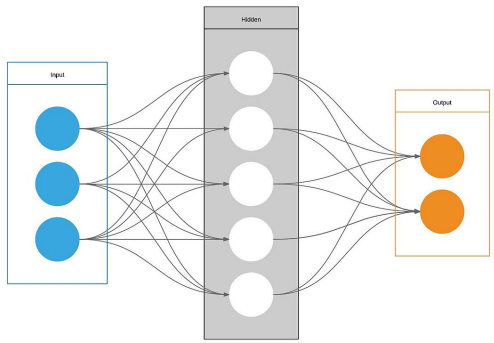
RNNs and CNNs

Cecilia Jarne

cecilia.jarne@unq.edu.ar

Organization For Computational Neurosciences



WTPC Workshop en técnicas de programación científica

Twitter: @ceciliajarne

CONICET

Universidad Nacional de Quilmes

What is this talk about?

- Recurrent Neural Networks (RNN).
- Convolutional Neural Networks (CNN).
- Motivation and some applications.

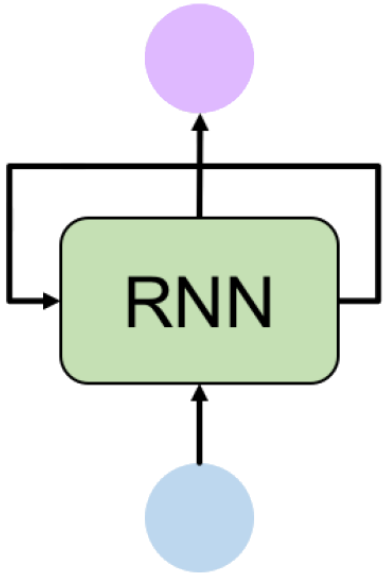
Recurrent Neural Networks (RNN)

- Sequence prediction: given a sequence of elements to predict the next one.
- Some example application are:
  - Text sequence prediction.
  - Time series in general.
  - Speech.
- Models of cortex and other brain areas great recurrence in their connections.

Recurrent Neural Networks

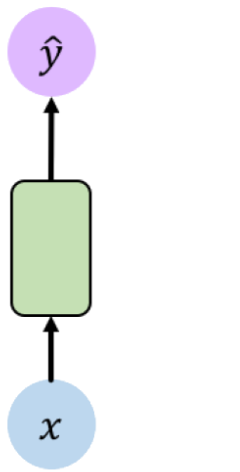
To model sequences, we need to:

1. Handle **variable-length** sequences
2. Track **long-term** dependencies
3. Maintain information about **order**
4. **Share parameters** across the sequence

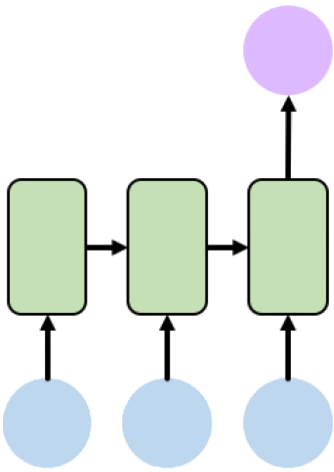


Today: **Recurrent Neural Networks (RNNs)** as an approach to sequence modeling problems

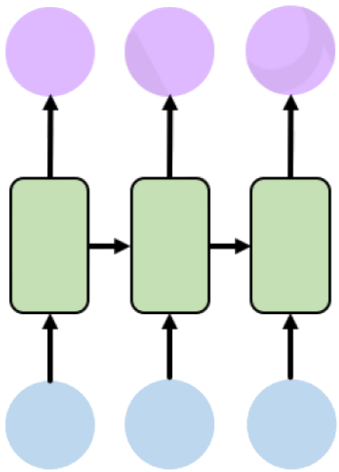
Recurrent Neural Networks



One to One
"Vanilla" neural network



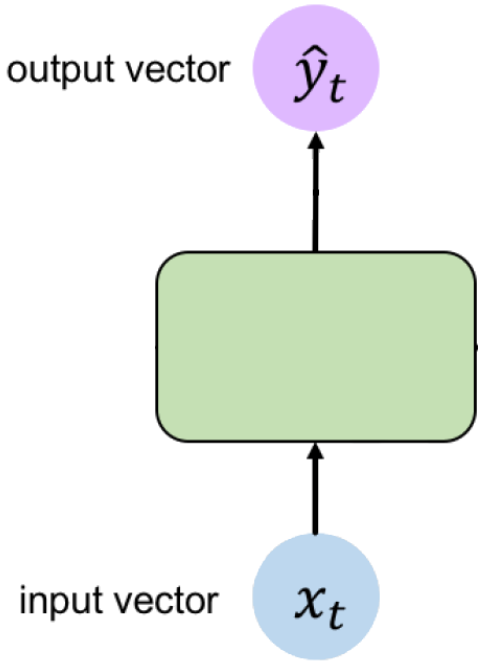
Many to One
Sentiment Classification



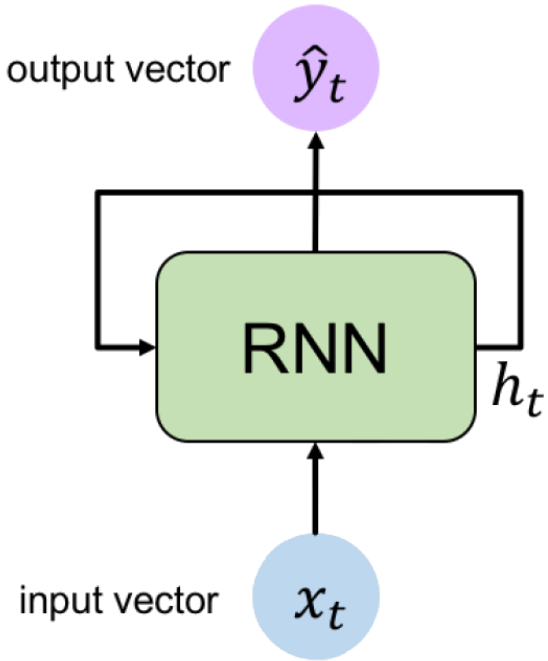
Many to Many
Music Generation

… and many other architectures and applications

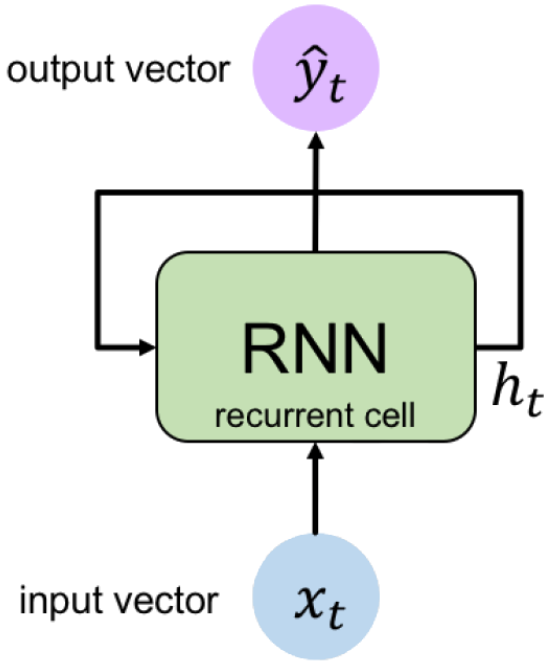
Recurrent Neural Networks



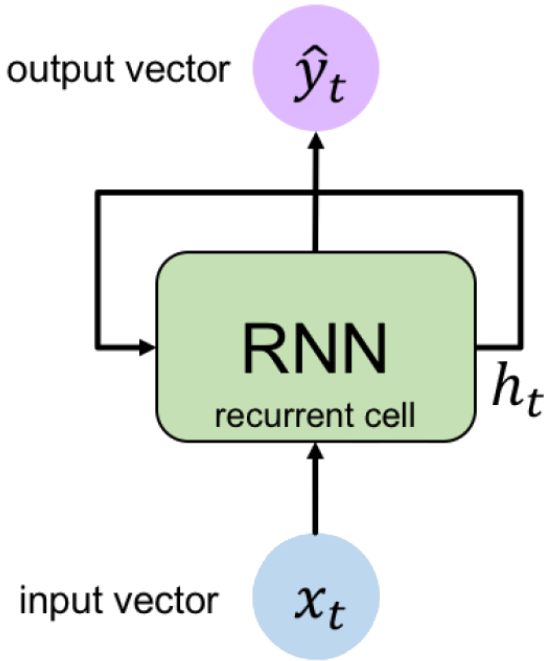
Recurrent Neural Networks



Recurrent Neural Networks

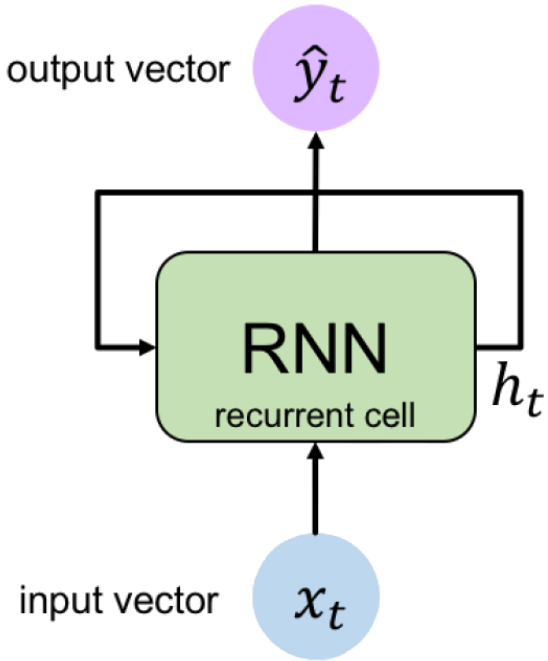


Recurrent Neural Networks



Apply a **recurrence relation** at every time step to process a sequence:

Recurrent Neural Networks



Apply a **recurrence relation** at every time step to process a sequence:

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

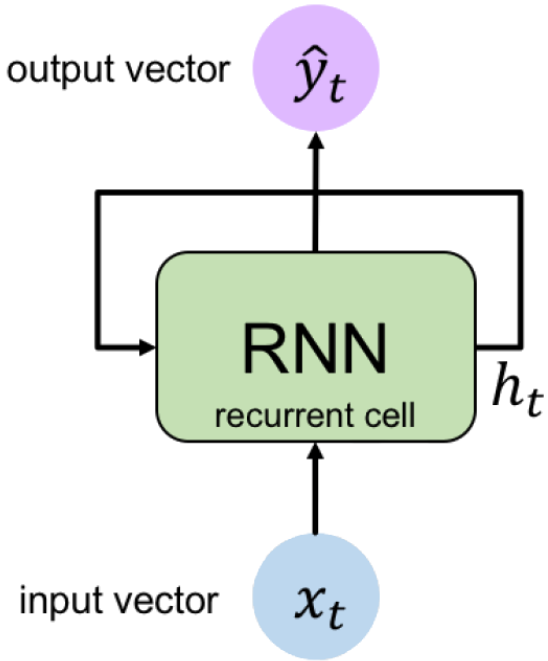
cell state

function parameterized by W

old state

input vector at time step *t*

Recurrent Neural Networks



Apply a **recurrence relation** at every time step to process a sequence:

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

cell state

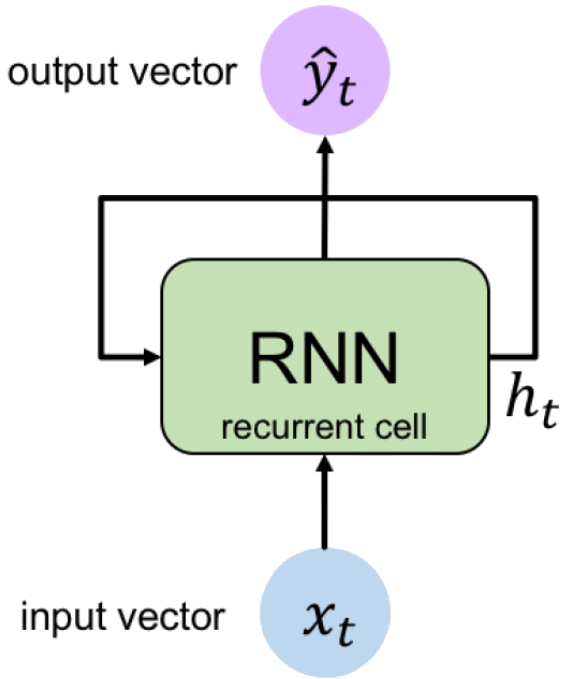
function parameterized by W

old state

input vector at time step t

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

Recurrent Neural Networks



Output Vector

$$\hat{y}_t = \boldsymbol{W}_{\boldsymbol{hy}}^{\boldsymbol{T}} h_t$$

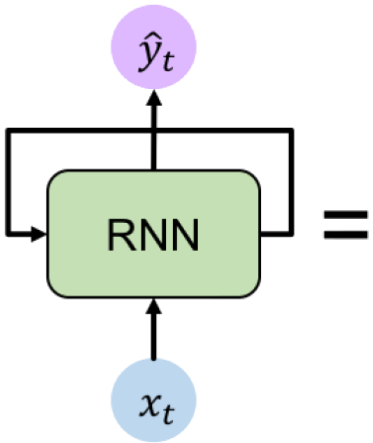
Update Hidden State

$$h_t = \tanh(\boldsymbol{W}_{\boldsymbol{hh}}^{\boldsymbol{T}} h_{t-1} + \boldsymbol{W}_{\boldsymbol{xh}}^{\boldsymbol{T}} x_t)$$

Input Vector

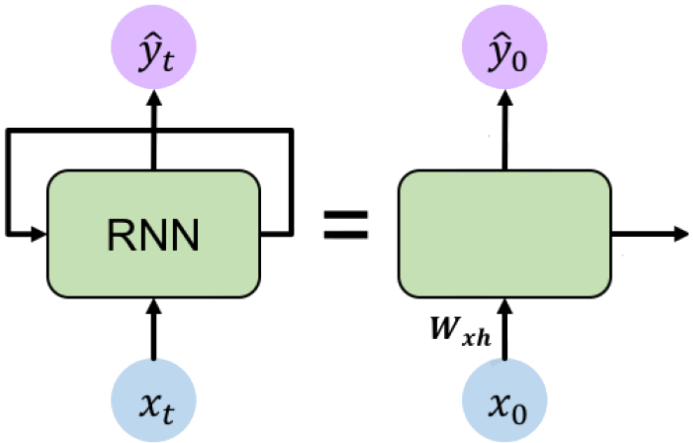
$$x_t$$

Recurrent Neural Networks

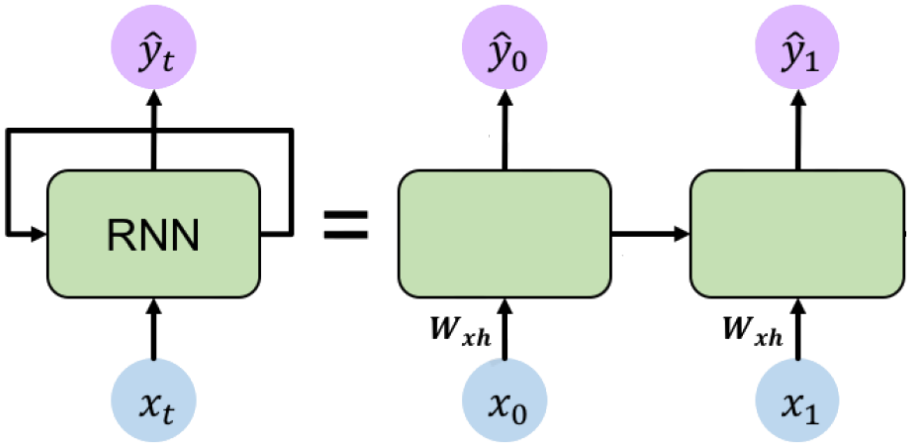


=

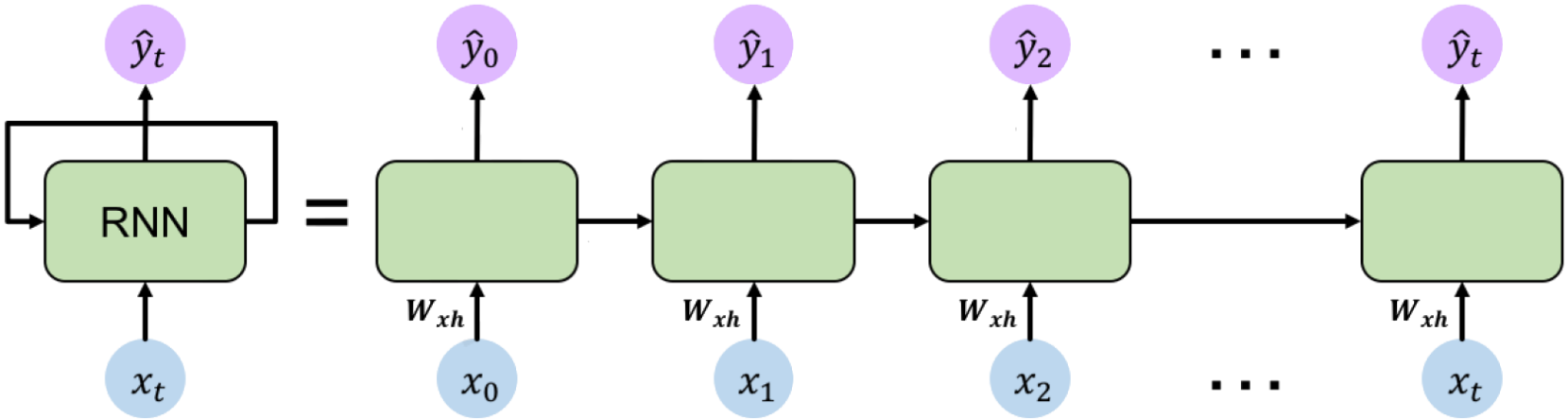
Recurrent Neural Networks



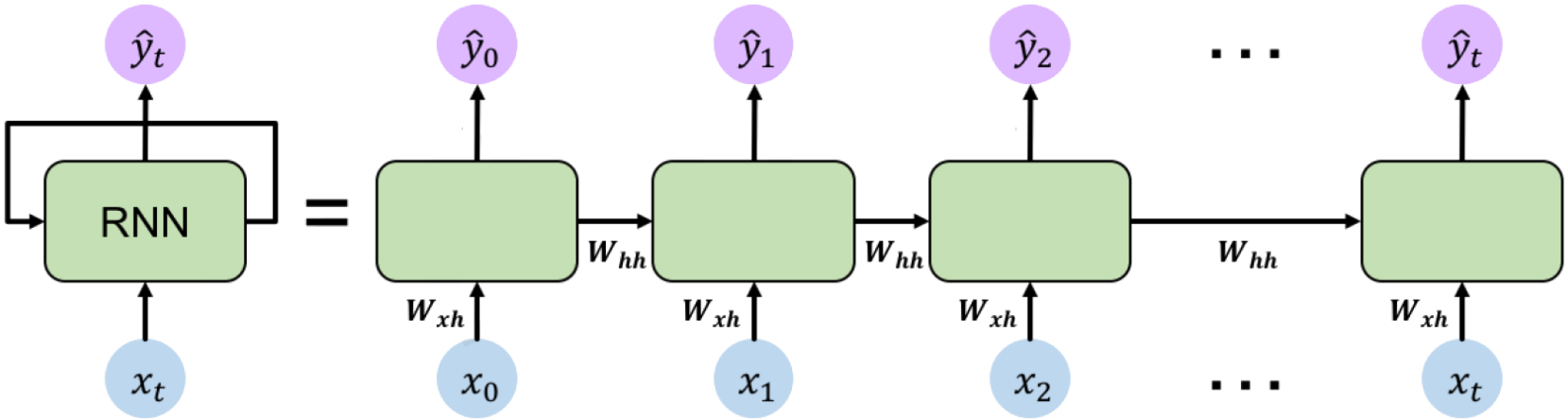
Recurrent Neural Networks



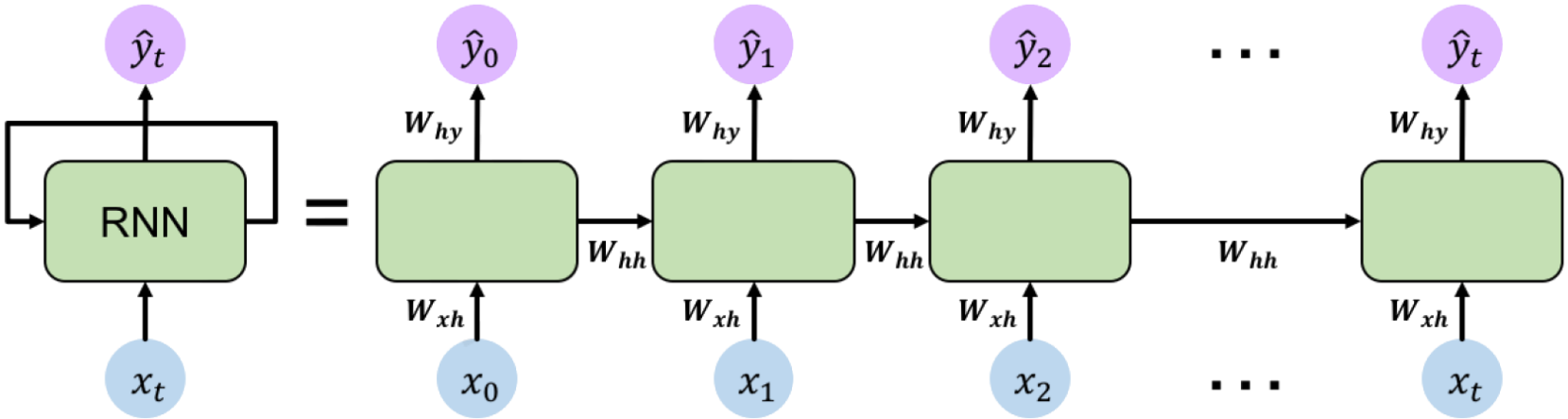
Recurrent Neural Networks



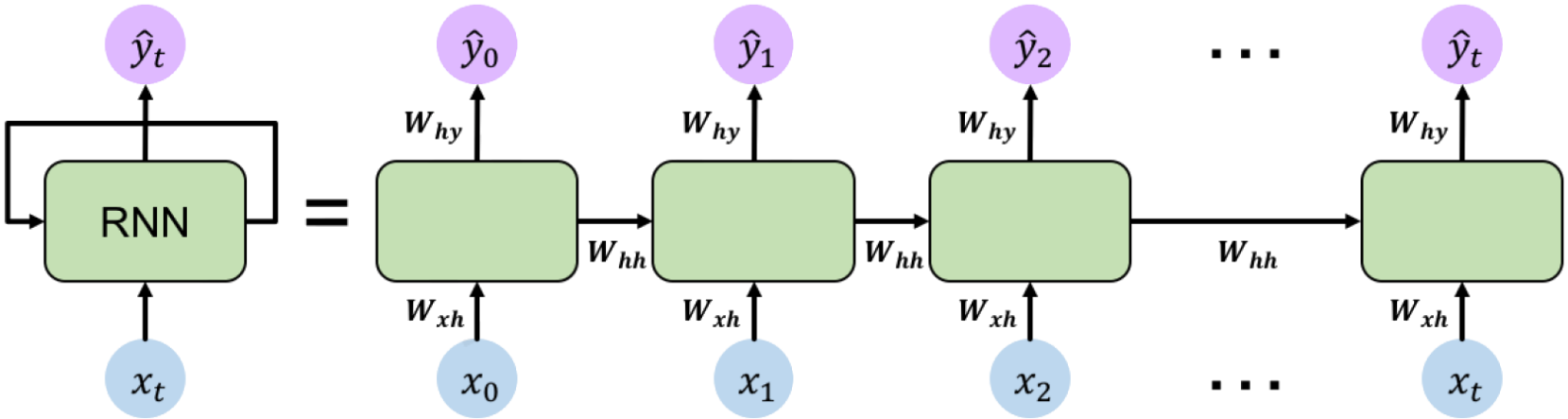
Recurrent Neural Networks



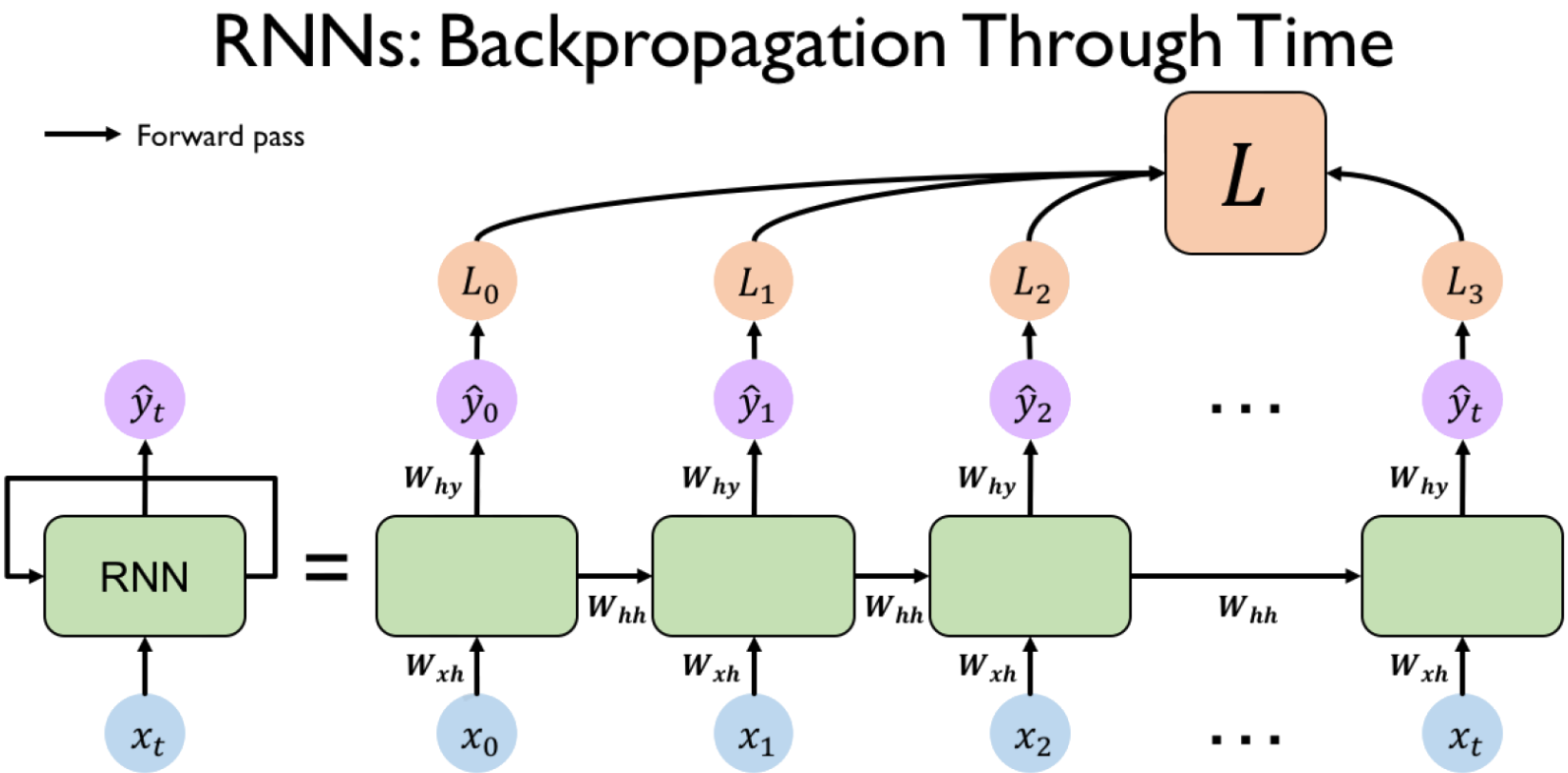
Recurrent Neural Networks



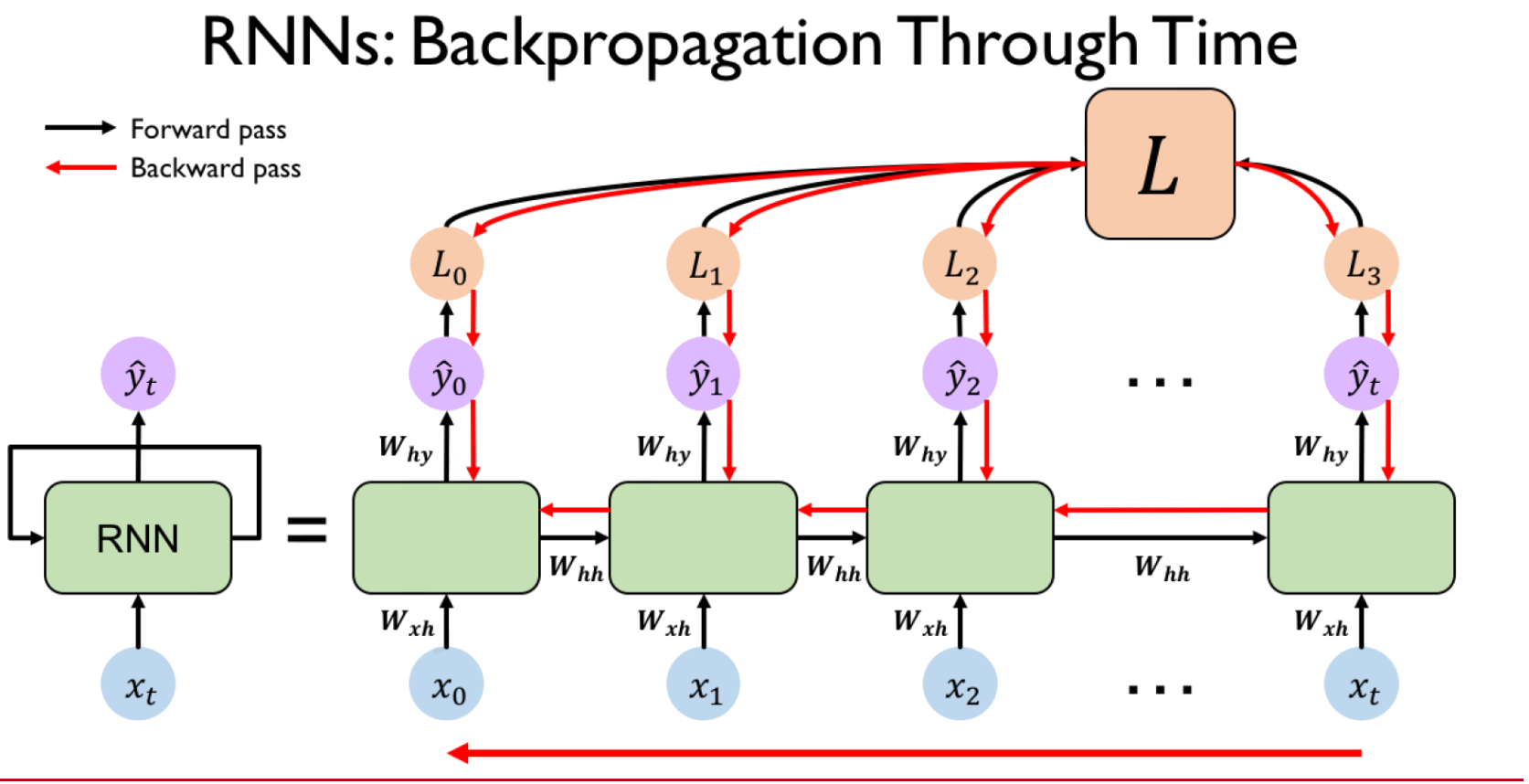
Backpropagation through time



Backpropagation through time



Backpropagation through time



Backpropagation through time: Long time dependencies

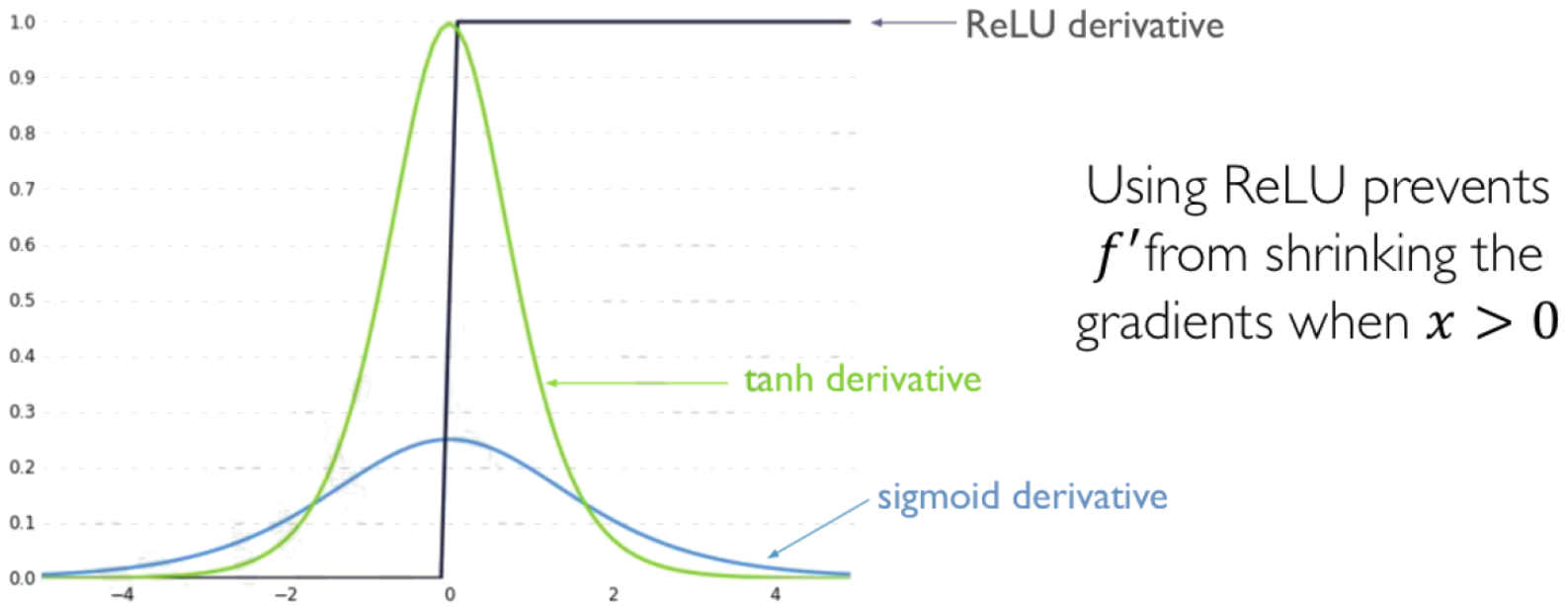
Why are vanishing gradients a problem?

Multiply many **small numbers** together

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

Bias parameters to capture short-term dependencies

How to solve it:



Using ReLU prevents f' from shrinking the gradients when $x > 0$

How to solve it:

Initialize **weights** to identity matrix

Initialize **biases** to zero

$$I_n = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

This helps prevent the weights from shrinking to zero.

How to solve it:

Idea: use a more **complex recurrent unit with gates** to control what information is passed through

Long Short Term Memory (LSTMs) networks rely on a gated cell to track information throughout many time steps.

Summary on RNN:

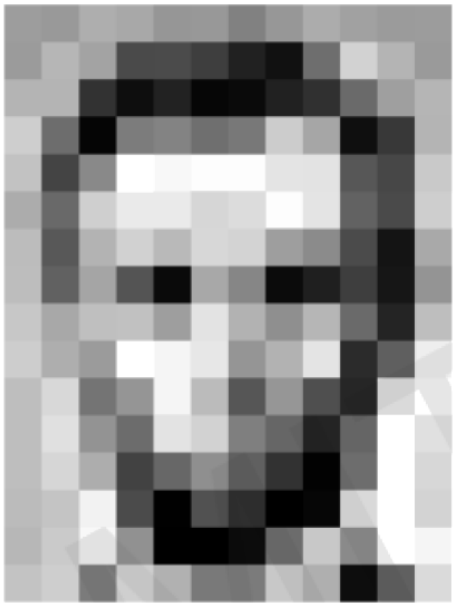
- RNN are well suitable for sequence modeling tasks.
- Model sequences via a recurrence relation.
- Training RNN with backpropagation through time.
- Gated cells allow let us model long time dependencies.
- We can model music generation, classification, machine translation and others.

Convolutional Neural Networks (CNN): A motivation

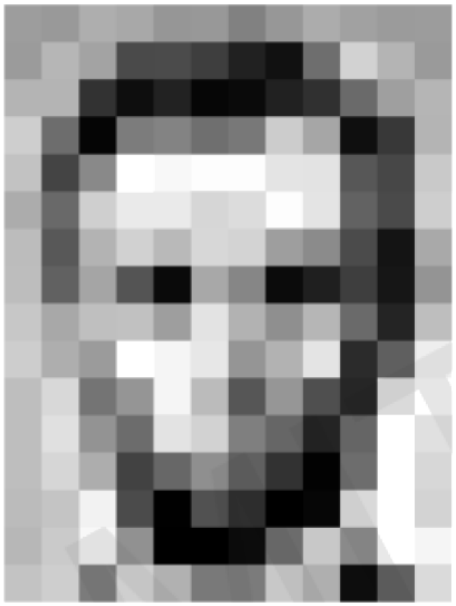
Most is all about Computer Vision, with implications on

- Facial detection and recognition.
- Healthcare, medicine and Biology.
- Self driving vehicles.

What do computers see?



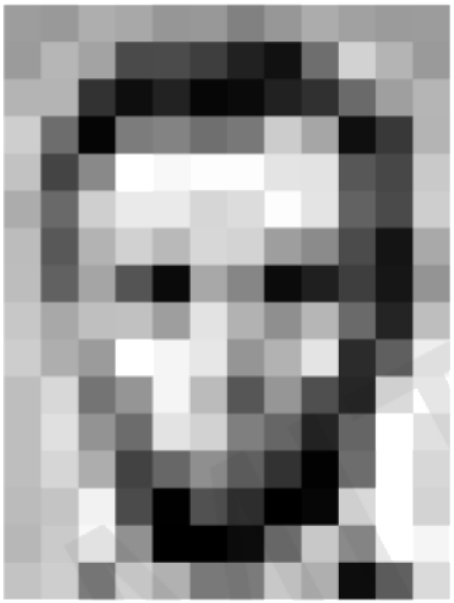
What do computers see?

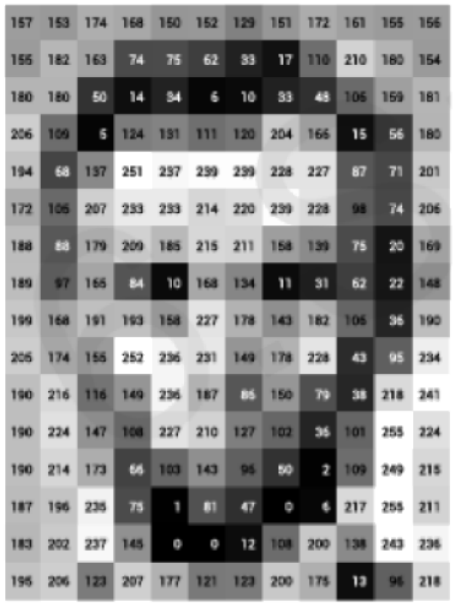


157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	95	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

An image is just a matrix of numbers [0,255]!
i.e., 1080x1080x3 for an RGB image

What do computers see?





What the computer sees

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

An image is just a matrix of numbers [0,255]!
i.e., 1080x1080x3 for an RGB image

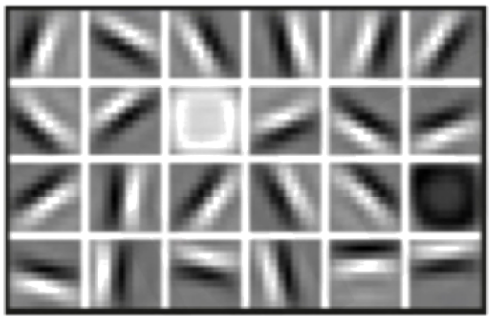
Tasks in computer vision

(As in other NN problems)

- Regression
- Classification
- High level feature detection

Features

Low Level Features



Lines & Edges

Mid Level Features

Eyes & Nose & Ears

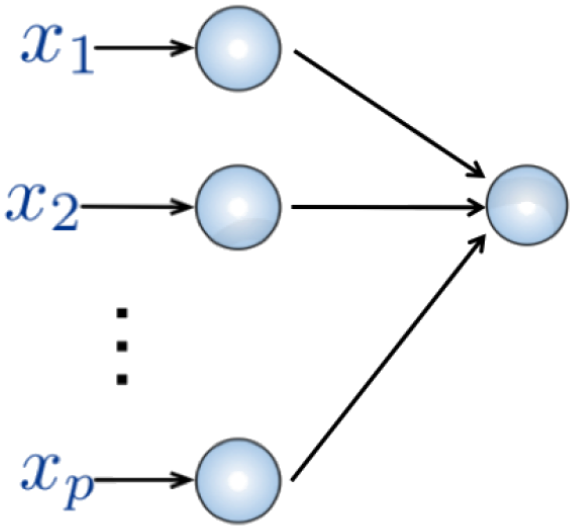
High Level Features

Facial Structure

Learning visual features

Input:

- 2D image
- Vector of pixel values



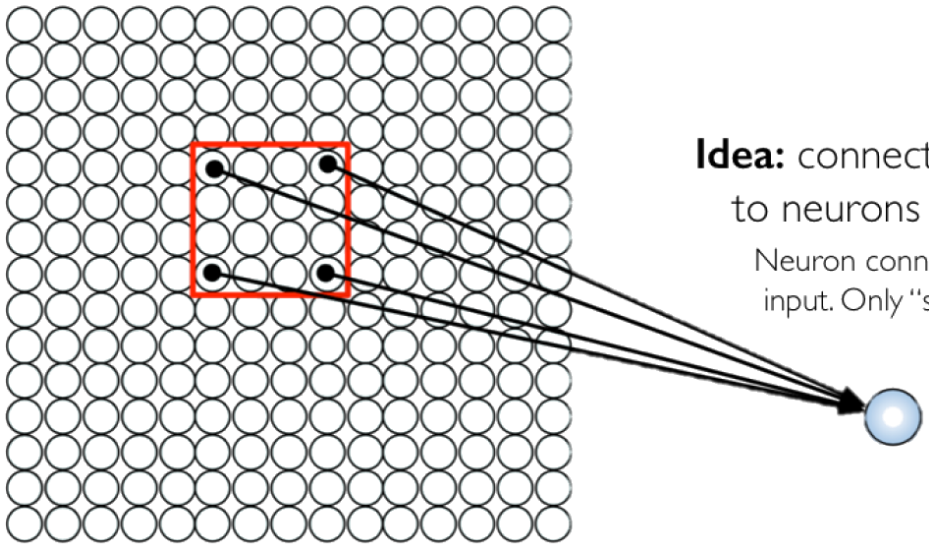
Fully Connected:

- Connect neuron in hidden layer to all neurons in input layer
- No spatial information!
- And many, many parameters!

How can we use **spatial structure** in the input to inform the architecture of the network?

Using spatial structure

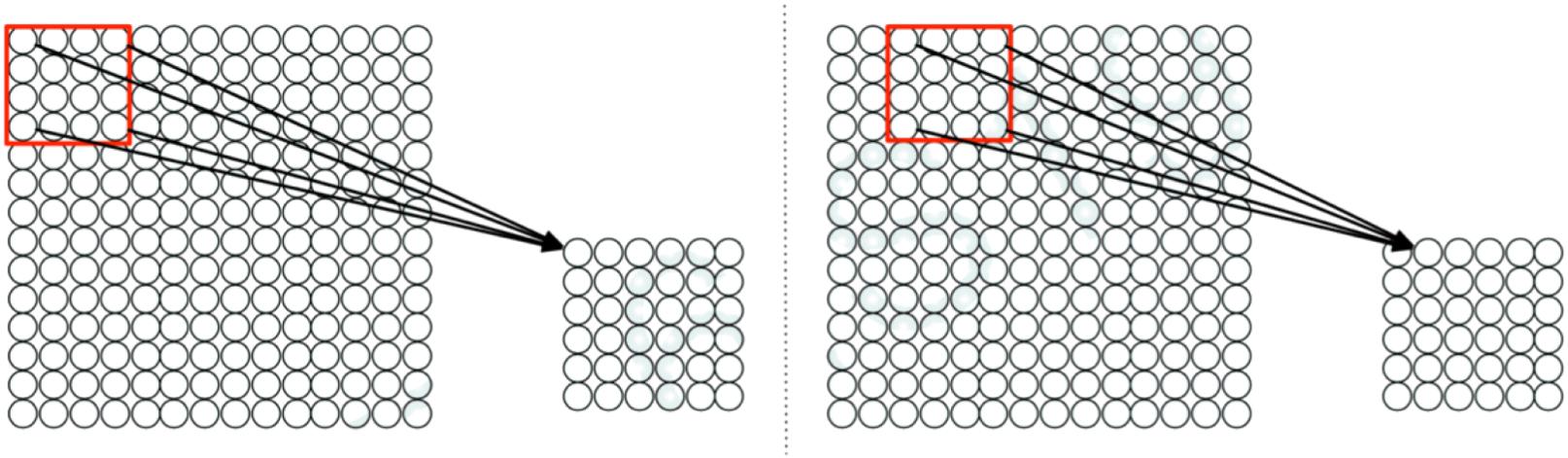
Input: 2D image.
Array of pixel values



Idea: connect patches of input to neurons in hidden layer.

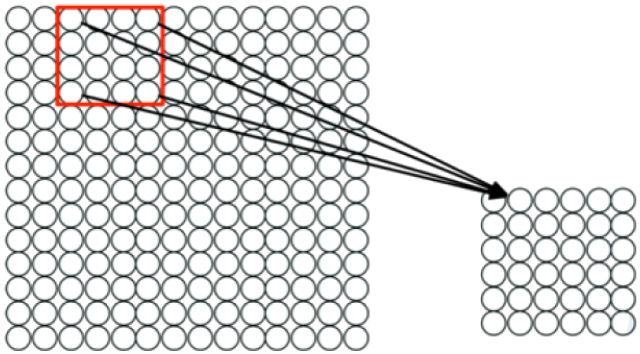
Neuron connected to region of input. Only "sees" these values.

Using spatial structure



Connect patch in input layer to a single neuron in subsequent layer.
Use a sliding window to define connections.
How can we **weight** *the patch to detect particular features?*

Applying filters to extract features



- 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**

1) Apply a set of weights – a filter – to extract **local features**

2) Use **multiple filters** to extract different features

3) **Spatially share** parameters of each filter

Convolutional operation

Suppose we want to compute the convolution of a 5x5 image and a 3x3 filter:

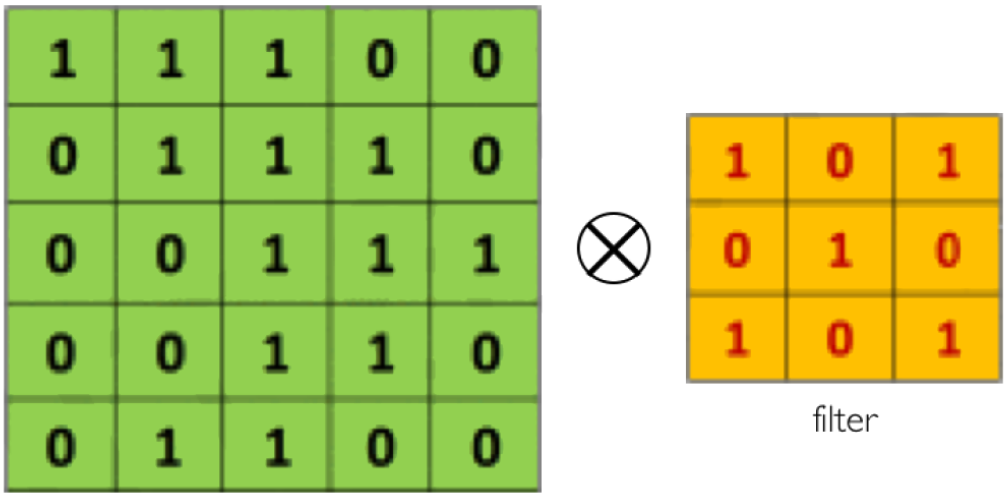
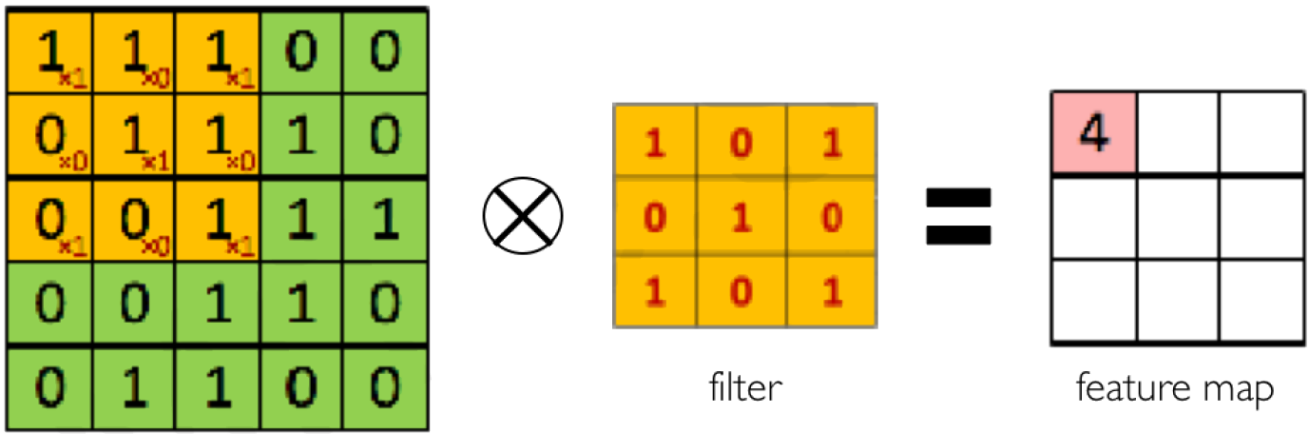


image filter

We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs…

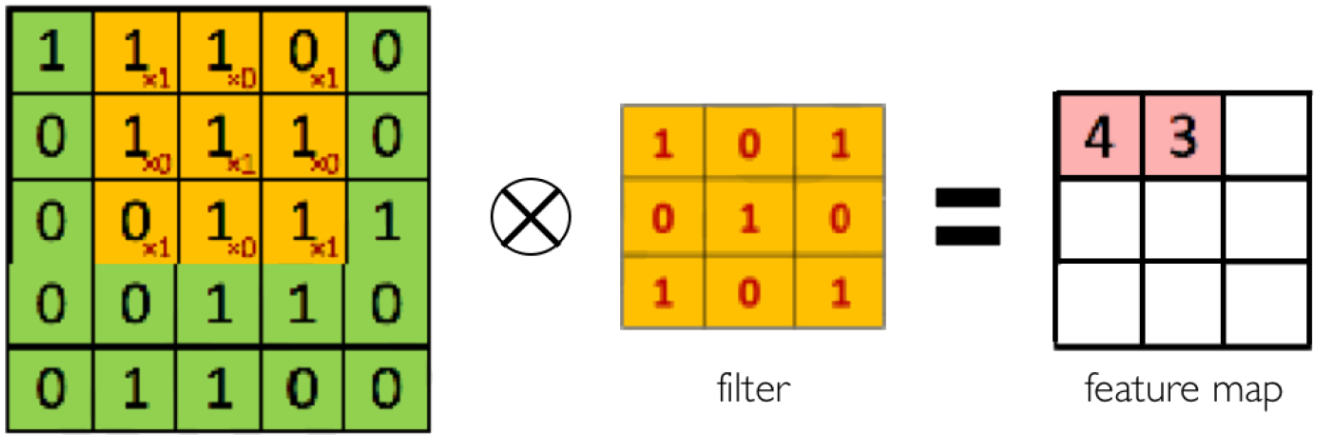
Convolutional operation

We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs:



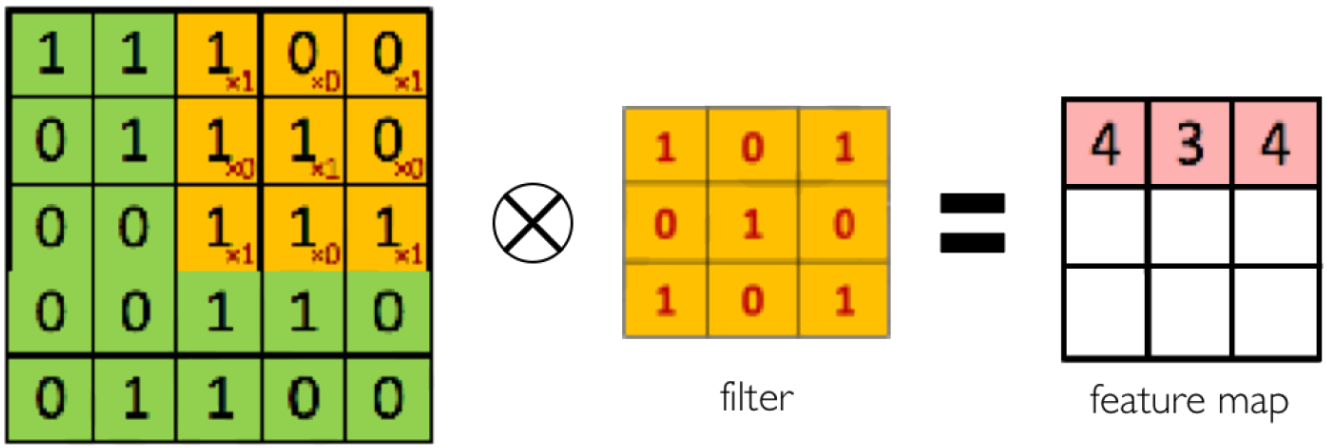
Convolutional operation

We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs:



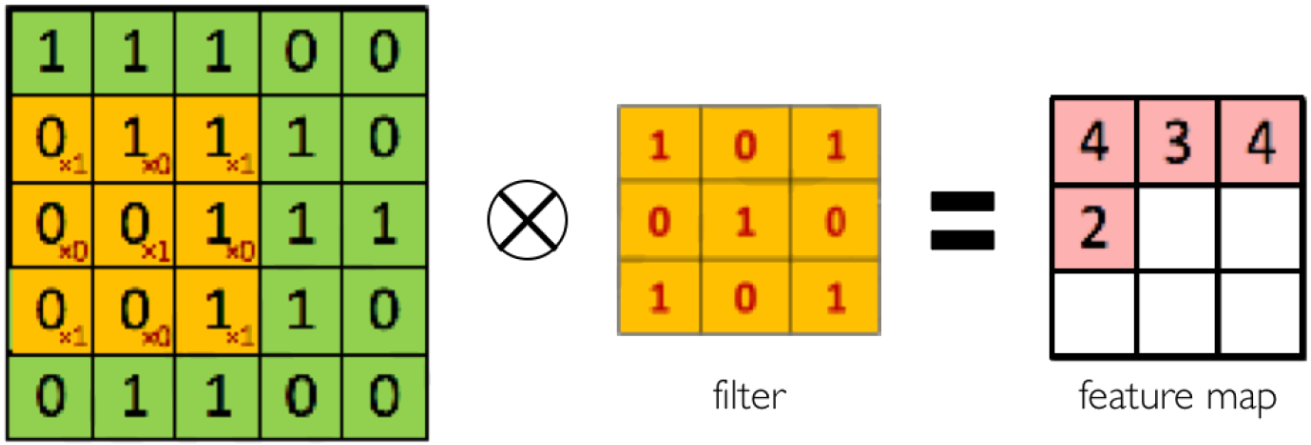
Convolutional operation

We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs:



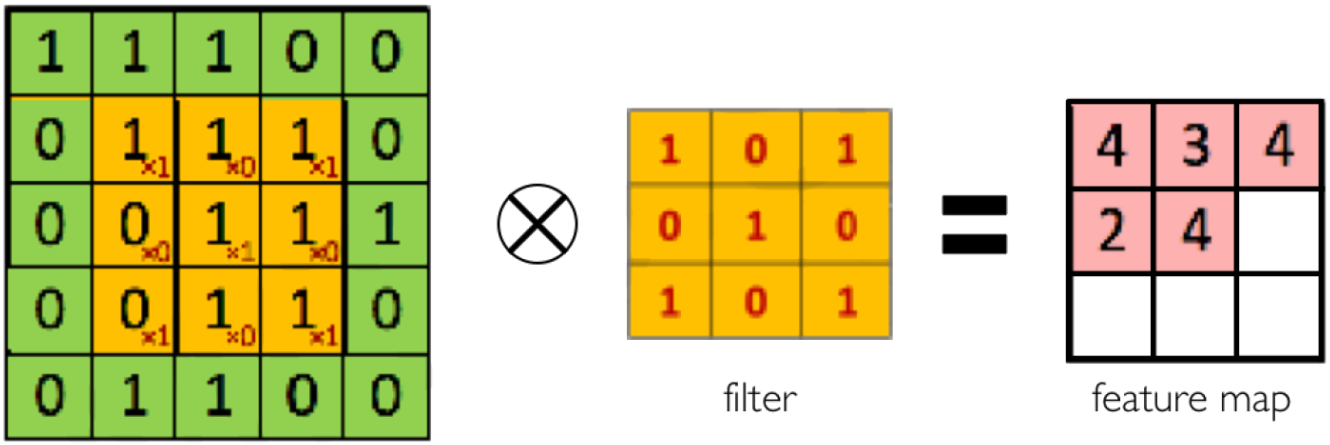
Convolutional operation

We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs:



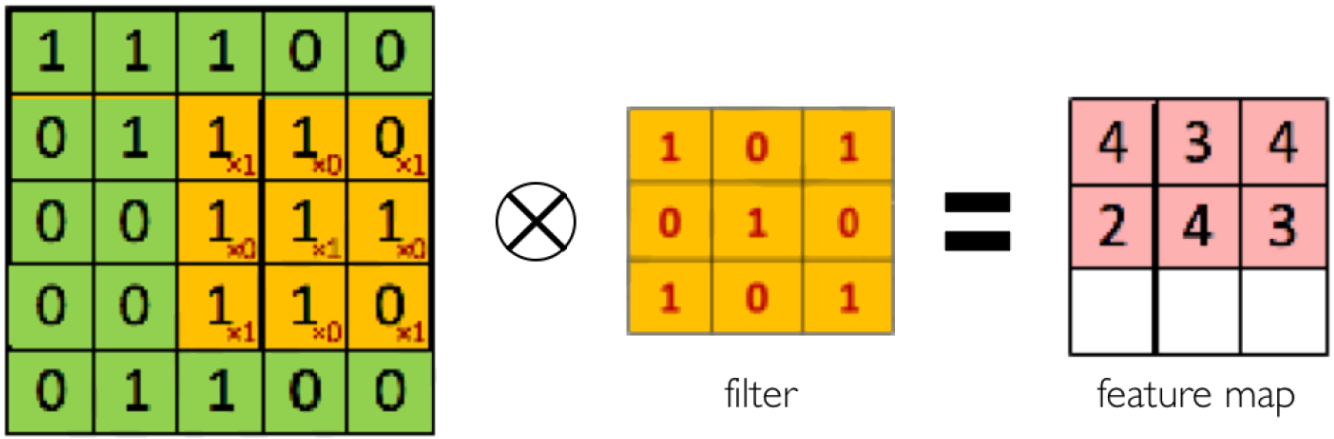
Convolutional operation

We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs:



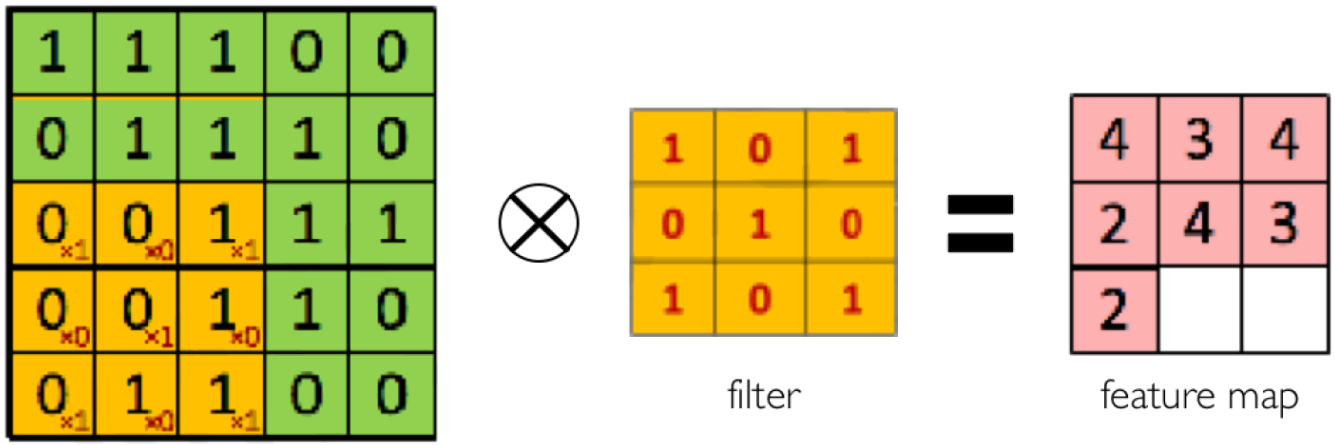
Convolutional operation

We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs:



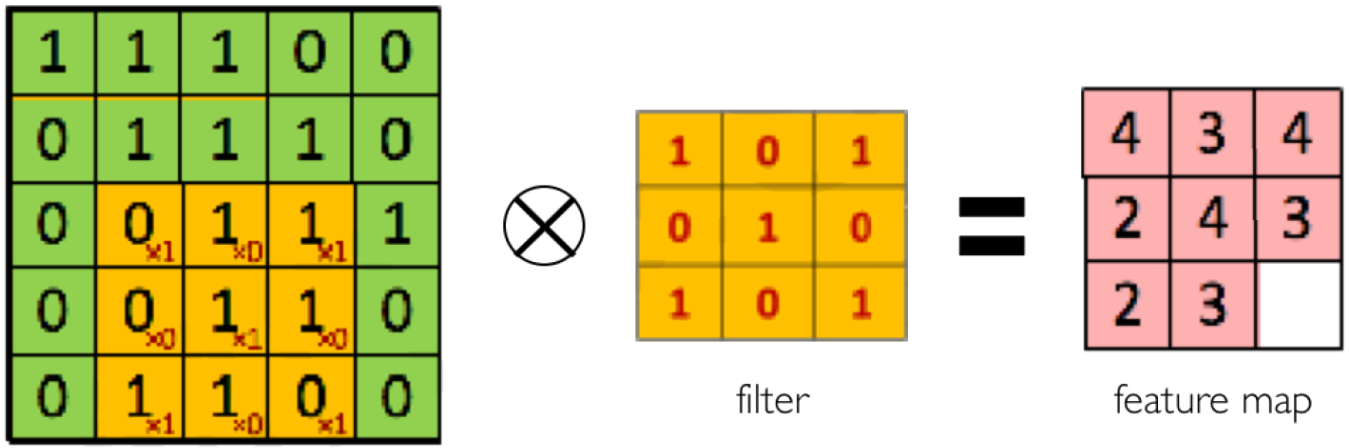
Convolutional operation

We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs:



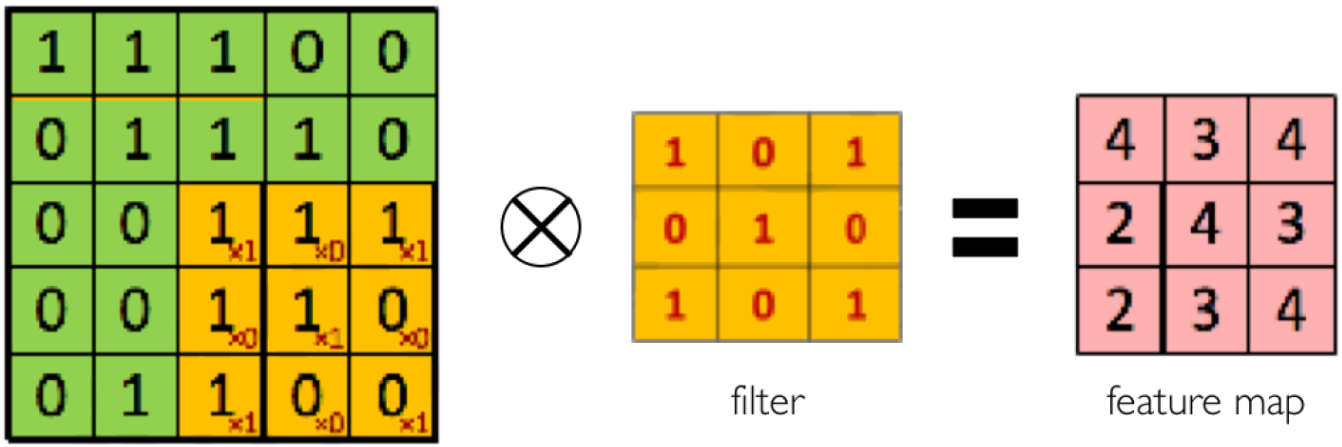
Convolutional operation

We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs:



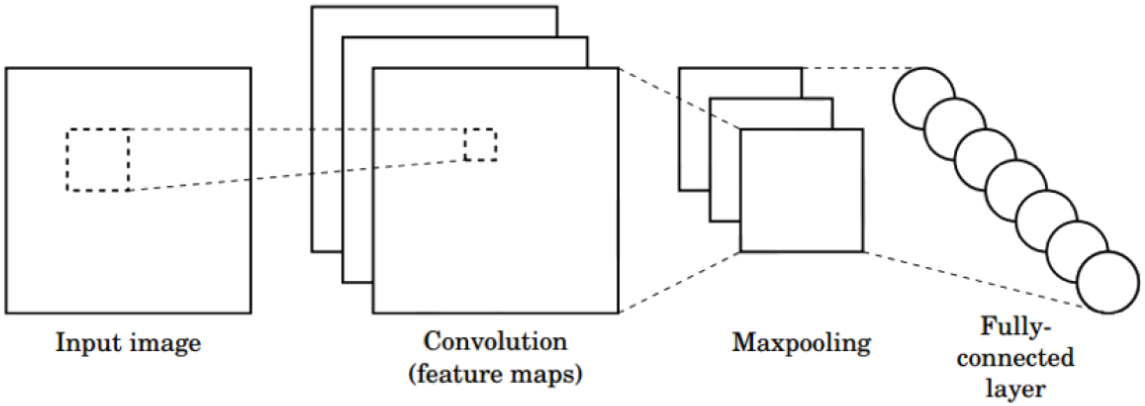
Convolutional operation

We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs:



CNN for classification

CNNs for Classification



1. **Convolution**: Apply filters to generate feature maps.
2. **Non-linearity**: Often ReLU.
3. **Pooling**: Downsampling operation on each feature map.

`tf.keras.layers.Conv2D`

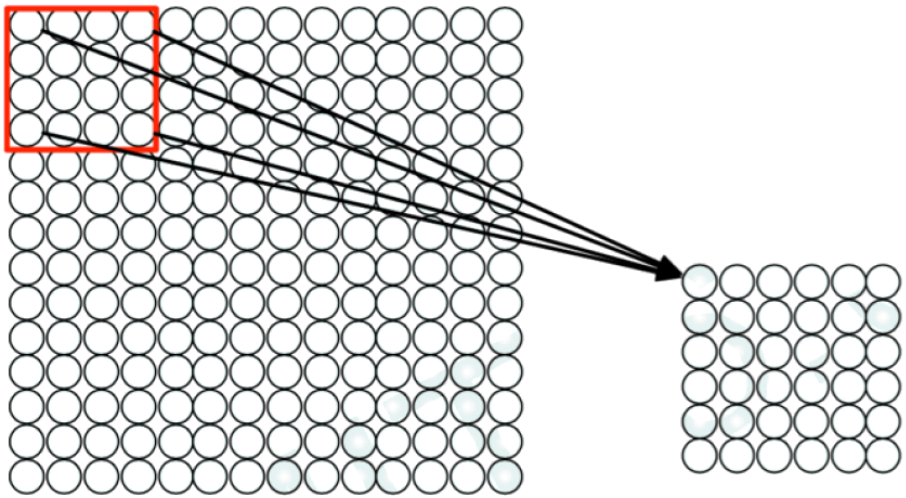
`tf.keras.activations.*`

`tf.keras.layers.MaxPool2D`

Train model with image data.
Learn weights of filters in convolutional layers.

Local connectivity

Convolutional Layers: Local Connectivity



`tf.keras.layers.Conv2D`

For a neuron in hidden layer:

- Take inputs from patch
- Compute weighted sum
- Apply bias

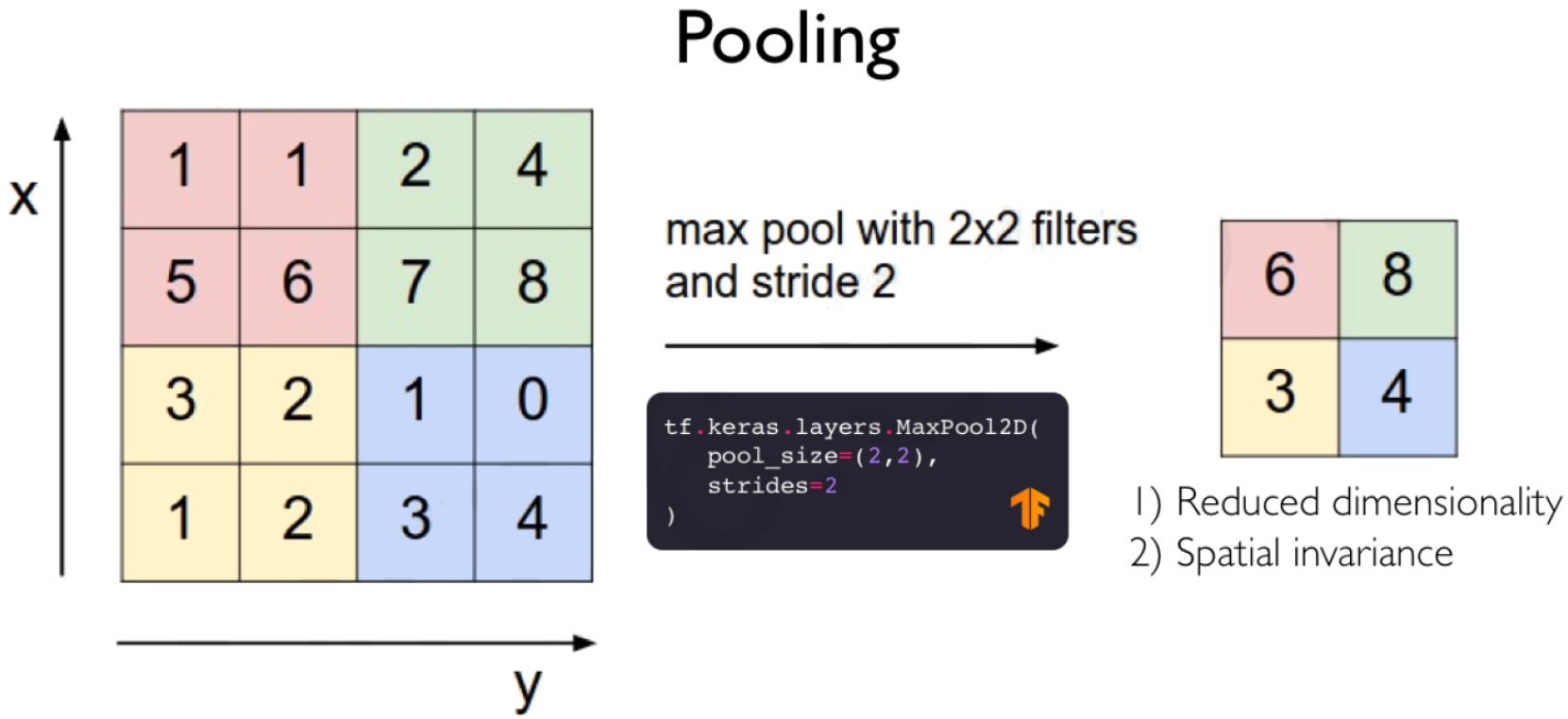
4x4 filter: matrix of weights w_{ij}

$$\sum_{i=1}^{4}\sum_{j=1}^{4} w_{ij}\, x_{i+p,j+q} + b$$

for neuron (p,q) in hidden layer

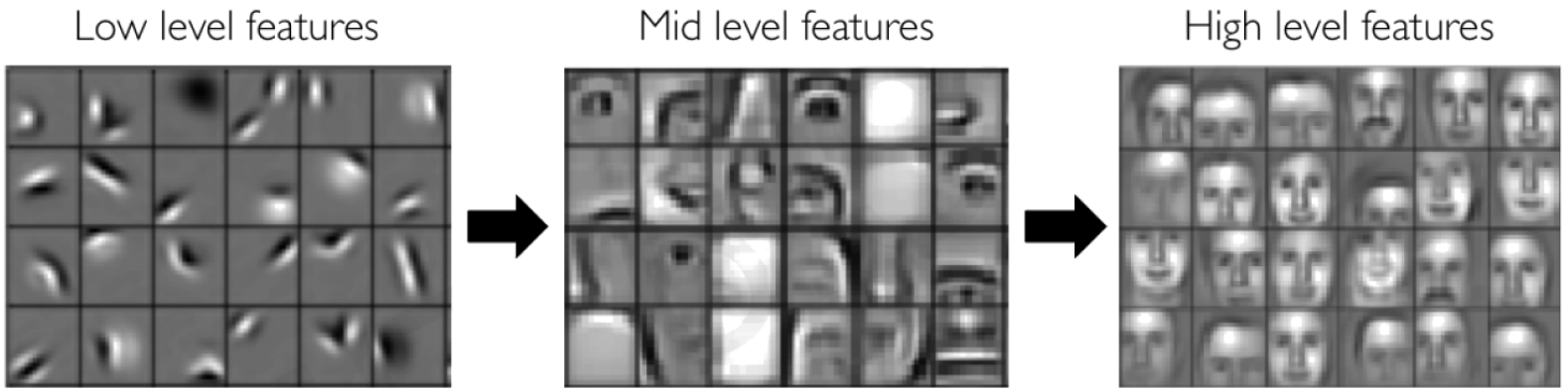
1) applying a window of weights
2) computing linear combinations
3) activating with non-linear function

Pooling



How else can we downsample and preserve spatial invariance?

Representation Learning in Deep CNNs



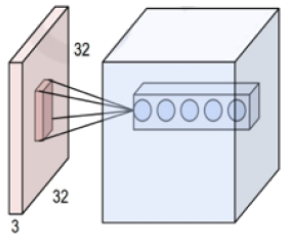
Summary on CNNs

Foundations

- Why computer vision?
- Representing images
- Convolutions for feature extraction

CNNs

- CNN architecture
- Application to classification
- ImageNet

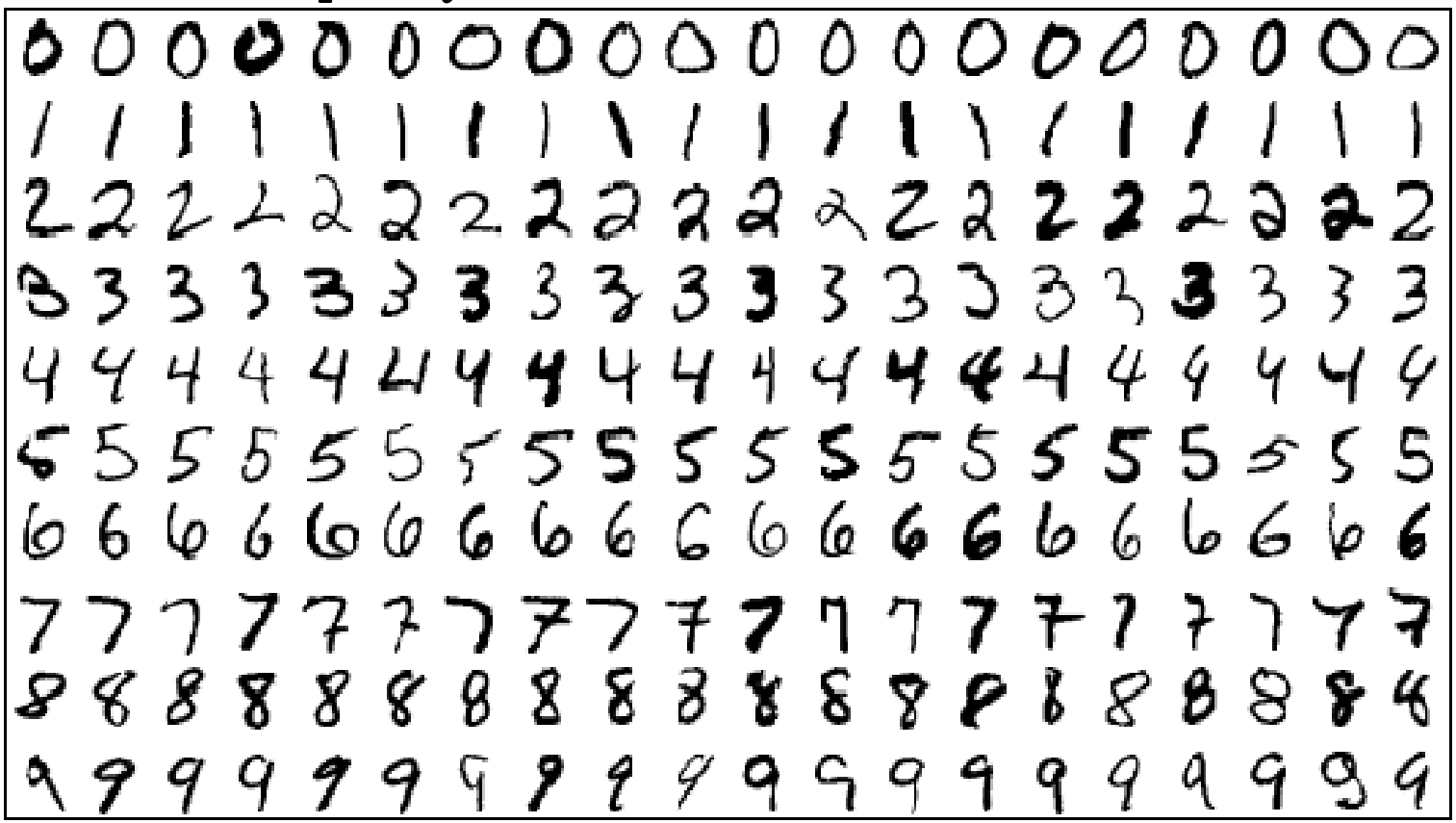


Applications

- Segmentation, image captioning, control
- Security, medicine, robotics

Images datasets: MNIST

http://yann.lecun.com/exdb/mnist/



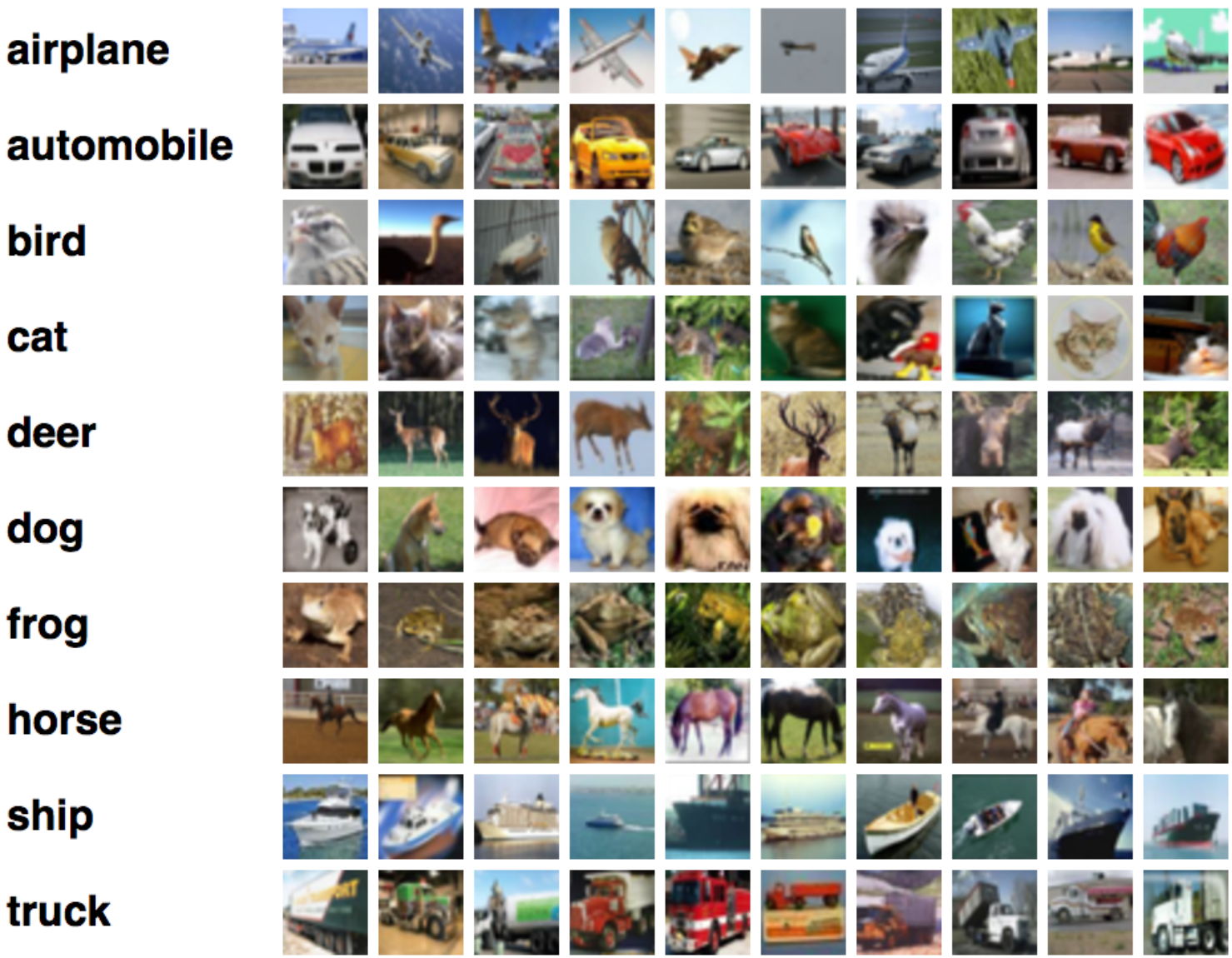
Images datasets: Fashion-MNIST

https://research.zalando.com/welcome/mission/research-projects/fashion-mnist/

Label	Description	Examples
0	T-Shirt/Top	
1	Trouser	
2	Pullover	
3	Dress	
4	Coat	
5	Sandals	
6	Shirt	
7	Sneaker	
8	Bag	

Images datasets: CIFAR

https://www.cs.toronto.edu/~kriz/cifar.html



Images datasets: IMAGENET

http://www.image-net.org/