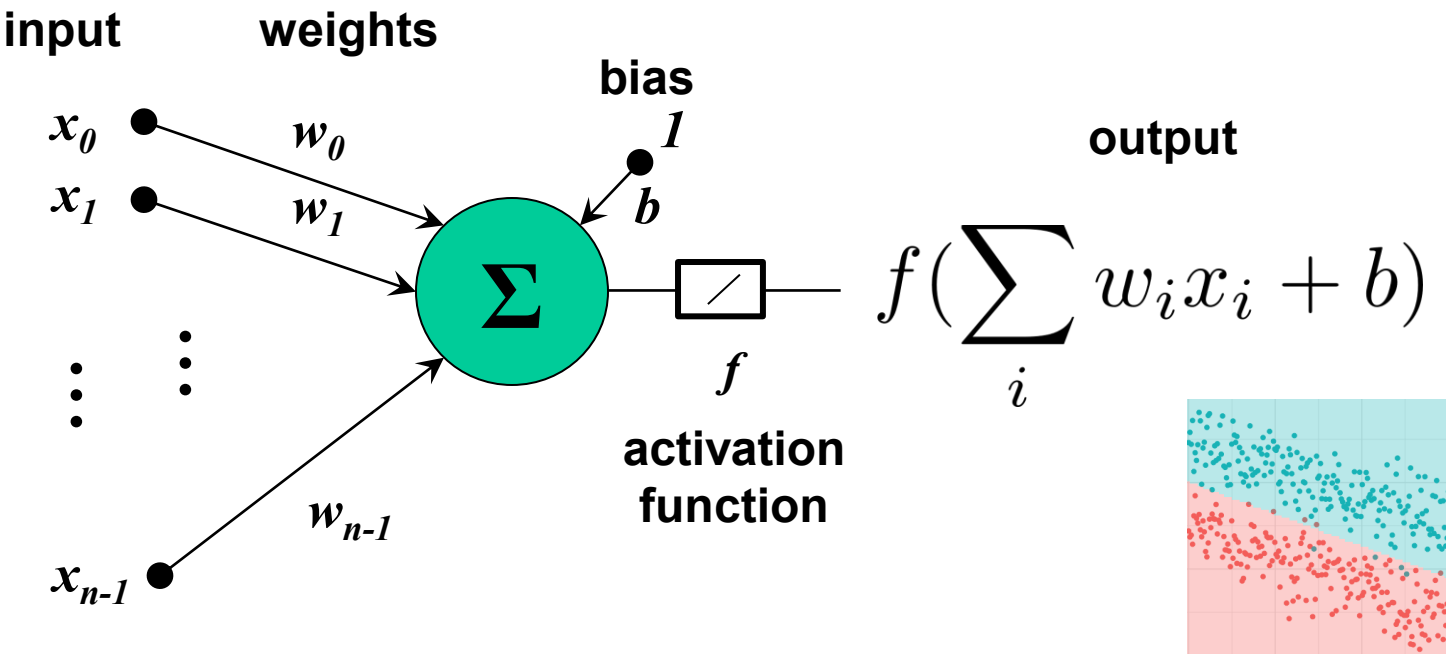
Convolutional Neural Networks

Andrej Lúčny

Image representation. Image processing using kernels. Implementing kernels using convolutional neural networks (CNNs). Implementing parallel perceptrons sharing weights via convolutional layers

`https://github.com/andylucny/OAI`

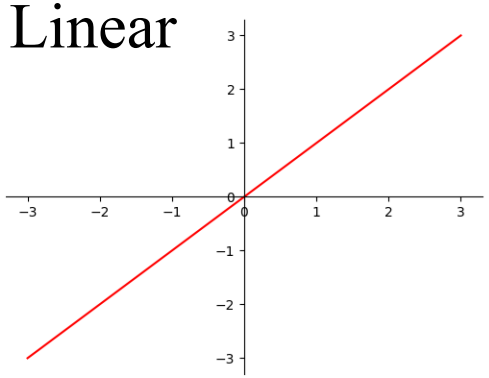
Neuron – Linear Regressor



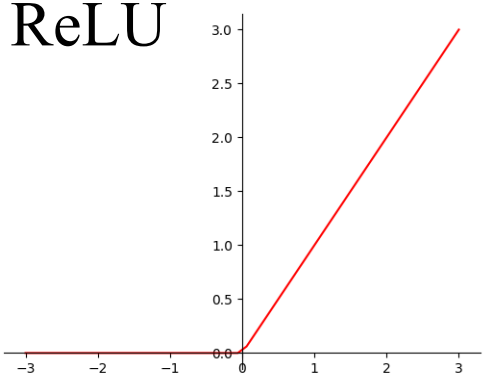
- a neuron computes the dot product of inputs and weights, adds a bias, and applies an activation function
- Activation functions may include: Linear, Sigmoid, Hyperbolic tangent (tanh), ...

Activation Functions

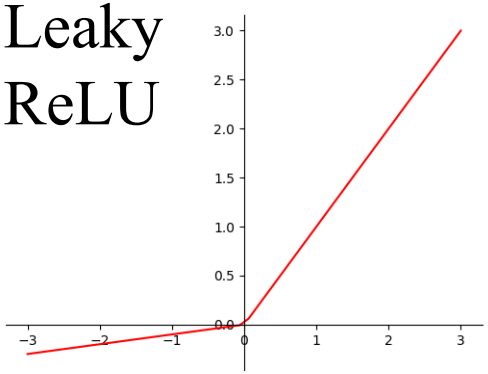
Linear



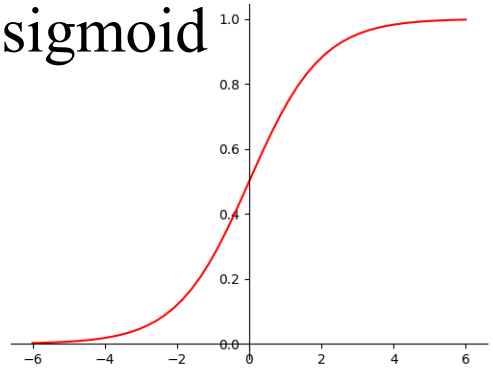
ReLU



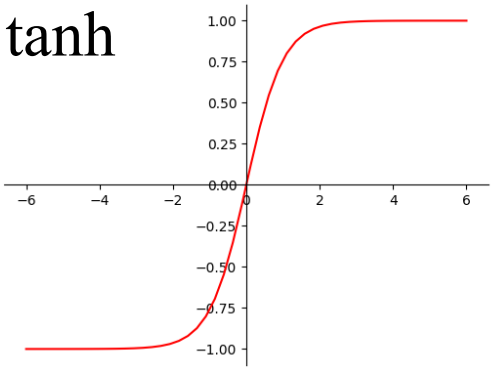
Leaky ReLU



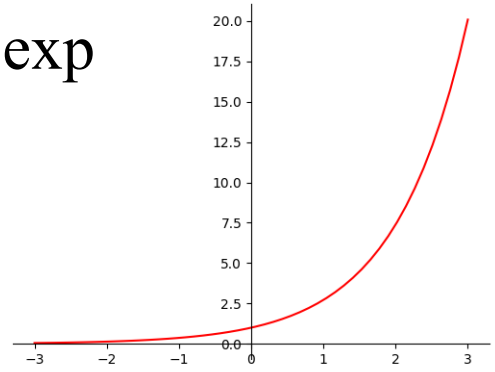
sigmoid



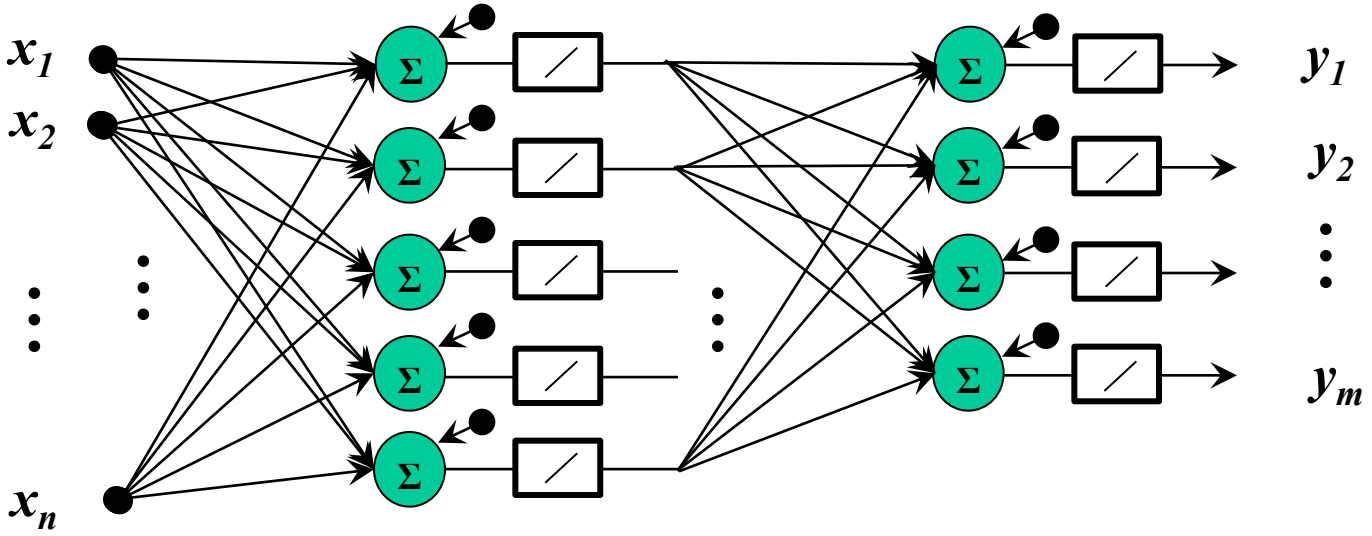
tanh



exp



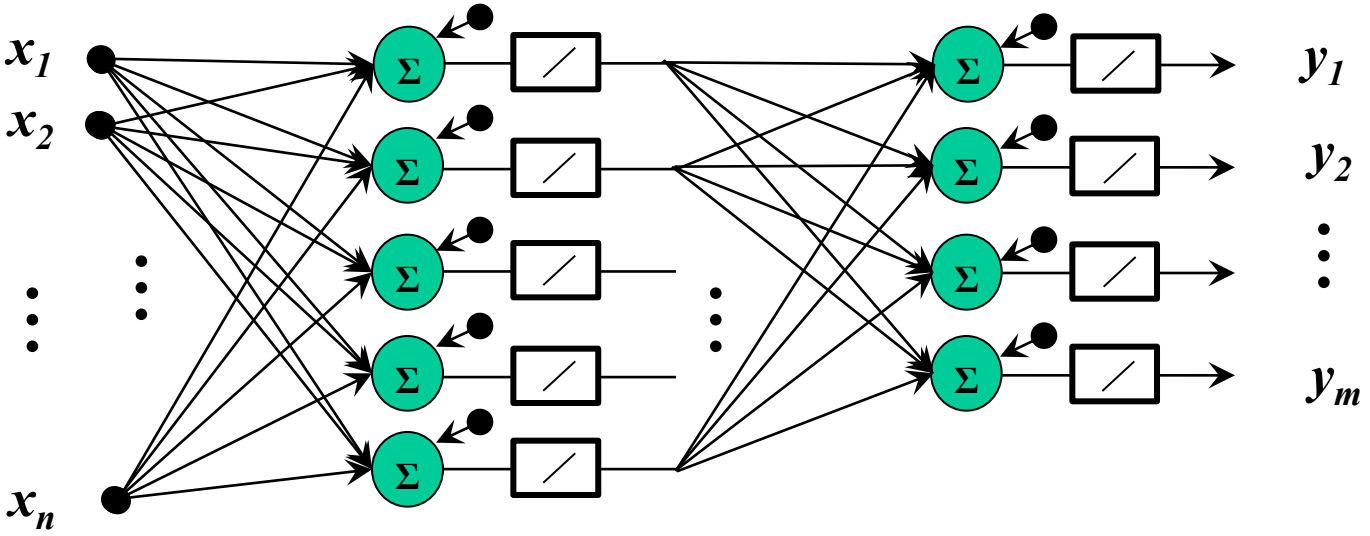
Perceptron



- single hidden layer and one output layer
- only linear activation
- only output layer weights can be trained

[1958 Rosenblatt]

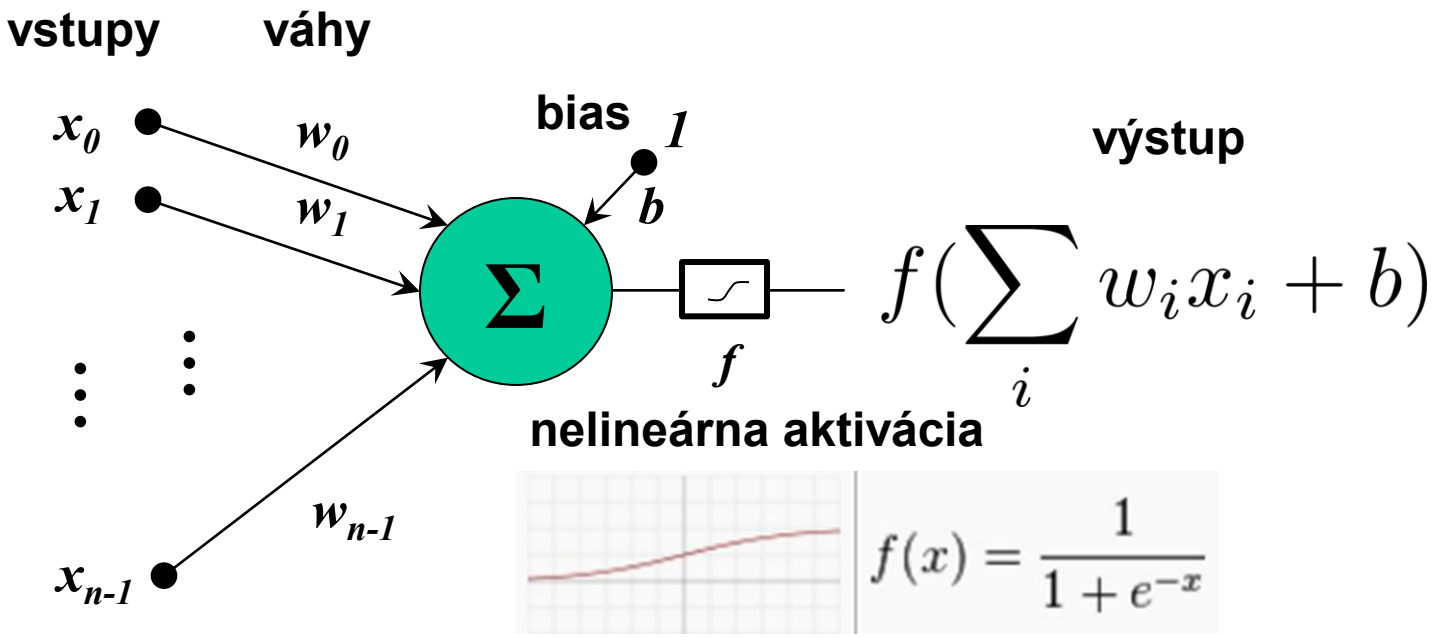
Perceptron



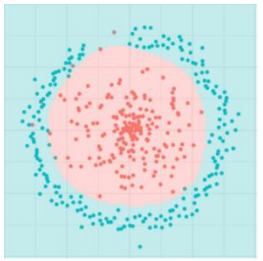
- Analysis showed perceptrons are very limited
- Later it was discovered that small modifications significantly improve capability

[1968 Minsky]

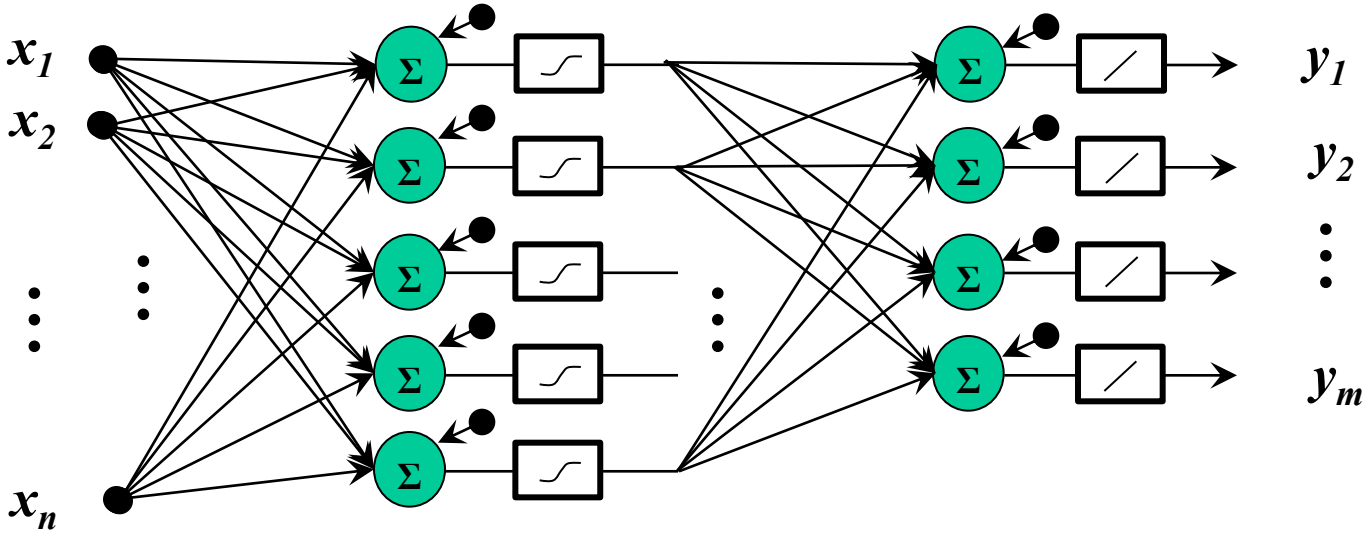
Neuron – Logistic Regressor



- If we replace linear activation with sigmoid (logistic function) or hyperbolic tangent, training corresponds to logistic regression.
- The neuron becomes much more powerful as a building block



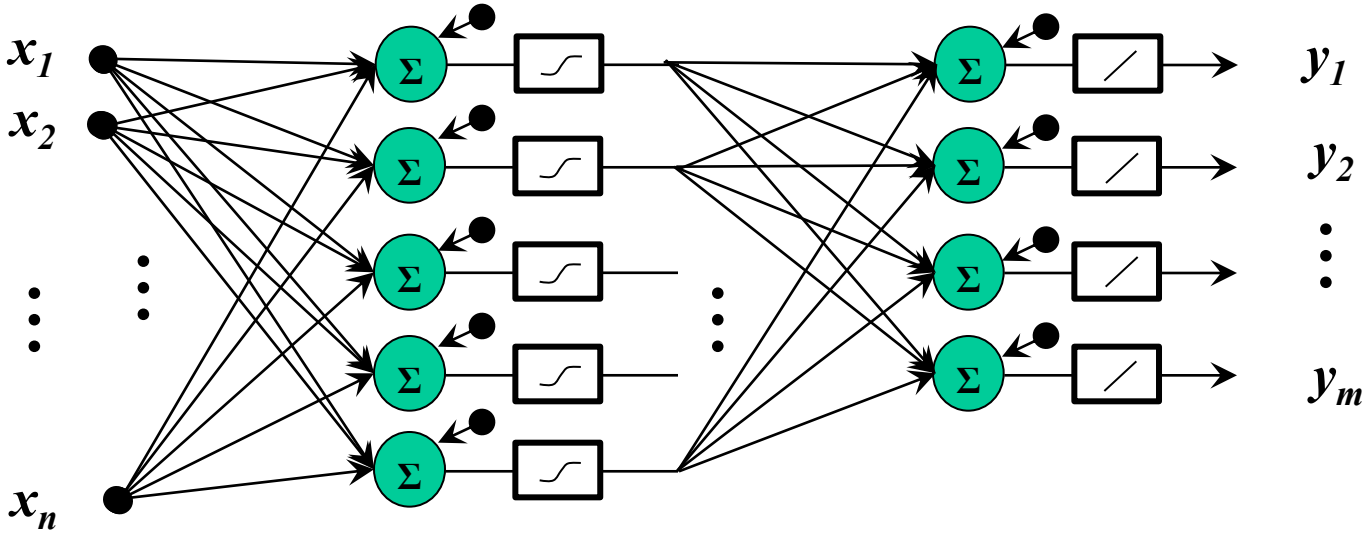
Perceptron + nonlinearity + BP



- Backpropagation algorithm, based on the chain rule [Leibnitz 1676], enables effective training of neural networks

[1986 Rumelhart, Hinton & Williams]

Univerzal approximation



- perceptron can theoretically approximate any continuous function on a compact interval [1989 Cybenko] [1989 Hornik]
- In practice: it works only for low-dimensional inputs, not feasible for images (e.g., 416×416×3 = 519168 dimensions)

Schedule

- We need to: Reduce image dimensionality without losing information
- Step 1: How do we represent images
- Step 2: How process images (via kernels)
- Step 3: How to reduce dimension

Grayscale image

resolution

$1 \times H \times W$

number of channels

W

H

intensity

range:

0 .. 255

0.0 .. 1.0

-1.0 .. 1.0

-0.5 .. 0.5

-3 .. 3

Color image

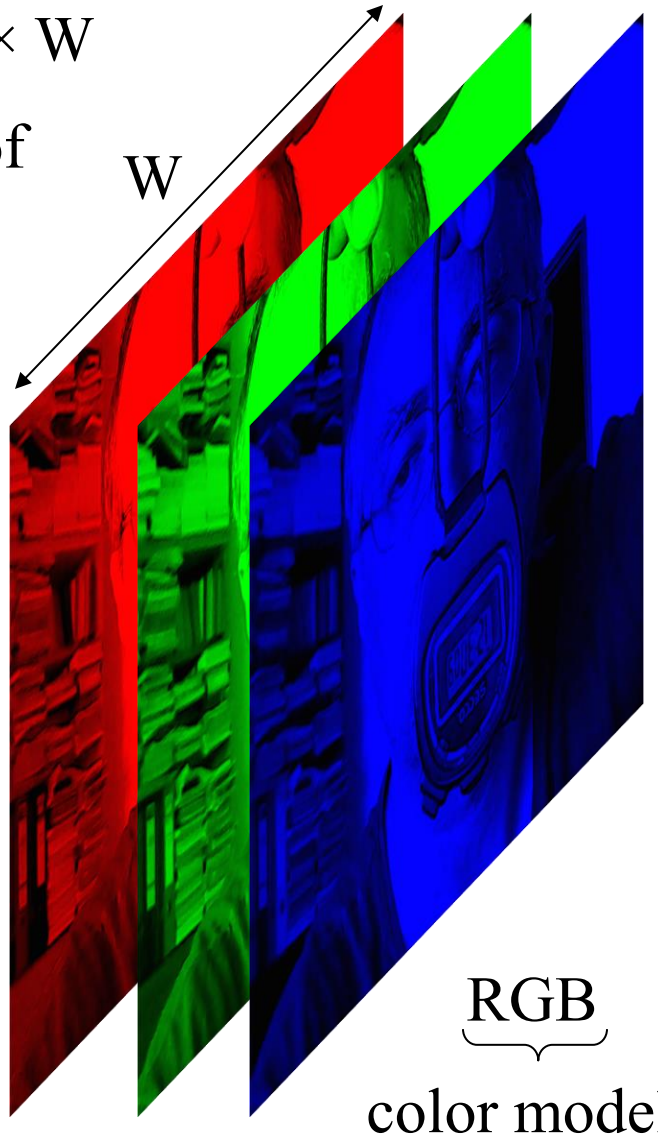
resolution

$C \times H \times W$

number of channels

W

H



RGB

color model

range:

0 .. 255

0.0 .. 1.0

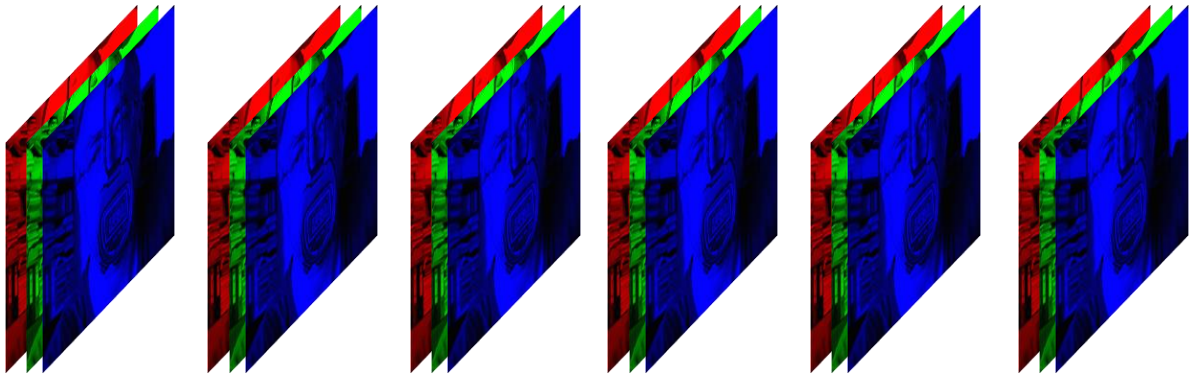
-1.0 .. 1.0

-0.5 .. 0.5

-3 .. 3

Batch

$B \times C \times H \times W$



Kernel

240x320x1

1x1x1

1.5

weight

0.15

bias

240x320x1

intensity x weight + bias

Kernel

240x320x1

1x1x1

1.5

weight

0

bias

240x320x1

weight is contrast

Kernel

240x320x1

1x1x1

1.0

weight

0.15

bias

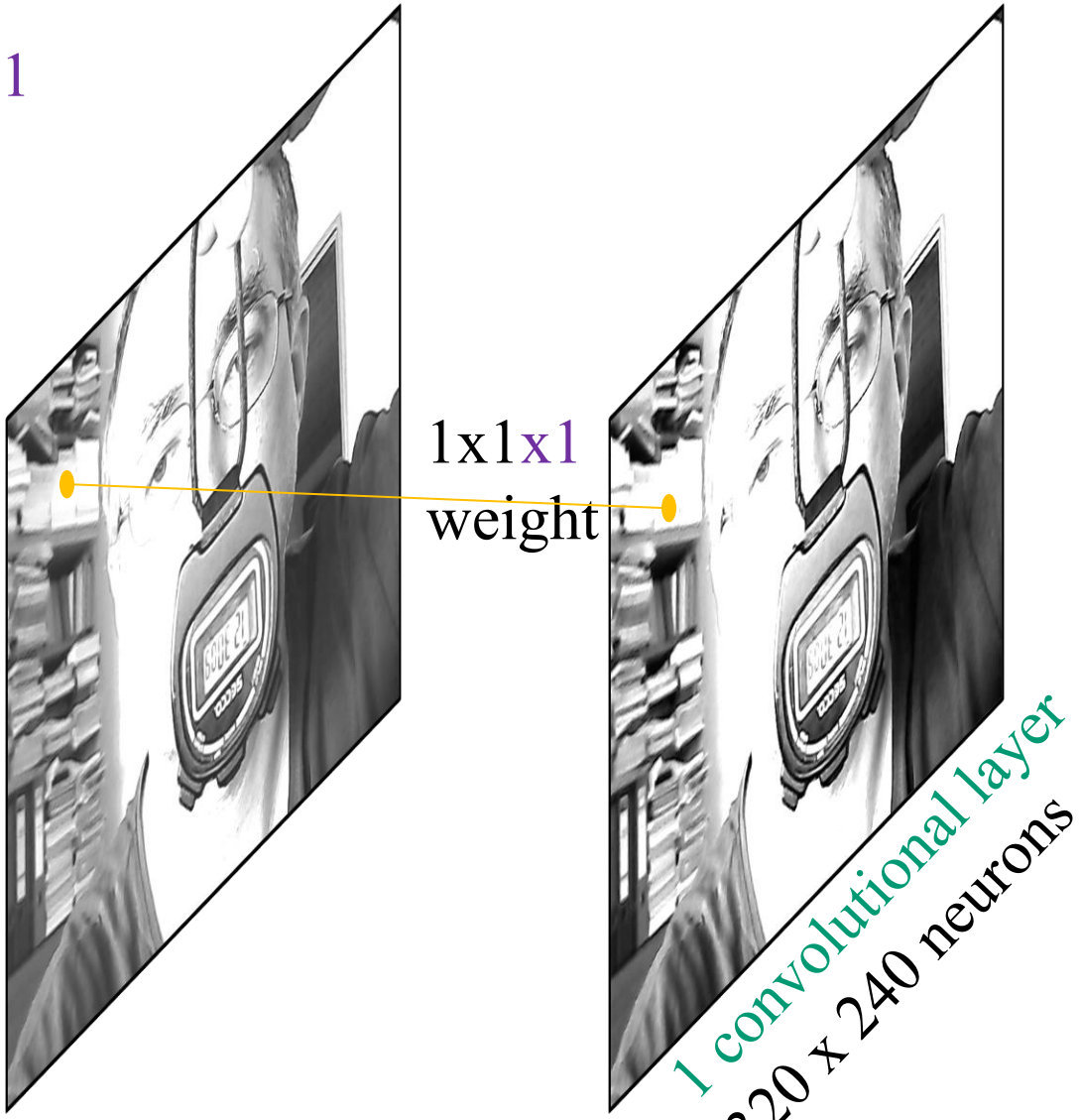
240x320x1

bias is brightness

Convolutional Layer

320x240x1
input
values

input
dimension
76800



neurons share
1 weight and
1 bias

the layer contains
76800 neurons
but has 2
parameters only

320x240x1
output
values

activation: linear

Training

sample input

1.5 contrast

weight

0.15 brightness

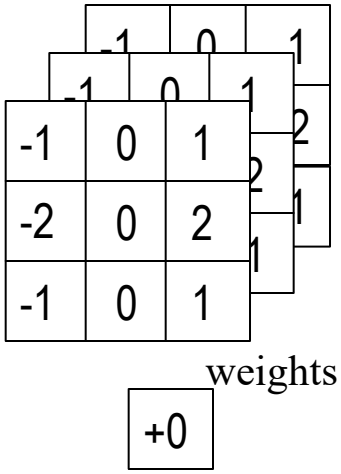
bias

sample output

Kernel

240x320x3

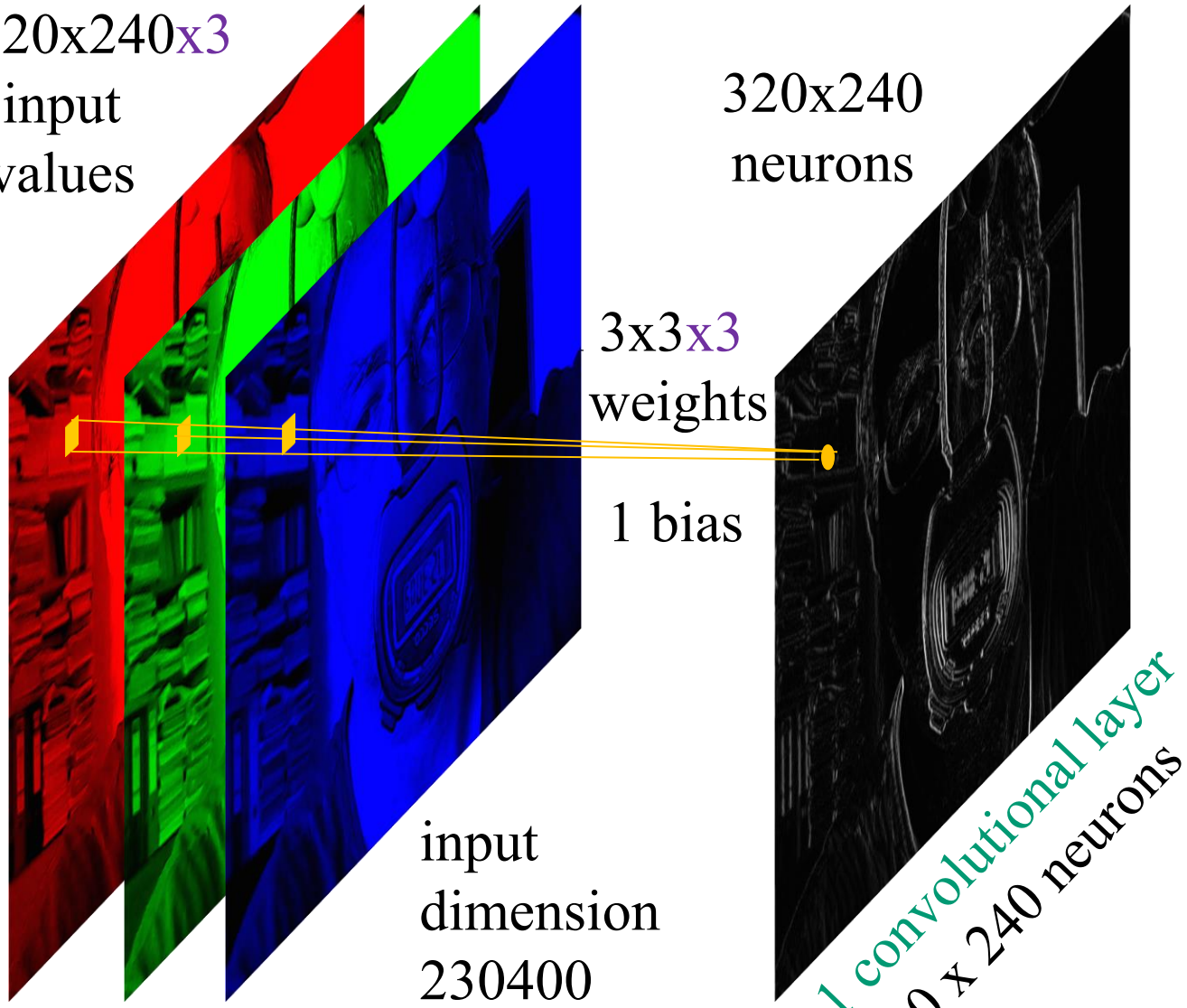
3x3x3



240x320x1

Sobel (vertical) kernel provides vertical edges

Convolutional Layer



320x240x3
input
values

320x240
neurons

neurons share
27 weights and
1 bias

3x3x3
weights

1 bias

the layer contains
76800 neurons
and 28 parameters

input
dimension
230400

1 convolutional layer
320 x 240 neurons

320x240x1
output
values

activation: linear

Training

sample input

sample output

Sobel kernel

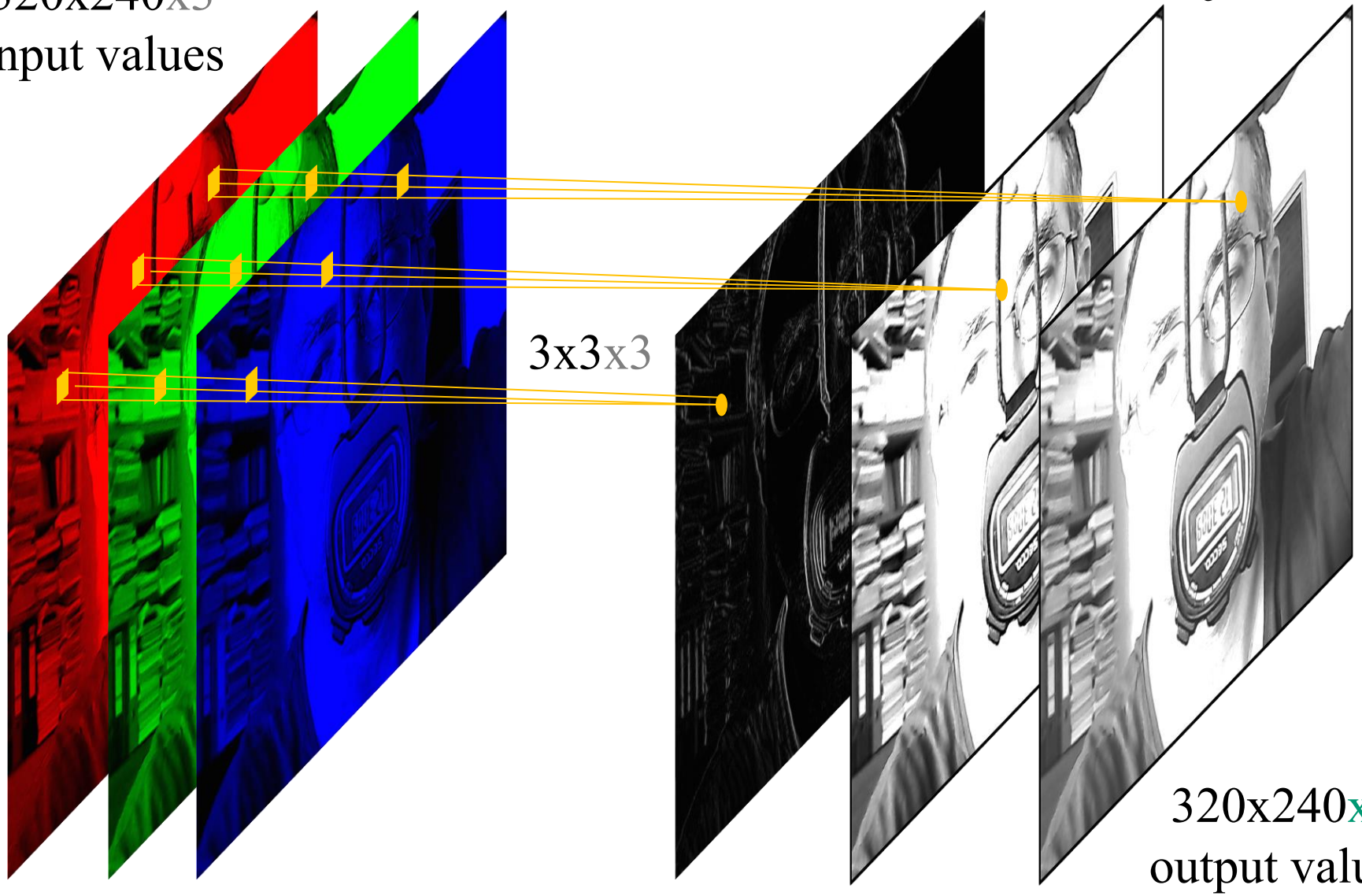
-1	0	1
-2	0	2
-1	0	1

weights

+0

bias

Block of convolutional layers



Typical activation function: ReLU or LeakeyReLU

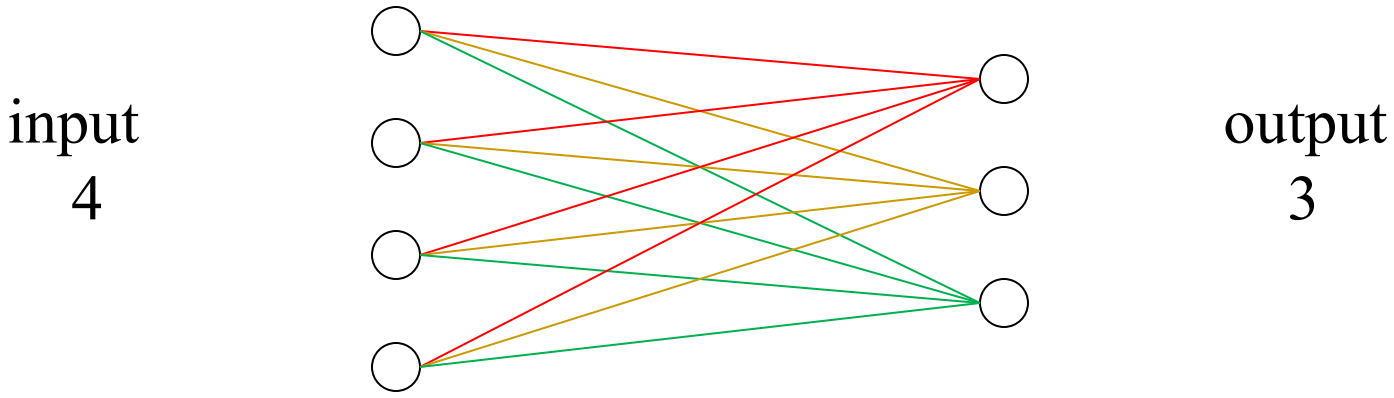
Weight initialization

- Default initialization in Pytorch is Kaiming distribtion:

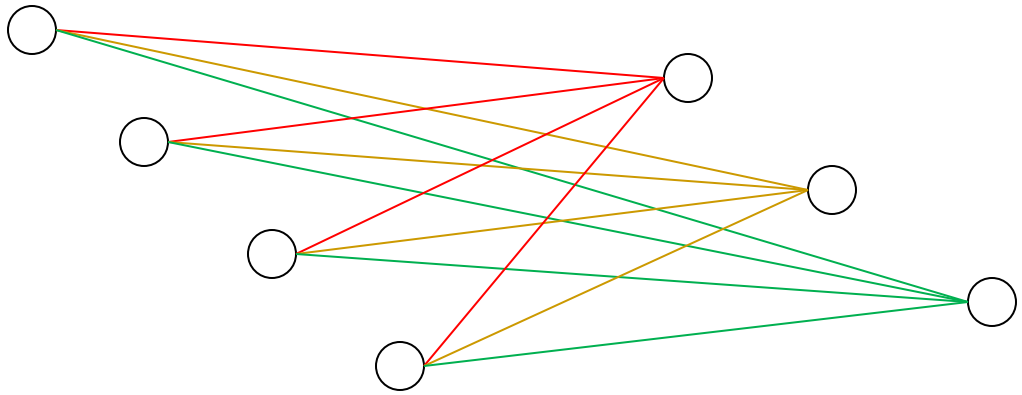
U[-x,x] *where* $x = \sqrt{\frac{6}{N}}$

where *N* is the number of inputs

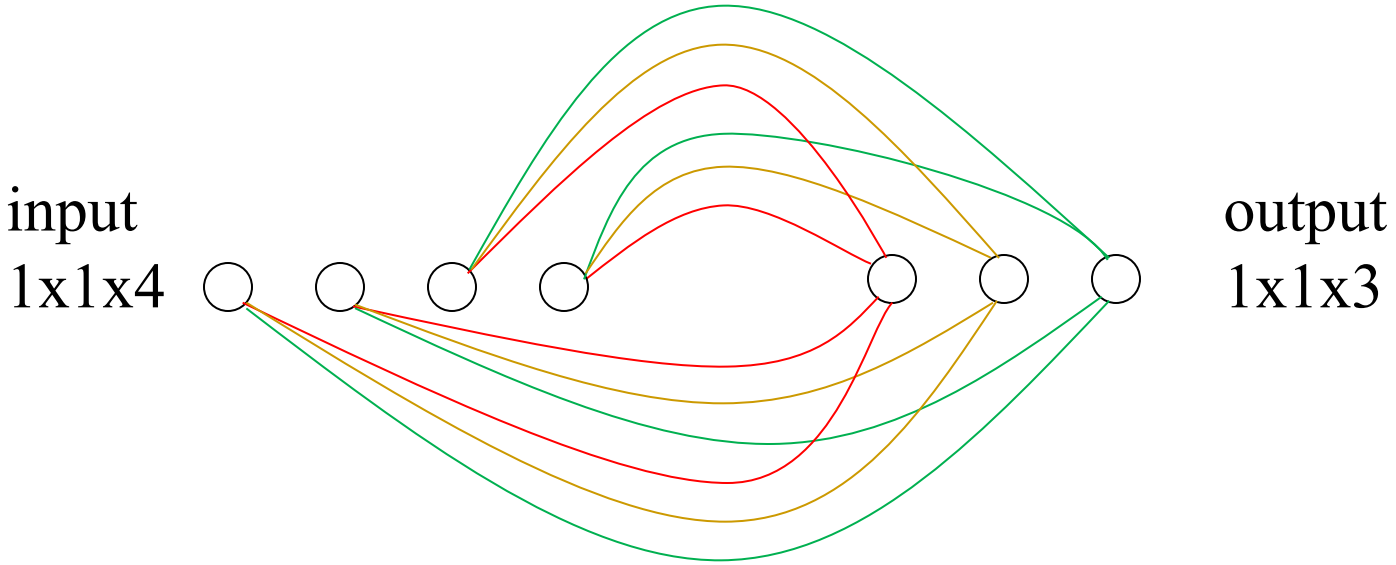
Fully – Connected layer (Linear) with N inputs and M neurons (N x M + M parameters) corresponds to a block of M convolutional layers connected to the input 1x1xN (resolution 1x1 and N channels) with kernel 1x1



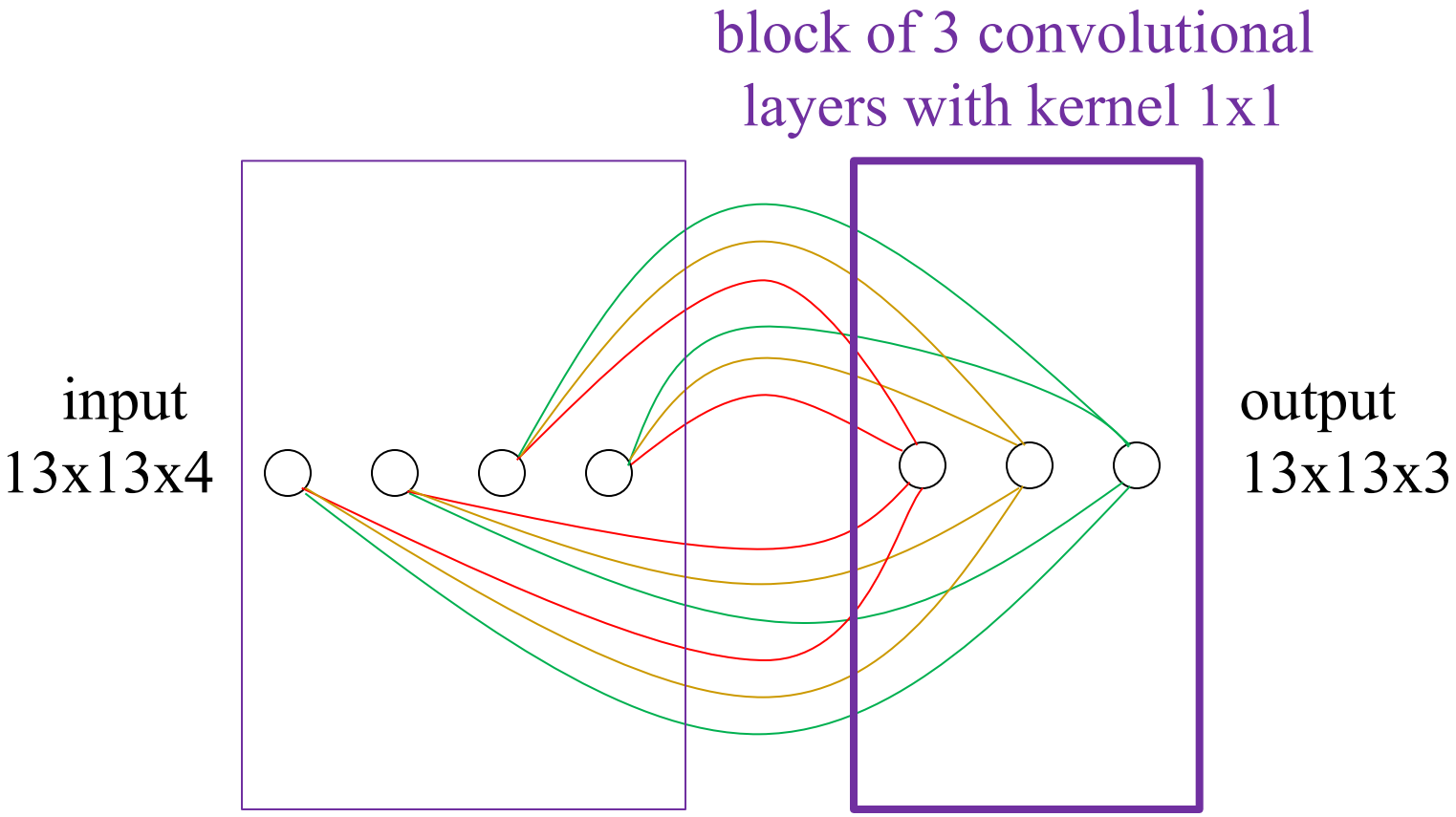
Fully – Connected layer (Linear) with N inputs and M neurons (N x M + M parameters) corresponds to a block of M convolutional layers connected to the input 1x1xN (resolution 1x1 and N channels) with kernel 1x1



Fully – Connected layer (Linear) with N inputs and M neurons (N x M + M parameters) corresponds to a block of M convolutional layers connected to the input 1x1xN (resolution 1x1 and N channels) with kernel 1x1



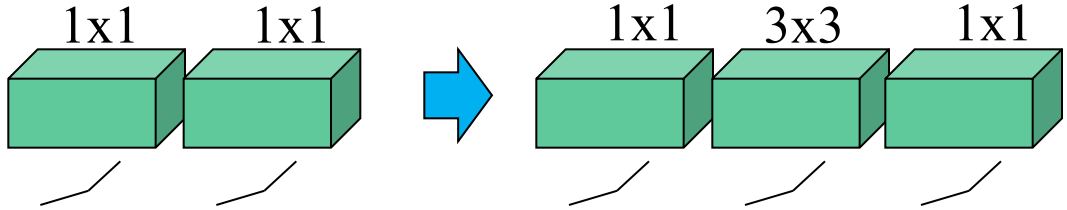
Now, when we input something with a larger resolution, such as 13x13x4, we run 169 perceptrons running in parallel, sharing weights and biases.



What exactly is CNN?

- What if we now use a kernel larger than 1x1, for example 3x3?

- This will ensure the exchange of information between neighboring perceptrons running in parallel.



Čo je vlastne CNN?

- CNN je teda sieť, kde paralelne púšťame rovnaké spolupracujúce perceptrony nad rôznymi miestami obrazu

- Tým premieňame obraz (mapa farieb) na mapu príznakov

Deep Neural Networks

- By laying down perceptrons, the number of layers in the network increases significantl
- That is why we started to call such networks "deep"

Fully-convolutional Networks

- DNNs where all Linear layers are replaced by Conv2D are fully-convolutional.

- These networks do not (unlike other DNNs) have a fixed input resolution.

- However, they are always trained to some fixed resolution.