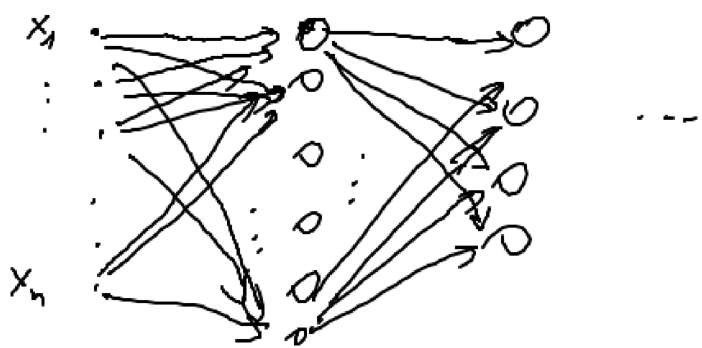
Neural nets



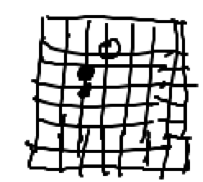
So far we have discussed neural nets with dense layers, i.e., layers in which all neurons are connected to all inputs

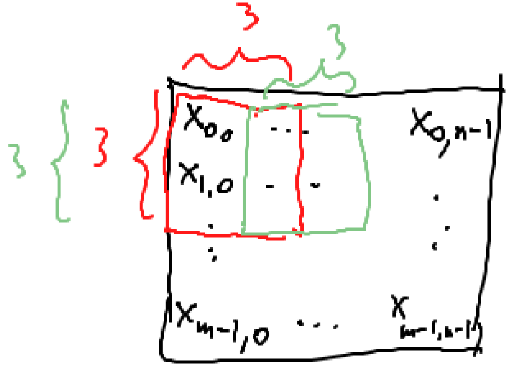
Convolutional neural networks

'80 Yann LeCun

Idea: exploit the 2-dim. nature of the input data

Usage: image recognition, pattern recognition

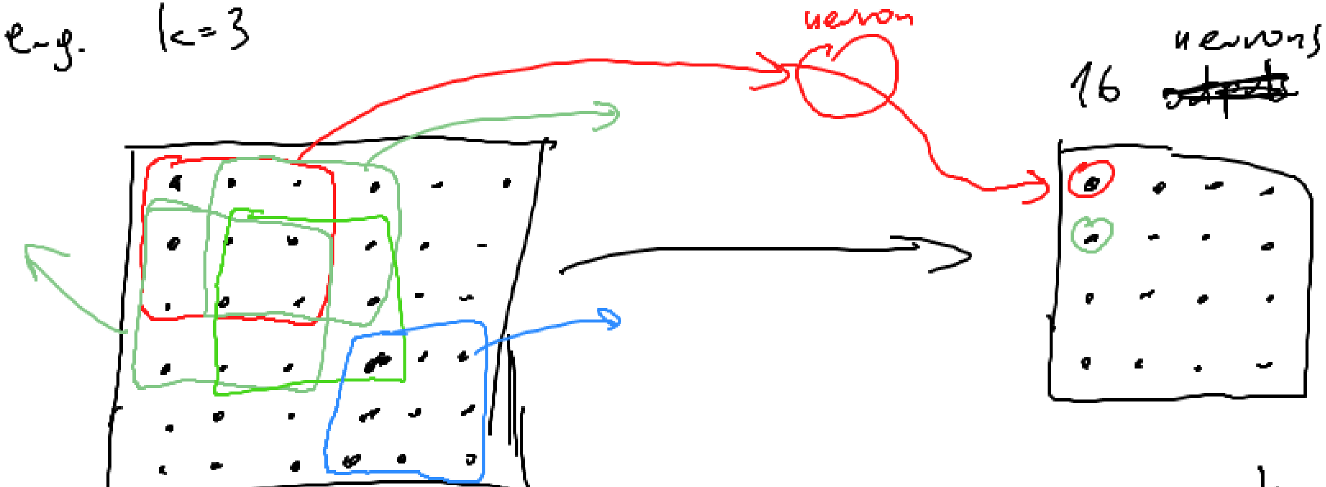




Dense layer would treat this image data as a sequence $(x_{ij})_{i=0,\ j=0}^{n-1,\ n-1}$

Convolution layer with filter size $k \times k$.

e.g. $k=3$



The weights are the same for each of the neurons

-1	1	-1
-1	1	-1
-1	1	-1

In this example we would have just 10 weights (9 for the input + 1 for the bias)

Weights: $(a_{ij})_{i,j=0}^{k-1}$, b - bias

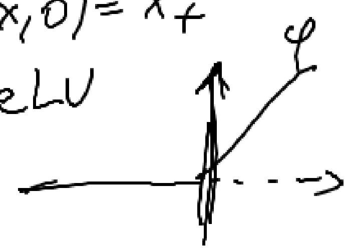
For $p = 0, \dots, m-k$; $q = 0, \dots, n-k$ we put

$$net_{p,q} = \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} a_{ij} \, x_{i+p,\, j+q} + b$$

$$output_{p,q} = \varphi(net_{p,q})$$

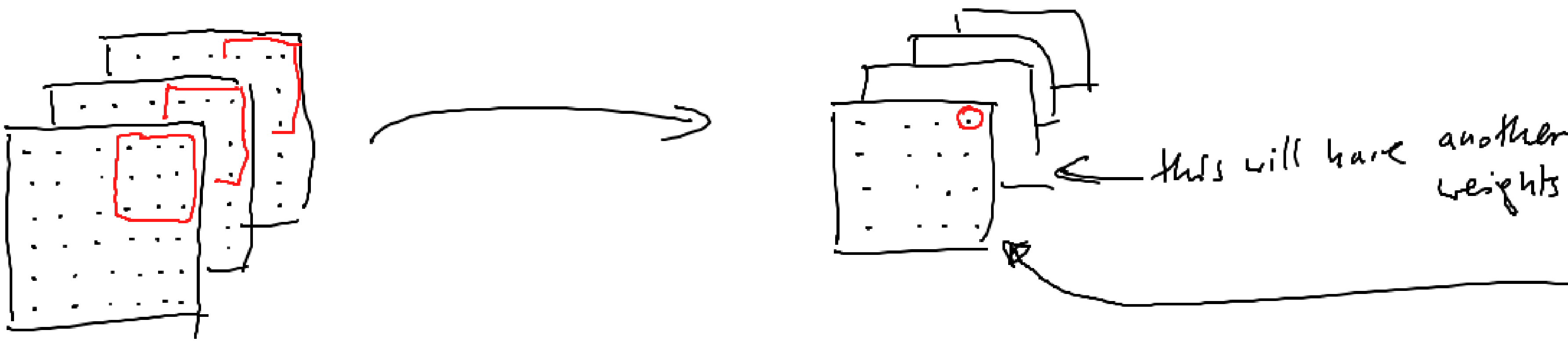
In CNN one usually takes $\varphi(x) = \max(x, 0) = x_+$

ReLU



We have discussed a setup with just 1 input channel and 1 output channel

If the input consists of several matrices (channels):



For each channel in the input we will have a separate set of weights

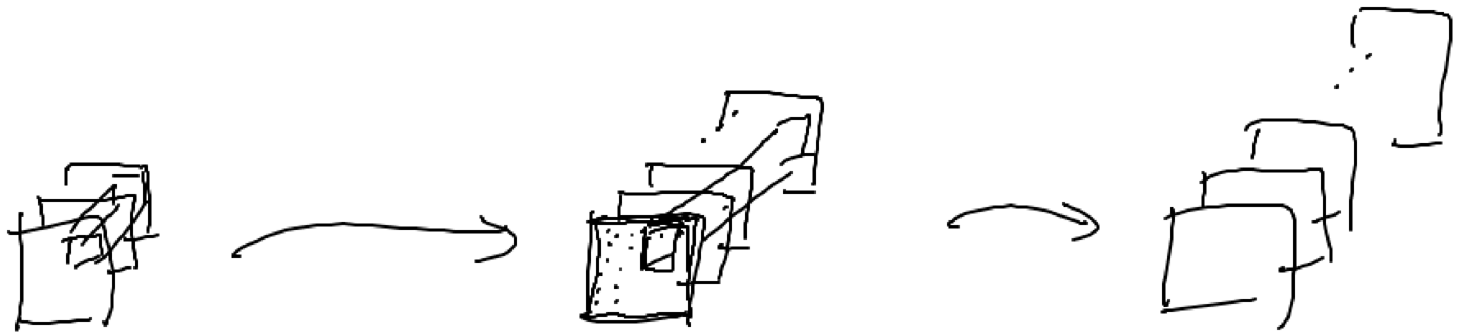
$$\left(a_{ij}^{(f)}\right)_{i,j=0}^{k-1}, \quad b, \quad f = 1, \dots, F$$

↑ number of input channels

$$net_{p,q} = \sum_{f=1}^{F} \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} a_{ij}^{(f)} \cdot X_{i+p,j+q}^{(f)} + b$$

$$output_{p,q} = \varphi(net_{p,q})$$

Then



input data — 1. layer

Example: input: 3 planes (channels) 100x100

kernel size: 5x5

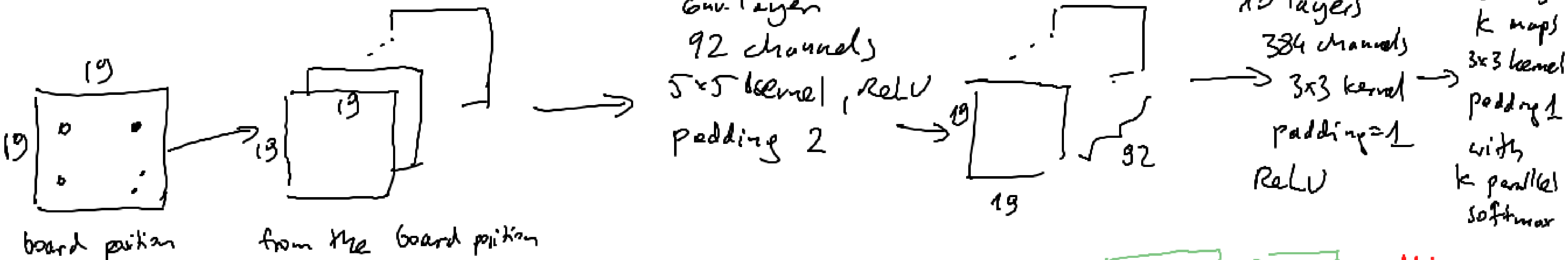
output: 64 planes (channels) 96x96 - size → in the ~~output~~ layer we have $64 \cdot 96 \cdot 96$ neurons

number of weights $= 64 \cdot (5 \cdot 5 \cdot 3 + 1) = 64 \cdot 76 = 4864$

dense layer with N neurons would have: $(10000+1) \cdot N = 10001 \cdot N$

The training for conv. layer is much faster.

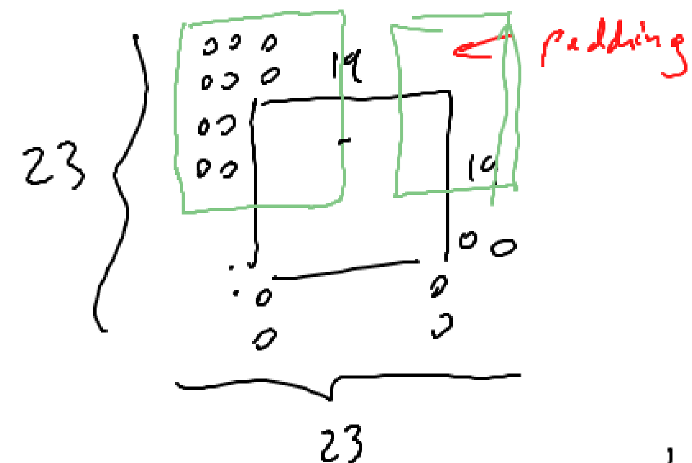
Example (almost pure CNN) – DarkForest, used for move prediction in the game Go

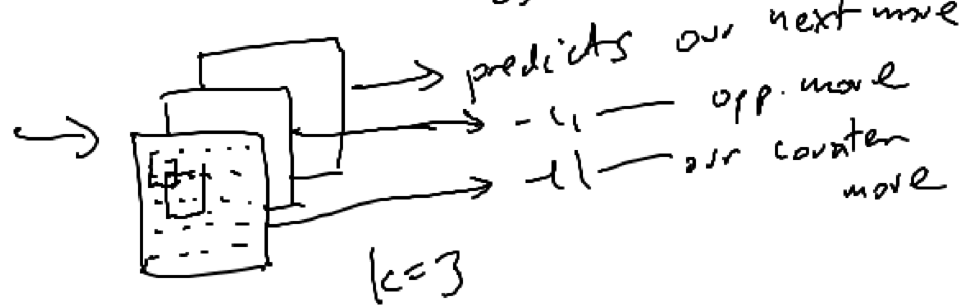


from the board position we extract 25 feature planes

for example:

1) the position of white stones
2) –"– black stones
3) –"– empty places
...) the white groups with 1 liberty
...) –"– 2 liberties
...) –"– ≥3 –"–





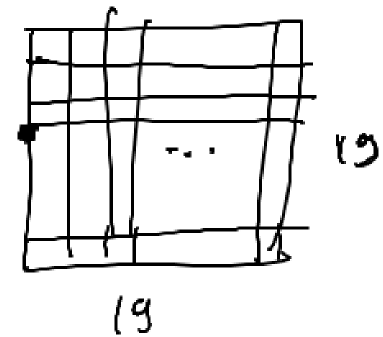
DarkForest – CNN developed around 2015

AlphaGo 2015–16 the first program which won with the best humans (Go on 19x19)

1996–97 Deep Blue (IBM)

↑ Kasparov won ↑

Go



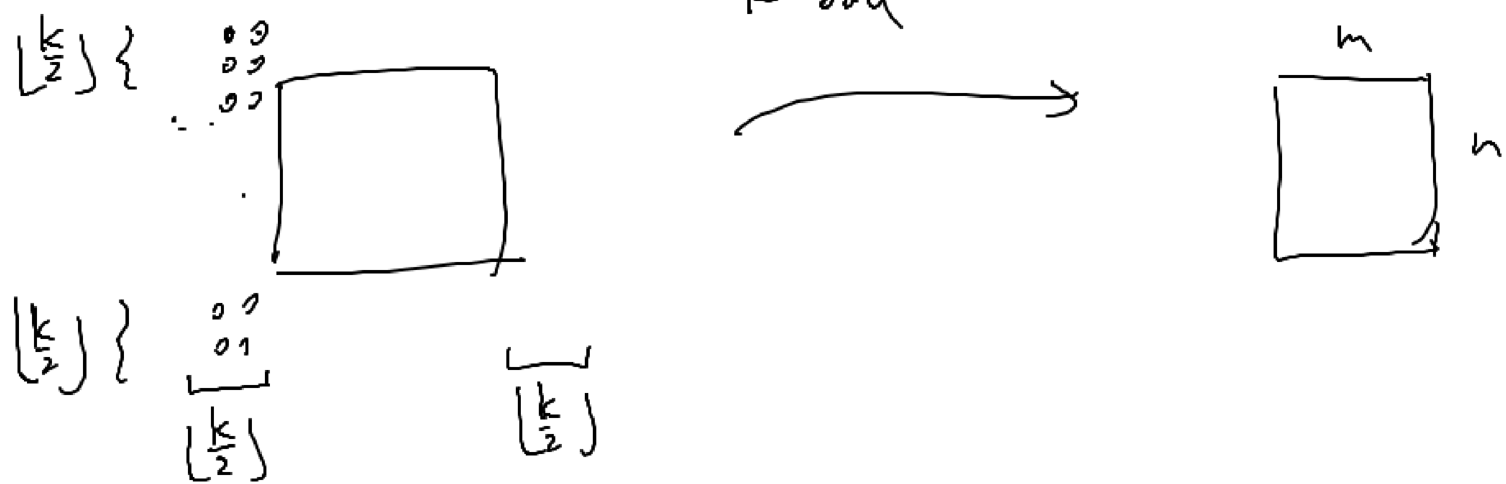
361 of possibilities for the first move

without padding

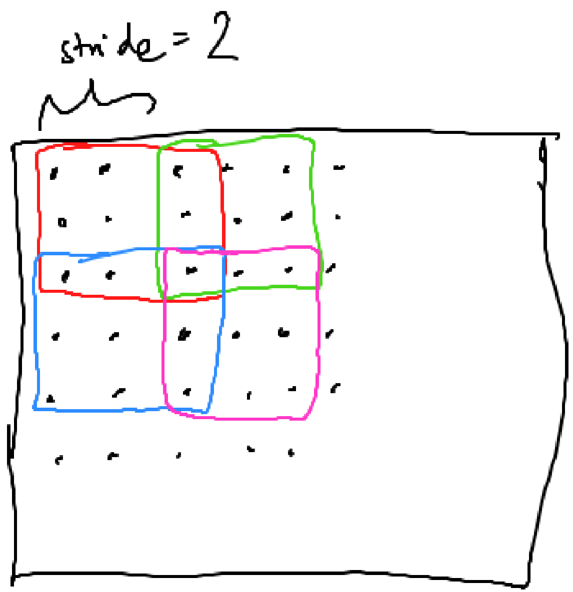
kernel size
$k \times k$

with padding one can obtain the same size

k - odd



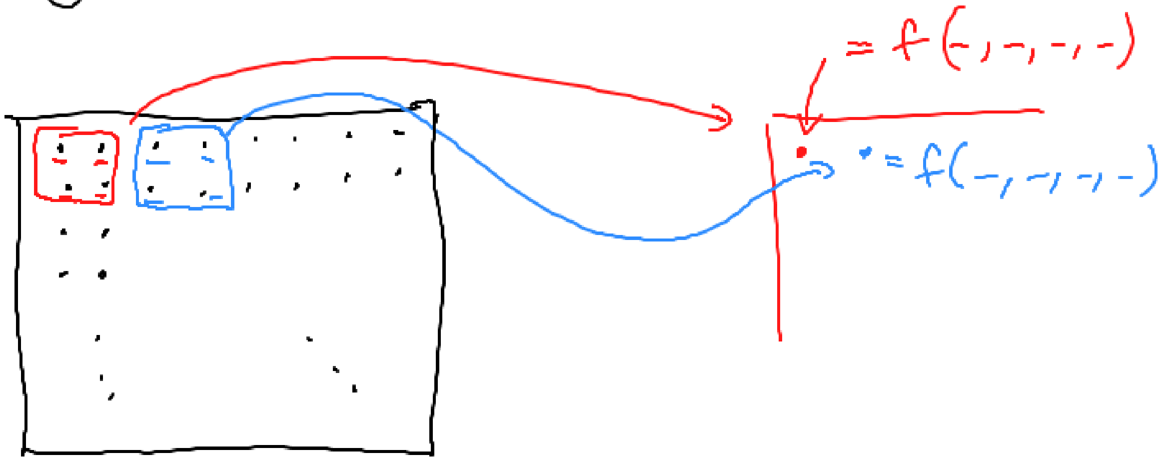
- strides: so far we considered only stride = 1



stride = 2 ~~tha~~ more-less halves the width and the height of the image

reducing the dimensions by a factor ≈ 4

- Pooling layers — reduces the dimensions in a different way



e.g. f = max

f = average

usually max pooling is used

- often a dense layer is used as the last one

CNN → conv. layer → conv. layer → ... → conv. layer → dense ↓ dense ↓ dense

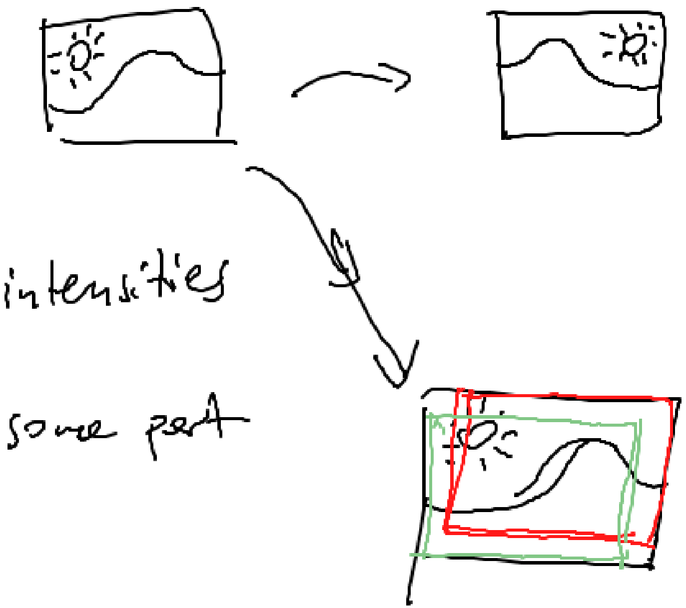
pooling is possible here

- Training: back-propagation as for the classical NN

- Data augmentation

  for example, use reflections

  - (slight) rotations
  - (slight) changes in the pixel intensities
  - expand slightly and then crop some part



<u>Remark</u>: convolutional layers can be applied to inputs of different sizes

Example 2012 ReLU in all conv./dense layers, except the last one, in which softmax was used

AlexNet