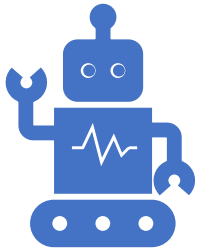
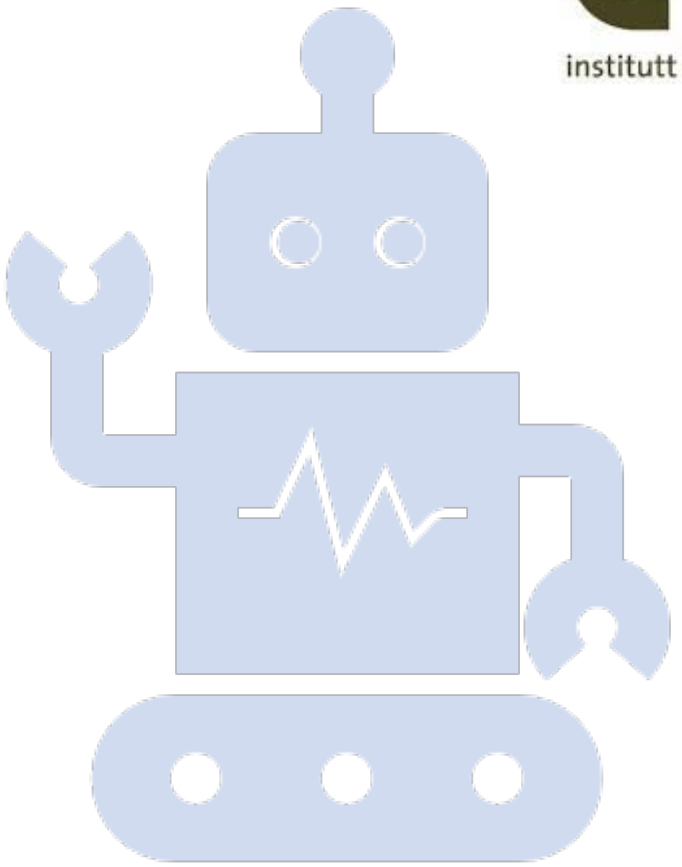


UiO : University of Oslo

institutt for informatikk



IN3050/IN4050 - Introduction to Artificial Intelligence and Machine Learning



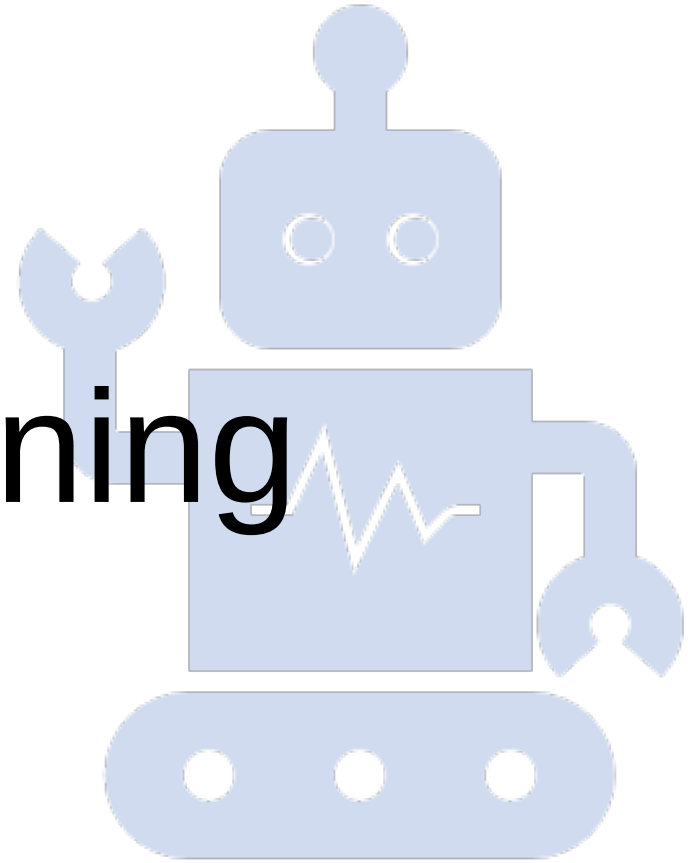
Lecture 12

Glimpses beyond: Deep Neural Networks

Slides by Jan Tore Lønning with revisions by Andrey Kutuzov

Today

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

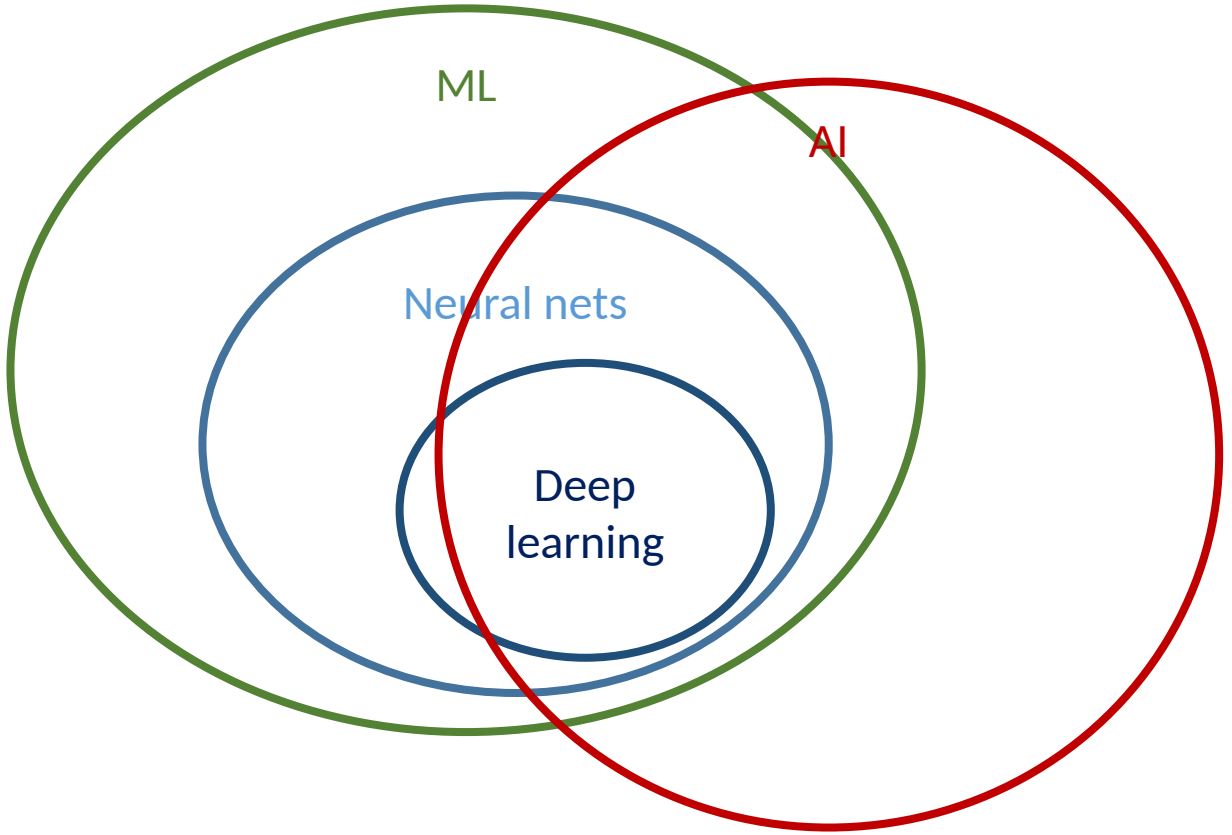


12.1 The deep-learning revolution

IN3050/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

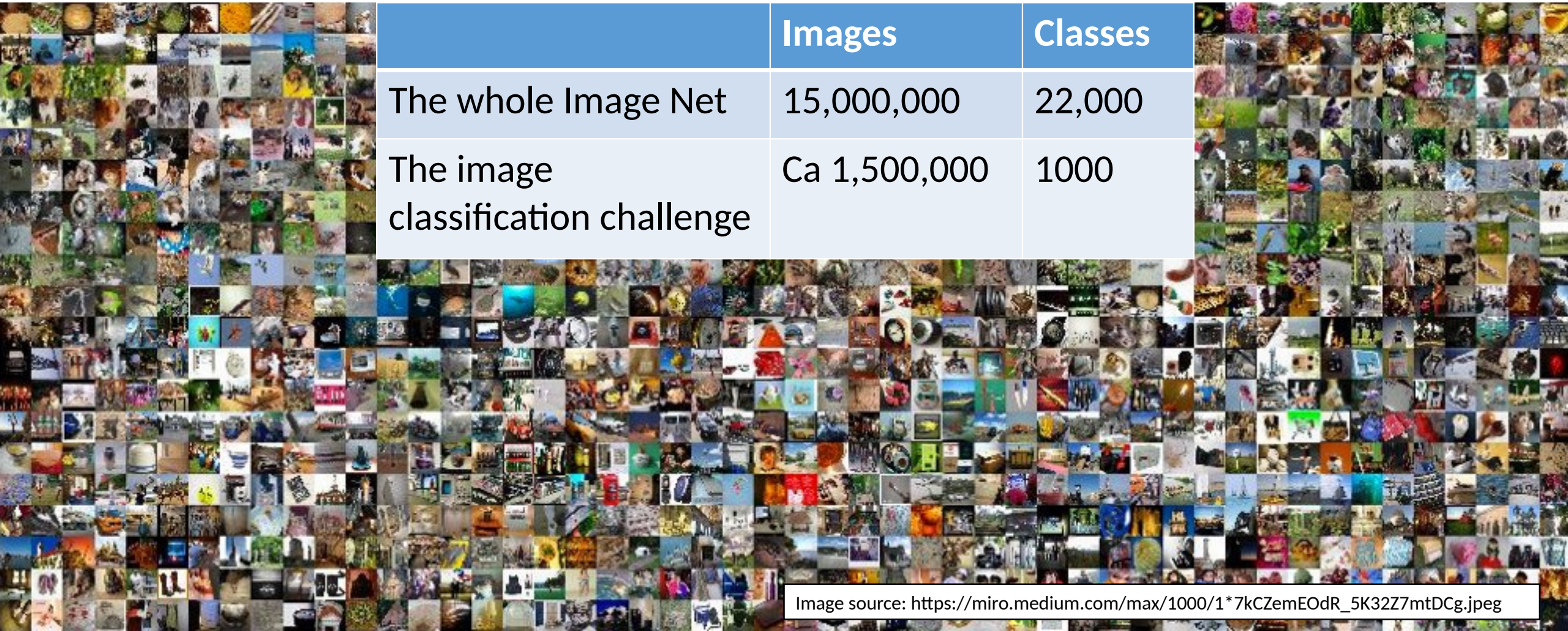


The revolution

- Deep learning breakthrough 12 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

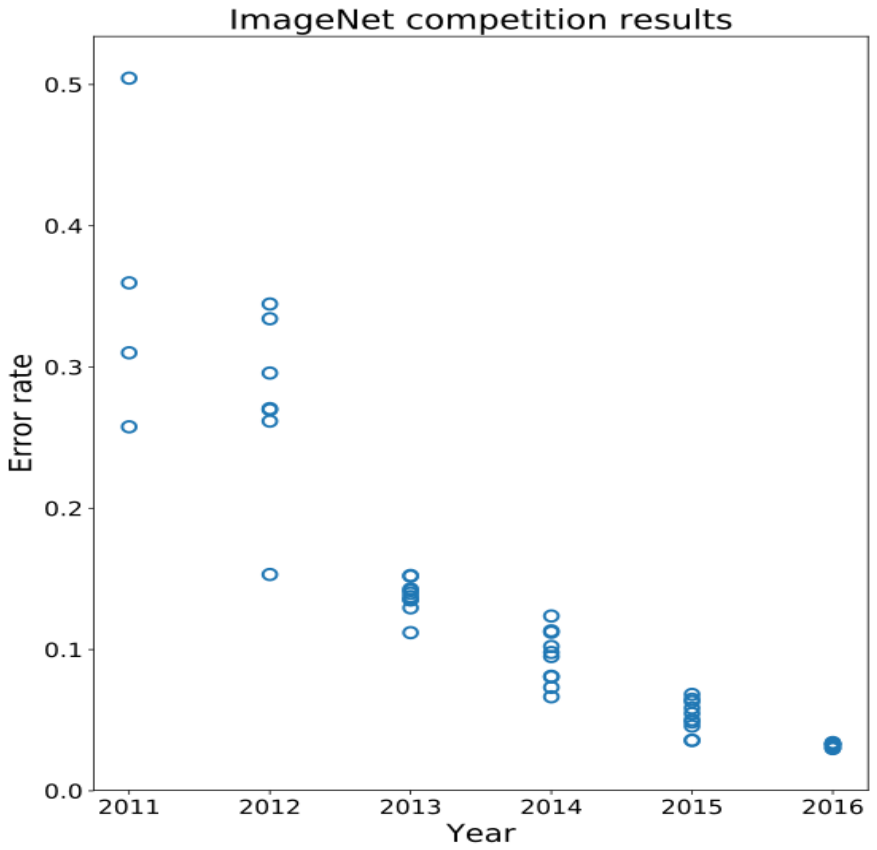


	Images	Classes
The whole Image Net	15,000,000	22,000
The image classification challenge	Ca 1,500,000	1000

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
- Started an immediate renewed interest in neural nets
- Started an 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

Image recognition today

- Google: https://cloud.google.com/vision
- MicroSoft: https://azure.microsoft.com/en-us/products/ai-services/ai-vision

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



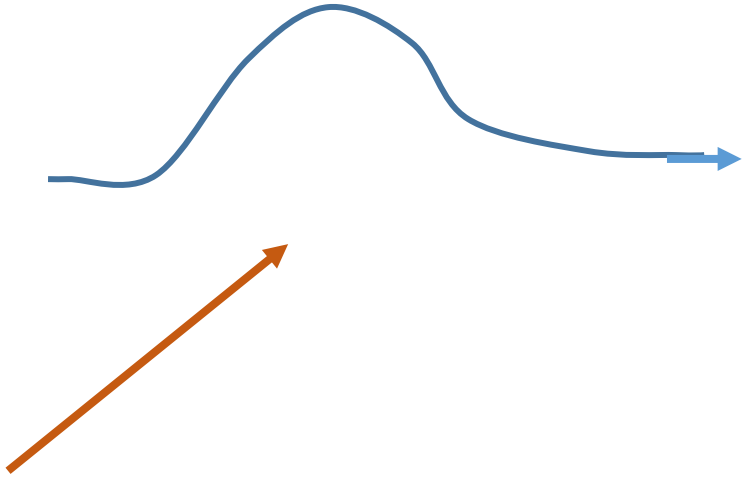
Other applications

- 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)

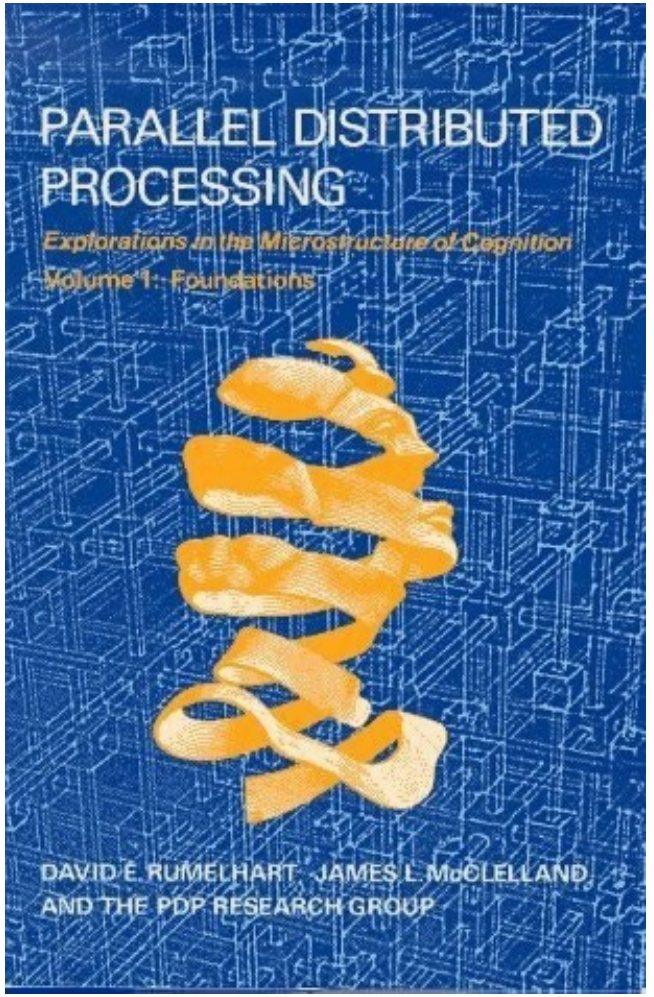
- 1958, Rosenblatt invented the perceptron
- 1969, Minsky & Papert, *The perceptron:*
  - Networks without hidden layers can only learn linear classifiers
  - Networks with hidden layers are probably impossible to train
- Less interest in perceptrons afterwards



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 other approaches to ML was preferred in usage and developed, e.g., logistic regression, SVM
- But some brave and stubborn researchers continued the work on neural nets, including Hinton, and got their rewards in 2010s
- https://awards.acm.org/about/2018-turing

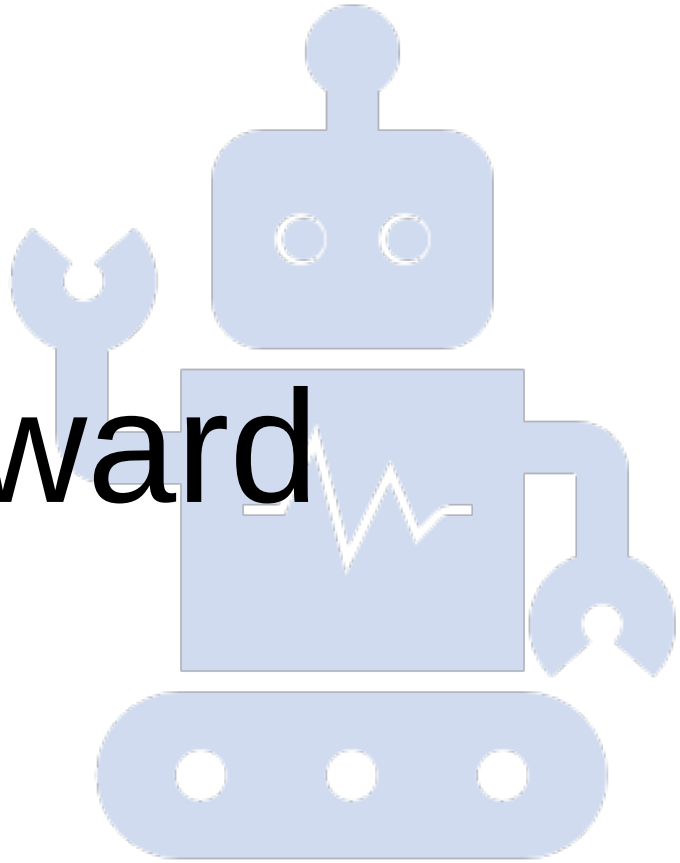


Why did NNs finally succeed?

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





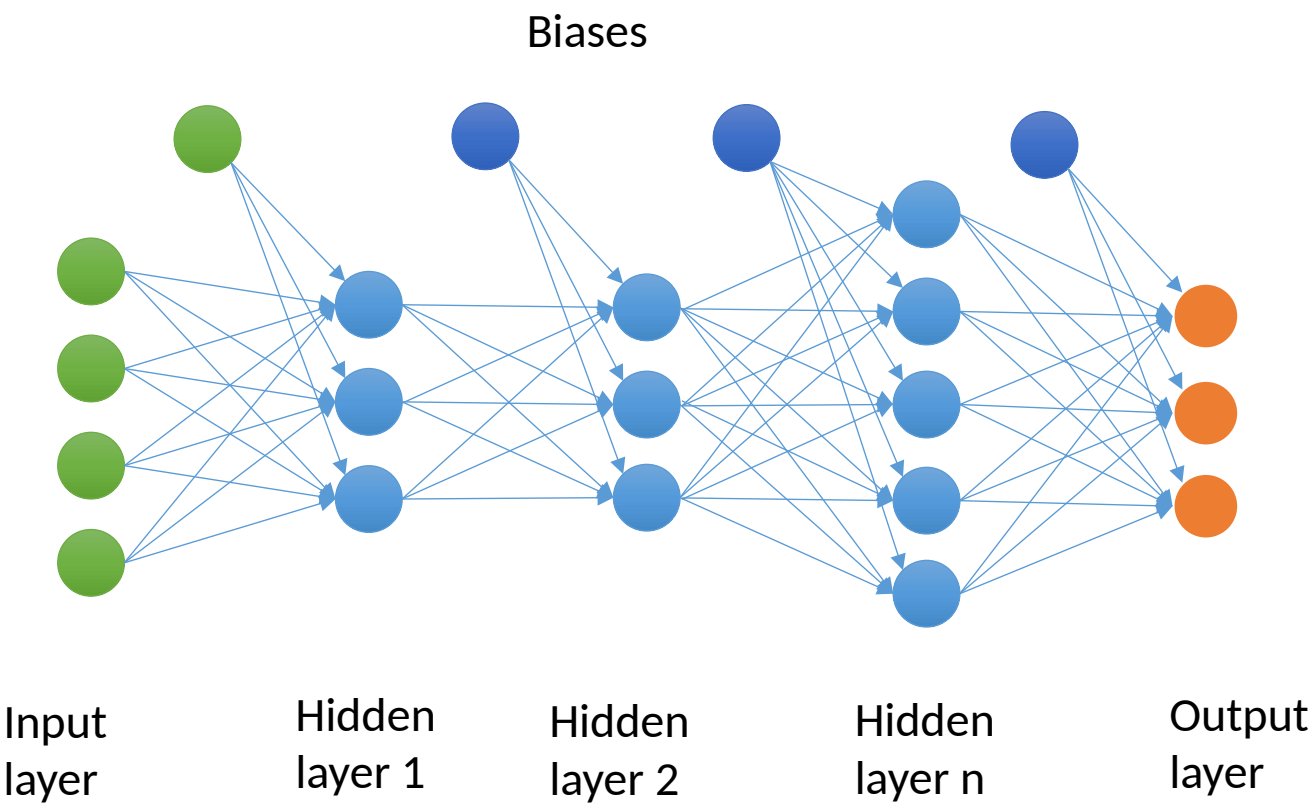


12.2 Deep feed-forward Neural networks

IN3050/IN4050 Introduction to Artificial Intelligence and Machine Learning

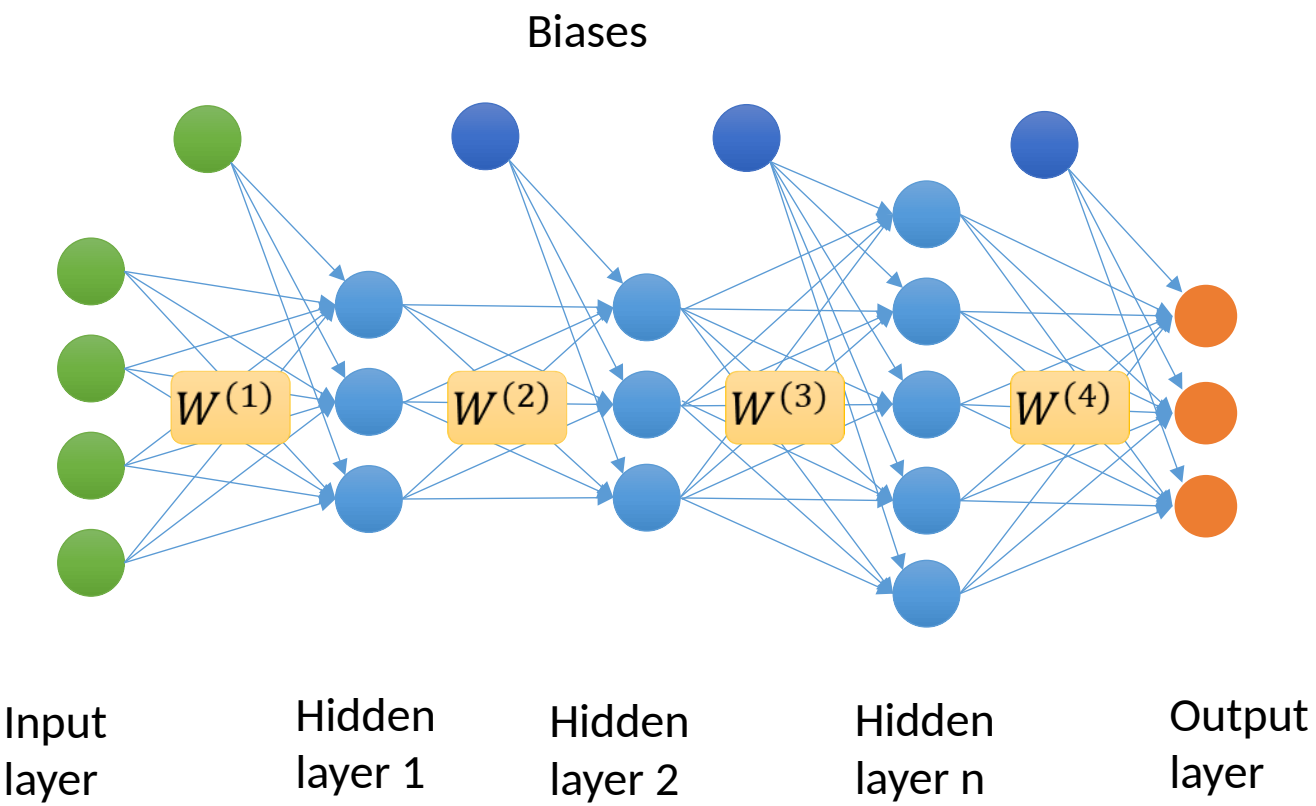
Deep feed-forward NN

- 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



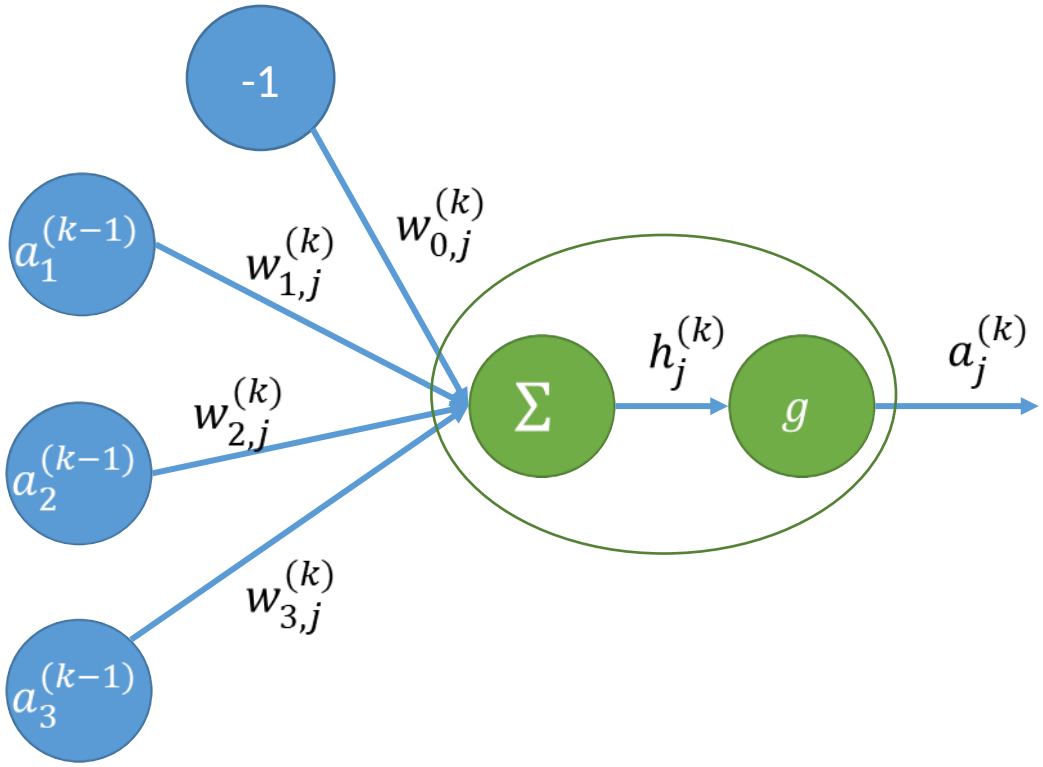
Weight matrices

- One matrix of weights for each layer



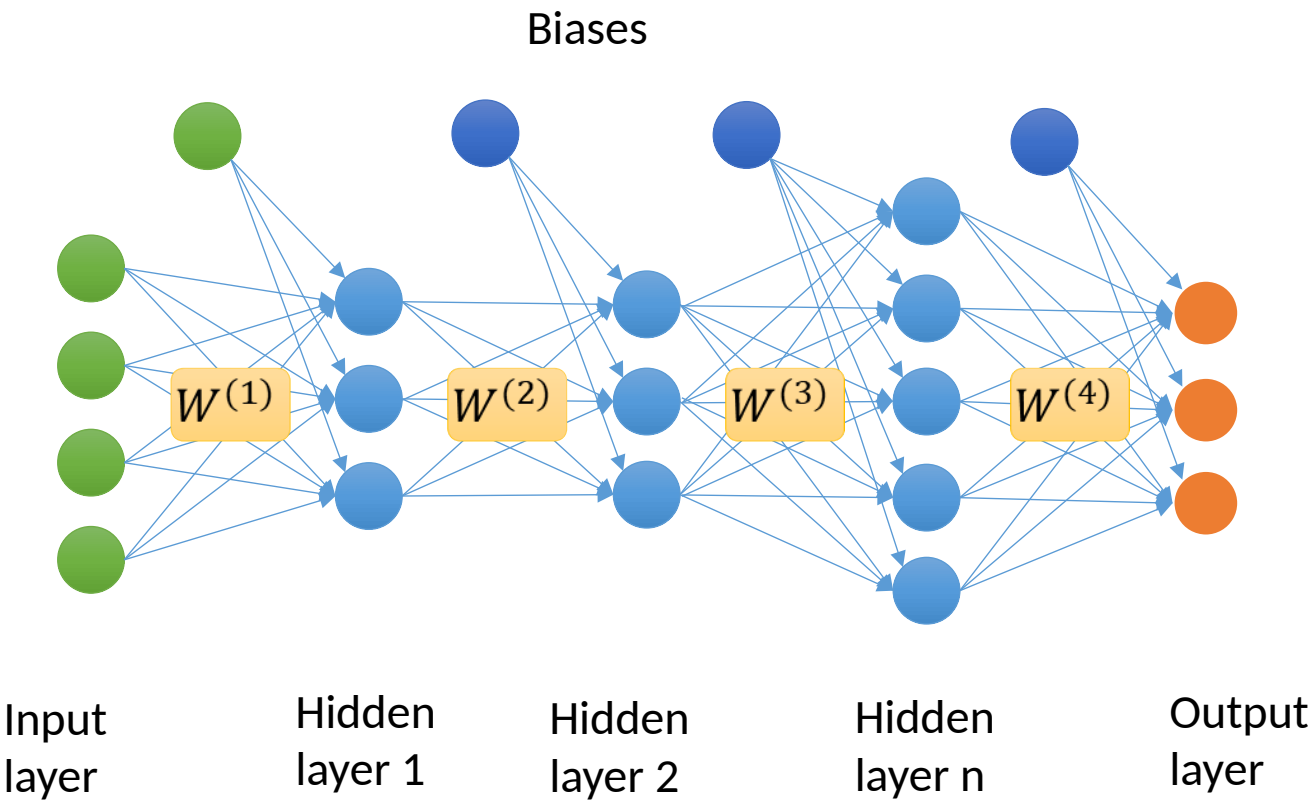
The hidden nodes - forward

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



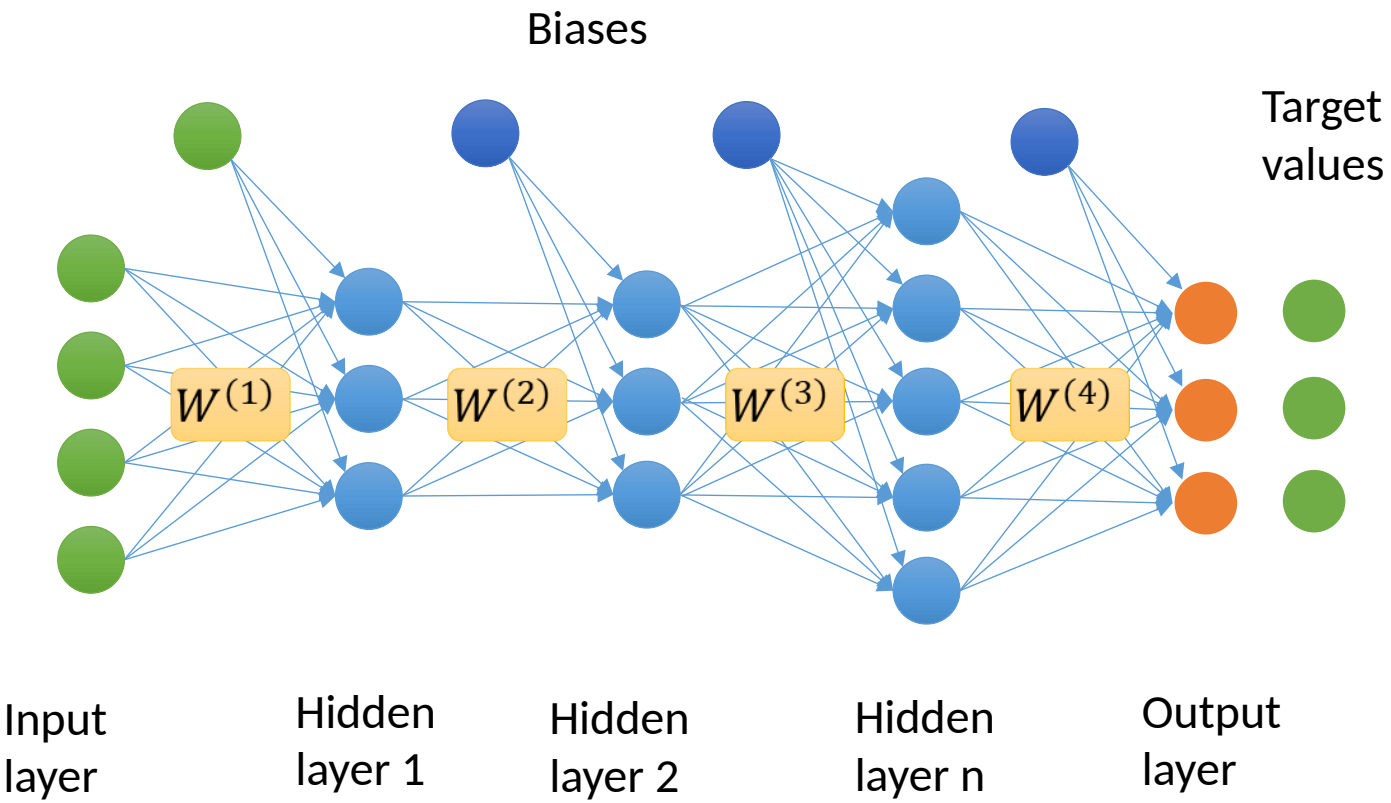
Forward

- 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



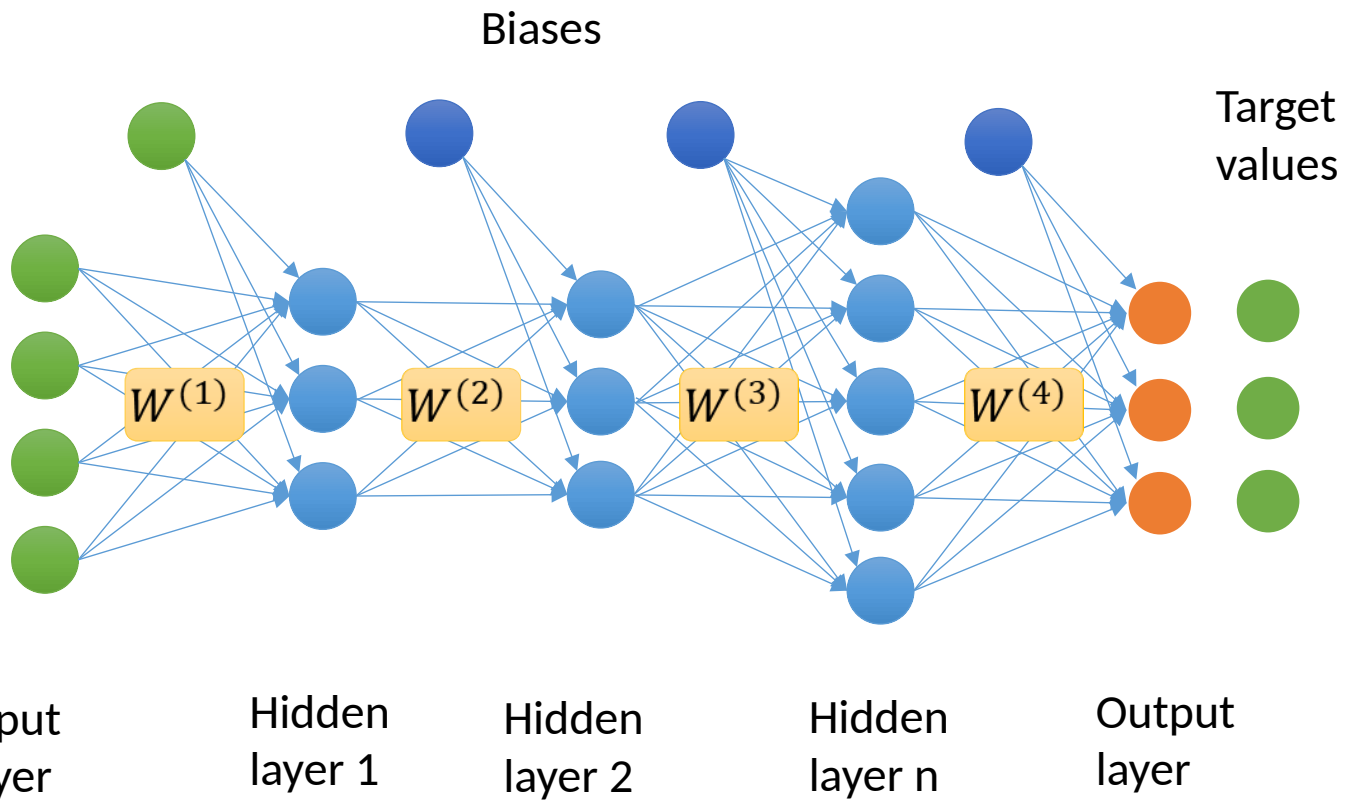
Update

- 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



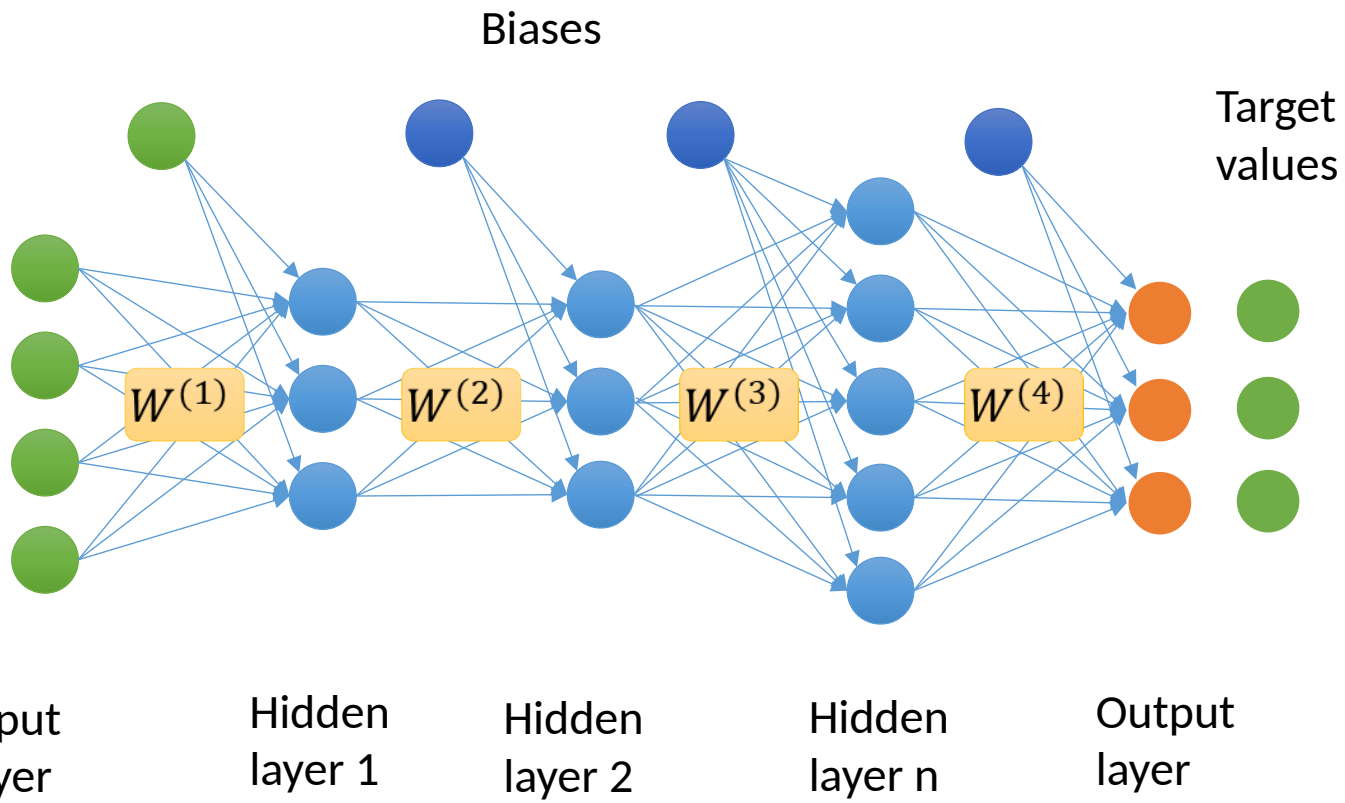
Update

- 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 $\frac{\partial}{\partial w_{i,j}^{(k)}} L(\mathrm{t}, y)$



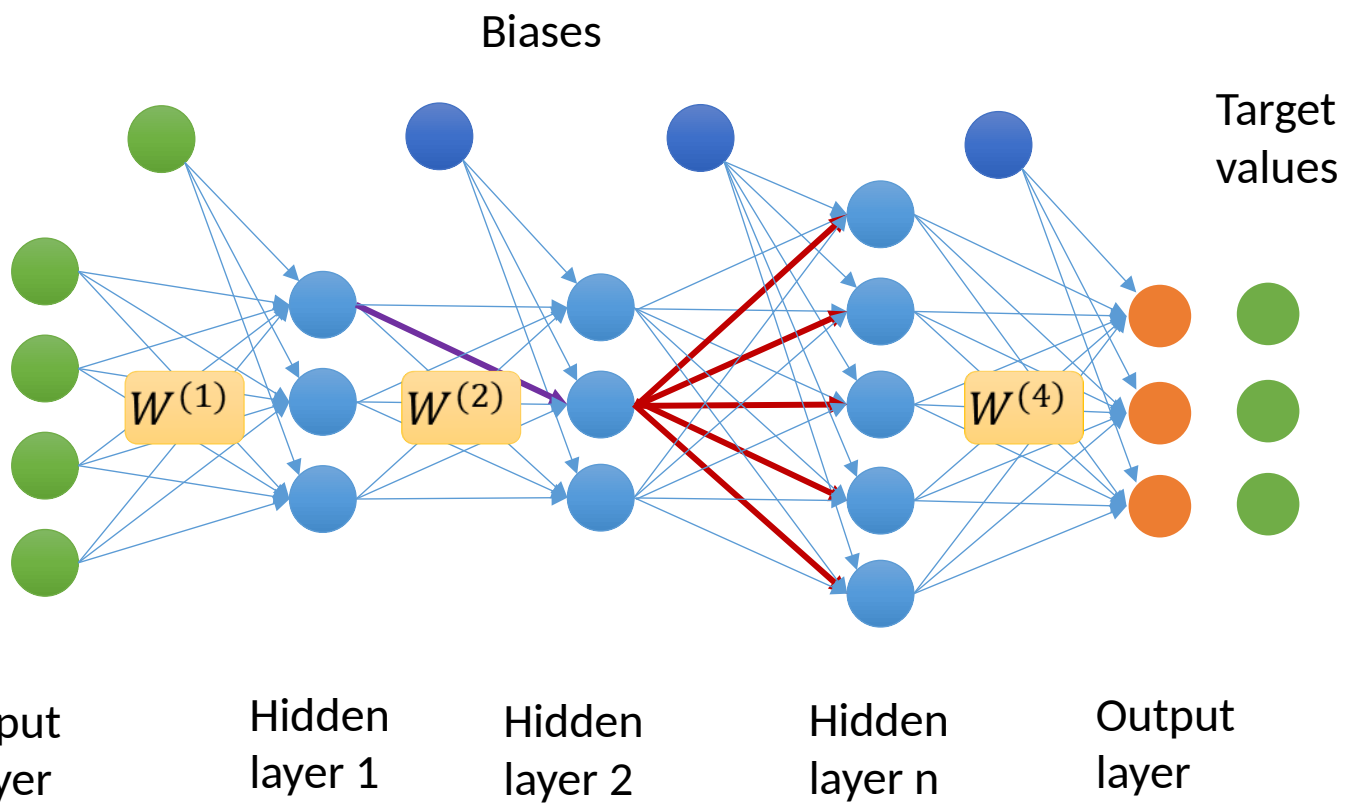
Update

- At the last layer with activation function f,
  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:) $w_{i,j}^{(4)} = w_{i,j}^{(4)} - \eta \delta_j^{(4)} a_i^{(3)}$



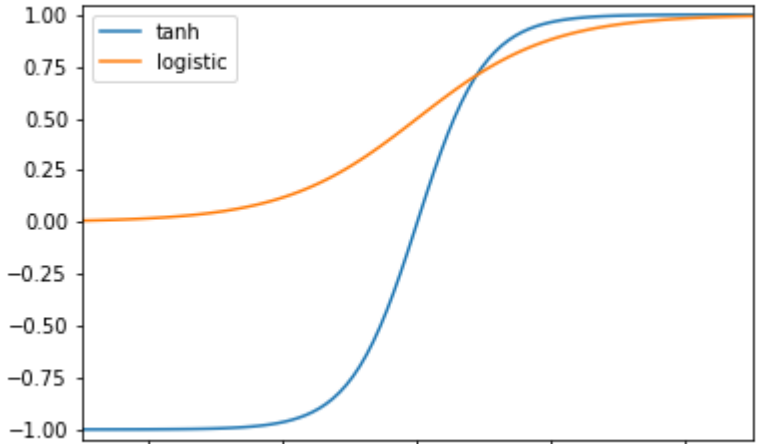
At the hidden layer

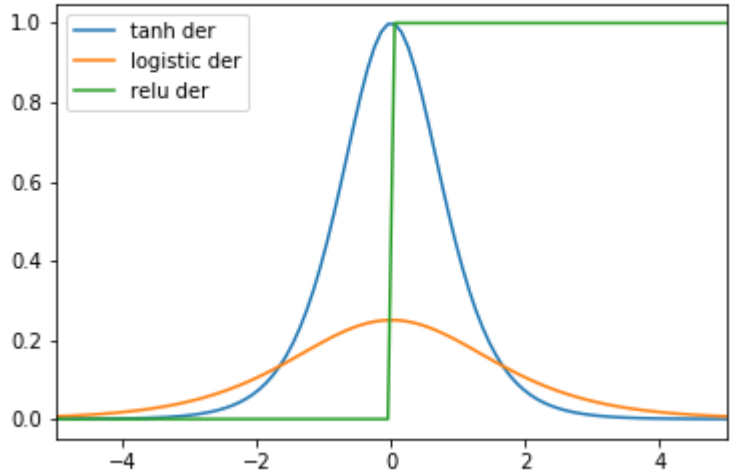
- At the hidden layer with activation function g,
  i.e., $a_j = g(h_j), h_j = \sum_i w_{i,j}^{(k)} a_i^{(k-1)}$
- $\delta_j^{(k)} = \frac{\partial}{\partial h_j} g(h_j) \sum_m \delta_m^{(k+1)} w_{j,m}^{(k+1)}$
- (eventually:)
  $w_{i,j}^{(k)} = w_{i,j}^{(k)} - \eta \delta_j^{(k)} a_i^{(k-1)}$
- Follow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



Vanishing gradient problem

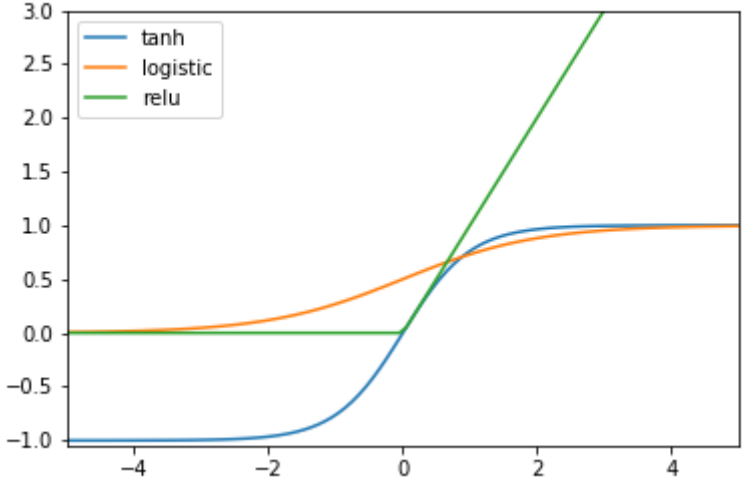
- The derivative of the logistic function (and the tanh) is close to 0 except in a small interval around 0
  - It gets easily saturated
- At each backwards step calculating the deltas, we multiply with the derivative of the activation function
- The gradient comes close to 0. Unbearable slow update,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

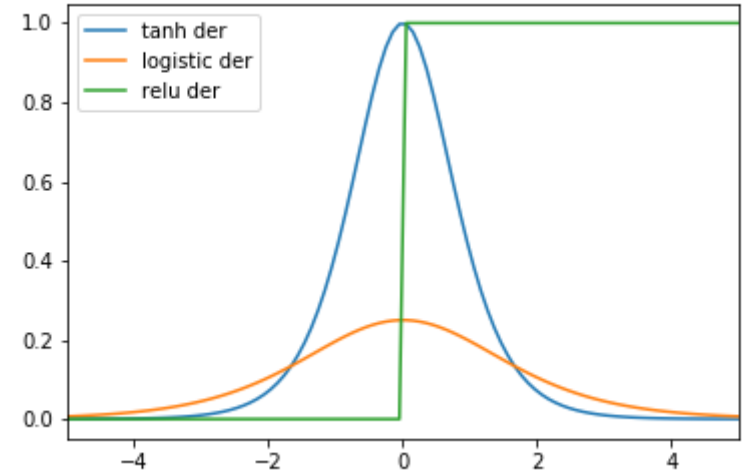




Rectified linear unit, ReLU

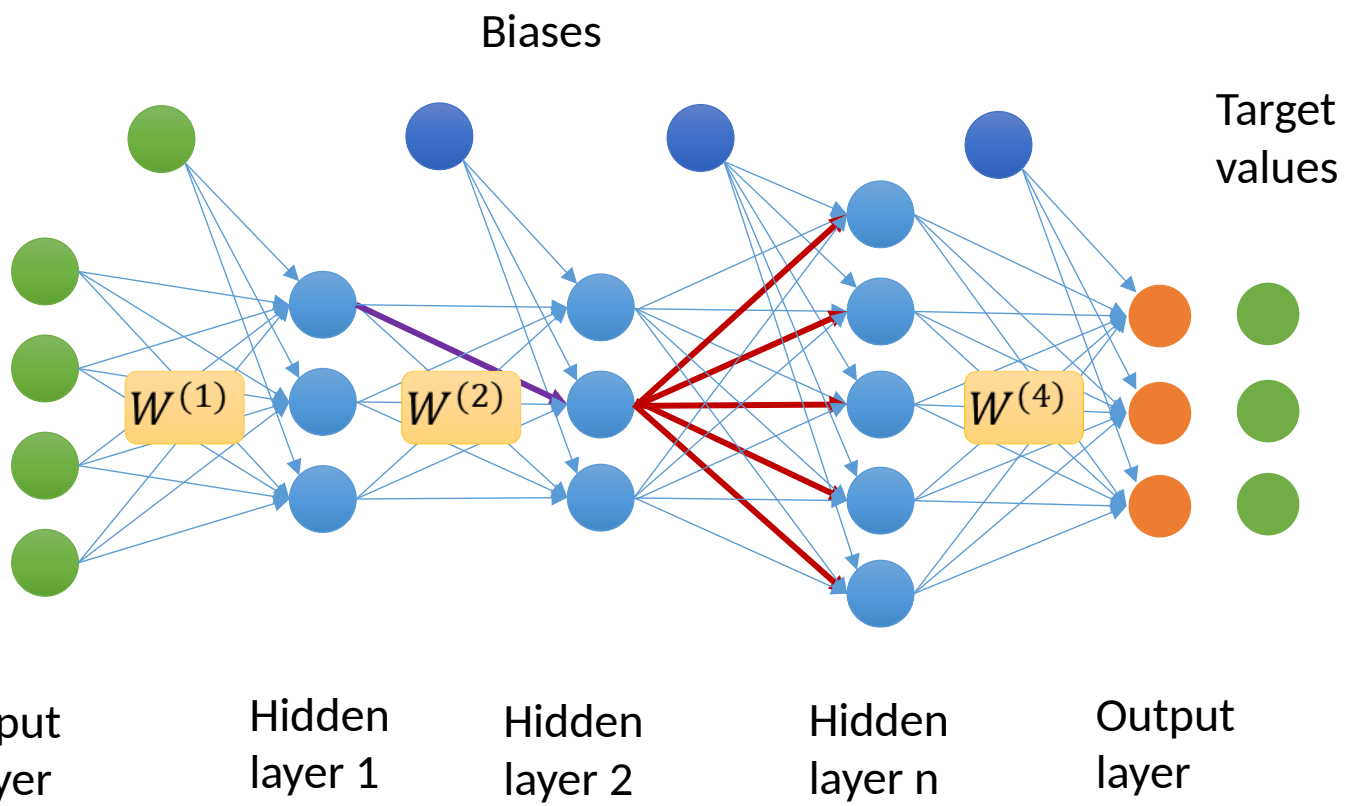
- Alternative activation functions in the hidden layers
- $ReLU(x) = \max(x, 0)$
- $ReLU'(x) = 1$ for $x > 1$
- $ReLU'(x) = 0$ for $x < 1$
- Use 0 for $ReLU'(0)$
- ReLU is the preferred method in deep networks
  - (There are various modified versions of ReLU)





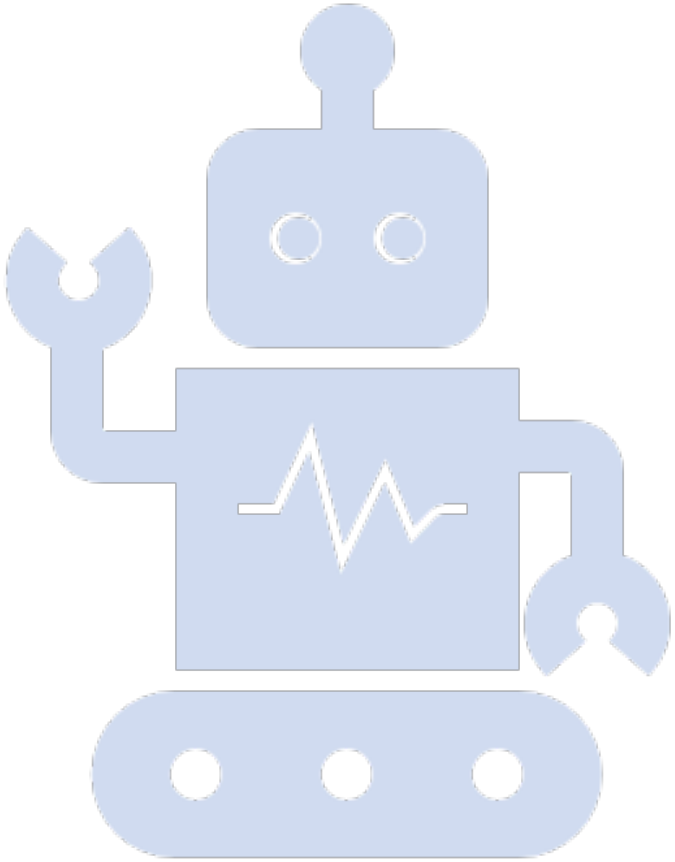
At the hidden layer

- At the hidden layer with activation function $g = \mathrm{ReLU}$,
  i.e., $a_j = g(h_j), h_j = \sum_i w_{i,j}^{(k)} a_i^{(k-1)}$
- $\delta_j^{(k)} = \frac{\partial}{\partial h_j} g(h_j) \sum_m \delta_m^{(k+1)} w_{j,m}^{(k+1)}$
- $\delta_j^{(k)} = \sum_m \delta_m^{(k+1)} w_{j,m}^{(k+1)}$ for $x > 0$
- $\delta_j^{(k)} = 0$ for $x \leq 0$



12.3 Convolutional NNs and image processing

IN3050/IN4050 Introduction to Artificial Intelligence and Machine Learning



The MNIST data set (lecture 5 example)

Domain

- Hand-written digits



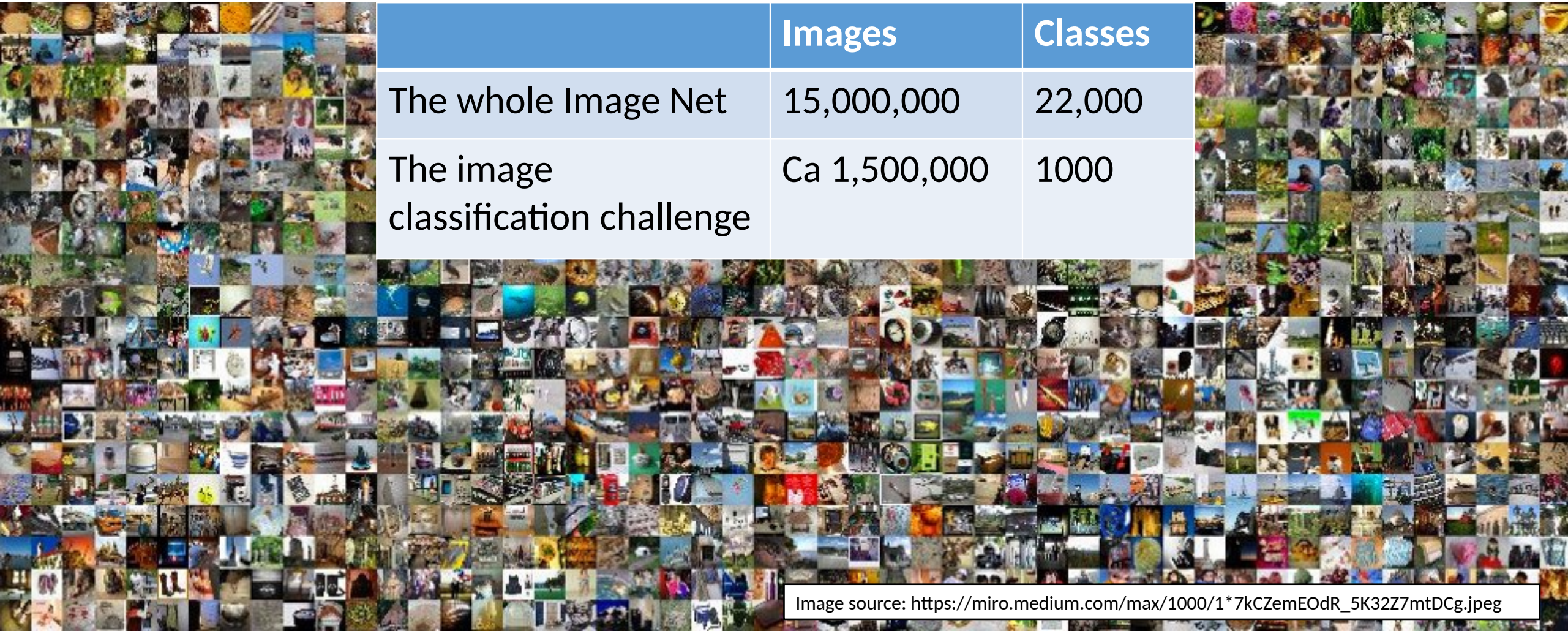
Labels

0
1
2
3
4
5
6
7
8
9

- To each hand-written picture of a digit, predict the correct digit
- There are 10 different classes
- 60,000 training images
- 10,000 test images
- Each picture 28x28

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

Image Net

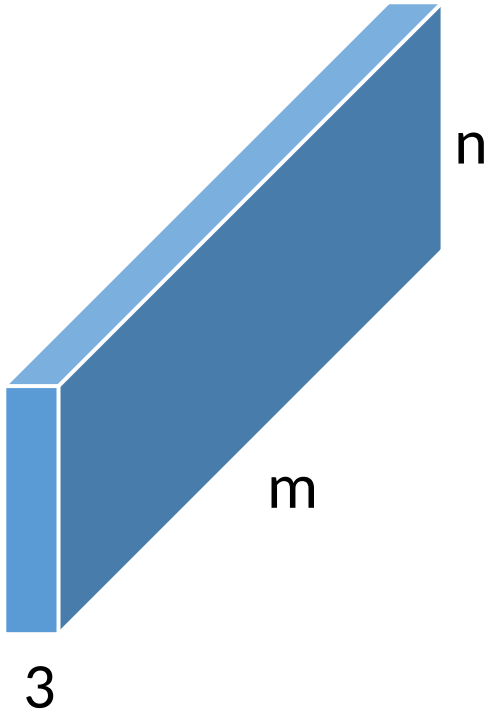


	Images	Classes
The whole Image Net	15,000,000	22,000
The image classification challenge	Ca 1,500,000	1000

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 picture, 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
- But:
  - The solution of this problem illustrates one step to the solution of the more complex problem

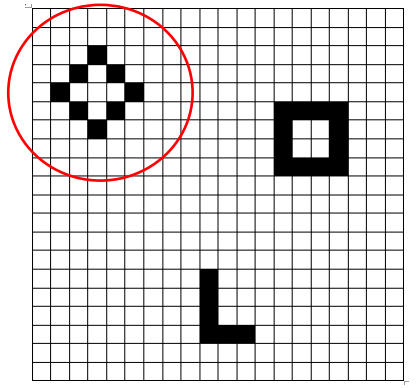


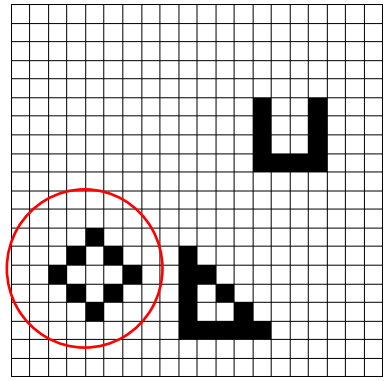
A very simplified example

The problem

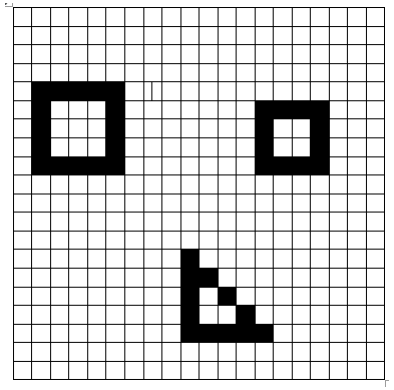
- Positive class if it contains at least one subfigure of exactly this shape and size
- How can a classifier which takes pixels as input recognize this?
- There is no similarity in the pixels.

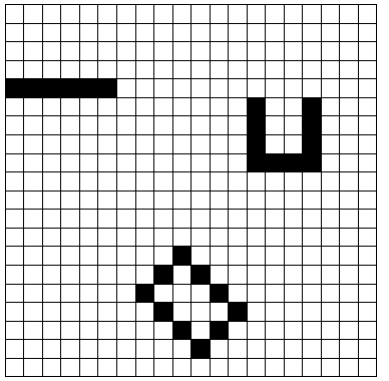
Positive examples





Negative examples

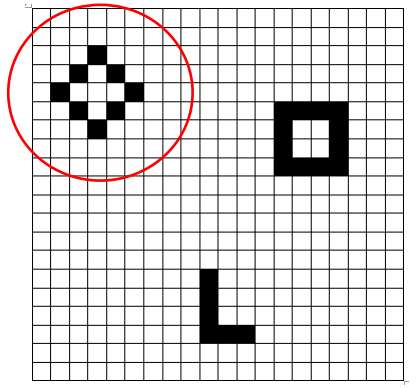


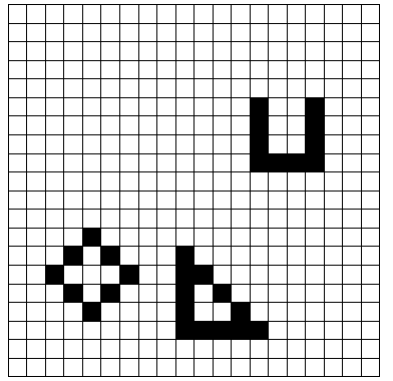


Approach for solution

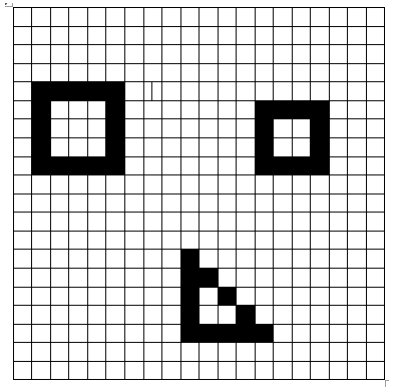
- Positive class if it contains at least one subfigure of exactly this shape and size
- Split the task in two:
  - For each 5x5 subpicture, decide whether it has this shape or not
  - Answer whether the picture has at least one such subpicture

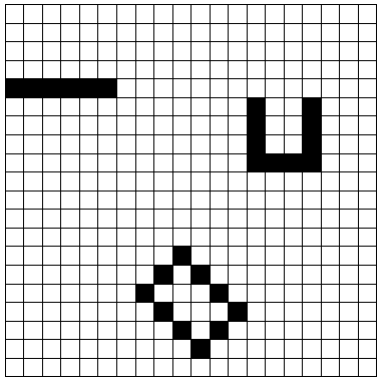
Positive examples





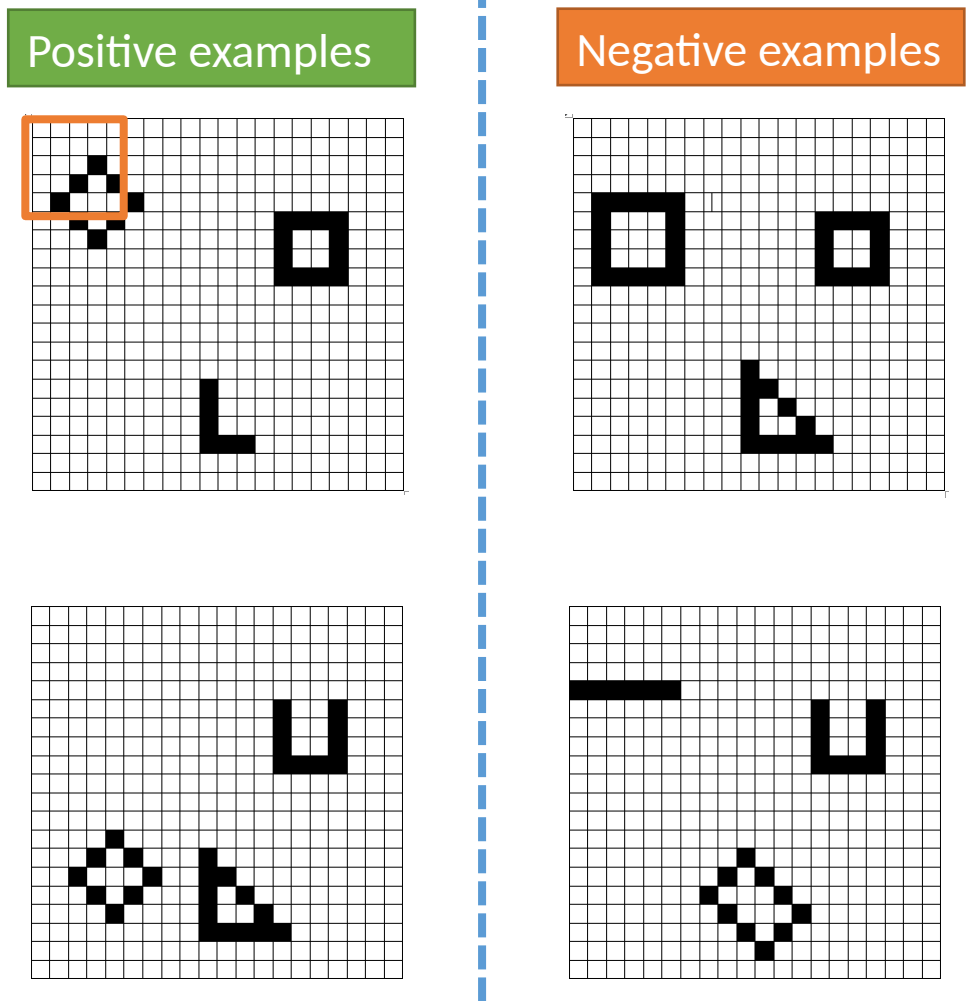
Negative examples

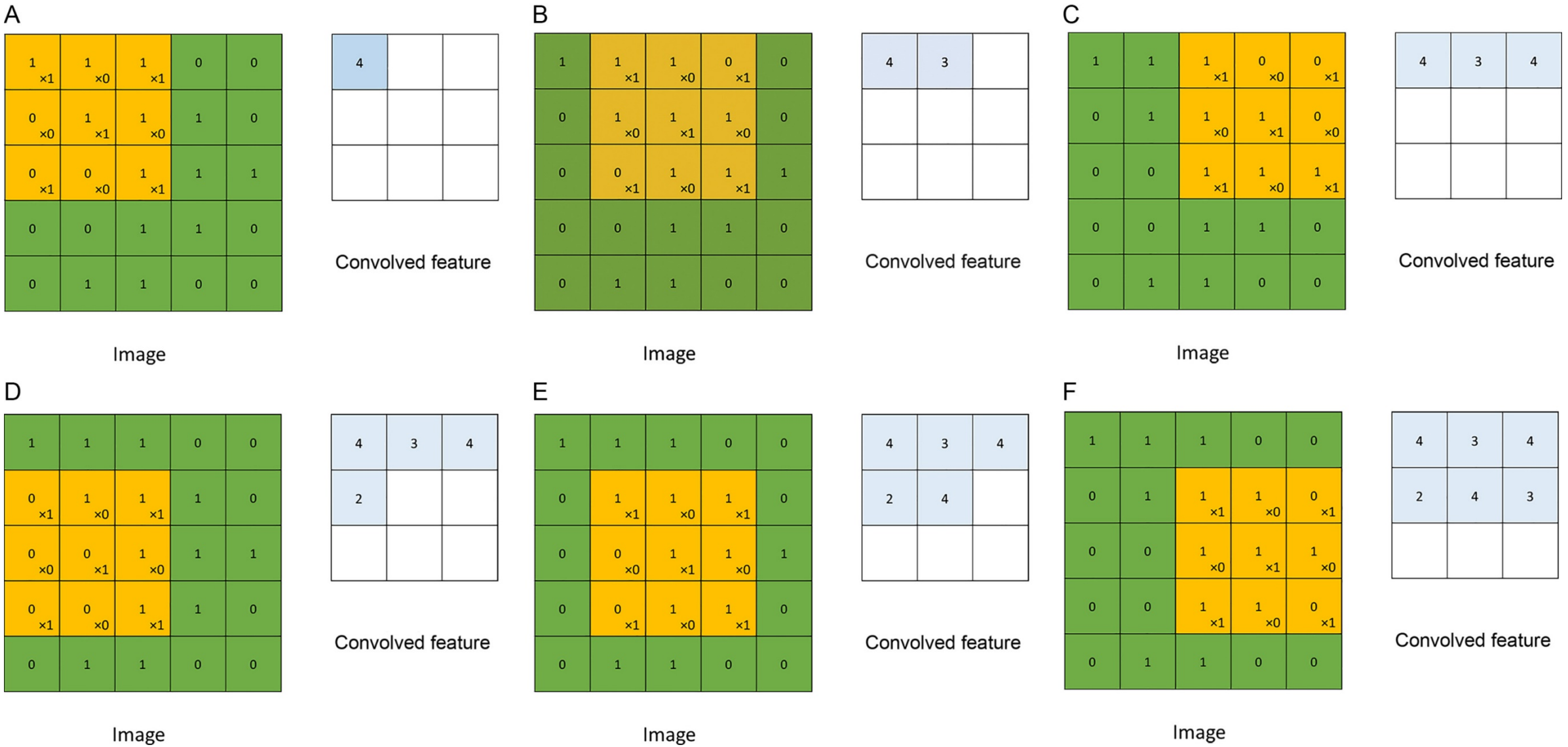




The filter

- 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 subfigure

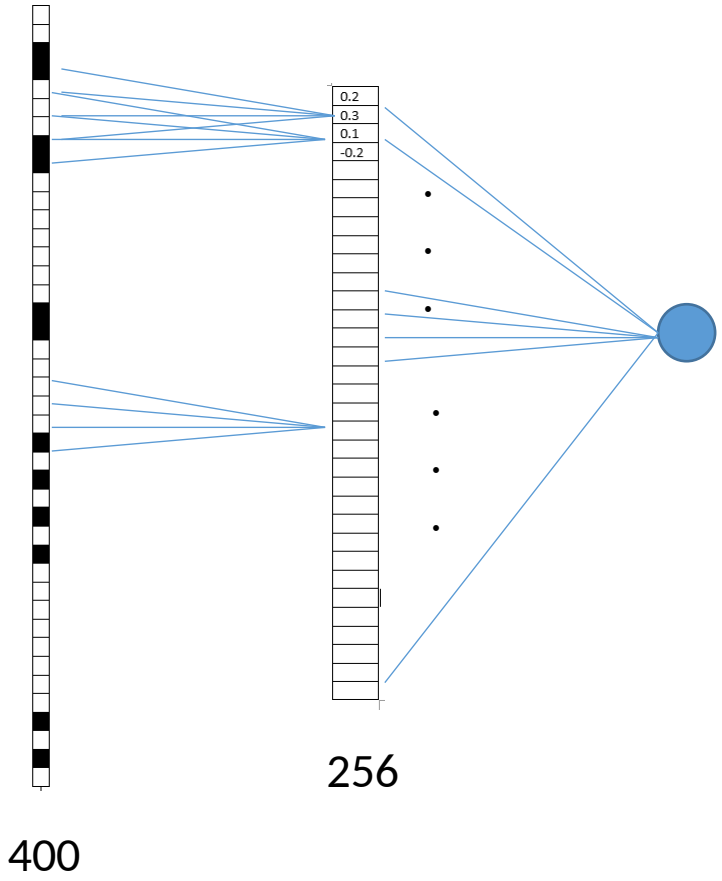




https://www.sciencedirect.com/topics/computer-science/convolutional-layer

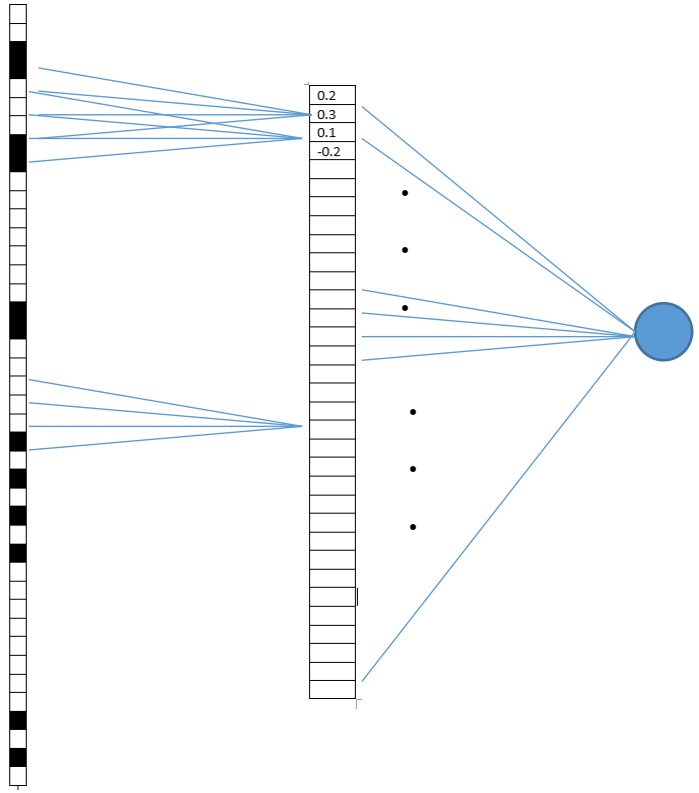
The network

- 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

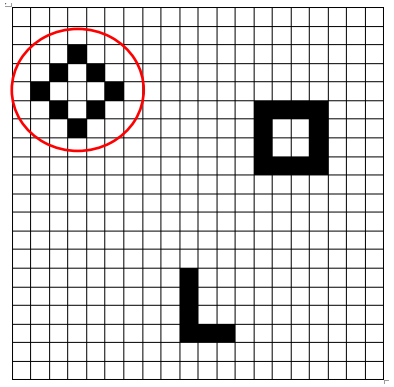
- First: There are only
  - 25x256 connections in the first layer, and not 400x256
  - 256 connections in the second layer
- The clue: Each hidden node should learn the same:
  - $w_{i,j} = w_{i+k,j+k}$
  - We use the same (26x1) weight matrix for all hidden nodes, which we update through backprop.
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

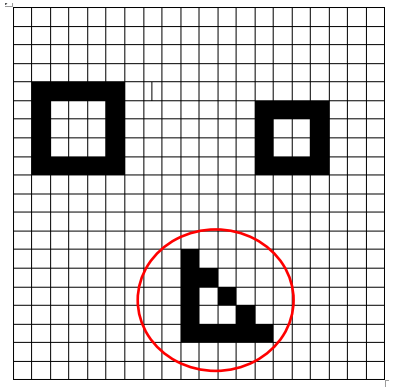


A more complex task

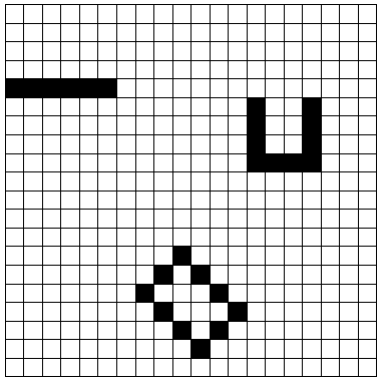
- Positive class if it contains any of the two 5x5 patterns
- What now?
- We can have several filters
  - Each of them can learn one specific pattern
  - We can put a numeric calculation on top of them in the final fully connected layer
    - ‘’More than 3 of pattern 1 and 2 or less of pattern 2, none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

Positive examples





Negative examples



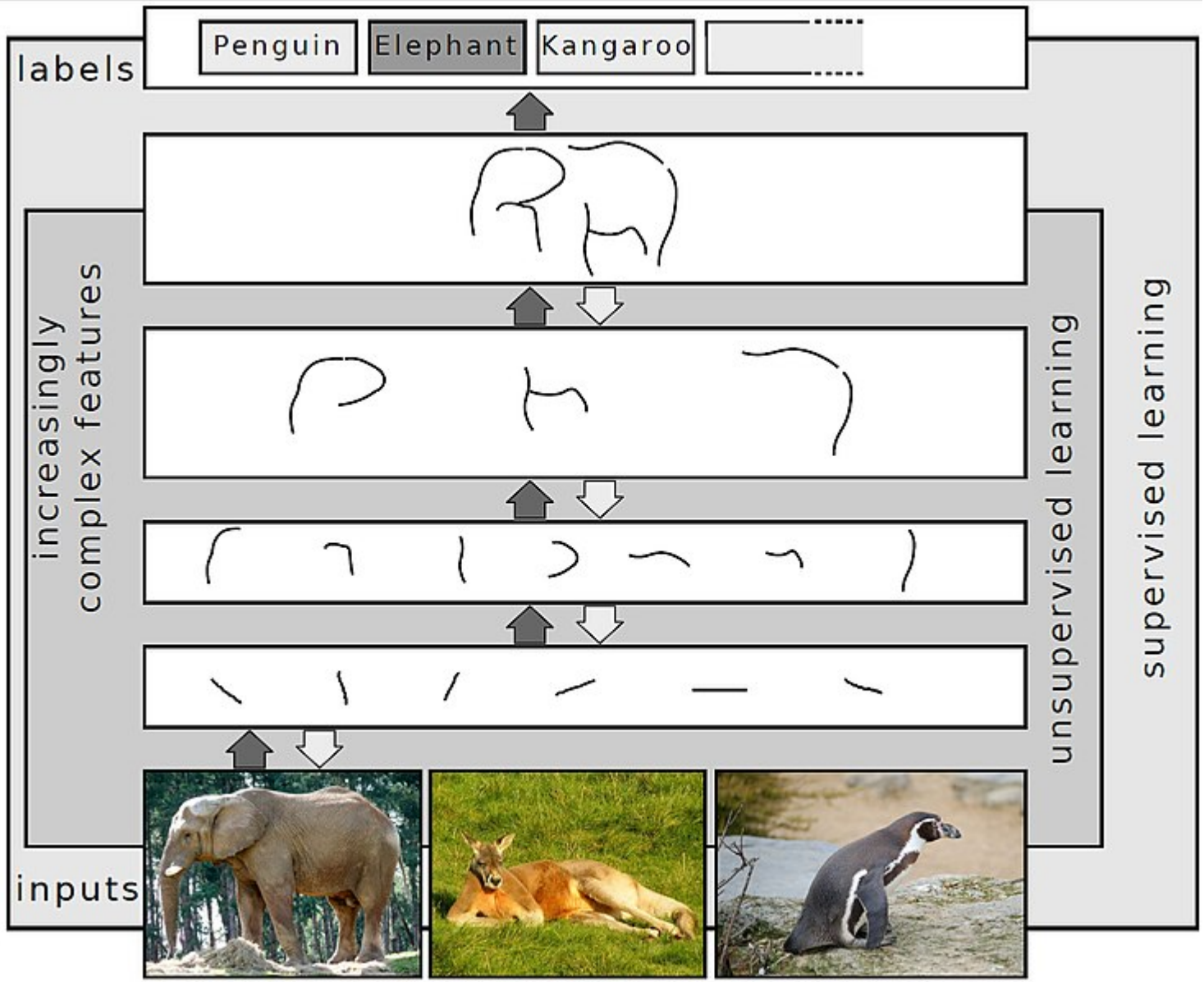
Towards convolutional networks

So far

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

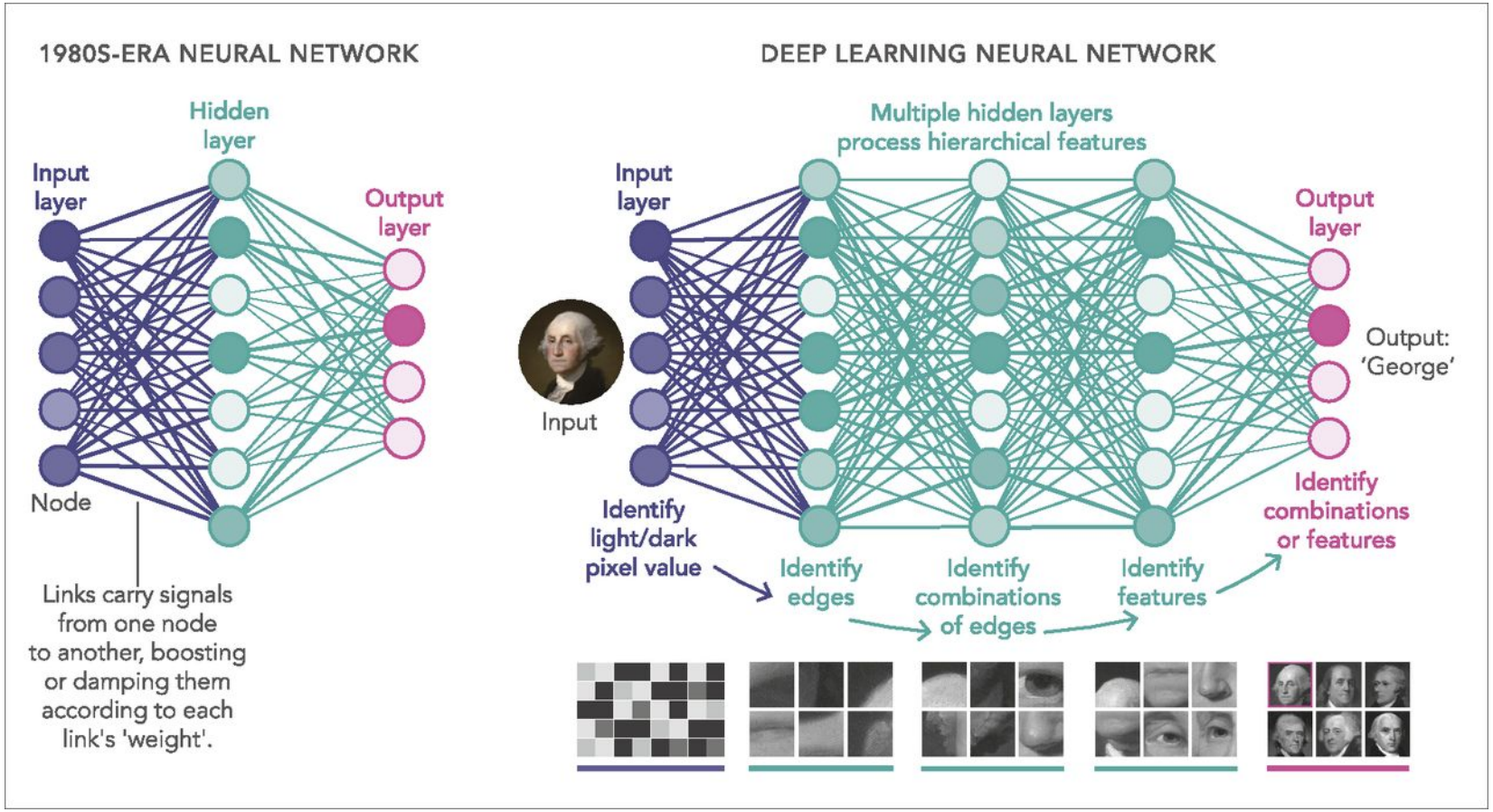
The convolutional network

- Use filters as indicated
- Several layers
- Recognize
  1. Simple patterns, e.g., 5x5 pixels
  2. Lines, curves
  3. Contours
  4. Figures
  5. Etc.
  - (The results at each layer do not have such a crisp interpretation).



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

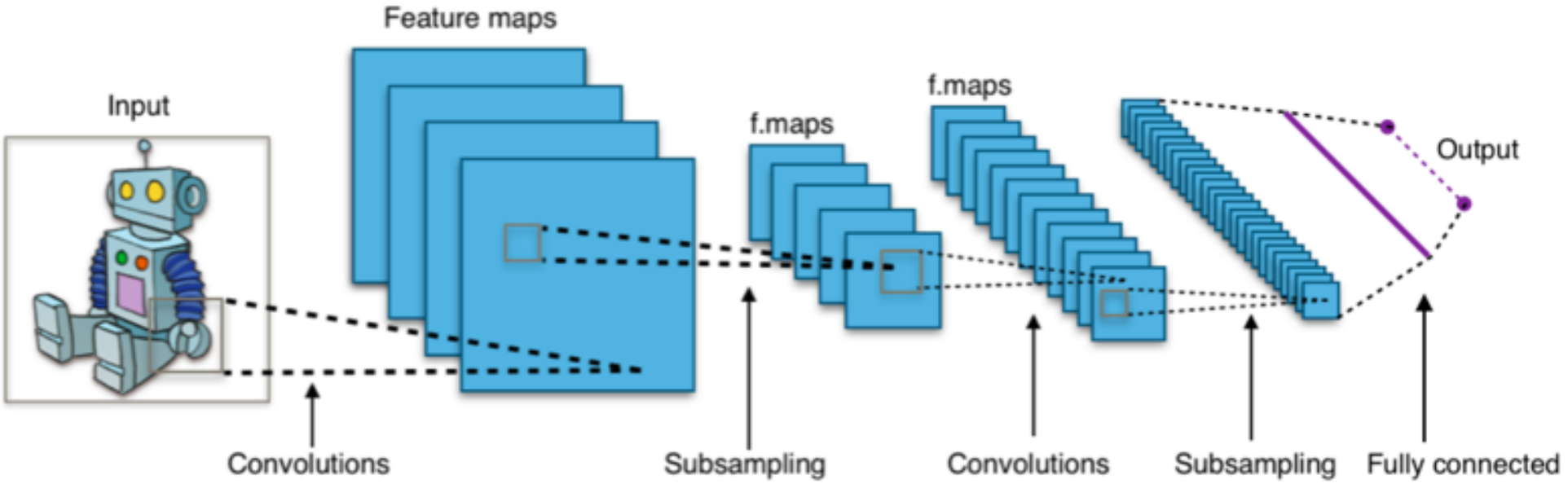
“Neural network” models of AI process signals by sending them through a network of nodes analogous to neurons.



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

©2019 by National Academy of Sciences

Typical architecture (simplified)

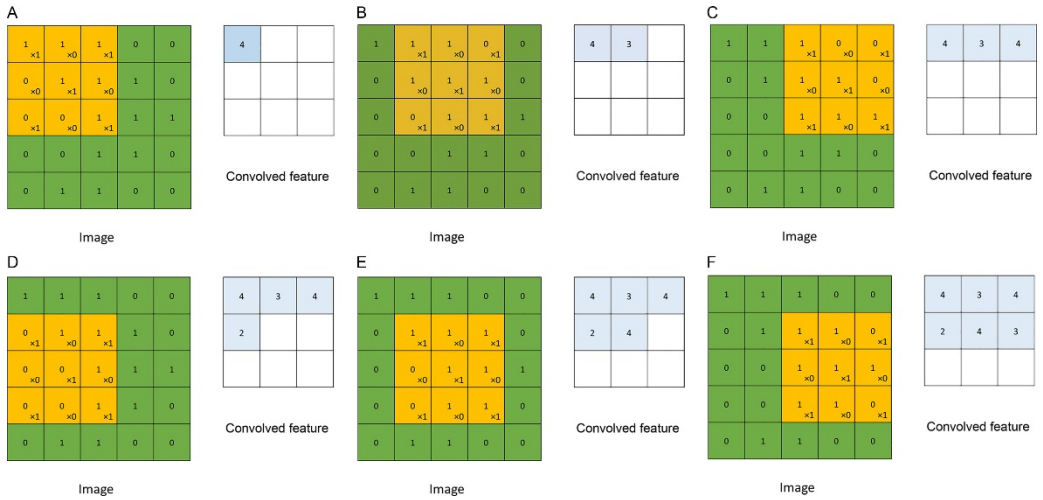


https://en.wikipedia.org/wiki/Convolutional_neural_network

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

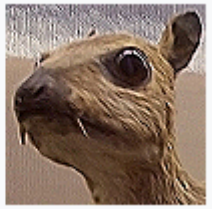
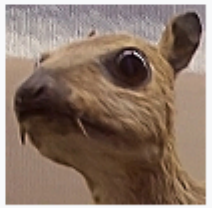
Filters

- We have already considered them.
- They work as in the simplified example.
- 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



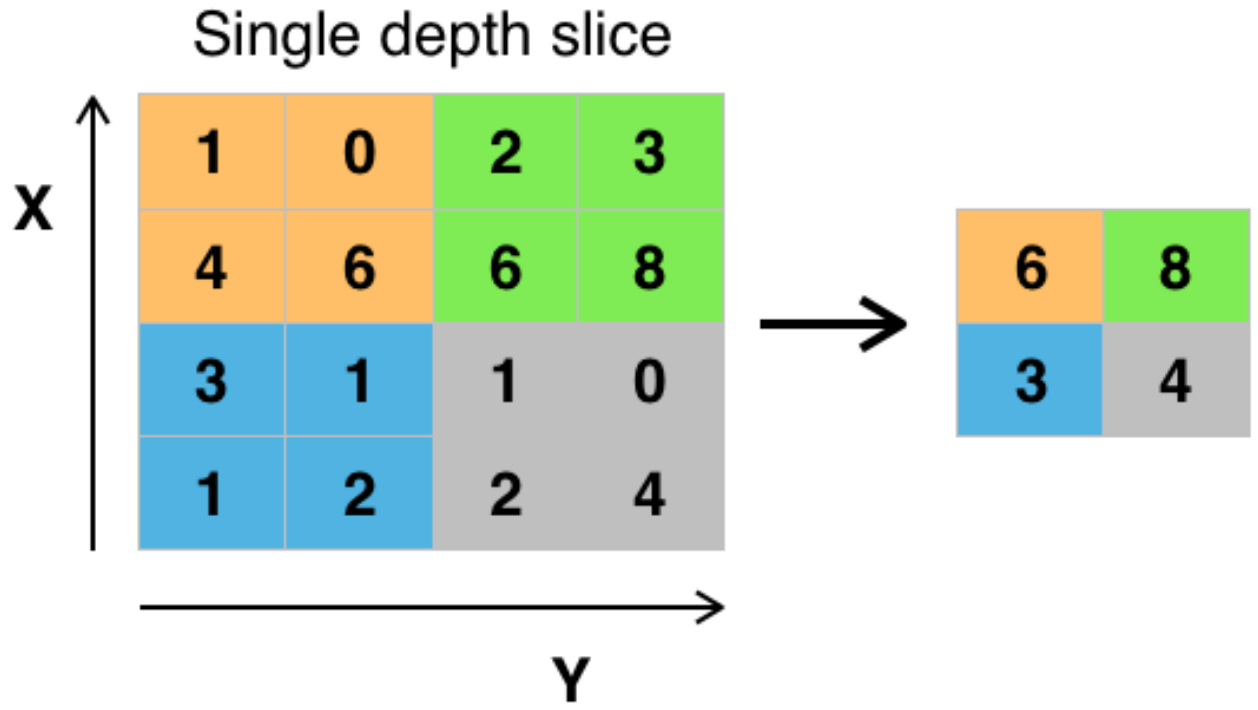
https://www.sciencedirect.com/topics/computer-science/convolutional-layer

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)

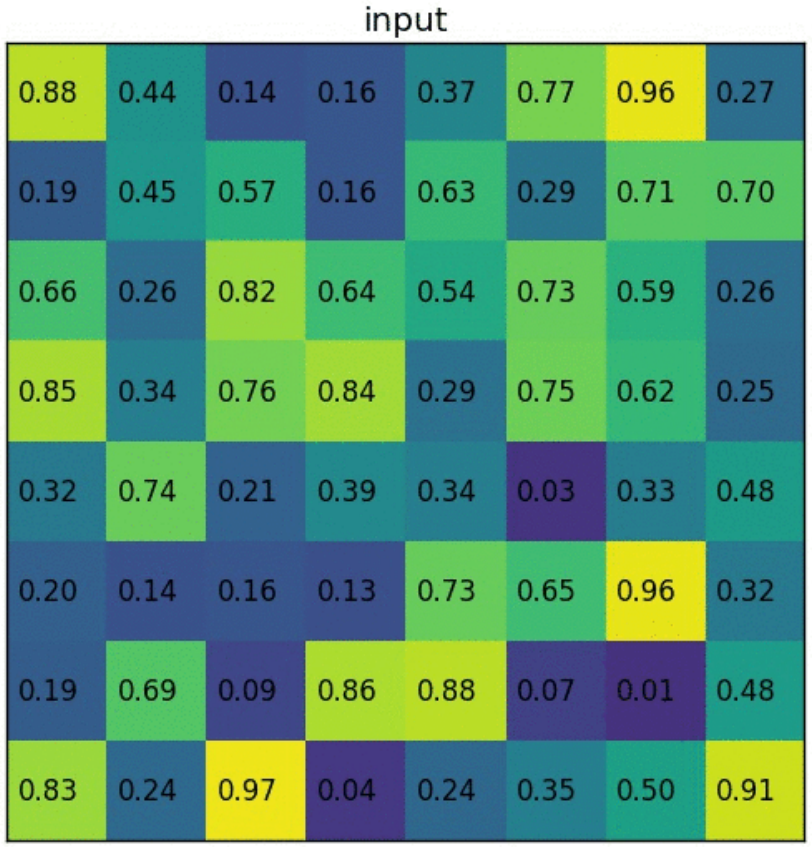
Pooling



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

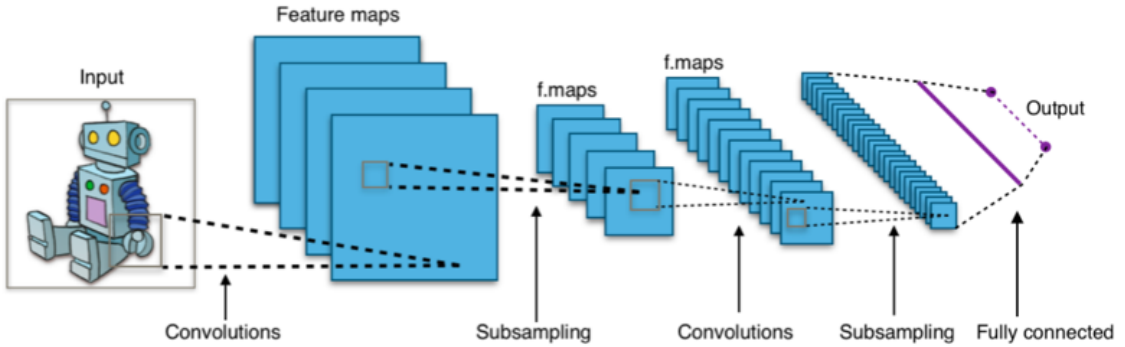
Pooling

- Variants:
  - Max pooling
  - Average



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

Typical architecture (simplified)

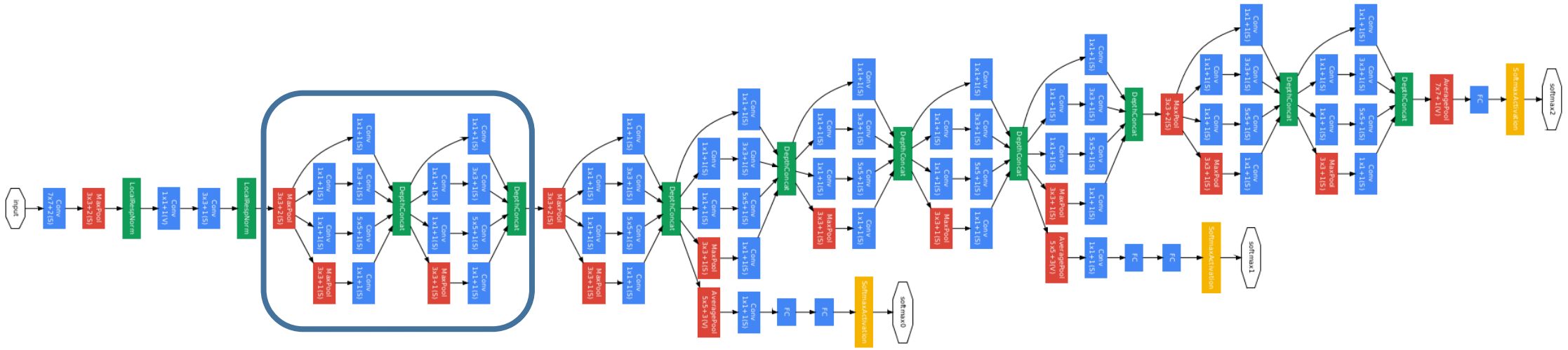


https://en.wikipedia.org/wiki/Convolutional_neural_network

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

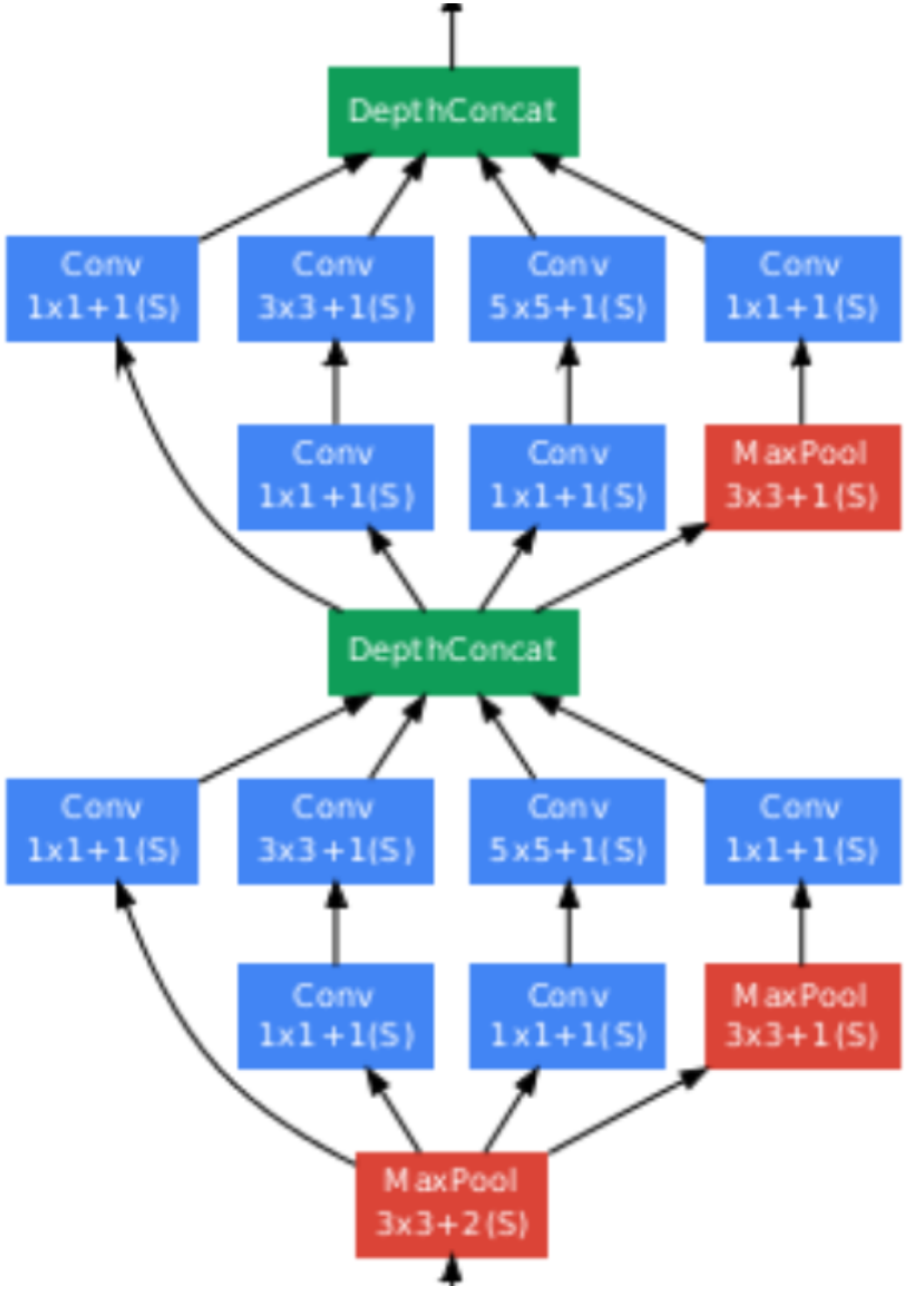
- This is more an art than science
- The better and better solutions to Image Net in general grew larger and larger
- More than 100 layers

GoogLeNet



Enlarged at next page

GoogLeNet


DepthConcat
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
MaxPool 3x3+1(S)
DepthConcat
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
MaxPool 3x3+1(S)
MaxPool 3x3+2(S)

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 convolutional networks

''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

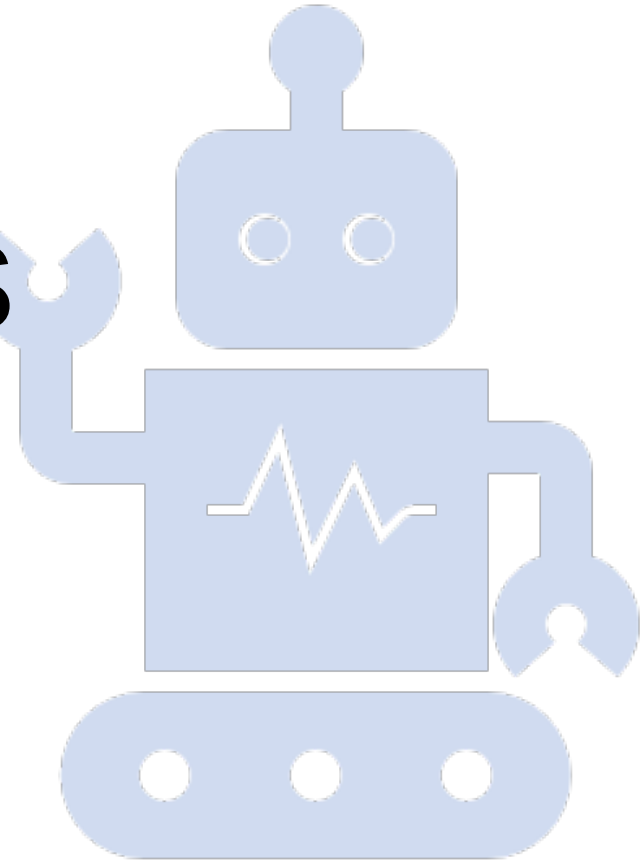
Convolutional NNs

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

Where can I learn more?

- IN5400 – Machine Learning for Image Analysis (spring semester)
- On the web, e.g.,
  - Stanford: CS231n: Convolutional Neural Networks for Visual Recognition
    - (Coursera course)

12.4 Recurrent NNs and language processing



IN3050/IN4050 Introduction to Artificial Intelligence and Machine Learning

RNNs

- Recurrent Neural Networks
- Applications in Language Technology, examples
- The small print

‘*The times they are a-changin*’

There are two things that we can do: add some backwards connections, so that the output neurons connect to the inputs again, or add more neurons. The first leads into recurrent networks. These have been studied, but are not commonly used. (Marsland, p. 71)

- Today, recurrent networks (RNNs) are heavily used, in Language Technology
- But they are already giving way to the Transformer architecture

Recurrent neural nets

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

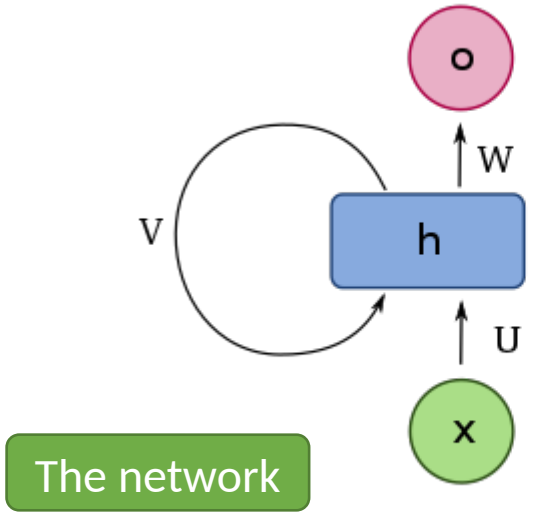


Image source: https://en.wikipedia.org/wiki/Recurrent_neural_network

Recurrent neural nets

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

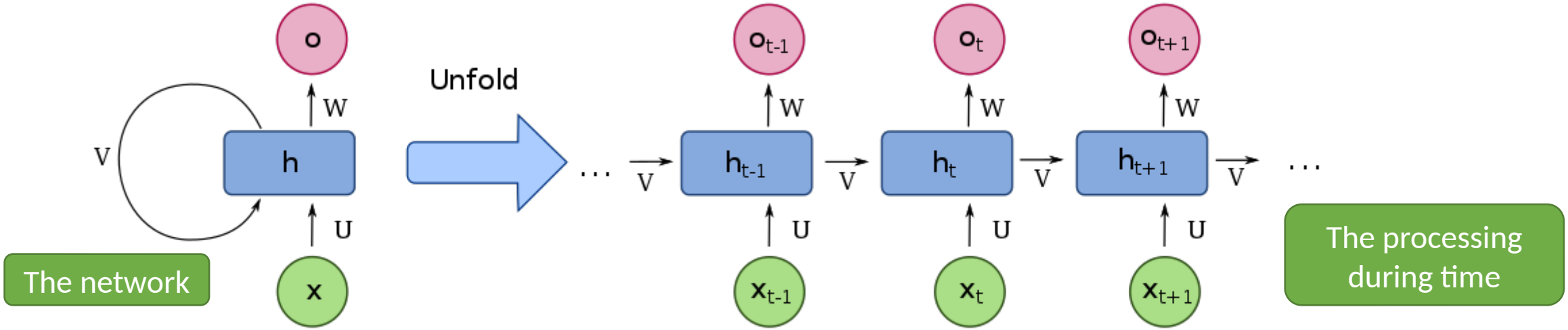
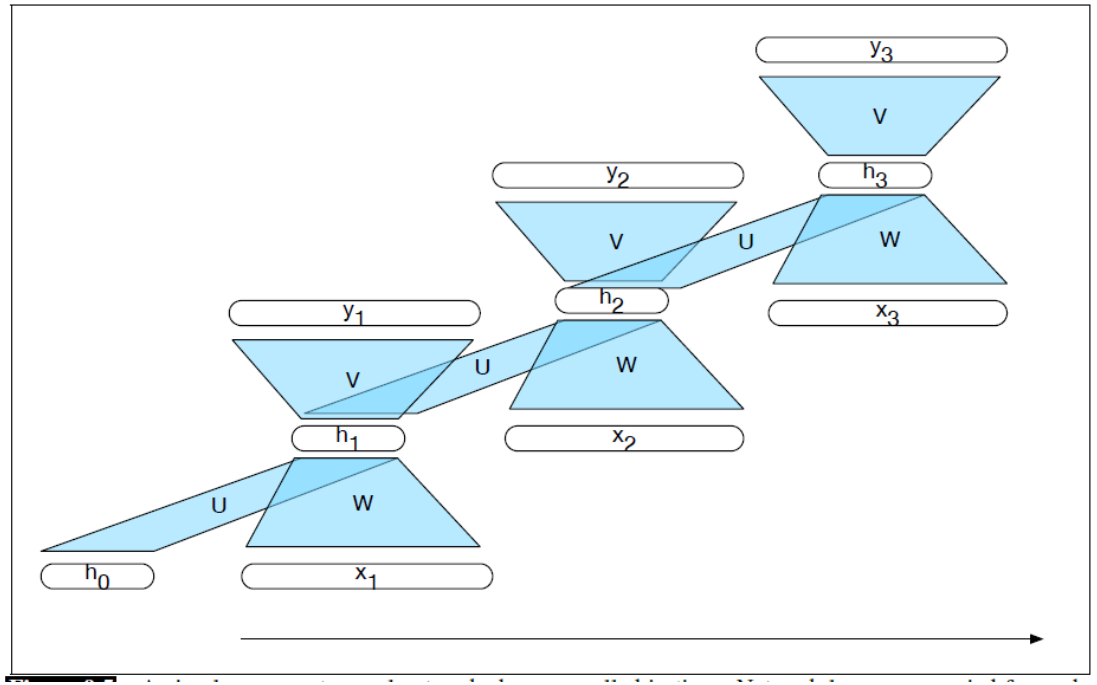


Image source: https://en.wikipedia.org/wiki/Recurrent_neural_network

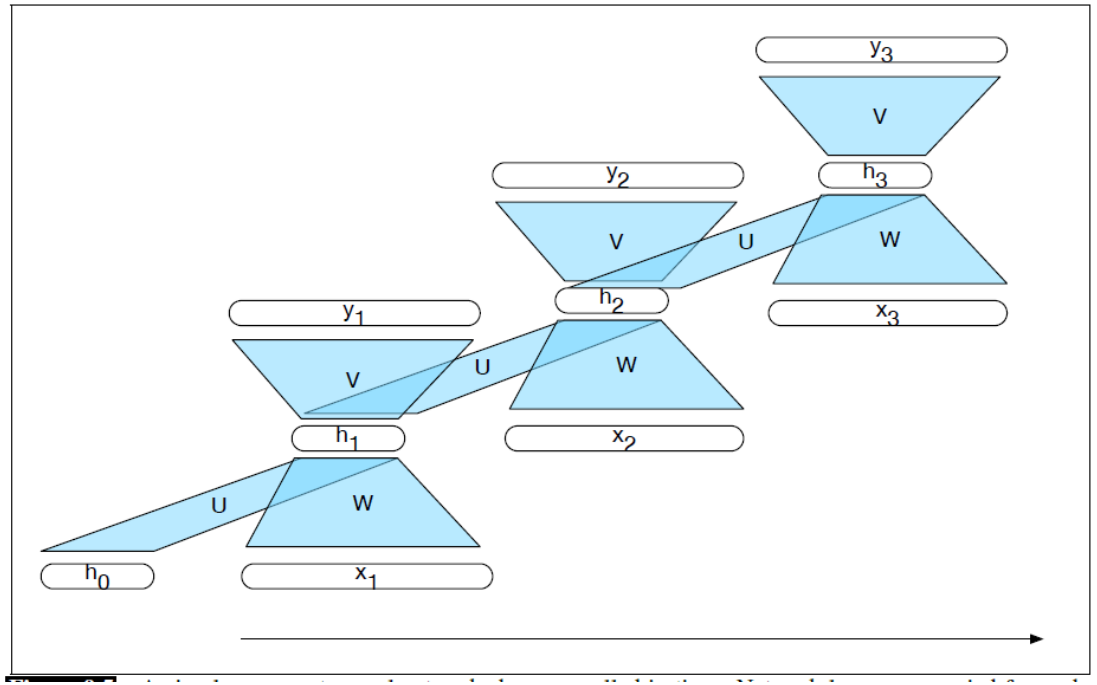
Forward



From J&M, 3.ed., 2019

- $x_1, x_2, \ldots, x_n$ 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 $i = 1, \ldots, n$

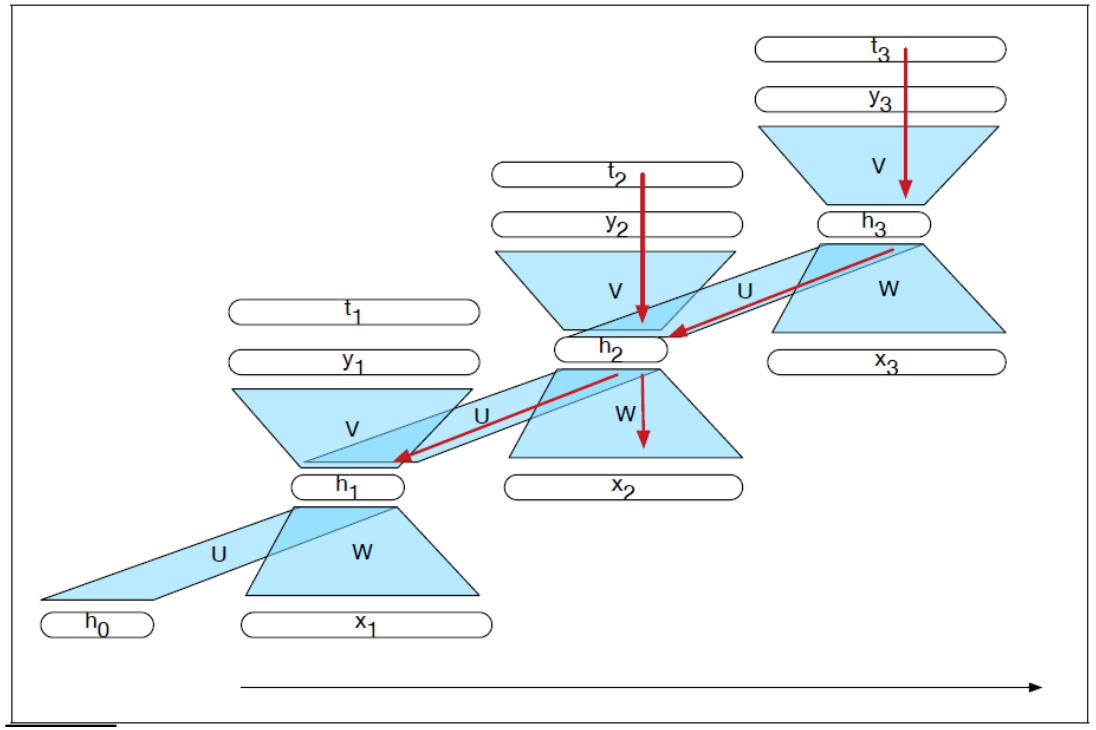
Forward



From J&M, 3.ed., 2019

- 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)

Training



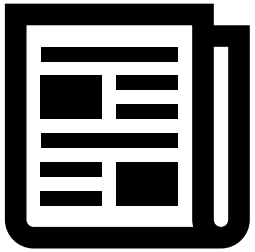
From J&M, 3.ed., 2019

- At <u>each output node</u>:
  - Calculate the loss and the
  - δ-terms
- Back-propagate the error, e.g.
  - the δ-terms at h_2 are calculated
    - from the δ-terms at h_3 by U and
    - the δ-terms at y_2 by V
- Update
  - V from the δ-terms at all the y_i-s
  - U, W from the δ-terms at all the h_i-s

RNNs

- Recurrent Neural Networks
- Applications in Language Technology, examples
- The small print

Lecture 5: Language model

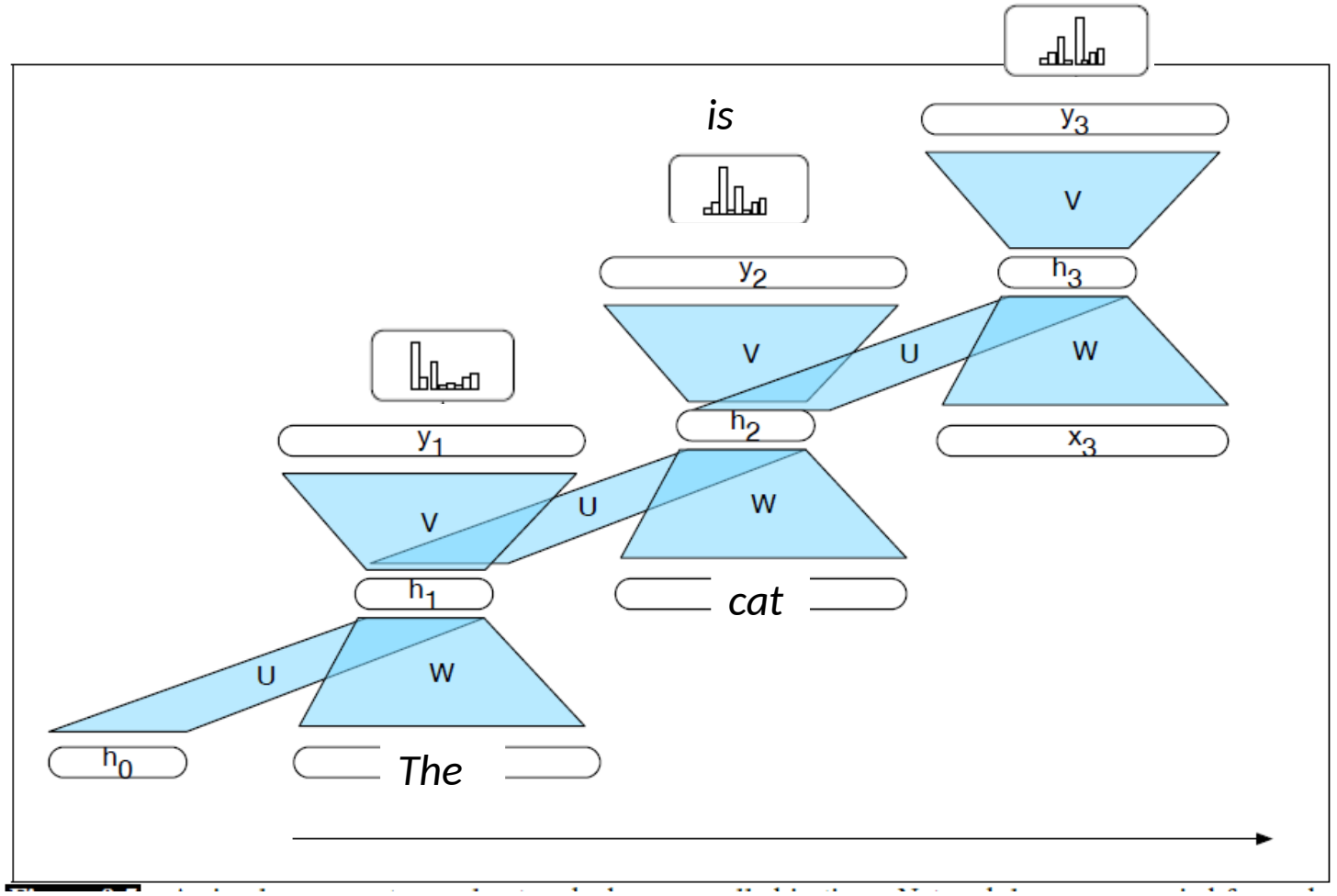


The cat is on the …

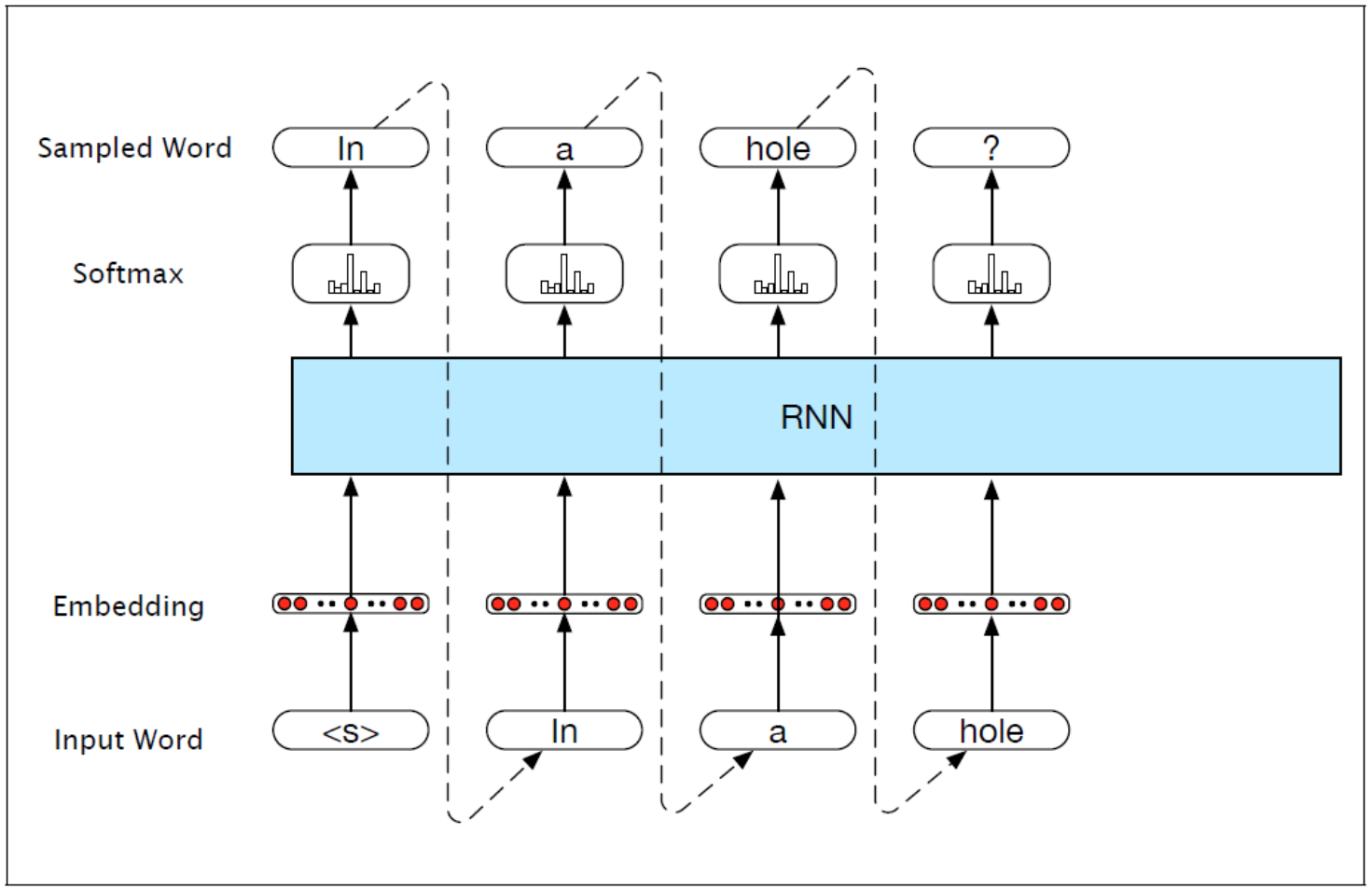
- Task: Predict the next word!
- Labels:
  - A vocabulary of words:
    - *aardvark*, …, *mat*, …, *roof*, …
    - E.g., 100,000 words/labels
- Domain:
  - Finite sequences of words

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

RNN Language model



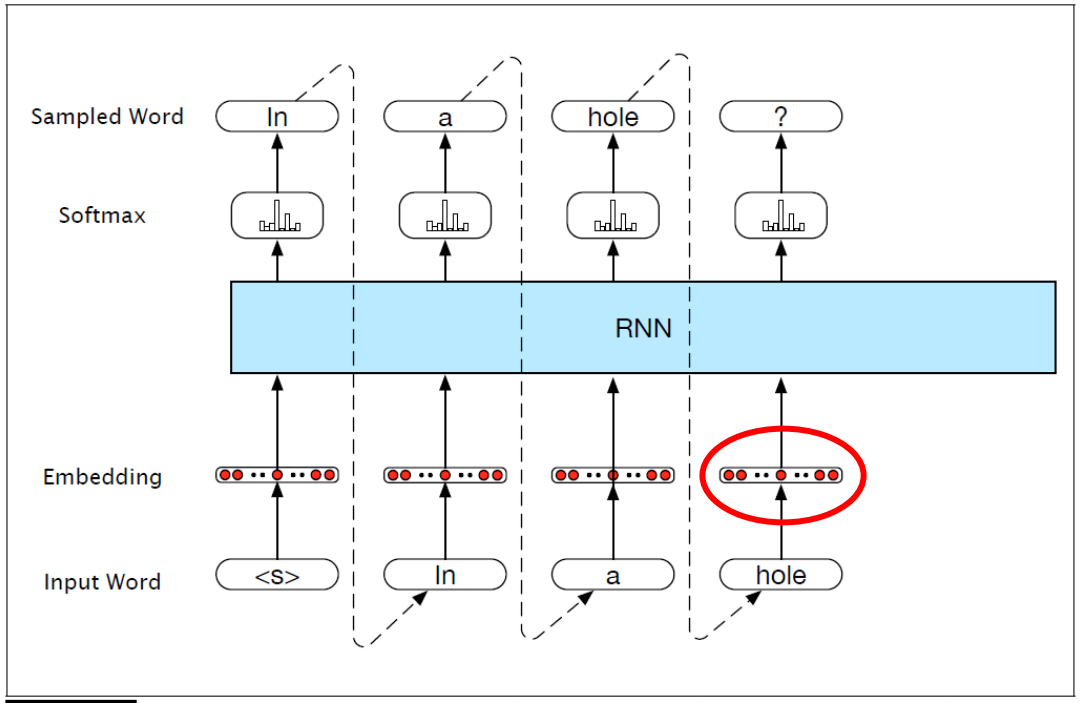
Autoregressive generation



- 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

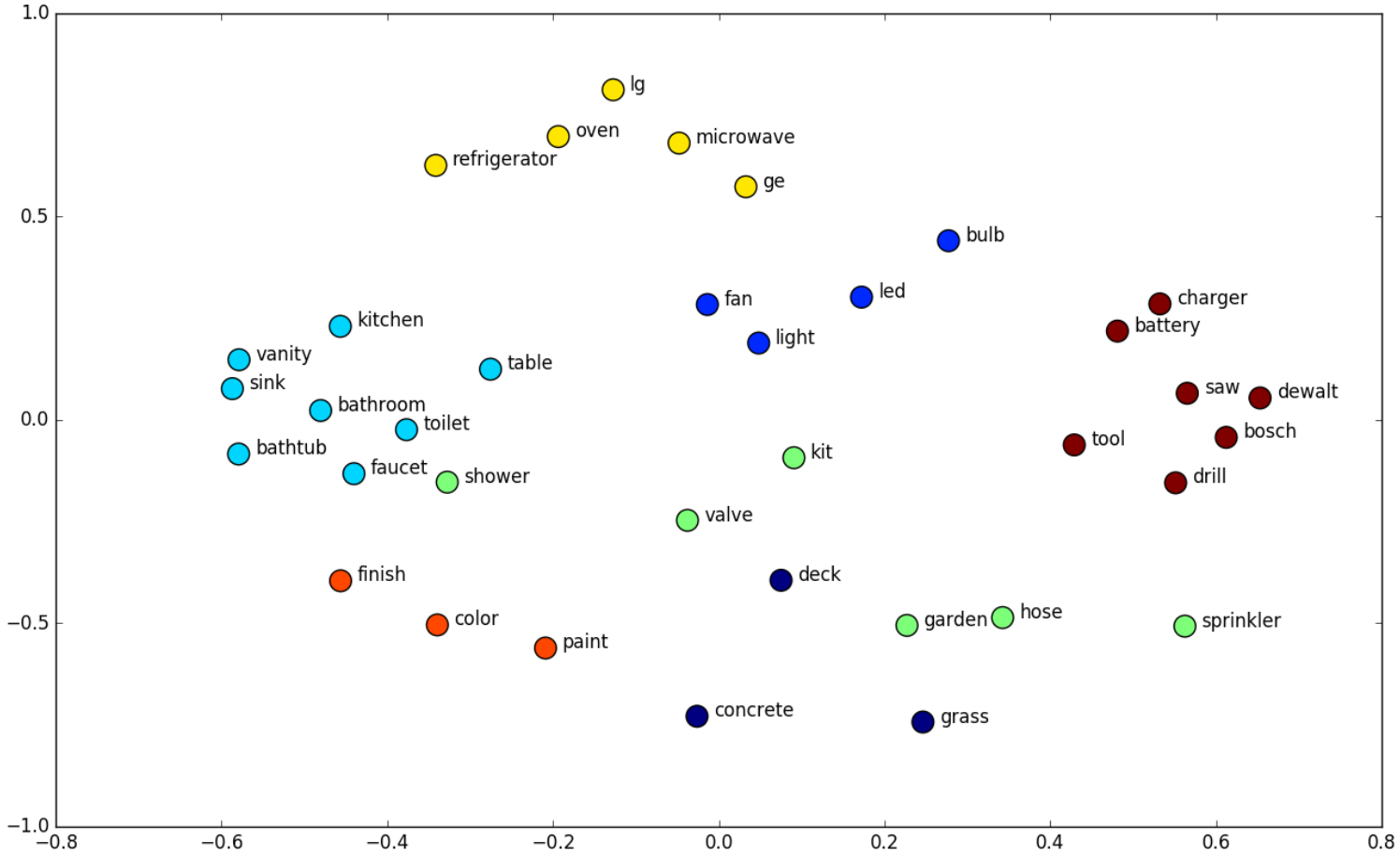
- A word is a categorical feature.
- We assume a vocabulary of e.g., 100,000 different words.
- We could use a ”one-hot” encoding (”one out of *n*”):
  - $(0, 0, 0, \ldots, 1, \ldots, 0)$
  - One 1 and 99,999 many 0-s
  - Different words, different positions
- But:
  - Inefficient, so many features and weights
  - No sharing between similar words

Embeddings

- Represent each word with a vector of
  - Reals
  - A fixed number of positions, 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 representations (**embeddings**) can be learned from a form of language modeling task:
  - Predict whether a neighboring word can occur together with this word
  - We skip the details

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
- Train an RNN on these concatenated pairs
- Apply by reading a source sentences and from there predict a target sentence

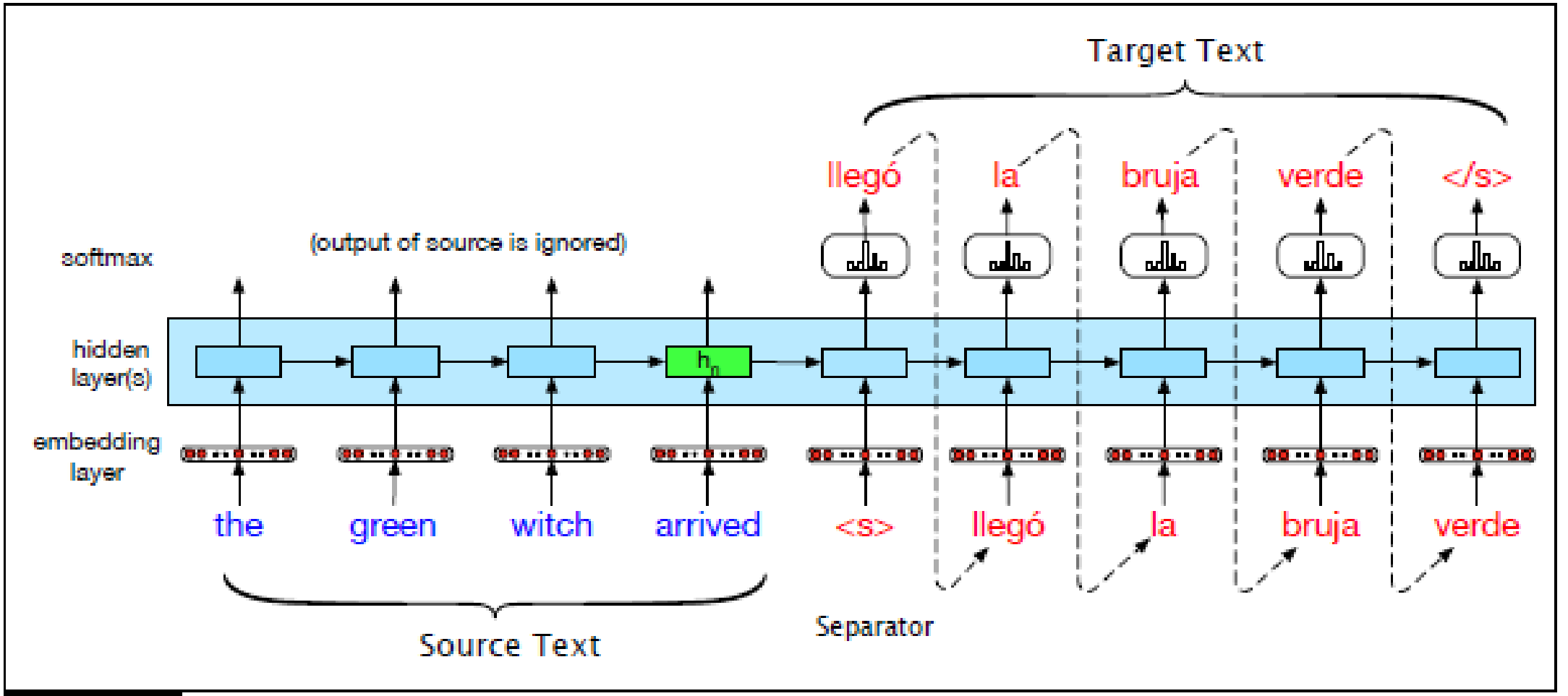


Figure 11.4 Translating a single sentence (inference time) in the basic RNN version of encoder-decoder approach to machine translation. Source and target sentences are concatenated with a separator token in between, and the decoder uses context information from the encoder's last hidden state.

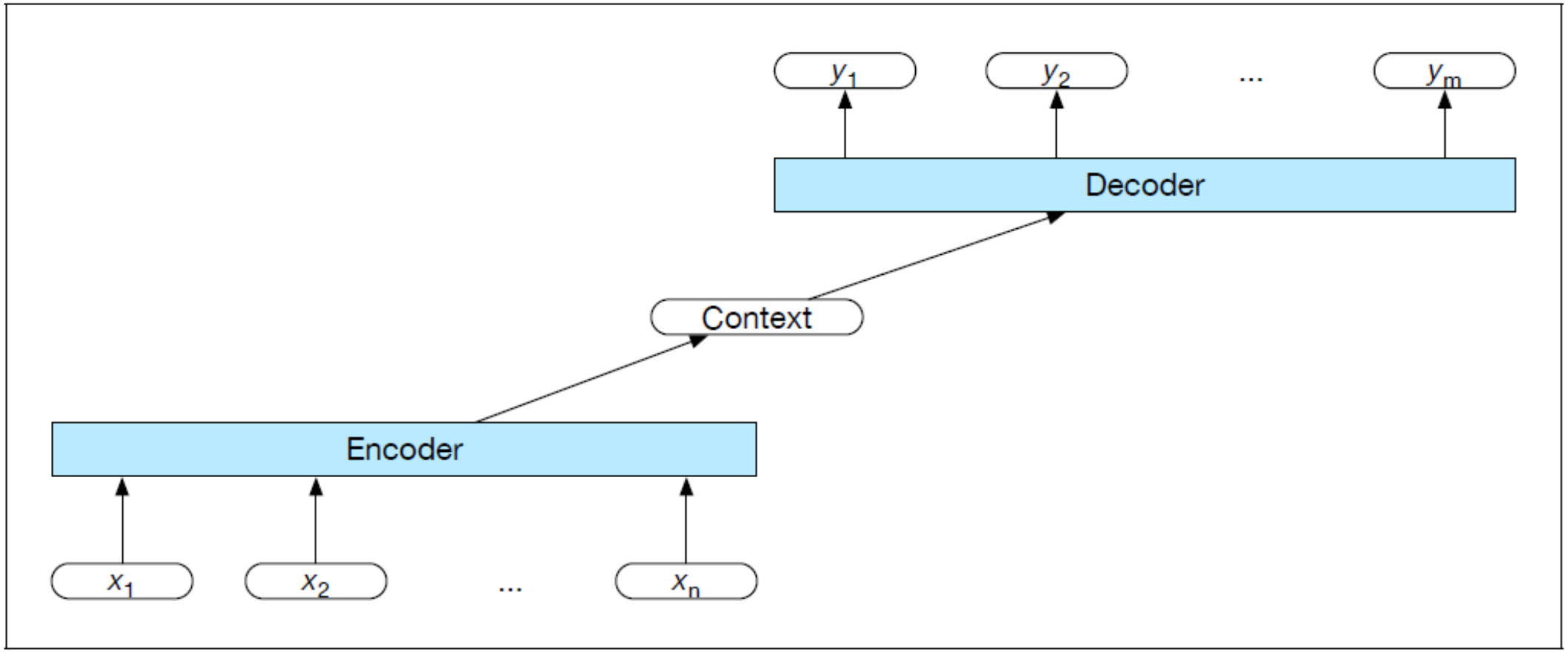


Figure 10.4 Basic architecture for an abstract encoder-decoder network. The context is a function of the vector of contextualized input representations and may be used by the decoder in a variety of ways.

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

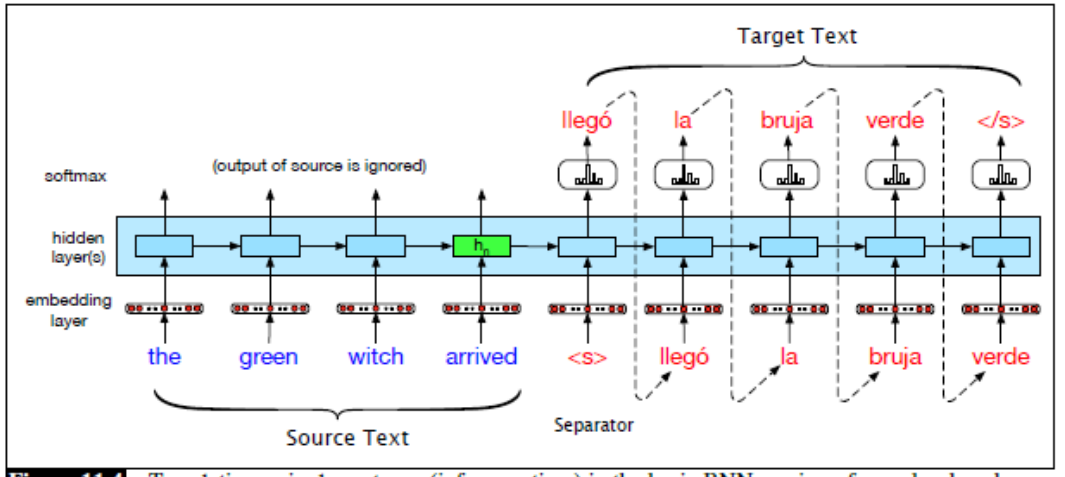


Figure 11.4 Translating a single sentence (inference time) in the basic RNN version of encoder-decoder approach to machine translation. Source and target sentences are concatenated with a separator token in between, and the decoder uses context information from the encoder's last hidden state.

RNNs

- Recurrent Neural Networks
- Applications in Language Technology, examples
- The small print

Warnings: We have simplified

1. Long Short-Term Memory (LSTM)
   - RNNs have problems with
     - Keep track of distant information
     - Vanishing gradients.
   - One way out is LSTMs
   - An advanced architecture with additional layers and weights
     - Not consider the details here
2. Often one uses several LSTMs
   - E.g., BiLSTM:
     - One LSTM left to rights
     - One LSTM going right to left
   - Stacked LSTMs
     - Several layers with LSTMs

Warnings: We have simplified

3. The models for translation are more complex, using more connections between the first and the second sentence
4. There have been proposed other deep learning architectures for machine translation

Where can I learn more?

- IN4080 – Natural Language Processing (fall semester)
  - Broad approach to NLP using both rules, traditional ML and some deep learning
- IN5550 – Neural Methods in Natural Language Processing (spring)
  - Advanced, deep learning
  - HPC, large data sets
  - Focus on large language models and Transformer architecture