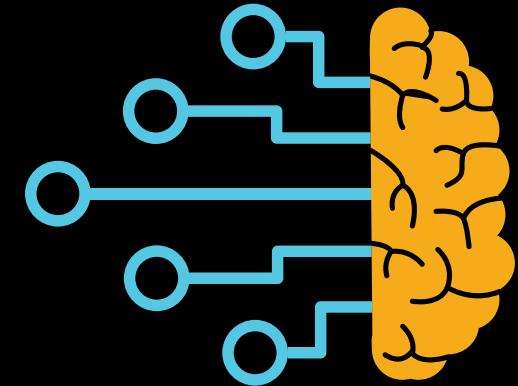


DCAO, UBA
08-12 AGOSTO 2022

APRENDIZAJE AUTOMÁTICO.

Fundamentos y Aplicaciones en
Meteorología del Espacio



Dra María Graciela Molina

FACET-UNT / CONICET
Tucumán Space Weather Center - TSWC

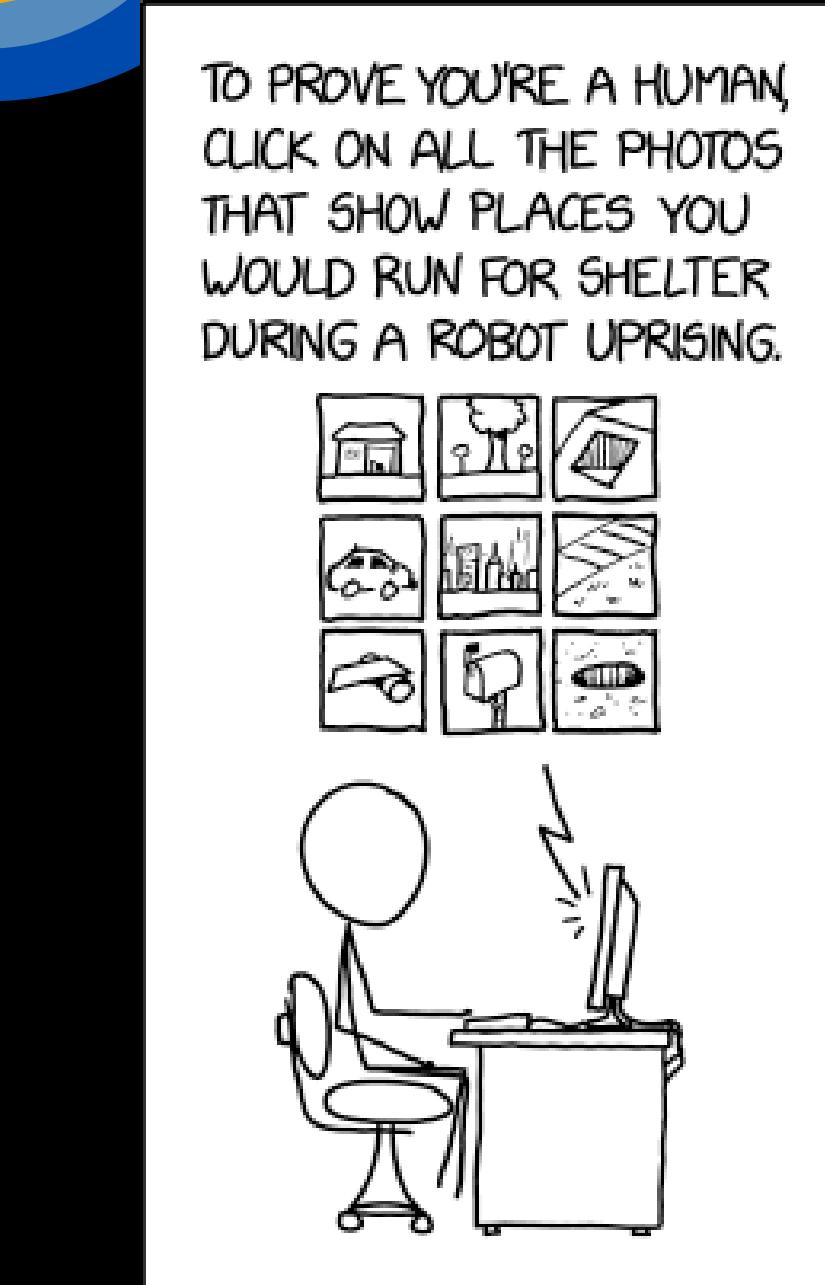
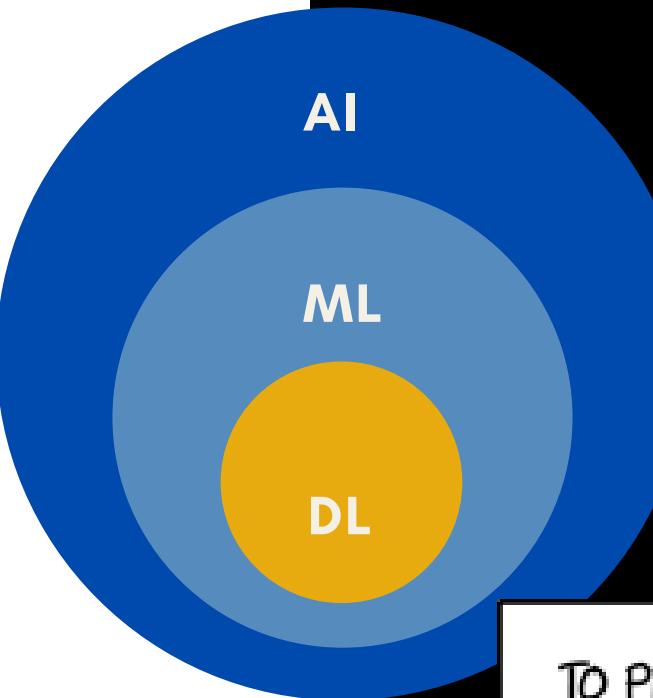
<https://spaceweather.facet.unt.edu.ar/>
IG -> @spaceweatherargentina

gmolina@herrera.unt.edu.ar

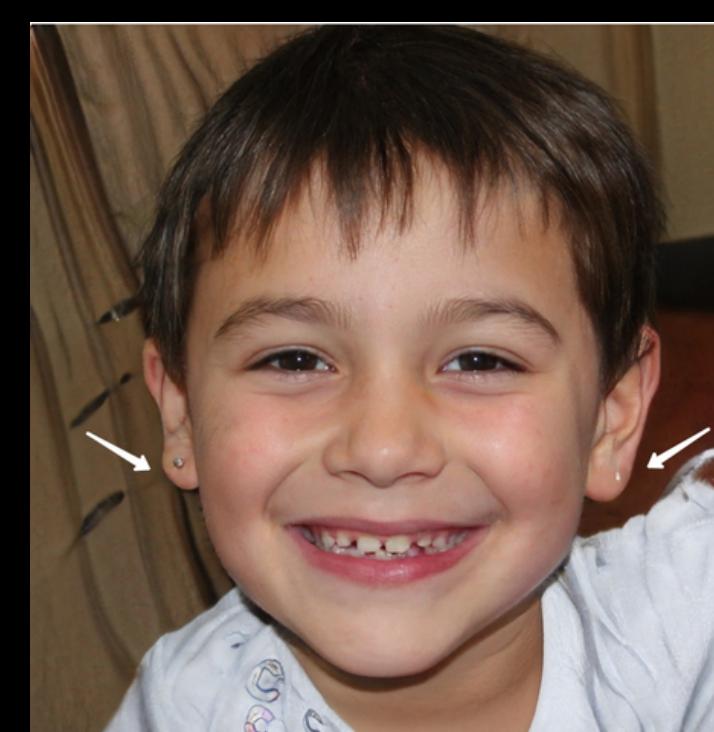


Deep Learning

Extract patterns from data
using neural networks



- <https://this-person-does-not-exist.com/en>

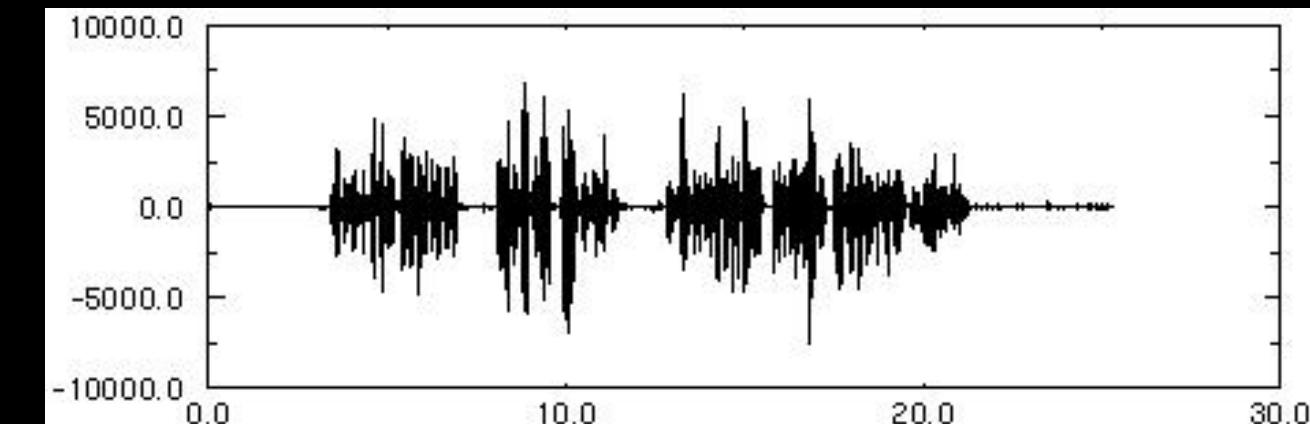
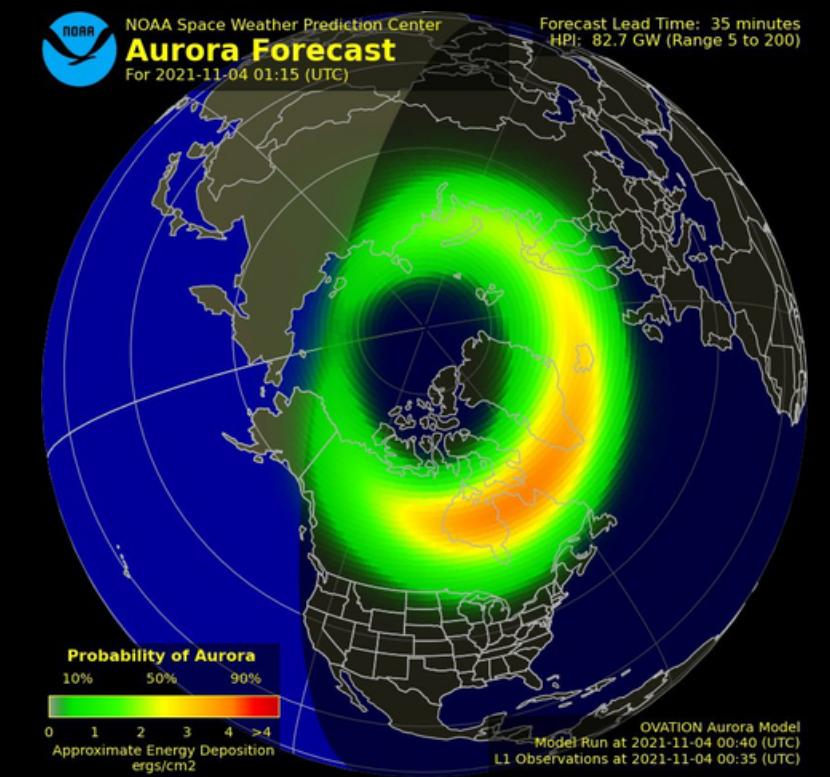
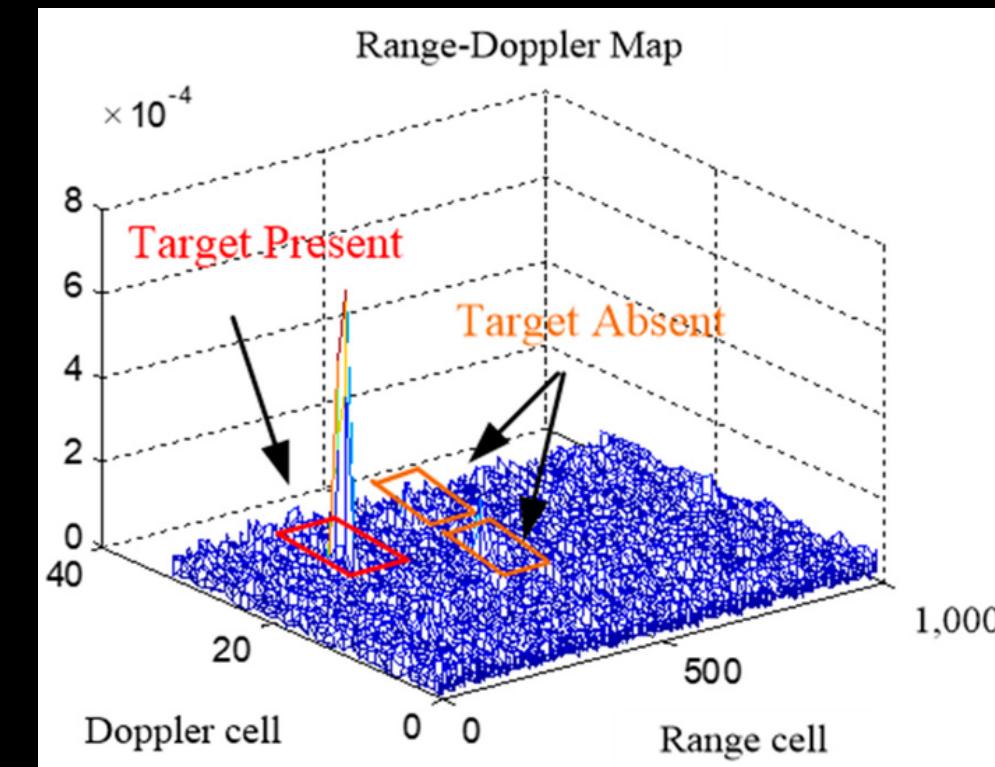


TSWC, 2022



Recurrent Neural Networks

- Processing a sequence of data $x(t) = x(1), \dots, x(\tau)$
- Recurrent -> perform the same task for every element of a sequence, with the output being depended on the previous computations.
- Have a “memory”



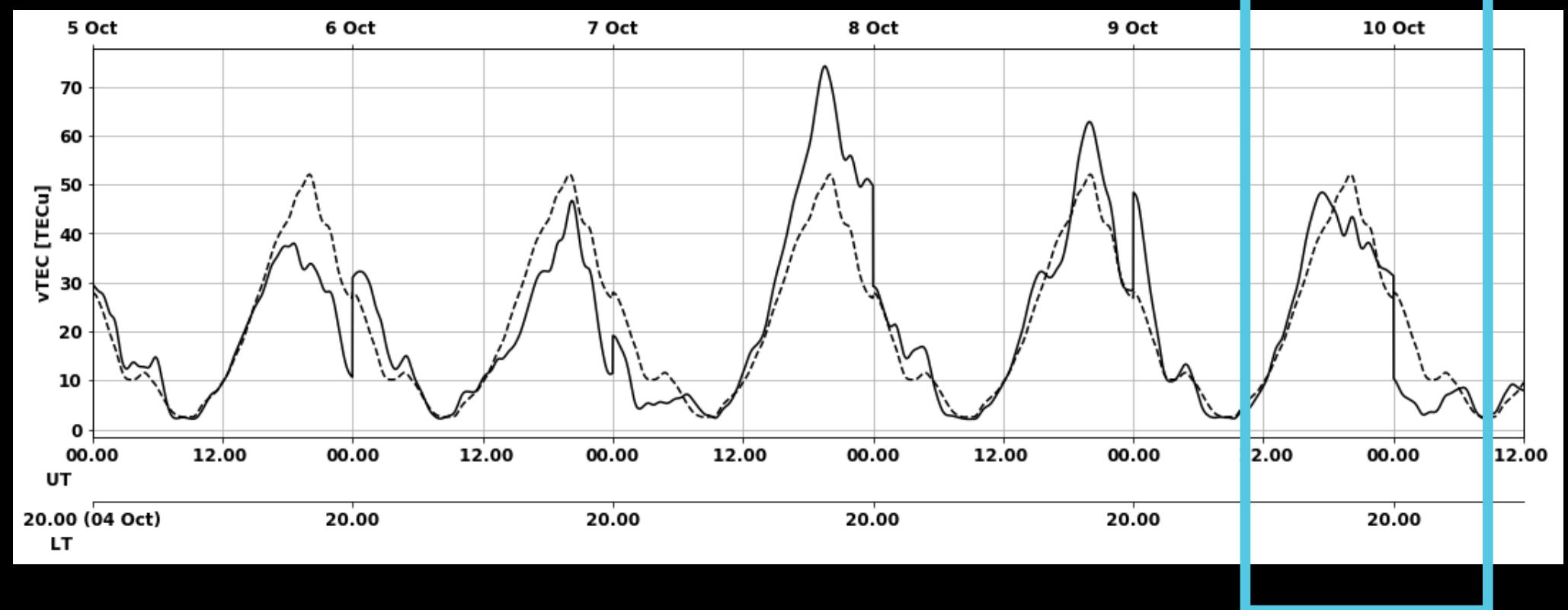
Recurrent Neural Networks

RNNs as an approach to sequence modeling problems

We want to forecast this

To model sequences, we need to:

- Handle **variable-length** sequences
- Track **long-term** dependences
- Maintain information about **order**
- **Share parameters** across the sequence



TSWC, 2022

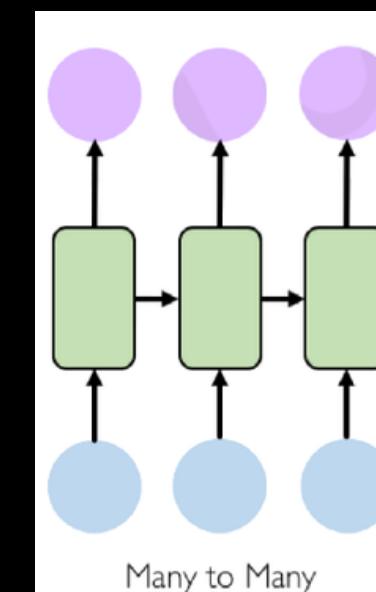
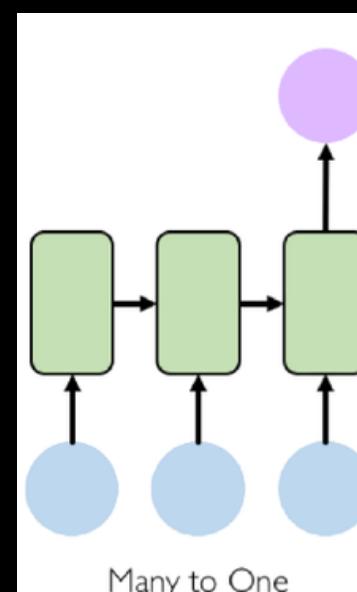
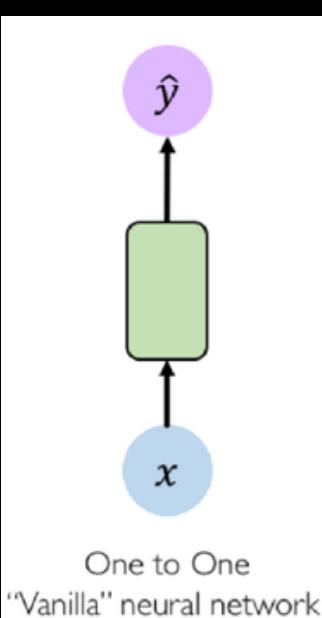
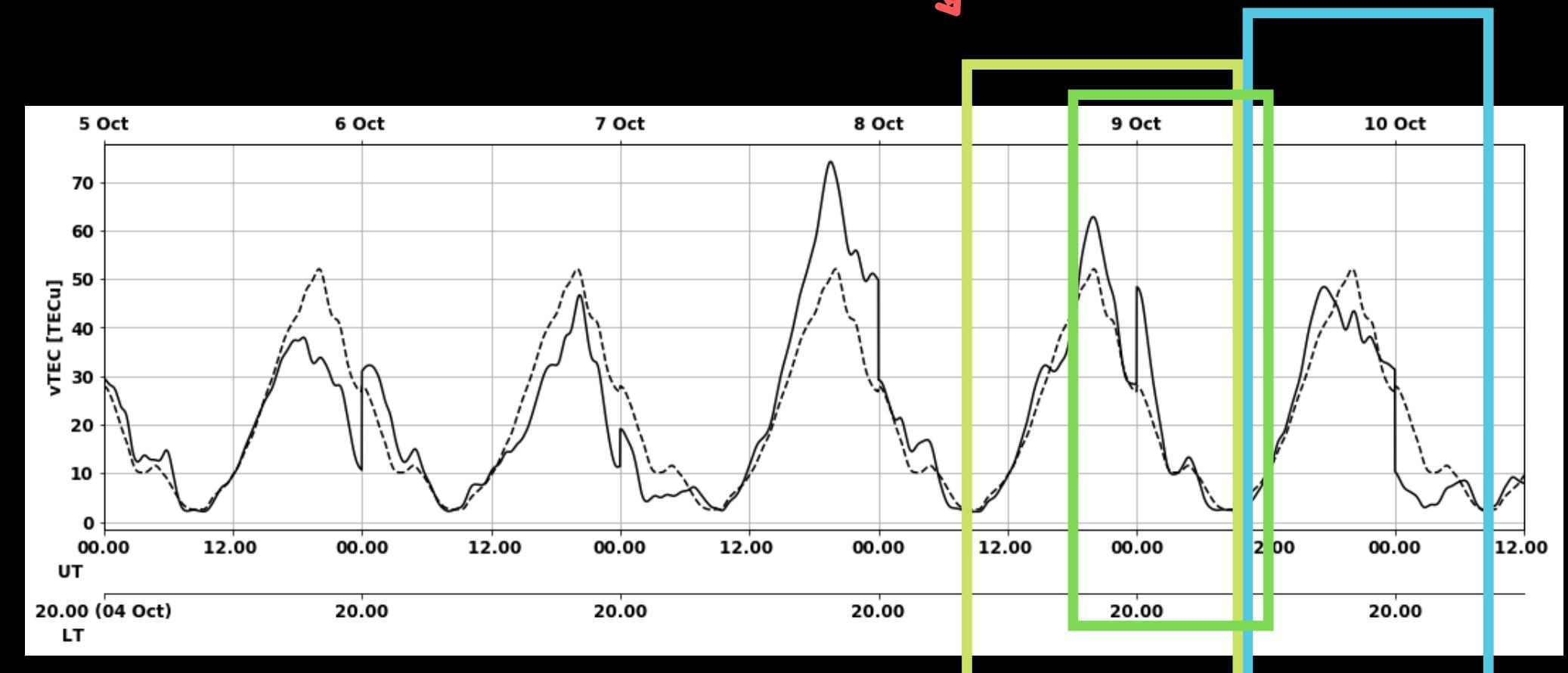
Recurrent Neural Networks

RNNs as an approach to sequence modeling problems

To model sequences, we need to:

- Handle **variable-length** sequences
- Track **long-term** dependences
- Maintain information about **order**
- **Share parameters** across the sequence

how many steps? (length)



and more architectures



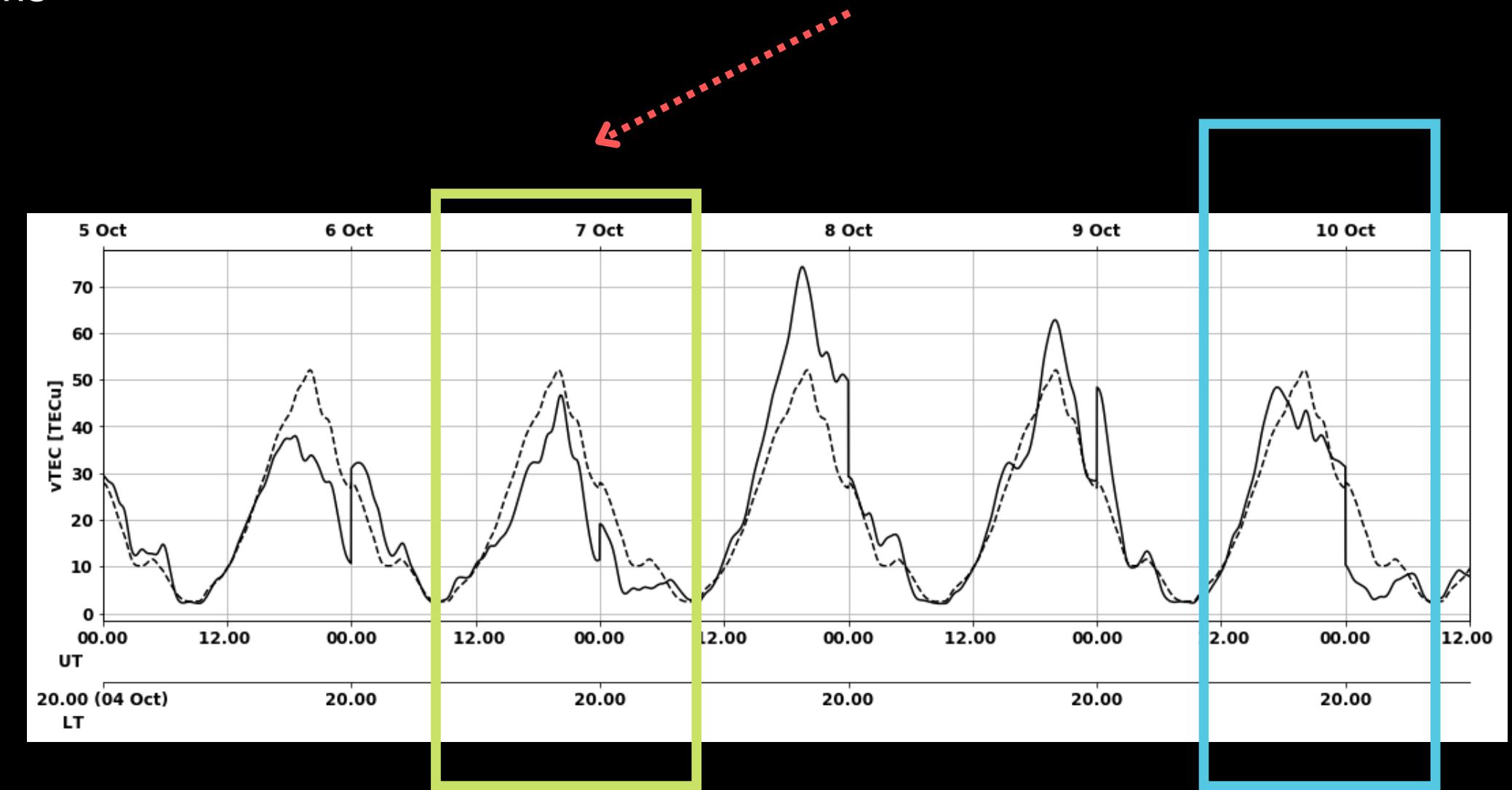
Recurrent Neural Networks

RNNs as an approach to sequence modeling problems

how important is the information on
the (far) past ?

To model sequences, we need to:

- Handle **variable-length** sequences
- Track **long-term** dependences
- Maintain information about **order**
- **Share parameters** across the sequence



E.g. Ionosphere regular behaviour
(daily, seasonal, solar cycle, ...)



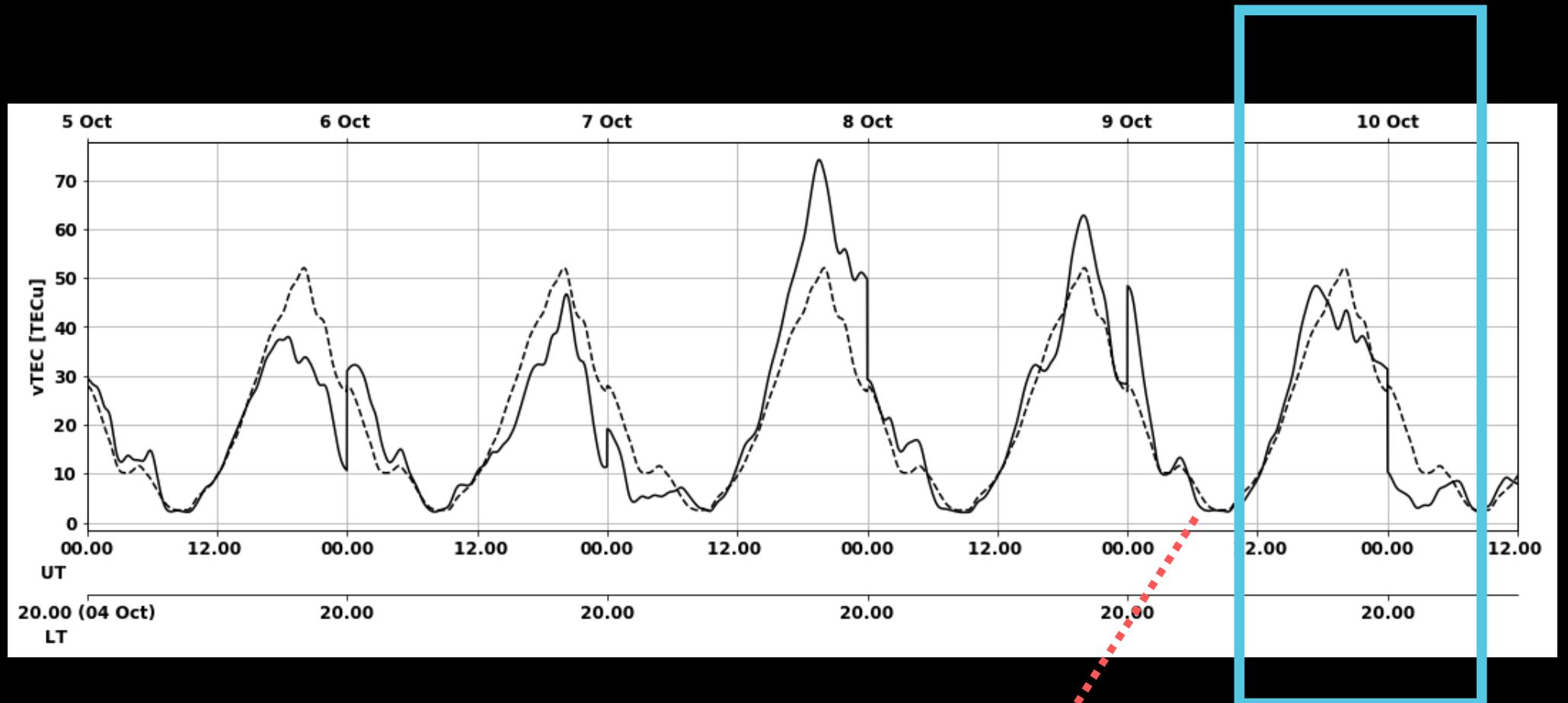
TSWC, 2022

Recurrent Neural Networks

RNNs as an approach to sequence modeling problems

To model sequences, we need to:

- Handle **variable-length** sequences
- Track **long-term** dependences
- Maintain information about **order**
- **Share parameters** across the sequence



$vTEC(t-2), vTEC(t-1), vTEC(t-0) \leftrightarrow vTEC(t-0), vTEC(t-2), vTEC(t-1)$



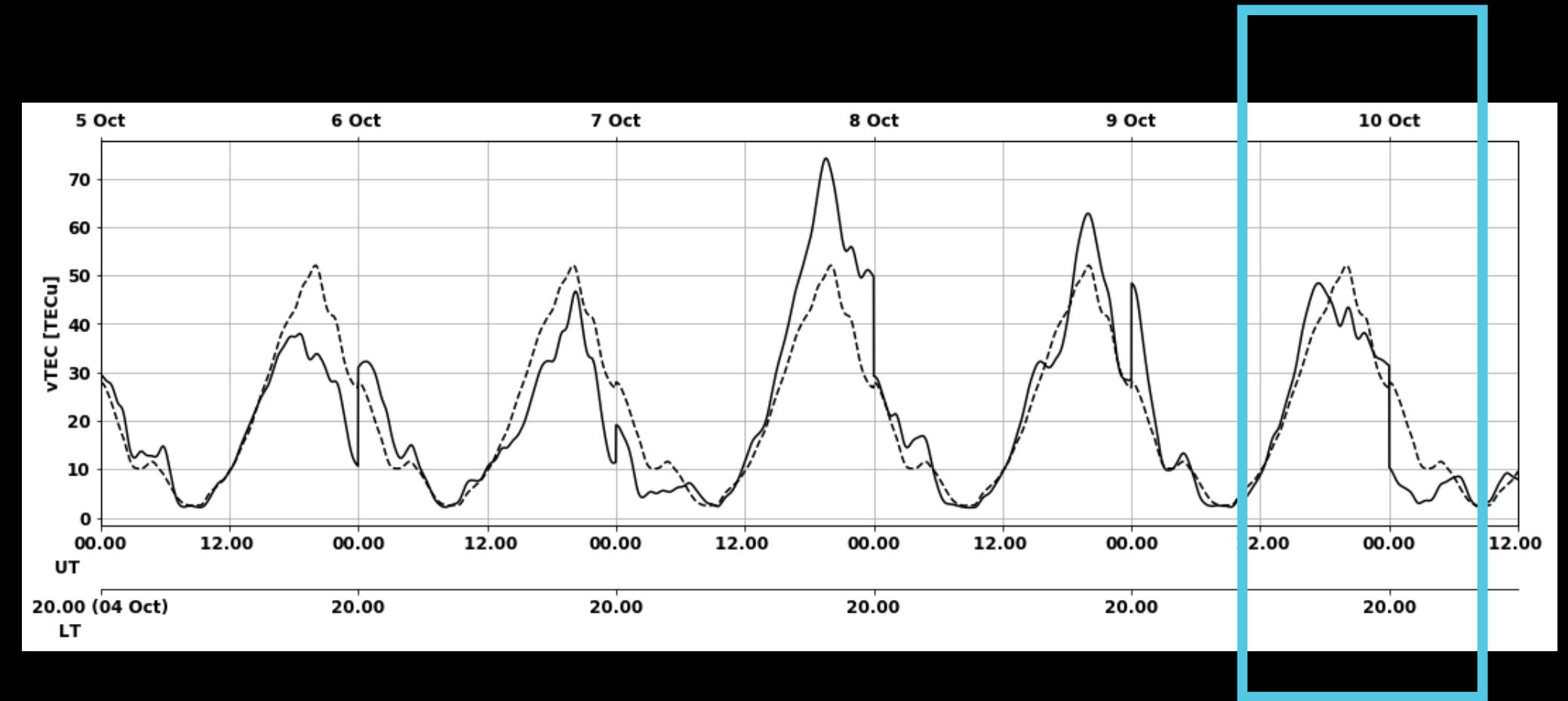
TSWC, 2022

Recurrent Neural Networks

RNNs as an approach to sequence modeling problems

To model sequences, we need to:

- Handle **variable-length** sequences
- Track **long-term** dependences
- Maintain information about **order**
- **Share parameters** across the sequence



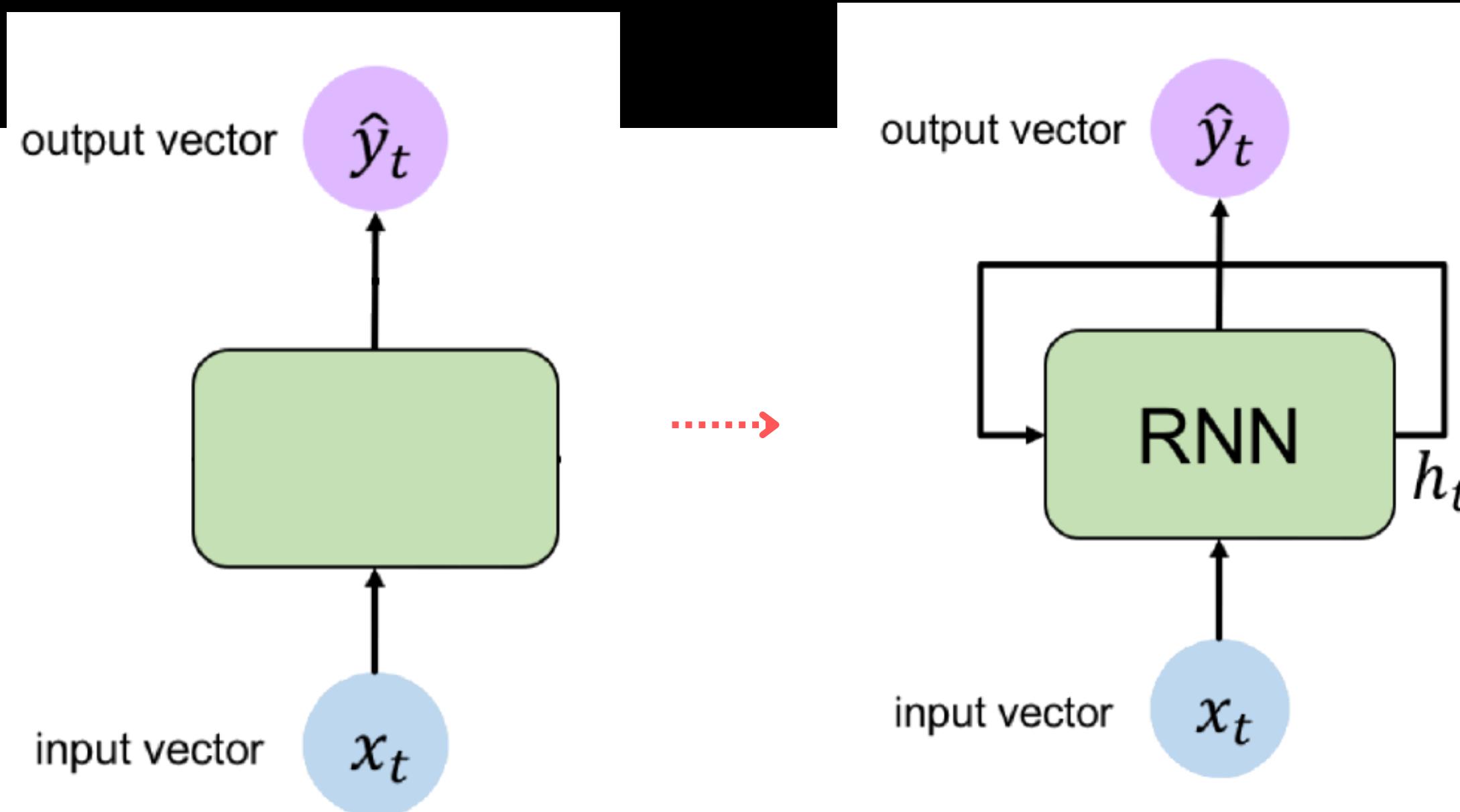
RNNs have a **state** (h_t), that is **updated at each time** as a sequence is processed using the **same parameters** each time step



TSWC, 2022



Recurrent Neural Networks



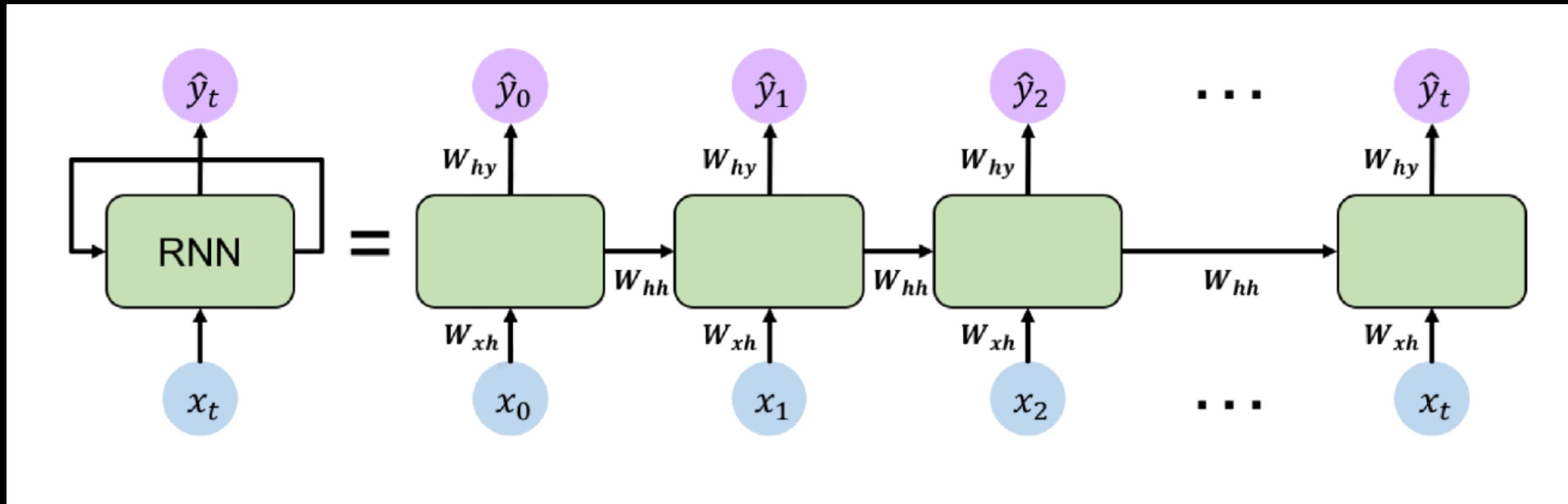
$$h_t = f_W(h_{t-1}, x_t)$$

cell state function parameterized by W old state input vector at time step t

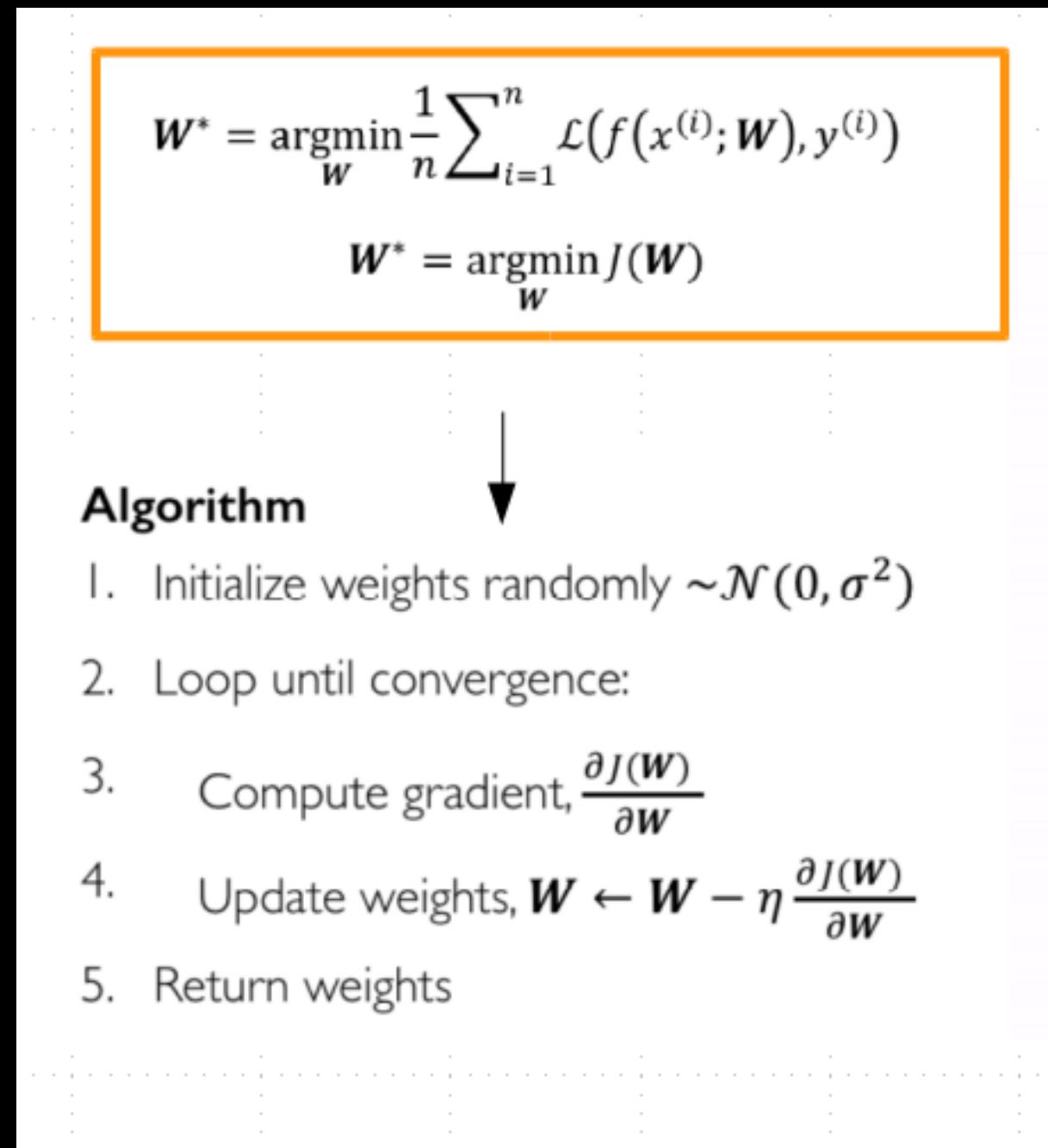
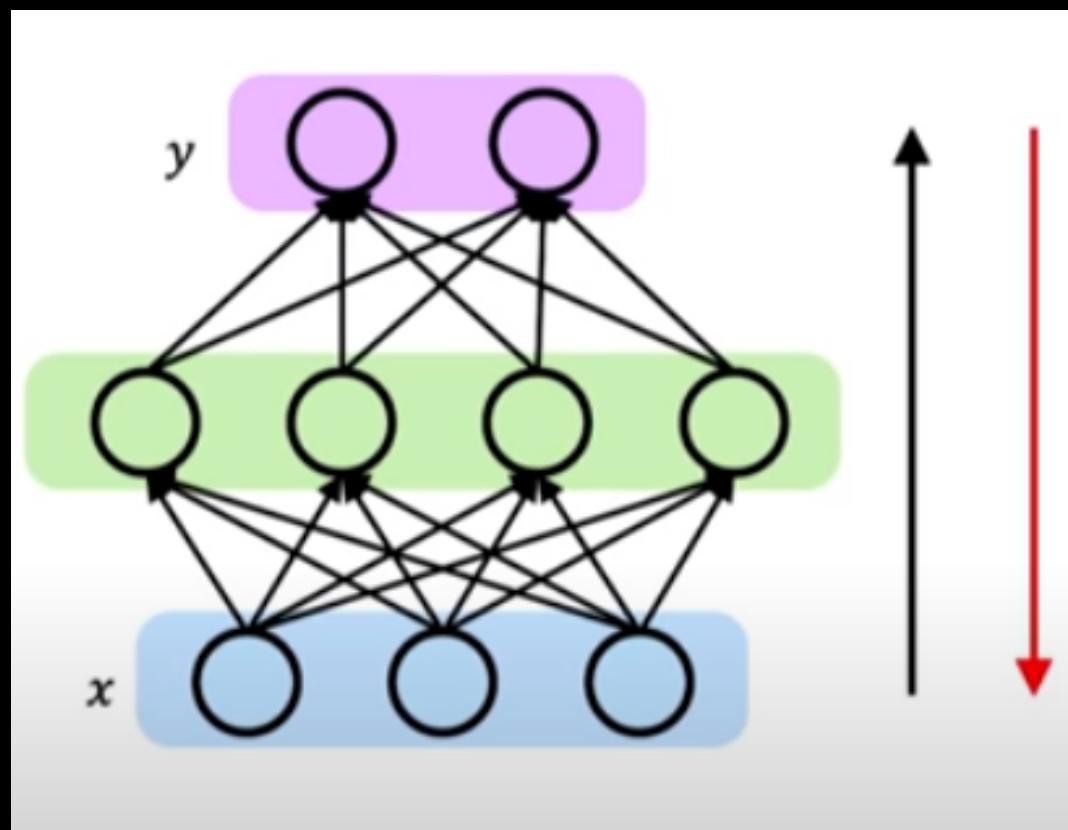
Apply a **recurrent relation** at every time step to process a sequence



Recurrent Neural Networks



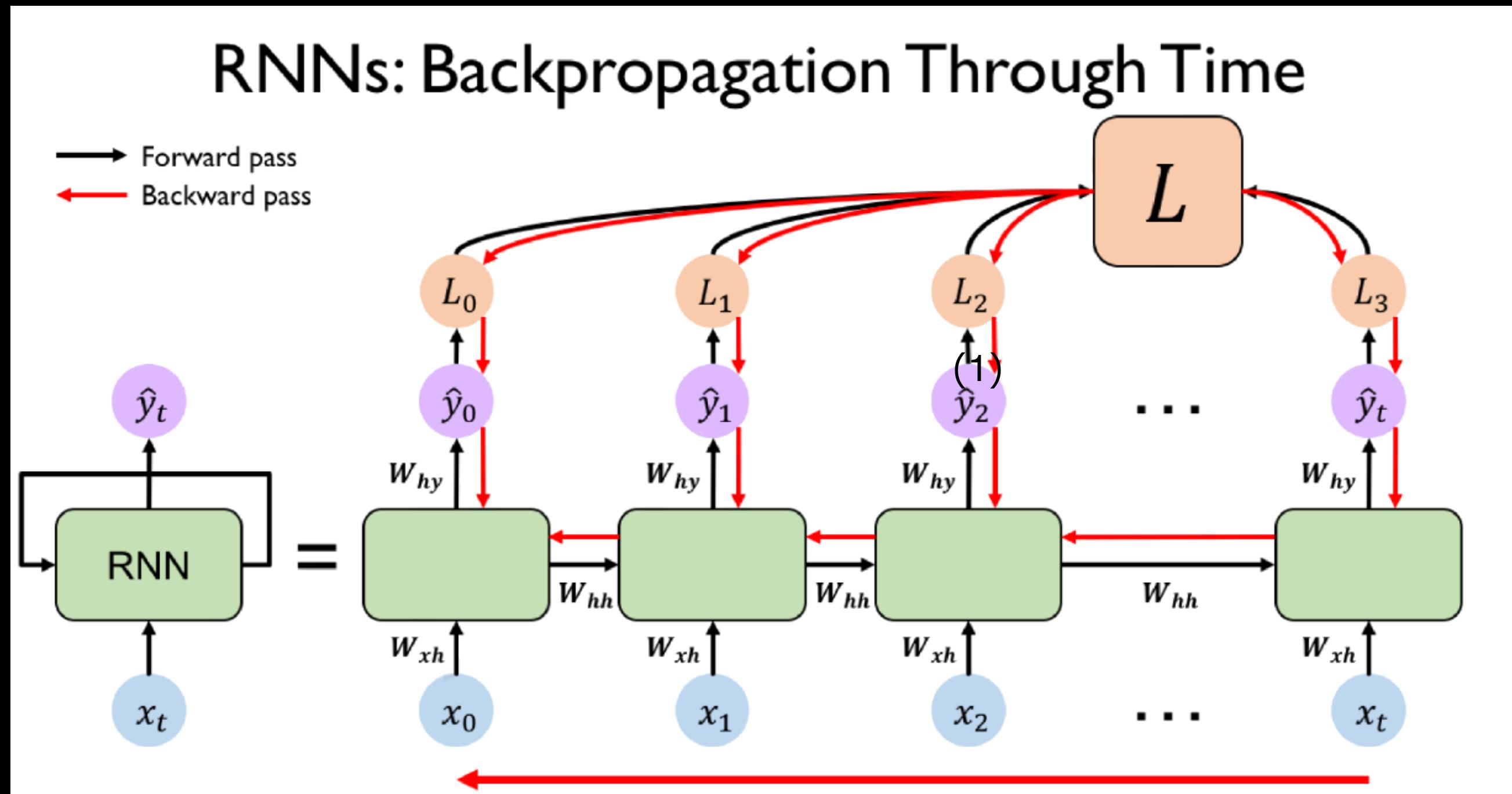
Backpropagation through time: long time dependences



- take the gradient of loss with respect to each parameter
- shift parameters in order to minimize loss



Backpropagation through time: long time dependences



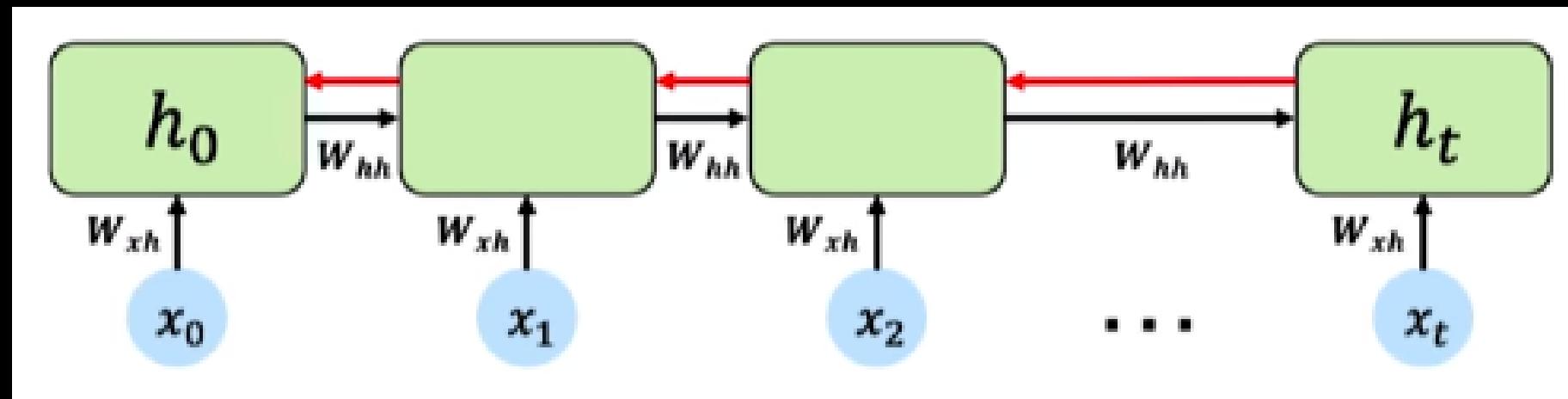
- Compute individual L_i for individual time steps and sum them
- Backpropagate errors individually for each time step and then to all the time steps to the beginning of the sequence.

<https://kharshit.github.io/blog/2019/02/22/backpropagation-through-time>



TSWC, 2022

Backpropagation through time: long time dependences



- high computation time!

- Many values $\gg 1$ -> exploding gradient (*)
- Many values $\ll 1$ -> vanishing gradient

(*) Gradient clipping is a simple technique: If the gradient gets too large, we rescale it to keep it small.



Vanishing gradient 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 tackle the problem:

- Activation function
- Weight initialization
- Network architecture





Vanishing gradient problem



Multiply many small numbers together



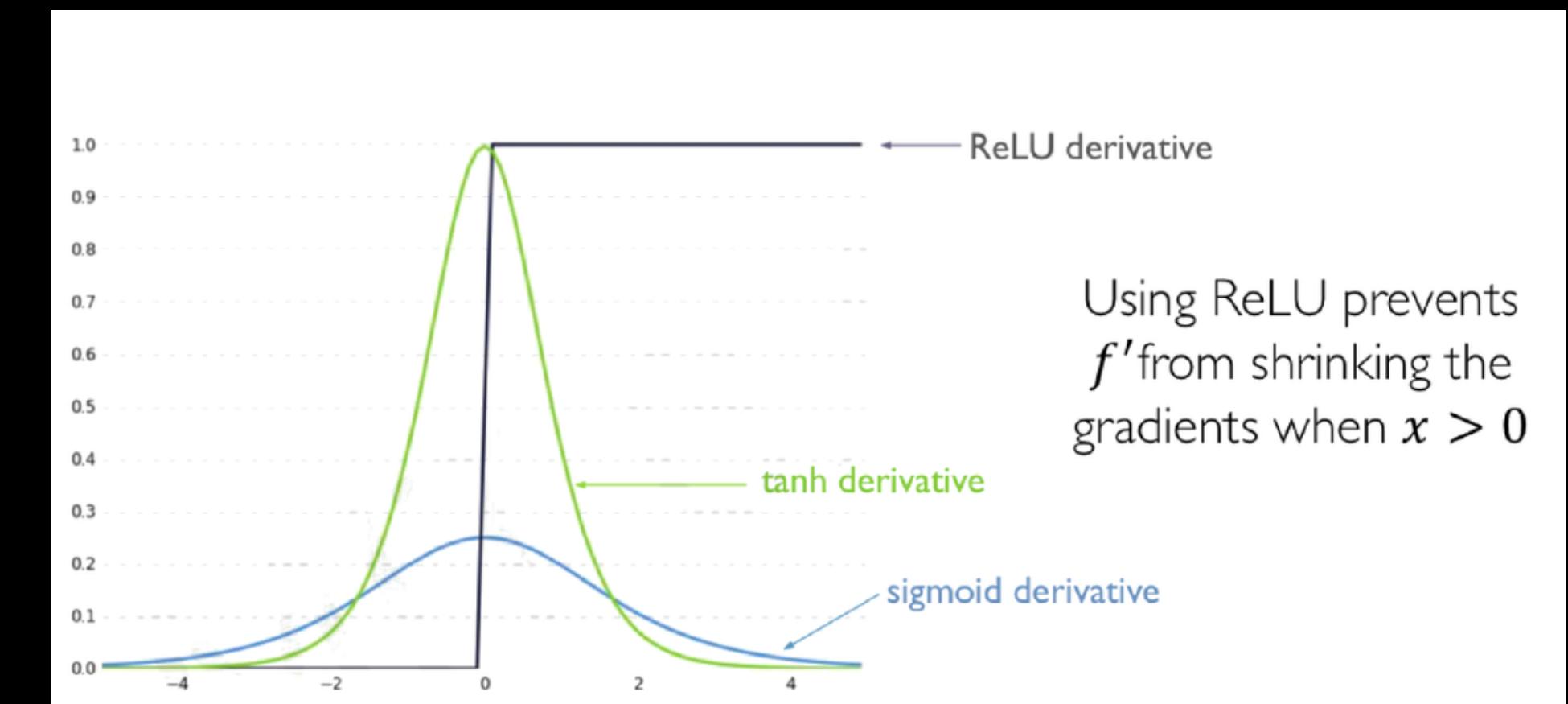
Errors due to further back time steps
have smaller and smaller gradients



Bias parameters to capture short -term
dependencies

How to tackle the problem:

- Activation function
- Weight initialization
- Network architecture





Vanishing gradient 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 (we "lose" long-term
dependencies)

How to tackle the problem:

- Activation function
- **Weight initialization**
- Network architecture

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

Initialize **weights** to identity matrix
Initialize **biases** to zero

prevent the weights to shrinking to
zero



Vanishing gradient problem



Multiply **many small** numbers together



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

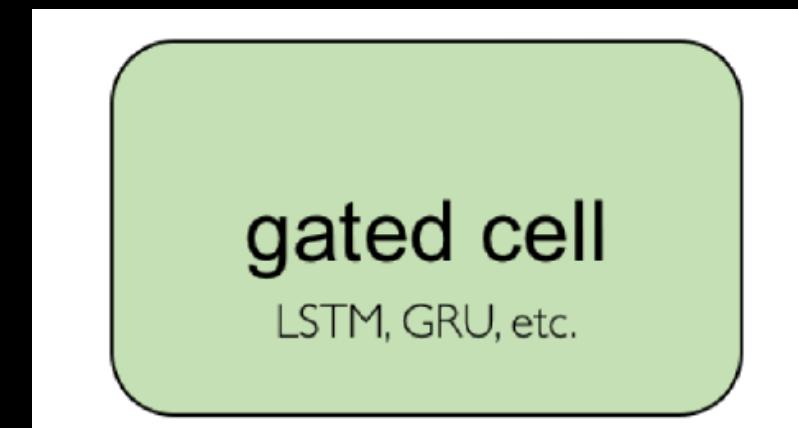


Bias parameters to capture short -term
dependencies (we "lose" long-term
dependencies)

How to tackle the problem:

- Activation function
- Weight initialization
- **Network architecture**

more robust solution



- Use a more **complex recurrent unit with gates** to control what information is passed through.
- gates selectively **add or remove** information within each recurrent unit



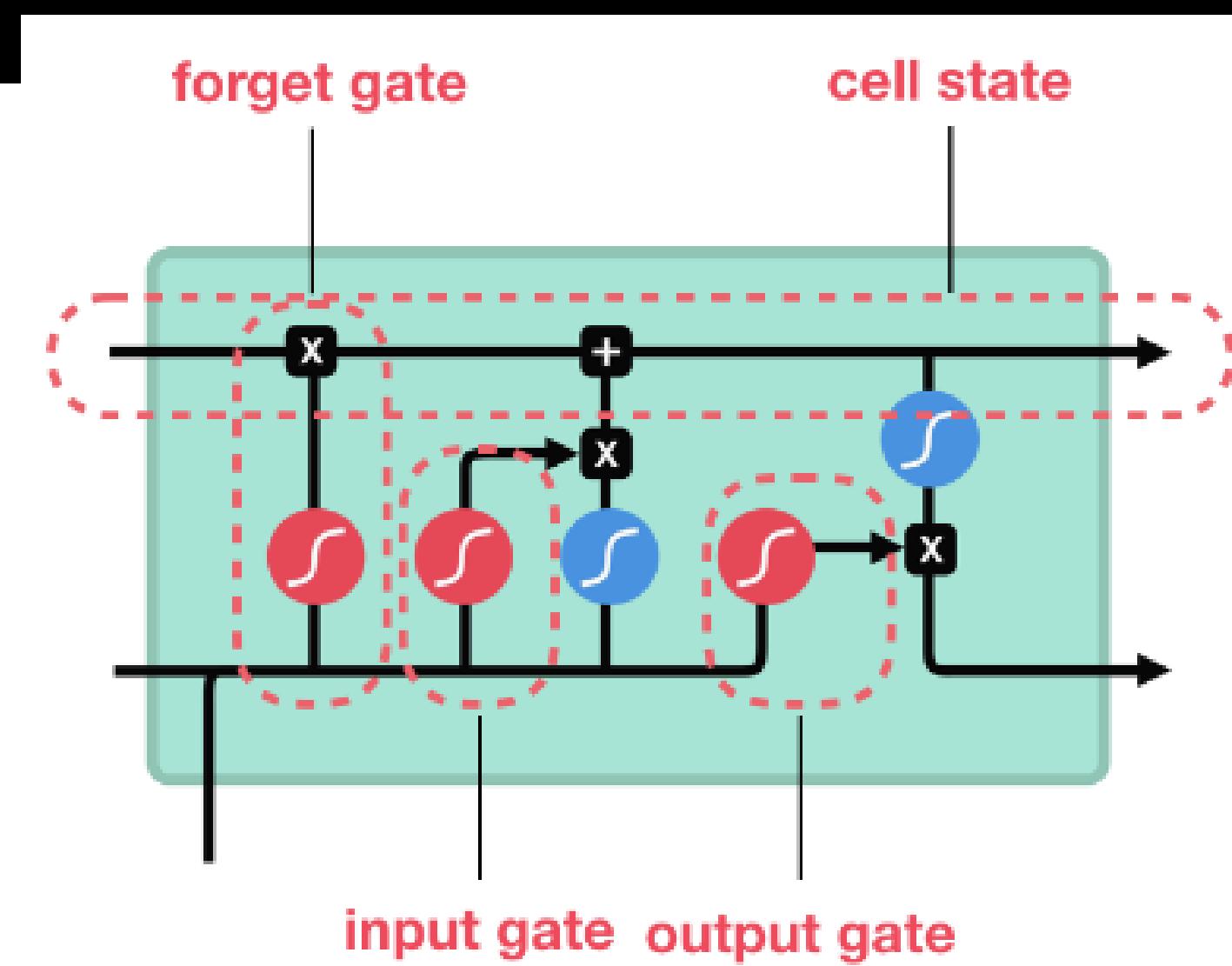
Long short term memory (LSTM)

Gates: 1) Forget

2) Input (store)

3)Update

4) Output



How it works:

- 1) Maintain a **cell state**
- 2) Use gates to control the flow of information
 - **Forget** gate gets rid of irrelevant information
 - Store relevant information from the current **input**
 - Selectively **update** cell state
- 3) Backpropagation T_T with partially uninterrupted gradient flow



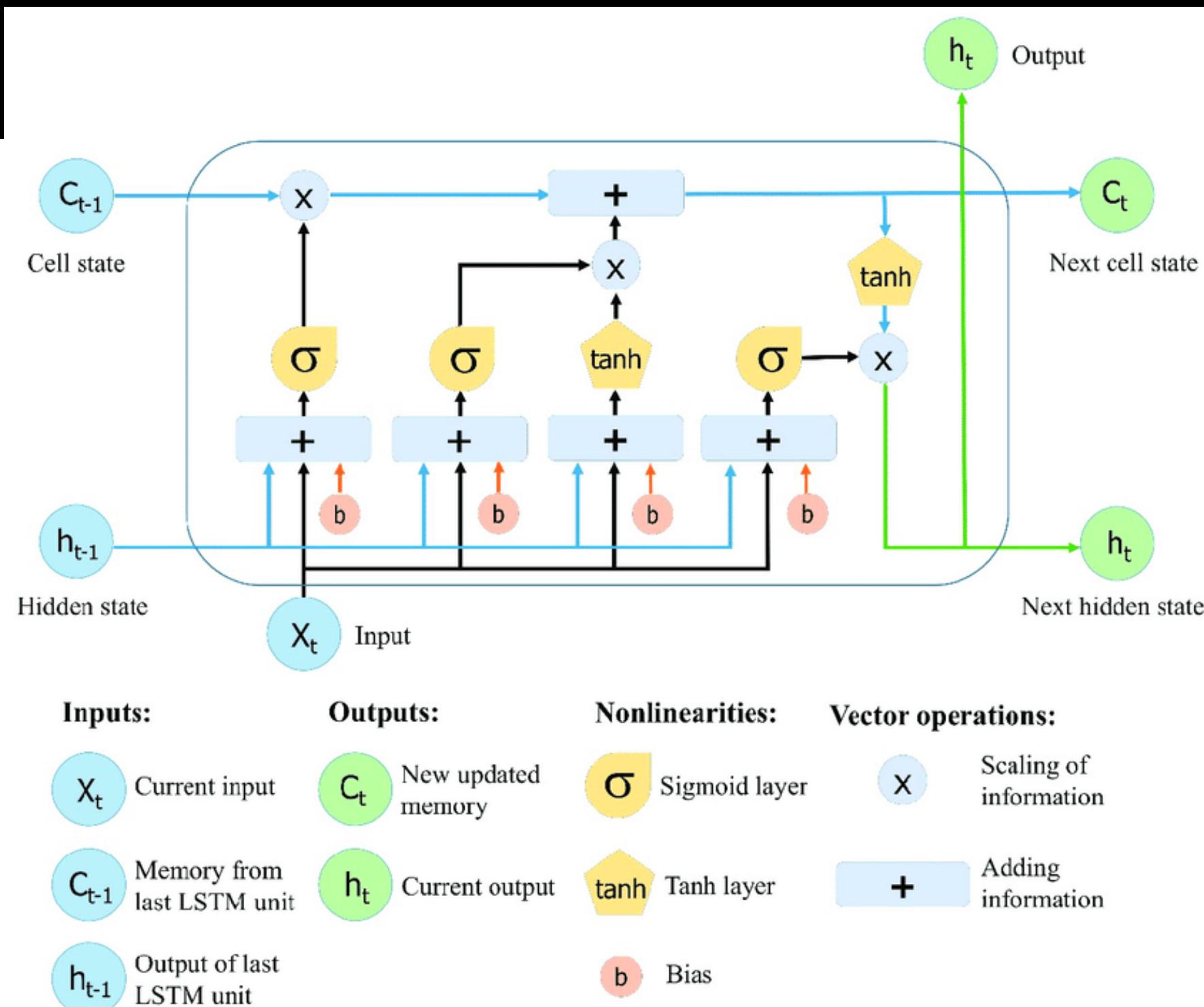
Long short term memory (LSTM)

Gates: 1) Forget

2) Input (store)

3) Update

4) Output



How it works:

- 1) Maintain a **cell state**
- 2) Use gates to control the flow of information
 - **Forget** gate gets rid of irrelevant information
 - Store relevant information from the current **input**
 - Selectively **update** cell state
- 3) Backpropagation TT with partially uninterrupted gradient flow



LSTM "not all that glitters is gold"

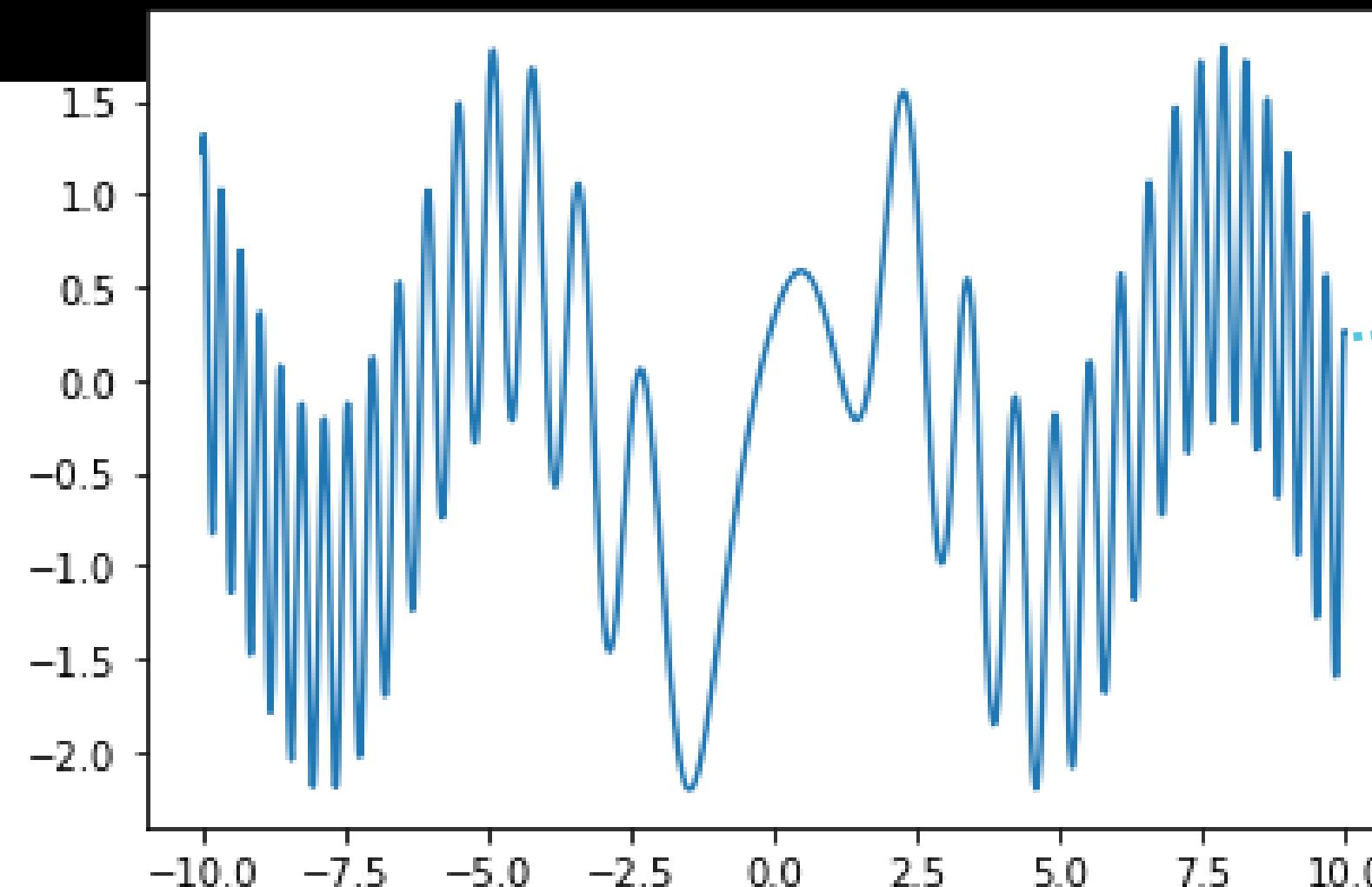
- << vanishing gradient problem but it doesn't completely remove it.
- >> computational resources
- affected by different random weight initialization
- Drop-out difficult to implement
- Prone to overfitting
- << performance for very long time problems



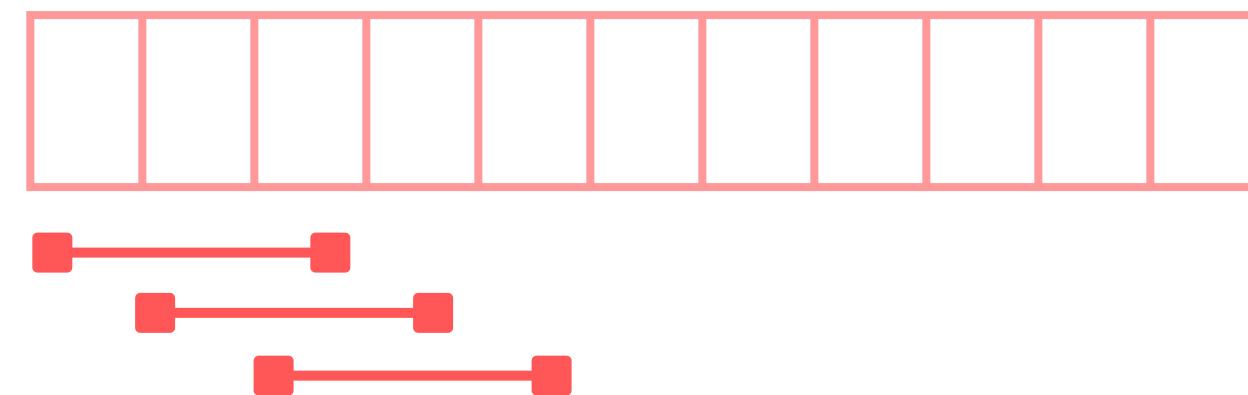


LSTM example

vanilla LSTM



training set $t \in [-10, 10)$
test set $t=10$



the training set is an array of
sequences (len = 3)



Convolutional Neural Networks

Received July 15, 2019, accepted July 30, 2019, date of publication August 5, 2019, date of current version August 19, 2019.
Digital Object Identifier 10.1109/ACCESS.2019.2933060

Automated Individual Pig Localisation, Tracking and Behaviour Metric Extraction Using Deep Learning

JAKE COWTON^{①,2}, ILIAS KYRIAZAKIS², AND JAUME BACARDIT^①

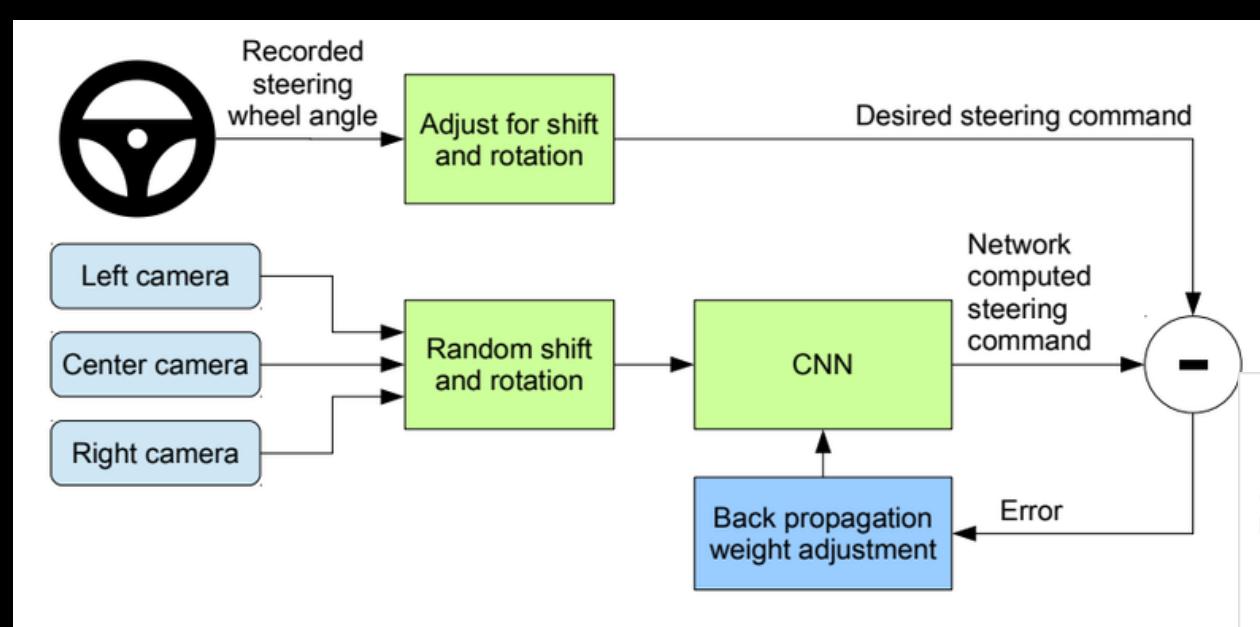
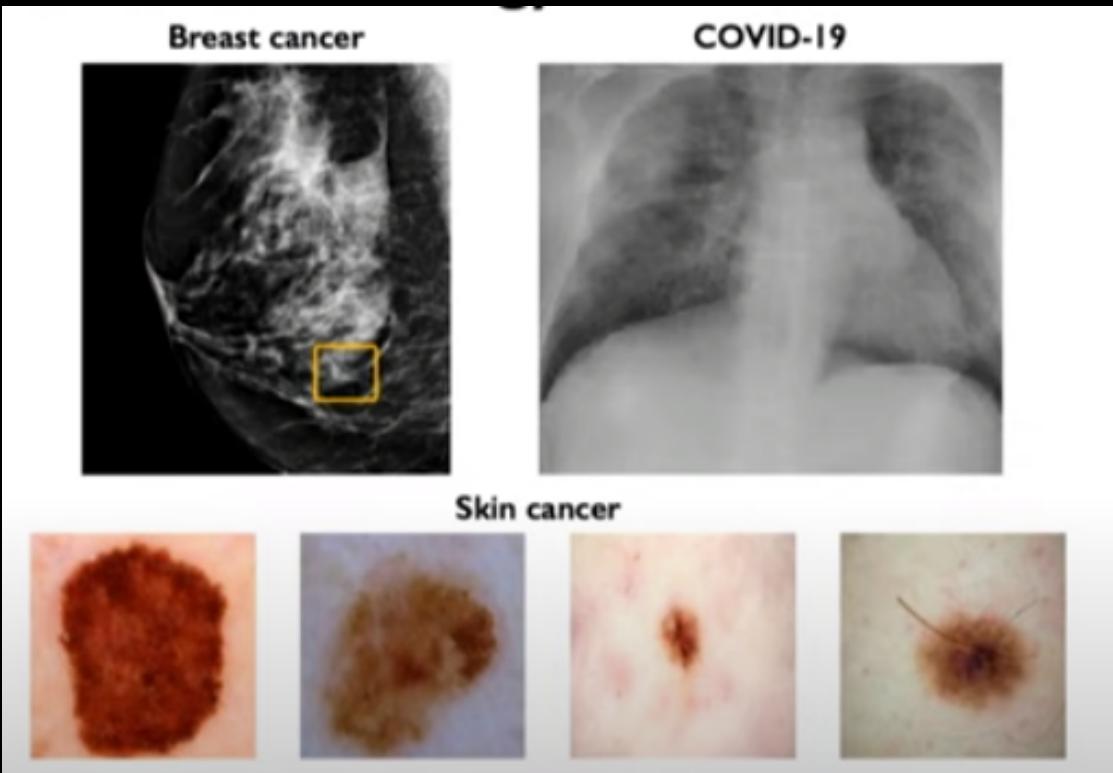
¹School of Computing, Newcastle University, Newcastle upon Tyne NE1 7RU, U.K.
²School of Natural and Environmental Sciences, Agriculture, Newcastle University, Newcastle upon Tyne NE1 7RU, U.K.

Corresponding author: Jake Cowton (j.cowton2@newcastle.ac.uk)

This work was supported by the European Commission under the European Union Framework Programme for Research and Innovation Horizon 2020 under Grant 633531. The work of J. Bacardit was supported by the Engineering and Physical Science Research Council under Grant EP/N031962/1 and Grant EP/M020576/1.



FIGURE 8. Four sample images from our pig detection test set processed by the Faster R-CNN with the feature extraction layers pre-trained on ImageNet, the rest pre-trained on Pascal Visual Object Classes Challenge 2007 and an additional fully-connected layer for the pig dataset. Detections to the left of the red wall are ignored. The top left image is from the low-light test segment. The top right image is from the densely packed test-segment. The bottom left image is from the overexposed test segment. The bottom right image is from the many pigs test segment.



arXiv

End to End Learning for Self-Driving Cars
We trained a convolutional neural network (CNN) to map raw pixels from a single front-facing camera directly to...



Convolutional Neural Networks

Computer Vision. Some applications:

- Facial detection and recognition
- Healthcare, medicine and biology
- Self-driving vehicles



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
256	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	106	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	199	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	105	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	85	150	79	38	218	241
190	234	147	108	227	210	127	102	96	101	265	224
190	214	173	66	109	143	96	50	2	109	249	215
187	196	236	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	179	13	95	218

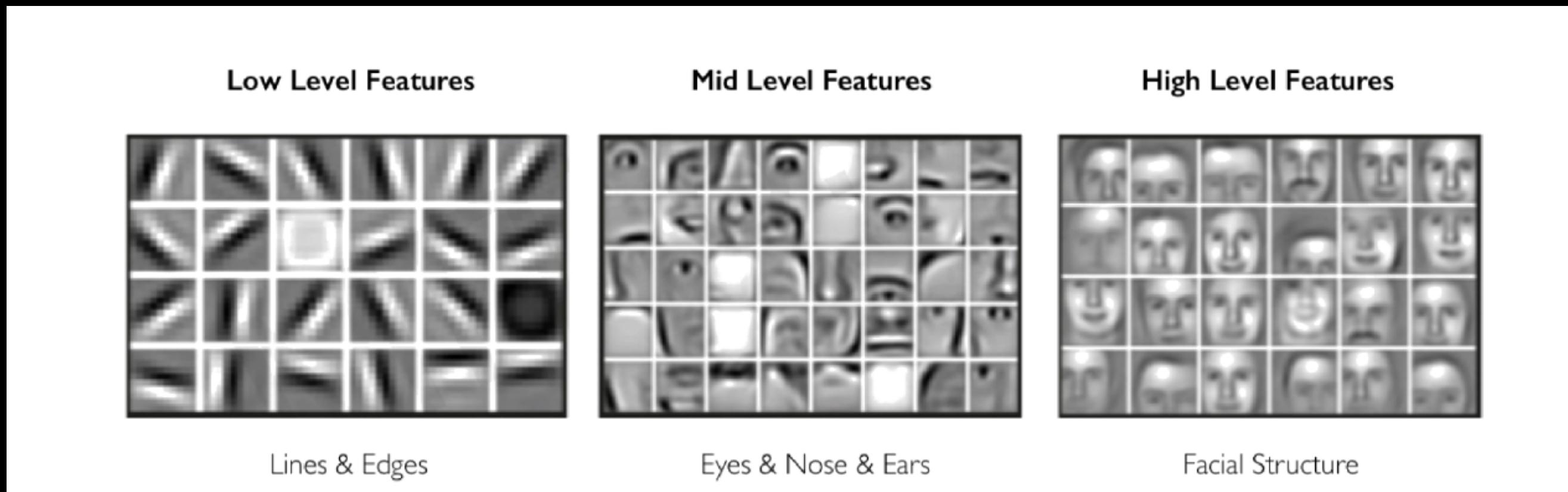
An image is just a matrix of numbers [0,255]!
i.e., 1080x1080x3 for an RGB image



Features

(As in other NN problems)

- Regression
- Classification
- High level feature detection

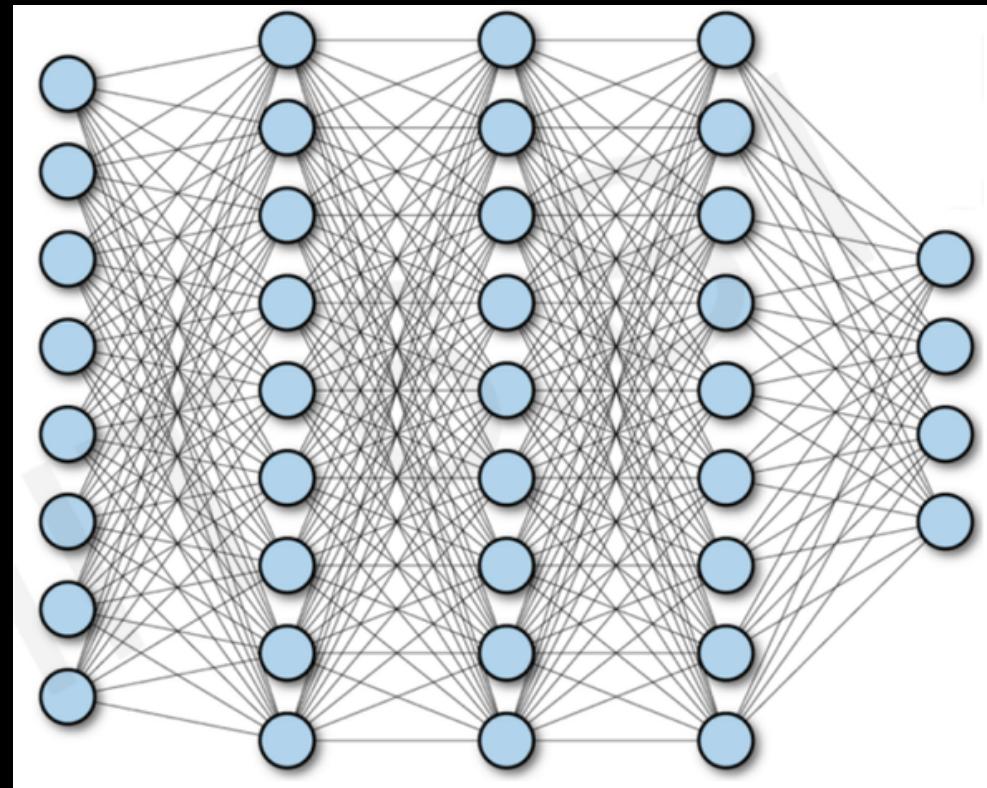


feature extraction from the data!
Learn hierarchy of features!



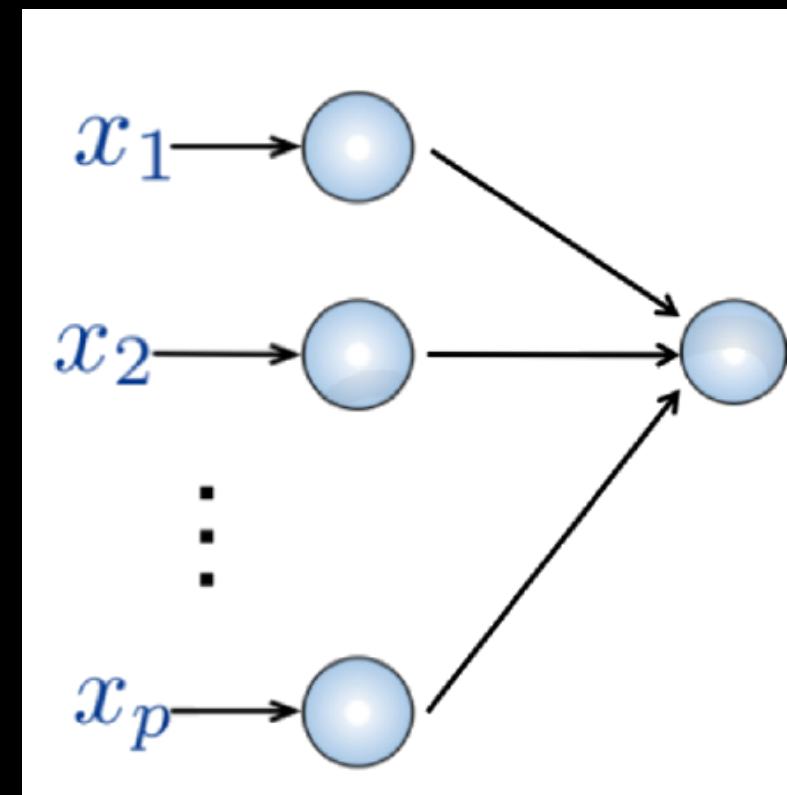
TSWC, 2022

Fully connected NN



Input:

- 2D image
- vector of pixel values
(flatten the image)



Fully connected:

- Connect neuron in hidden layer to all neurons in the input layer
- No spatial information!
- Lot of parameters

How to add spatial structure in the input?



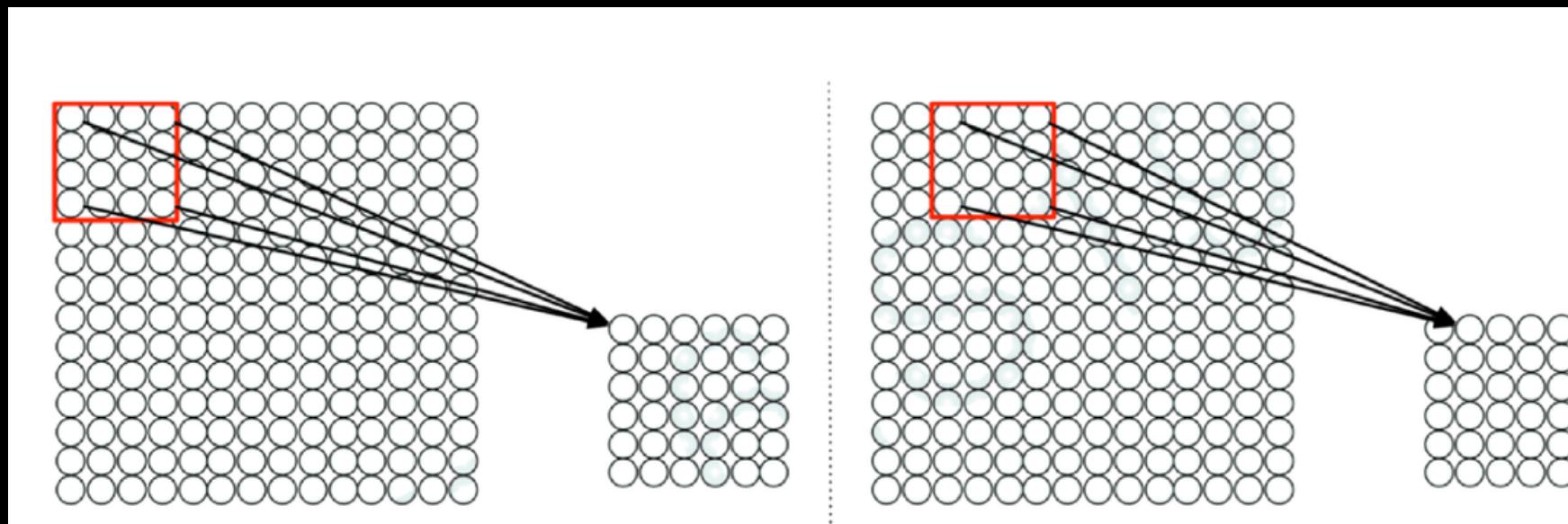
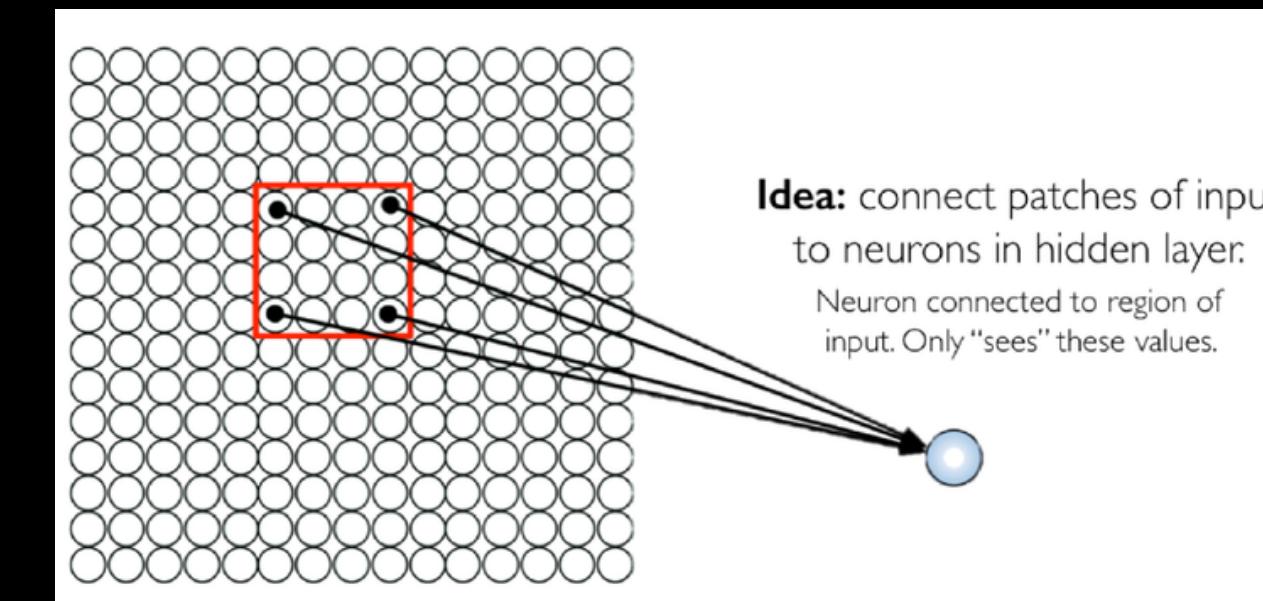
TSWC, 2022



Using spatial structure

Input:

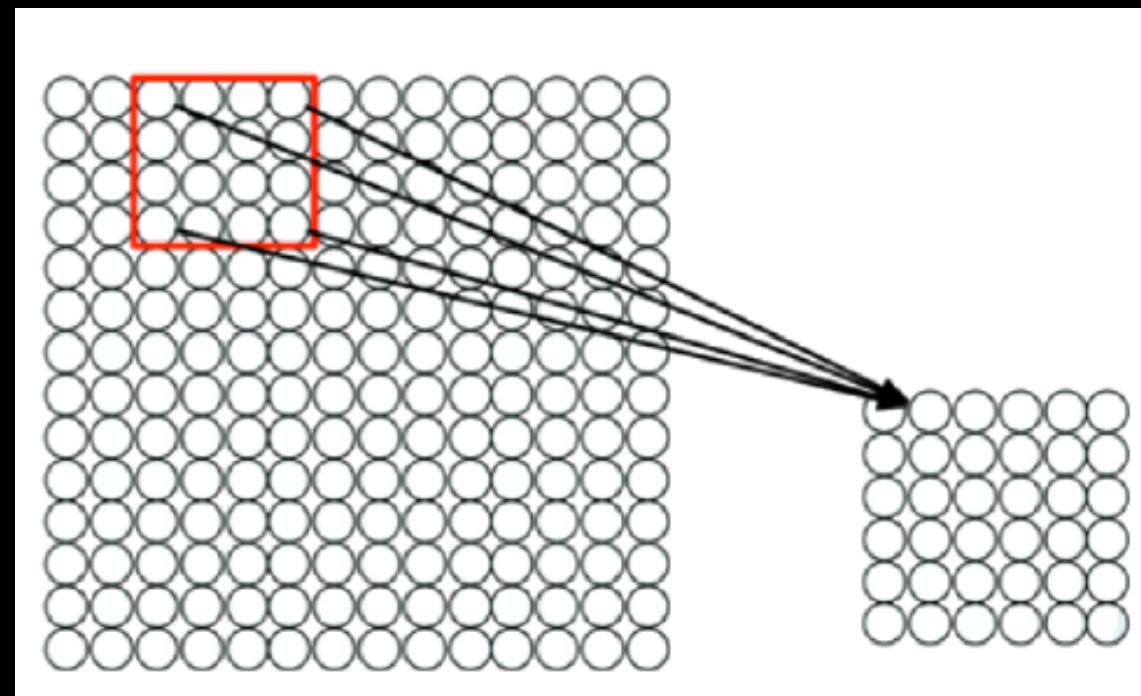
- 2D image
- **Array** of pixel values



Sliding window to define connections,(connect patch in input layer to a single neuron)

The key: how can we **weight** the patch to detect particular features?

Using spatial structure



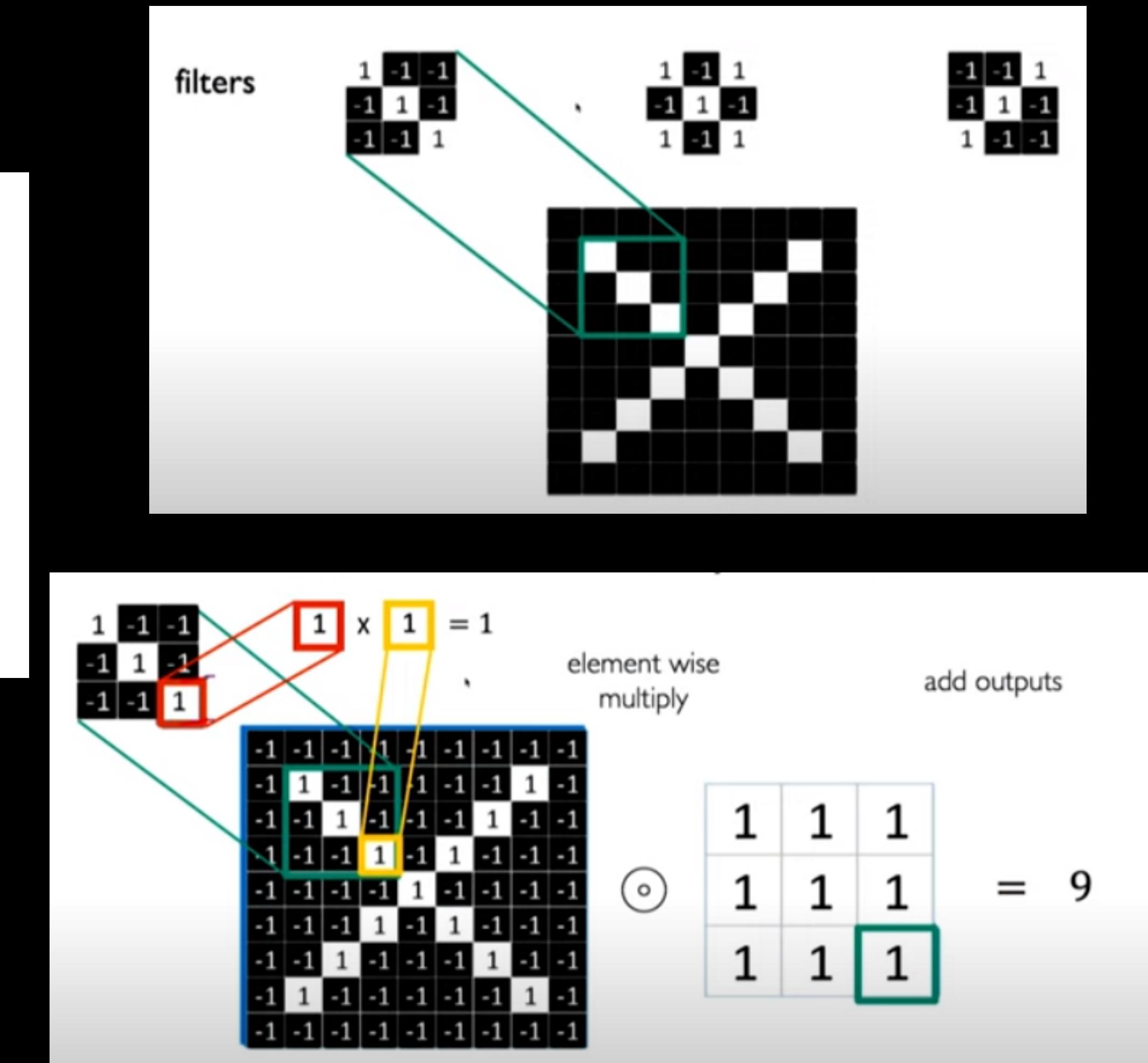
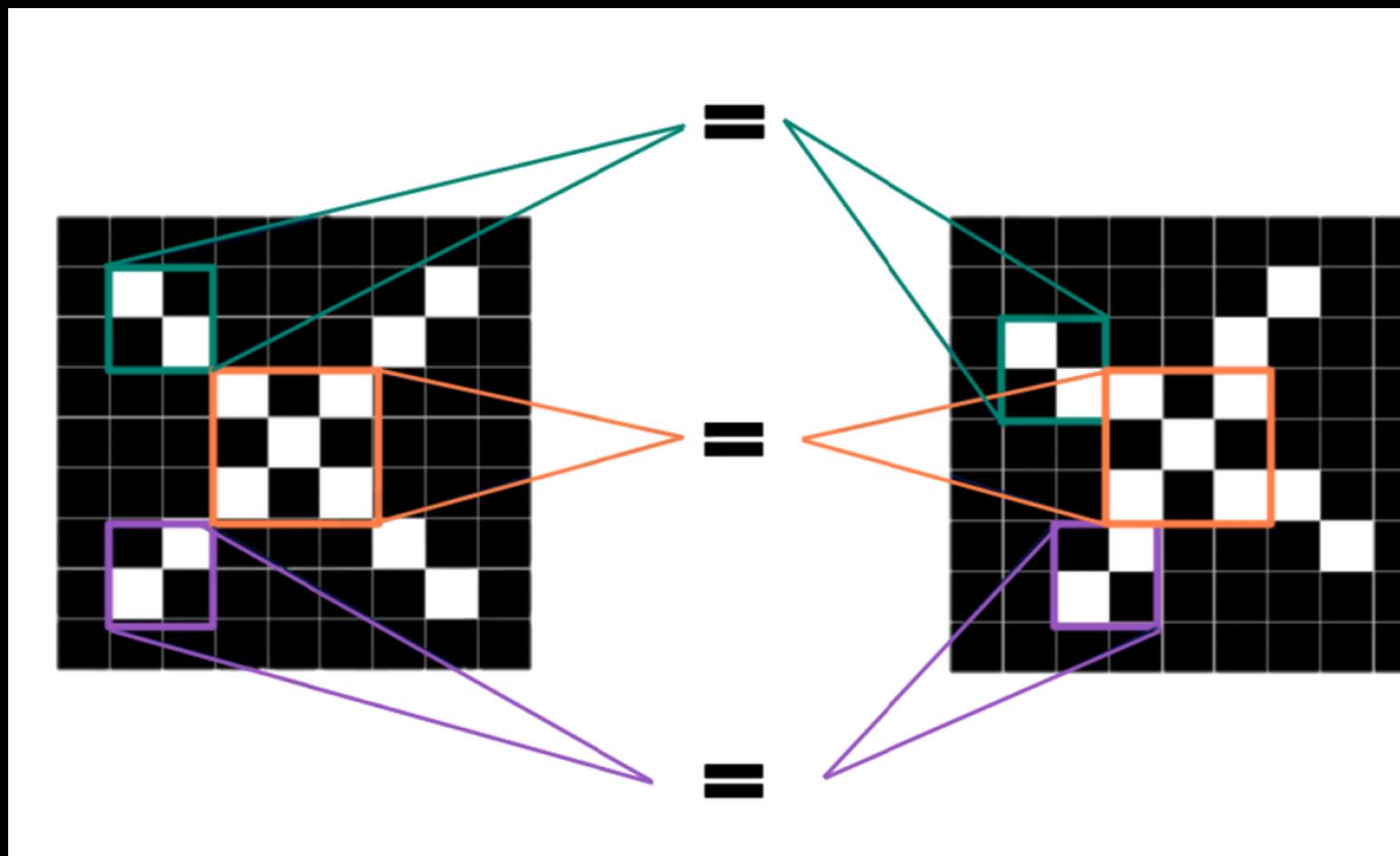
- Filter of size 4x4: 16 different weights
- Apply this same filter to a 4x4 patches in input
- Shift by 2 pixels for next patch

This "patchy" operation is **convolution**

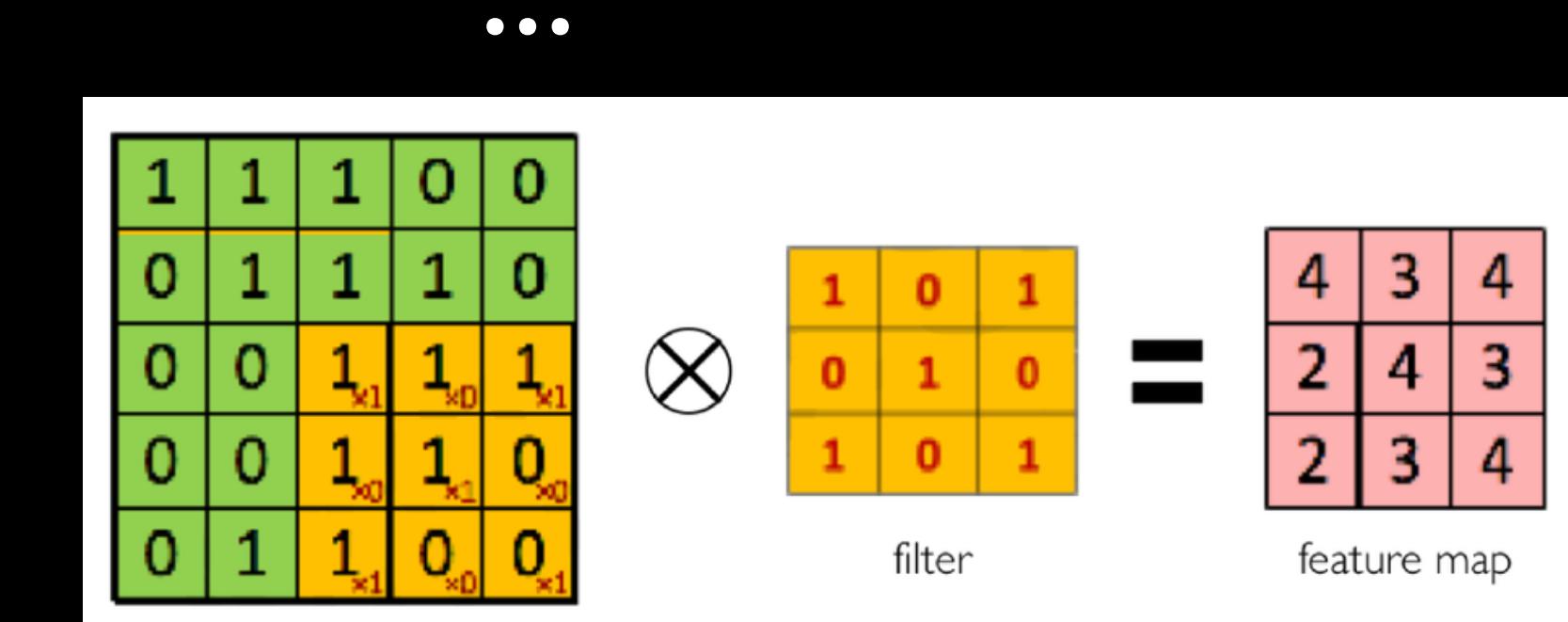
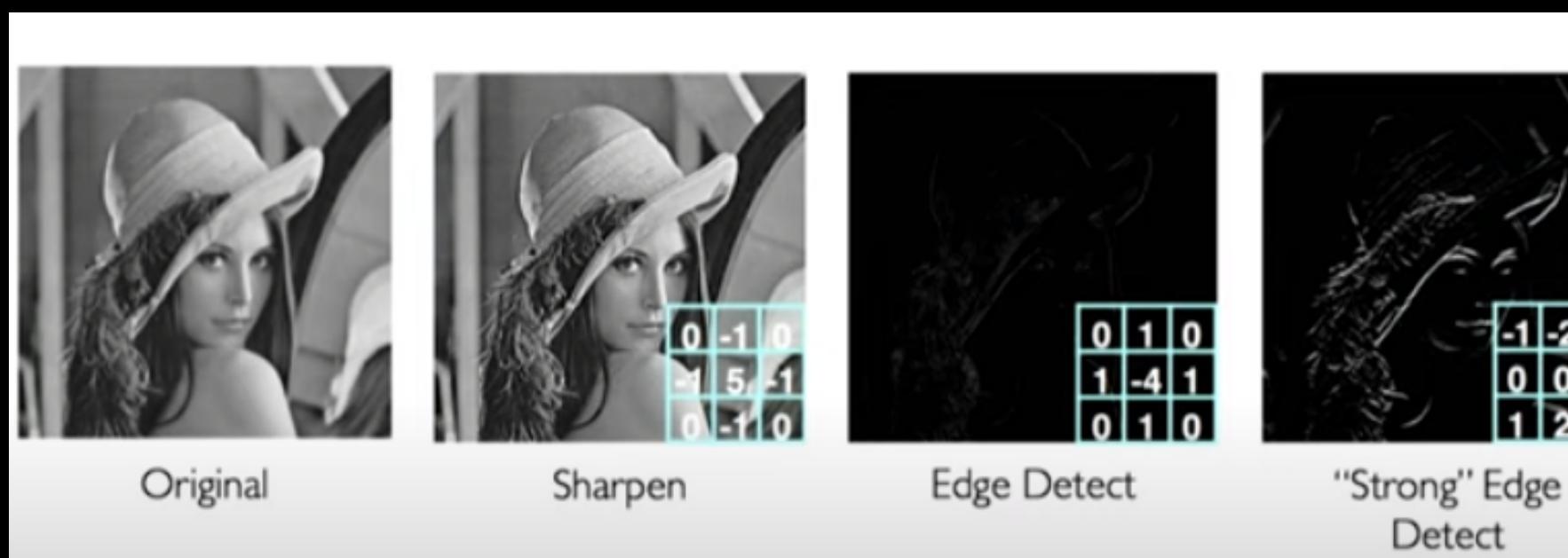
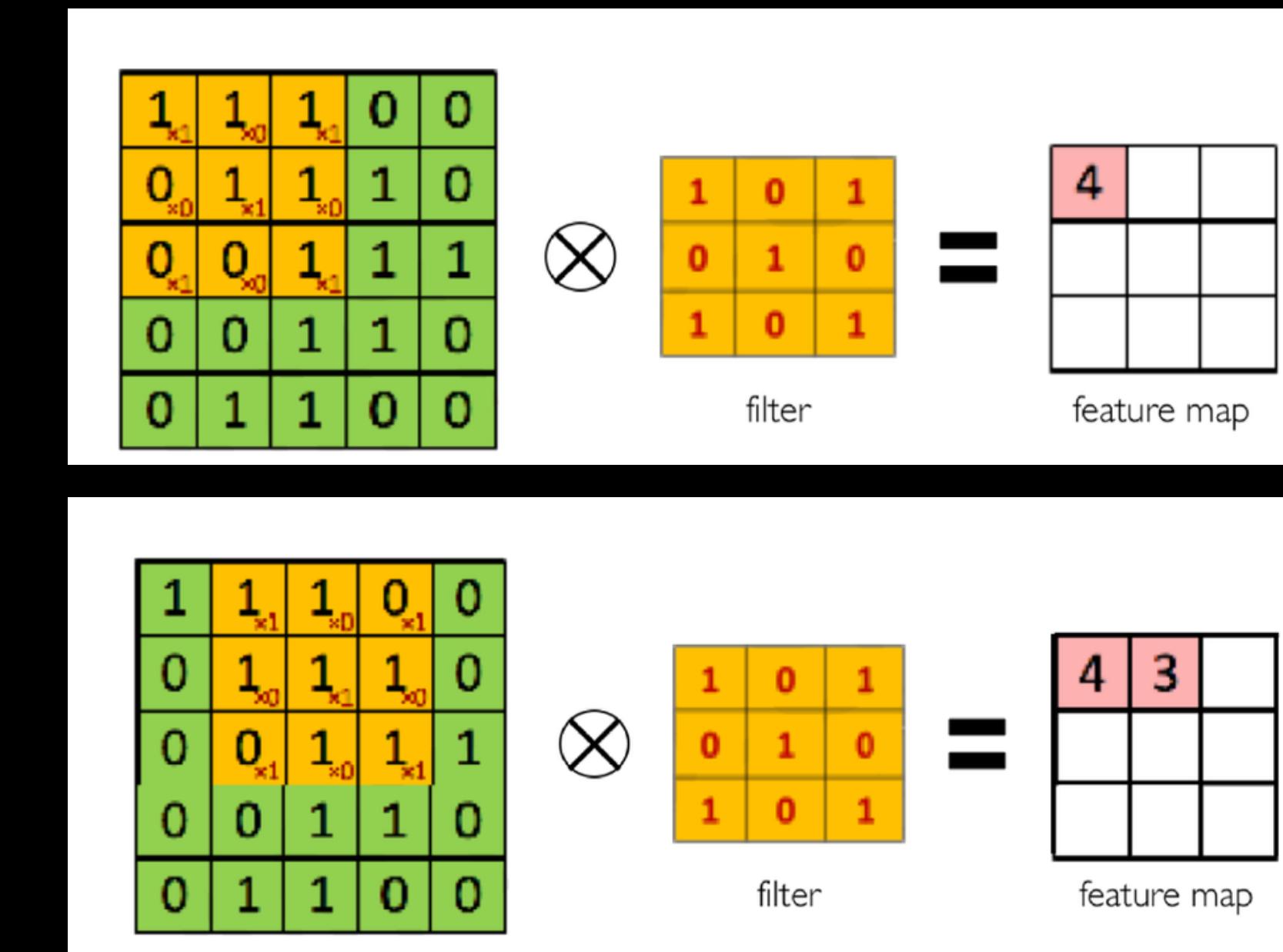
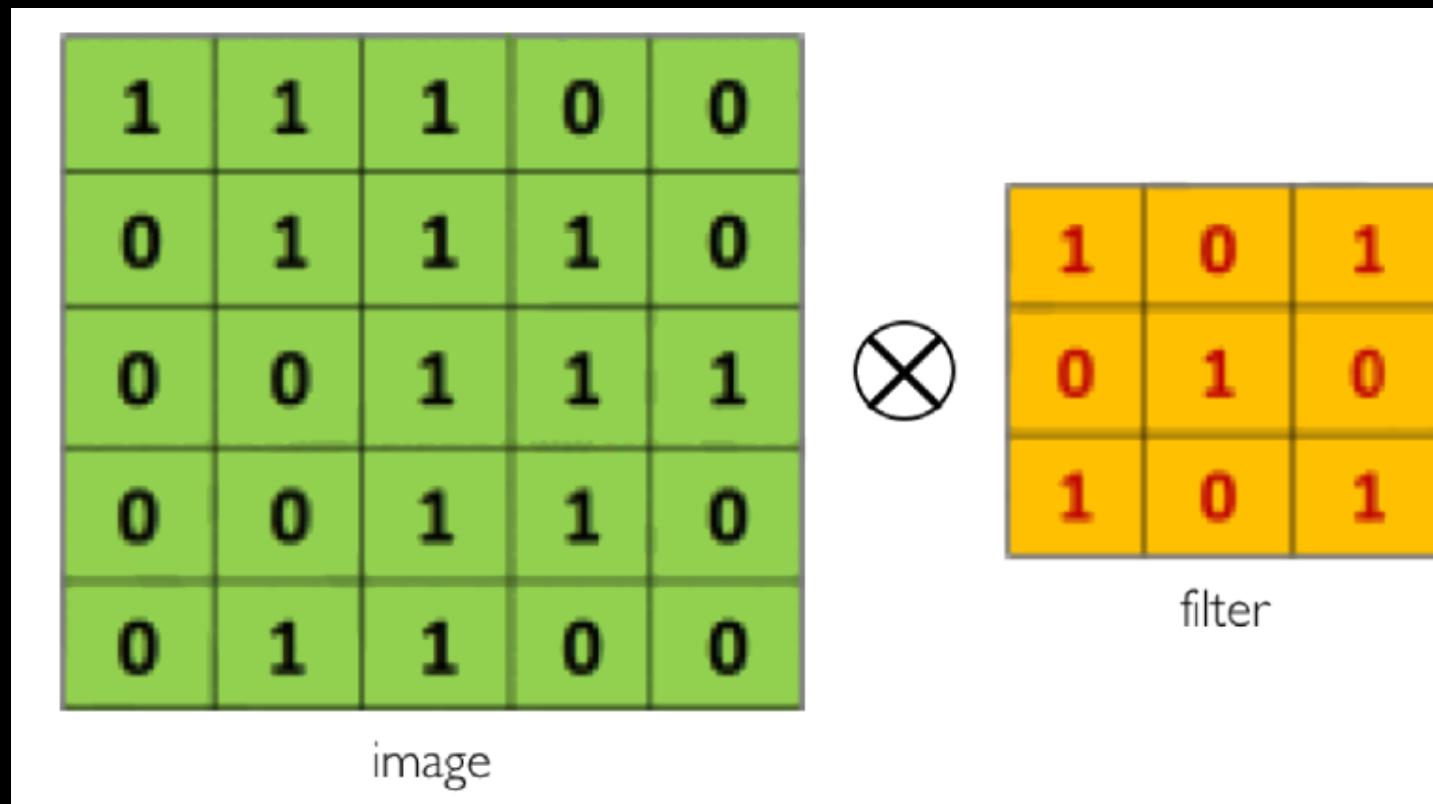
- Apply a set of weights (a filter) to extract **local features**
- Use multiple filters to extract different features
- **Spatially share** parameters of each filter



Convolutional operation

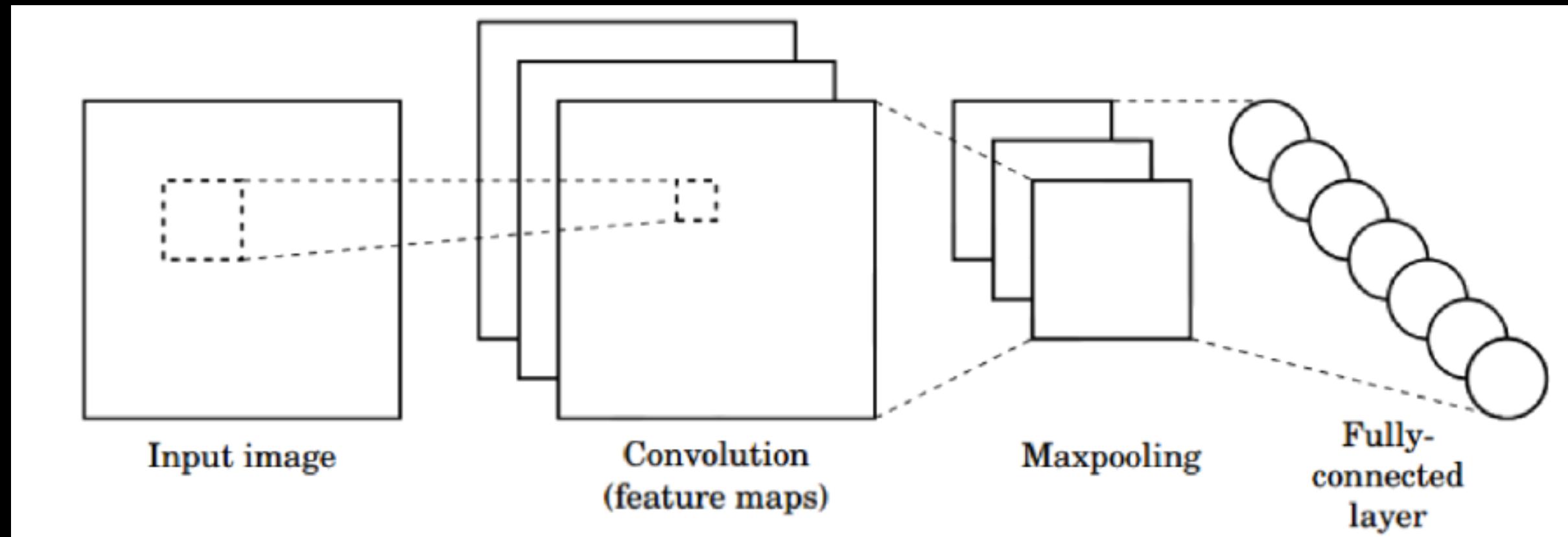


Convolutional operation



Convolutional neural networks (CNN)

- for classification



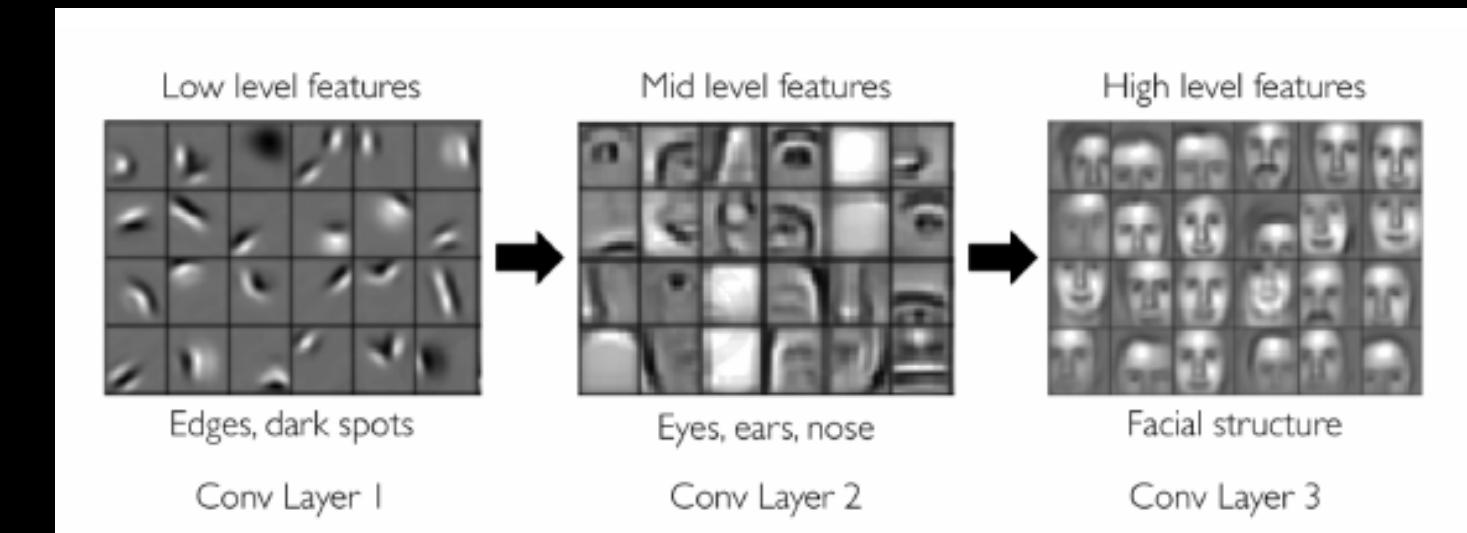
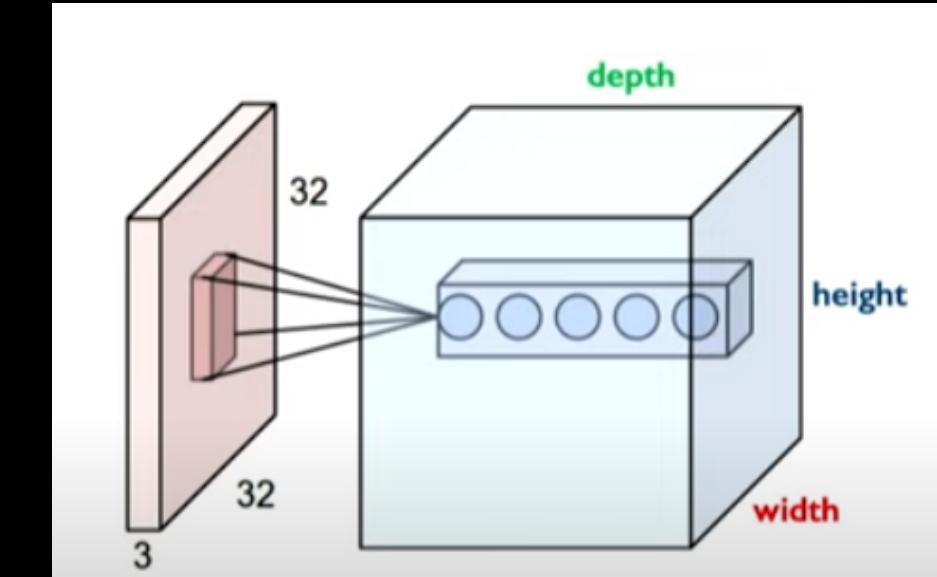
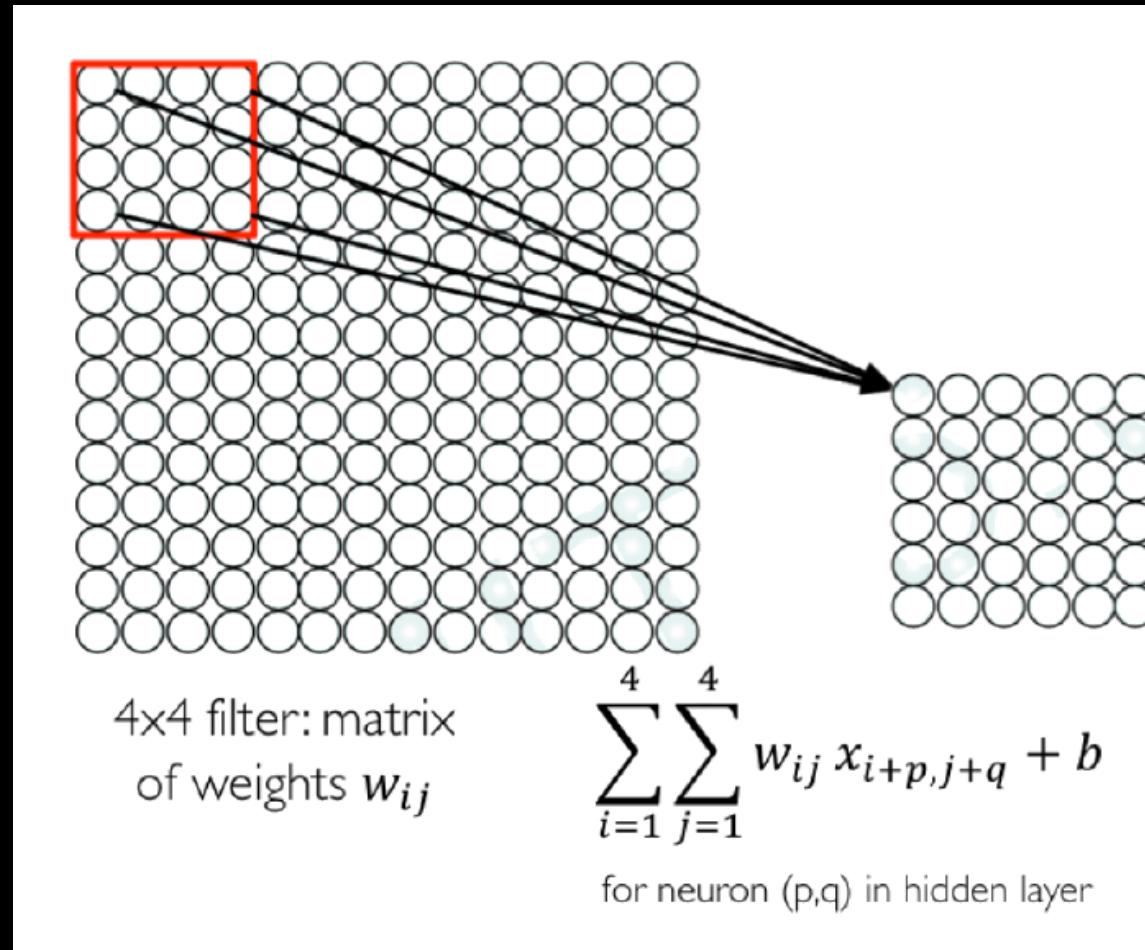
- **Convolution**: apply filters to generate feature maps
- **Non-linearity**: often ReLU
- **Pooling**: Downsampling operation on each feature map

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





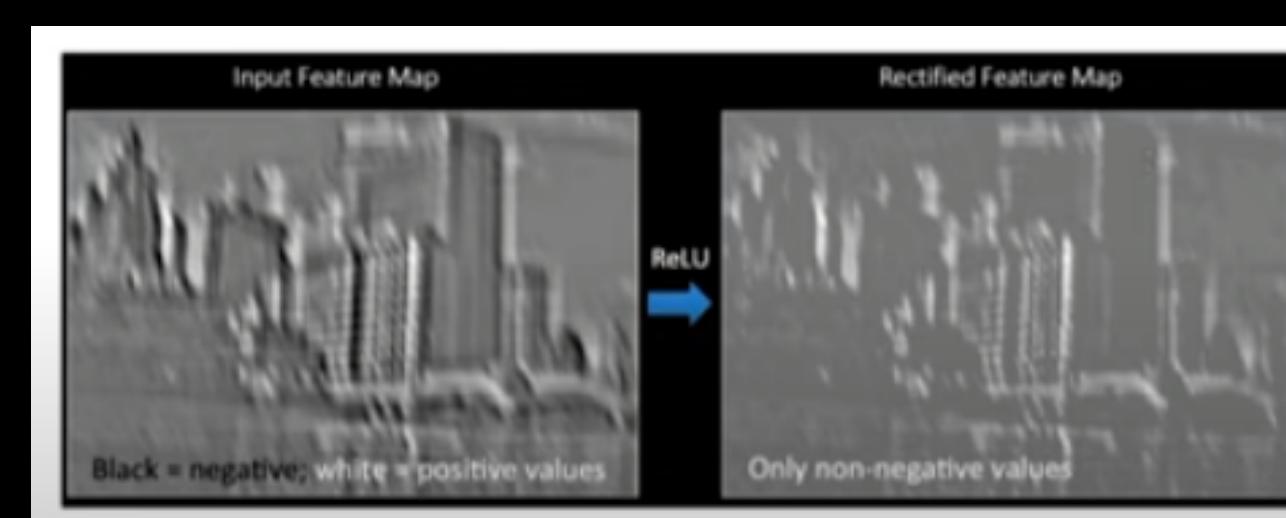
Convolutional layer



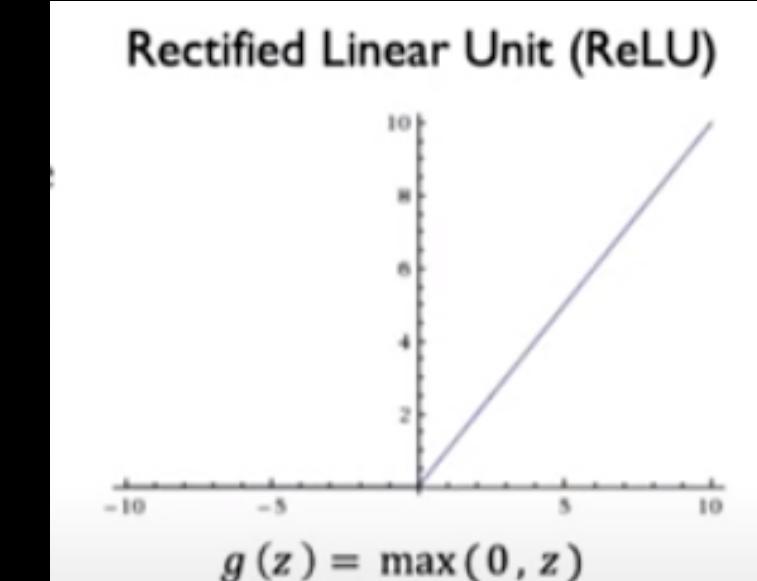
1. applying a window of weights
2. computing linear combinations

For a neuron in hidden layer

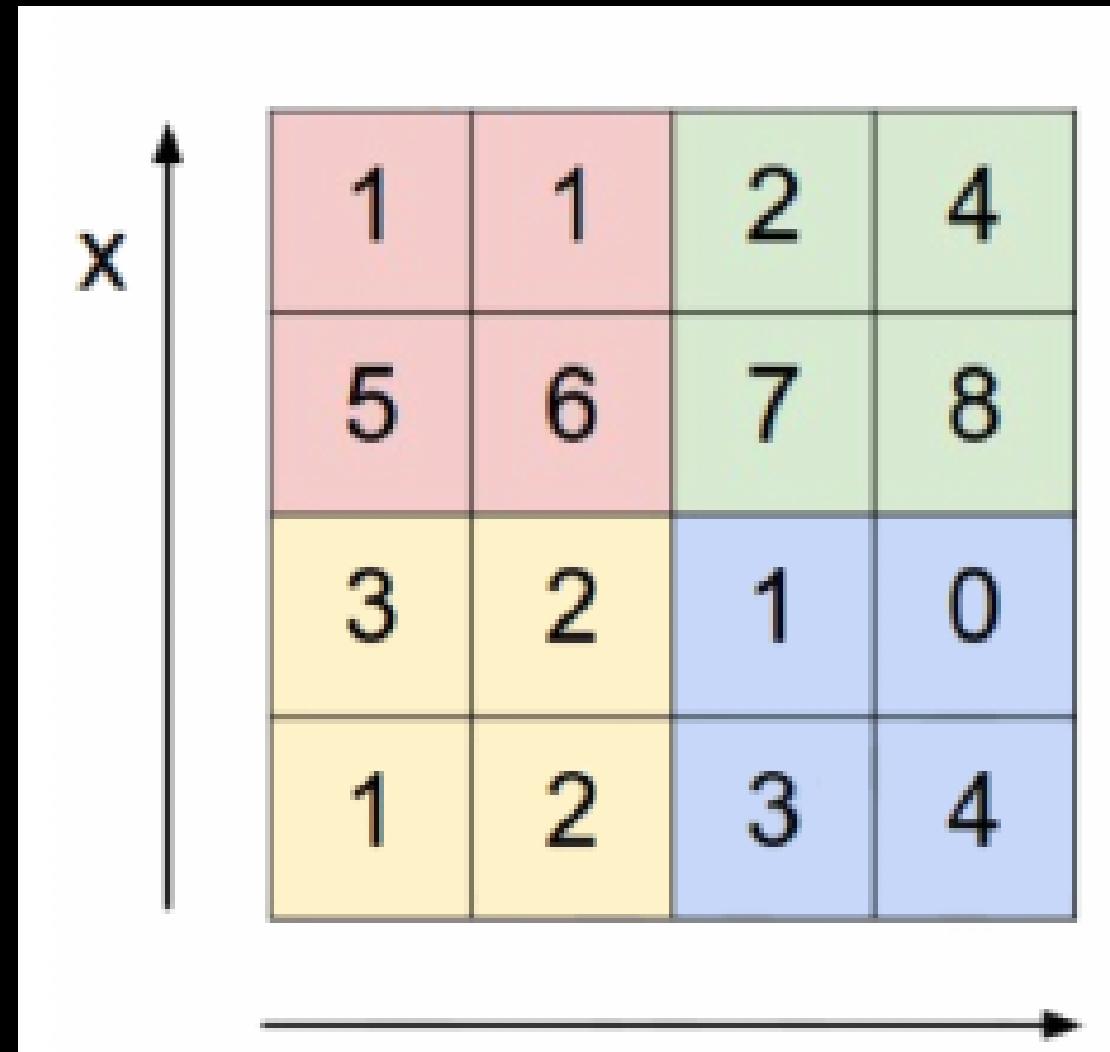
- Take inputs from patch
- Compute weighted sum
- apply bias



3. activating with non-linear function



Pooling



max pool with 2x2 filters
and stride 2

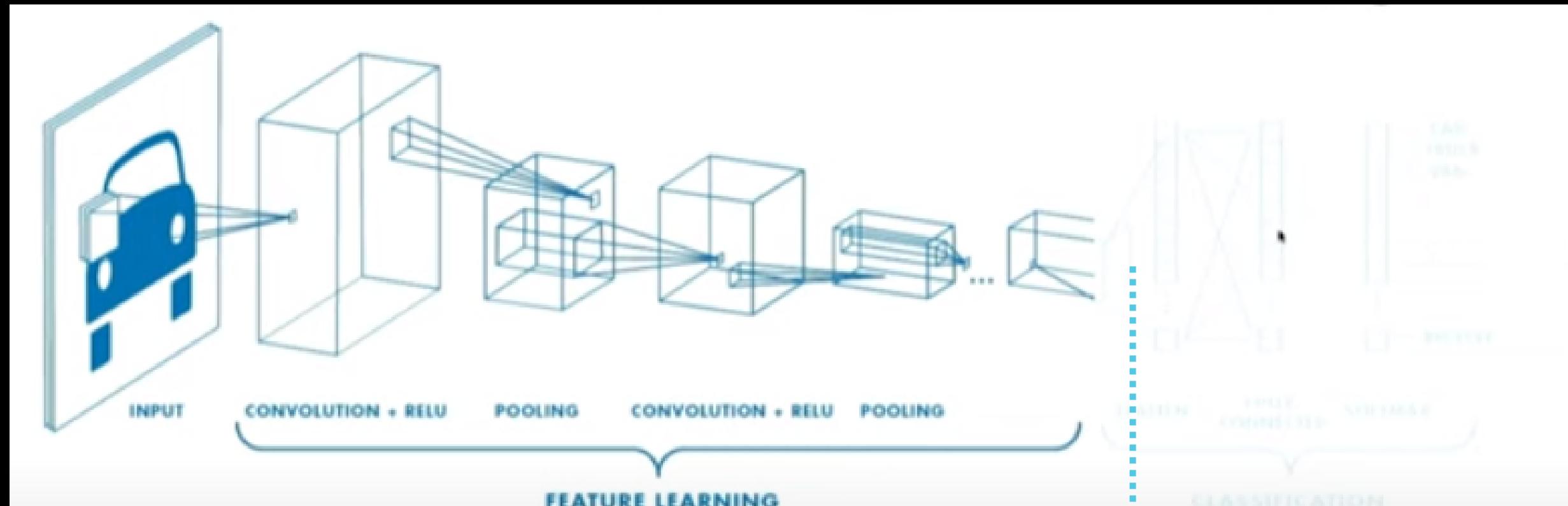


6	8
3	4

- Reduce dimensionality
- Spatial invariance



CNN: classification example



- Learns features in input image through **convolution**
 - Introduce non-linearity (activation func)
 - Reduce dimensionality and preserve spacial invariance (**pooling**)
- at this point the last output is the i-th feature map

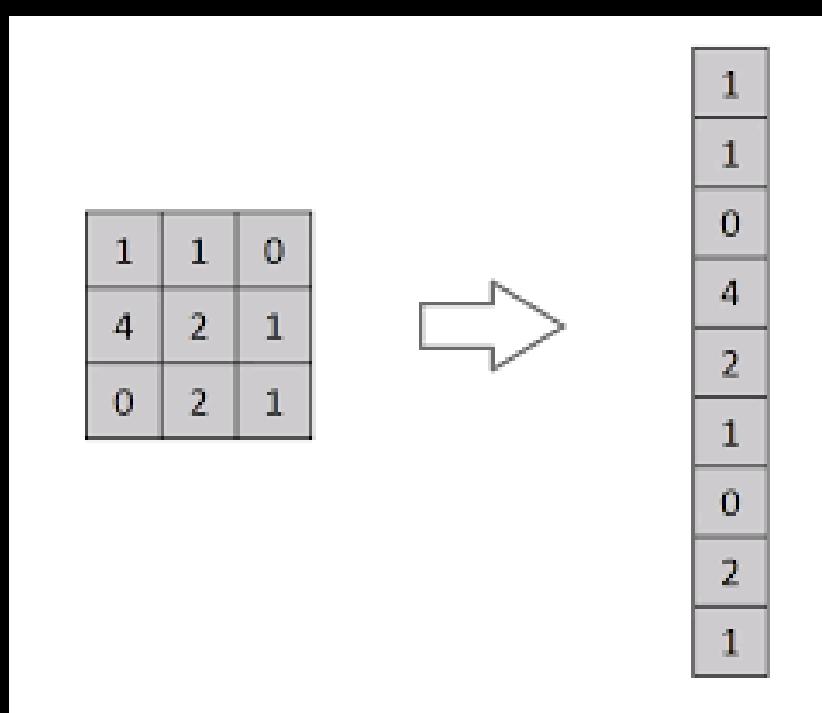




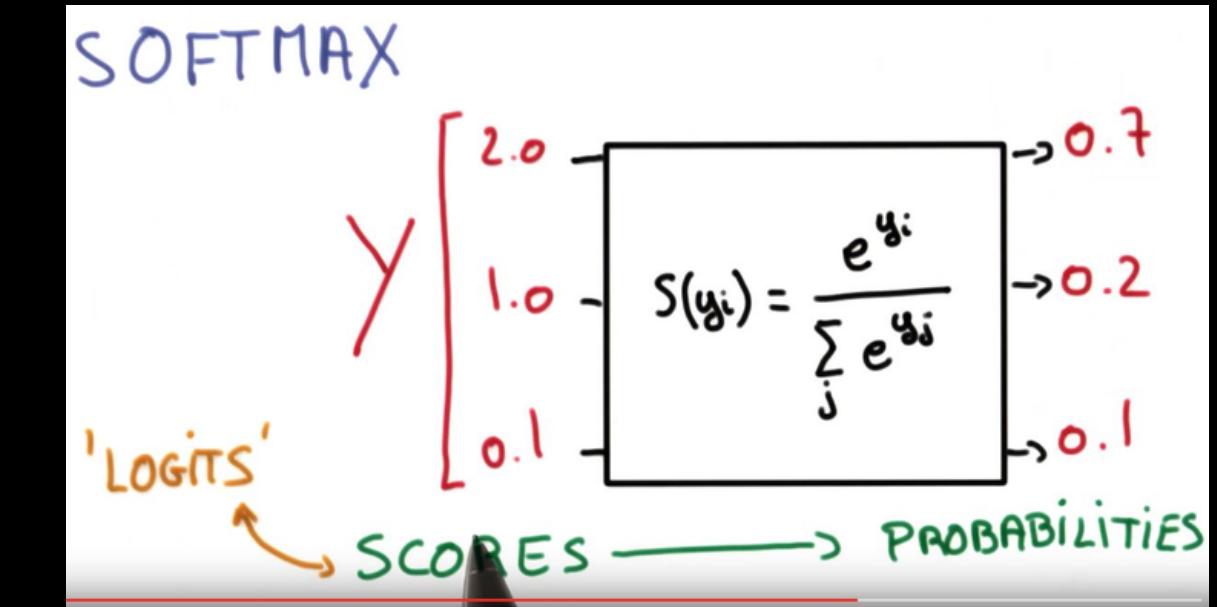
CNN: classification example



- CONV and POOL layers putput **high-level features input**
- Fully connected layer uses these **features to classifying** input image
- Express **output as probability** (image to certain class)



$$\text{softmax}(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$



logits = unnormalised (or not-yet normalised)
predictions (or outputs) of a model

```
32     self.file = None
33     self.fingerprints = set()
34     self.logdepth = 0
35     self.debug = False
36     self.logger = logging.getLogger('fingerprint')
37     if path:
38         self.file = open(path, 'w')
39         self.file.write('')
40         self.fingerprints.add(path)
41
42     @classmethod
43     def tree_settingscls(settings):
44         debug = settings.getoption('fingerprint.debug')
45         return cls(debug=debug)
46
47     def request_genuine(self, request):
48         fp = self.request_fingerprint(request)
49         if fp in self.fingerprints:
50             return True
51         self.fingerprints.add(fp)
52         self.file.write(fp + '\n')
53
54     def request_fingerprint(self, request):
55         return self.request_genuine(request)
```

HANDS-ON LAB

TP - 3

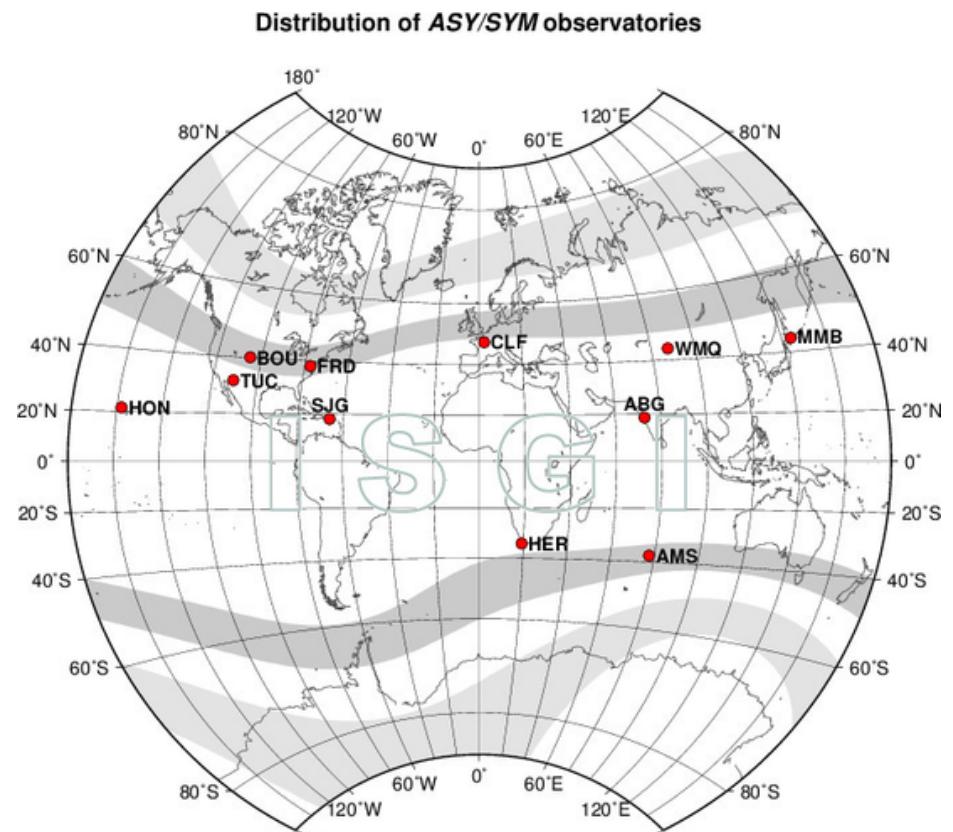
Sym-H forecasting



TSWC, 2022

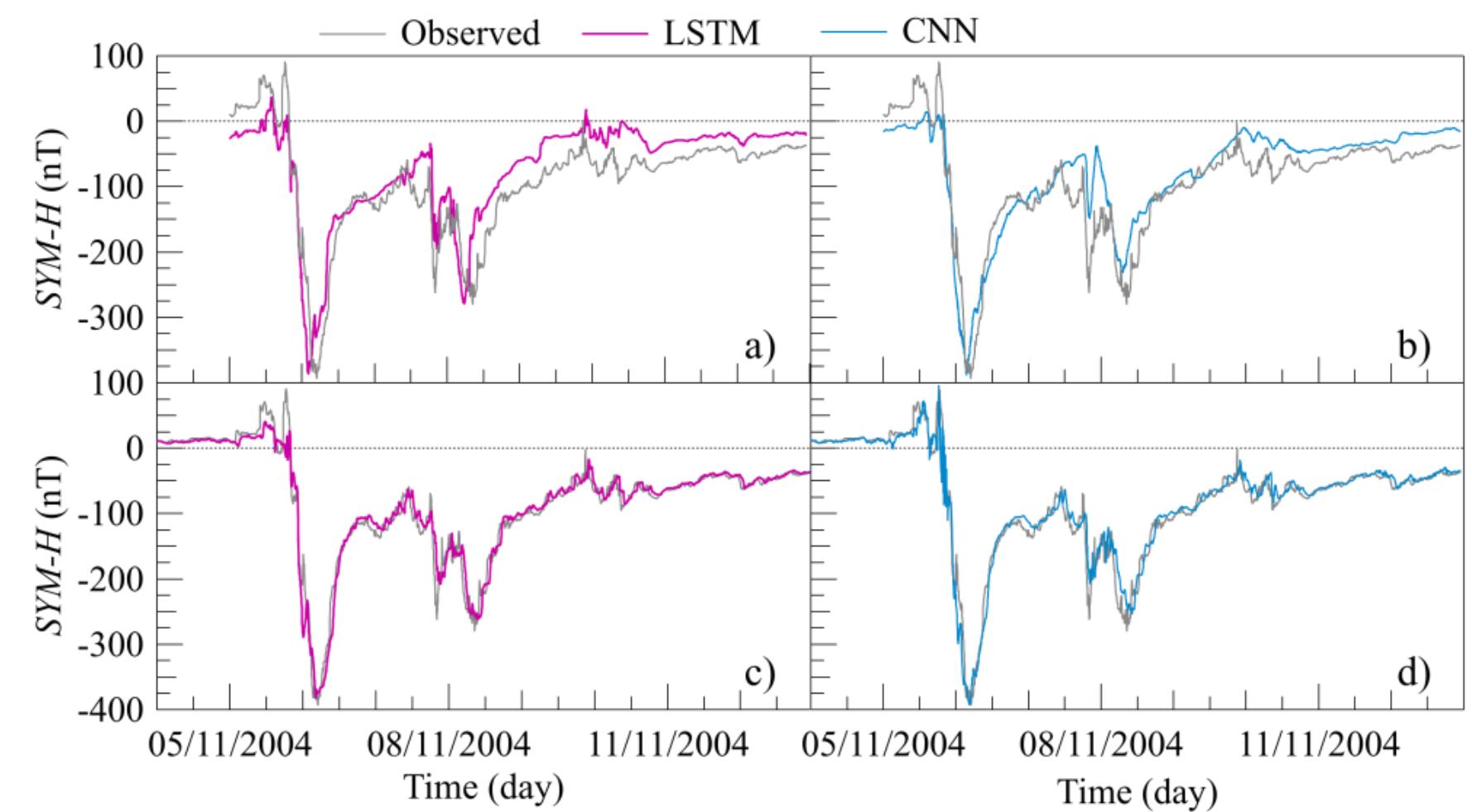
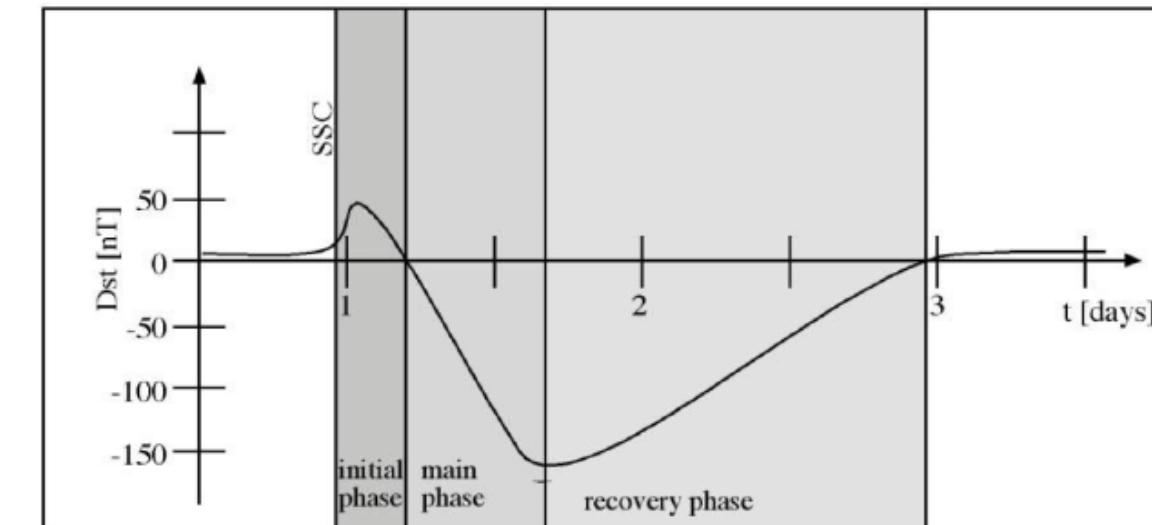
From science to data science

http://isgi.unistra.fr/indices_asy.php



To describe the geomagnetic disturbances at mid-latitudes in terms of longitudinally asymmetric (ASY) and symmetric (SYM) disturbances for both H and D components respectively parallel and perpendicular to the dipole axis.

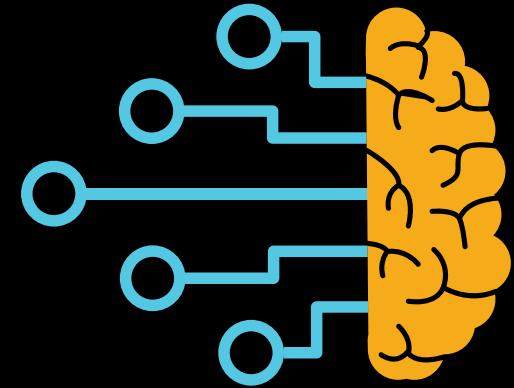
SYM-H is essentially the same as the Dst index with a different time resolution.



DCAO, UBA
08-12 AGOSTO 2022

APRENDIZAJE AUTOMÁTICO.

Fundamentos y Aplicaciones en
Meteorología del Espacio



Dra María Graciela Molina

FACET-UNT / CONICET
Tucumán Space Weather Center - TSWC

<https://spaceweather.facet.unt.edu.ar/>
IG -> @spaceweatherargentina

gmolina@herrera.unt.edu.ar

