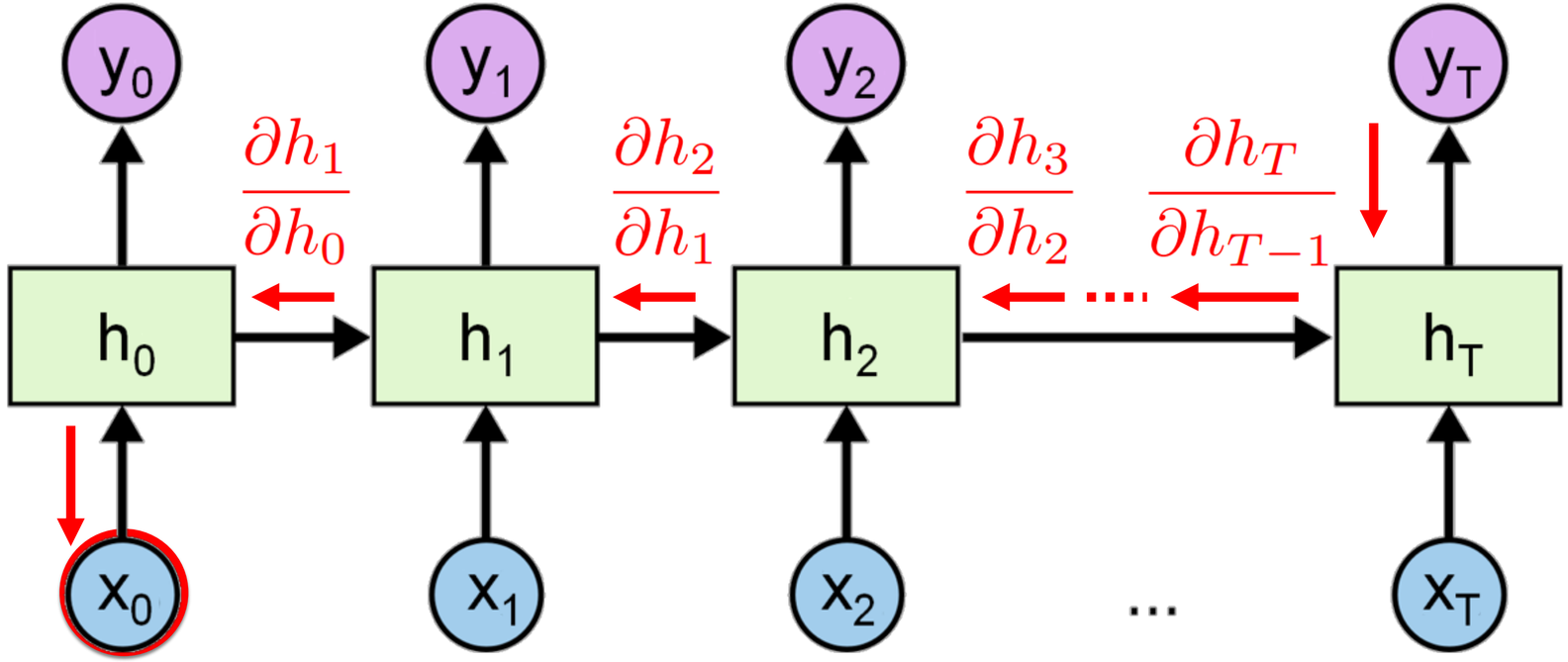
Limitations

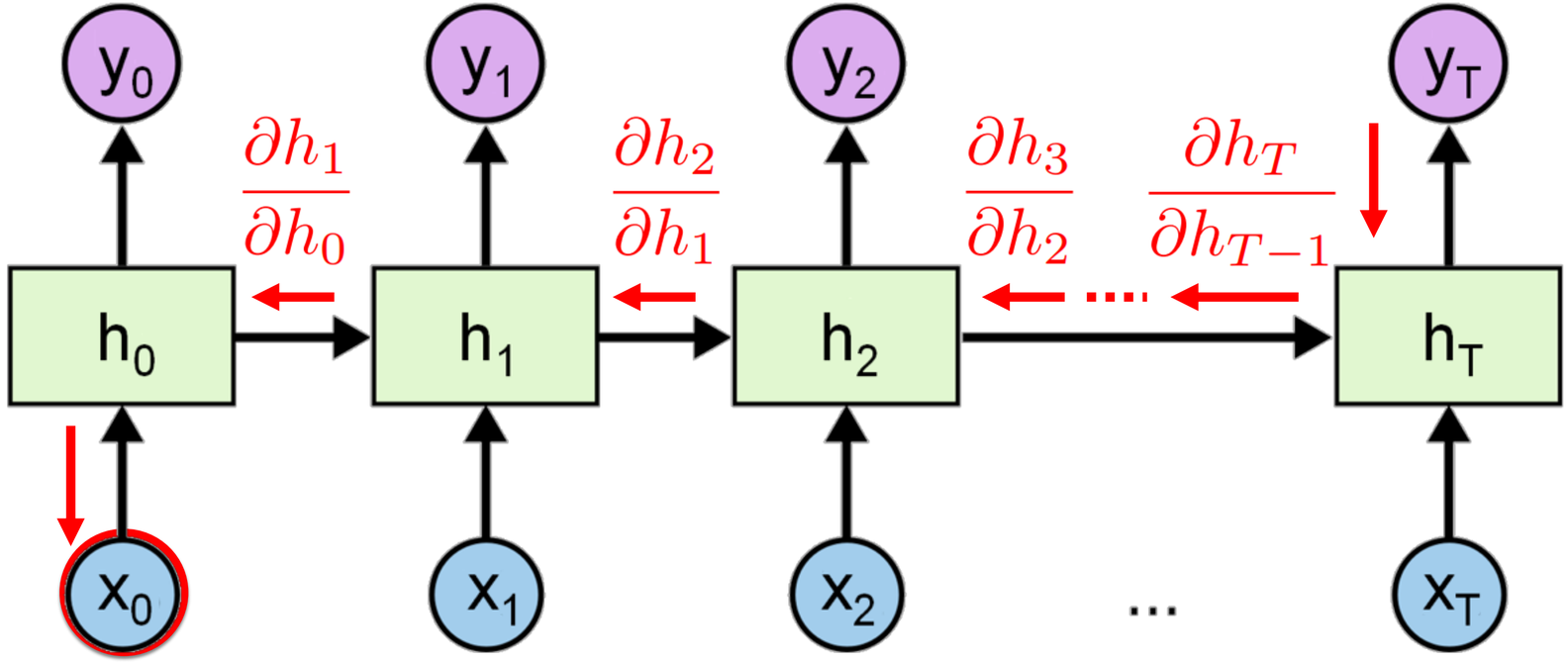
- **Long-term dependencies are difficult to learn**

Q: Who joined the class?

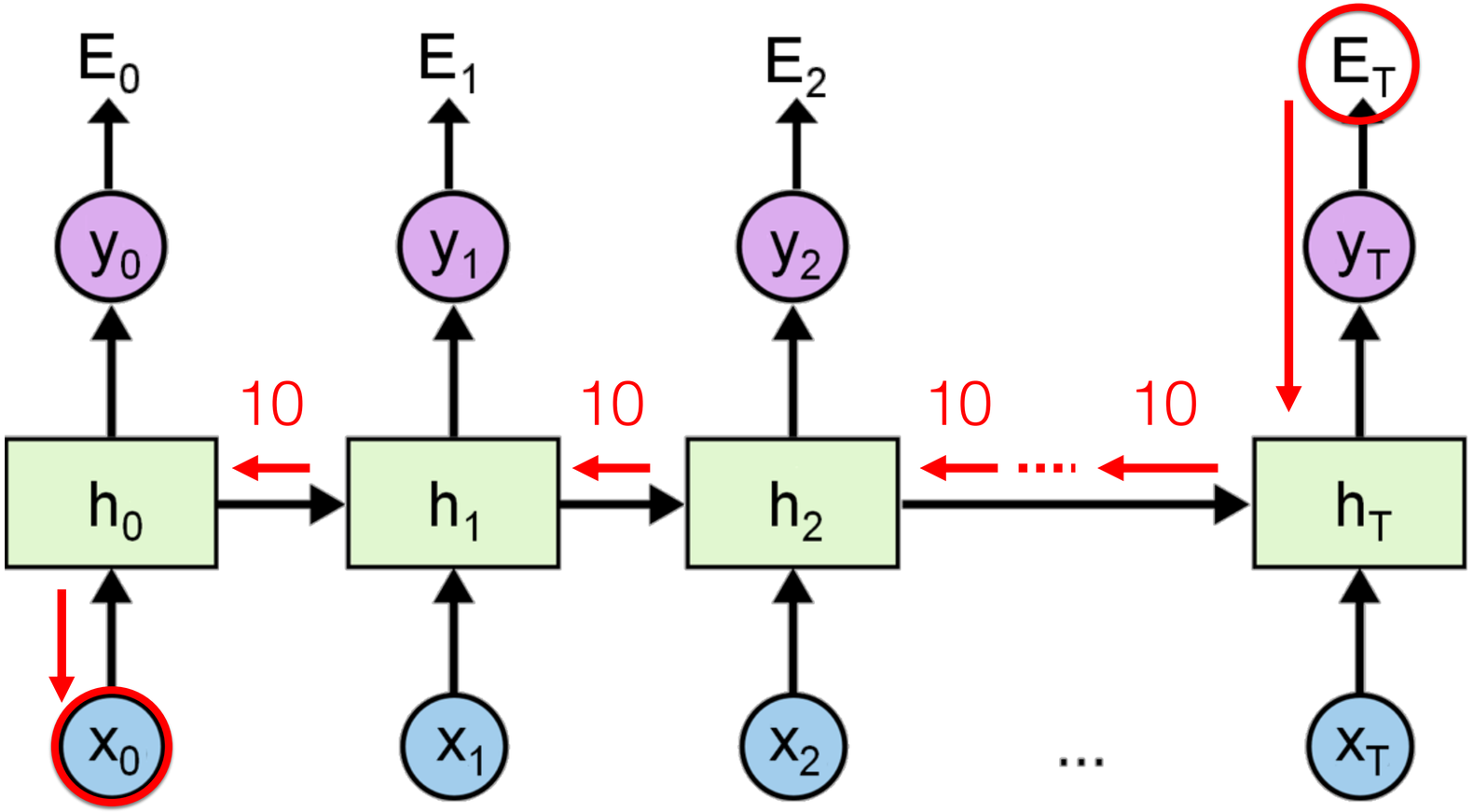


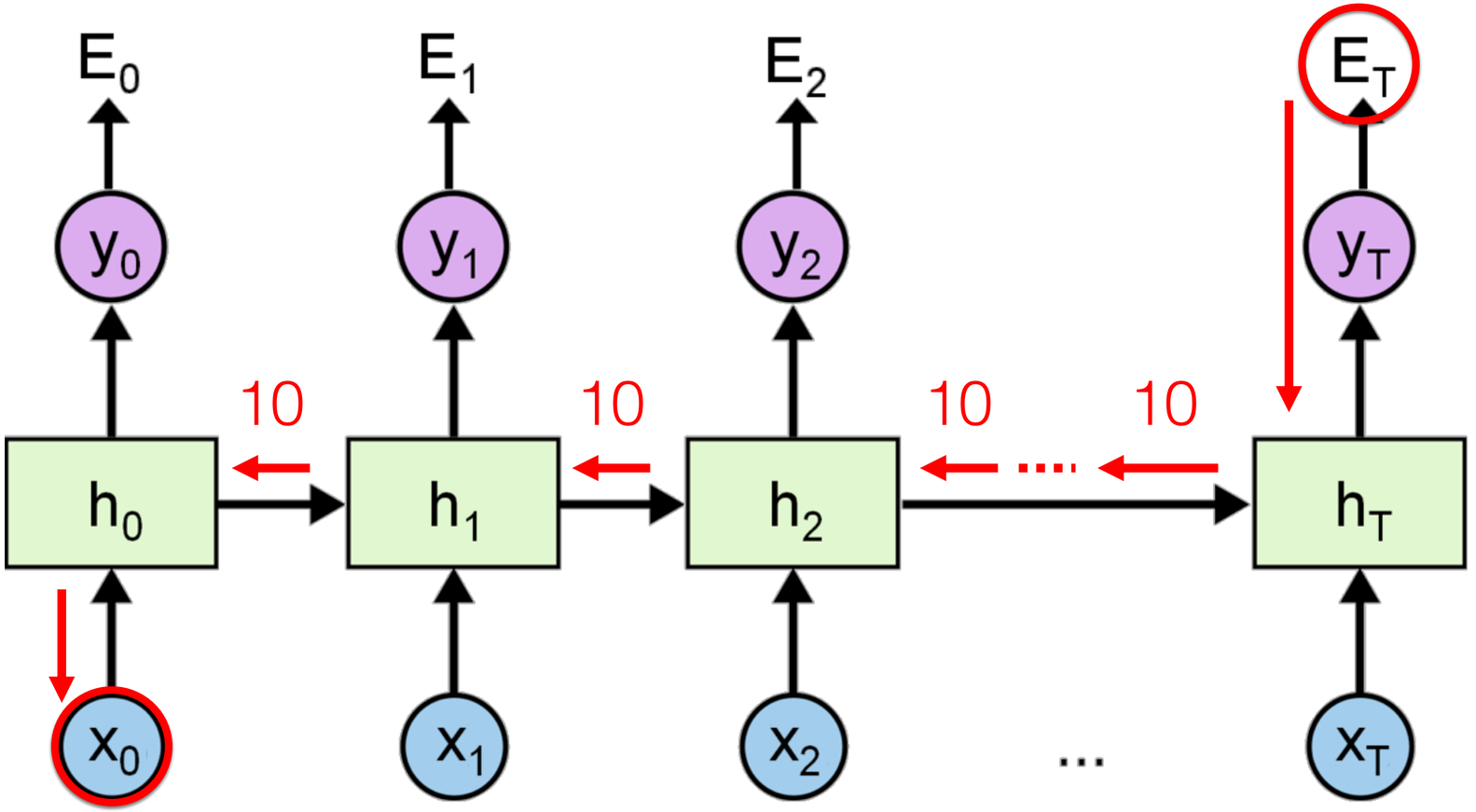
http://colah.github.io/


y_0
y_1
y_2
y_T
$\frac{\partial h_1}{\partial h_0}$
$\frac{\partial h_2}{\partial h_1}$
$\frac{\partial h_3}{\partial h_2}$
$\frac{\partial h_T}{\partial h_{T-1}}$
h_0
h_1
h_2
h_T
x_0
x_1
x_2
...
x_T



- Can be difficult because it is unstable
  - Each partial derivative depends on the parameters W
    - The largest eigenvalue of W:
      - >1: the gradient will explode
      - <1: the gradient will vanish

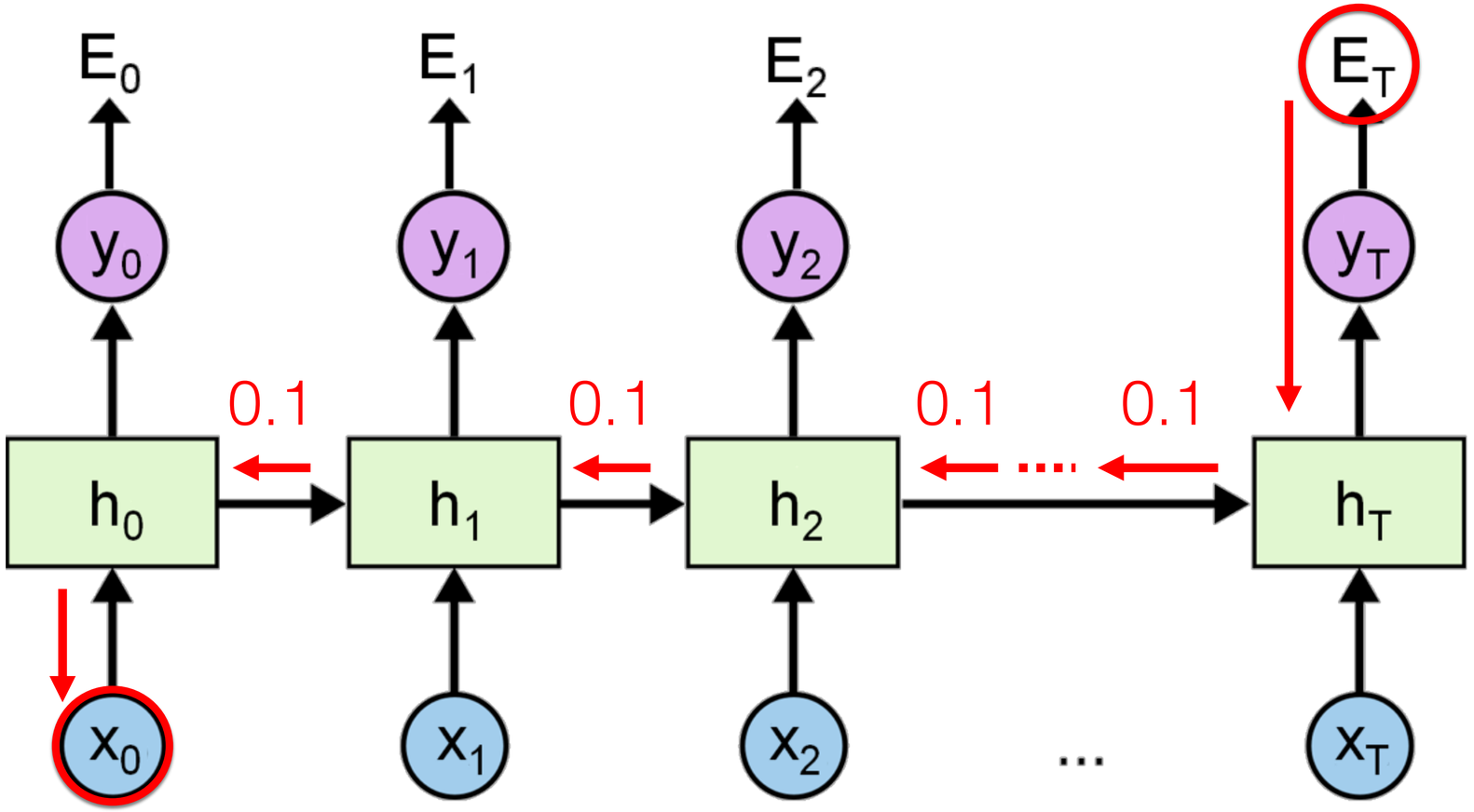

E_0
E_1
E_2
E_T
y_0
y_1
y_2
y_T
10
10
10
10
h_0
h_1
h_2
h_T
x_0
x_1
x_2
...
x_T

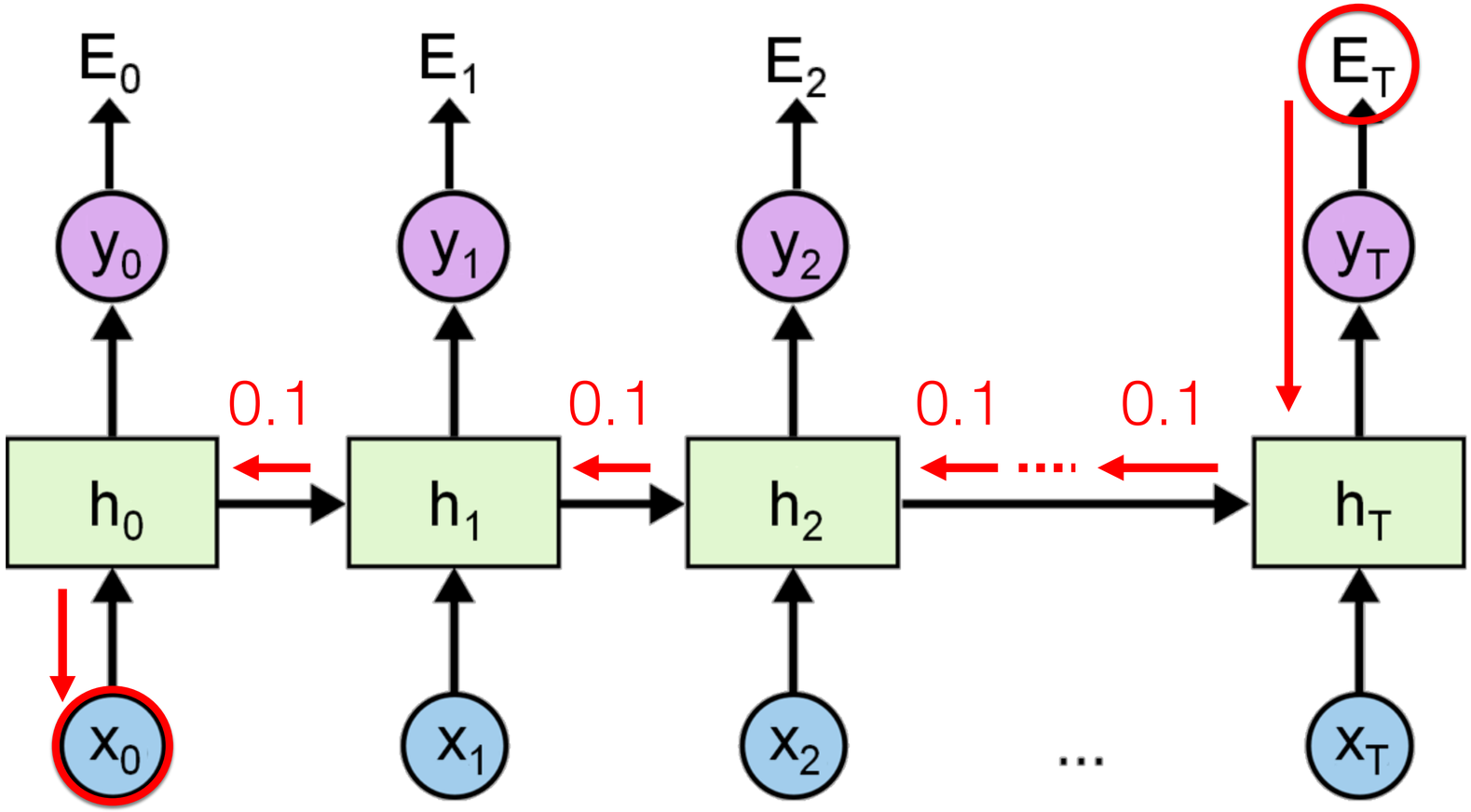


- **Largest eigenvalue of W is > 1**

Gradient Clipping

- Heuristic to prevent gradient explosion
- If gradient norm is too large then reduce


E_0
E_1
E_2
E_T
y_0
y_1
y_2
y_T
0.1
0.1
0.1
0.1
h_0
h_1
h_2
h_T
x_0
x_1
x_2
...
x_T



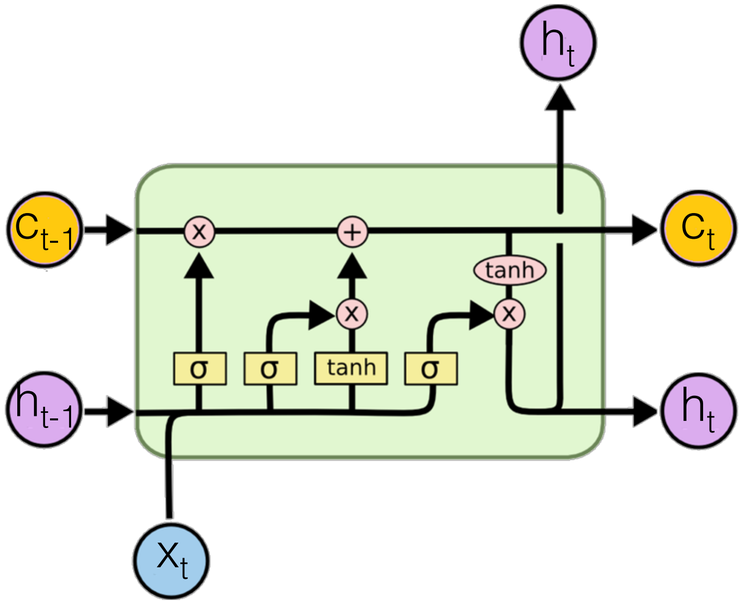
- **Largest eigenvalue of W is < 1**

Gradient Vanishing

- **No simple solution**
- **Change the “memory cells” of the RNN**

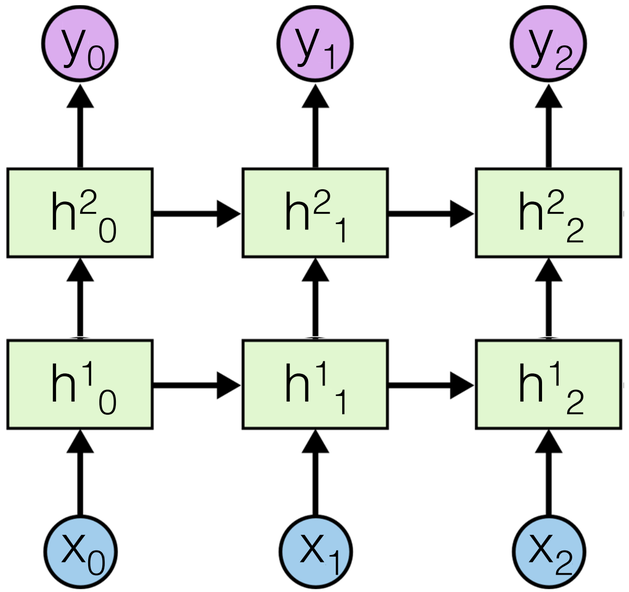
$$\boldsymbol{h_t} = \tanh(\boldsymbol{U x_t} + \boldsymbol{W h_{t-1}})$$
$$\boldsymbol{y_t} = \boldsymbol{f}(\boldsymbol{V h_t})$$

LSTM

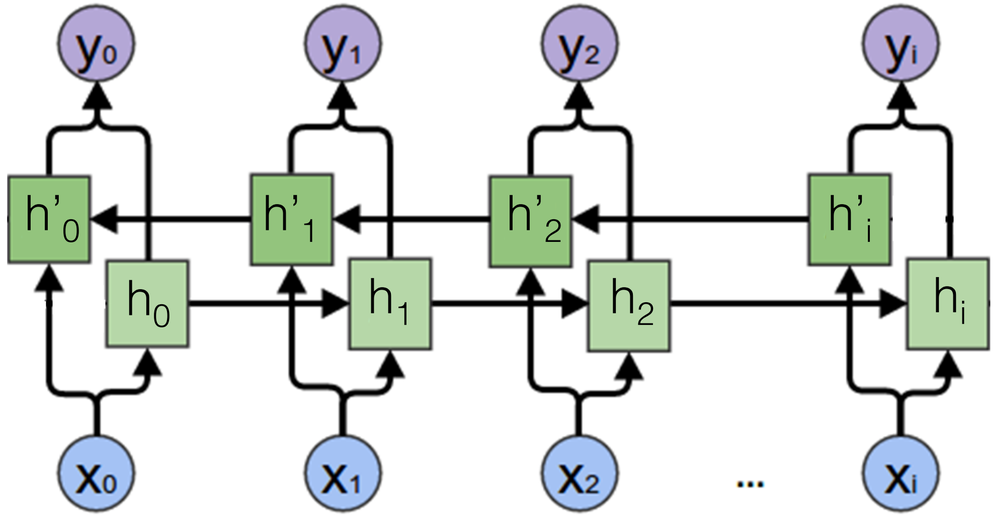


Different architectures of RNNs

Deep RNNs



Bi-directional RNNs



Language Modelling using RNNs

Language modelling. An example

The guardian of the land of an heir who is under age shall take from it only reasonable revenues, customary dues, and feudal services. He shall do this without destruction or damage to men or property. If we have given the guardianship of the land to a sheriff, or to any person answerable to us for the revenues, and he commits destruction or damage, we will exact compensation from him, and the land shall be entrusted to two worthy and prudent men of the same 'fee', who shall be answerable to us for the revenues, or to the person to whom we have assigned them. If we have given or sold to anyone the guardianship of such land, and he causes destruction or damage, he shall lose the guardianship of it, and it shall be handed over to two worthy and prudent men of the same 'fee', who shall be similarly answerable to us.

$$P(w_1, w_2, w_3, \ldots, w_T)$$

Prevalent methodology

- Frame language modelling as a prediction task
  - Predict the next word given the previous word(s)

$$P(w_1, w_2, w_3, \ldots, w_T) = \prod_{t=0}^{T} P(w_t \mid w_{t-1}, w_{t-2}, \ldots w_0)$$

Prevalent methodology

- Frame language modelling as a prediction task
  - Predict the next word given the previous word(s)

$$P(w_1, w_2, w_3, \ldots, w_T) = \prod_{t=0}^{T} \underbrace{P(w_t \mid w_{t-1}, w_{t-2}, \ldots w_0)}_{P(y_t | x_t, x_{t-1}, \ldots x_0)}$$

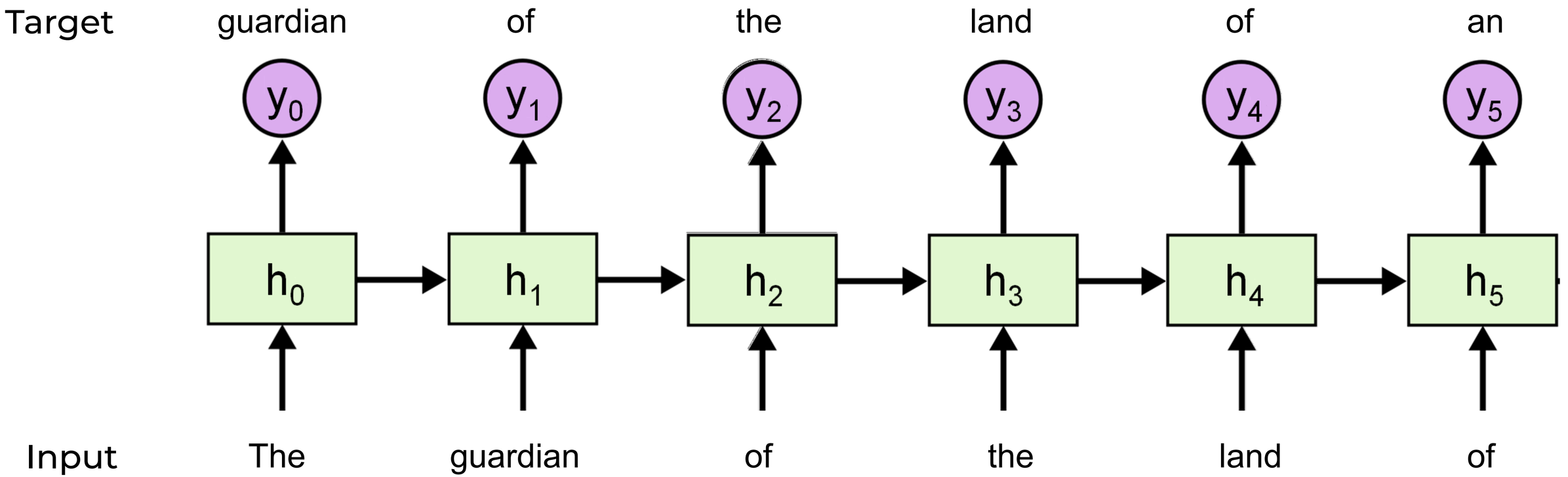
Prevalent methodology

- Frame language modelling as a prediction task
  - Predict the next word given the previous word(s)

$$P(w_1, w_2, w_3, \ldots, w_T) = \prod_{t=0}^{T} \underbrace{P(w_t \mid w_{t-1}, w_{t-2}, \ldots w_0)}_{P(y_t | x_t, x_{t-1}, \ldots x_0)}$$

- Intuition
  - Success at this task implies that you have an understanding of the text (at least short-term)

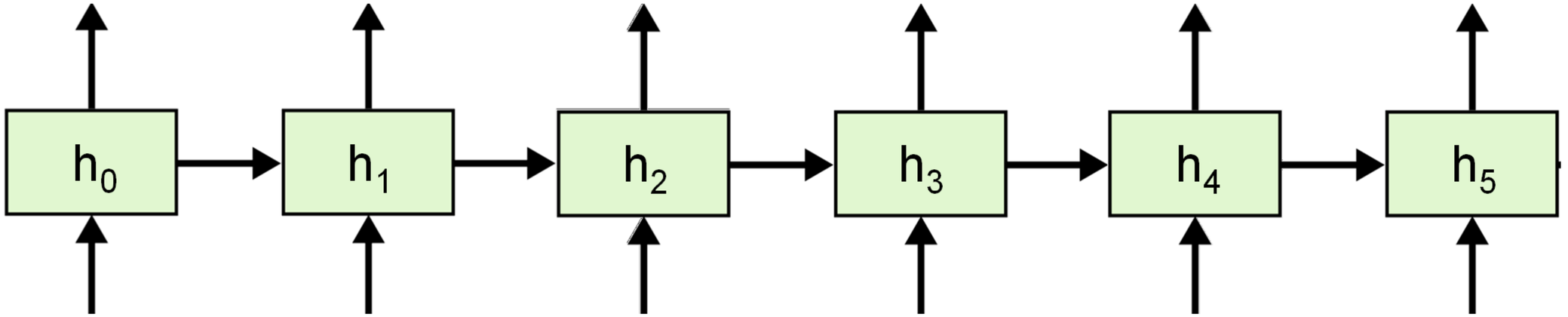
Training.
Given previous words,
predict the next word



Test.

1. Predict One word at a time.
2. Feed the prediction back to the model

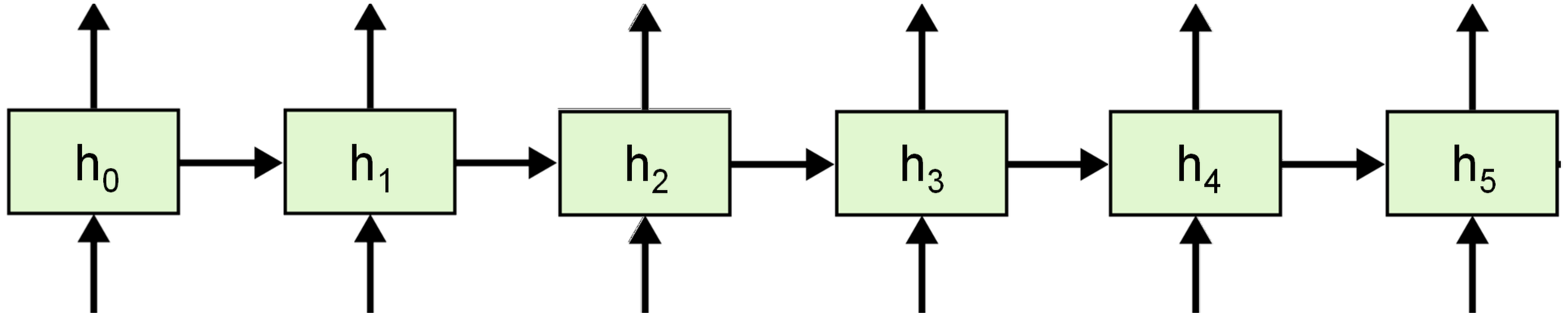
Prediction



Input

Test.
1. Predict One word at a time.
2. Feed the prediction back to the model

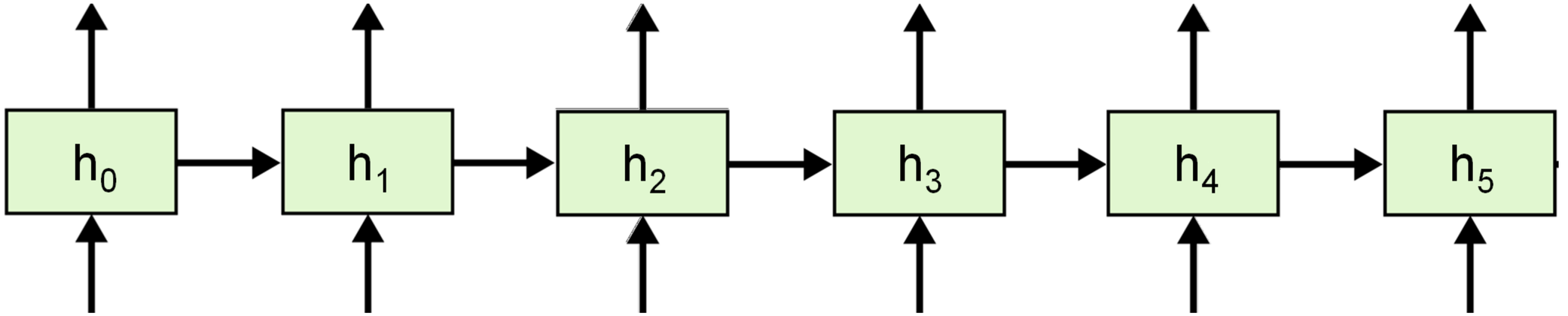
Prediction



Input The

Test.
1. Predict One word at a time.
2. Feed the prediction back to the model

Prediction woman

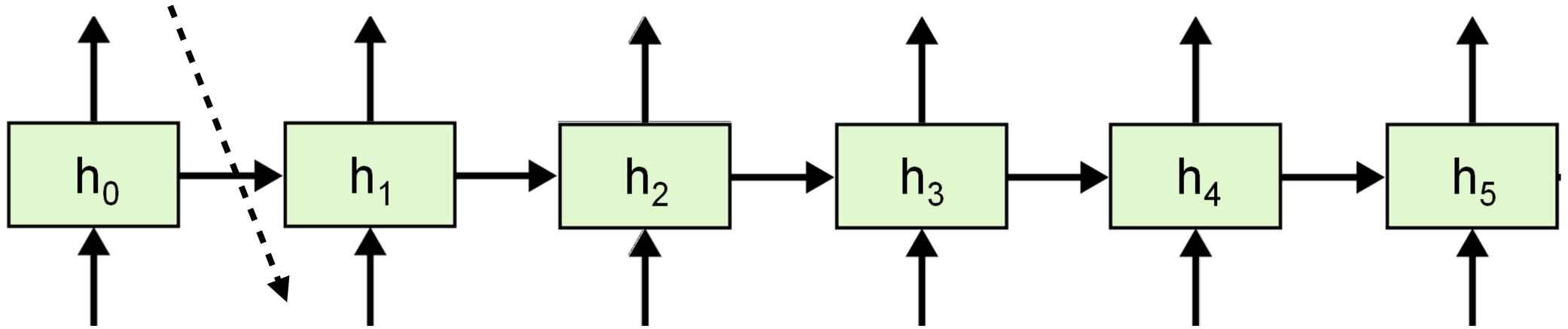


Input The

Test.

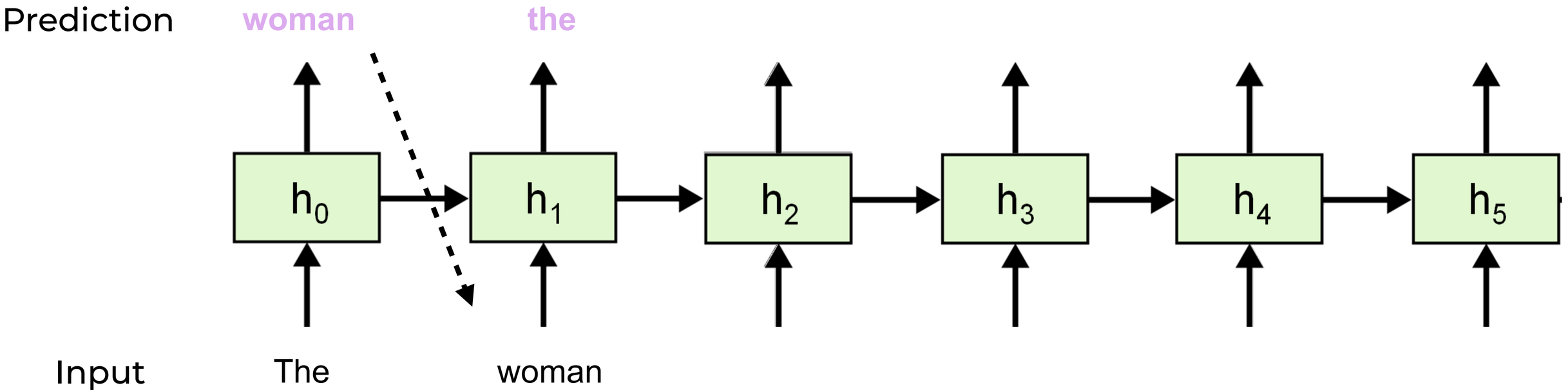
1. Predict One word at a time.
2. Feed the prediction back to the model

Prediction woman

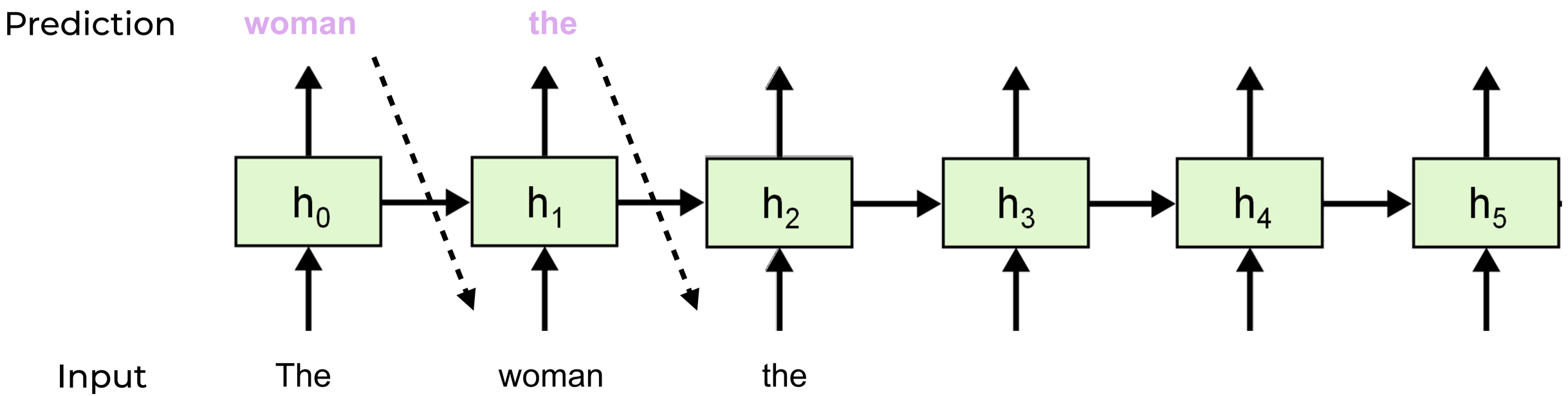


Input The woman

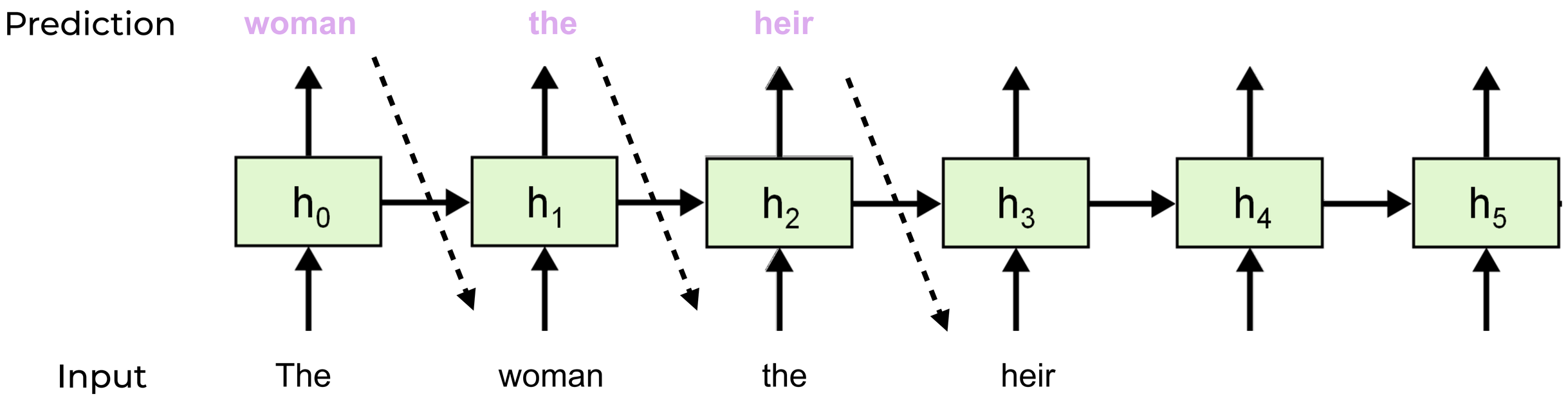
Test.
1. Predict One word at a time.
2. Feed the prediction back to the model



Test.
1. Predict One word at a time.
2. Feed the prediction back to the model

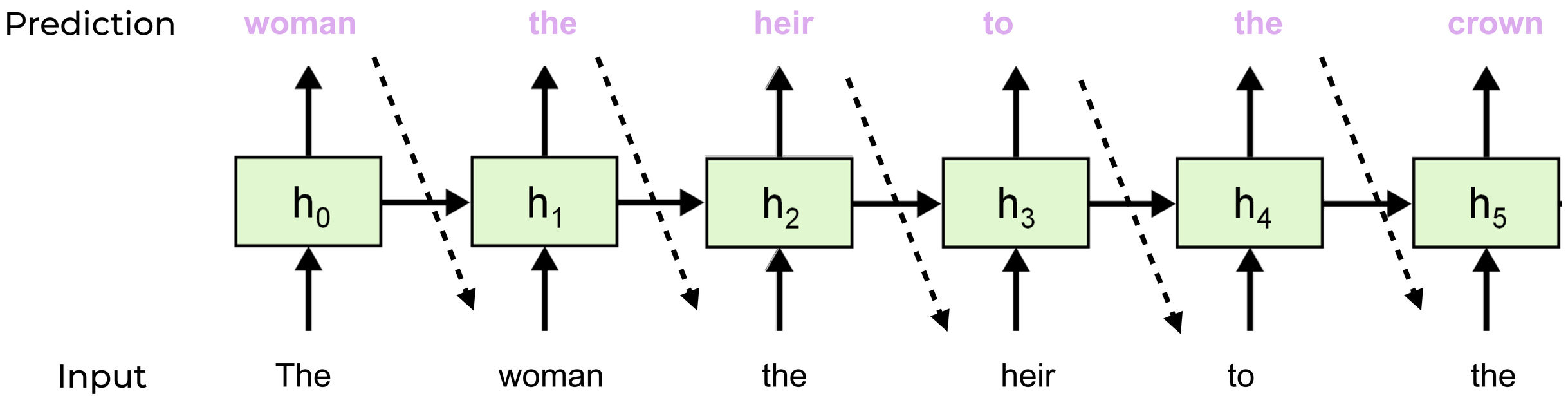


Test.
1. Predict One word at a time.
2. Feed the prediction back to the model

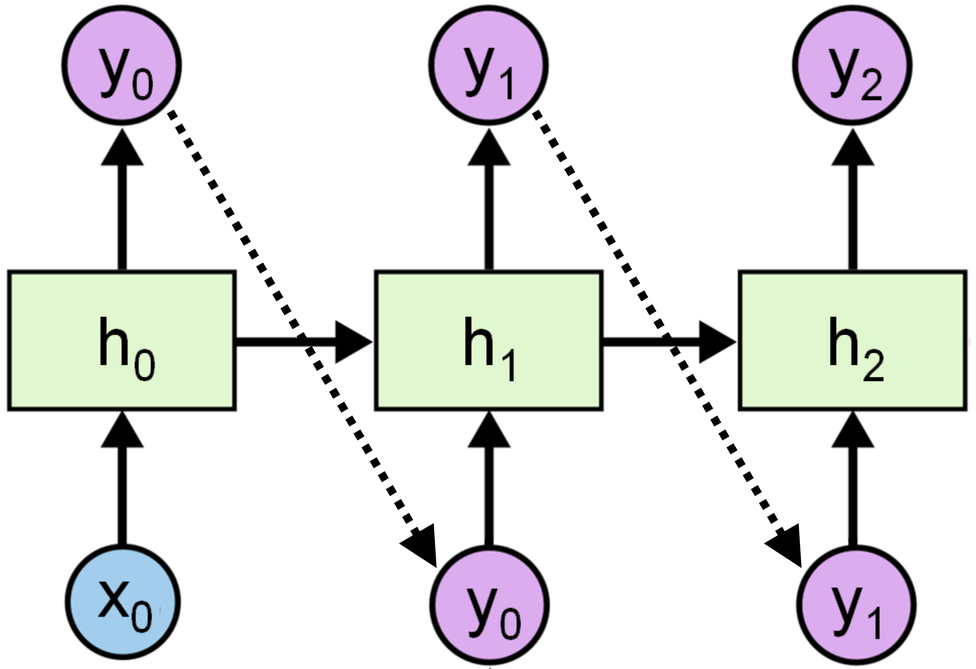


Test.

1. Predict One word at a time.
2. Feed the prediction back to the model

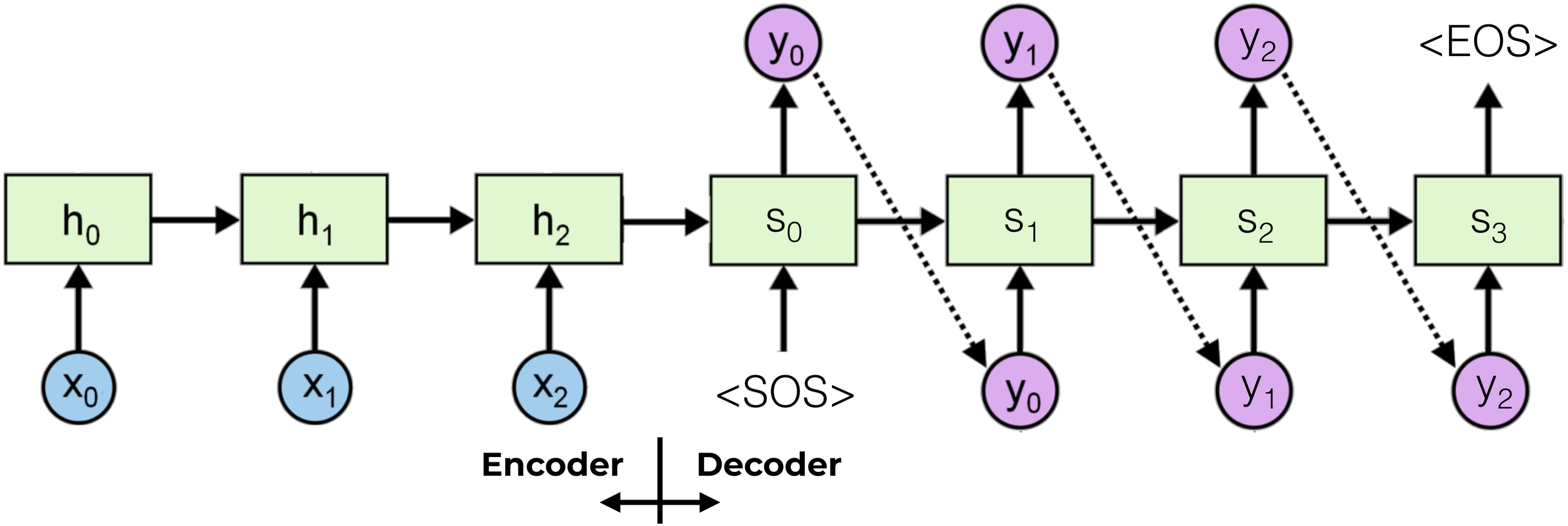


Generating Sequences from RNNs

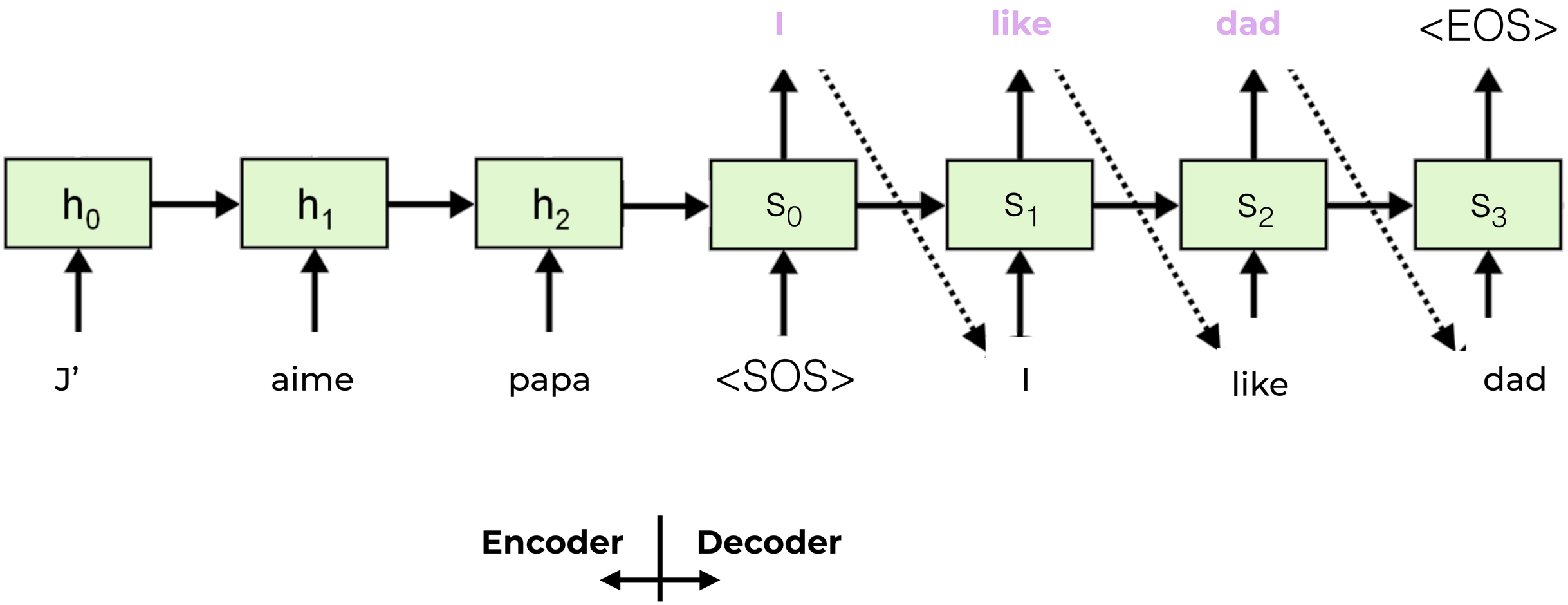


Encoder-Decoder architecture

- One of the most influencial recent ideas



Encoder-Decoder for translation



RNNs takeaways (I)

- Can be used to learn from varying-length input
  - Typically for discrete data (e.g., words)
  - Can be used both for predictions and for representations

RNNs takeaways (II)

- Can easily be used as modules inside more complex systems
  - Current applications: Natural language understanding (Q&A, Dialogs), machine translation
- Active research field
  - Still difficult to learn very long sequences (reading a book)
  - Transformers
- Not available in scikit-learn

Overview of Convolutional Neural Networks (CNNs)

Neural Network architectures

- Feed-forward neural nets are standard
- We can specialize neural networks for particular tasks
  - Different data have different characteristics
- Grid data: for example an image
  - Convolutional nets replace matrix multiplications by convolutions and pooling operations

This is how I see

88	126	145	85	123	142	85	123	142	86	124
86	125	142	84	123	140	83	122	139	85	124
85	124	141	82	121	138	82	121	138	84	123
82	119	135	80	117	133	80	117	133	85	122
78	114	128	77	113	127	79	115	129	84	120
79	115	129	78	114	128	80	116	130	83	119
82	118	130	81	117	129	81	117	129	82	118
83	117	129	82	116	128	82	116	128	82	116
79	113	123	79	113	123	80	114	124	81	115
76	108	119	76	108	119	77	109	120	80	112
76	109	118	76	109	118	77	110	119	79	112

This is how my computer sees

Source: https://medium.com/deep-math-machine-learning-ai/chapter-8-0-convolutional-neural-networks-for-deep-learning-364971e34ab2

This is how I see

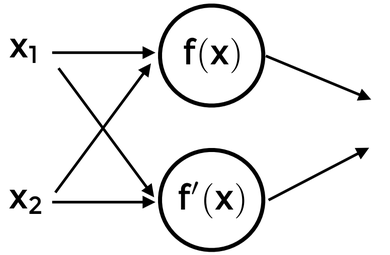
88	126	145	85	123	142	85	123	142	86	124
86	125	142	84	123	140	83	122	139	85	124
85	124	141	82	121	138	82	121	138	84	123
82	119	135	80	117	133	80	117	133	85	122
78	114	128	77	113	127	79	115	129	84	120
79	115	129	78	114	128	80	116	130	83	119
82	118	130	81	117	129	81	117	129	82	118
83	117	129	82	116	128	82	116	128	82	116
79	113	123	79	113	123	80	114	124	81	115
76	108	119	76	108	119	77	109	120	80	112
76	109	118	76	109	118	77	110	119	79	112

This is how my computer sees

- A 100 x 100 pixel image has 10,000 dimensions
  - Often have a color dimension (data is a tensor)
- Modern iPhone (12 MP): 4032 x 3024 pixels

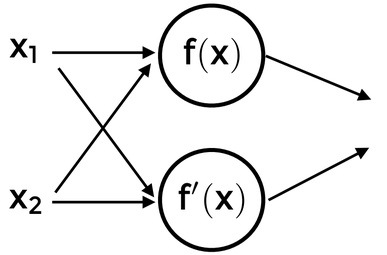
Source: https://medium.com/deep-math-machine-learning-ai/chapter-8-0-convolutional-neural-networks-for-deep-learning-364971e34ab2

Convolutional Neural Networks



Convolutional Neural Networks

- Feed-forward networks may not scale to high-dimensional data
  - Each additional input/feature adds “m” parameters
  - In practice it can be hard to scale to thousands of dimensions

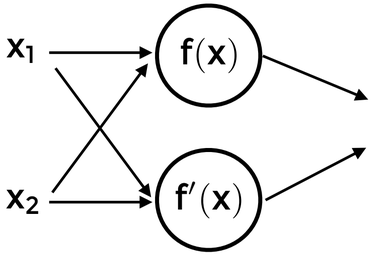


Convolutional Neural Networks

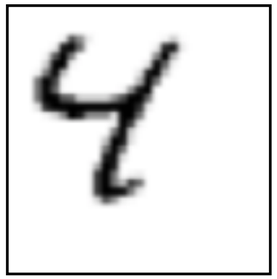
- Feed-forward networks may not scale to high-dimensional data
  - Each additional input/feature adds “m” parameters
  - In practice it can be hard to scale to thousands of dimensions
- Remedies: Convolutions and Pooling

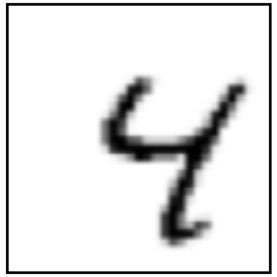
A. Sparse Connections

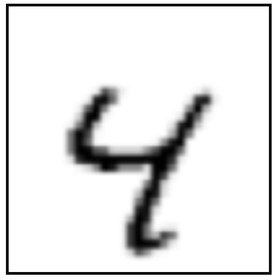
B. Parameter sharing



Some intuitions

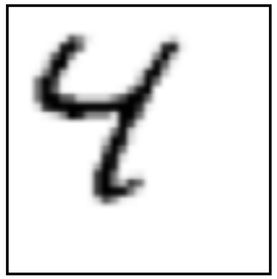


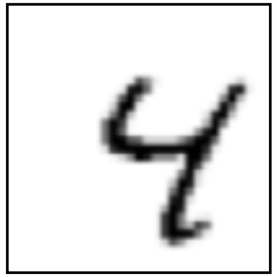


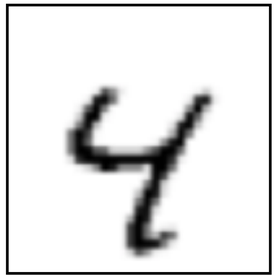


- You would like to recognize objects:
  - Regardless of their position (translation invariance)

Some intuitions

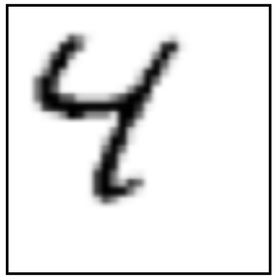




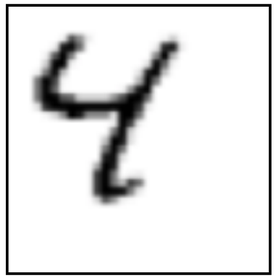


- You would like to recognize objects:
  - Regardless of their position (translation invariance)
- Have small detectors for object parts
  - Run the detectors over all regions of the image
  - Patterns of detection may indicate the presence of a full object

Image



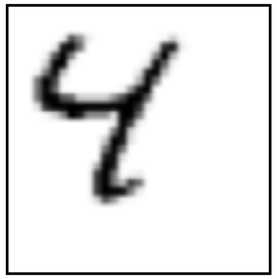
Image



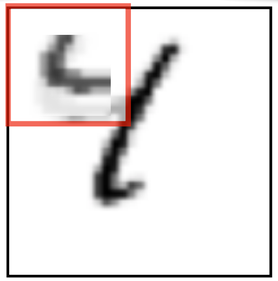
Filter (Kernel)

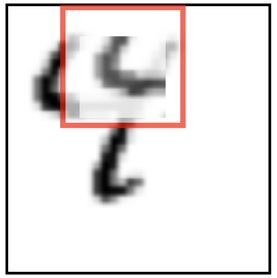
Pass the filter over the image (Convolutions)

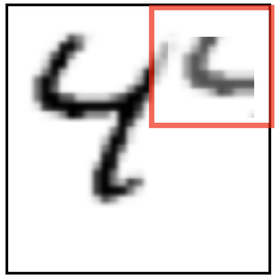
Image

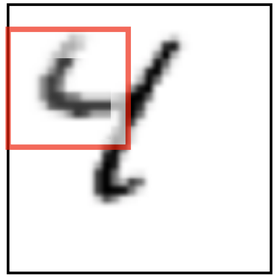


Filter (Kernel)





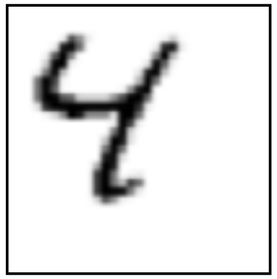




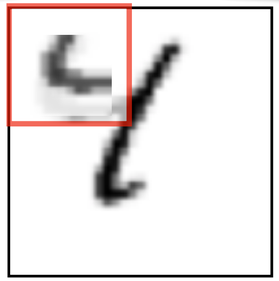
...

Pass the filter over the image
(Convolutions)

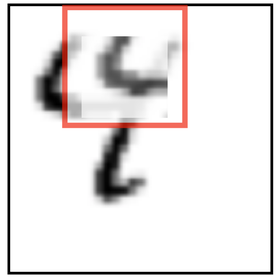
Image



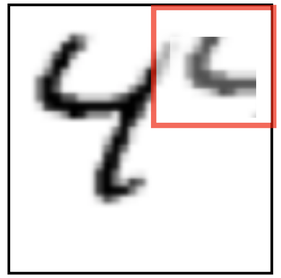
Filter (Kernel)



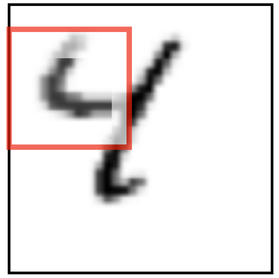
No detection
Output: Low



No detection
Output: Low



No detection
Output: Low



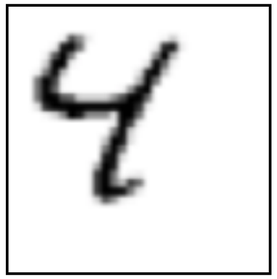
Detection
Output: High

...

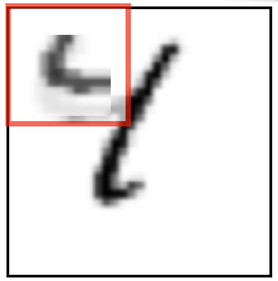
Pass the filter over the image
(Convolutions)

Aggregate the output of
the filter
(Pooling)

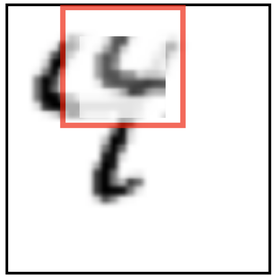
Image



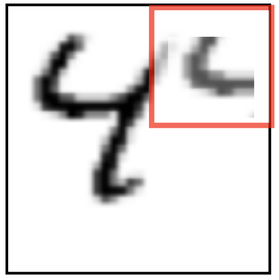
Filter (Kernel)



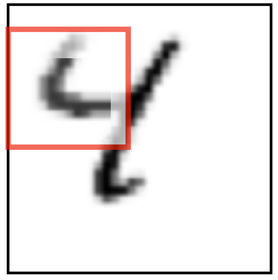
No detection
Output: Low



No detection
Output: Low



No detection
Output: Low



Detection
Output: High

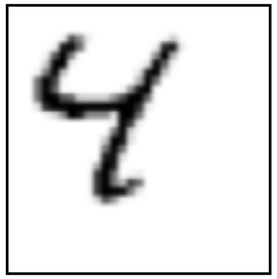
...

One of the filters detected an object part
Output: High

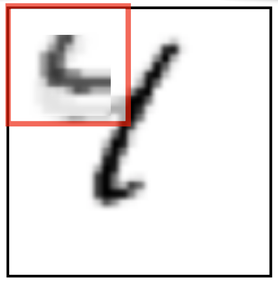
Pass the filter over the image (Convolutions)

Aggregate the output of the filter (Pooling)

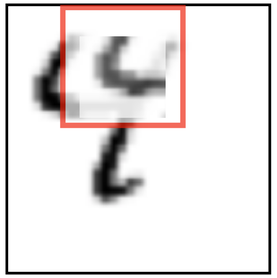
Image



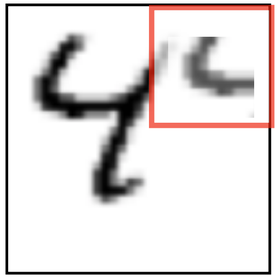
Filter (Kernel)



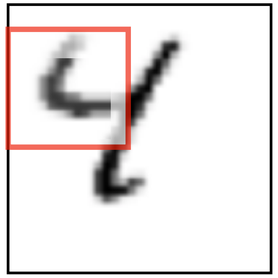
No detection
Output: Low



No detection
Output: Low



No detection
Output: Low



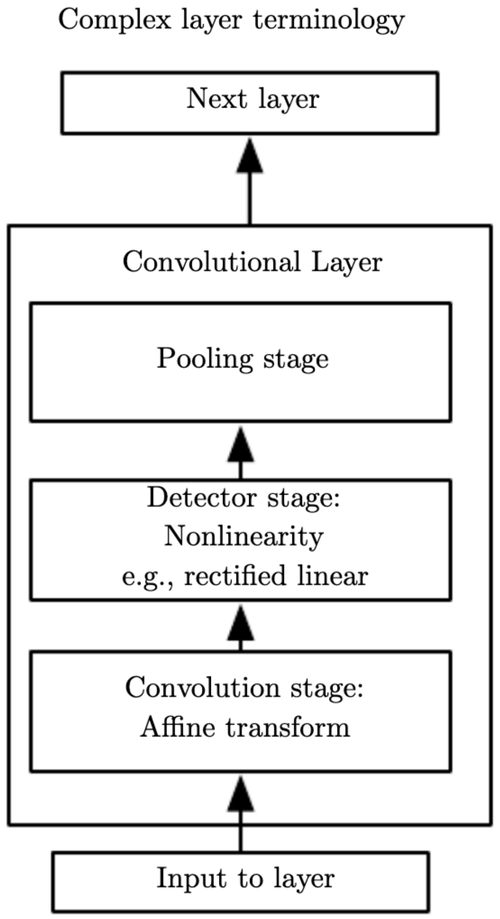
Detection
Output: High

...

One of the filters detected an object part
Output: High

To do well you need to:
1) learn the filters;
2) use many filters.

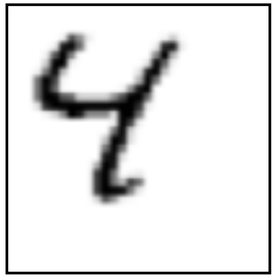
A layer in a CNN



[Figure 9.7, Deep Learning, Book]

Convolutions and Pooling

Convolutions (to the rescue)



For pixel (i,j):

$$S(i,j) = (K * I)(i,j) = \sum_{m}^{\text{k. width}} \sum_{n}^{\text{k. height}} I(i+m, j+n) K(m,n)$$

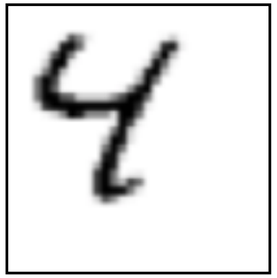
Kernel

Input

- **Dot product between “the kernel and the region”**

[Figure 9.1, Deep Learning, Book]

Convolutions (to the rescue)

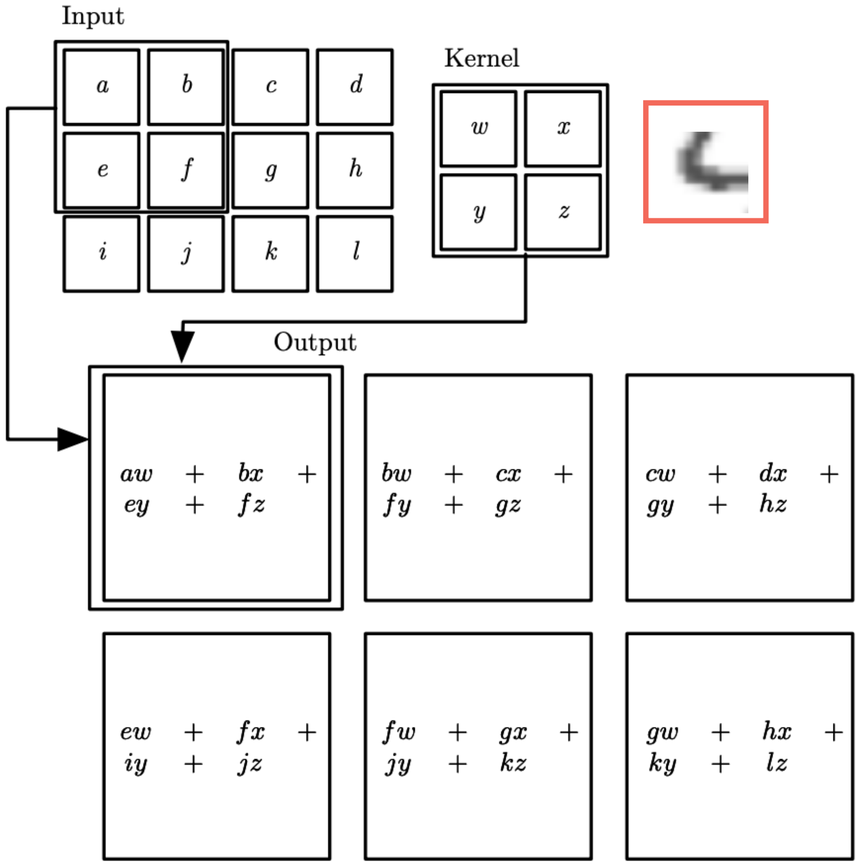


For pixel (i,j):

$$S(i,j) = (K * I)(i,j) = \sum_{m}^{\text{k. width}} \sum_{n}^{\text{k. height}} I(i+m, j+n) K(m,n)$$

Kernel

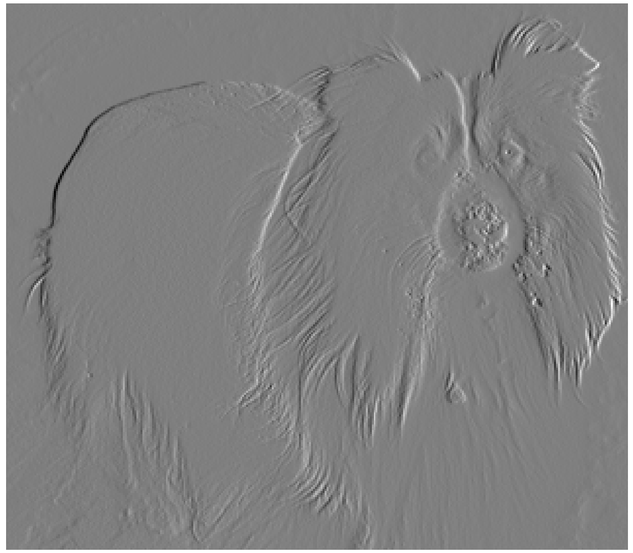
Input



- **Dot product between “the kernel and the region”**

[Figure 9.1, Deep Learning, Book]

Input

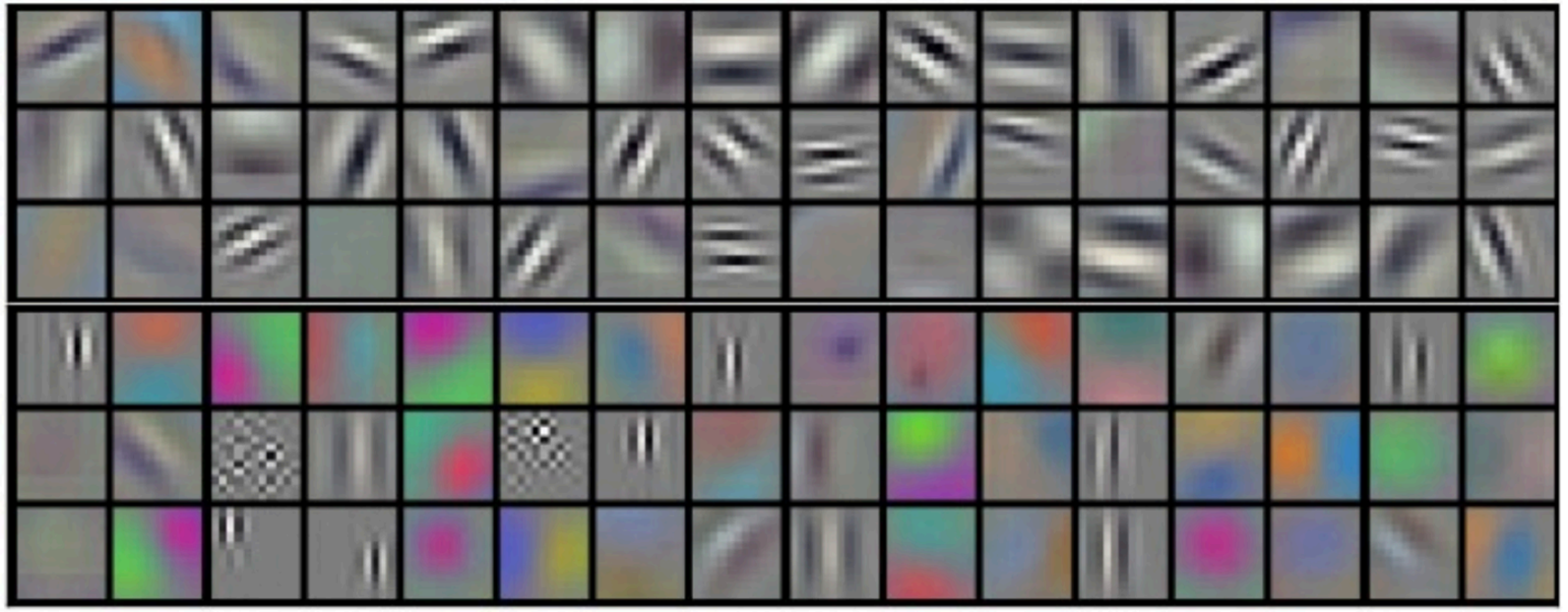


Output

-1	1

Kernel

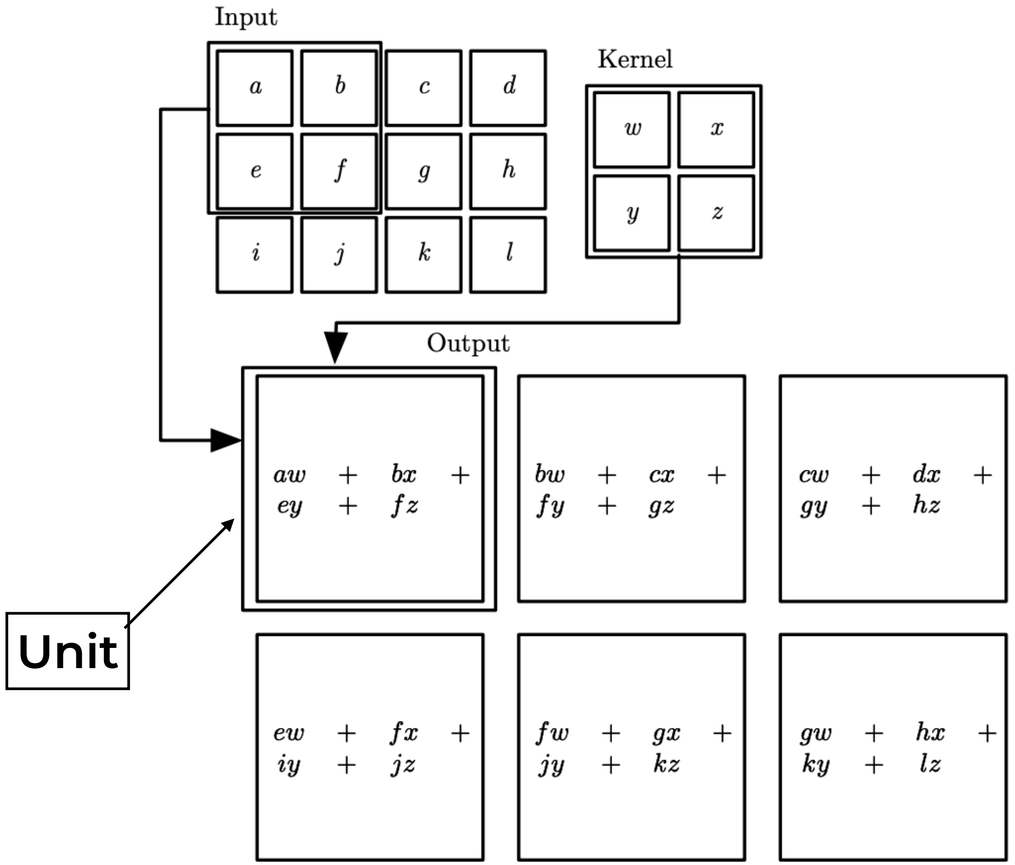
[Figure 9.6, Deep Learning, Book]

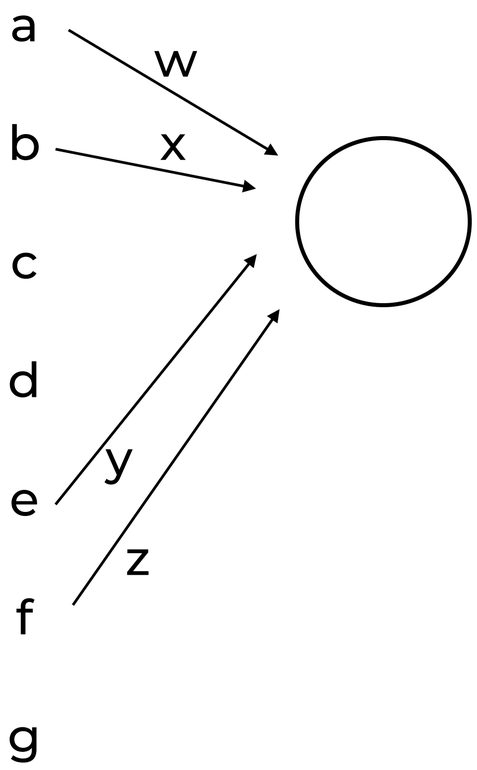


- **Learned kernels (filters)**

[https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf]

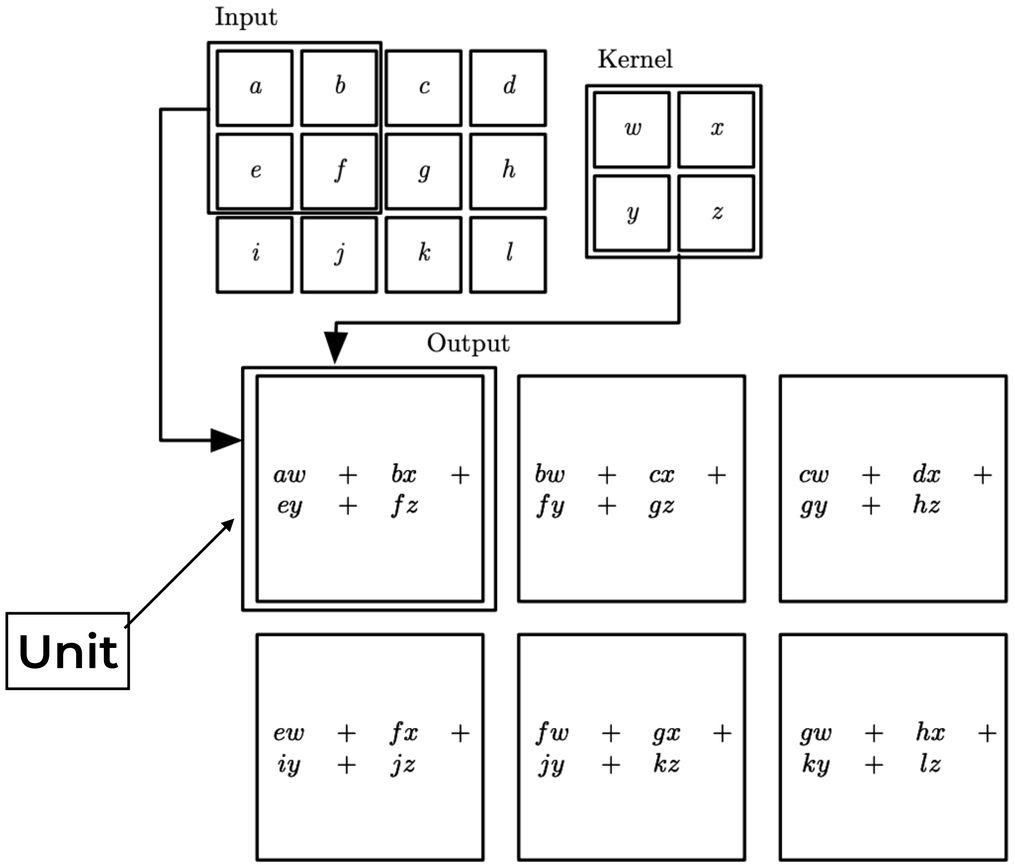
Sparse connections and shared weights

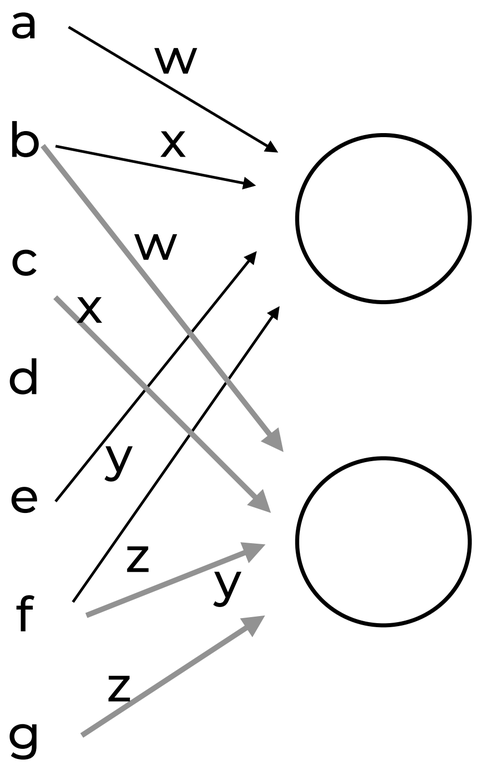




[Figure 9.1, Deep Learning, Book]

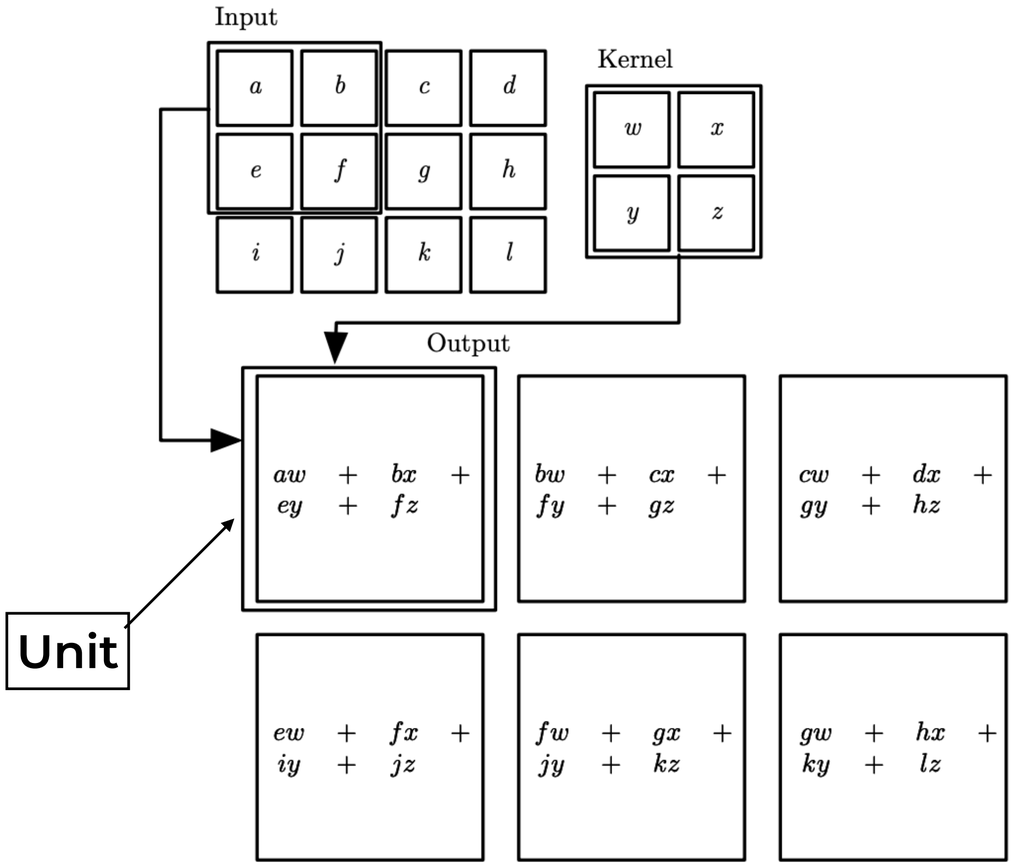
Sparse connections and shared weights

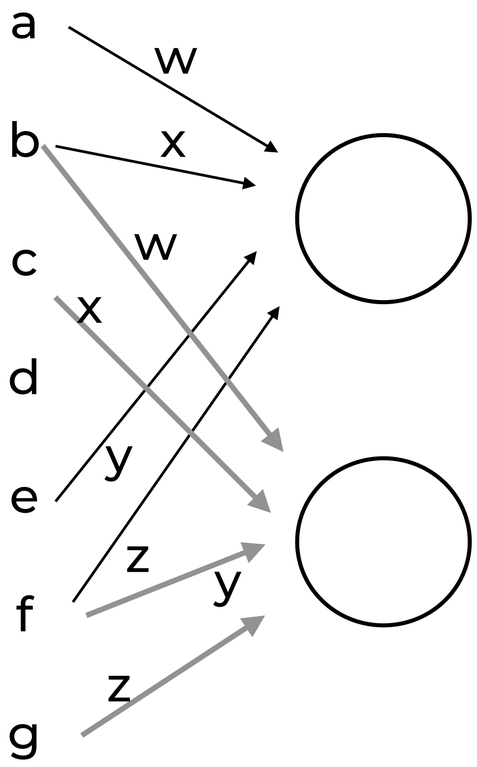




[Figure 9.1, Deep Learning, Book]

Sparse connections and shared weights

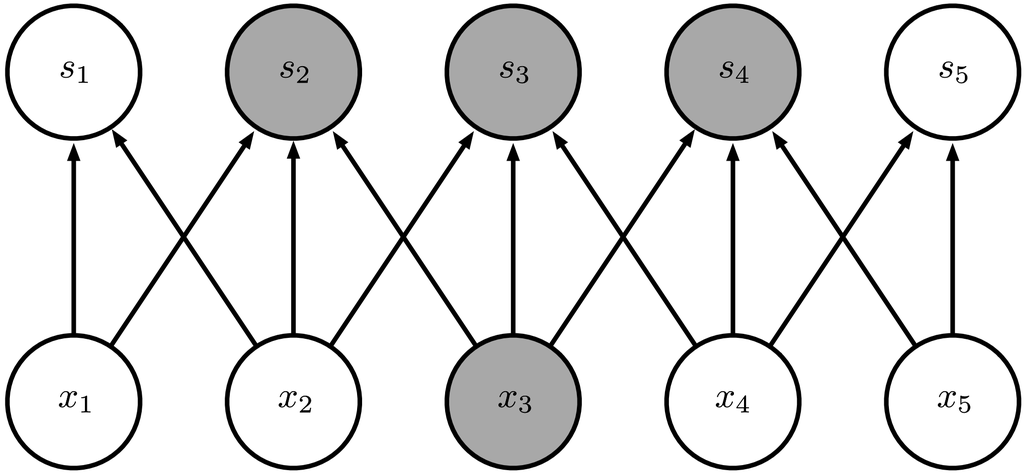




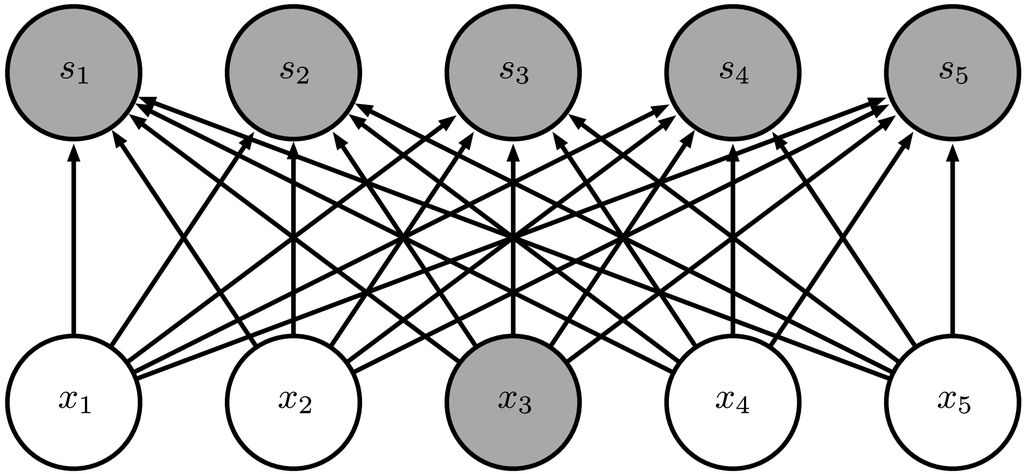
- Kernels induce sparse and shared connections
- Kernels must be small compared to the data

[Figure 9.1, Deep Learning, Book]

Sparse Connections

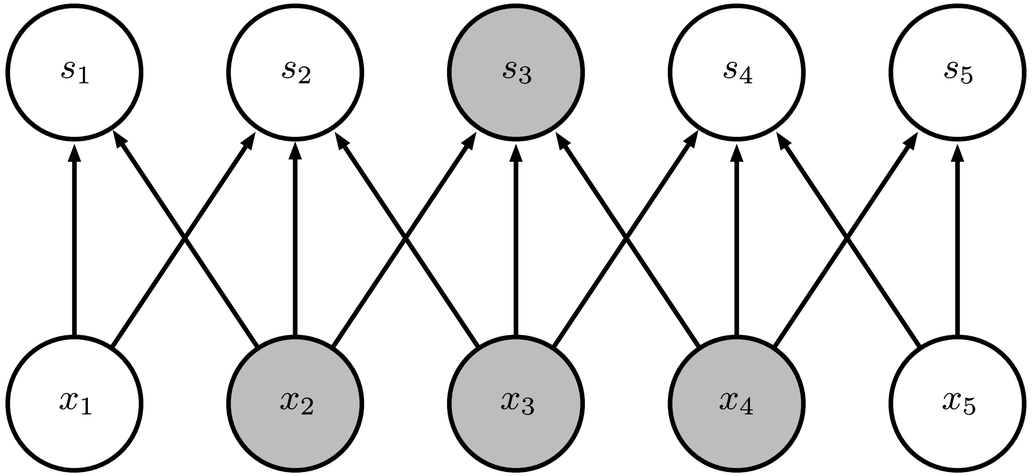


Dense Connections

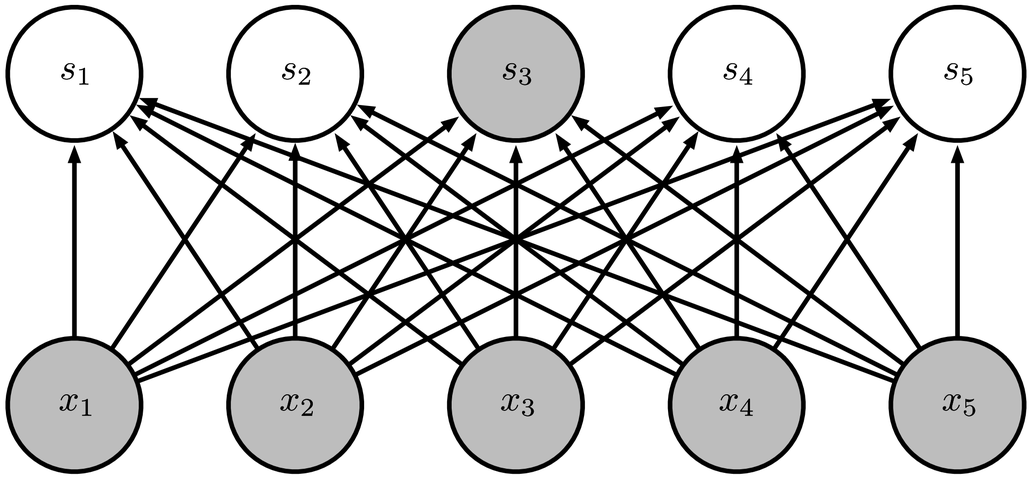


[Figure 9.2, Deep Learning, Book]

Sparse Connections



Dense Connections



[Figure 9.3, Deep Learning, Book]

Remarks

- Sparsity can reduce the number of parameters
  - Fully connected: O(m x n)
  - Sparsely connected: O(m x k)
    - In practice for image data we can use k << n

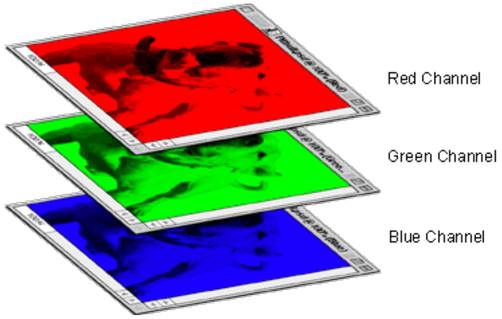
Remarks

- Sparsity can reduce the number of parameters
  - Fully connected: O(m x n)
  - Sparsely connected: O(m x k)
    - In practice for image data we can use k << n
- Parameter sharing reduces the memory requirements of the network

Remarks

- Sparsity can reduce the number of parameters
  - Fully connected: O(m x n)
  - Sparsely connected: O(m x k)
    - In practice for image data we can use k << n
- Parameter sharing reduces the memory requirements of the network
- Convolution is usually followed by a non-linear transformation (detector stage)

- We assumed grayscale images
- The same principle applies to color images where you learn kernels for each RGB channel
  - In that case an image is encoded as a 3D tensor
  - Kernels are also 3D



[http://www.sketchpad.net/channels1.htm]

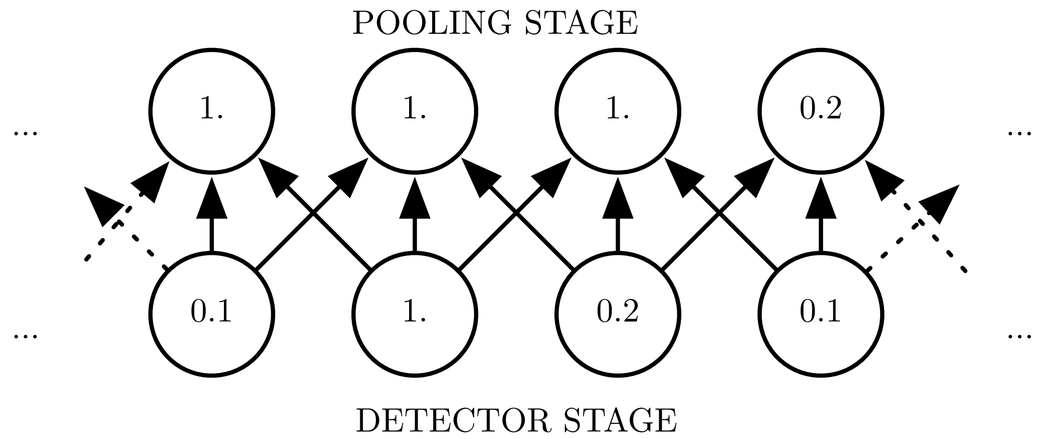
Pooling

- Make the representation invariant to small translations in the input
  - “Pool” the value of neighbour units
  - E.g., max-pooling takes the max from its input.

[Figure 9.8, Deep Learning, Book]

Pooling

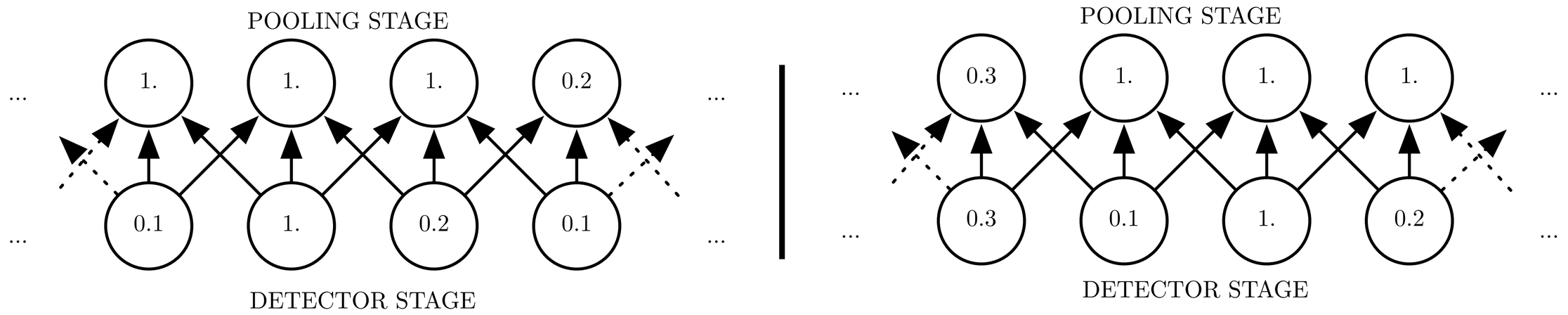
- Make the representation invariant to small translations in the input
  - “Pool” the value of neighbour units
  - E.g., max-pooling takes the max from its input.



[Figure 9.8, Deep Learning, Book]

Pooling

- Make the representation invariant to small translations in the input
  - “Pool” the value of neighbour units
  - E.g., max-pooling takes the max from its input.

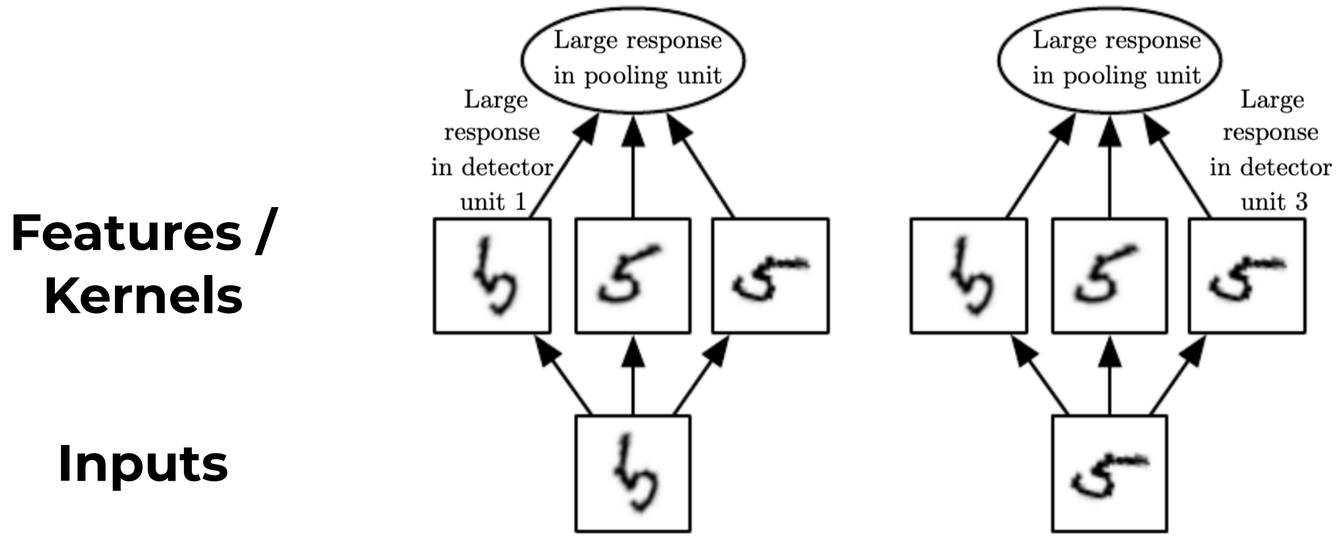


[Figure 9.8, Deep Learning, Book]

Pooling (II)

- Pooling over different features can learn other types of invariances

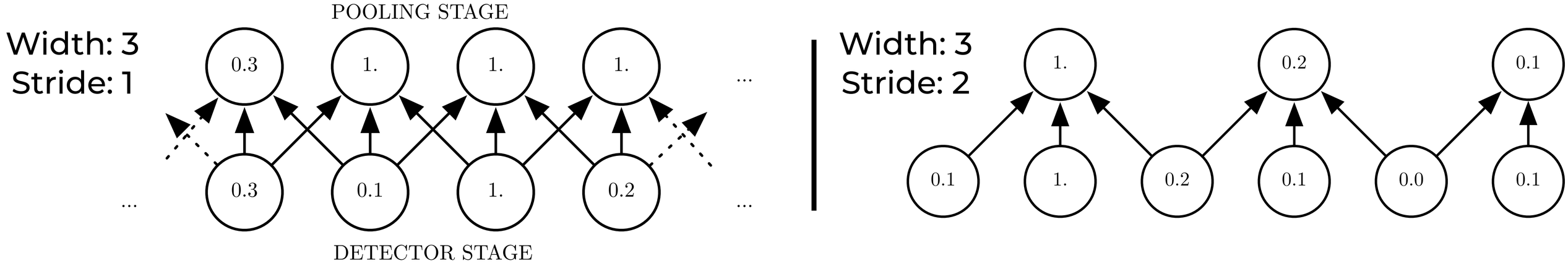
  - Here, the model learns to be invariant to certain rotations:



[Figure 9.9, Deep Learning, Book]

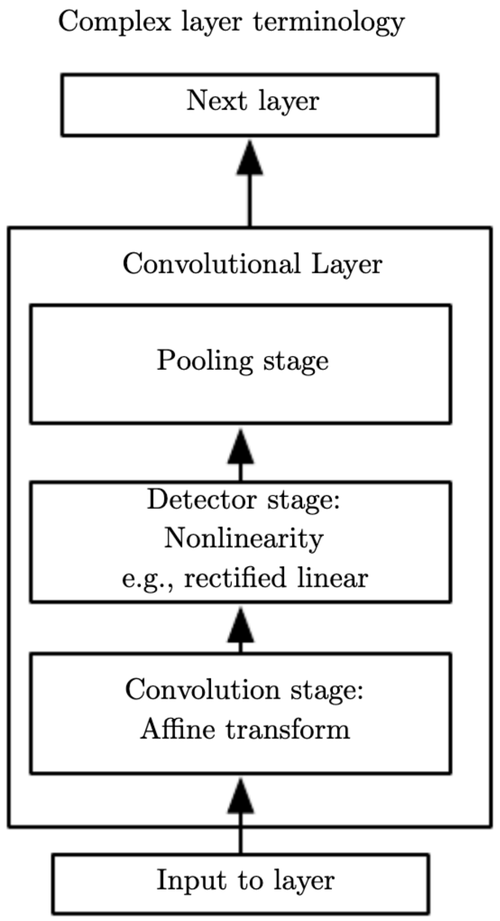
Pooling (III)

- Often used to reduce the dimensionality
  - Two parameters:
    - *Width:* size of neighbourhood to pool from
    - *Stride:* space between different neighbourhoods

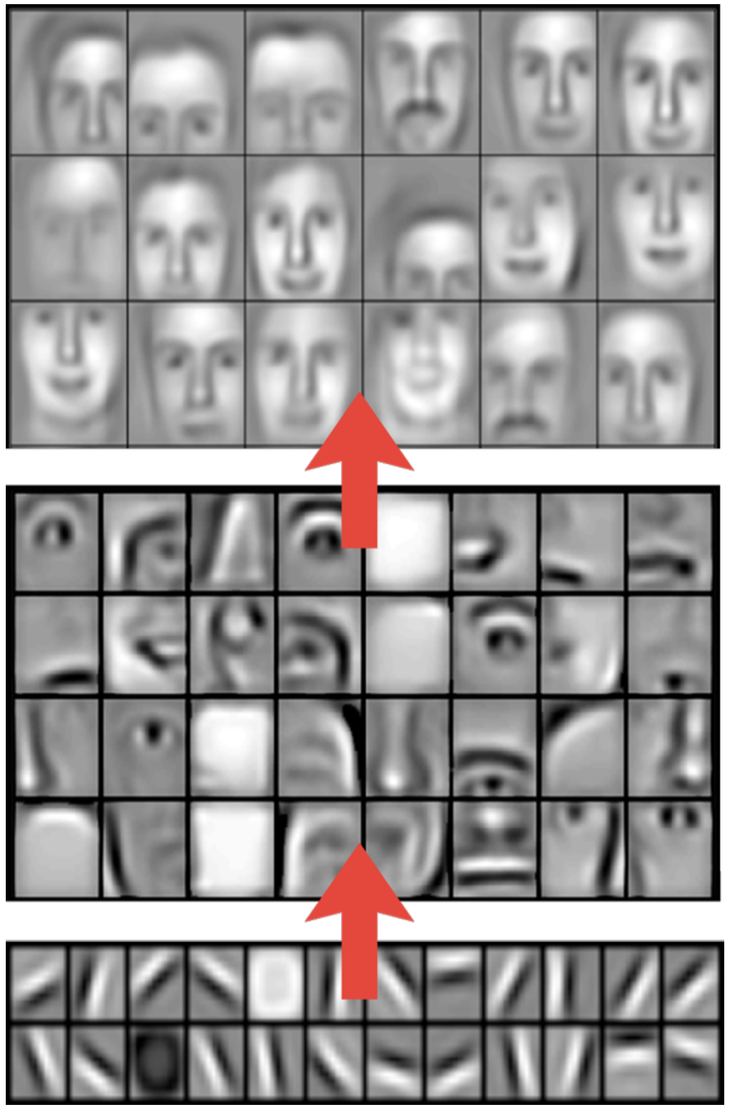


[Figure 9.10, Deep Learning, Book]

Putting it all together



[Figure 9.7, Deep Learning, Book]

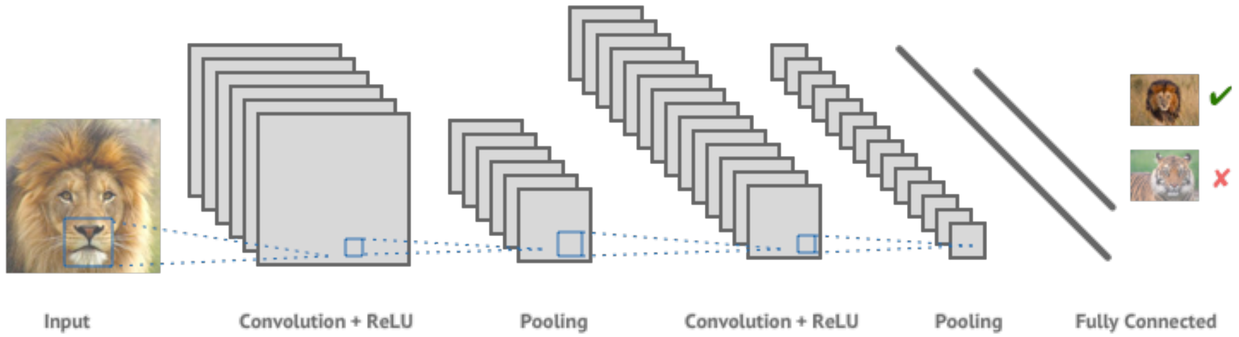


[From Yoshua Bengio]

Conclusions and a few practical aspects

Typical Task

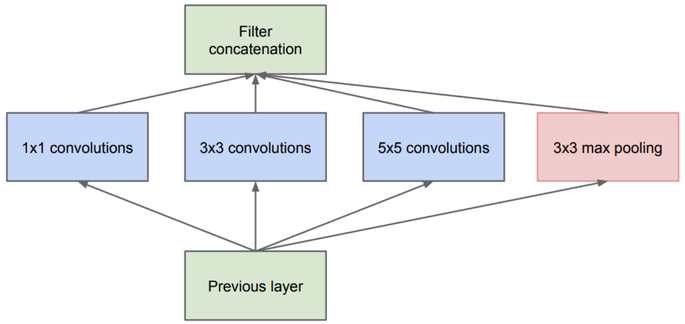
- Object recognition
  - Architectures:
    - Repeat: Conv-Relu layers followed by pooling
    - Last layer is fully-connected



[https://blog.goodaudience.com/convolutional-neural-net-in-tensorflow-e15e43129d7d]

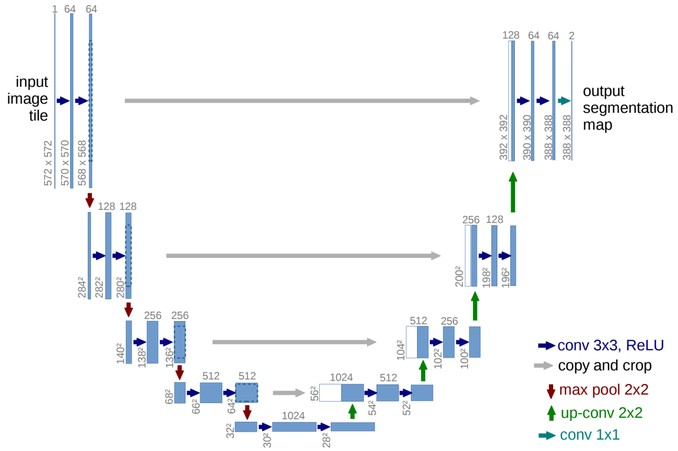
Variety of architectures

Inception



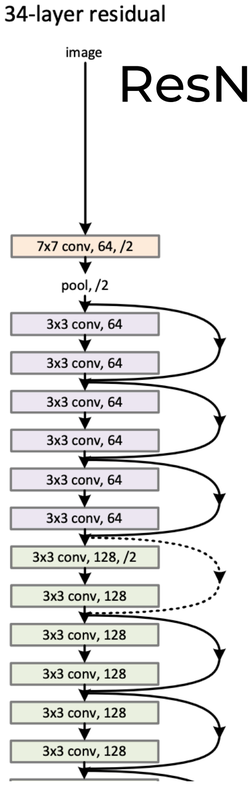
[https://arxiv.org/pdf/1409.4842v1.pdf]

U-nets



[https://arxiv.org/pdf/1505.04597.pdf]

ResNets



[https://arxiv.org/pdf/1512.03385.pdf]

Pretrained models are available

- https://keras.io/applications/

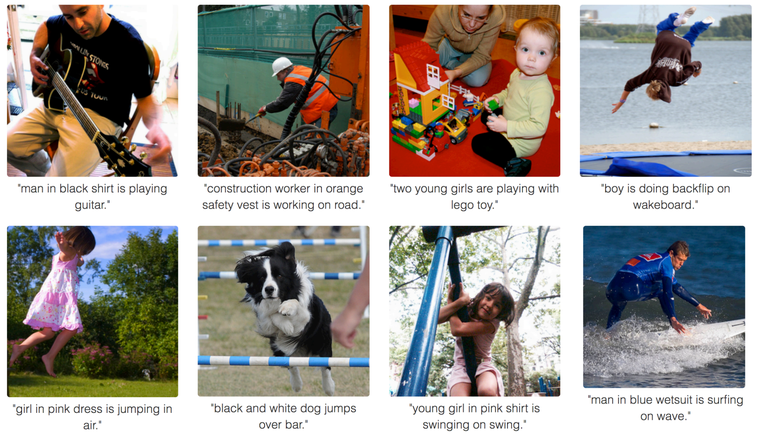
Documentation for individual models

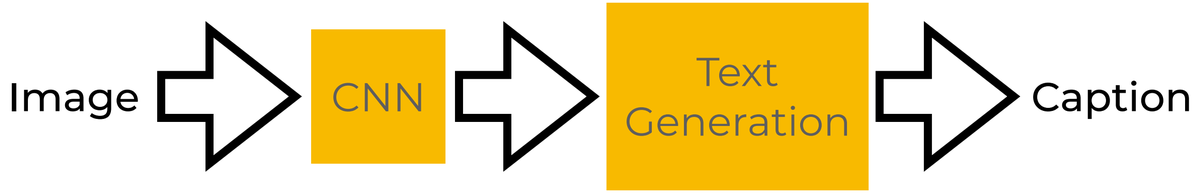
Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	98 MB	0.749	0.921	25,636,712	-
ResNet101	171 MB	0.764	0.928	44,707,176	-
ResNet152	232 MB	0.766	0.931	60,419,944	-
ResNet50V2	98 MB	0.760	0.930	25,613,800	-
ResNet101V2	171 MB	0.772	0.938	44,675,560	-
ResNet152V2	232 MB	0.780	0.942	60,380,648	-
ResNeXt50	96 MB	0.777	0.938	25,097,128	-
ResNeXt101	170 MB	0.787	0.943	44,315,560	-
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
DenseNet121	33 MB	0.750	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201
NASNetMobile	23 MB	0.744	0.919	5,326,716	-
NASNetLarge	343 MB	0.825	0.960	88,949,818	-

The top-1 and top-5 accuracy refers to the model's performance on the ImageNet validation dataset.

CNNs can be used as modules inside larger networks

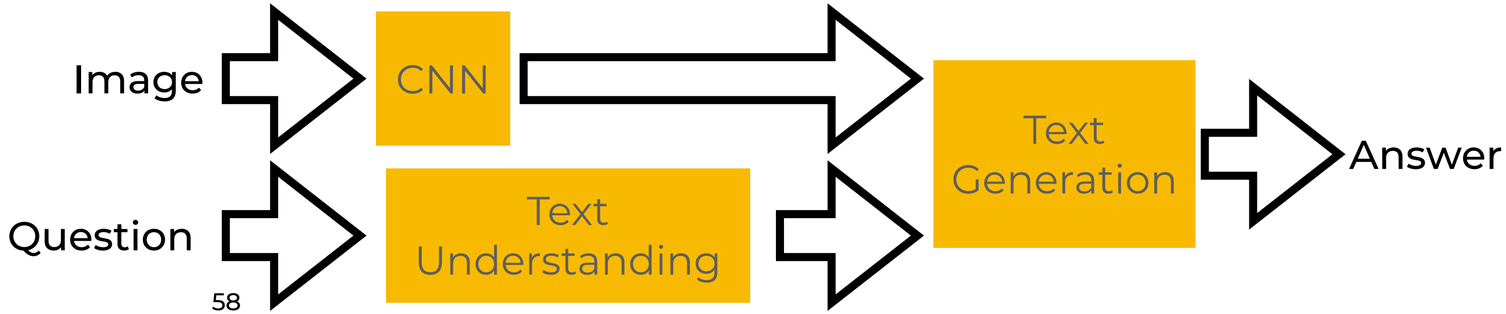
- **Image Captioning**





[https://cs.stanford.edu/people/karpathy/cvpr2015.pdf]

- **Visual Question Answering**



[https://arxiv.org/pdf/1612.00837.pdf]

Conclusions

- CNNs and RNNs are two classes of neural networks

  1. Lift limitations of feed-forward networks

  2. Allow modelling of sequential and image data

  3. Can be composed and learned end-to-end

Conclusions

- CNNs and RNNs are two classes of neural networks

  1. Lift limitations of feed-forward networks

  2. Allow modelling of sequential and image data

  3. Can be composed and learned end-to-end

- Other classes may emerge (e.g., graph networks)

Data

Neural Network Model

Data

Neural Network Model

Data

Neural Network Model

Convolutional Neural Network

Data

Neural Network Model

Convolutional Neural Network

Recurrent Neural Network
& Transformers

Data

Neural Network Model

Convolutional Neural Network

Recurrent Neural Network
& Transformers

Data	Neural Network Model
	Convolutional Neural Network
	Recurrent Neural Network & Transformers
	Graph Neural Networks (Graph Convolutional Networks)