ACDS Lecture Series

Lecture - 12

CNN and RNN

G. N. Sastry and Team

ADVANCED COMPUTATION AND DATA SCIENCES (ACDS) DIVISION

CSIR-North East Institute of Science and Technology, Jorhat, Assam, India

12.3 Recurrent Neural Network

12.4 Exercises

The Topics Covered in This Section

12.1.2 Motivation
12.1.2 ML vs. DL
12.1.3 Applications

12.1.1 Motivation

- Everyone is talking about it because a lot of money is invested in it:
  - DeepMind: Acquired by Google for $400 million.
  - DNNResearch: Startup (including Hinton) acquired by Google for unknown price.
  - Enlitic, Ersatz, MetaMind, Nervana, Skylab: DL startups commanding millions of VC dollars.
- Deep learning has won numerous challenges and does so with minimal feature engineering.
- Since 1980s: Form of models hasn't changed much, but lots of new tricks.
  - More hidden units
  - Better (online) optimization
  - New nonlinear functions (ReLUs)
  - Faster computers (CPUs and GPUs)

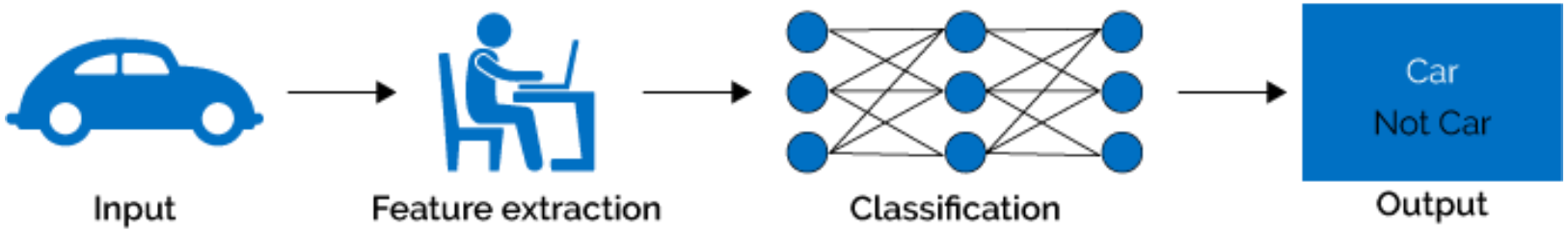
Definition:

Yann LeCun [1919]

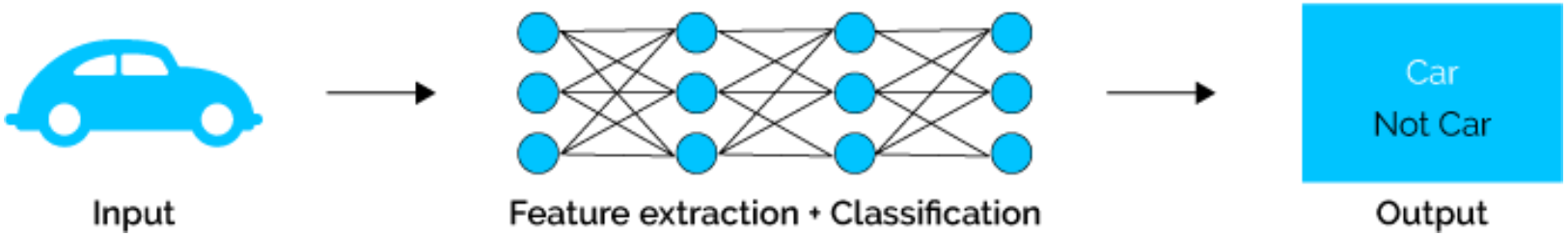
Deep learning is constructing networks of parameterized functional modules and training them from examples using gradient descent optimization.

- Deep Learning term introduced to the ML community by Rina Dechter [1986].
- DL is part of a broader family of machine learning methods based on ANN.
- DL uses multiple layers to progressively extract higher level features from the raw input.

Machine Learning



Deep Learning



- Like ML, Deep Learning can be supervised, semi-supervised or Reinforcement.

Image Classification

Image Captioning

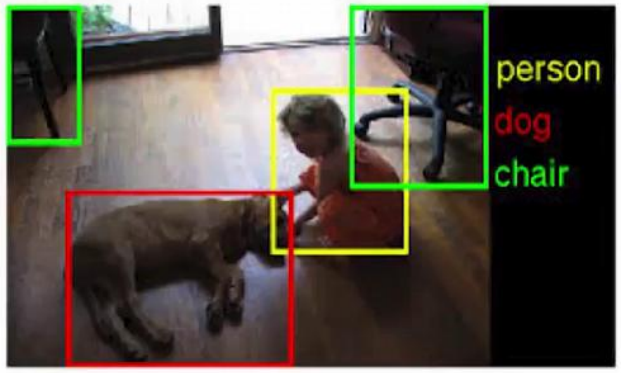
1. Top view of the lights of a city at night, with a well-illuminated square in front of a church in the foreground;
2. People on the stairs in front of an illuminated cathedral with two towers at night;

A square with burning street lamps and a street in the foreground;

1. Tourists are sitting at a long table with beer bottles on it in a rather dark restaurant and are raising their bierglaeser;
2. Tourists are sitting at a long table with a white table-cloth in a somewhat dark restaurant;

Tourists are sitting at a long table with a white table cloth and are eating;

Object Detection and Recognition



12.1.3 Applications

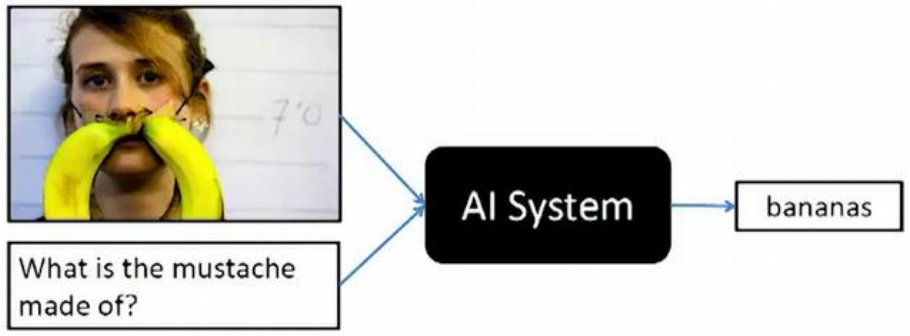
Video Captioning

A group of young men playing a game of soccer

A man riding a wave on top of a surfboard.

Video Summarization

Visual Question Answering

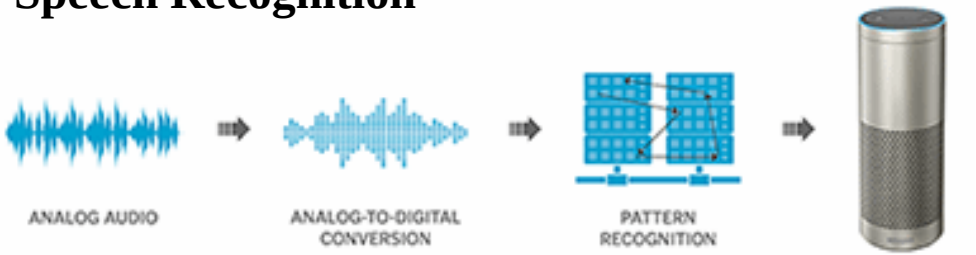


Question: What is the cat doing? Answer: playing with a tablet

Textual Question Answering

Joe went to the garden then Fred picked up the milk; Joe moved to the bathroom and Fred dropped the milk, and then Dan moved to the living_room.
Where is Dan? A: living room I believe
Where is Joe? A: the bathroom
Fred moved to the bedroom and Joe went to the kitchen then Joe took the milk there and Dan journeyed to the bedroom; Joe discarded the milk.
Where is the milk now ? A: the milk is in the kitchen
Where is Dan now? A: I think he is in the bedroom
Joe took the milk there, after that Mike travelled to the office, then Joe went to the living_room, next Dan went back to the kitchen and Joe travelled to the office.
Where is Joe now? A: I think Joe is in the office

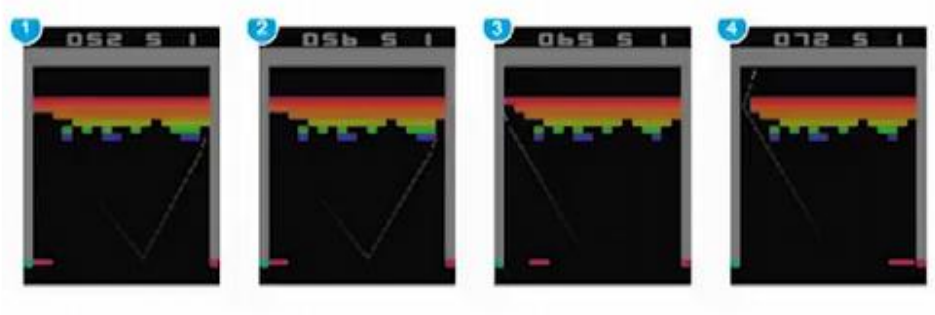
Speech Recognition



Speech to Text and Text to Speech Conversion

12.1.3 Applications

- Atari Player (through reinforcement learning)



- Alpha Go Zero (through deep reinforcement learning)

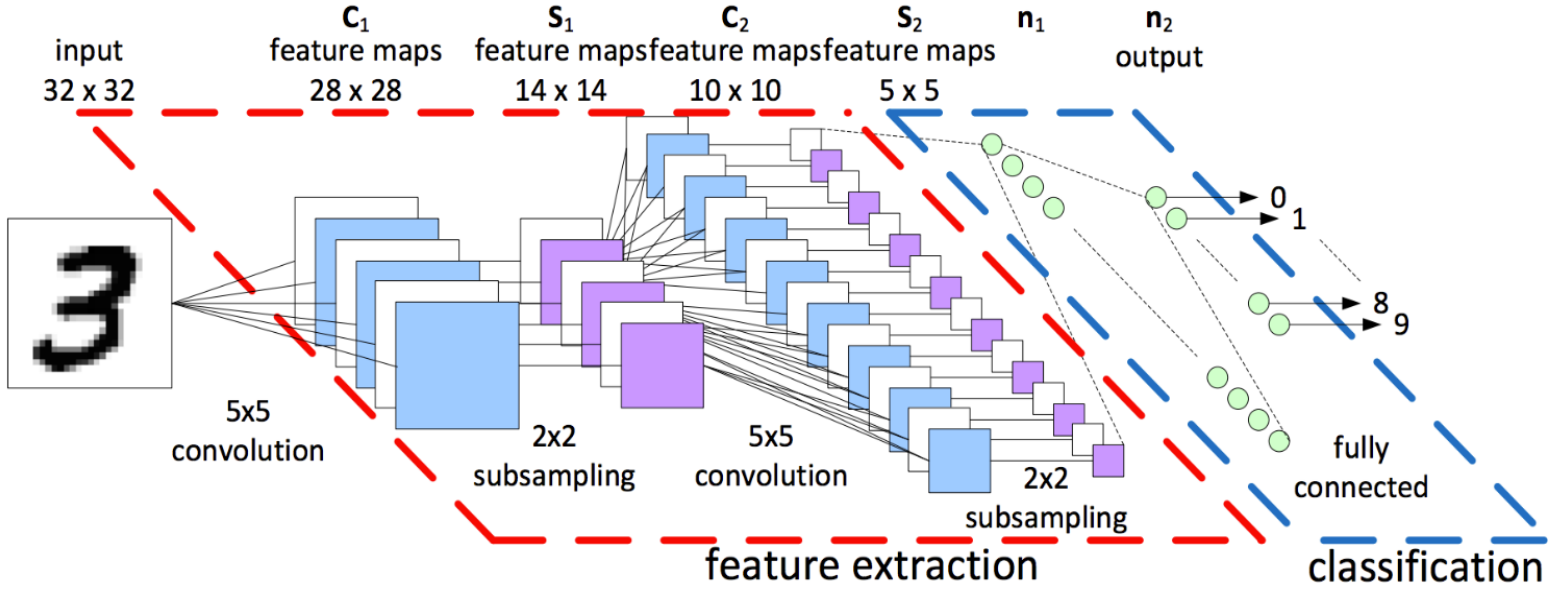
The Topics Covered in This Section

12.2.1 Introduction
12.2.2 Regular ANN vs. CNN
12.2.3 CNN Topology
12.2.4 Convolutional Layer
12.2.5 Pooling Layer
12.2.6 Fully Connected Layer
12.2.7 CNN for Text
12.2.8 DNN to CNN
12.2.9 Optimization
12.2.10 Choosing Hyperparameters
12.2.11 Regularization
12.2.12 Different CNN Architectures
12.2.13 CNN Disadvantages
12.2.14 CNN Summary
12.2.15 Exercises

- A kind of Feed forward NN designed to work on images for computer vision problems.
- Introduced by Yann LeCun [1995].

Applications

- Image object detection and recognition.
- Image classifications.
- Semantic segmentation



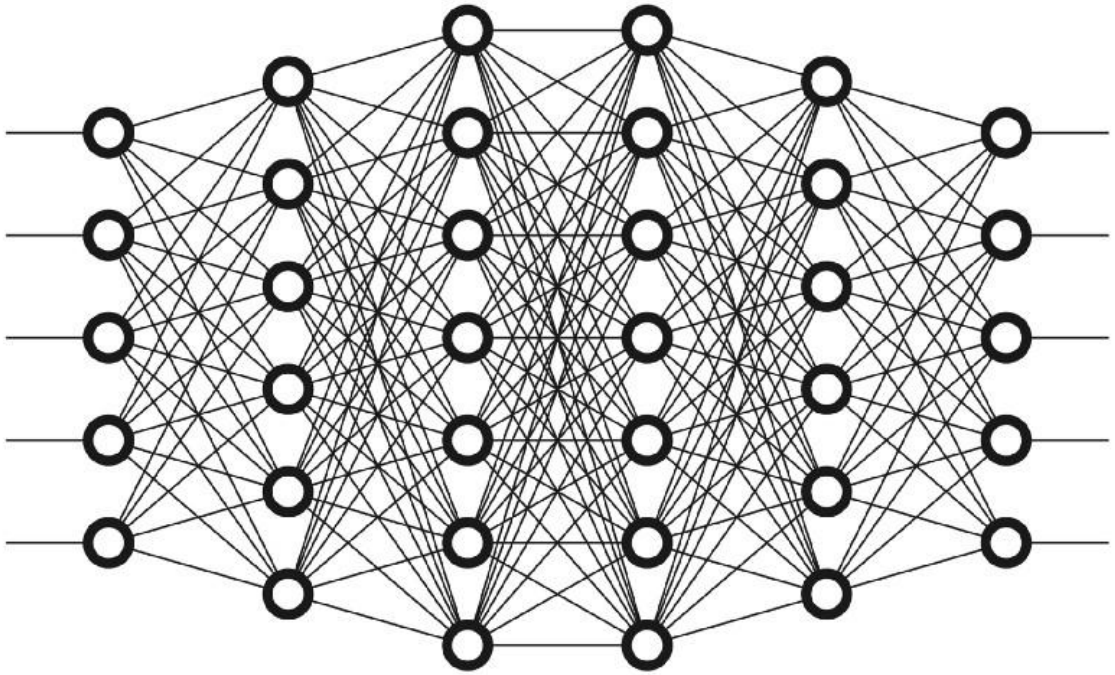
Algorithms to train CNN

- Backpropagation (using Gradient Descent)

Why Not Regular NN?

- Regular NN are good for already defined features.
- We need CNN order to effectively solve the problems that image data can present!
- CNN implicitly extracts relevant features.

Images with 200 * 200 * 3 pixels

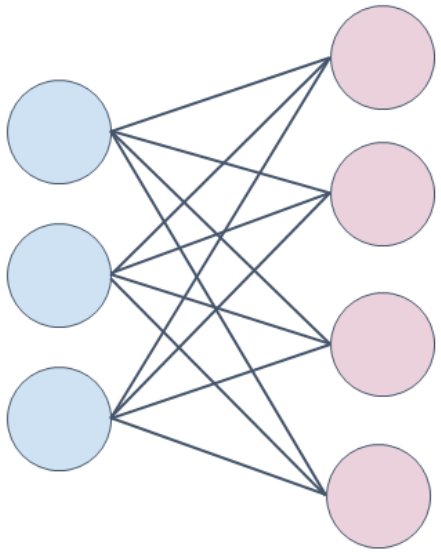


Numbers of weights For the first hidden layer is 120,000

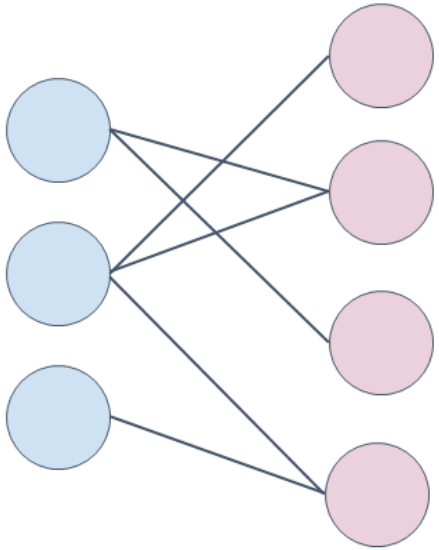
How CNN Deals?

- Since the input to these networks are images, the architecture got modified from regular ANN.
  - **Sparse Connections**: Less connections means less weights parameters.
  - **Parameters Sharing**: Reduces the number of weights to exploit local connectivity in image.
- Different nodes are responsible for different regions of the image.

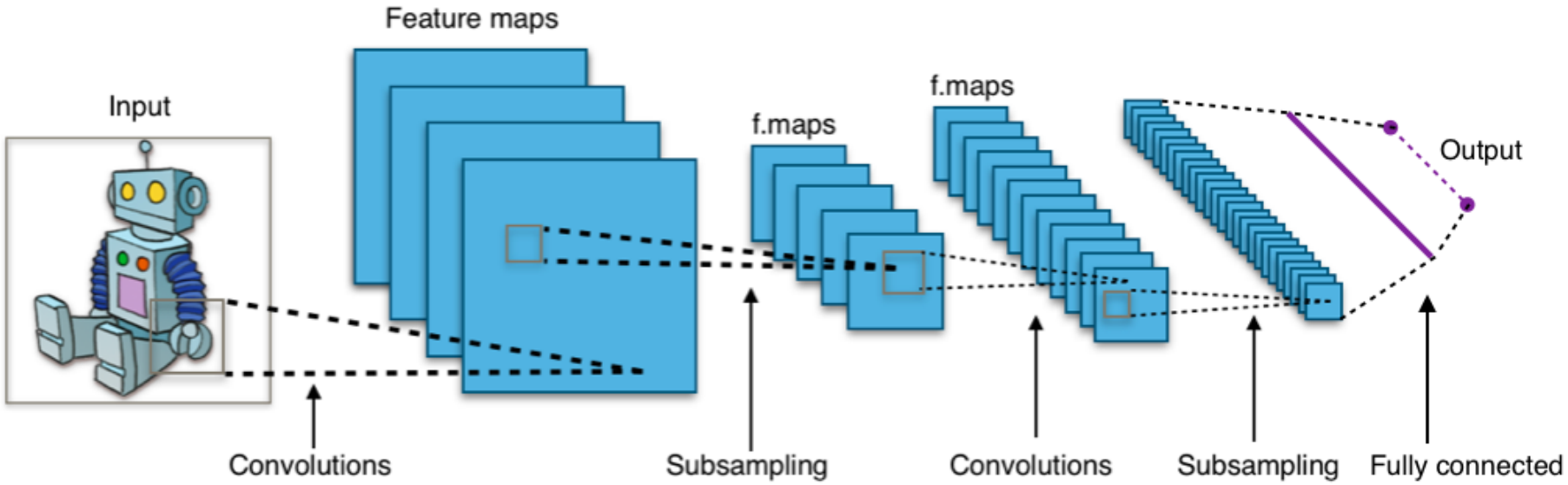
Dense Connectivity



Sparse Connectivity



- Convolution or Feature Extraction Layer (+ ReLU)
- Pooling or Subsampling Layer (max or average Pooling)
- Fully Connected + ReLU
- Softmax



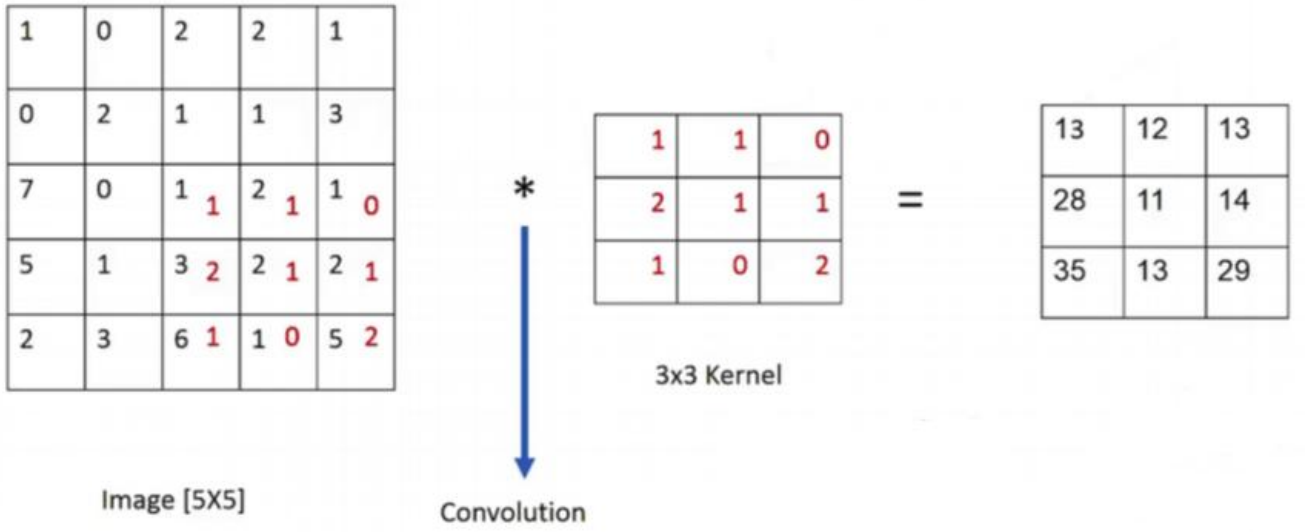
Note: CNN hidden layers are basically alternation of convolution and pooling layers.

Note: Though the layers are colloquially referred to as convolutions, this is only by convention. Mathematically, it is technically a *sliding dot product* or cross-correlation.

12.2.4 Convolutional Layer

Convolution

- Filter kernels if operated on an image able to extract different features like edge map from it.
- For every convolution layer we can just define multiple filter kernels.
- The output of every layer is connected to a small neighbourhood in the input.



n_x , n_y

f_x , f_y

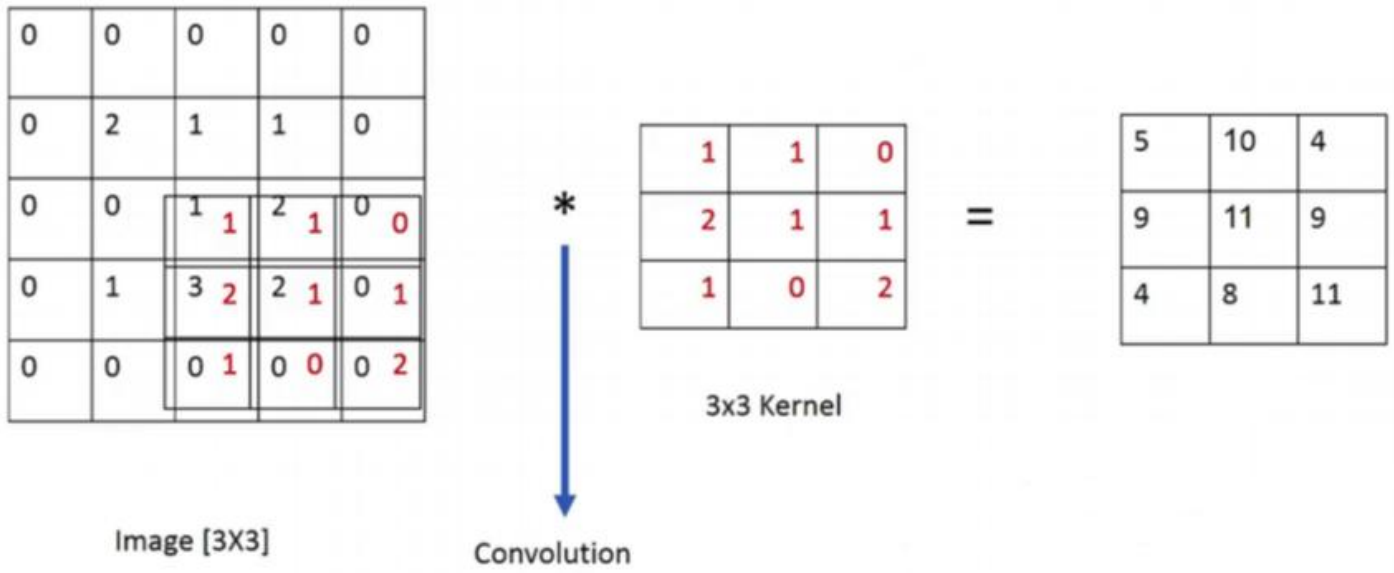
n_x-f_x+1 , n_y-f_y+1

Why Convolution

- **Parameter Sharing:** A feature detector such as vertical edge detector that is useful in one part of the image is probably useful in another part of the image.
- **Sparsity of connections:** In each layer, each output depends only on a small number of inputs.

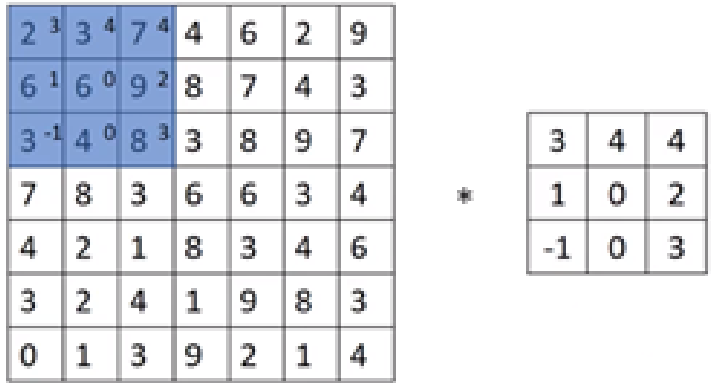
Padded Convolution

- Convolution operation reduces size of the feature maps so that at some time filtering not possible.
- Padded convolution helps to preserve the size of the output than the size of the input.



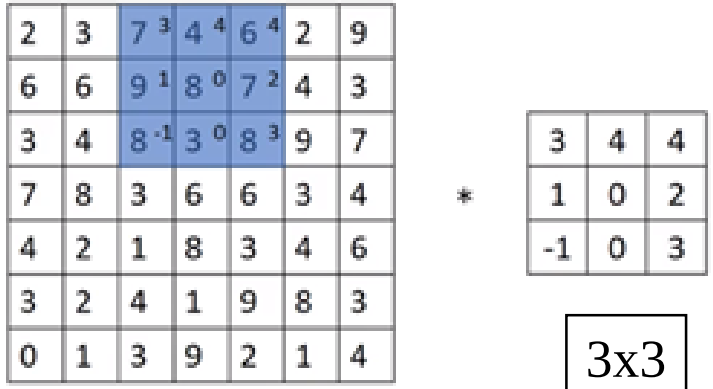
n_x , n_y f_x , f_y $n_x-f_x+1+2p , n_y-f_y+1+2p$

Stride Convolution



91	100	83
69	91	127
44	71	74

3x3



3x3

7x7

Stride, s=2

n_x , n_y

f_x , f_y

$$\frac{n_x - f_x + 2p}{s} + 1 \quad , \quad \frac{n_y - f_y + 2p}{s} + 1$$

12.2.4 Convolutional Layer

Note: Cross-correlation vs Convolution

- Convolution (Actually) in mathematics textbook

2	3	7	4	6	2
6	6	9	8	7	4
3	4	8	3	8	9
7	8	3	6	6	3
4	2	1	8	3	4
3	2	4	1	9	8

*

3	4	5
1	0	2
-1	9	7

7	2	5
9	0	4
-1	1	3

Receptive field (The input area of a neuron**)**

- In a fully connected layer, each neuron receives input from *every* element of the previous layer.
- In a convolutional layer, neurons receive input from only a subarea of the previous layer (e.g.5x5).
- So, in a fully connected layer, the receptive field is the entire previous layer.
- In a convolutional layer, the receptive area is smaller than the entire previous layer.

Summary of Convolution

$n \times n$ image $f \times f$ filter

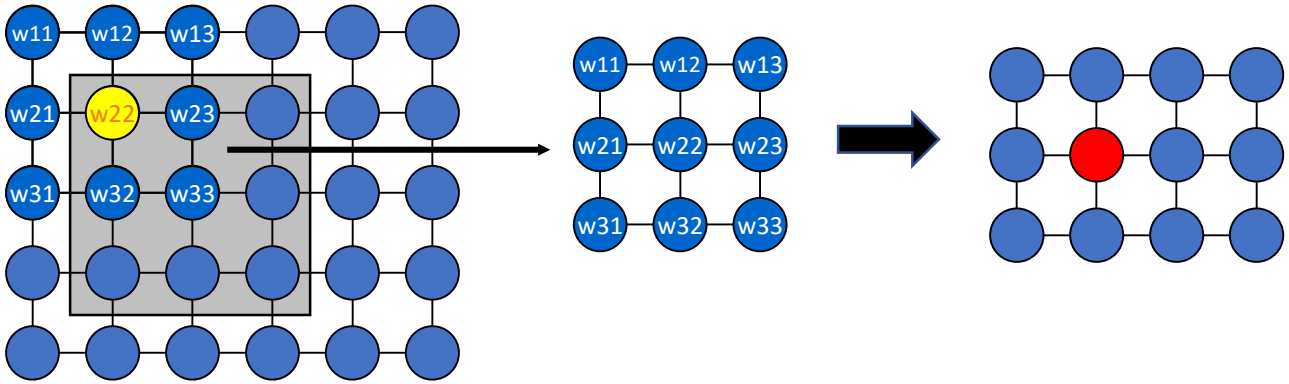
padding p stride s

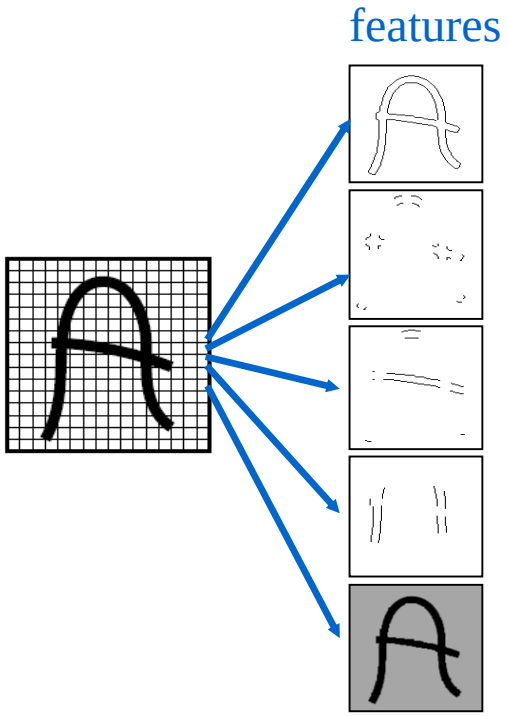
$$\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$$

Importance of Convolution Layer

- Apply a set of weights – a filter – to extract local features.
- Use multiple filters to extract different features.

Features extraction with convolution



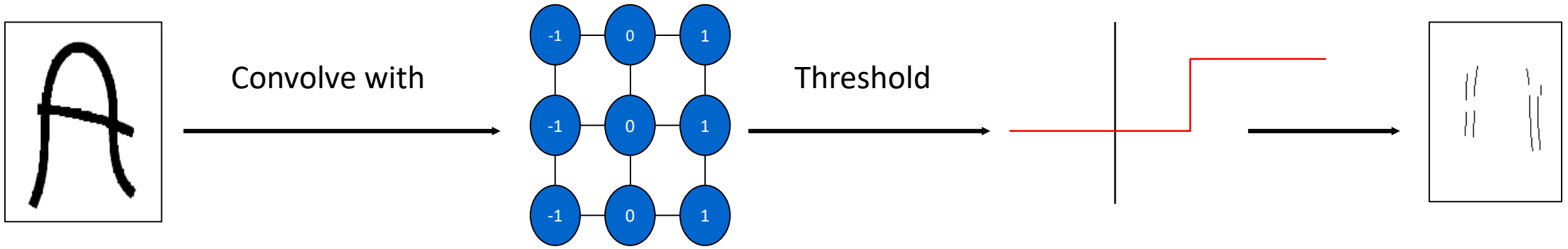


- Filter of size 4x4: 16 different weights
- Apply this same filter to 4x4 patches in input
- Shift by 2 pixels for next patch
- This patchy operation is convolution

12.2.4 Convolutional Layer

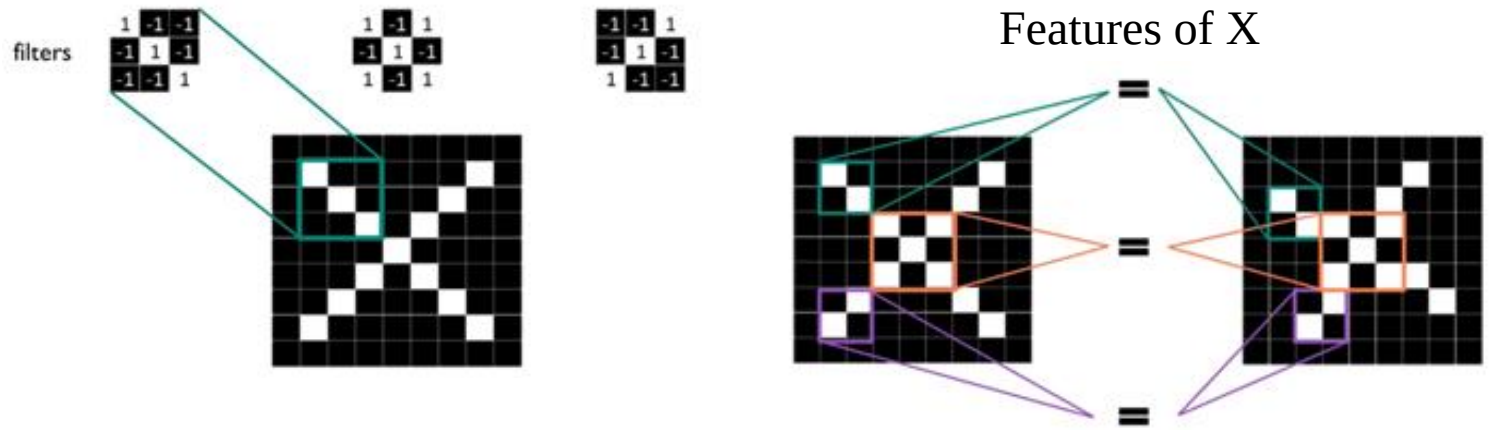
Extracting Features

- Detect the same feature at different positions in the input image.
- Shared weights: all neurons in a feature share the same weights (but not the biases).
- In this way all neurons detect the same feature at different positions in the input image.
- Reduce the number of free parameters.

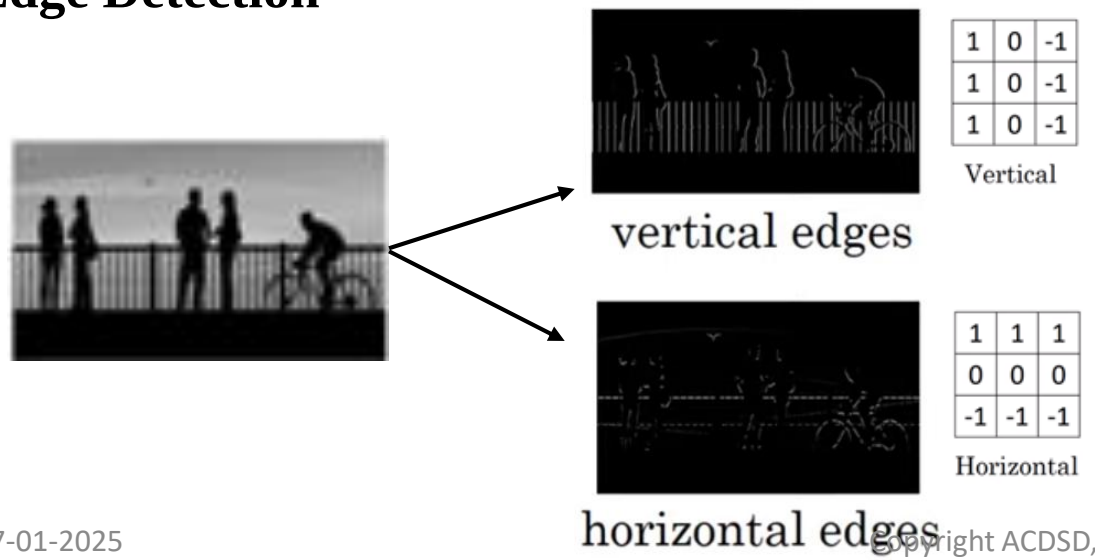


Note: For this reason CNN also known as **shift invariant ANN (SIANN)**, based on their shared-weights architecture and translation invariance characteristics.

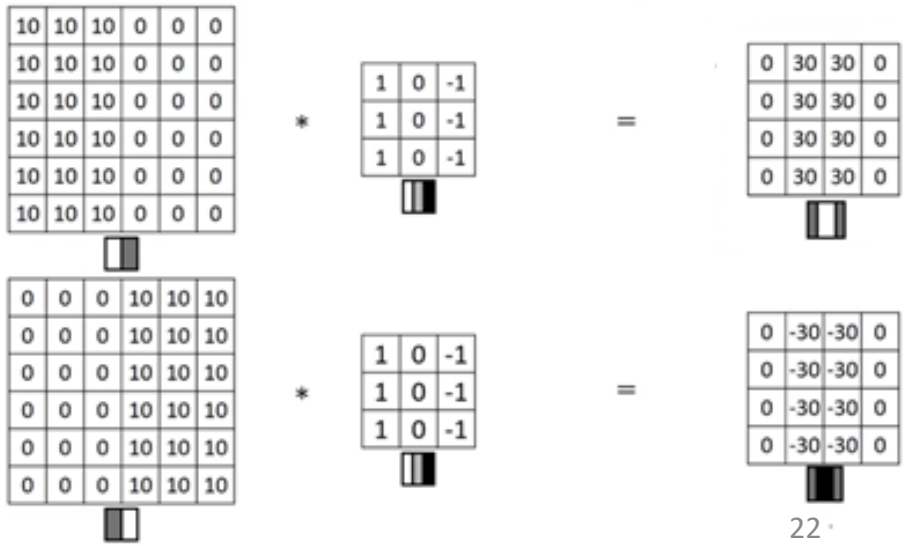
Filters to Detect X features



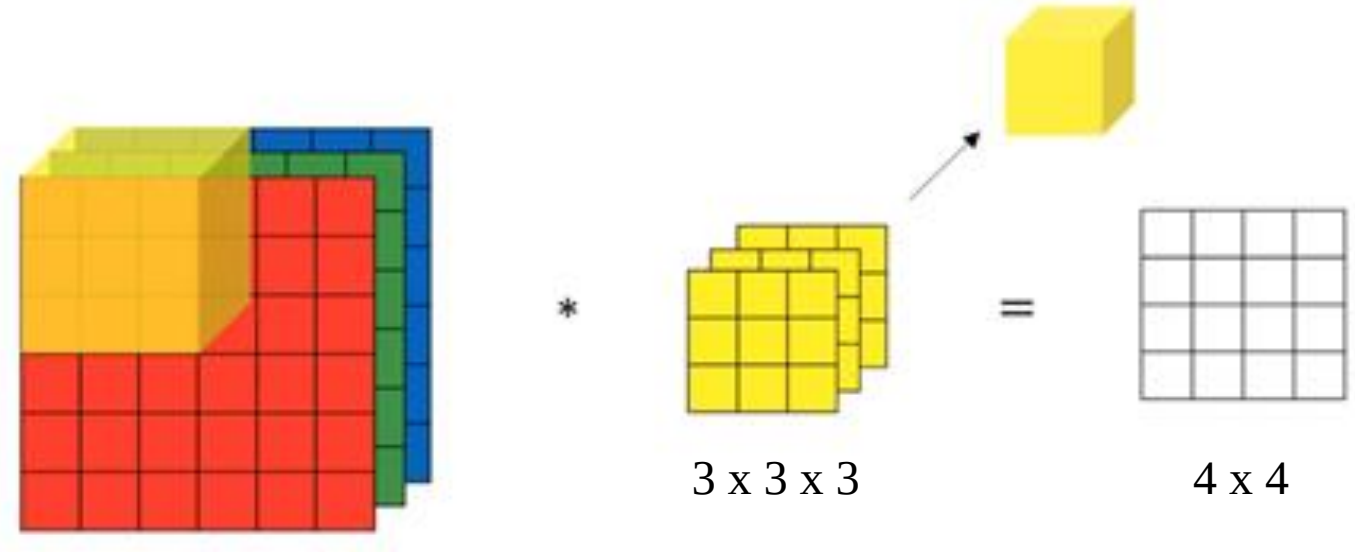
Edge Detection



Vertical Edge Detection Examples



Convolution on RGB images



Convolution Parameters

- For 5 number of 3x3x3 filter the number of parameters for one single layer will be –

➢ 5 x [(3x3x3) +1] = 140

12.2.5 Pooling Layer

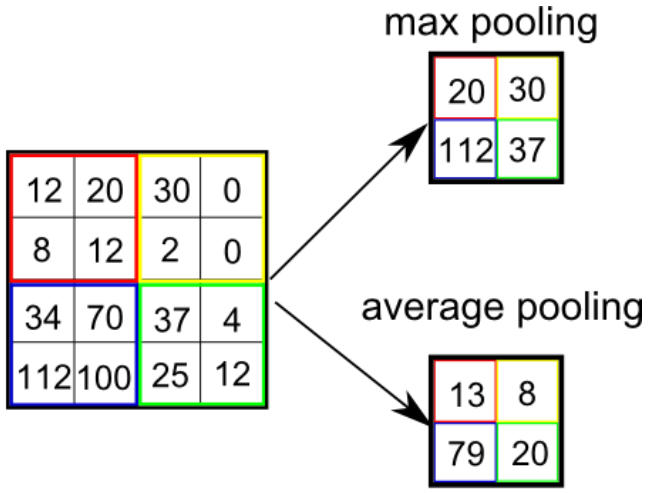
- Subsample the feature maps i.e. reduces the size of the feature maps.
- Average pooling and Max polling are commonly used.
- Networks may include local or global pooling layers to streamline the underlying computation.

Parameters

- No Parameters
- Means gradient descent has to nothing to learn

Hyperparameters

- f: Filter Size (here 2)
- s: Stride (here 2)



Feature Maps

6	4	8	5
5	4	5	8
3	6	7	7
7	9	7	2

n_x , n_y

2x2 filters and stride 2

f_x , f_y

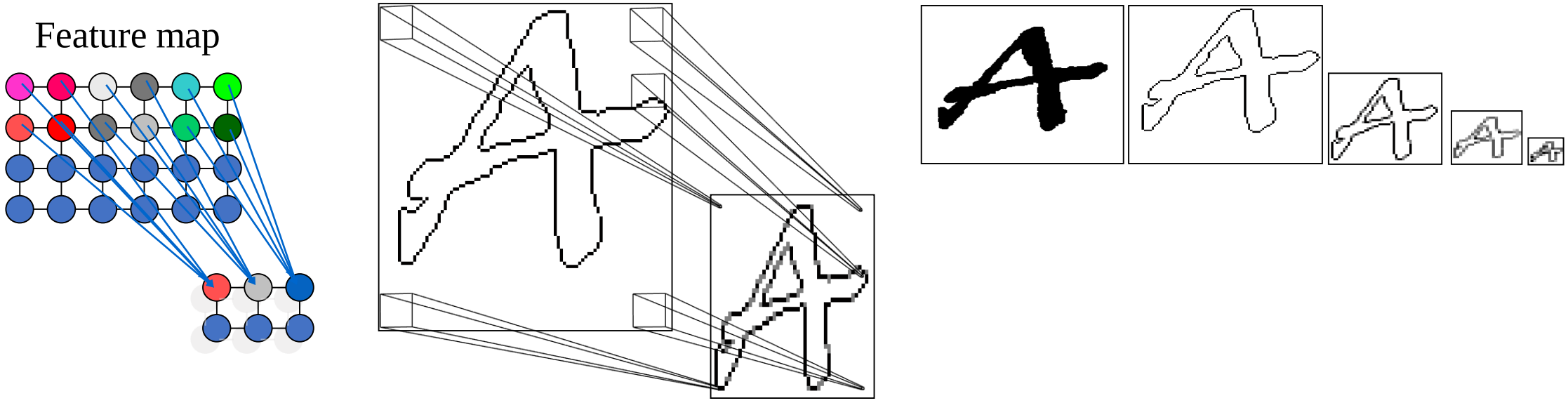
Max Pooling

6	5	8
9	7	8
7	9	7

$$\frac{n_x - f_x}{s} + 1 \, , \, \frac{n_y - f_y}{s} + 1$$

Importance of Subsampling Layer

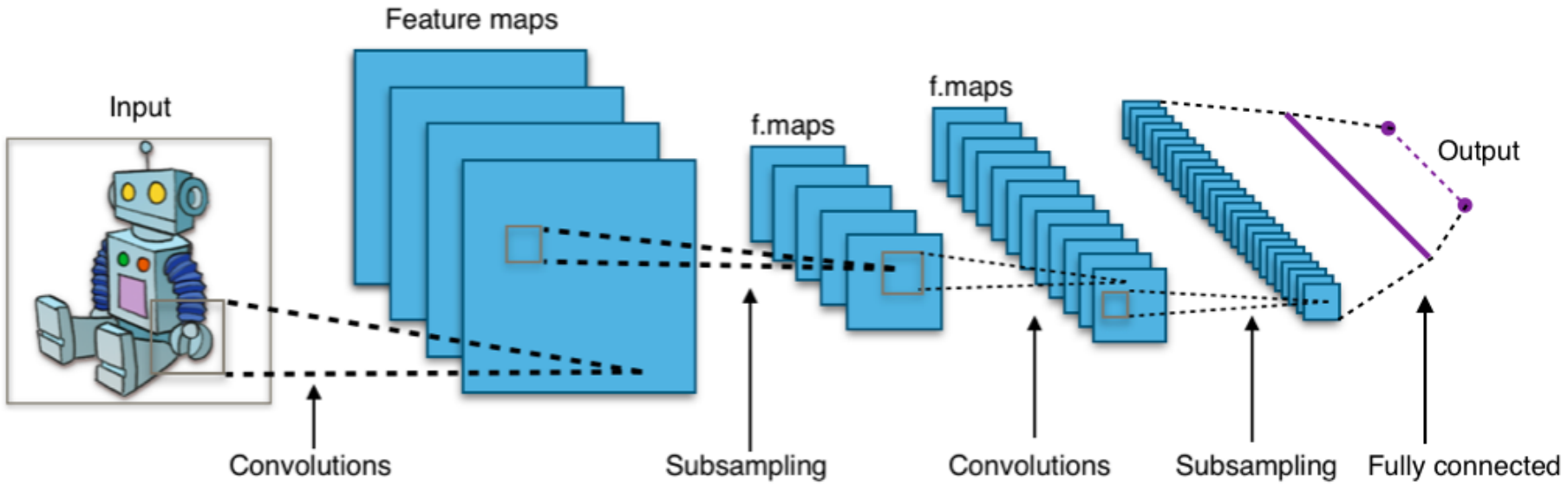
- Subsample the feature maps i.e. reduces the size of the feature maps.
  - Reduces the memory use and computer load
  - Reducing spatial resolution of feature map
  - Reducing effects of noise, shift or distortion invariance.
- The weight sharing is also applied in pooling layers resulting reducing the number of parameters.



Note: Pooling layer end up removing a lot of information, even a small pooling "kernel" of 2x2 with a stride of 2 will remove 75% of the input data.

12.2.6 Fully Connected Layer

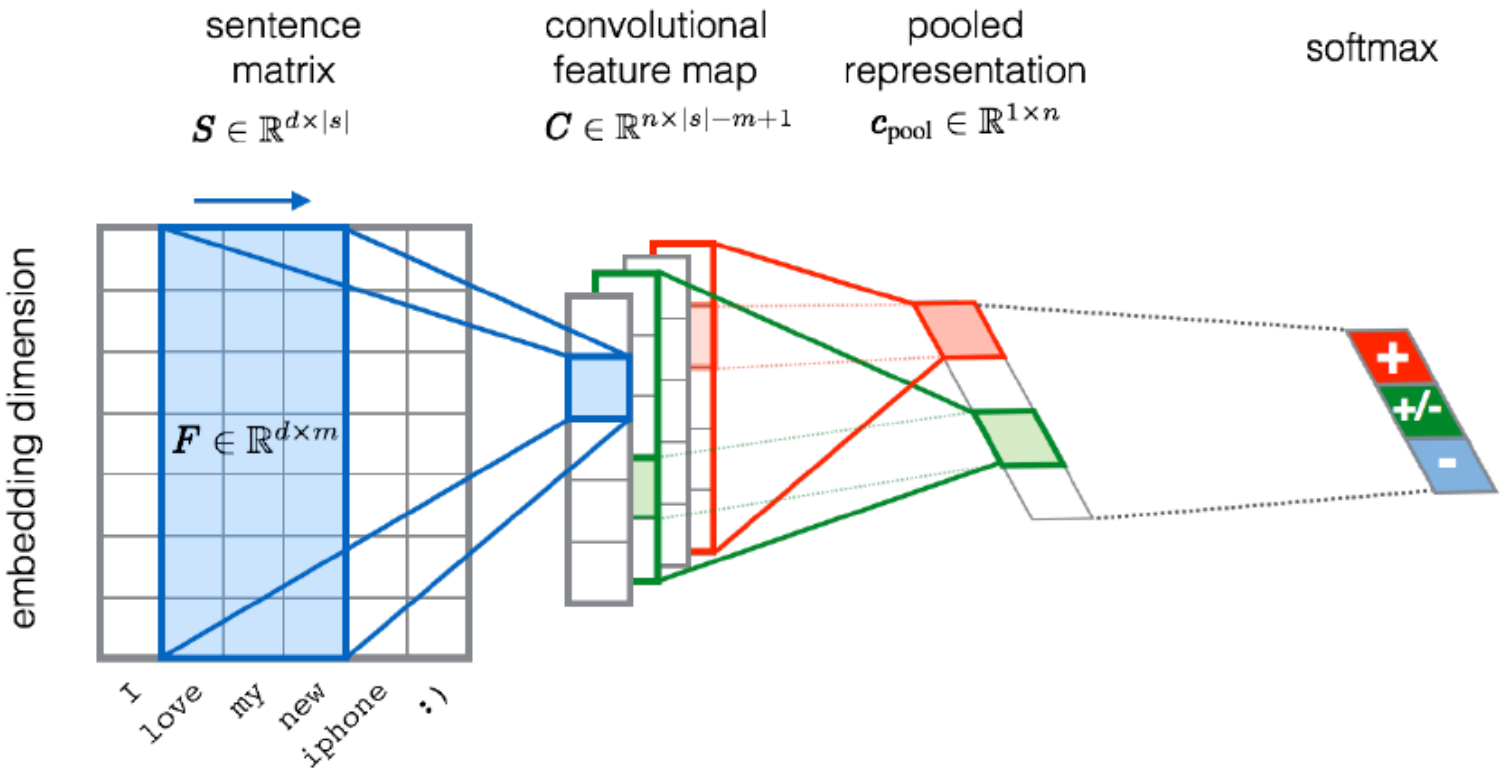
- Fully connected layers connect every neuron in one layer to every neuron in another layer.
- It is in principle the same as the traditional multi-layer perceptron neural network (MLP).
- The flattened matrix goes through a fully connected layer to classify the images.

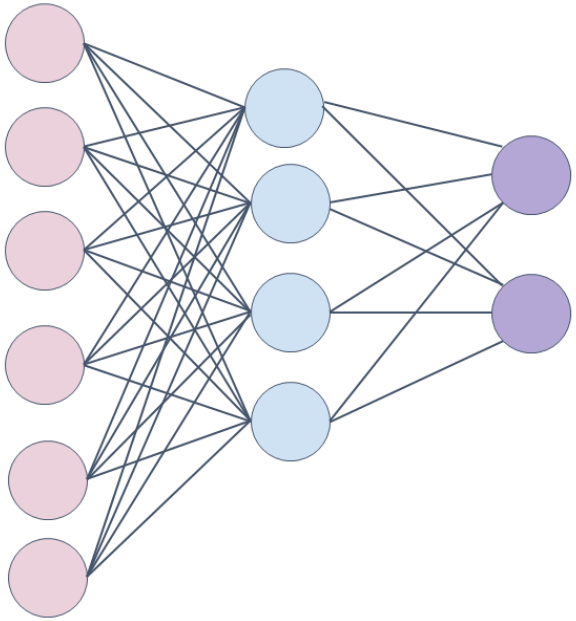


Main CNN idea for text: Compute vectors for n-grams and group them afterwards

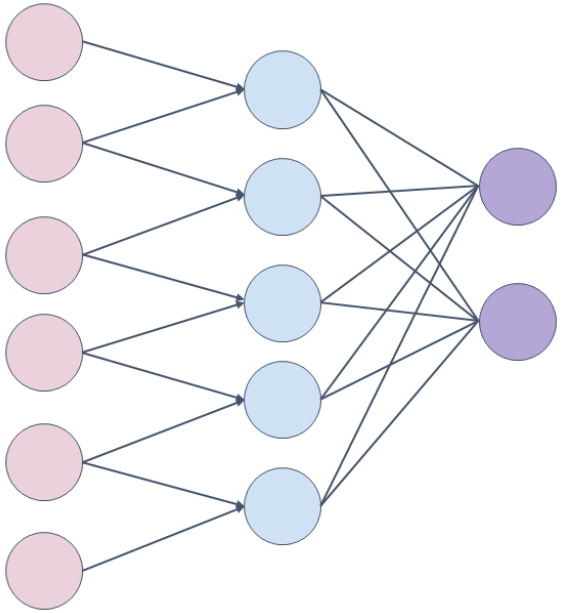
Example: "this takes too long"

Compute vectors for: This takes, takes too, too long, this takes too, takes too long, this takes too long

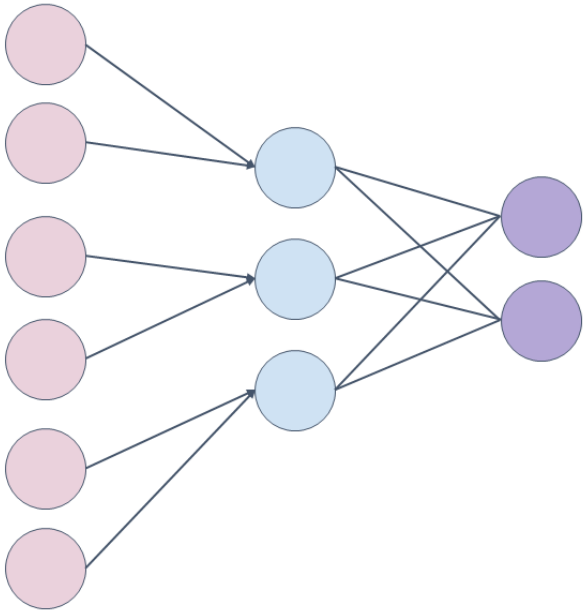




DNN



CNN
Filter Size: 2
Filters: 1
Stride: 1

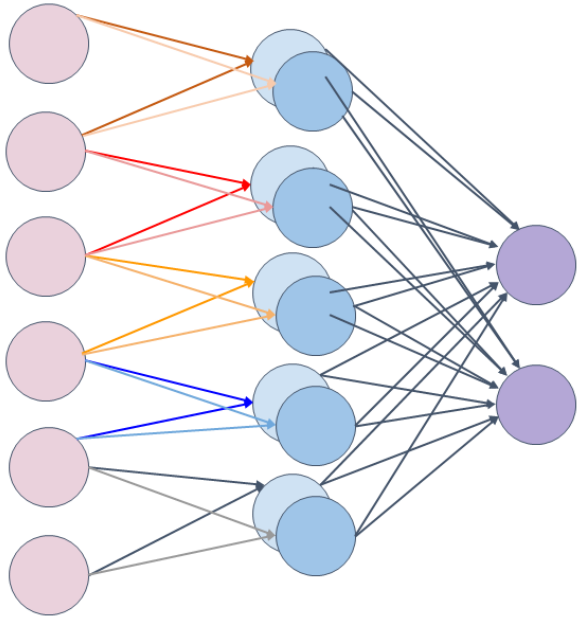


CNN
Filter Size: 2
Filters: 1
Stride: 2

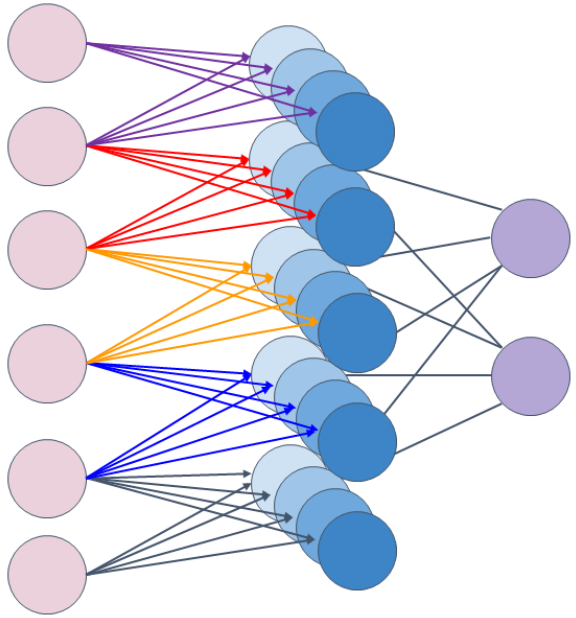
Note: The weights can be treated as filter for edge detection! Where, $y = w_1x_1 + w_2x_2$

Note: We can do pooling for size reduction.

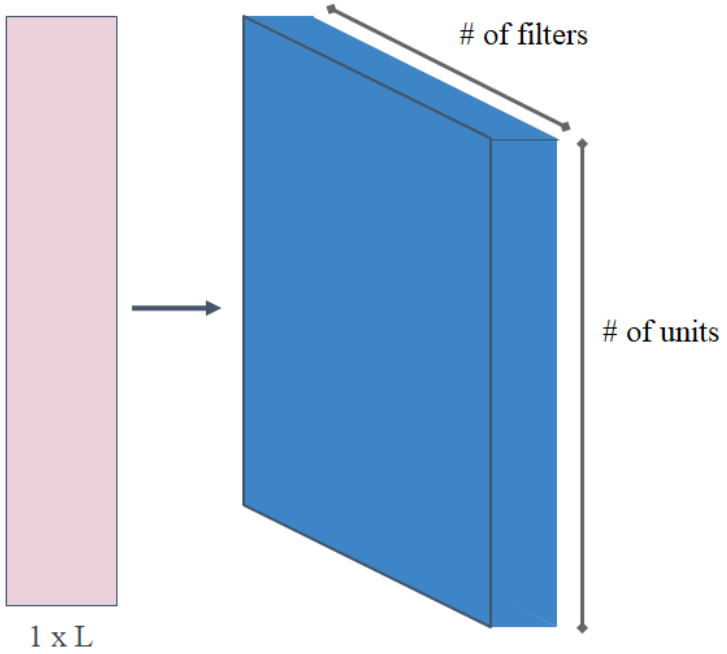
Note: add zero padding to include more edge pixels.



CNN
Filters: 2
Filter Size: 2
Stride: 1

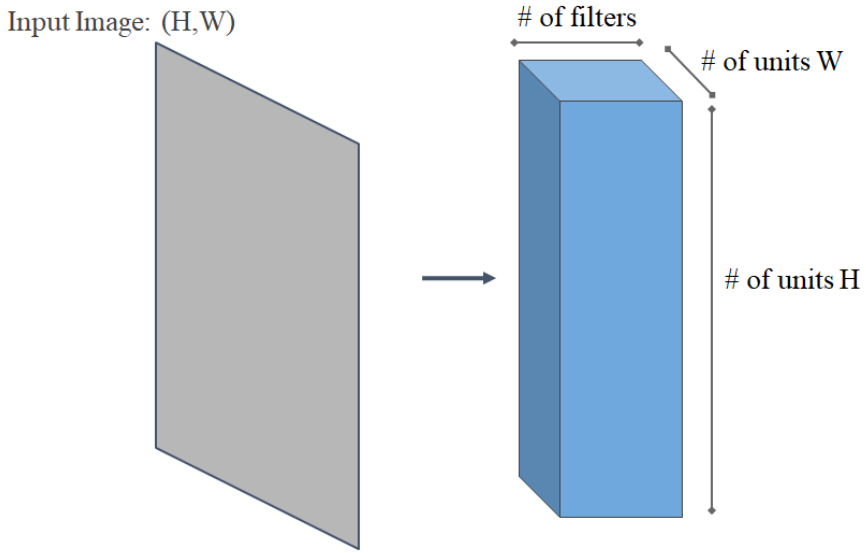


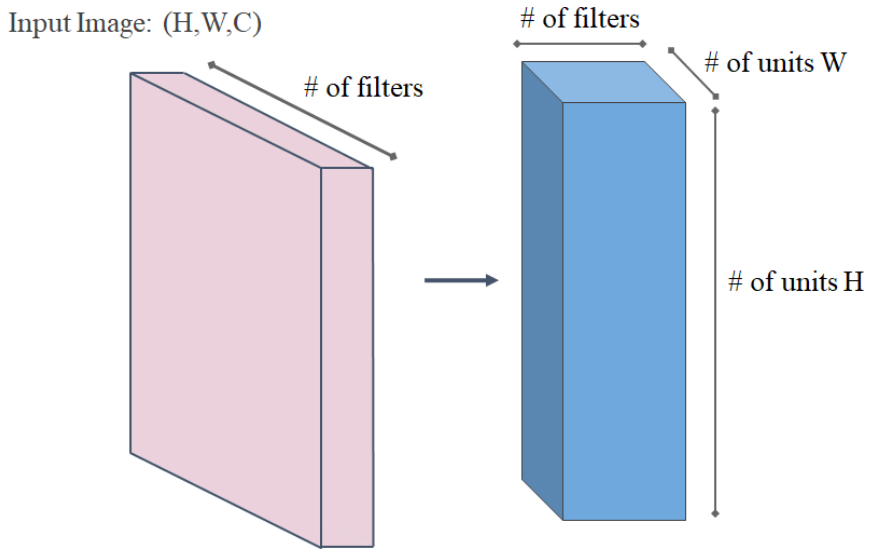
CNN
Filters: 4
Filter Size: 2
Stride: 2

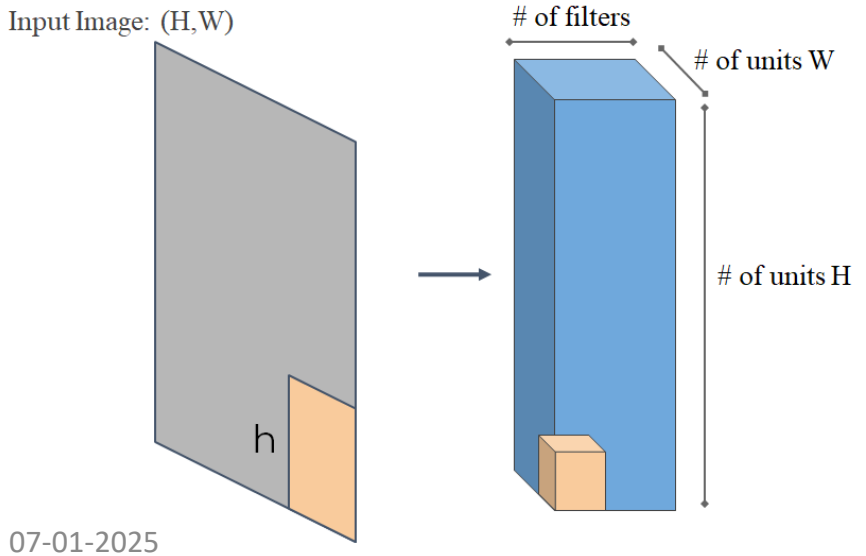


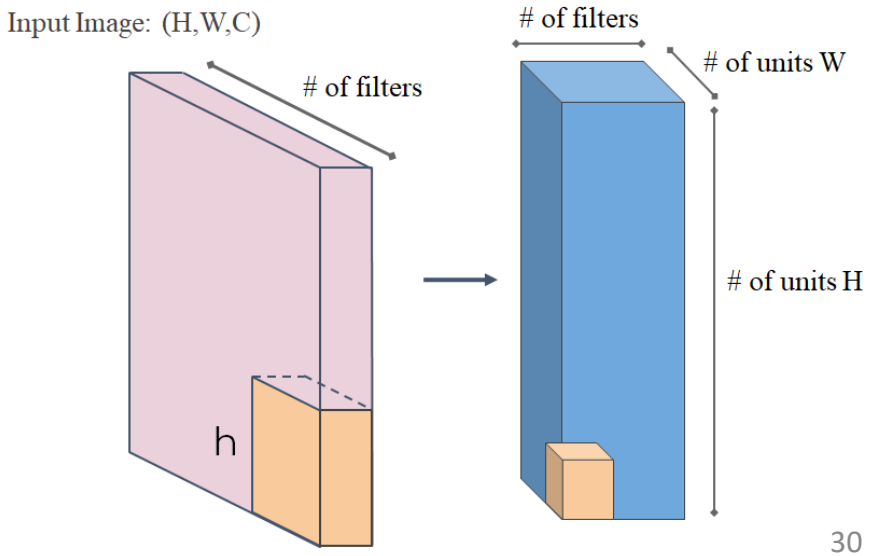
visualizing the neurons as blocks

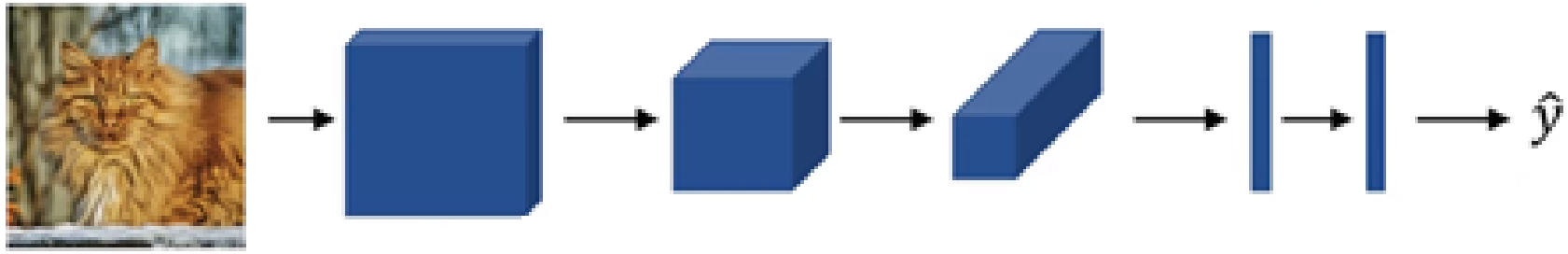
Note: Each filter is detecting a different feature











- Training set $(x^{(1)}, y^{(1)}) \dots (x^{(m)}, y^{(m)})$
- Cost Function, J = $\frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$
- Use Gradient Descent to optimize parameters to reduce cost J

Filter Size

- Common filter shapes found in literature vary greatly, and are usually chosen based on the dataset.
- The challenge is thus to find right filter shape, given a particular dataset without overfitting.

Number of Filters

- Since feature map size decreases with depth, lower layers can have fewer filters while higher layers can have more.
- The number of feature maps depends on the number of available examples and task complexity.

Pooling Size

- Typical values are 2×2.
- Very large input volumes may need 4×4 pooling in the lower layers.
- However, choosing larger shapes will dramatically reduce the dimension of the signal and may result in excess information loss.

12.2.11 Regularization Methods

Empirical

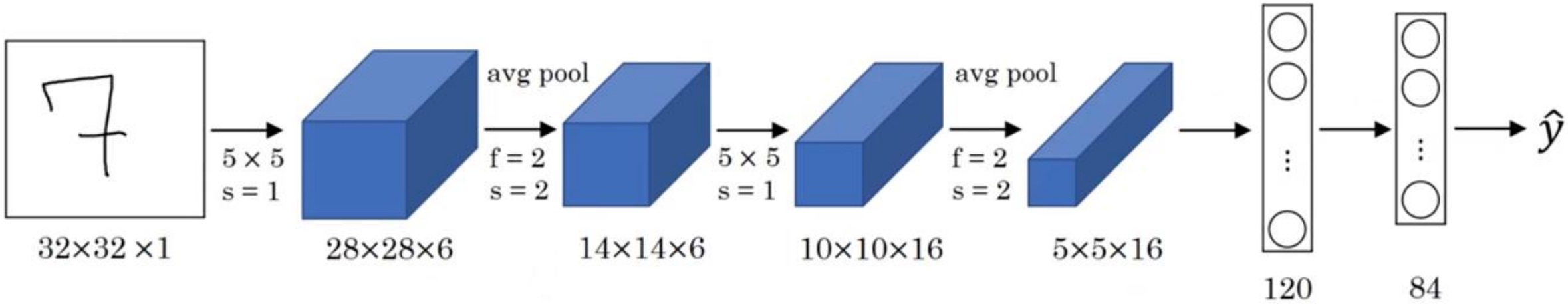
- Dropout
- DropConnect
- Hand Engineering

Explicit

- Early Stopping
- Number of Parameters
- Weight Decay

12.2.12 Different CNN Architectures

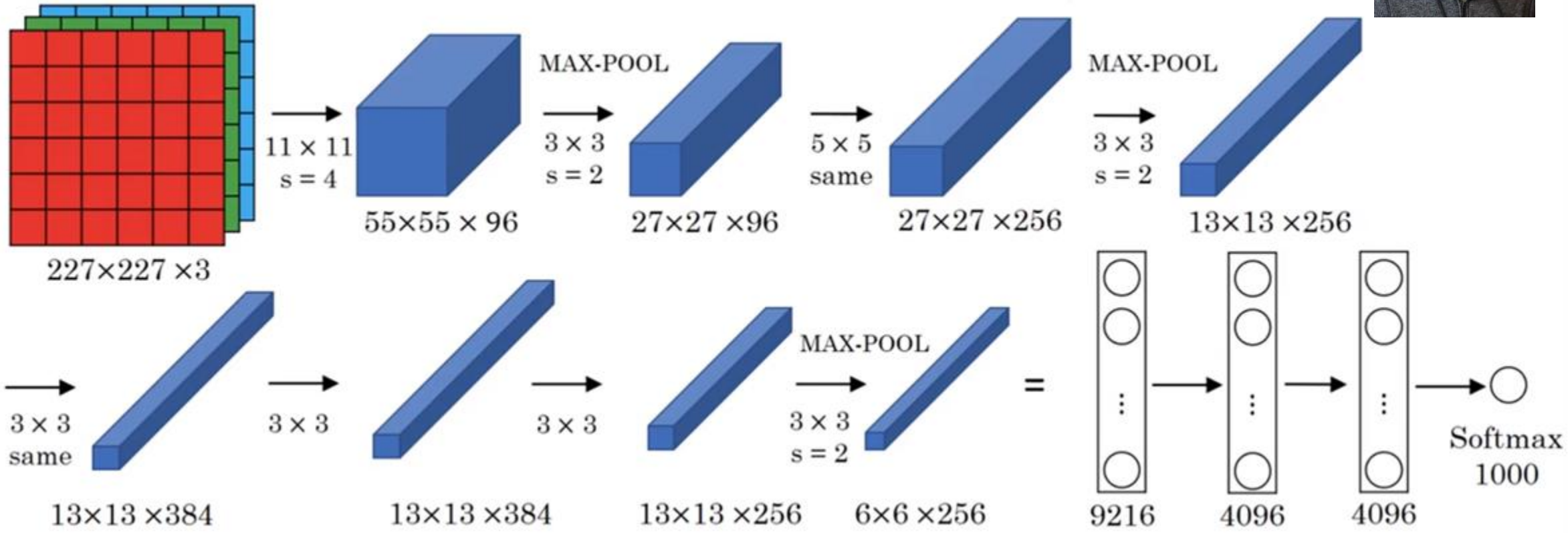
LeNet-5 [Yann LeCun]



- The LeNet-5 architecture was first used to classify the MNIST data set.

12.2.12 Different CNN Architectures

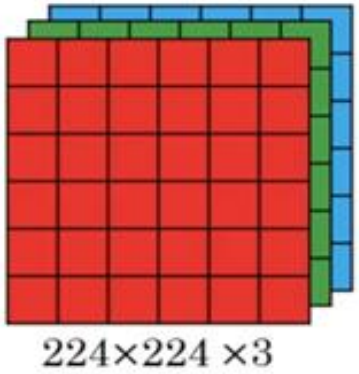
AlexNet-5 [Alex Krizhevsky et al]



VGG-16 [Karen Simonyan and Andrew Zisserman]

CONV = 3x3 filter, s=1

MAX-POOL = 2x2, s=2



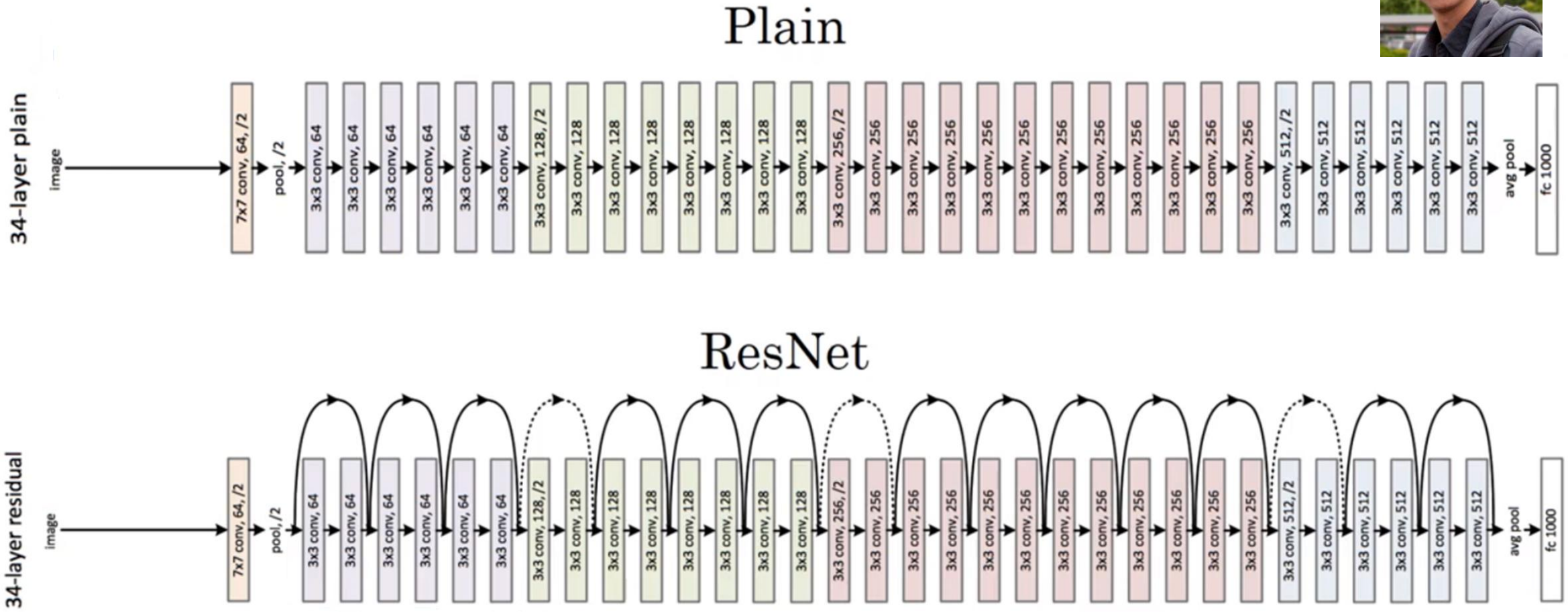
224×224 ×3

→ [CONV 64] ×2 → 224×224×64 → POOL → 112×112 ×64 → [CONV 128] ×2 → 112×112 ×128 → POOL → 56×56 ×128

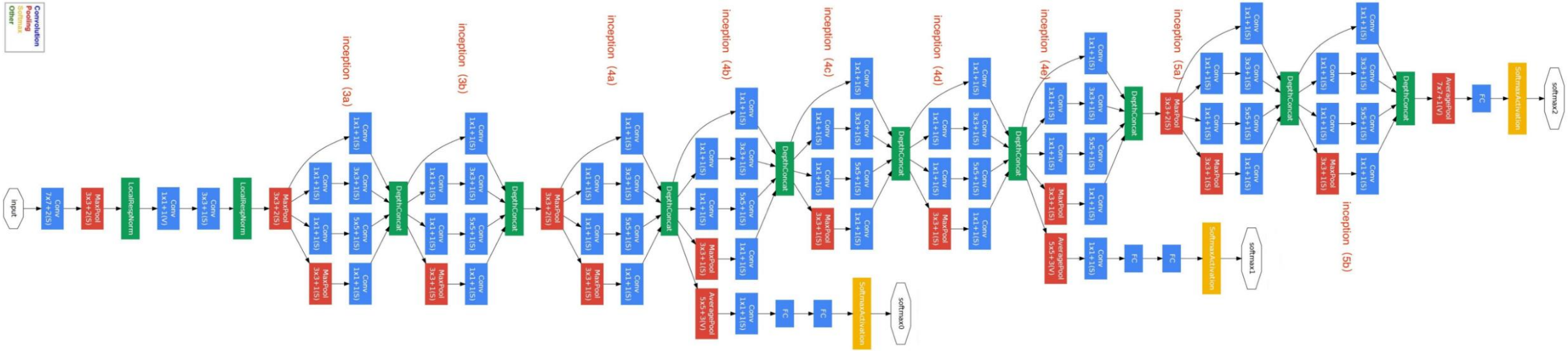
→ [CONV 256] ×3 → 56×56 ×256 → POOL → 28×28 ×256 → [CONV 512] × 3 → 28×28 ×512 → POOL → 14×14×512

→ [CONV 512] ×3 → 14×14 ×512 → POOL → 7×7×512 → FC 4096 → FC 4096 → Softmax 1000

ResNet [Kaiming He et al.]



GoogLeNet [Christian Szegedy at Google Research]



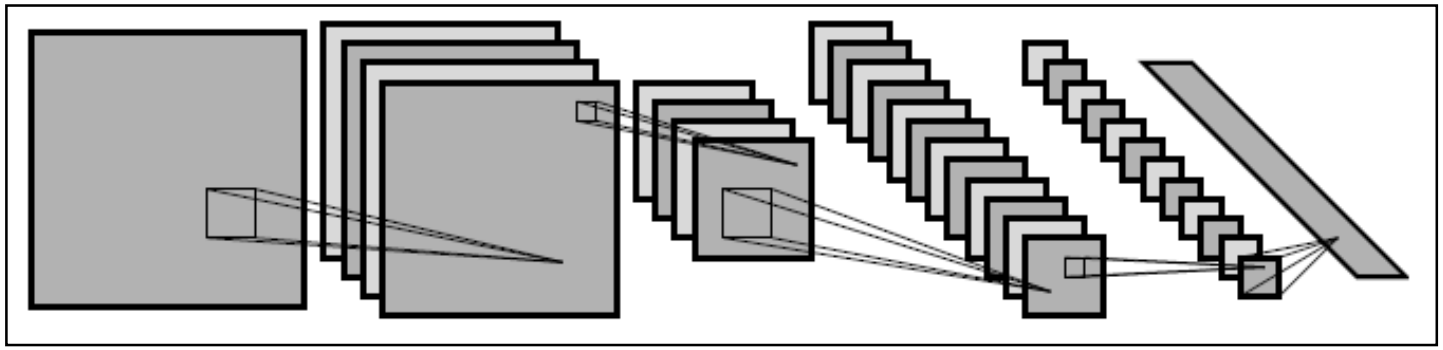
12.2.13 CNN Disadvantages

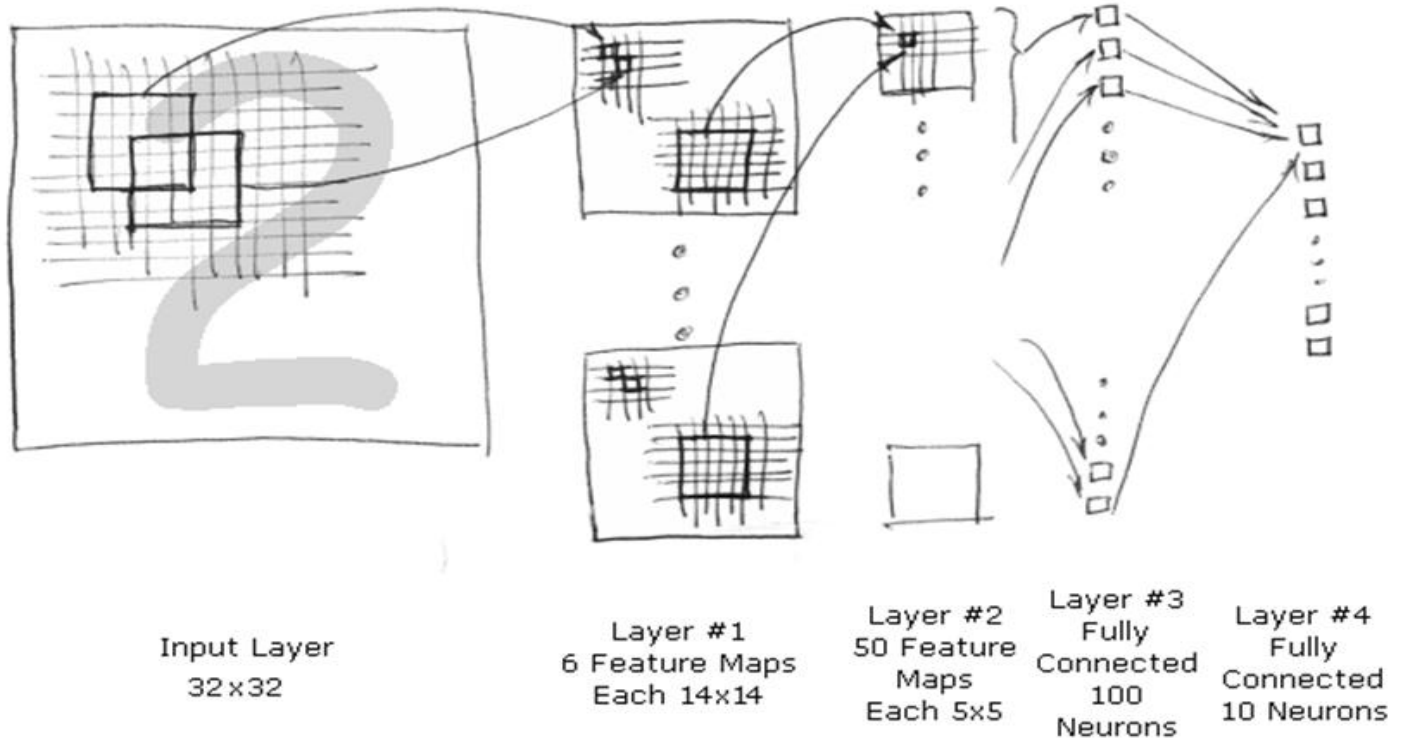
- From a memory and capacity standpoint, CNN is not much bigger than a regular two layer ANN.

- Convolution operations are computationally expensive and take about 67% of the runtime.
- CNN's are about 3X slower than their fully connected equivalents (size-wise).
- Convolution operation 4 nested loops (2 loops on input image & 2 loops on kernel)
- Small kernel size make the inner loops very inefficient as they frequently JMP.

- Cash unfriendly memory access because
  - Back-propagation require both row and column-wise access to the input and kernel image.
  - 2-D Images represented in a row-wise-serialized order.
  - Column-wise access to data can result in a high rate of cash misses in memory subsystem.

12.2.14 CNN Summary





12.2.15 Exercises

- The following repeated convolution and pooling layers are conducted in series with the mentioned filter size, pooling size and strides. Fill up the vacant slots and construct the corresponding architecture block in diagram for the following CNN.

	Activation shape	Activation Size	# parameters
Input:	(32,32,3)	3,072	0
CONV1 (f=5, s=1)	(28,28,8)	6,272	208
POOL1	(14,14,8)	1,568	0
CONV2 (f=5, s=1)	(10,10,16)	1,600	416
POOL2	(5,5,16)	400	0
FC3	(120,1)	120	48,001
FC4	(84,1)	84	10,081
Softmax	(10,1)	10	841

The Topics Covered in This Section

12.3.1 Introduction
12.3.2 Regular ANN vs. RNN
12.3.3 RNN Topology
12.3.4 Training
12.3.5 Loss Function
12.3.6 Backpropagation Through Time
12.3.7 RNN Issues
12.3.8 Bidirectional RNN
12.3.9 LSTM
12.3.10 LSTM Architecture
12.3.11 LSTM Variants
12.3.12 LSTM Training
12.3.13 LSTM Advantages
12.3.14 Gated Recurrent Unit
12.3.15 RNN Summary
12.3.16 Exercises

12.3.1 Introduction

- A special kind of feedback ANN designed to handle sequential data.
- Introduced by John Hopfield [1982, Hopfield Network, a special RNN]

Applications

- In Sequential Data to perform Classification, sequence prediction etc.

  like, sentences → sequences of words

  words → sequences of characters

  speech → sequences of phonemes

  videos → sequences of images

Algorithms to train RNN

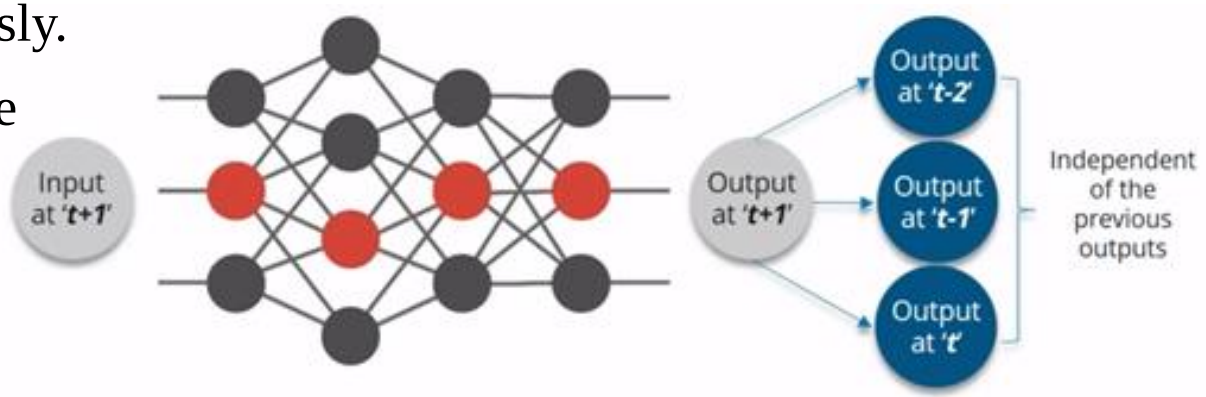
- Backpropagation through time (using Gradient Descent)

12.3.2 Regular ANN vs. RNN

Why not regular ANN or CNN?

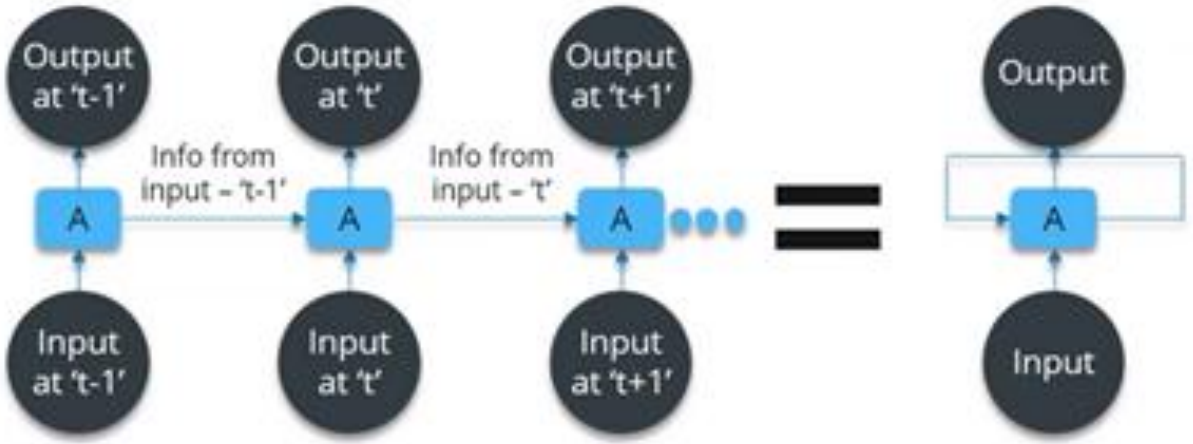
- Regular ANN is good at dealing numerical data. ***For example*** for weather prediction.
- CNN is good at dealing image data. ***For example*** for a image predicting or classifying the object.
- RNN is for sequential data. ***For example*** video, language, speech.

- ANNs and CNNs have
  - Fixed size inputs.
  - The whole input available simultaneously.
- But in video, language, speech data we have
  - Variable sized input
  - Sequential time-series information
- So Regular NN or CNN unable to handle.

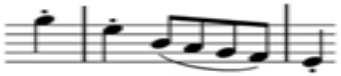


12.3.2 Regular ANN vs. RNN

How RNN handles?

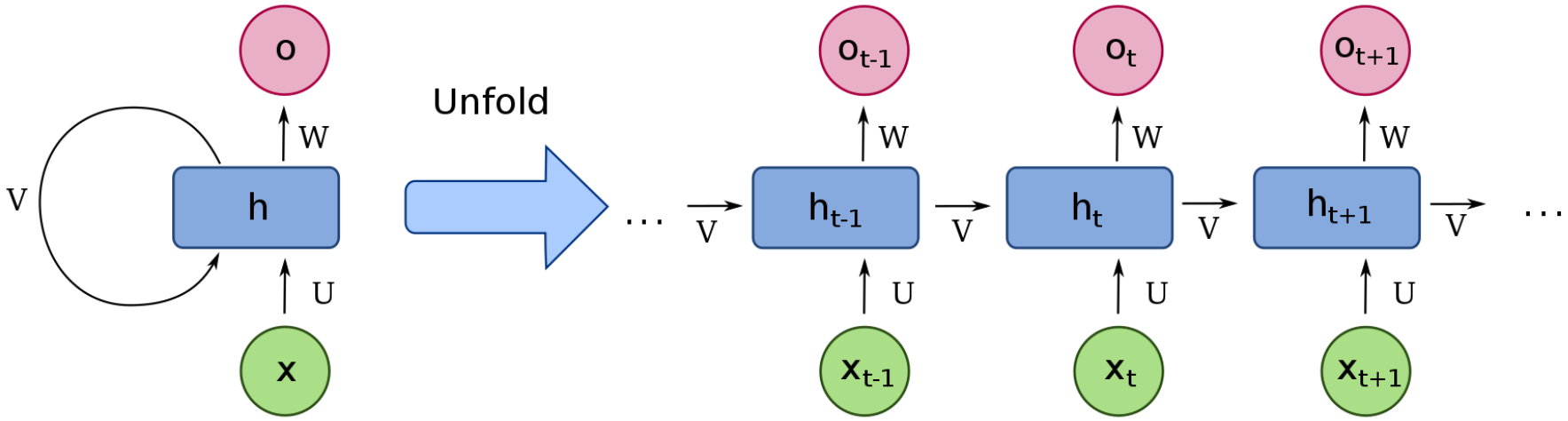


Examples of Sequence Data

- Video Activity Recognition  → A group young boy playing
- Machine Translation Assam is a beautiful state → অসম এখন ধুনীয়া ৰাজ্য
- Speech Recognition  → The class going on
- Music Generation ∅ → 
- DNA Sequence Analysis AGCCCCTGTGAGGAACTAG → AGCCCCTGTGAGGAACTAG

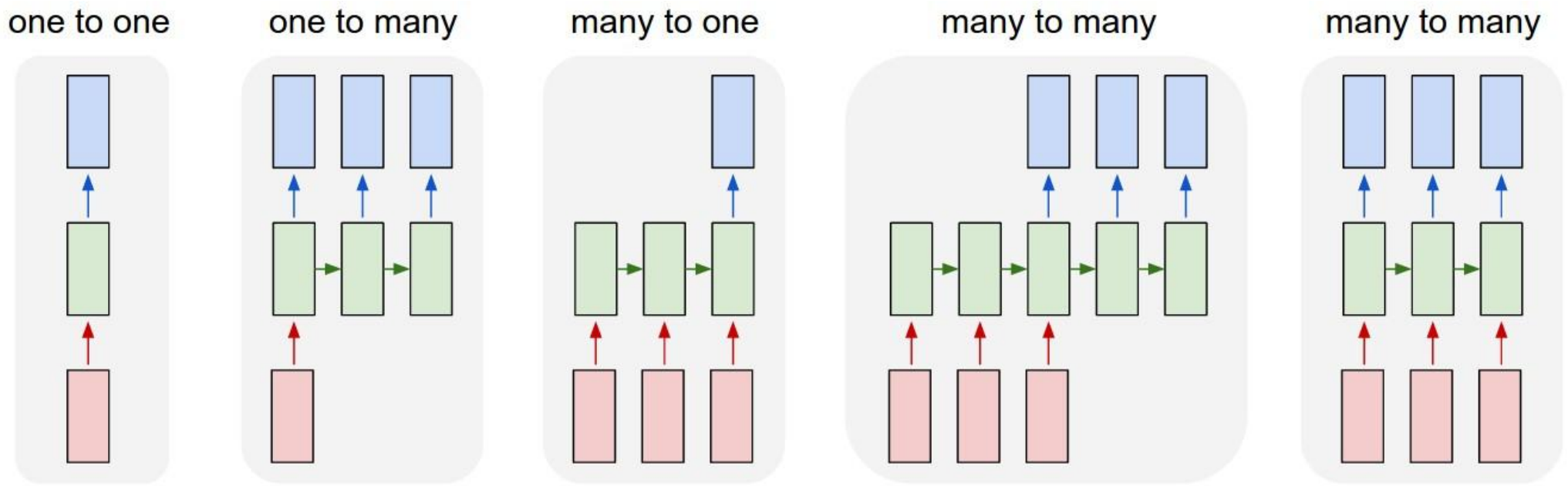
12.3.3 RNN Topology

- One hidden vector depends on previous hidden vector and input vector at that particular time.
- Cells that are a function of inputs from previous time steps are also known as *memory cells*.



Size of h may be any number of neurons

12.3.3 RNN Topology

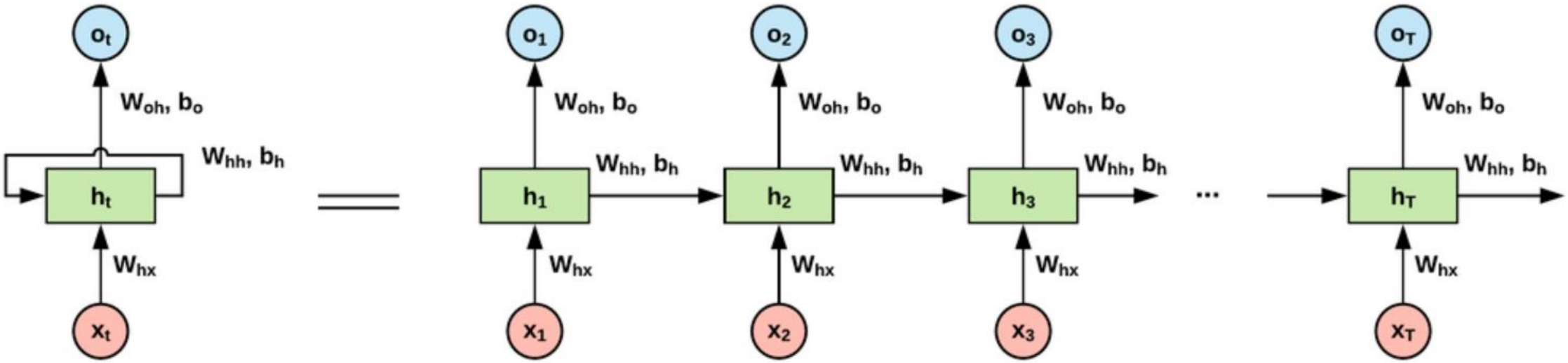


One to many: Vector to Sequence
(image →seq of words)
(Image captioning)

Many to one: Sequence to Vector
(seq of words → +ve or -ve)
Sentiment Analysis

Many to many: Sequence to Sequence
(seq of words/audio → seq of words)
(Language translation or speech recognition)

12.3.4 Training

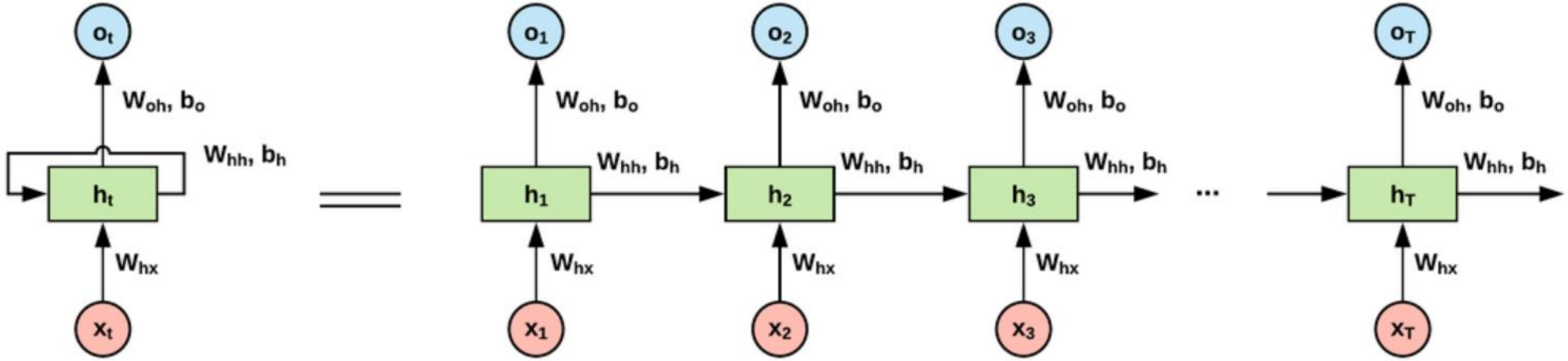


$$h_t = g\,(w_{xh}x_t + w_{hh}h_{t-1} + b_h)$$

$$o_t = g\,(w_{oh}h_t + b_o)$$

12.3.5 Loss Function

- Calculating Loss by loss function
- The local loss function we can use either/or –
  - Cross Entropy: Whenever the problem is a classification problem.
  - Least Squares: whenever the problem is a regression problem.
- So the total loss is actually summation of all the intermediate losses through the layers.
- Note: It also depends whether we are take out outputs at all intermediate layers also or not.

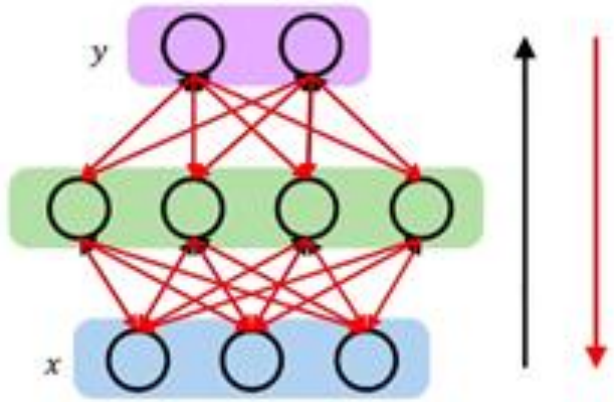


$$L = \sum_{i=1}^{t} L_t$$ L is total number of layers

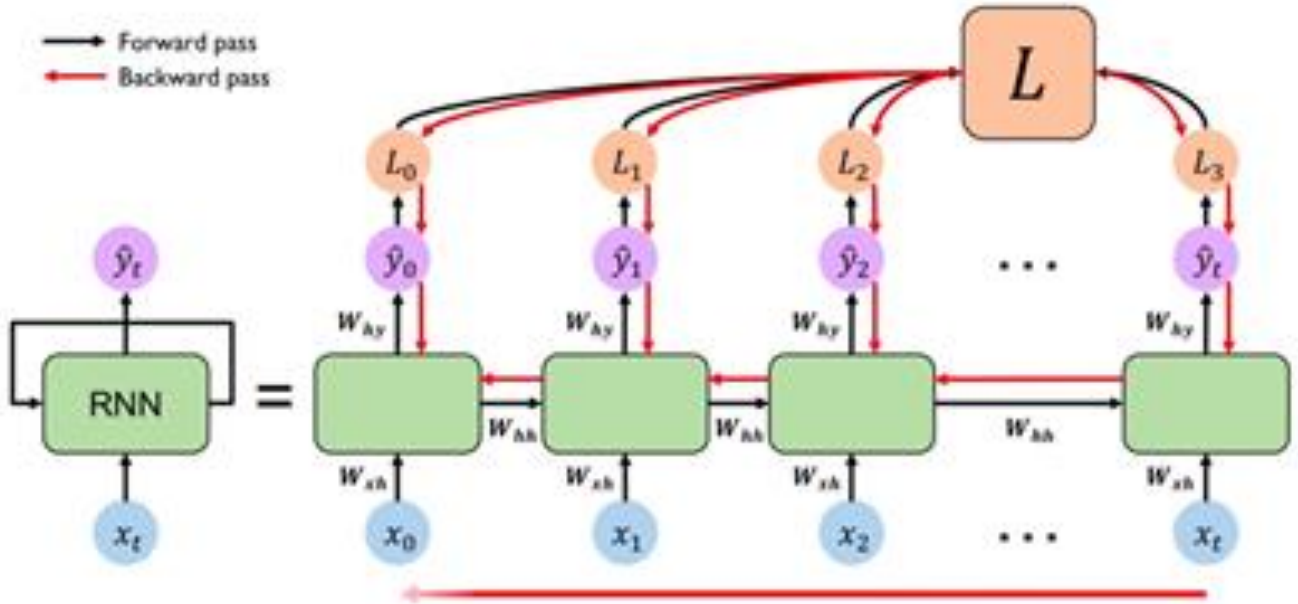
12.3.6 Backpropagation Through Time

Recall: Backpropagation

- Take the derivative (gradient) of the loss w.r.t. each parameter.
- Shift parameters in order to minimize loss.

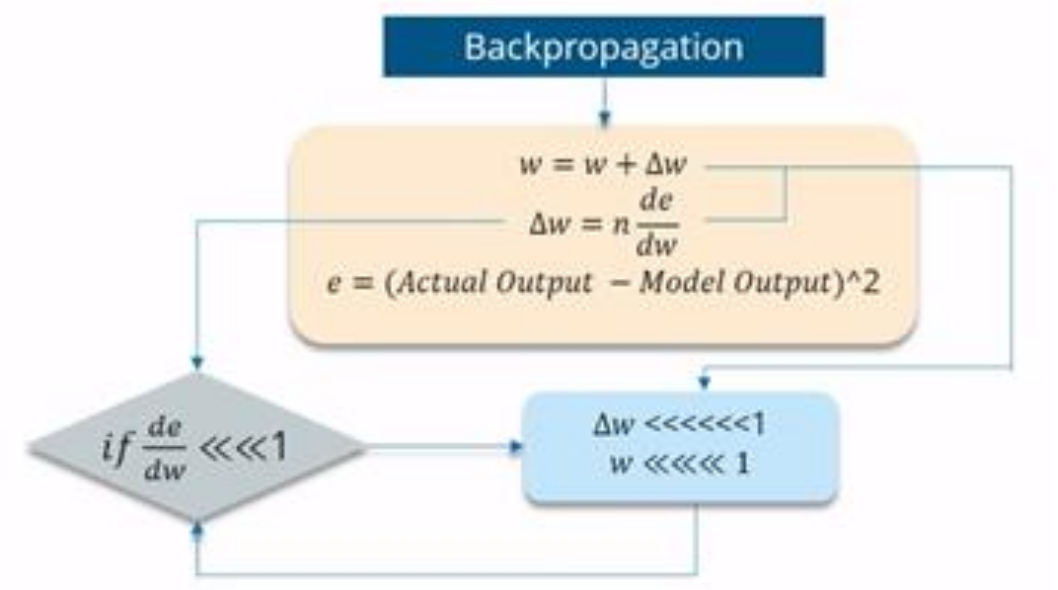


Backpropagation through time (BPTT) (as it is applied for every time stamp)



Vanishing Gradient

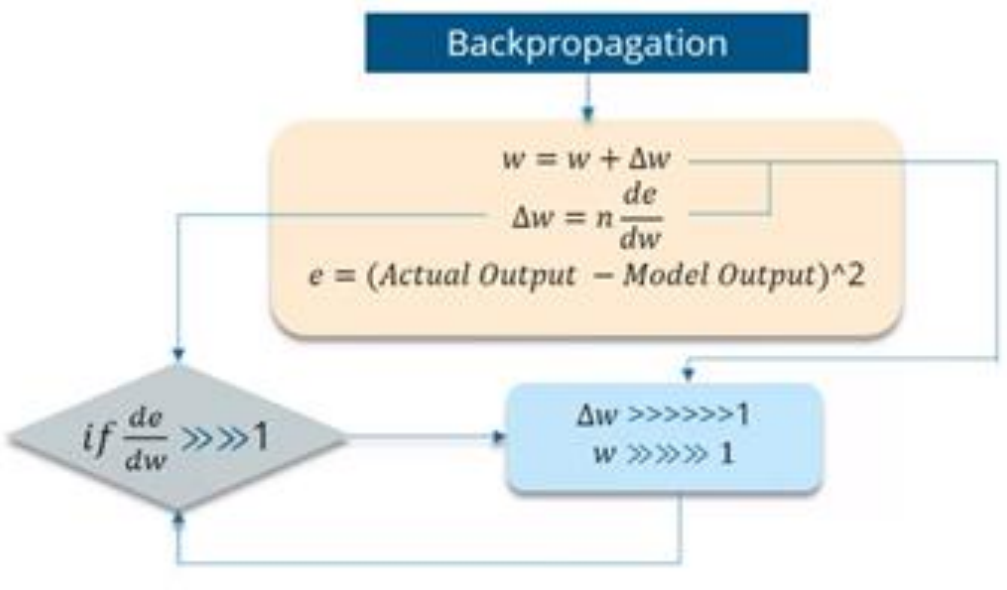
Errors due to further back time steps have smaller and smaller gradients



Solution:

- Activation Function - ReLu
- Architecture (Gated cell to track what information is passed through) - LSTM, GRU

Exploding Gradient



Solution:

- Gradient clipping to scale big gradient

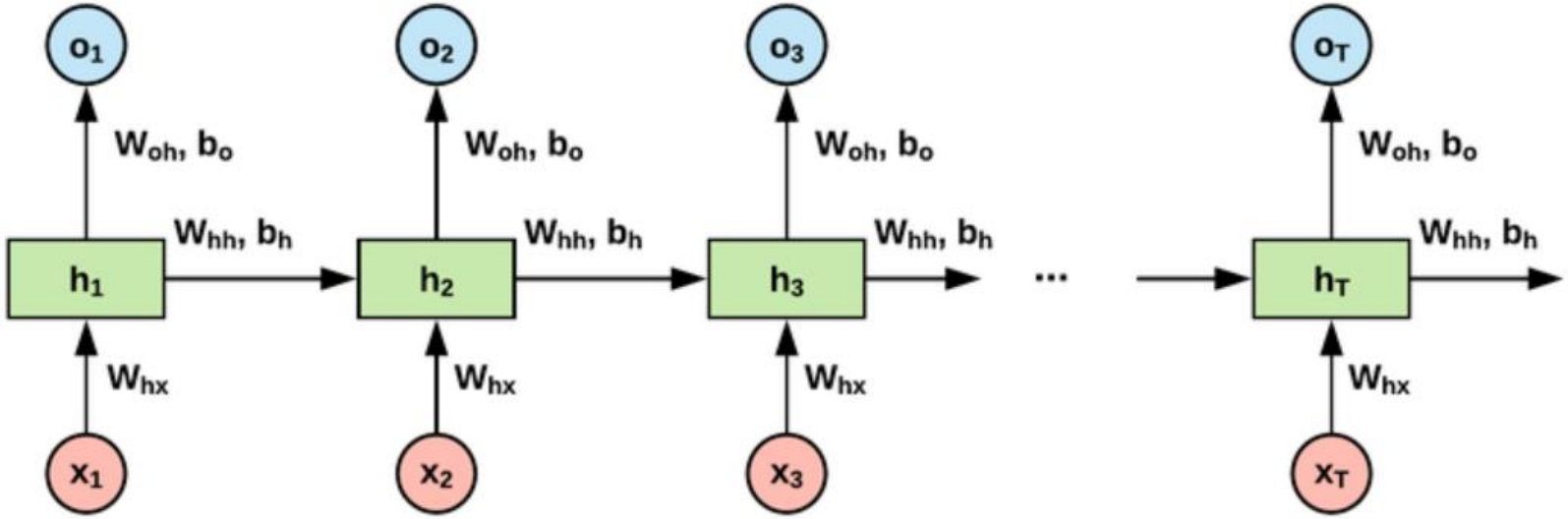
12.3.7 RNN Issues

- After awhile the network will begin to “forget” the first inputs, as information is lost at each step going through the RNN.
- We need some sort of “long-term memory” for our networks.

Example:

The cat, which actually ate ……., was full.

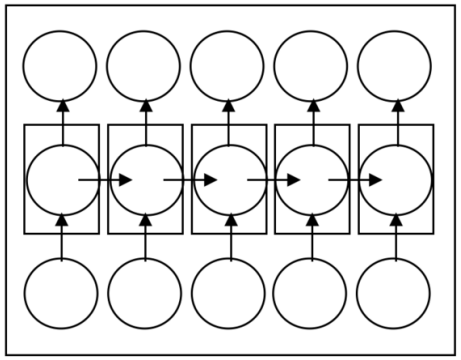
The cat, which actually ate ……., were full.



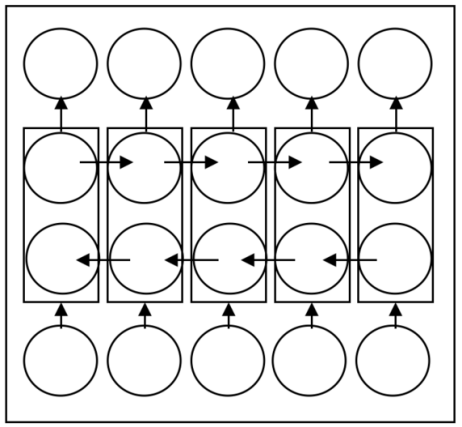
12.3.8 Bidirectional RNN

- Output may not only depend on the **previous** elements in the sequence, but also **future** elements.
- BRNN uses a finite sequence to predict or label each element of the sequence based on the element's past and future contexts (left and right context).
- This is done by concatenating the outputs of two RNNs, one processing the sequence from left to right, the other one from right to left.
- Especially useful when combined with LSTM RNNs.

Unidirectional RNN



Bidirectional RNN



$$\vec{h}_t = \sigma(\vec{W}^{(hh)}\vec{h}_{t-1} + \vec{W}^{(hx)}x_t)$$

$$\overleftarrow{h}_t = \sigma(\overleftarrow{W}^{(hh)}\overleftarrow{h}_{t+1} + \overleftarrow{W}^{(hx)}x_t)$$

$$y_t = f(\,[\vec{h}_t; \overleftarrow{h}_t\,]\,)$$

past and future around a single token

- Two RNNs stacked on top of each other
- output is computed based on the hidden state of both RNNs $[\vec{h}_t; \overleftarrow{h}_t\,]$

12.3.9 LSTM

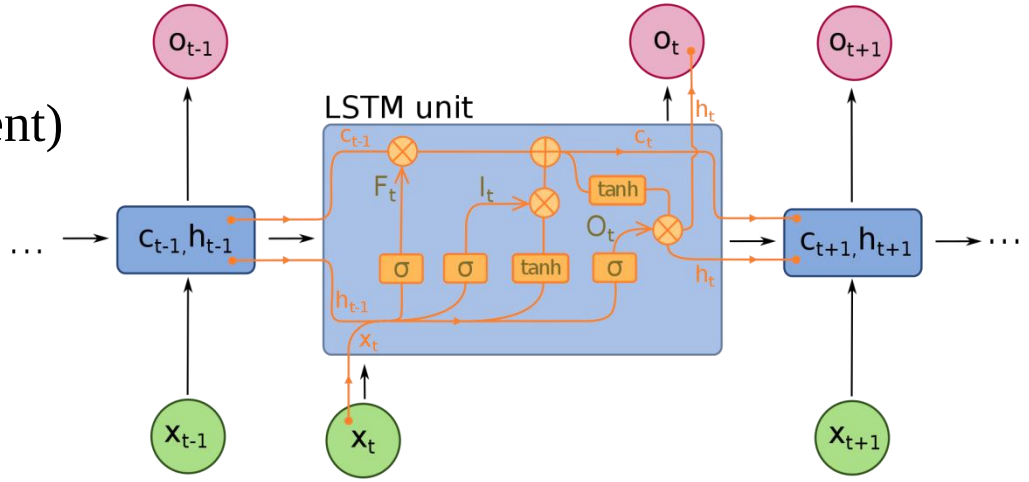

- Introduced by Sepp Hochreiter and Juergen Schmidhuber [1997].
- Special type of RNN to overcome the challenges of training RNN.
- Training RNN for long term dependencies is difficult.
- Relative insensitivity to gap length is an advantage of LSTM over RNNs, HMV and other.

Applications

- In Sequential Data to perform classification, sequence prediction, translation etc. specially for NLP.

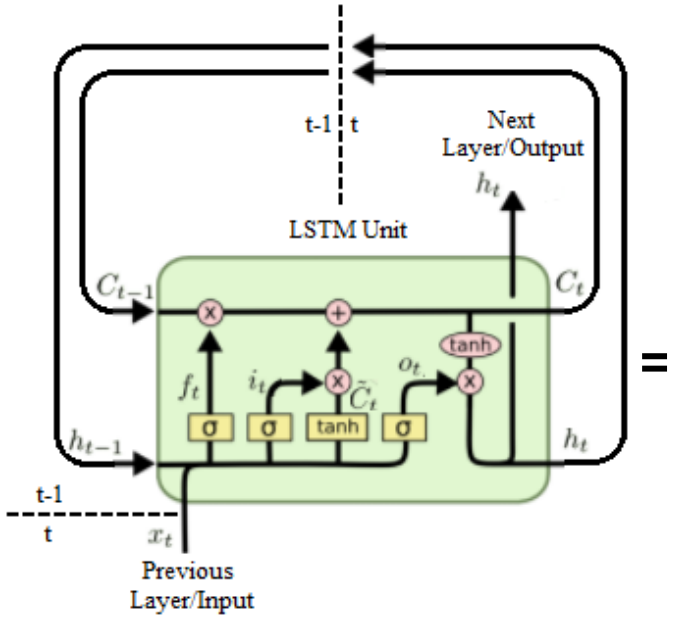
Algorithms to train RNN

- Backpropagation through time (using Gradient Descent)

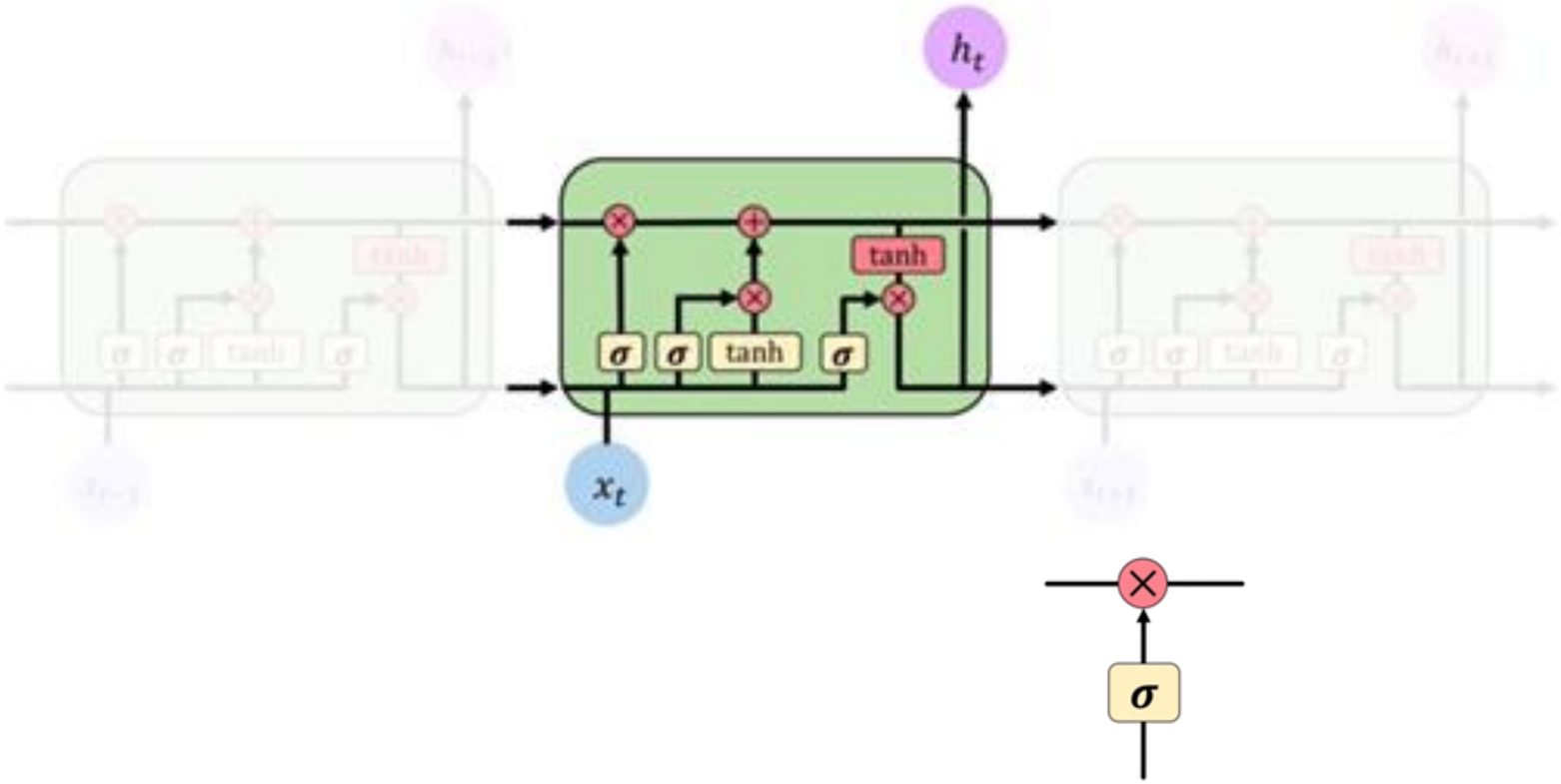


12.3.9 LSTM Architecture

- A LSTM unit is composed of a cell, an **input gate**, an **output gate** and a **forget gate**.
- The cell remembers values over arbitrary time intervals and the three *gates* regulate the flow of information into and out of the cell.



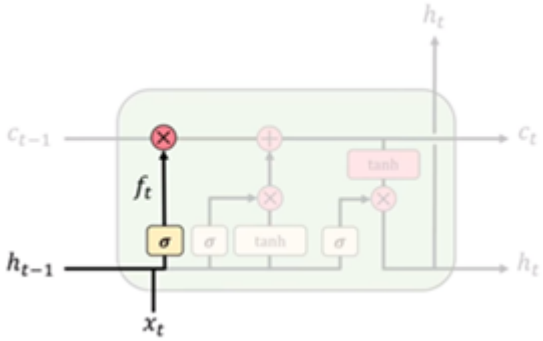
=



12.3.10 LSTM Architecture

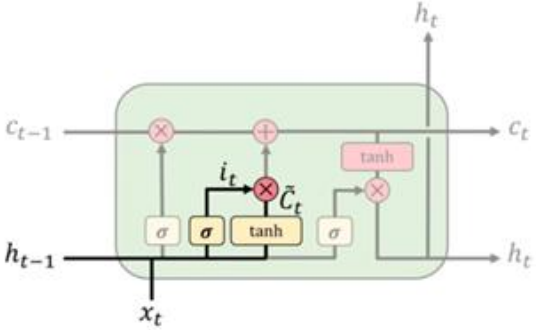
How it Works?

1. **Forget:** Forget irrelevant parts of the previous state
2. **New Input:** Identify new information to be stored
3. **Update:** Selectively update cell state value
4. **Output:** Output certain parts of the cell state



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

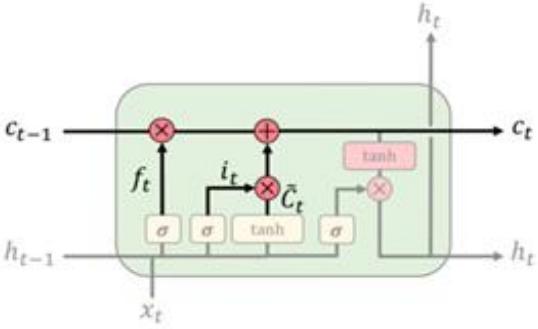
- Use previous cell output & input
- Sigmoid value: 0 for completely forget and 1 for completely keep



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

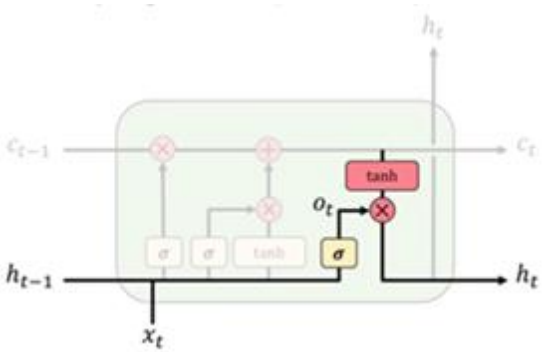
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

- Sigmoid layer: decide what value to update
- Tanh layer: generate new vector that can be added



$$C_t = f_t \circ C_{t-1} + i_t \circ \tilde{C}_t$$

- Apply forget operation to previous internal cell state
- Add new values scaled by how much need to update

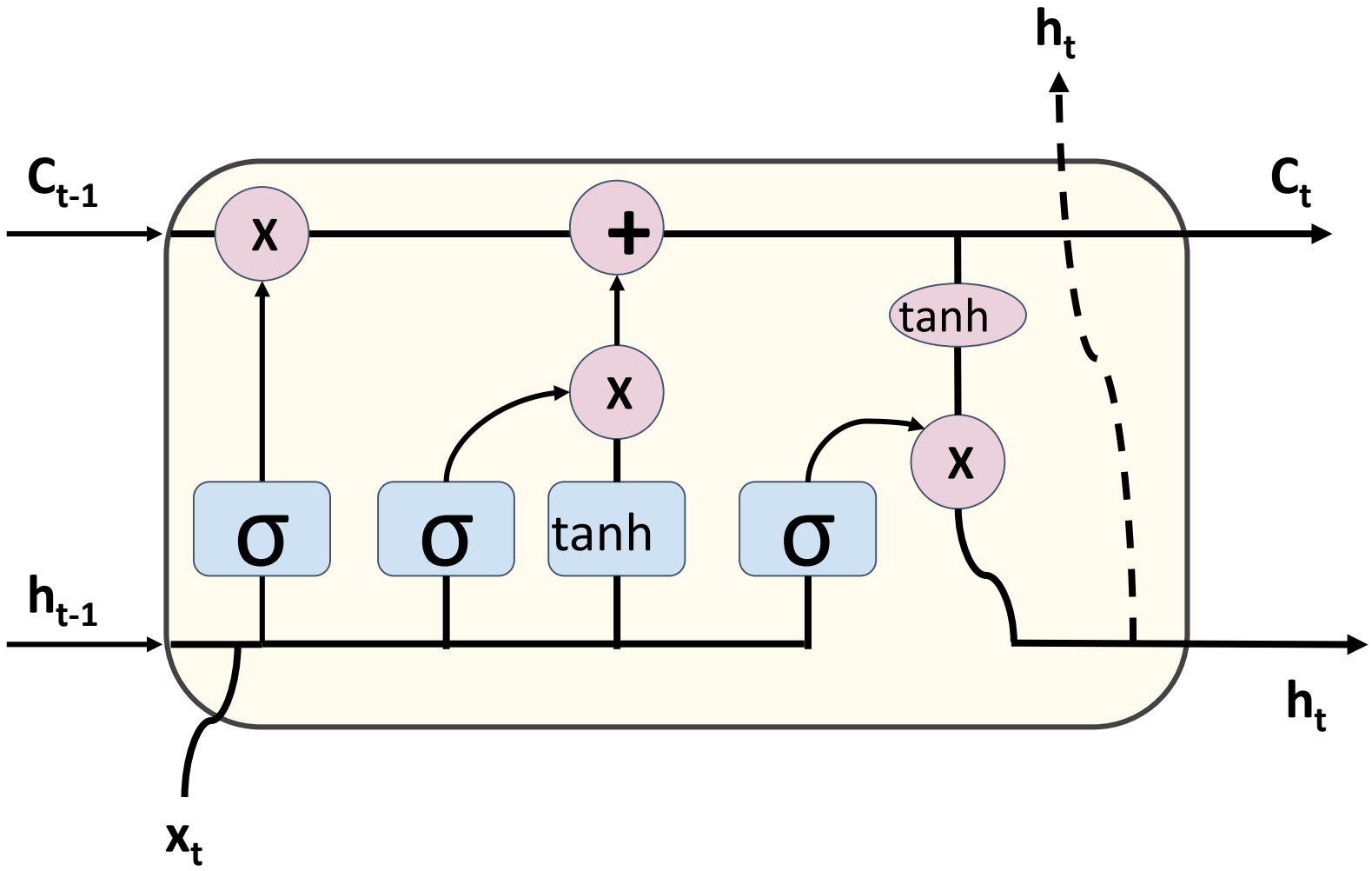


$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$

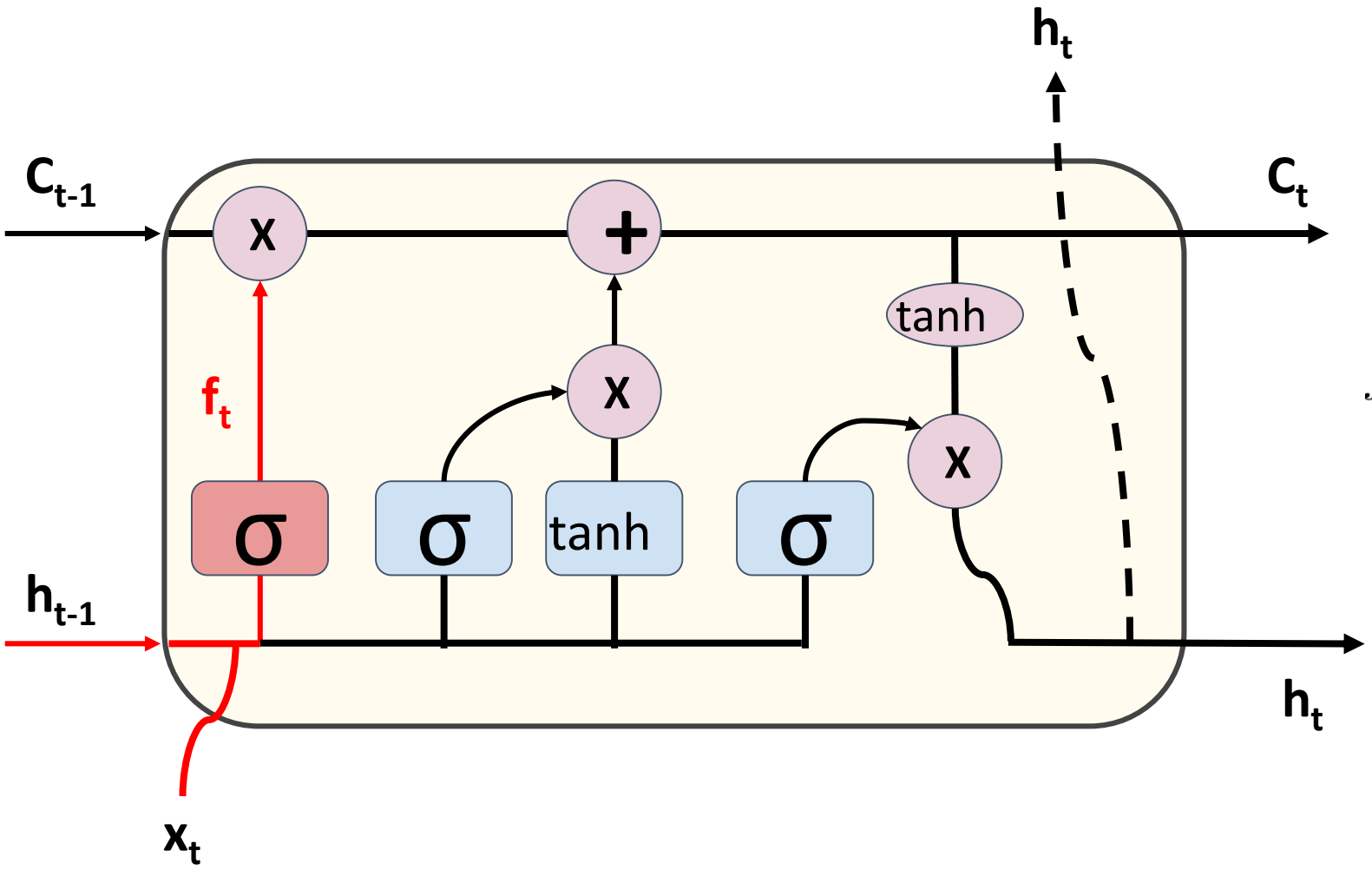
$$h_t = o_t \circ \tanh\left(C_t\right)$$

- Sigmoid layer: decides what parts of state to output
- Tanh layer: squash values between -1 and 1
- h_t: output of cell state

12.3.10 LSTM Architecture

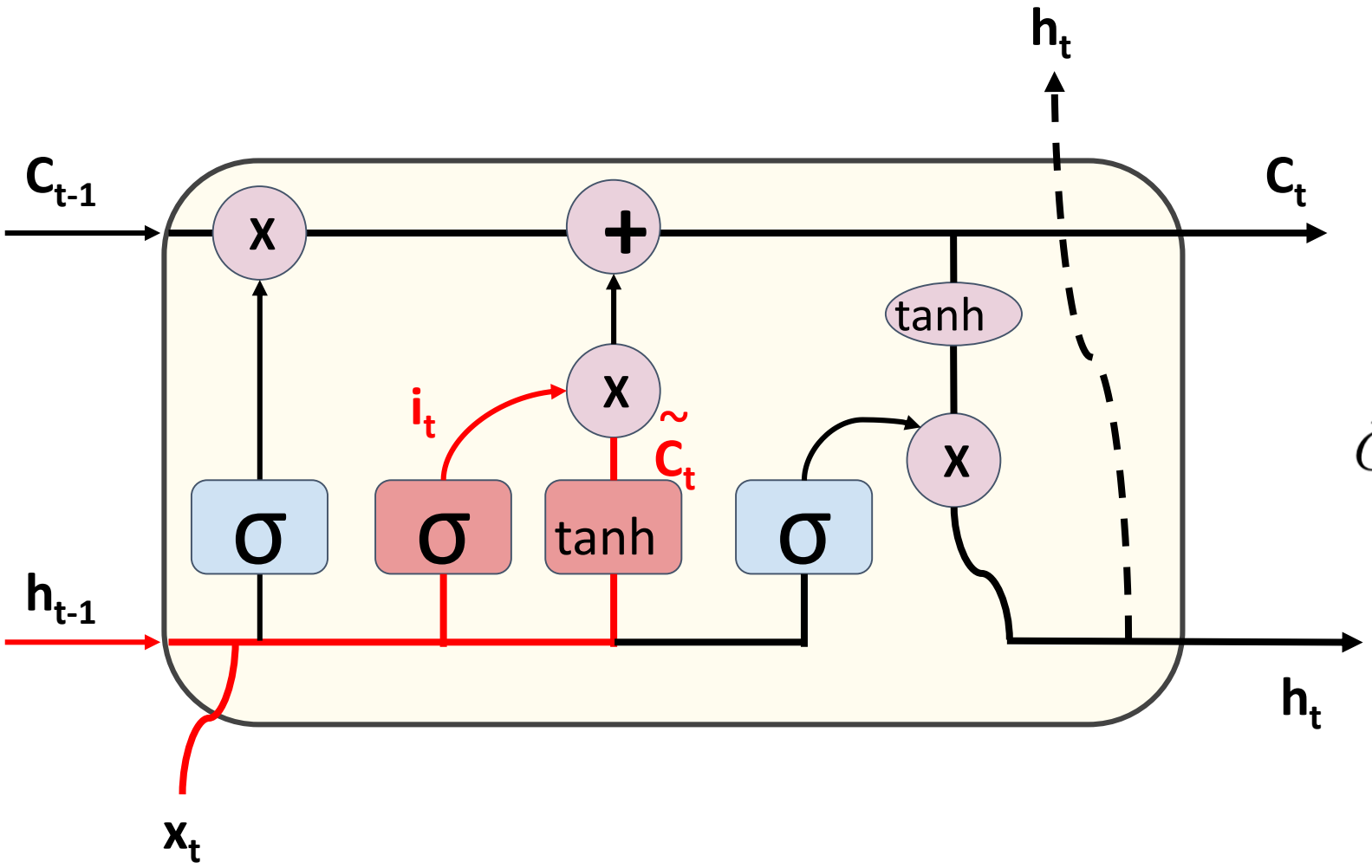


12.3.10 LSTM Architecture



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

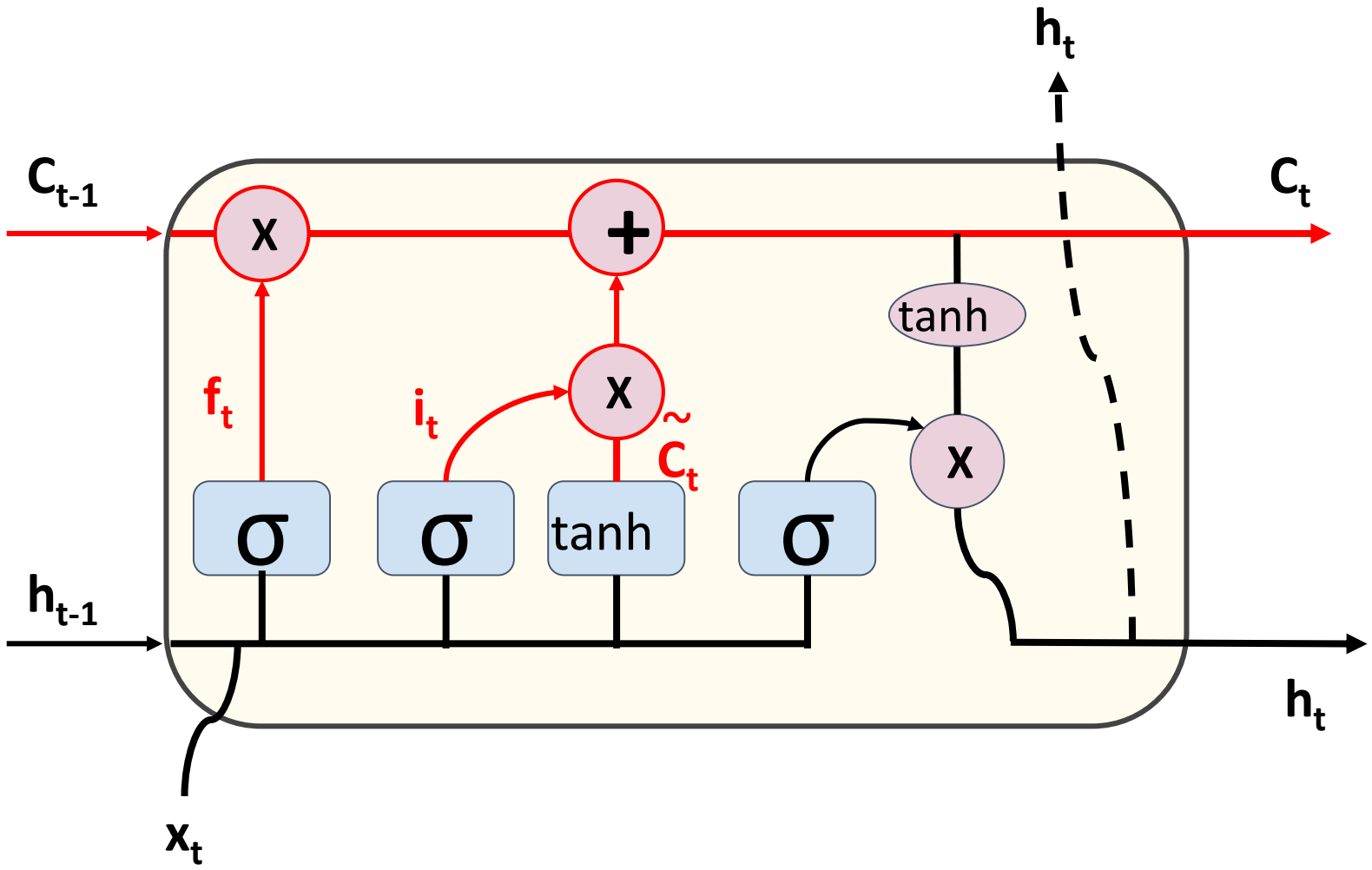
12.3.10 LSTM Architecture



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

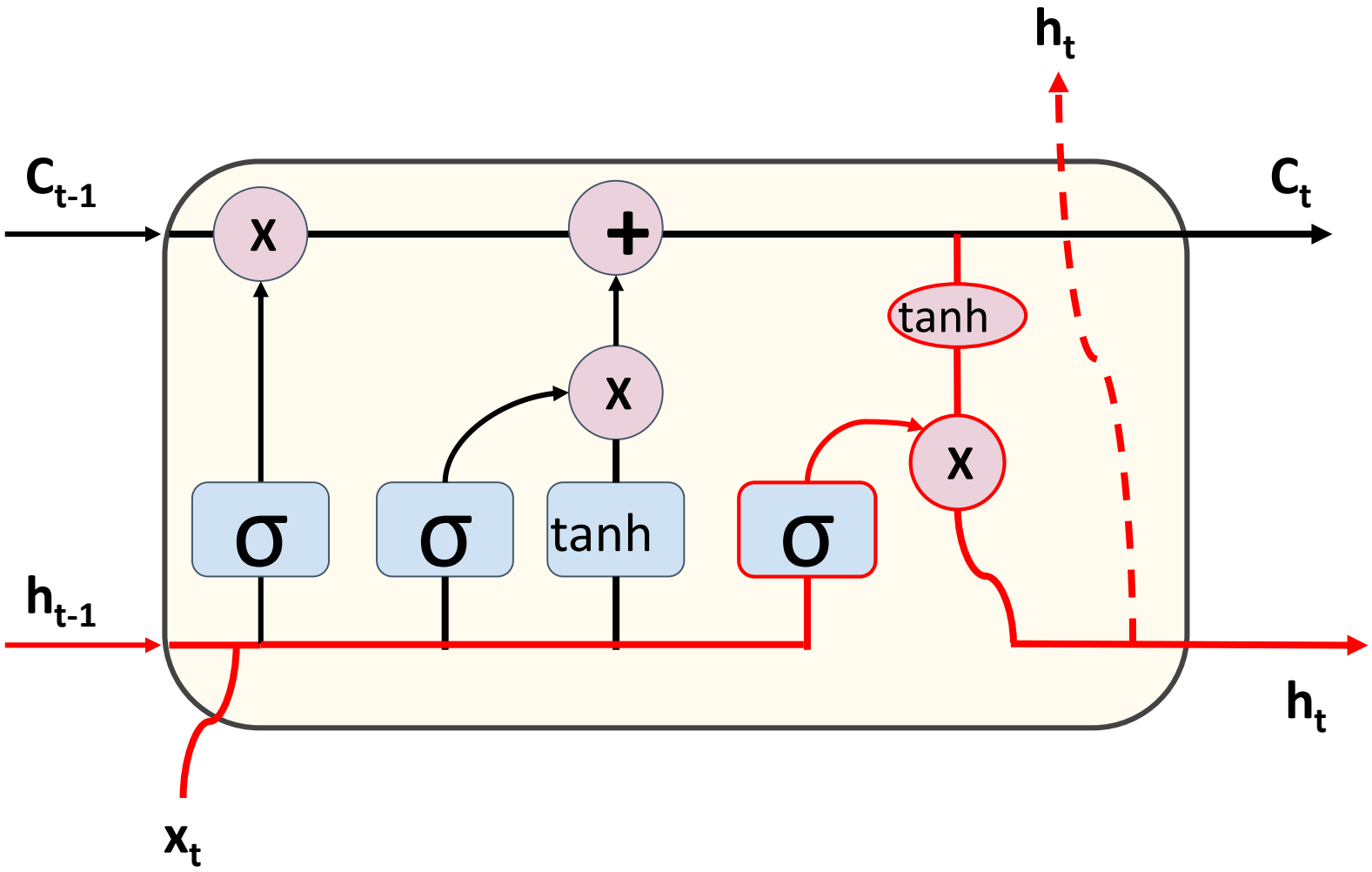
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

12.3.10 LSTM Architecture



$$C_t = f_t \circ C_{t-1} + i_t \circ \tilde{C}_t$$

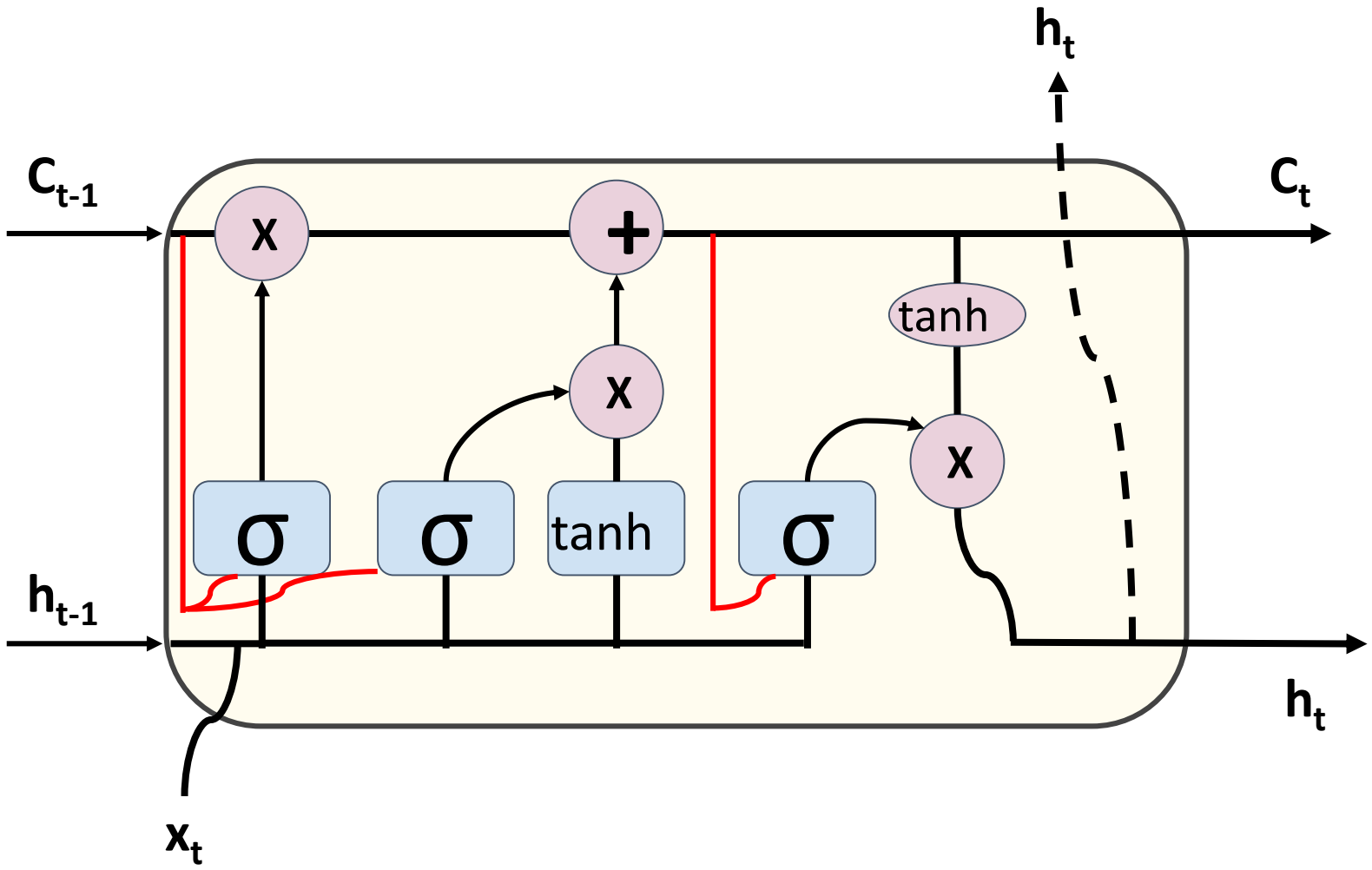
12.3.10 LSTM Architecture



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t \circ \tanh\left(C_t\right)$$

Peephole LSTM



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

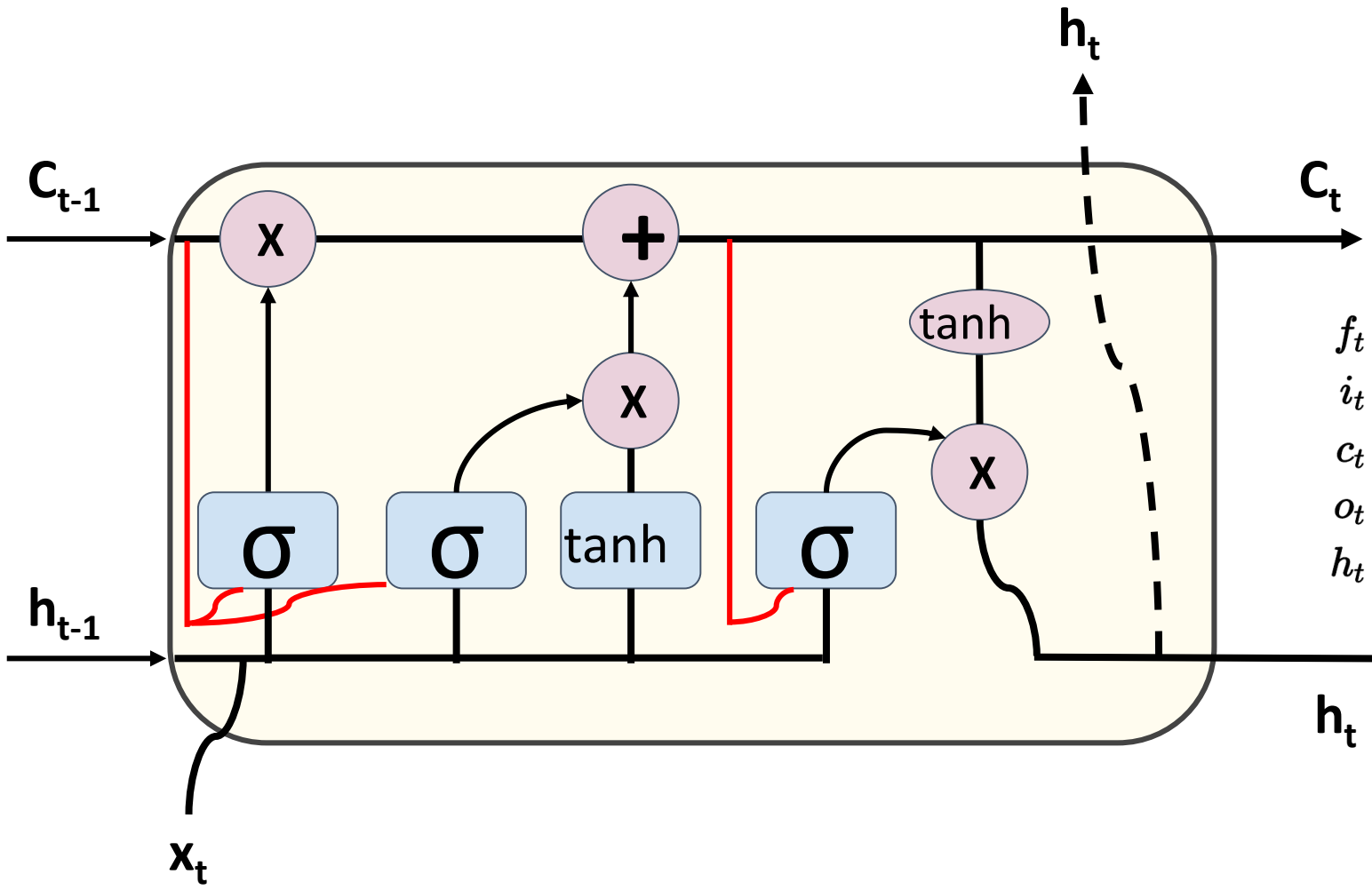
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$C_t = f_t \circ C_{t-1} + i_t \circ \tilde{C}_t$$

$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$

$$h_t = o_t \circ \tanh\left(C_t\right)$$

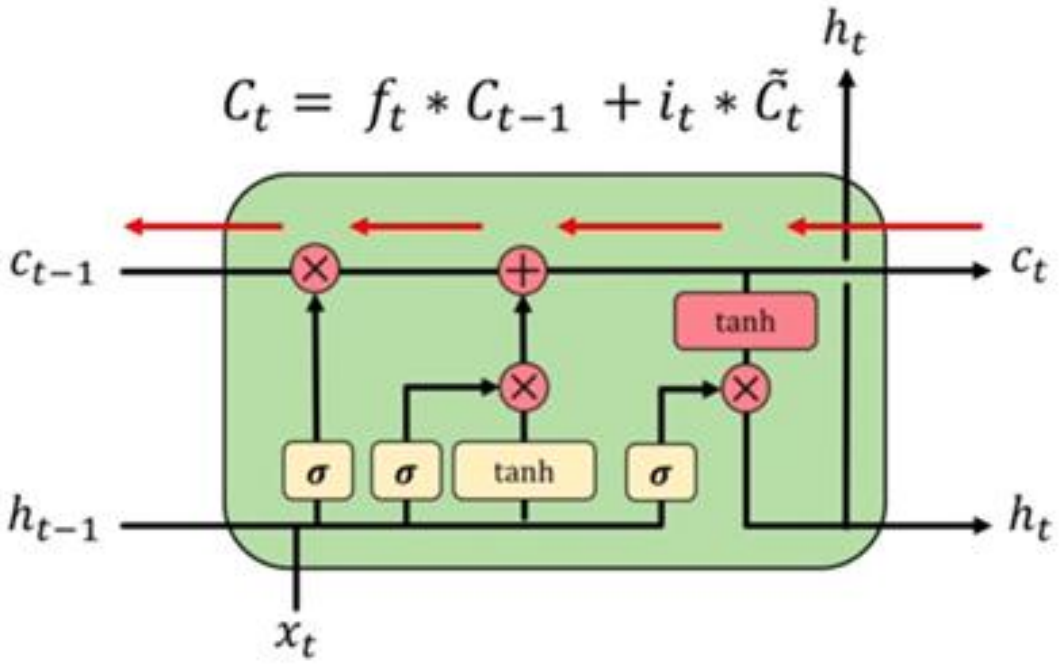
Peephole Convolutional LSTM



$$f_t = \sigma_g(W_f * x_t + U_f * h_{t-1} + V_f \circ c_{t-1} + b_f)$$
$$i_t = \sigma_g(W_i * x_t + U_i * h_{t-1} + V_i \circ c_{t-1} + b_i)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c * x_t + U_c * h_{t-1} + b_c)$$
$$o_t = \sigma_g(W_o * x_t + U_o * h_{t-1} + V_o \circ c_t + b_o)$$
$$h_t = o_t \circ \sigma_h(c_t)$$

12.3.12 LSTM Training

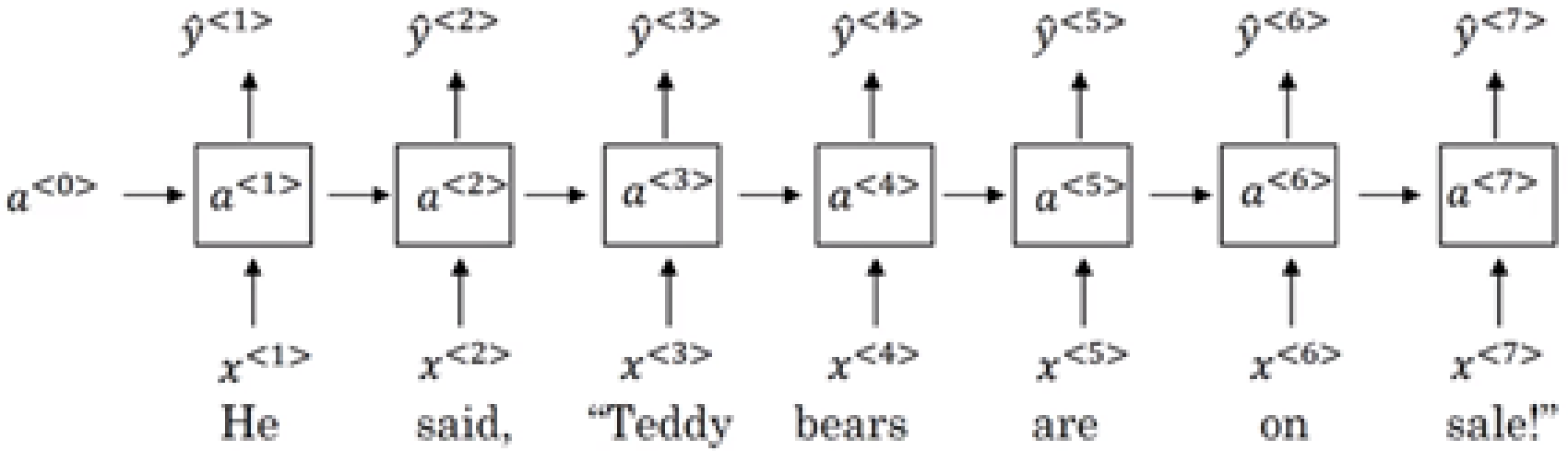
- Backpropagation from C_t to C_{t-1} requires only elementwise multiplication.
- No matrix multiplication → avoid vanishing gradient problem.



Getting Information from Future

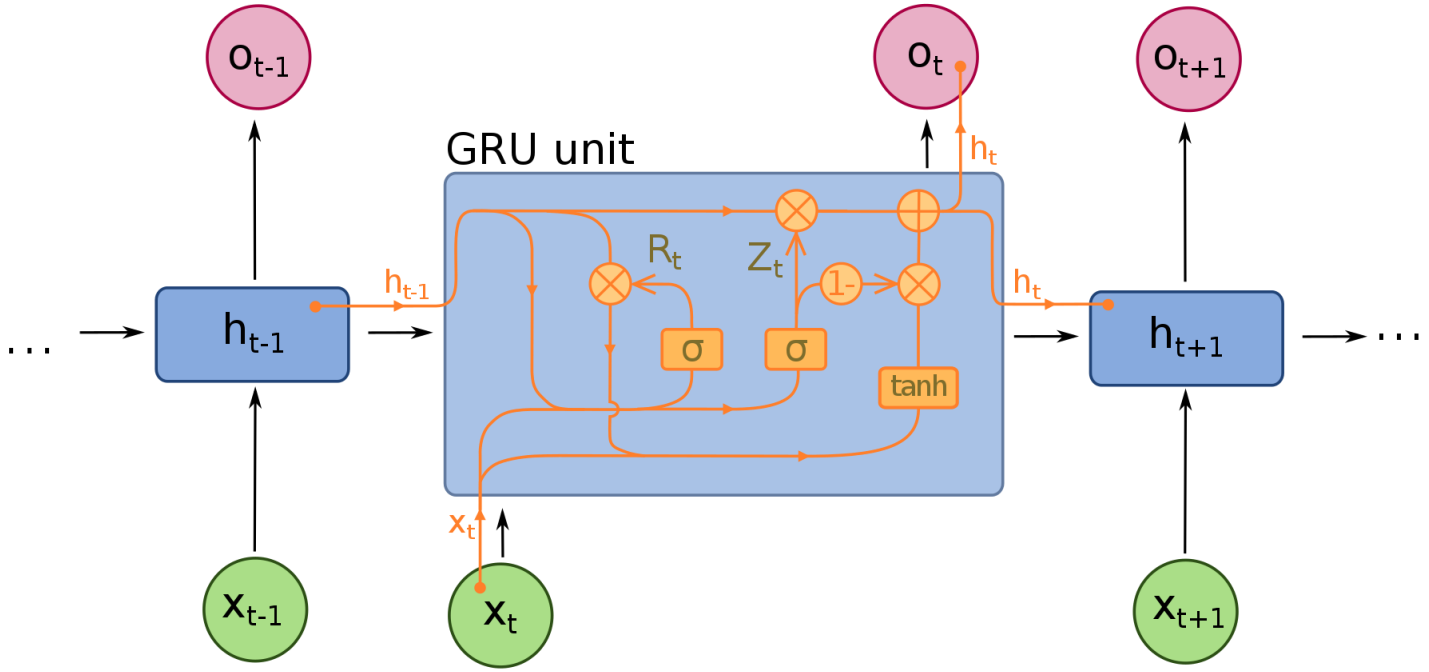
He said, "Teddy bears are on sale!"

He said, "Teddy Roosevelt was a great president!"



12.3.14 Gated Recurrent Unit

- Introduced by Kyunghyun Cho et. al. [2014] with gating mechanism in RNN.
- They are used in the full form and several simplified variants.
- They have fewer parameters than LSTM, as they lack an output gate.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$R_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [R_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

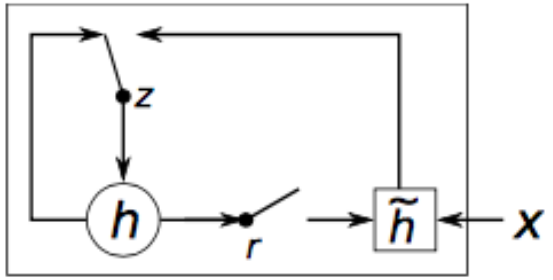
12.3.14 Gated Recurrent Unit

- Standard RNN computes hidden layer at next time step directly

$$h_t = \sigma(W^{(hh)}h_{t-1} + W^{(hx)}x_t)$$

- Compute an update gate based on current input word vector and hidden state

$$z_t = \sigma(U^{(z)}h_{t-1} + W^{(z)}x_t)$$



- Controls how much of past state should matter now.
- If z close to 1, then copy information in that unit through many steps

- Compute a reset gate similarly but with different weights

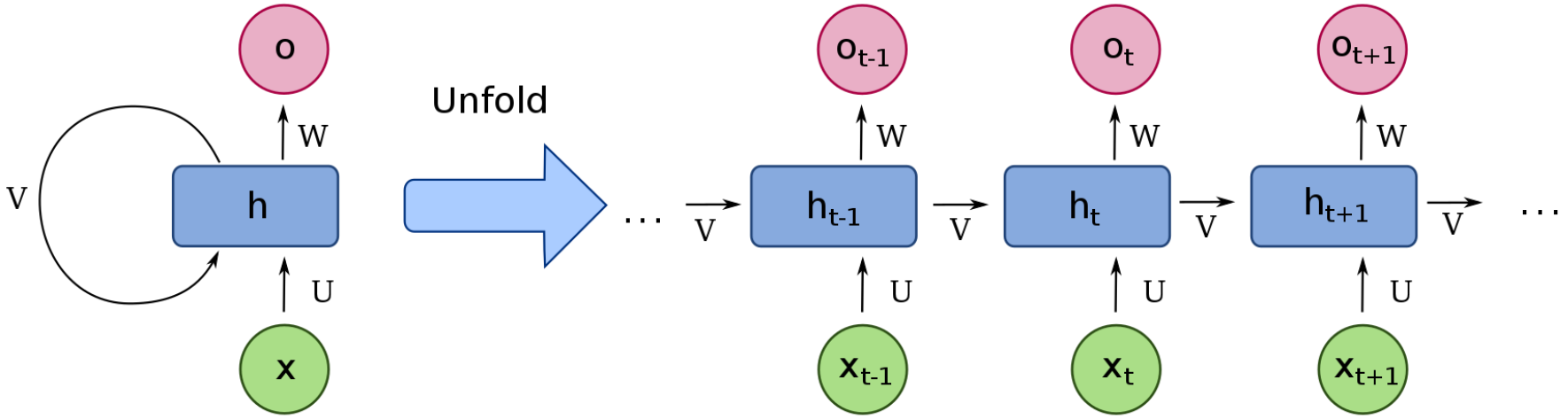
$$r_t = \sigma\left(U^{(r)}h_{t-1} + W^{(r)}x_t\right)$$

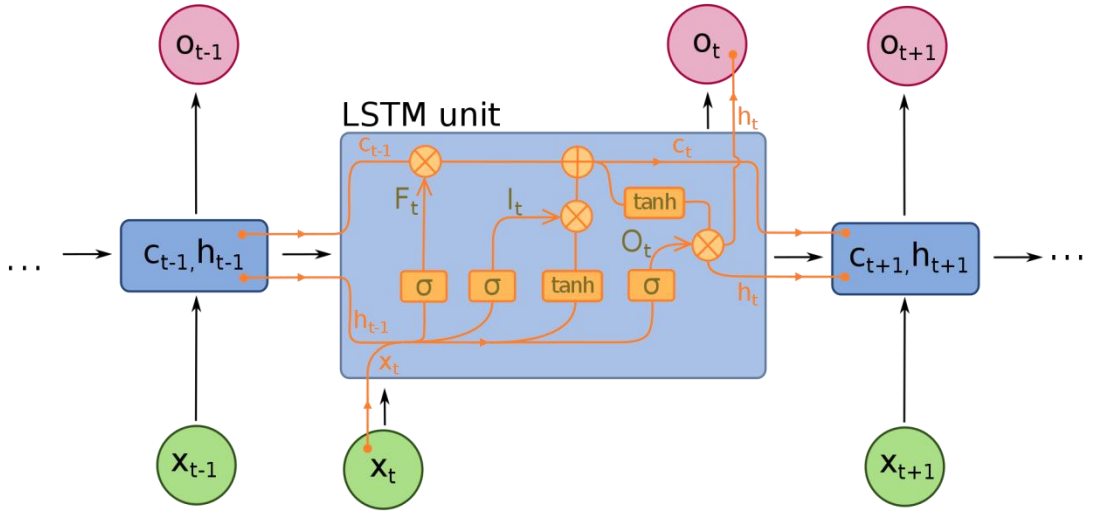
- If reset close to 0, ignore previous hidden state (allows model to drop information that is irrelevant in future)
- Units with short-term dependencies have reset gates very active and long-term dependencies have active update gates z

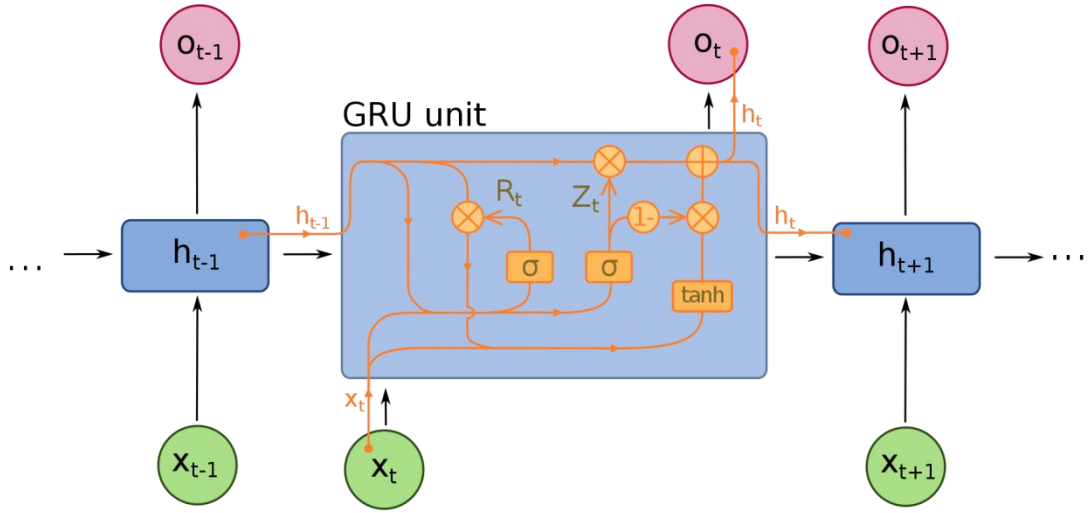
- New memory $\tilde{h}_t = tanh(r_t \circ Uh_{t-1} + Wx_t)$
- Final memory $h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t$

- Combines current & previous time steps

12.3.15 RNN Summary

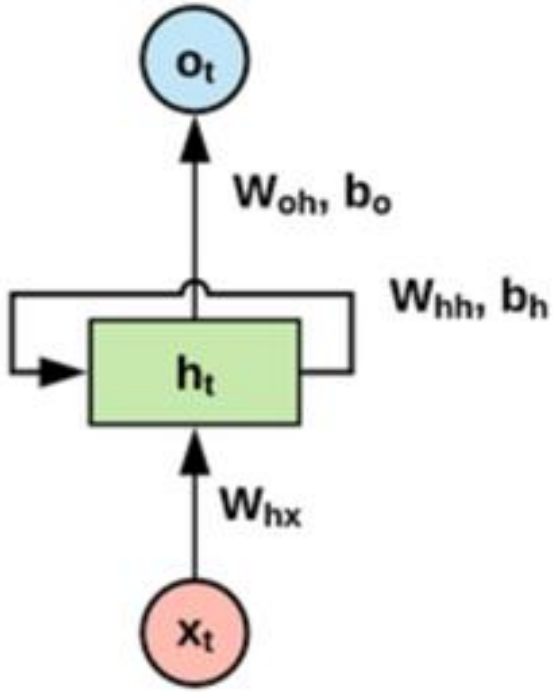






12.3.16 Exercises

- For the following recurrent feedback network. Calculate the output state for timestamp t=1 to 3.
  - For simplicity assuming inputs and weights are 1 and .5 respectively for all timestamps.



12.4 References

The Topics Covered in This Section

12.4.1 Books
12.4.2 Video Lectures

12.4.1 Books

- Kelleher, J.D. (2019). *Deep Learning*. MIT Press.

➢ This book explains some basic concepts of deep learning along with how it enables data-driven decisions by extracting patterns from large datasets. Deep learning architectures like auto encoders, CNN, RNN are also covered here. ★★★★

- Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

➢ This textbook covers mathematical and conceptual background covering relevant concepts in linear algebra, probability, machine learning. Deep learning techniques like deep feedforward networks, regularization, optimization algorithms and convolution networks. ★★★★

- Chollet, F. (2017). *Deep Learning with Python*. Manning.

➢ Introduces the field of deep learning using Python and Keras, a powerful tool. Concepts and applications in the fields of Computer Vision, NLP are also as part of the book.★★★★

- Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow.* O'Reilly Media.

➢ This book explores the machine learning landscape, particularly neural networks. Use of Scikit learn, Keras and Tensor flow libraries to build and train neural nets have been discussed in detail. ★★★

12.4.2 Video Lectures

- CS230: Deep Learning | Autumn 2018 | Stanford University

https://www.youtube.com/watch?v=PySo_6S4ZAg&list=PLoROMvodv4rOABXSygHTsbvUz4G_YQhOb

- MIT Introduction to Deep Learning | 6.S191 | Alexander Amini

https://www.youtube.com/watch?v=njKP3FqW3Sk&list=PLtBw6njQRU-rwp5__7C0oIVt26ZgjG9NI

- Carnegie Mellon University Deep Learning

https://www.youtube.com/watch?v=LmIjgmijyiI&list=PLp-0K3kfddPwz13VqV1PaMXF6V6dYdEsj

- Deep Learning UC Berkeley STAT-157 2019

https://www.youtube.com/watch?v=Va8WWRfw7Og&list=PLZSO_6-bSqHQHBCoGaObUljoXAyyqhpFW

- Deep Learning | IITM

https://www.youtube.com/watch?v=aPfkYu_qiF4&list=PL3pGy4HtqwD2kwldm81pszxZDJANK3uGV

- Machine Learning for Engineering and Science Applications | IITM

https://www.youtube.com/watch?v=_M-nDb0MIa4&list=PLyqSpQzTE6M-SISTunGRBRiZk7opYBf_K

- Deep Learning Lectures | Lex Fridman

https://www.youtube.com/watch?v=4zrU54VIK6k&list=PLrAXtmErZgOeiKm4sgNOknGvNjby9efdf&index=3

Thank You