UiO : University of Oslo

institutt for informatikk

IN4050 - Introduction to Artificial Intelligence and Machine Learning

Lecture 11

Deep Neural Networks

Ali Ramezani-Kebrya

Fill out Scary Form!

https://nettskjema.no/a/560343

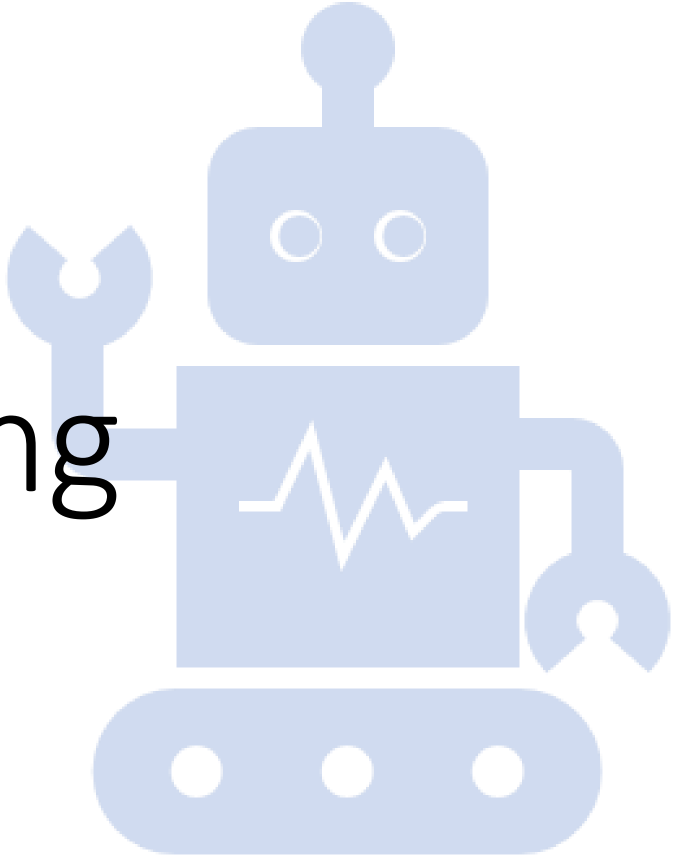
Today

1. The deep learning revolution
2. Deep feed-forward neural networks
3. Convolutional NNs and image processing
4. Recurrent NNs and language processing

12.1 The deep-learning revolution

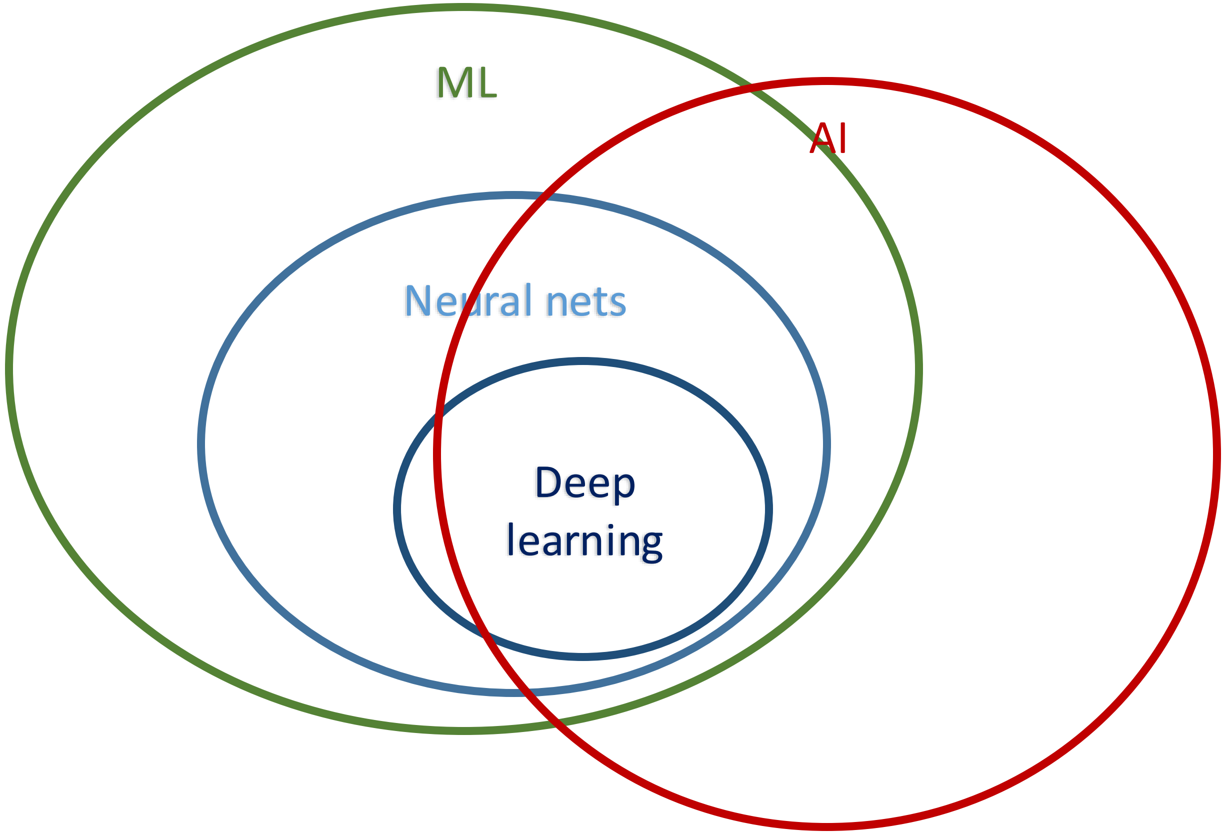
IN4050 Introduction to Artificial Intelligence

and Machine Learning



Deep learning

- A sub-class of neural nets
- No exact definition:
  - More an attitude/approach than a defined class
  - Normally: at least two hidden layers
  - Often: a much more specific architecture

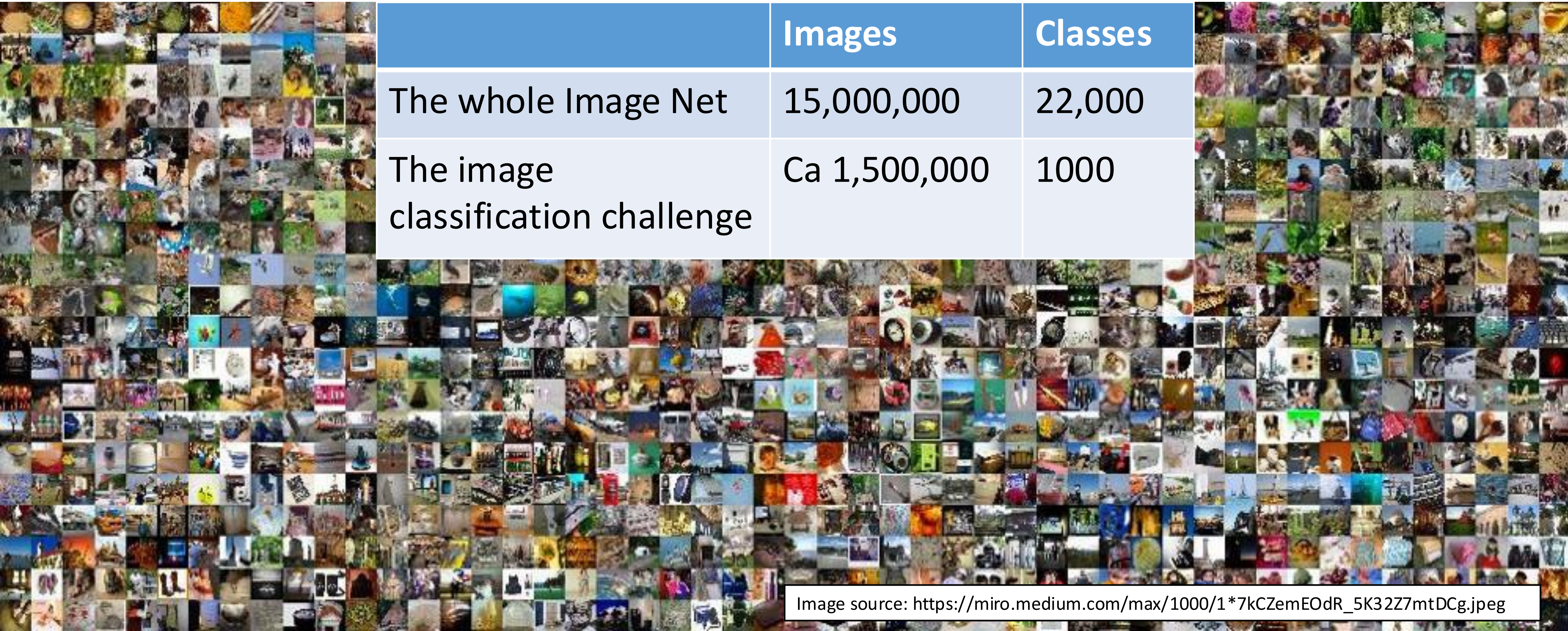


The revolution

- Deep learning breakthrough 13 years ago
- It spawned the great interest in AI we have seen the last years
  - One started to talk about AI again – not only ML

- Images:
  - Image classification
  - Object detection
  - Scene understanding
- Language:
  - Speech recognition
  - Machine translation
  - Chatbots
- Game playing
- Applications:
  - Self-driving cars

Image Net

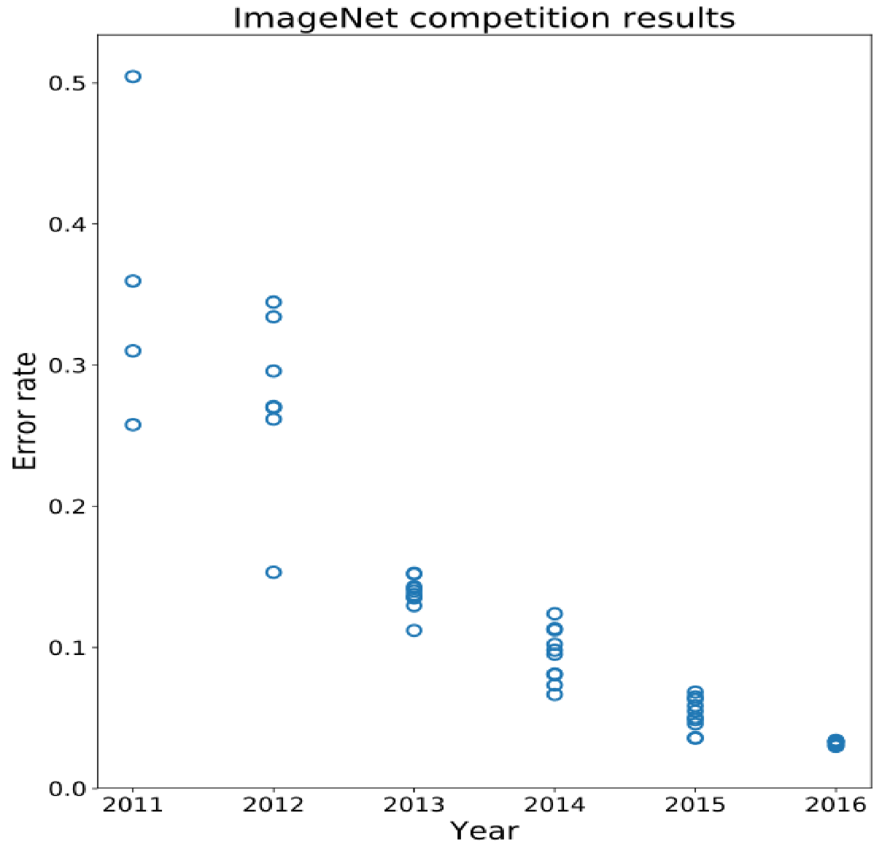


	Images	Classes
The whole Image Net	15,000,000	22,000
The image classification challenge	Ca 1,500,000	1000

Image source: https://miro.medium.com/max/1000/1*7kCZemEOdR_5K32Z7mtDCg.jpeg

The Image Net Competition

- 2012: Alex Net:
  - won the competition
  - lowered the error rate from 26% to 16%
  - Based on deep NNs
- Immediate renewed interest in neural nets
- Interest in AI in the population at large
- In 2014, GoogLeNet: 7%
- 2015: the winner <4%
  - Better than humans



https://en.wikipedia.org/wiki/File:ImageNet_error_rate_history_(just_systems).svg

Speech recognition

- This was the second big revolution
- There was large progress in the 1990s:
  - Bill Gates predicted we could get rid of the keyboard 10 years later
  - It was established a large company in Norway: Nordisk Språkteknologi
  - Too early!
- With deep learning, we got speech recognition which works (from 2015)
- This made these gadgets possible

Other applications

- Since then, deep learning has been applied to nearly all areas where ML is applied
- The improvements are not always as great as for image recognition and speech
- But DL performs on top in nearly all ML tasks
  - At least where you have large amounts of data

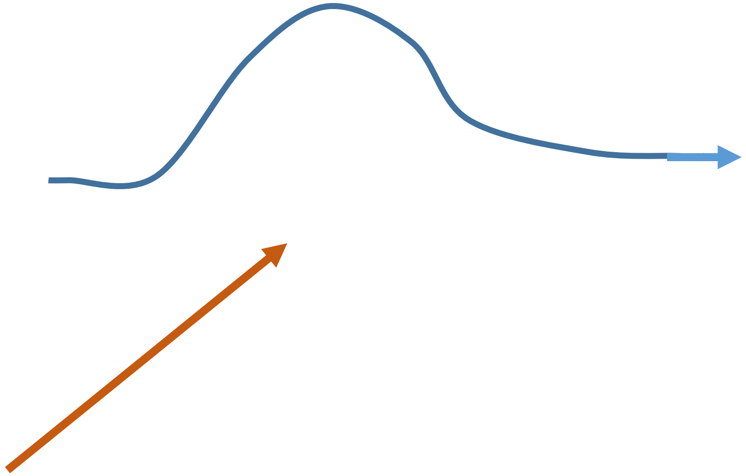
History of neural networks

Three main epochs:

1. The beginning (→ 1969)

2. Backpropagation (1986-)

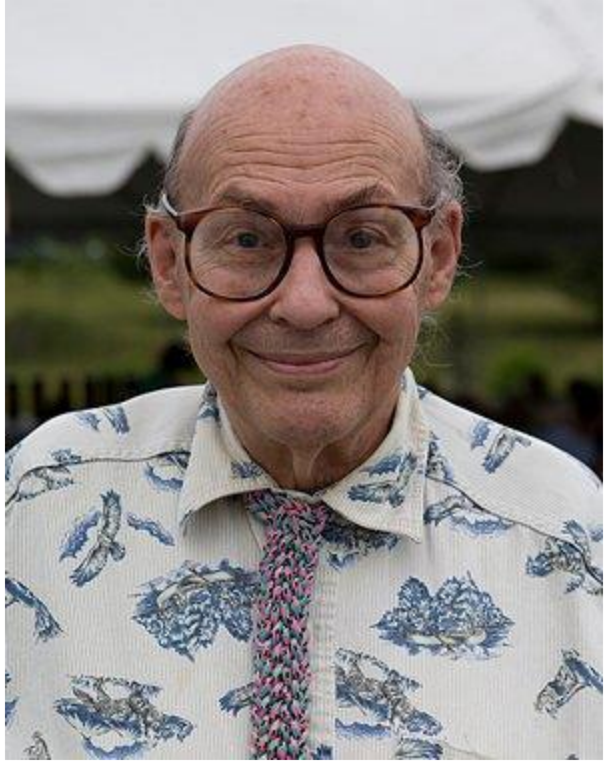
3. Deep learning (2011→)



- Marsland, originally 2009, lacks (3)

NN.1: The beginning (→ 1969)

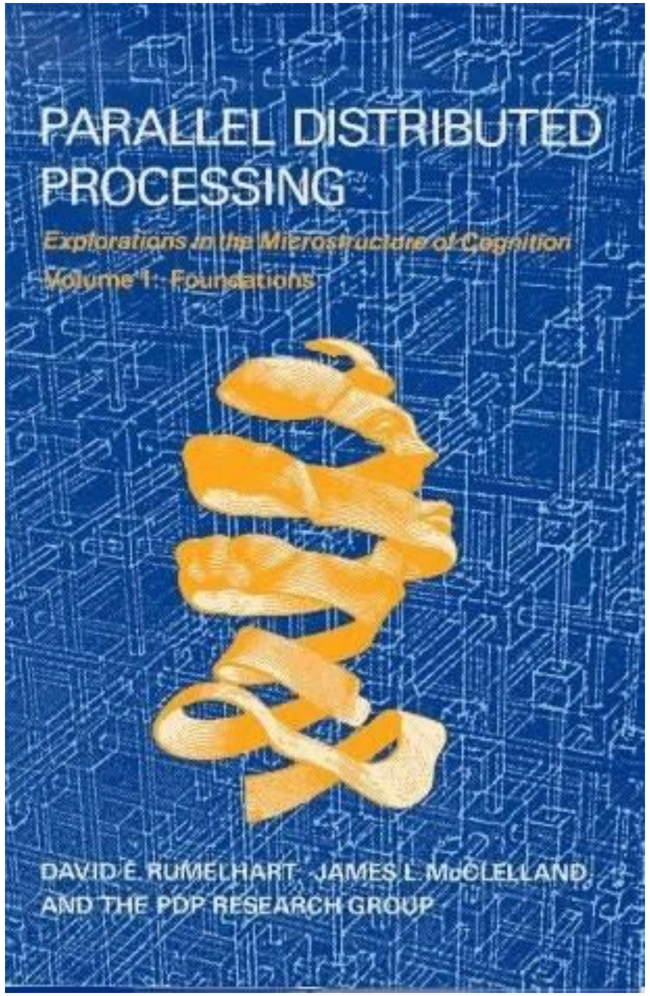
- 1958, Rosenblatt invented the perceptron
- 1969, Minsky & Papert, *The perceptron:*
  - Networks without hidden layers can only learn linear classifiers
  - Networks with hidden layers are probably impossible to train
- Less interest in perceptrons afterwards



Marvin Minsky (1927-2016)
AI pioneer, MIT AI Lab

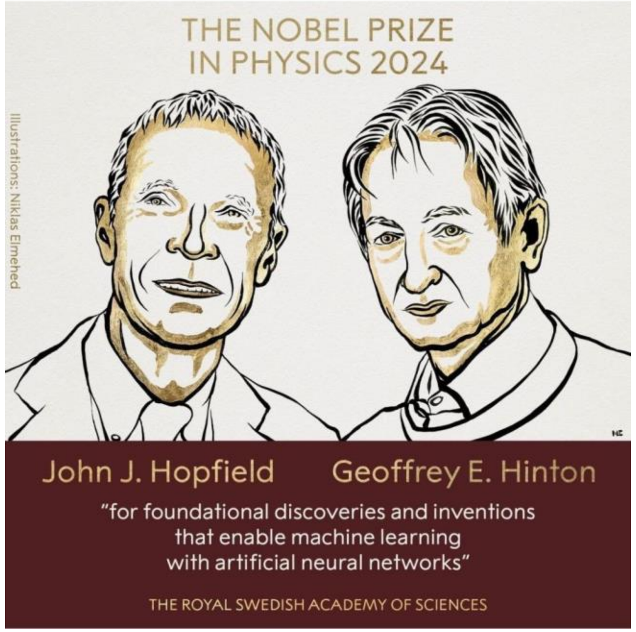
NN.2: Backpropagation (1986-)

- 1986, Rummelhart, Hinton, Williams (re)invented backpropagation
- An immediate enormous interest by researchers
- But the practical results weren't impressing, and the interest diminished



NN.3: Deep learning

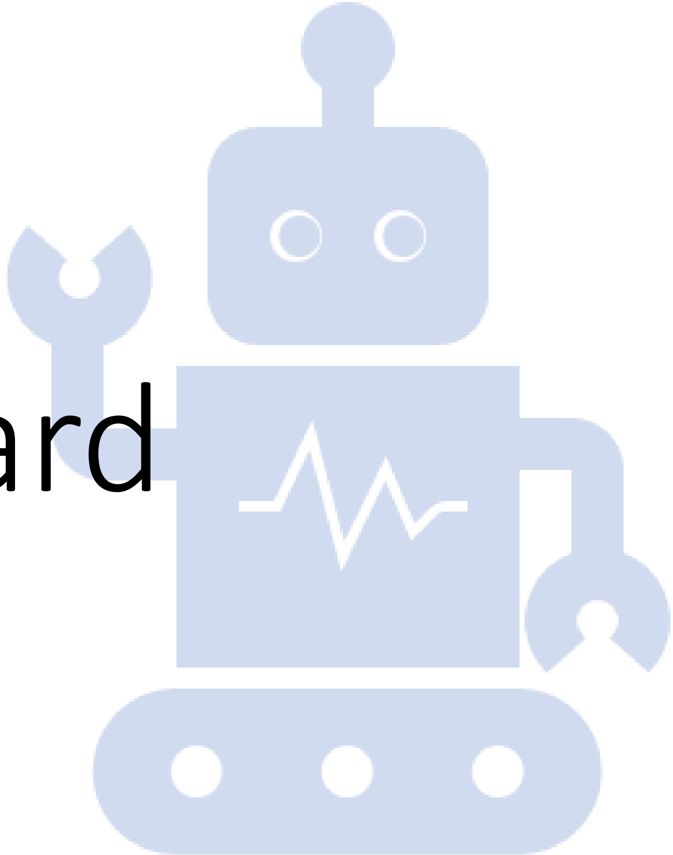
- In the 1990s and 2000s, logistic regression, SVM were popular
- Hinton and … continued working on neural nets got their rewards in 2010s
- https://awards.acm.org/about/2018-turing
- https://www.nobelprize.org/prizes/physics/2024



Why did NNs finally succeed?

- Better models
- More data
- More powerful machines
  - GPUs (graphical processing units)

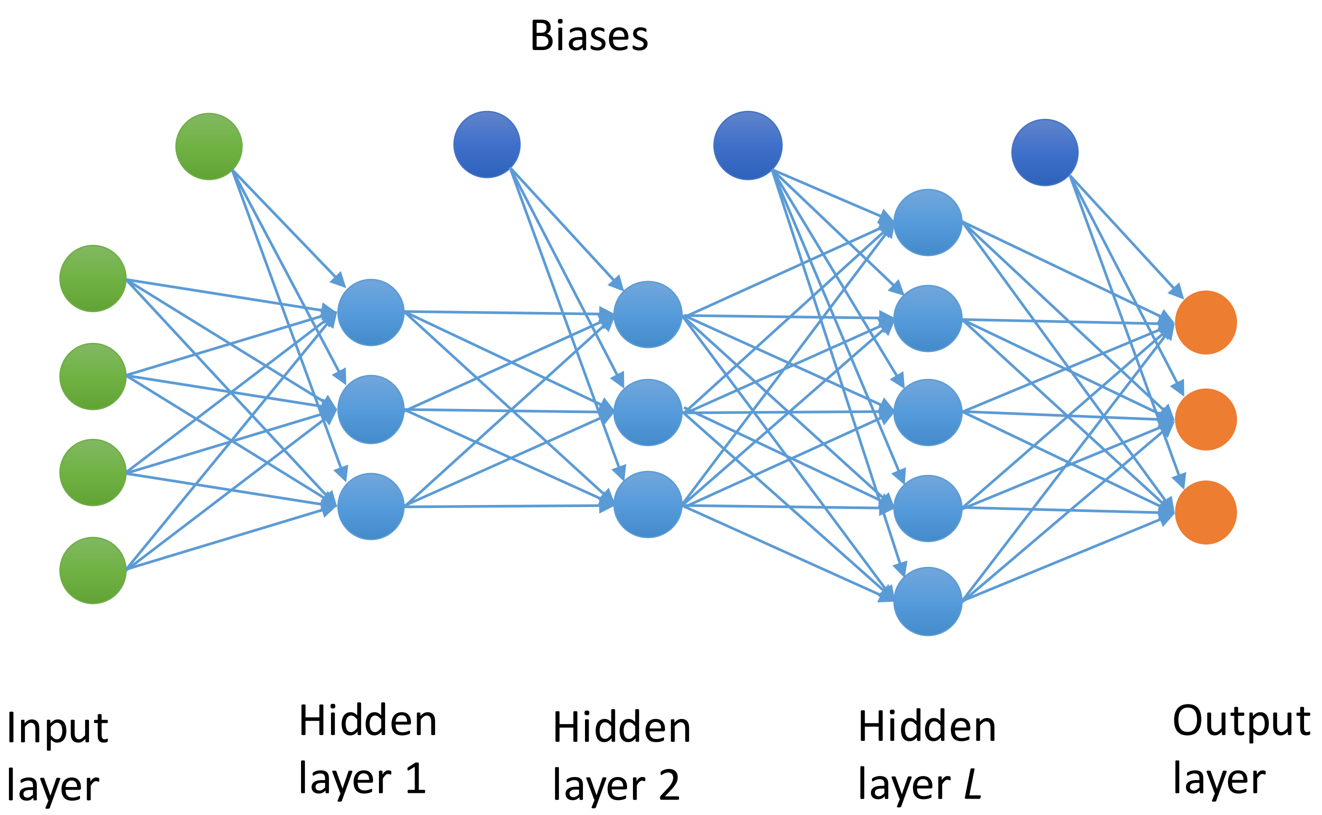
12.2 Deep feed-forward Neural networks



IN4050 Introduction to Artificial Intelligence and Machine Learning

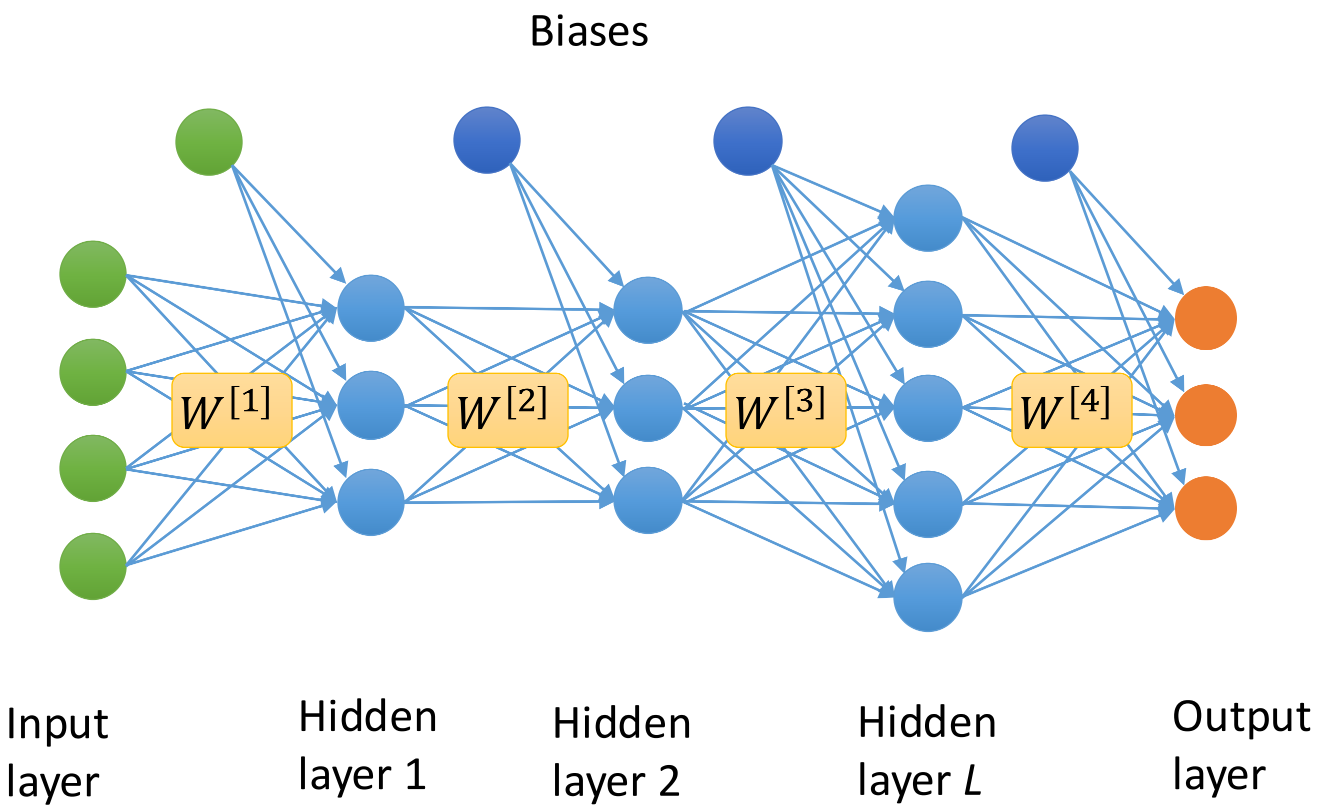
Deep feed-forward NN

- Several hidden layers
- The number of nodes in each layer may vary
- Fully-connected: edges from each node in one layer to each node in the next layer



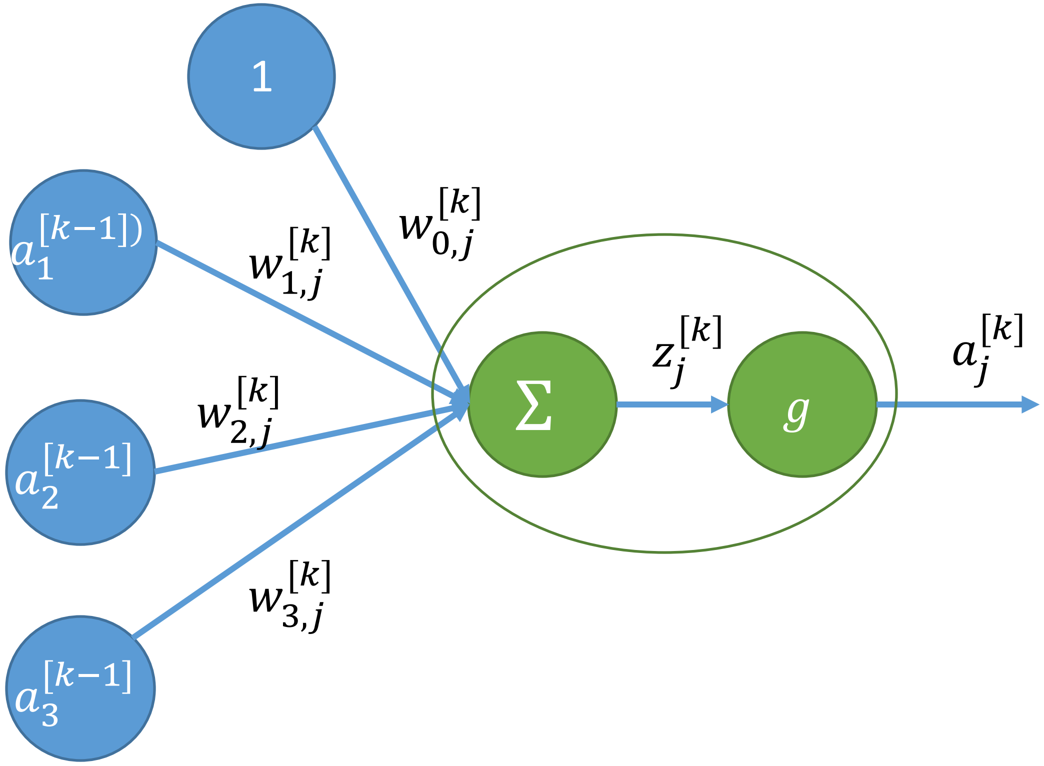
Weight matrices

- One matrix of weights for each layer



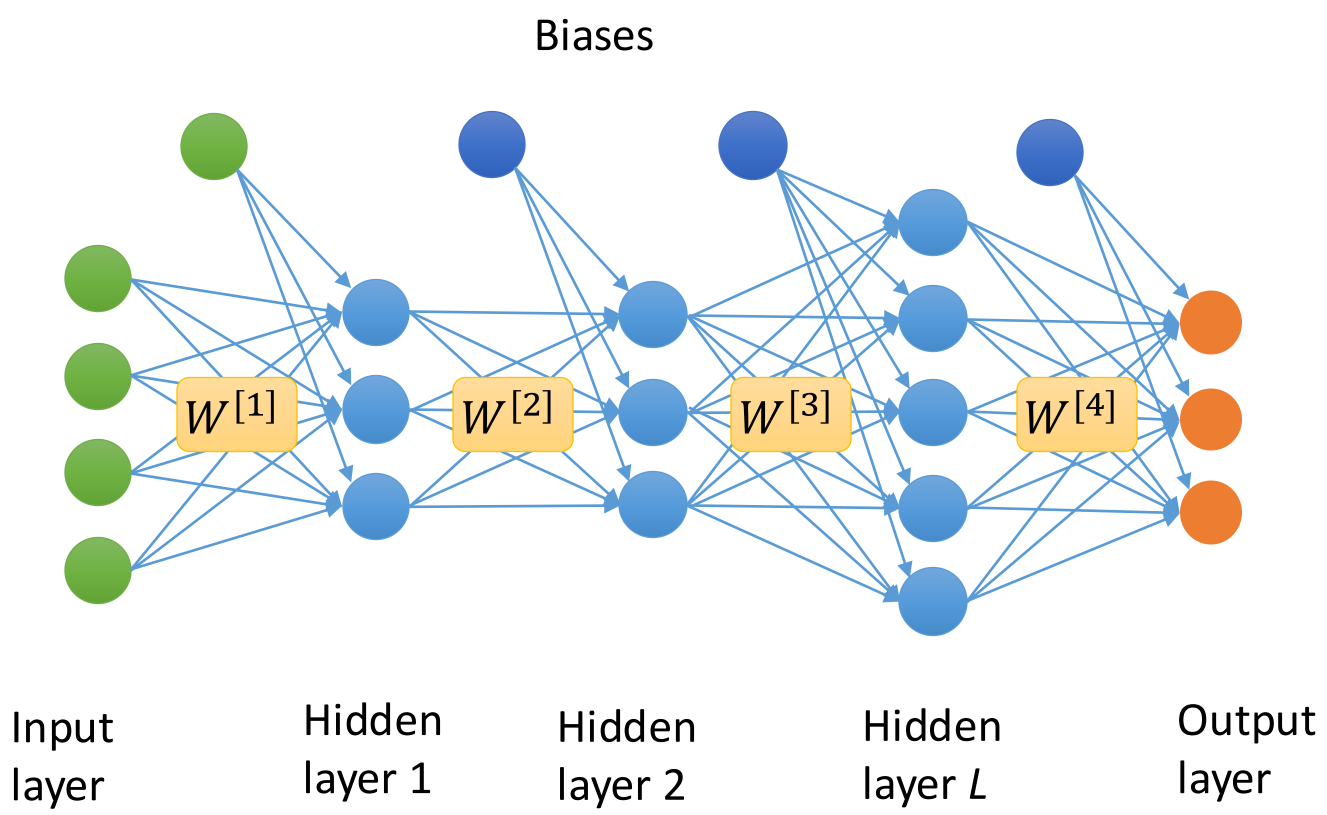
The hidden nodes - forward

- Same activation function at all hidden layers: g
  - *Logistic* or *ReLU* or…
- At node *j* in layer *k:*
  1. First sum of weighted inputs:
     - $z_j^{[k]} = \sum_{i=0}^{n^{[k-1]}} w_{i,j}^{[k]} a_i^{[k-1]}$
  2. Then $a_j^{[k]} = g(z_j^{[k]})$
  - (For the record: $a_i^{[0]} = x_i$)



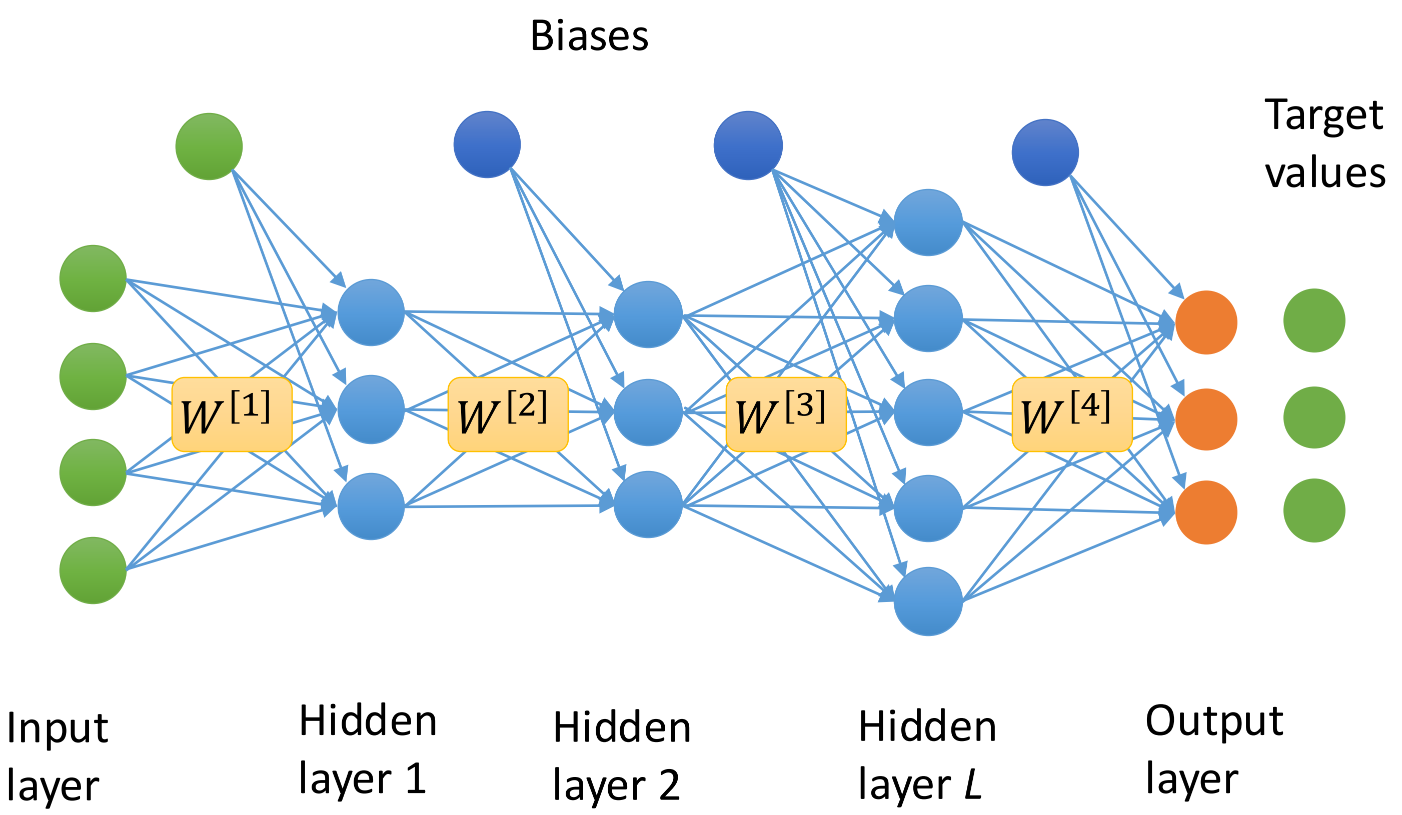
Forward

- Each hidden layer behaves like the hidden layer when there is only one
- The output layer
  - behaves like the output layer when there is only one hidden layer
  - How? depends on the task



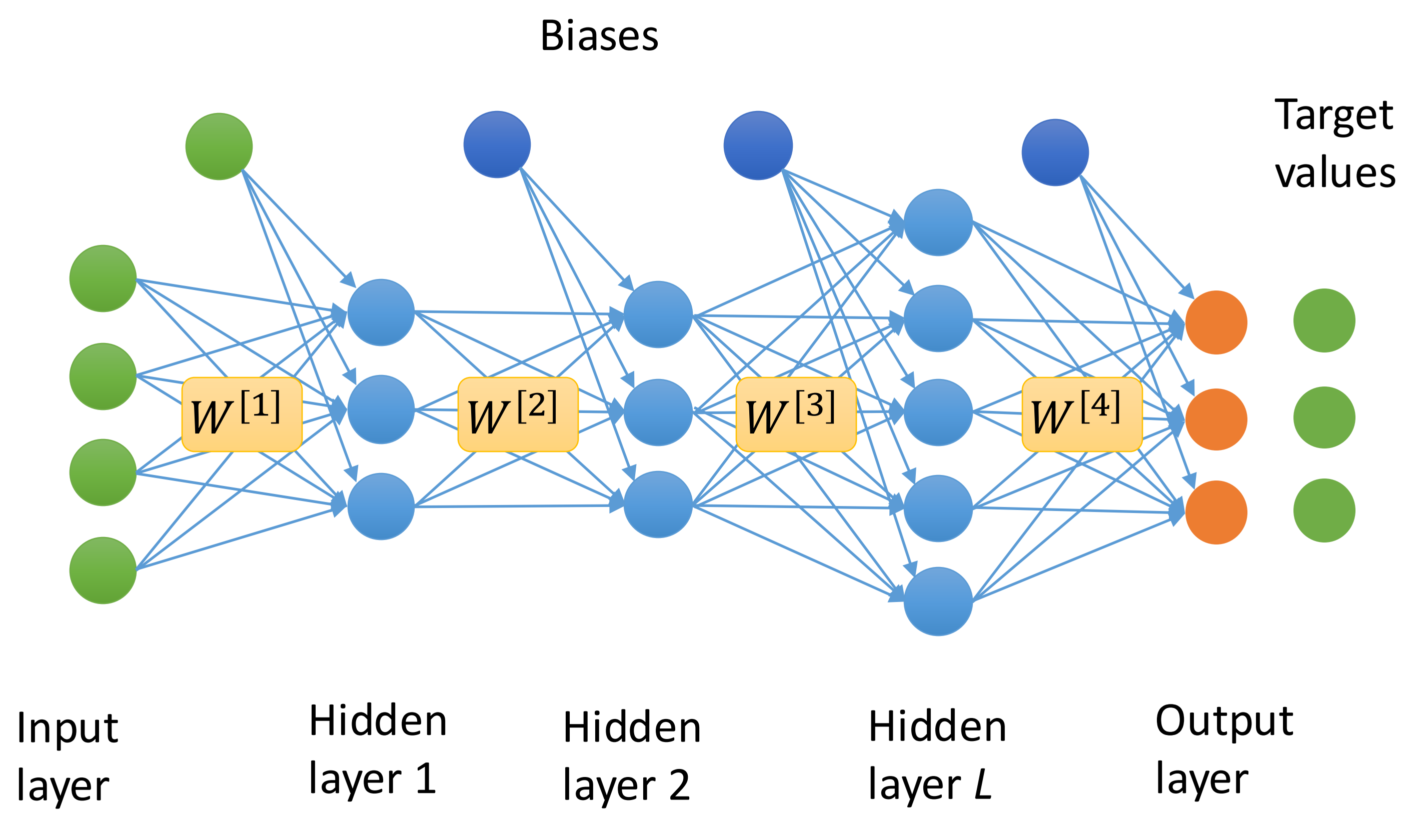
Update

- Compare output values to target values: $L(\boldsymbol{y}, \boldsymbol{t})$
- L is a loss-function
  - (There are alternative loss functions)
- If $\boldsymbol{y} = \boldsymbol{t}$ then $L(\boldsymbol{y}, \boldsymbol{t}) = 0$
  - No update
- The larger difference between $\boldsymbol{y}$ and $\boldsymbol{t}$, the larger loss and update



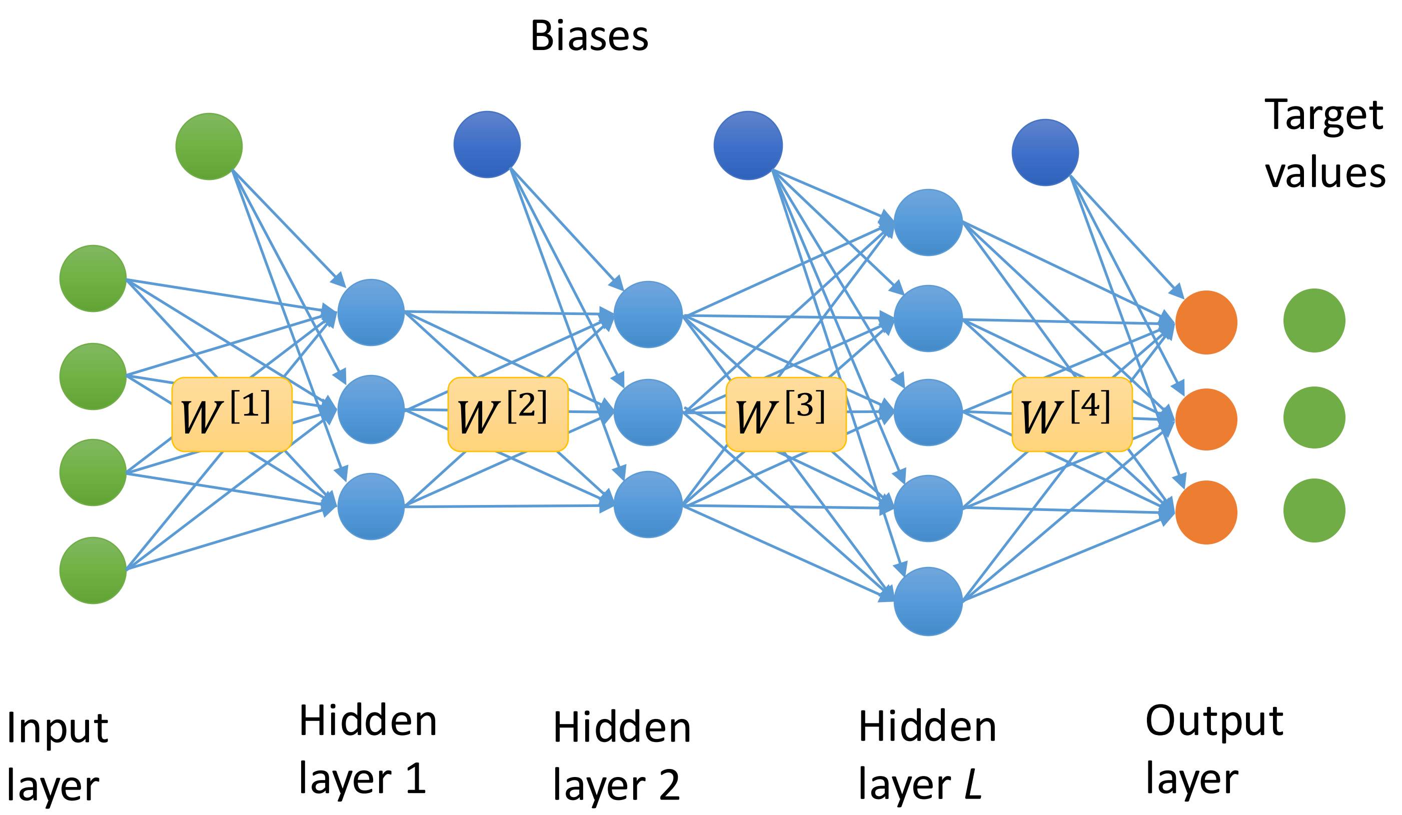
Update

- All weights are updated according to their contribution to the loss
- $w_{i,j}^{[k]} = w_{i,j}^{[k]} - \eta \frac{\partial}{\partial w_{i,j}^{[k]}} L(\mathrm{t}, y)$
- Use partial derivatives + chain rule for calculating $\frac{\partial}{\partial w_{i,j}^{[k]}} L(\mathrm{t}, y)$



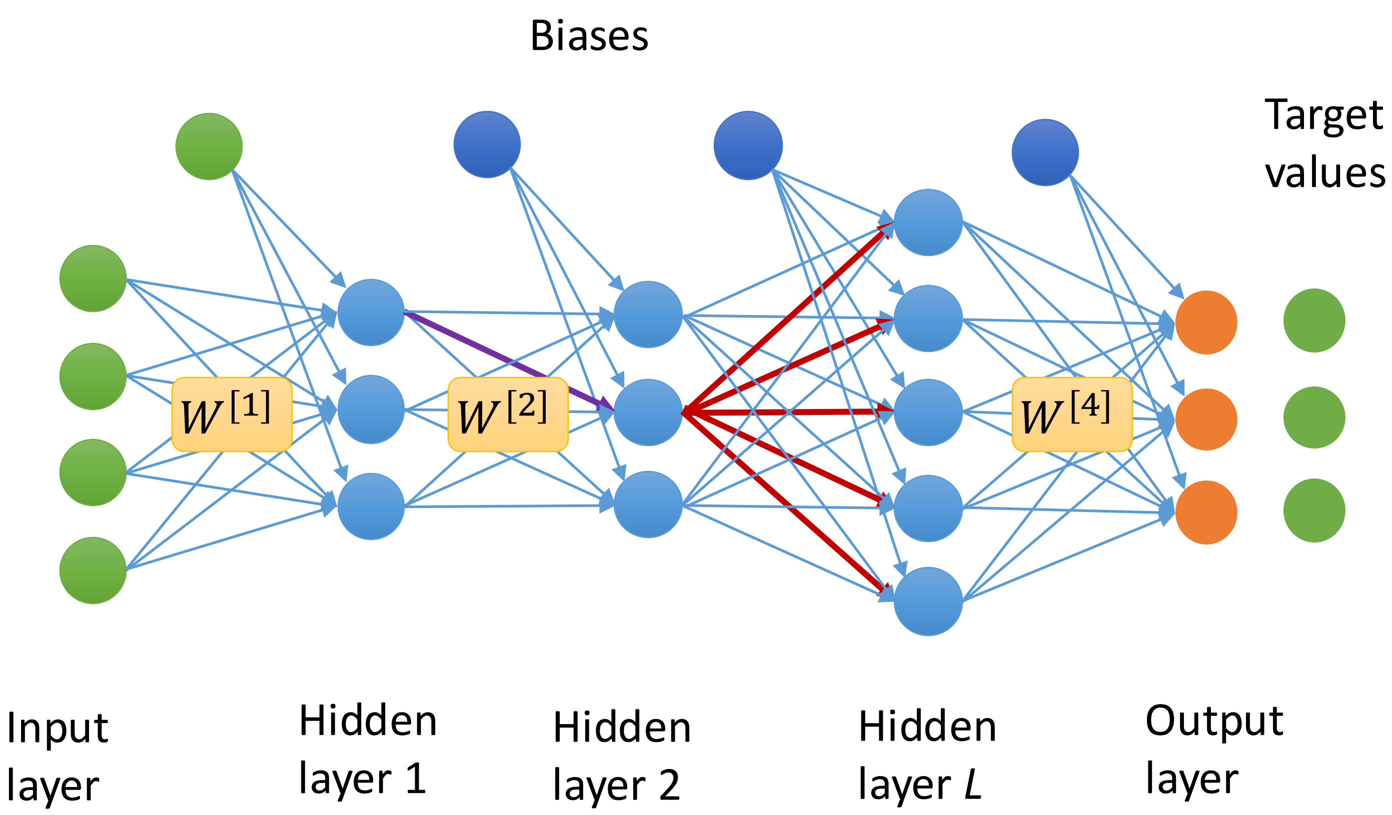
Update

- Activation function f at the last layer
  i.e., $y_j = f(z_j)$, $z_j = \sum_i w_{i,j}^{[4]} a_i^{[3]}$
- $$\frac{\partial}{\partial w_{i,j}^{[4]}} L(\mathrm{t}, y) = \underbrace{\frac{\partial}{\partial y_j} L(\mathrm{t}, y) \frac{\partial}{\partial z_j} f(z_j)}_{\delta_j^{[4]}} a_i^{[3]}$$
- (eventually:) $w_{i,j}^{[4]} = w_{i,j}^{[4]} - \eta \delta_j^{[4]} a_i^{[3]}$



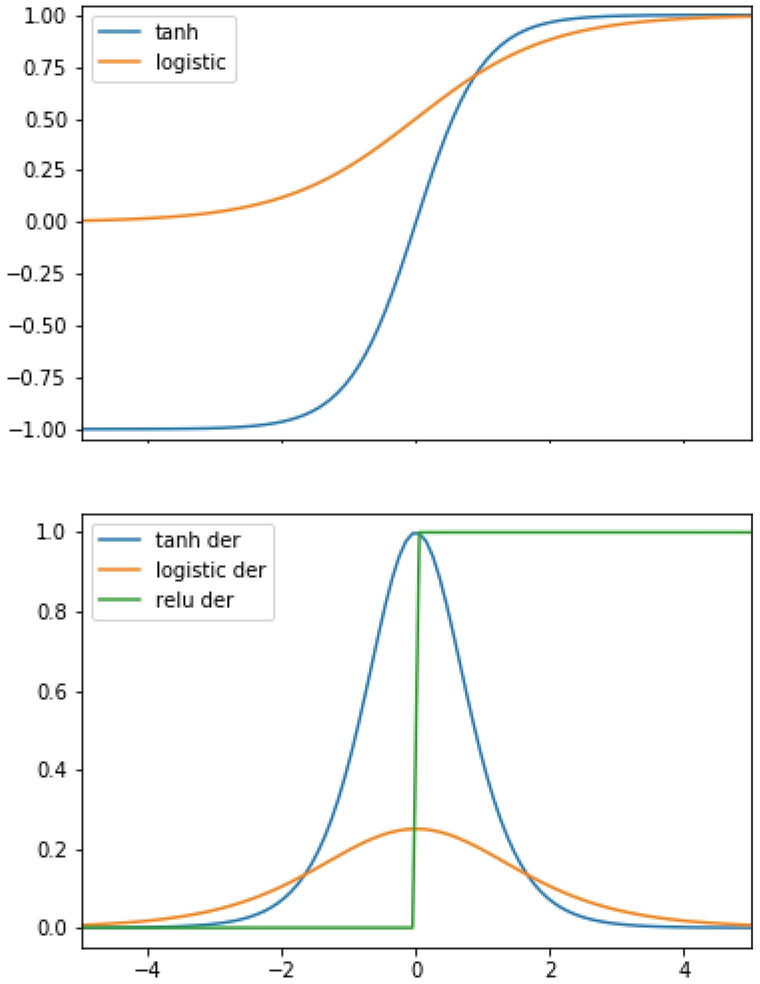
At the hidden layer

- At the hidden layer with activation function g,
  i.e., $a_j = g(z_j)$, $z_j = \sum_i w_{i,j}^{[k]} a_i^{[k-1]}$
- $\delta_j^{[k]} = \frac{\partial}{\partial z_j} g(z_j) \sum_m \delta_m^{[k+1]} w_{j,m}^{[k+1]}$
- (eventually:)
  $w_{i,j}^{[k]} = w_{i,j}^{[k]} - \eta \delta_j^{[k]} a_i^{[k-1]}$
- Follow the same recipe for all hidden layers (as we did last week)



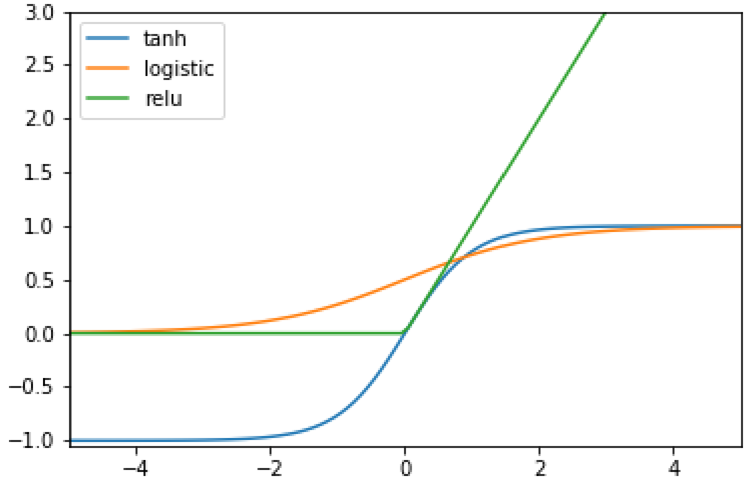
Vanishing gradient problem

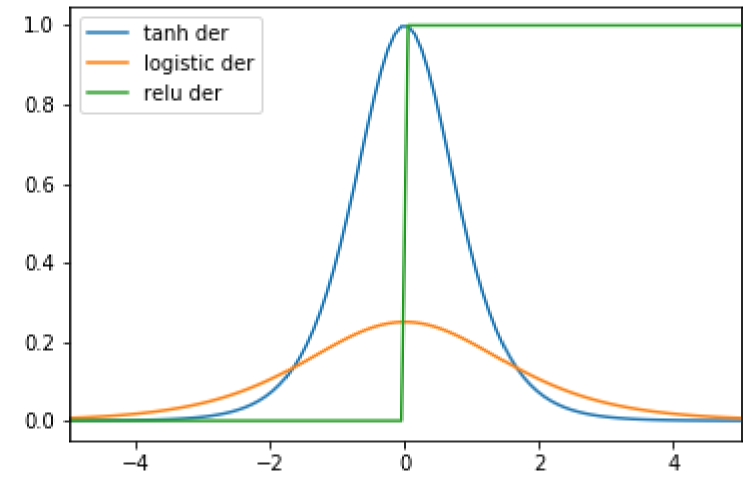
- The derivative of the logistic function (and tanh) is almost 0 except a small interval around 0
  - It gets easily saturated
- At each backward step, we multiply with the derivative of the activation function
- The gradient becomes close to 0.
  slow update, or no update at all.



Rectified linear unit, ReLU

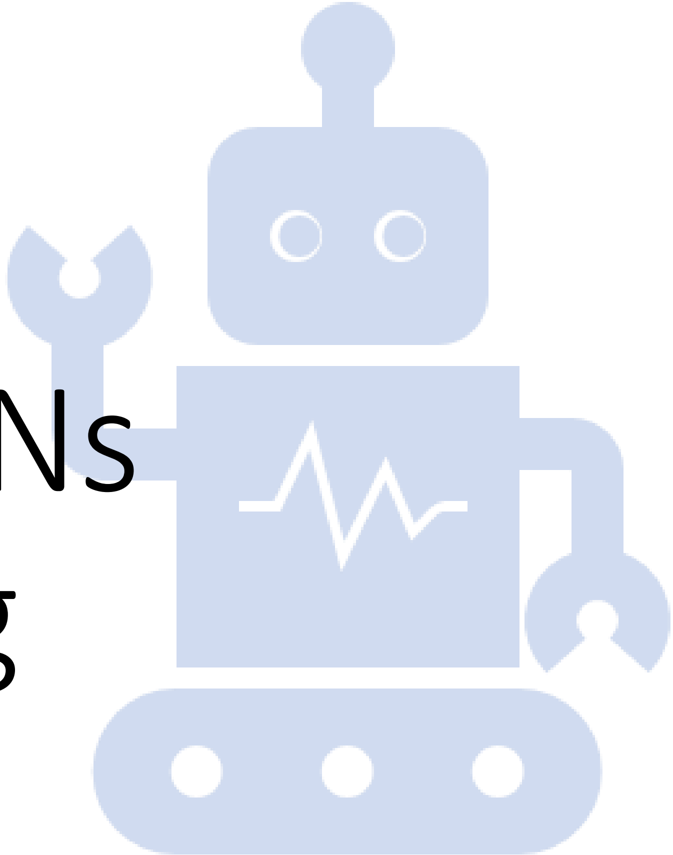
- Alternative activation functions in the hidden layers
- $ReLU(x) = \max(x, 0)$
- $ReLU'(x) = 1$ for $x > 1$
- $ReLU'(x) = 0$ for $x < 1$
- Use 0 for $ReLU'(0)$
- ReLU is the preferred method in deep networks
    - (There are various modified versions of ReLU)





12.3 Convolutional NNs and image processing

IN3050/IN4050 Introduction to Artificial Intelligence and Machine Learning



The MNIST data set

Domain

- Hand-written digits

https://en.wikipedia.org/wiki/MNIST_database

Labels

0
1
2
3
4
5
6
7
8
9

- Predict the correct hand-written digit
- There are 10 different classes
- 60,000 training images
- 10,000 test images
- Each picture 28x28

Image Net

	Images	Classes
The whole Image Net	15,000,000	22,000
The image classification challenge	Ca 1,500,000	1000

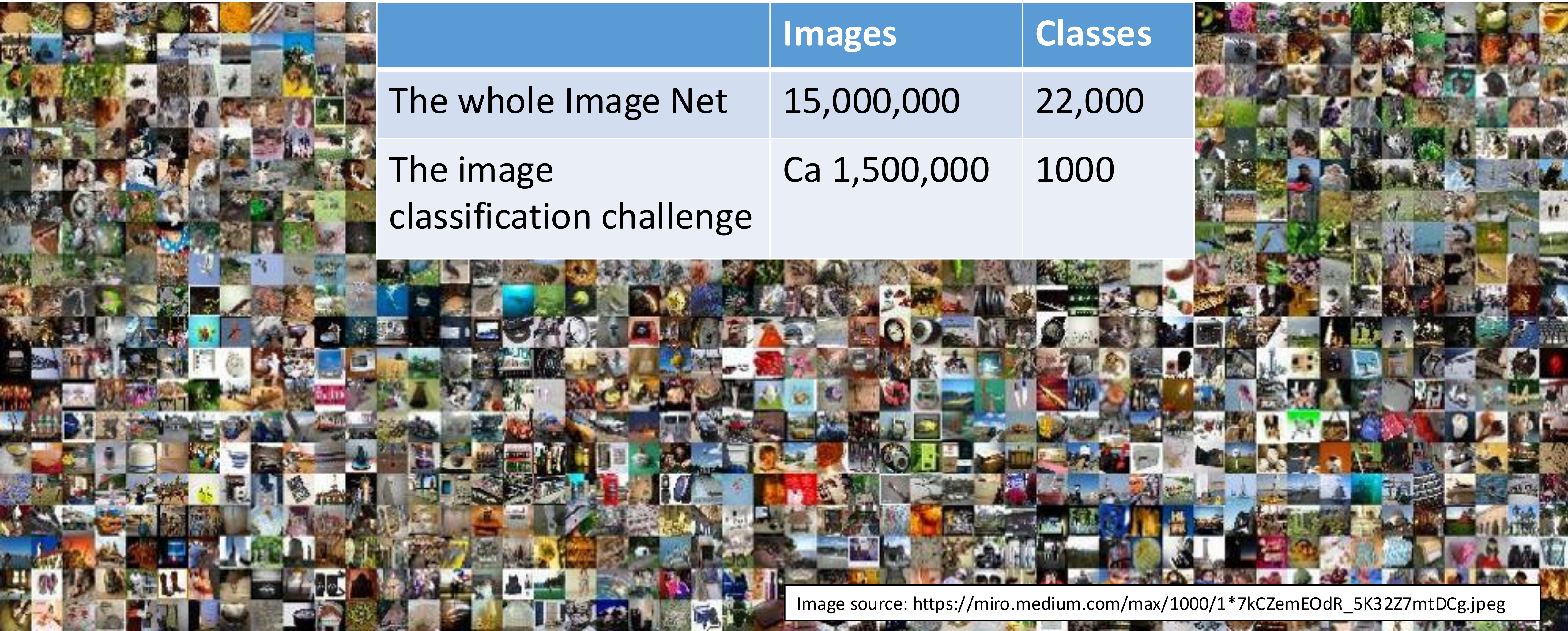
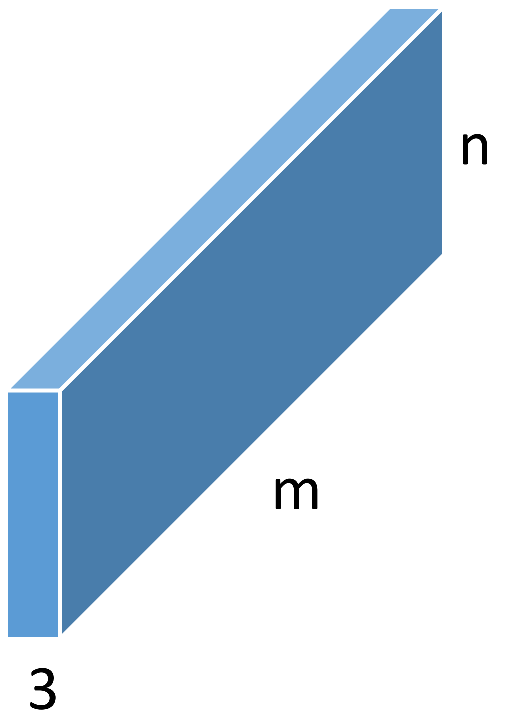


Image source: https://miro.medium.com/max/1000/1*7kCZemEOdR_5K32Z7mtDCg.jpeg

Image classification - input

- An image can be represented as $m \times n$ many pixels e.g., 28×28
- If it is in colors, each pixel can be three numbers, e.g., between 0 and 255,
  - e.g. (100, 50, 135)
- We can represent this in a neural net with $m \times n \times 3$ input nodes
- Challenge:
  - A small change to the image, e.g., rotation or dislocation changes the input values on each node
  - How can it then generalize?



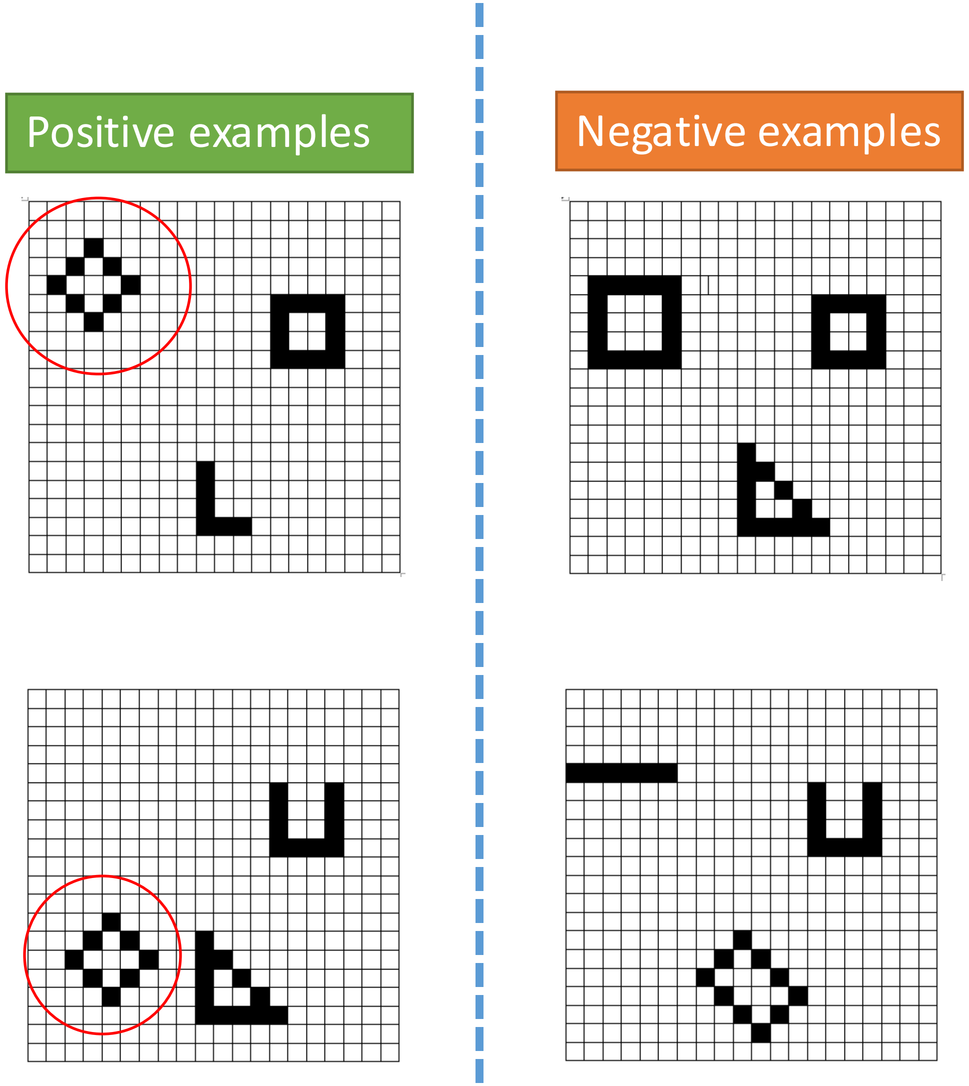
Warning!

- The following example is highly simplified and slightly misleading
- It does not show the convolutional network
- It considers a simplified problem

A very simplified example

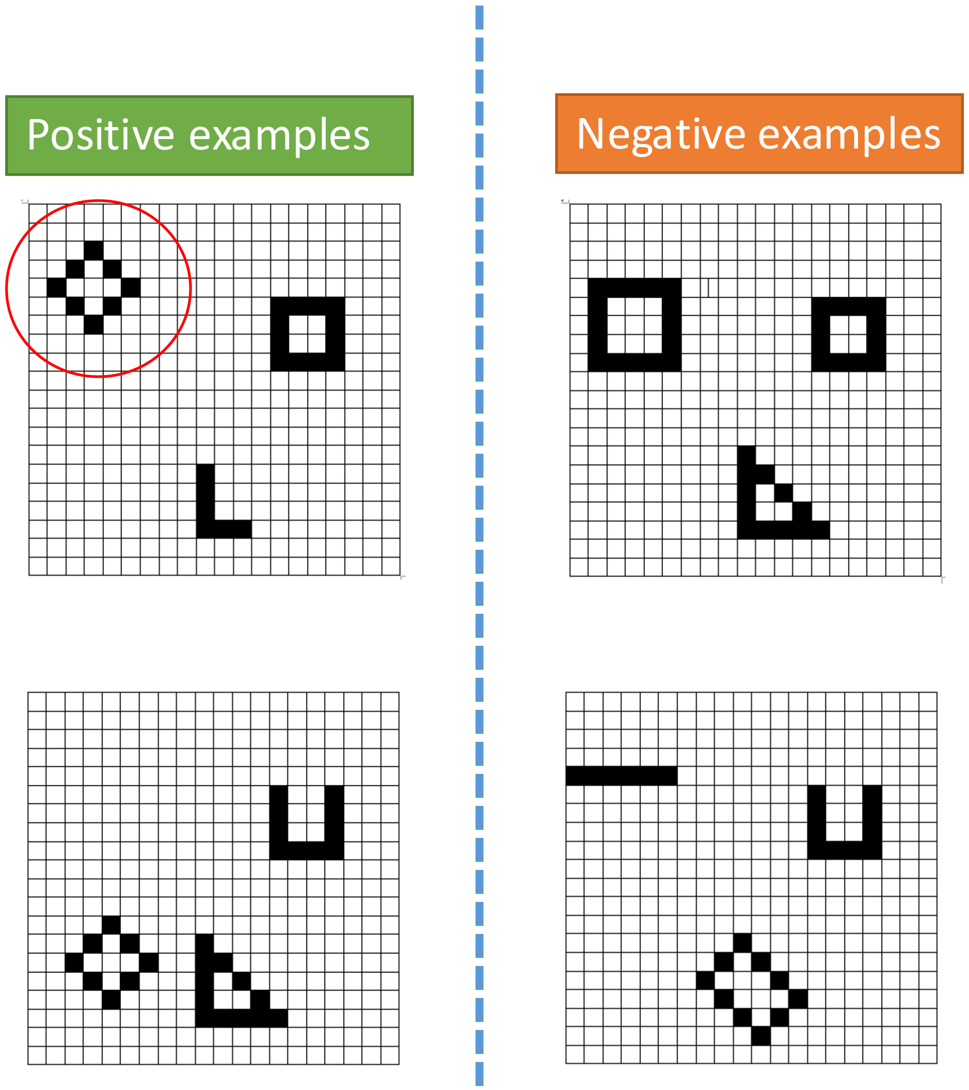
The problem

- Positive class if it contains at least one subfigure of exactly this pattern and size
- How can a classifier which takes pixels as input recognize this?
- There is no similarity in the pixels.



Approach for solution

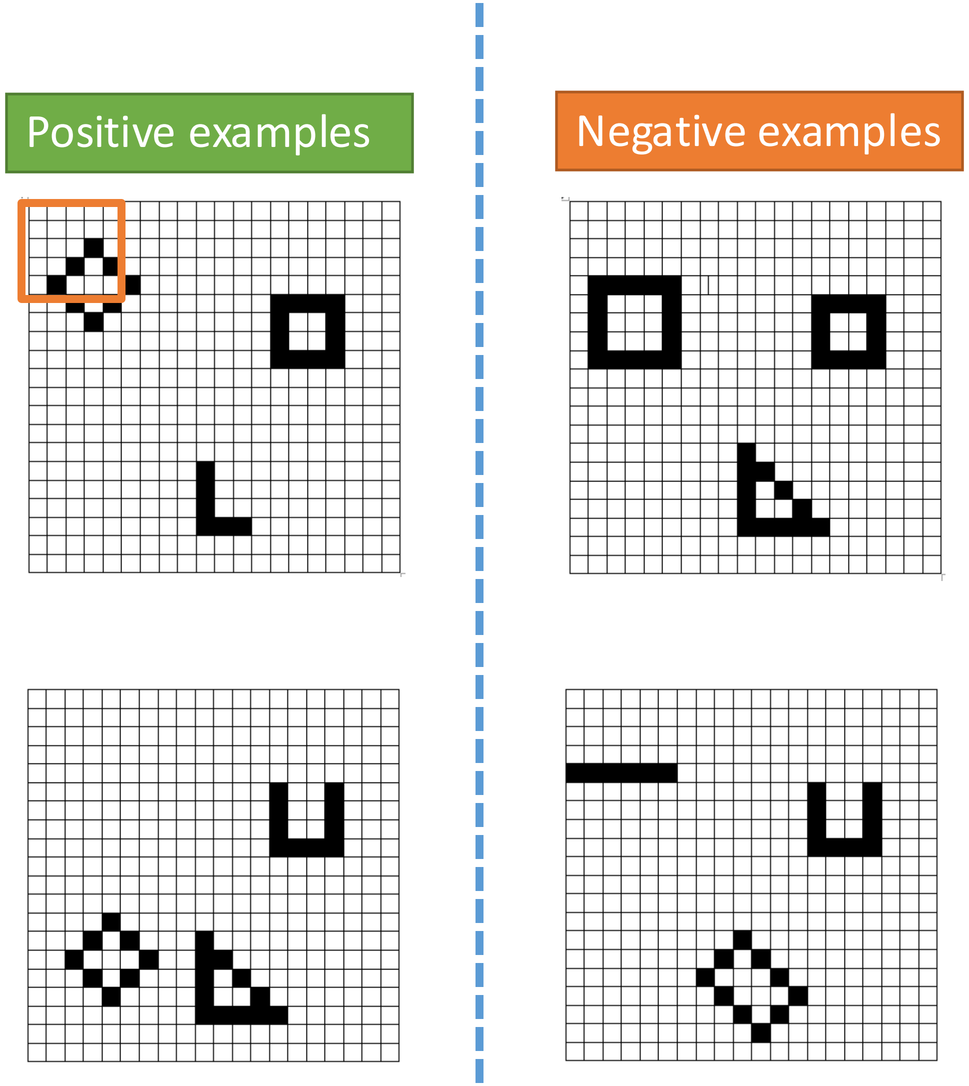
- Positive class if it contains at least one subfigure of exactly this pattern and size
- Split the task in two:
  - For each 5x5 subpicture, decide whether it has this pattern or not
  - Answer whether the picture has at least one such pattern

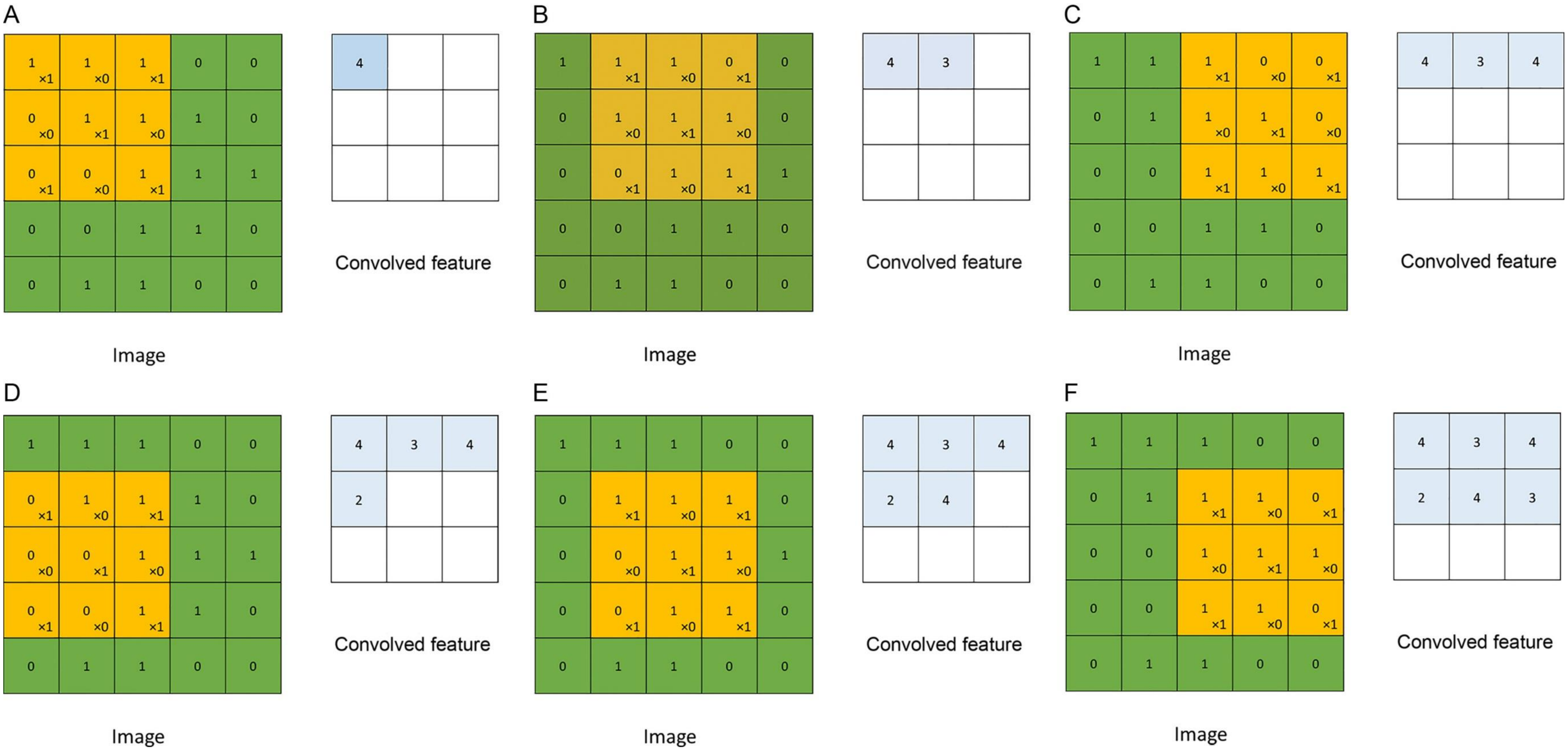


The filter

- We slide a 5x5 window over the picture:
  - Report the result each time

- We can solve this task:
  - Determine whether the picture contains exactly this 5x5 pattern

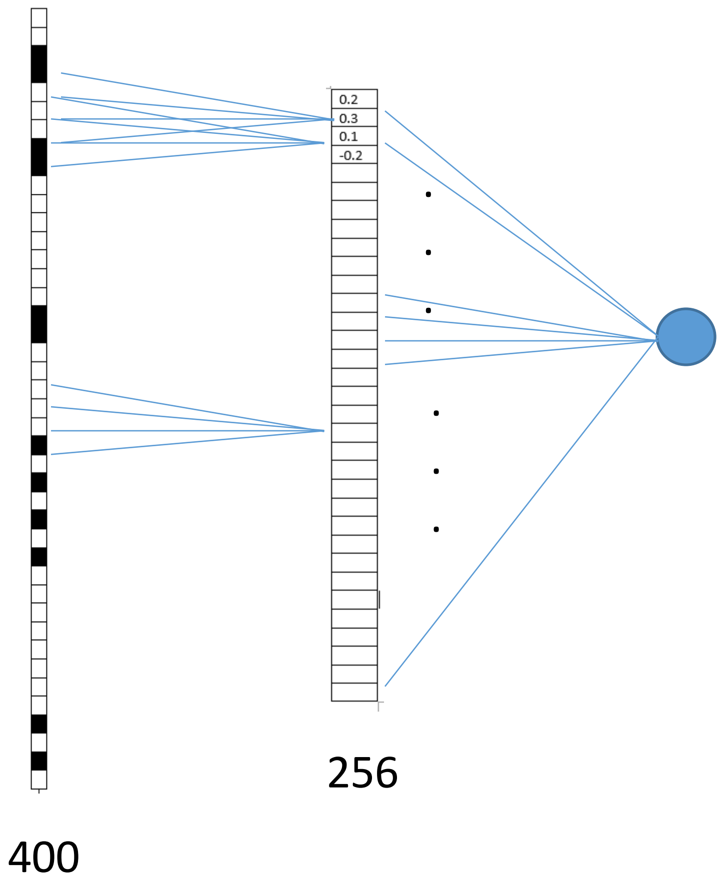




https://www.sciencedirect.com/topics/computer-science/convolutional-layer

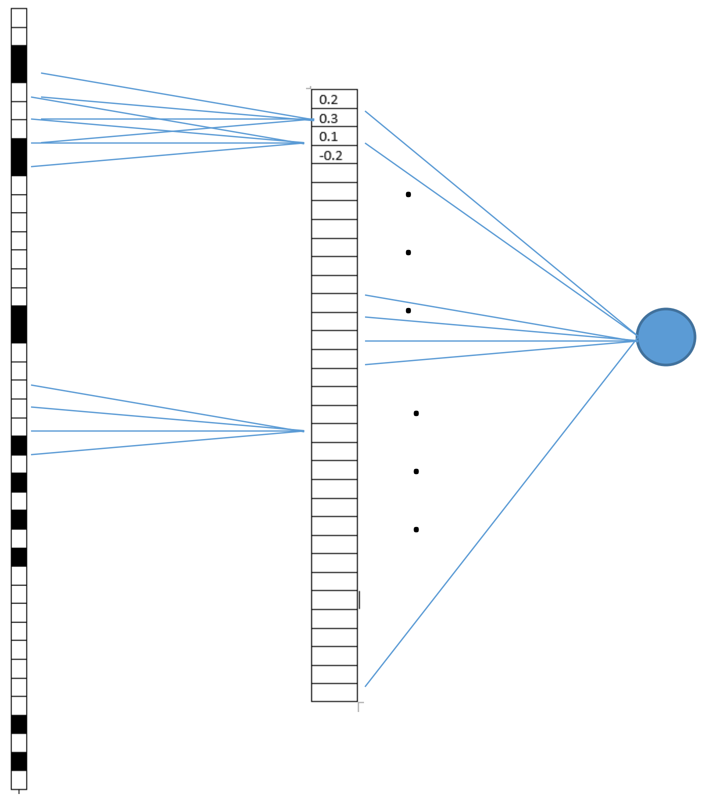
The network

- 400 (=20x20) input nodes
  - One per pixel
- 256 = (16x16) hidden nodes
  - One per 5x5 rectangle
- 25 edges to each hidden node
  - One for each pixel in the node
  - (Not fully connected)
- Fully connected hidden layer to output node



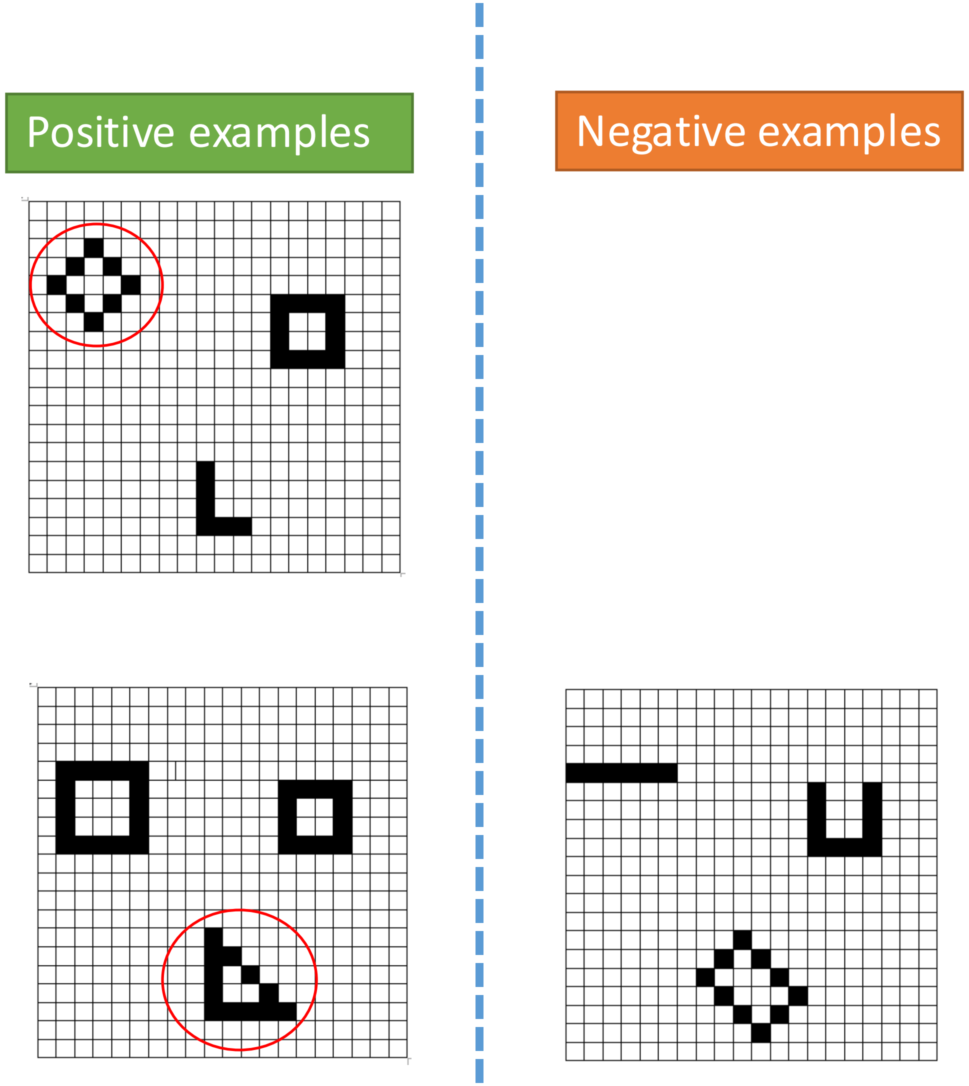
Example continued

- First: There are only
  - 25x256 connections in the first layer, and not 400x256
  - 256 connections in the second layer
- The clue: Each hidden node should learn the same:
  - $w_{i,j} = w_{i+k,j+k}$
  - We use the same (26x1) weight matrix for all hidden nodes, which we update through backprop.
  - This matrix is called a filter.



A more complex task

- Positive class if it contains any of the two 5x5 patterns
- What now?
- We can have several filters
  - Each of them can learn one specific pattern
  - We can put a numeric calculation on top of them in the final fully connected layer
    - ''More than 3 of pattern 1 and 2 or less of pattern 2, none of pattern 3,etc.''
- We can also handle colors by having three cells per pixel



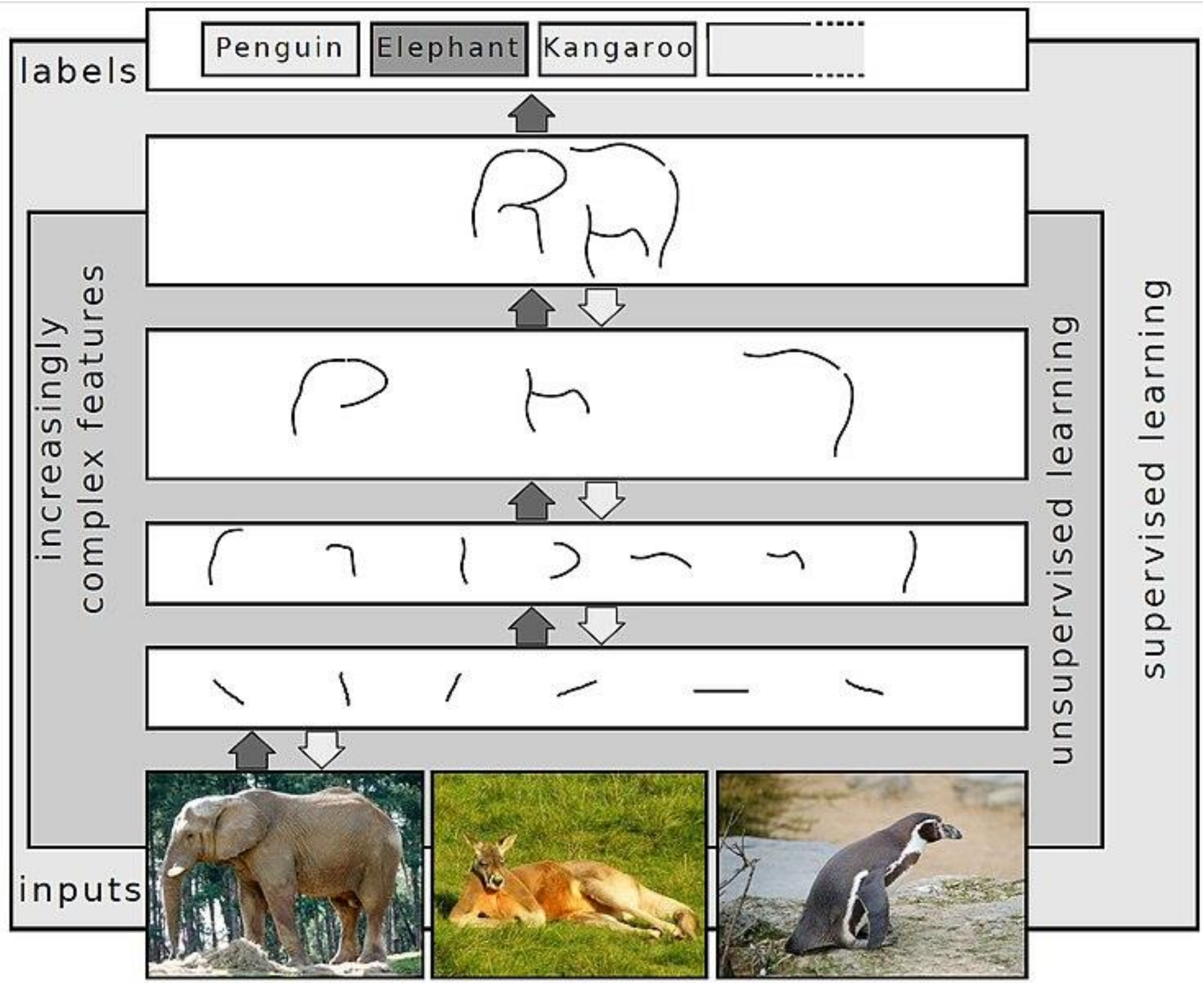
Towards convolutional networks

So far

- Does not handle:
  - Rotations
  - Variation in size:
    - We want to identify the same pattern across various sizes
  - Small distortions
    - Including perspective
  - Etc.

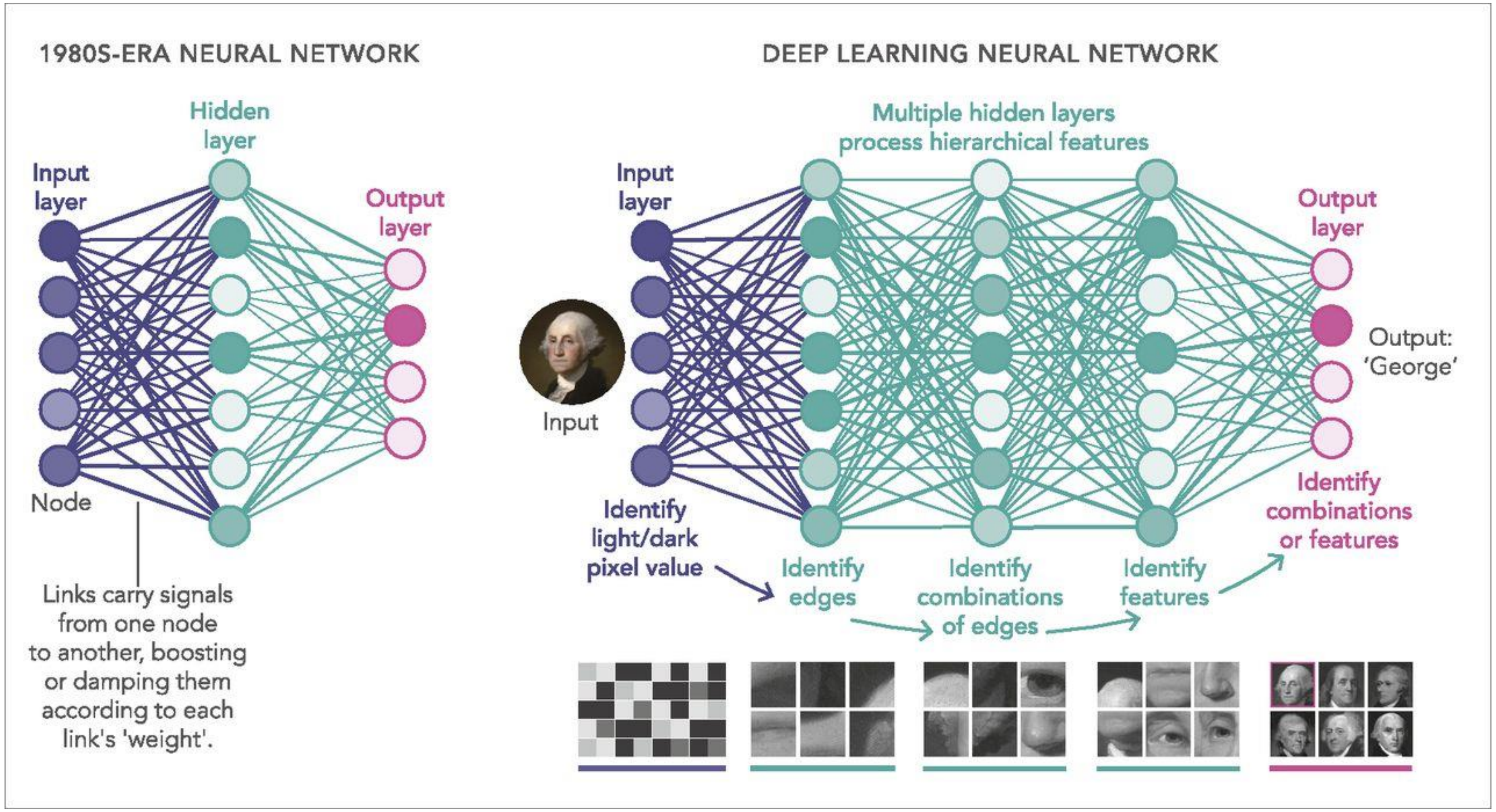
The convolutional network

- Use filters as indicated
- Several layers
- Recognize
  1. Simple patterns, e.g., 5x5 pixels
  2. Lines, curves
  3. Contours
  4. Figures
  5. Etc.



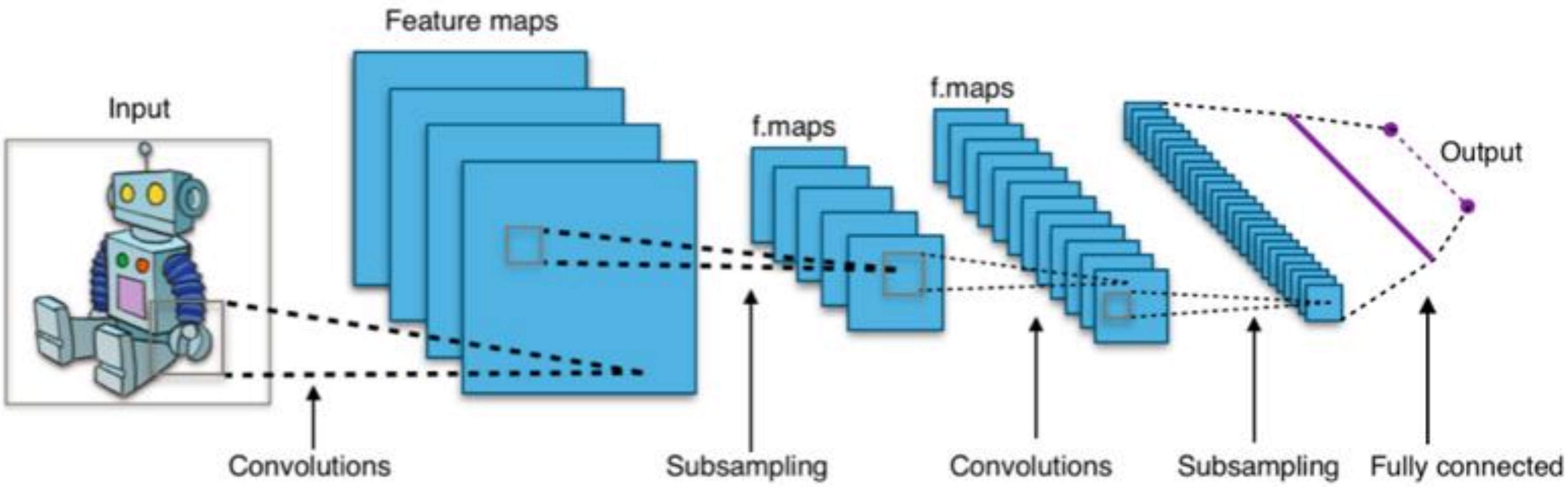
https://en.wikipedia.org/wiki/File:Deep_Learning.jpg

“Neural network” models of AI process signals by sending them through a network of nodes analogous to neurons.



M. Mitchell Waldrop PNAS 2019;116:4:1074-1077

Typical architecture (simplified)

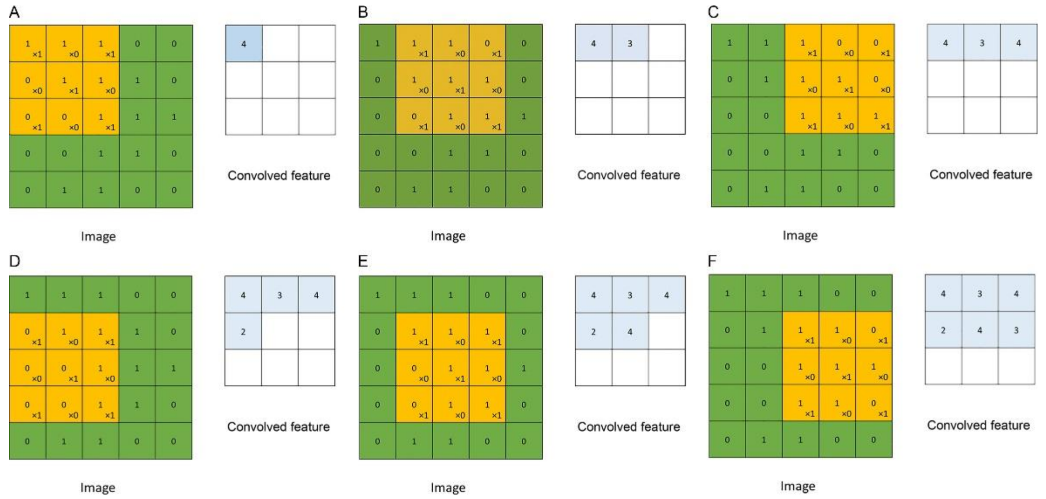


https://en.wikipedia.org/wiki/Convolutional_neural_network

- Four types of layers:
  1. Convolutions (filters)
  2. Pooling (Down sampling)
  3. Fully-connected layers
  4. ReLU layers

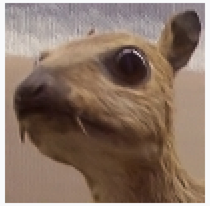
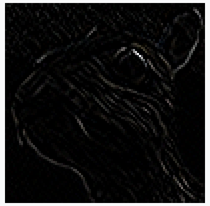
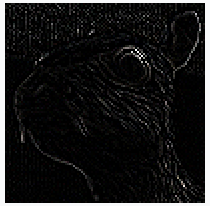
Filters

- We have already considered them.
- Several layers.
- They are called filters because there is a tradition of using such (man-made) filters in image processing
- In the ConvNets, the filters are learned



https://www.sciencedirect.com/topics/computer-science/convolutional-layer

Traditional man-made filters

Operation	Kernel ω	Image result g(x,y)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Gaussian blur 5 × 5 (approximation)	$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	
Unsharp masking 5 × 5 Based on Gaussian blur with amount as 1 and threshold as 0 (with no image mask)	$\frac{-1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & -476 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	

https://en.wikipedia.org/wiki/Kernel_(image_processing)

Why convolutional neural networks work?

Less Parameters and Parameter Sharing

Better off stacking simple functions in depth and many layers than learning complex functions in one layer

Each convolution layer detects localized patterns over all input tensor

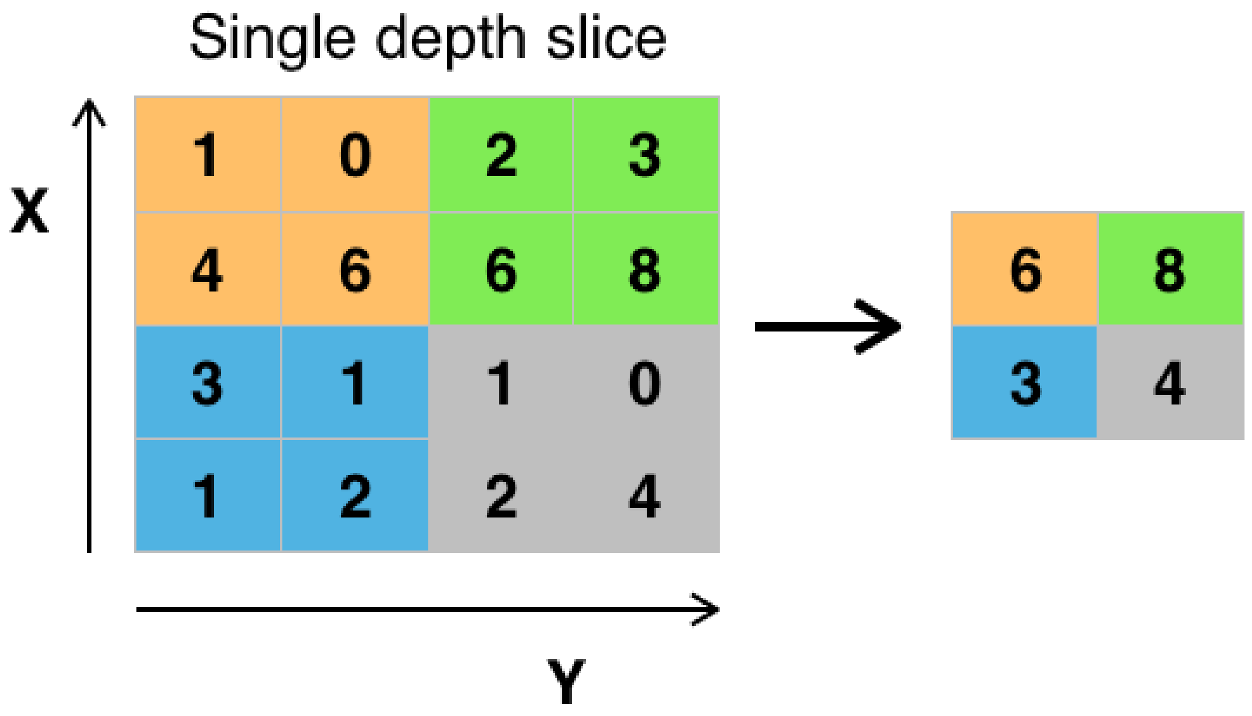
Inner product is a similarity measure: detect patterns in input similar to filter

Detection is global for fully connected, localized for convolutions

Convolution detects patterns in a translation-invariant manner!

Detects similar patterns in different sizes/scales/view angles/…

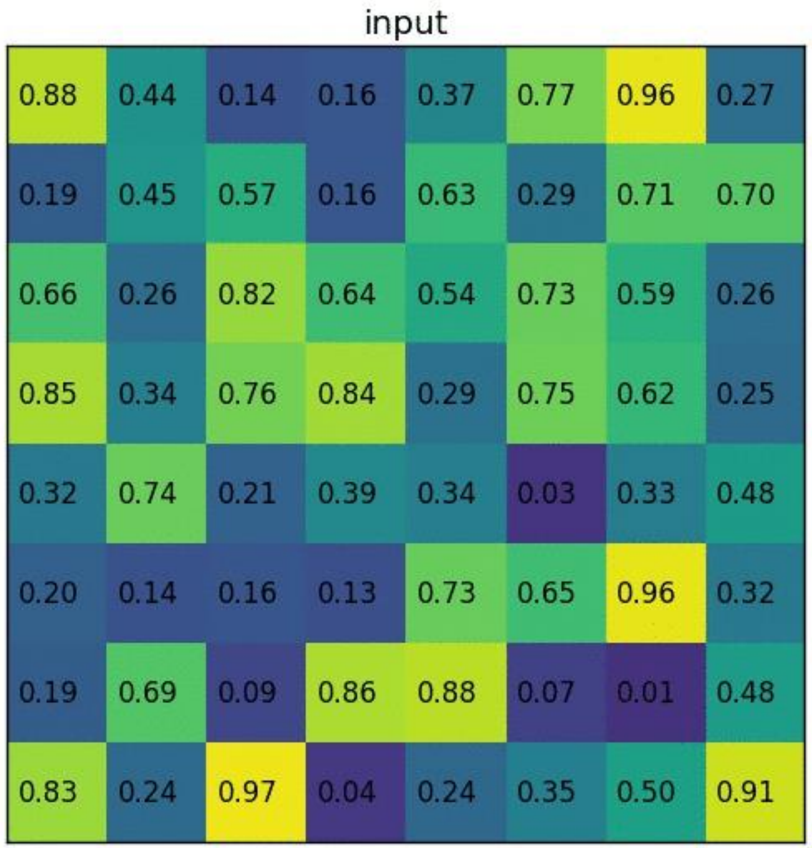
Pooling



https://en.wikipedia.org/wiki/File:Max_pooling.png

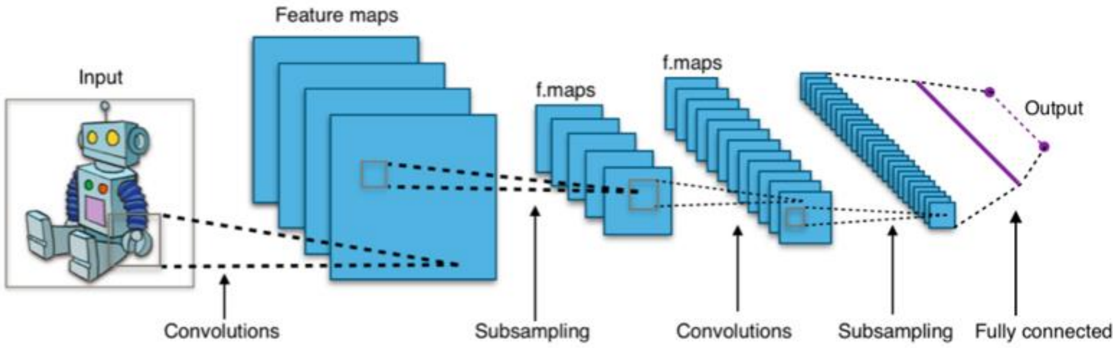
Pooling

- Variants:
  - Max pooling
  - Average



https://en.wikipedia.org/wiki/File:RoI_pooling_animated.gif

Typical architecture (simplified)



https://en.wikipedia.org/wiki/Convolutional_neural_network

- How many layers?
- How many nodes?
- The relationship between the layers?

- This is more an art than science
- The models become deeper and deeper
- More than 100 layers

Why Residual Connections?

Identity+ optional non-linearity

Never worse than identity

Filters can add non-linearities layer by layer

Gradually learn a more complex representation

ResNets [He et al., CVPR16]

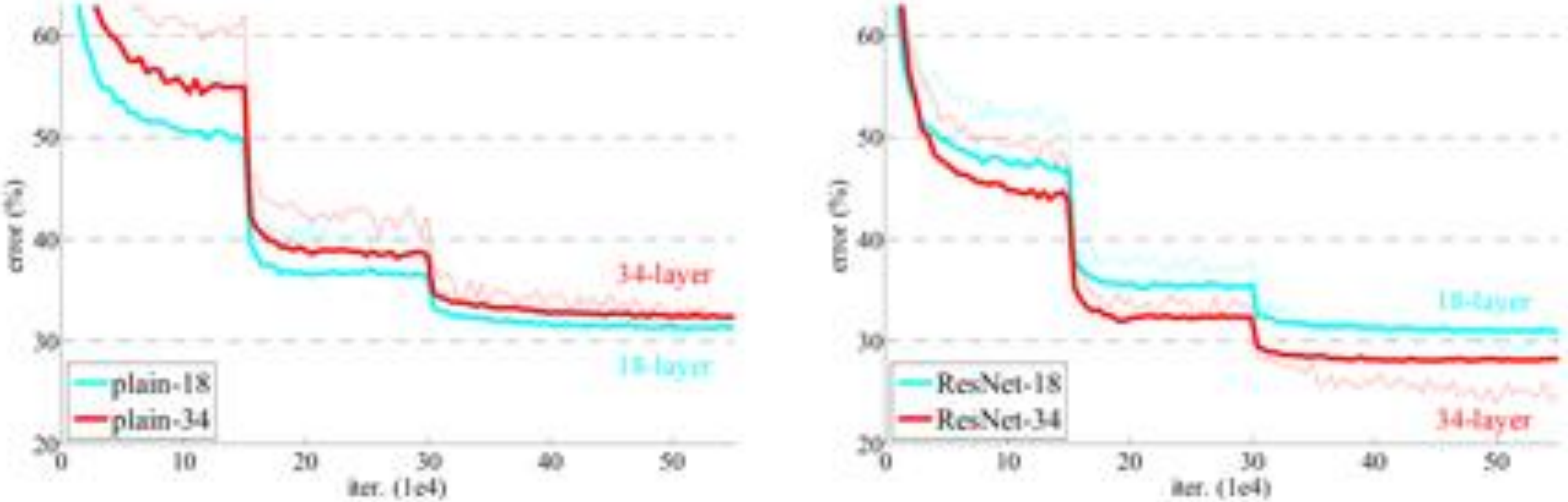


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

Convolutional networks for text classification

- ConvNets originally developed for images

- Also applied to texts:

  - The window is 1-dimensional: *n* words, or *n* characters from a sentence
  - It exploit that the same word or sequence of *n* words may occur at various places
  - The networks are normally not very deep.

Learning Convolutional NNs

- The learning is done as for other multi-layer neural nets by backpropagation.
  - We know how to do that.
- In frameworks, like TensorFlow, PyTorch, etc.,
  - we only have to specify the forward architecture.
  - The framework takes cares of the backpropagation
  - We must specify various hyperparameters, though, like learning rate and regularization.

Properties of deep convolutional networks

"Traditional" ML	Deep Convolutional NNs

"Traditional" ML

- The models had strong inductive biases, e.g., linear models
- The researchers had to put much effort into feature engineering, to fit the data to the model to get results
  - (We have mostly avoided that by using simple datasets.)
- Man-made filters in an example

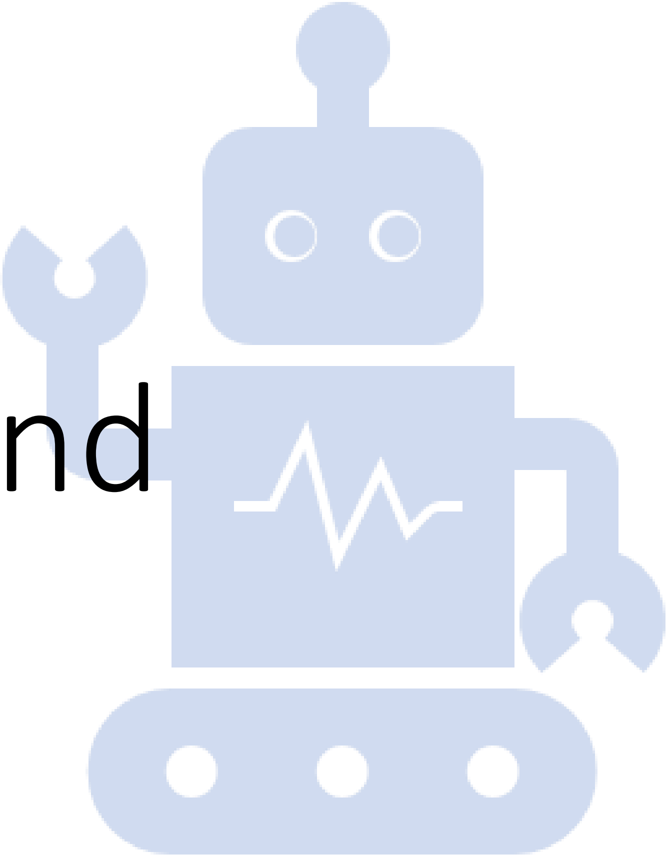
Deep Convolutional NNs

- The model can learn the representations, e.g., it learns the filters.
- More flexible models
  - (but still some inductive bias in the architecture chosen)
- Demands more training data
- More machine power

12.4 Recurrent NNs and language processing

IN4050 Introduction to Artificial Intelligence

and Machine Learning



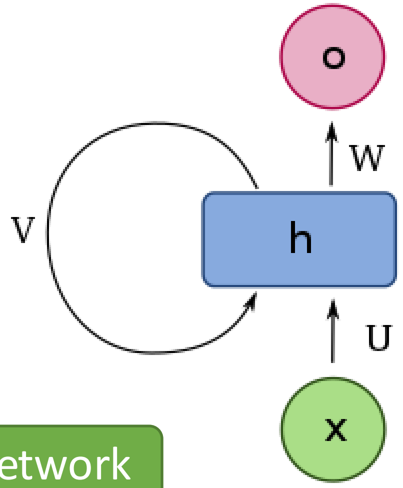
RNNs

- Recurrent Neural Networks

- Applications in Language Technology, examples

Recurrent neural nets

- Model sequences/temporal phenomena
- A cell may send a signal back to itself – at the next moment in time



The network

Image source: https://en.wikipedia.org/wiki/Recurrent_neural_network

Recurrent neural nets

- The state h_t in the cell at time t is determined by:
  - Input x_t at time t
  - State h_{t-1} at time $t-1$

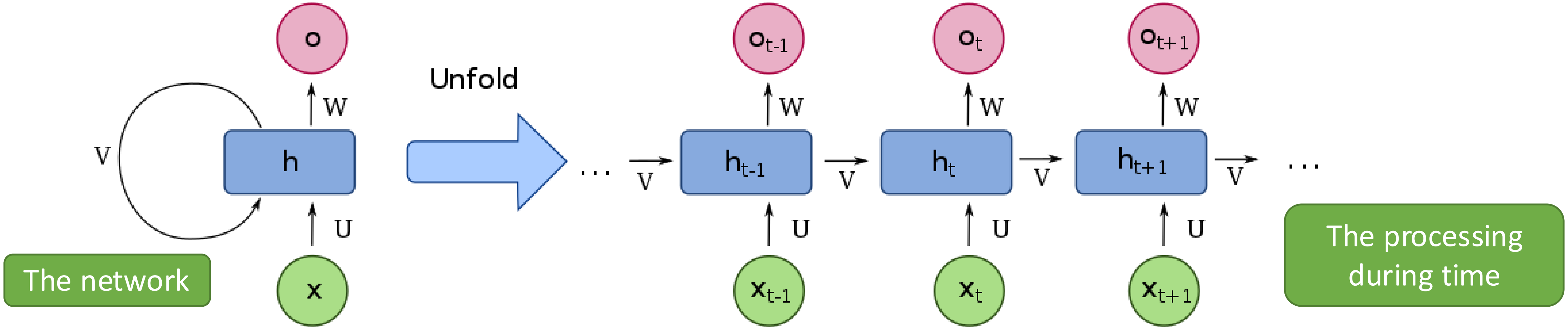
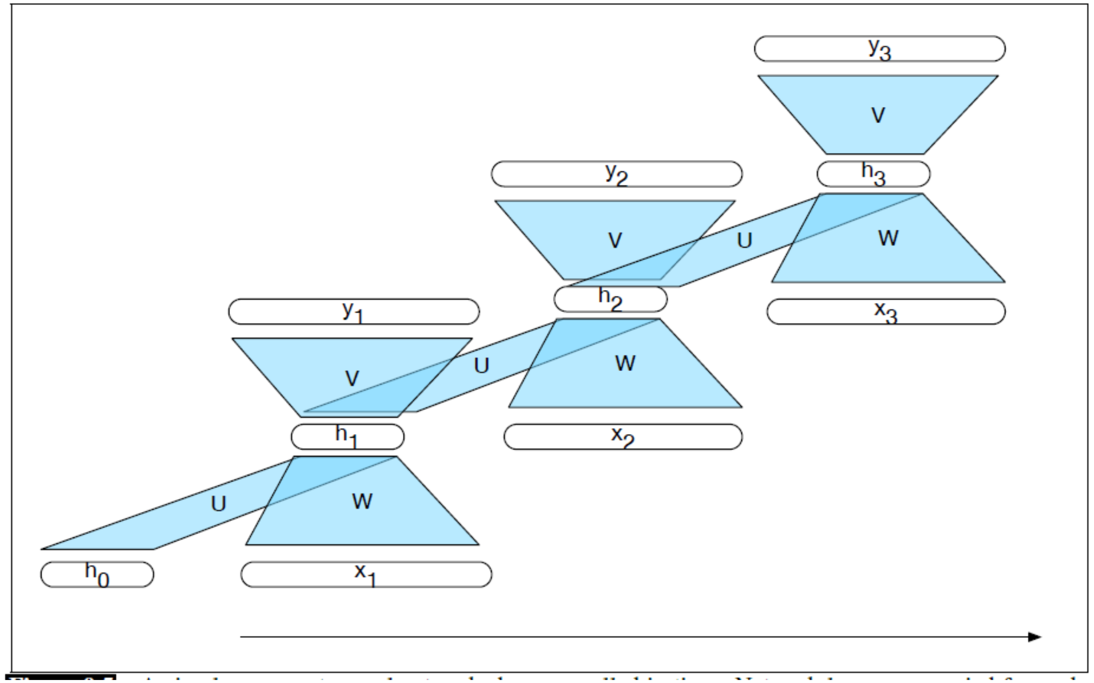


Image source: https://en.wikipedia.org/wiki/Recurrent_neural_network

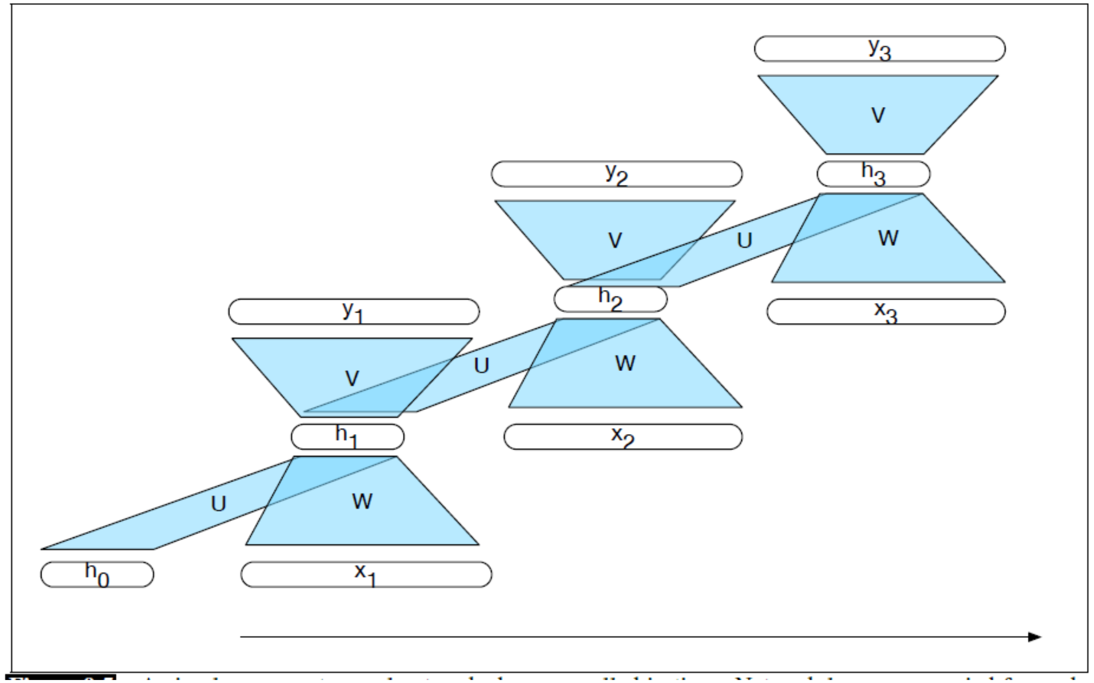
Forward



- $x_1, x_2, \dots, x_n$ is the input sequence
- Each U, V and W are edges with weights (matrices)
- Forward:
  1. Calculate h_1 from h_0 and x_1.
  2. Calculate y_1 from h_1.
  3. Calculate h_i from h_{i-1} and x_i, and y_i from h_i, for $i = 1, \dots, n$

From J&M, 3.ed., 2019 (Speech and Language Processing)

Forward

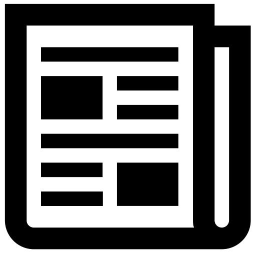


- The same $U, W, and V$ for all layers
- $\boldsymbol{h}_t = g(\boldsymbol{h}_{t-1}U + \boldsymbol{x}_t W)$
- $\boldsymbol{y}_t = f(\boldsymbol{h}_t V)$
- g and f are activation functions
- f is often softmax, i.e.,
  - $\boldsymbol{y}_t = softmax(\boldsymbol{h}_t V)$
- (There are also bias terms which we didn't include in the formulas)

From J&M, 3.ed., 2019 (Speech and Language Processing)

Language model

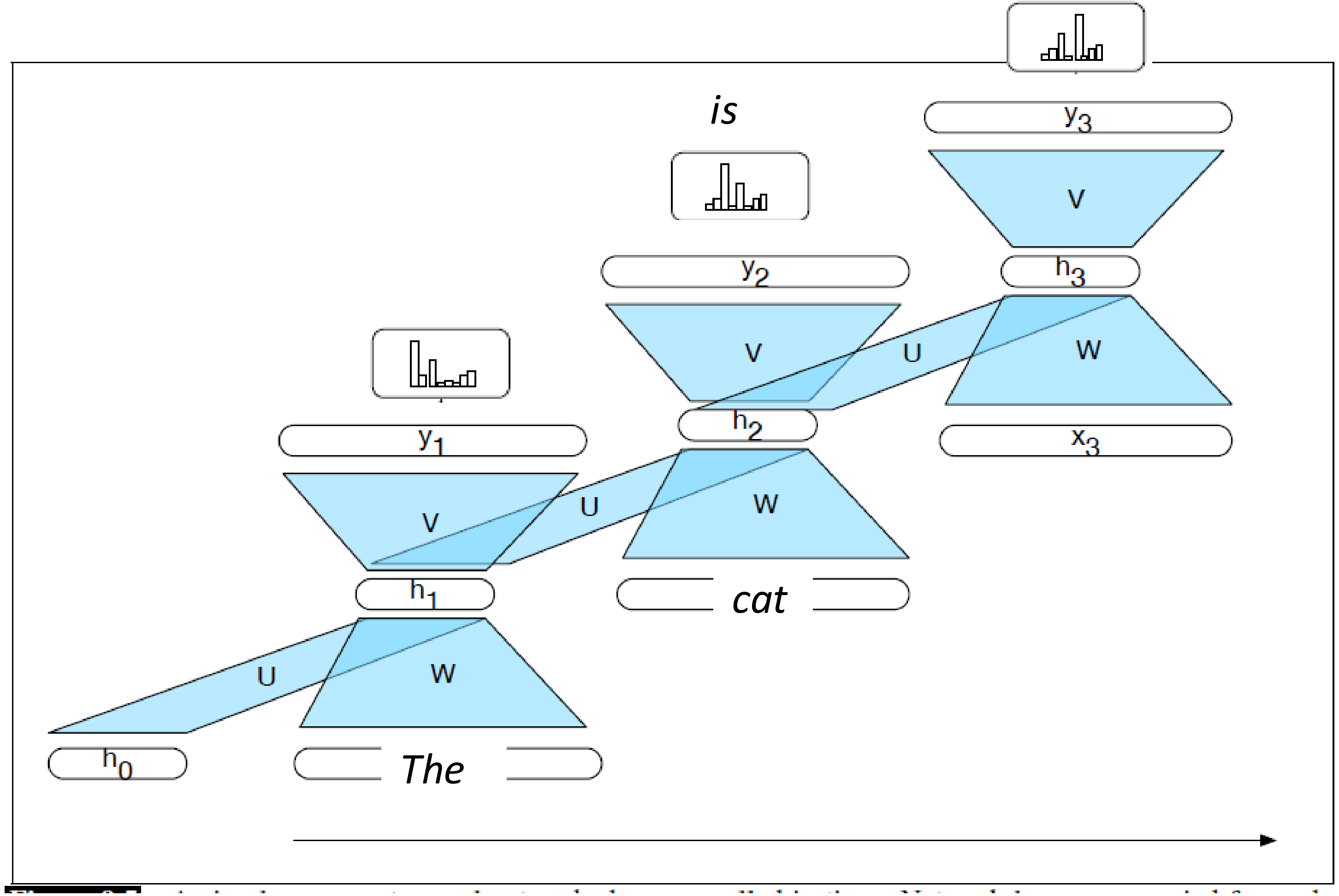
The cat is on the …



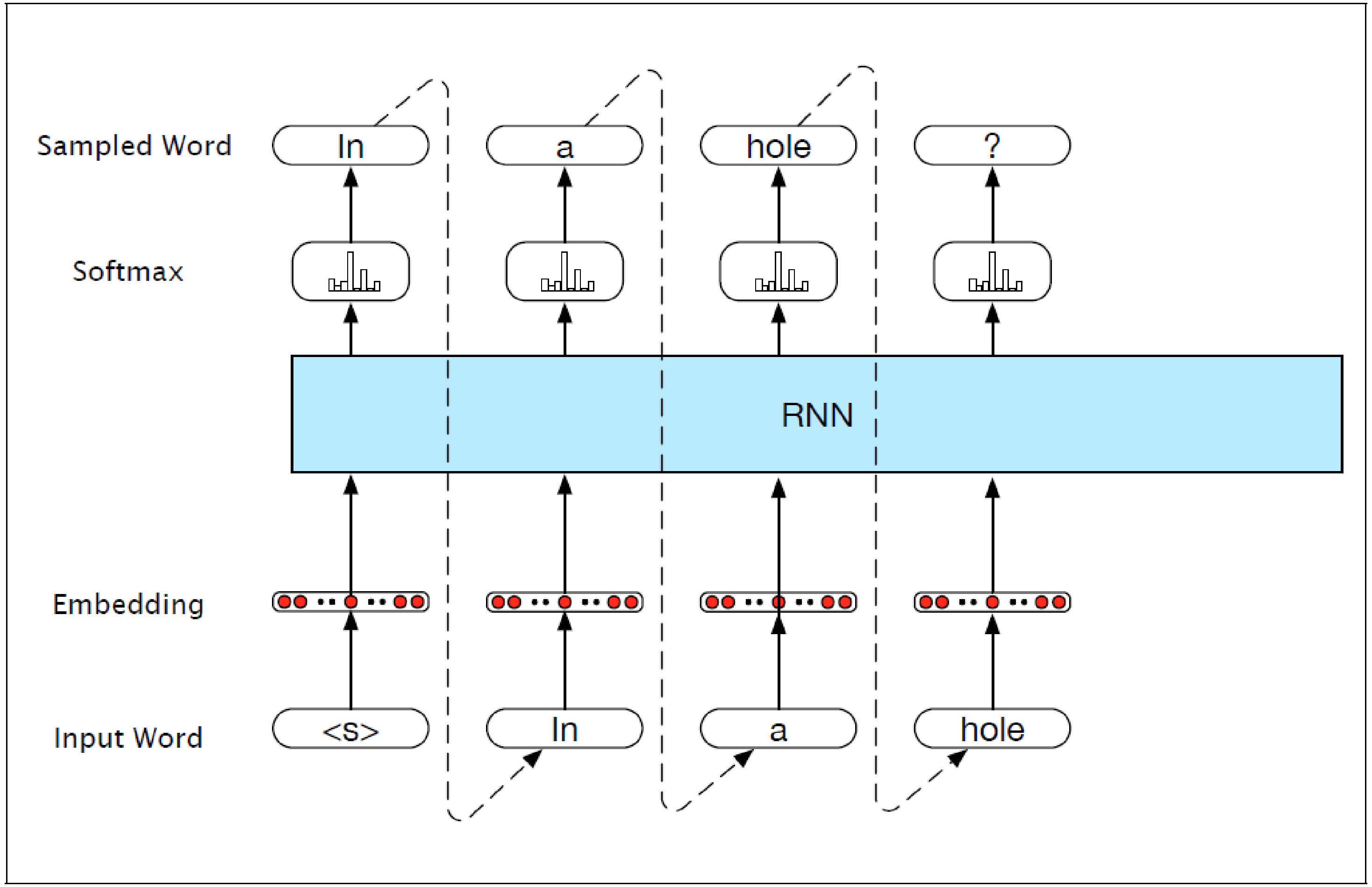
- Task: Predict the next word!
- Labels:
  - A vocabulary of words:
    - *aardvark, …, mat, …, roof, …*
    - E.g., 100,000 words/labels
- Domain:
  - Finite sequences of words

- Properties:
  - Billions of training instances
  - Supervised learning
    - But you do not have to hand-label the training data.

RNN Language model



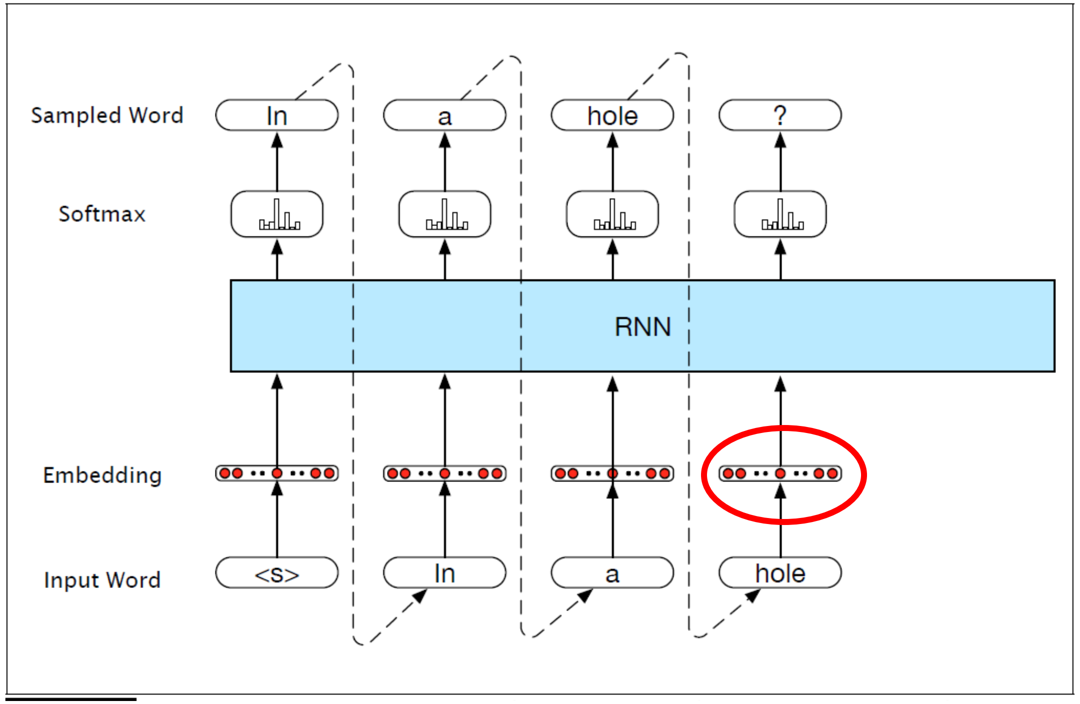
Autoregressive generation



- We could guess a whole sentence
- More interesting if we have some preconditions in addition

From J&M, 3.ed., 2019

Word embeddings



From J&M, 3.ed., 2019

- How should words be represented?
- What is this?

One-hot encoding

- A word is a categorical feature.

- We assume a vocabulary of e.g., 100,000 different words.

- We could use a ''one-hot'' encoding (''one out of *n*''):
  - $(0, 0, 0, \dots, 1, \dots, 0)$
  - One 1 and 99,999 many 0-s
  - Different words, different positions

- But:
  - Inefficient, so many features and weights
  - Nothing in common among similar words

Embeddings

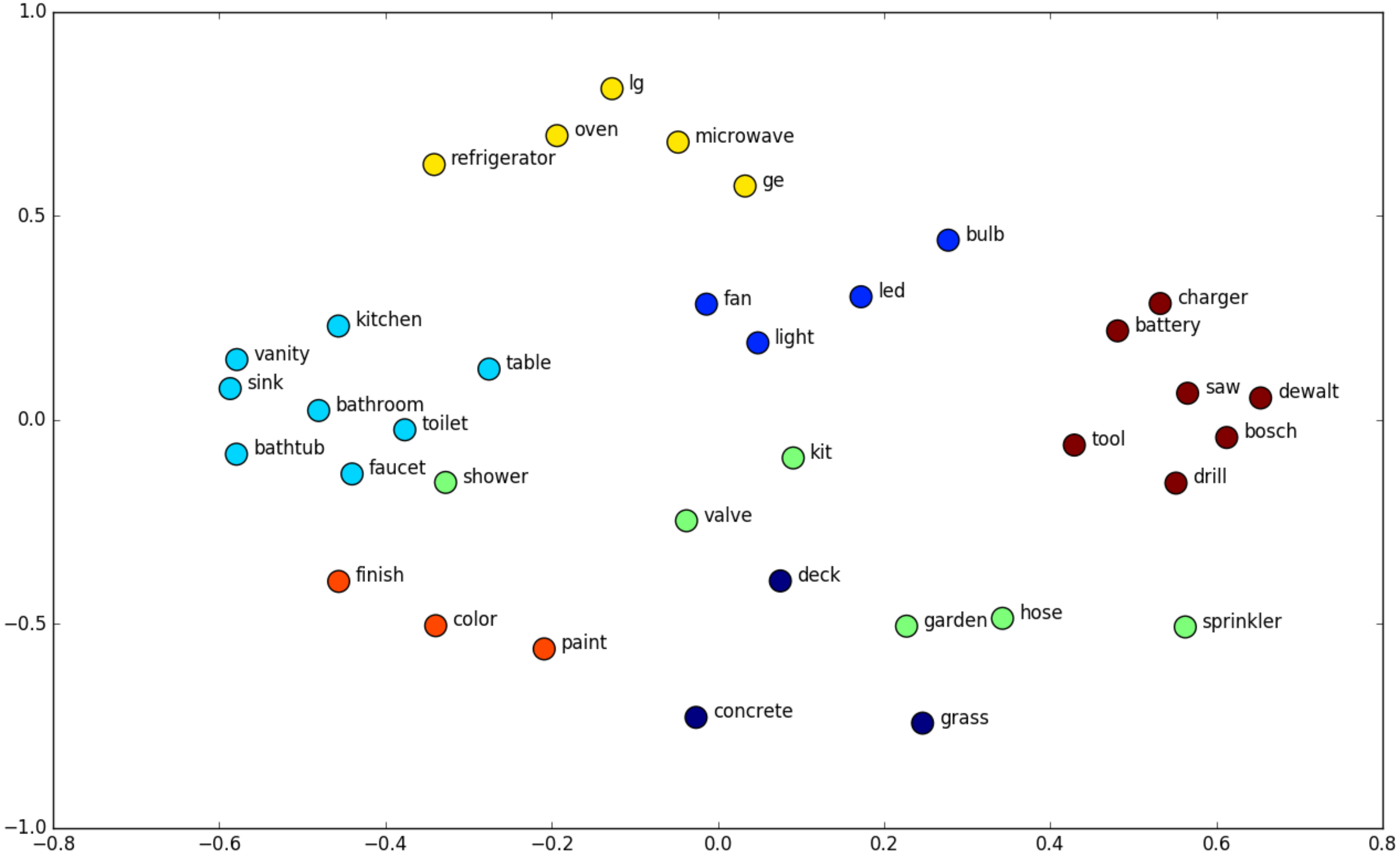
- Represent each word with a vector of
  - Reals
  - A fixed number of dimensions, e.g., 100 (typically, between 50 and 300)

- Try to get similar vectors for similar words
  - Words can be considered similar if they occur in similar positions, e.g.,
    - *Milk*, *water, soda* can all occur with *She drank ____, a glass of ___*, etc.

- These **embeddings** can be learned from a language modeling task:
  - Predict whether a neighboring word can occur together with this word

Embeddings applied

- These embeddings can be used for semantic similarities
- The figure is a projection of the 100 or so dimensions into 2 dimensions.
- http://vectors.nlpl.eu/explore/embeddings/en/



https://www.shanelynn.ie/get-busy-with-word-embeddings-introduction/

Machine Translation

- Bi-text
  - Text translated between two languages
  - The translated sentences are aligned into sentence pairs

- Machine learning based translation systems are trained on large amounts of bitext:
  - English sentence 1 – Norwegian translation 1
  - English sentence 2 – Norwegian translation 2
  - …
  - English sentence n – Norwegian translation n

Encoder-decoder based translation

- Concatenate the two sentences in a pair:
  - source sentence_<\s>_target sentence
    <\s>: end-of-sequence token

- Train an RNN on these concatenated pairs

- Apply by reading a source sentences and predict a target sentence

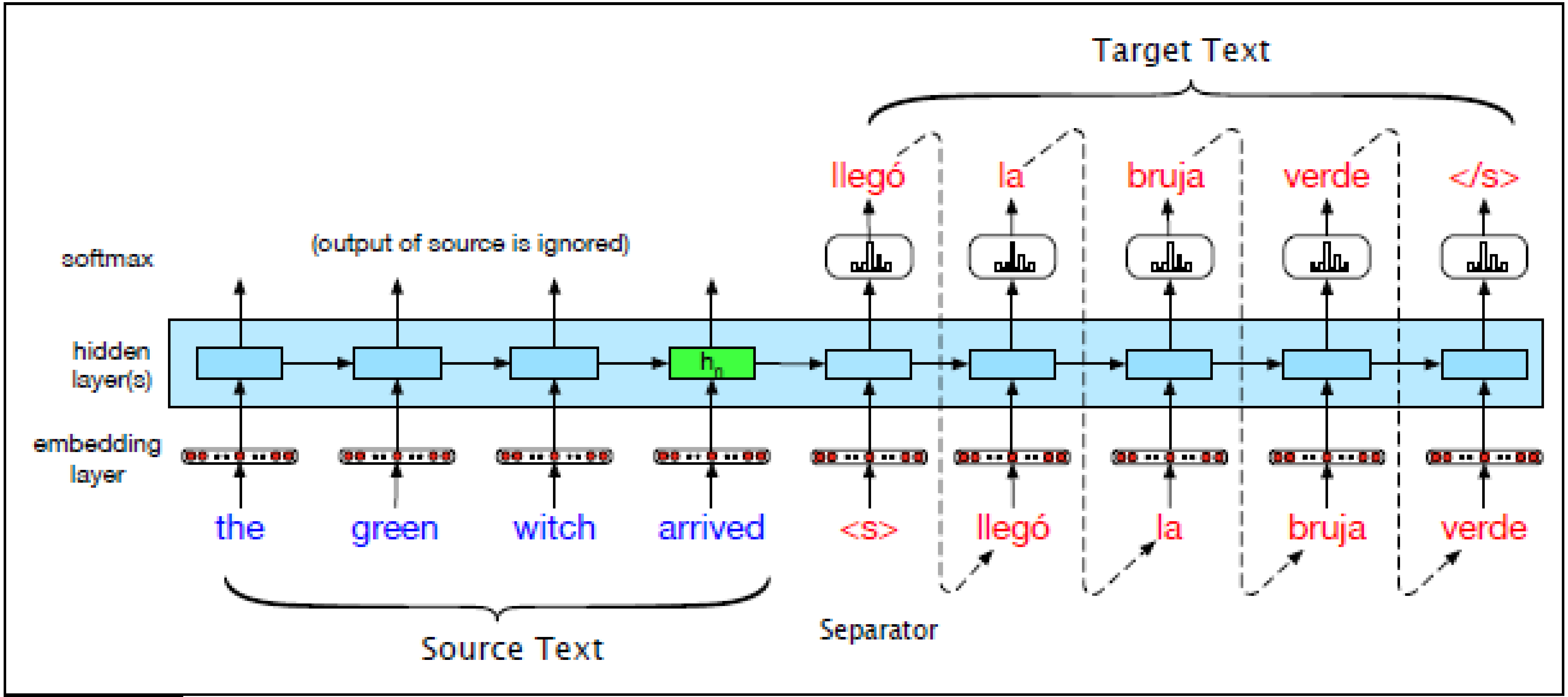


Figure 11.4 Translating a single sentence (inference time) in the basic RNN version of encoder-decoder approach to machine translation. Source and target sentences are concatenated with a separator token in between, and the decoder uses context information from the encoder's last hidden state.

Machine translation

- We train an auto-regressive network on a pair of sentence:
- Source <s> Target
- To apply the translation system, we feed it the source sentence and generate the target sentence

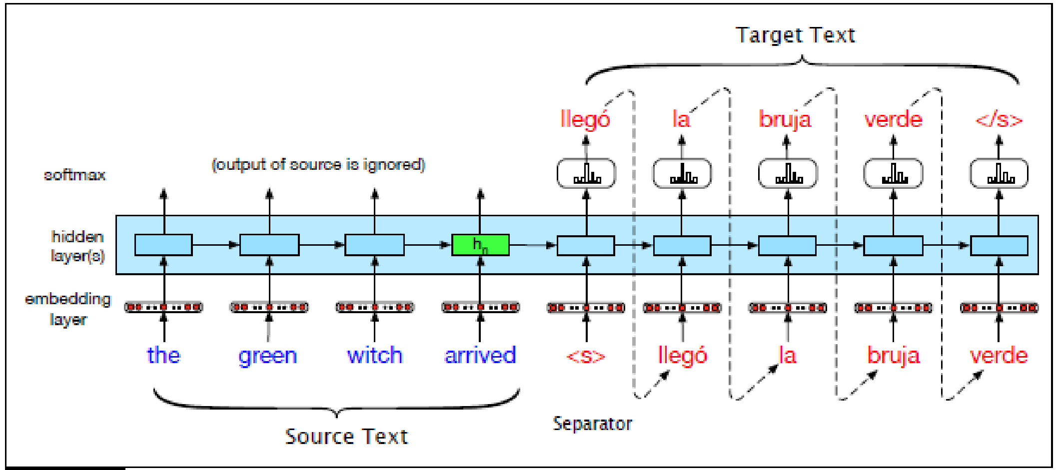


Figure 11.4 Translating a single sentence (inference time) in the basic RNN version of encoder-decoder approach to machine translation. Source and target sentences are concatenated with a separator token in between, and the decoder uses context information from the encoder's last hidden state.

Where can I learn more?

- IN4310 – Deep Learning for Image Analysis (spring)

- IN5550 – Neural Methods in Natural Language Processing (spring)

Frontier Technologies/MS Research Topics

- Reasoning Models
- Multimodality
- Foundation Models
- Autonomous AI Agents