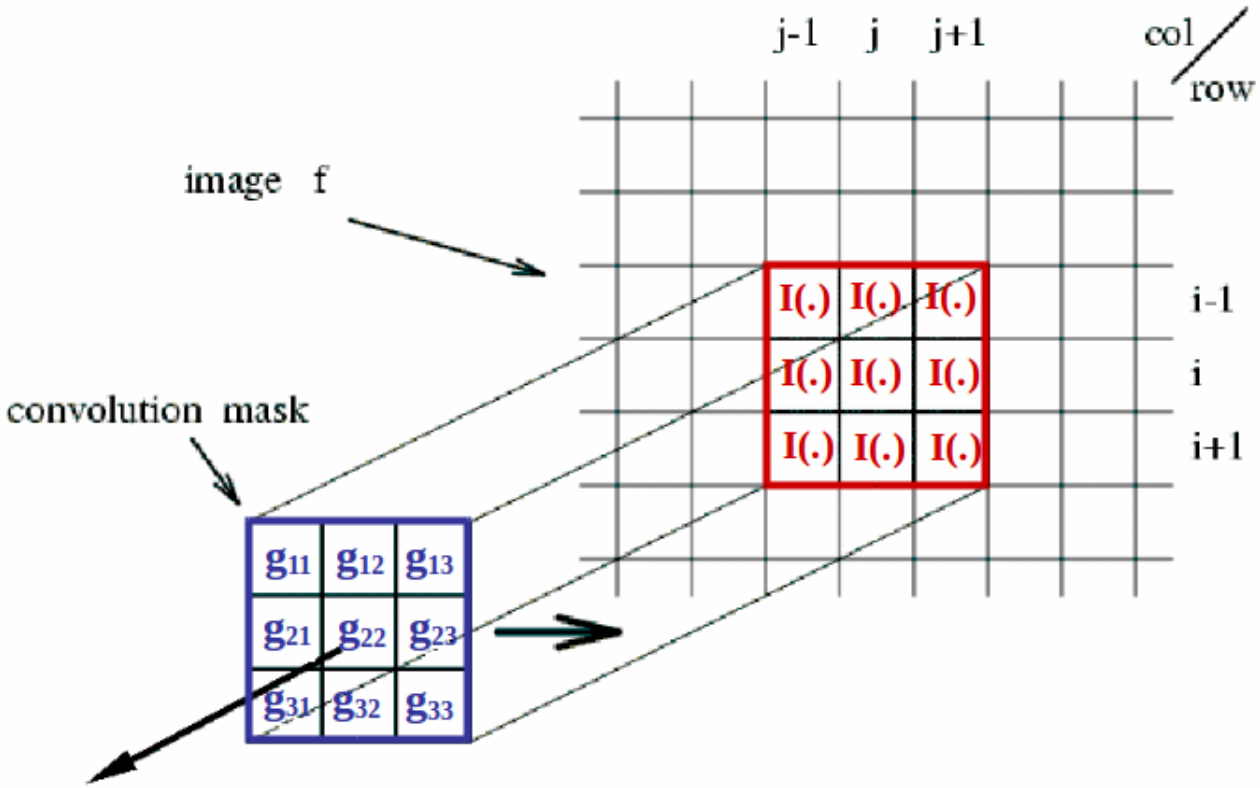
Computer Vision
Convolution neuron networks - lecture 6

Adam Szmigielski
aszmigie@pjwstk.edu.pl
materials: $ftp(public) : //aszmigie/WMAEnglish$

linear filter - convolution



$$f(i,j) = g_{11}\, I(i-1,j-1) + g_{12}\, I(i-1,j) + g_{13}\, I(i-1,j+1) + g_{21}\, I(i,j-1) + g_{22}\, I(i,j) + g_{23}\, I(i,j+1) + g_{31}\, I(i+1,j-1) + g_{32}\, I(i+1,j) + g_{33}\, I(i+1,j+1)$$

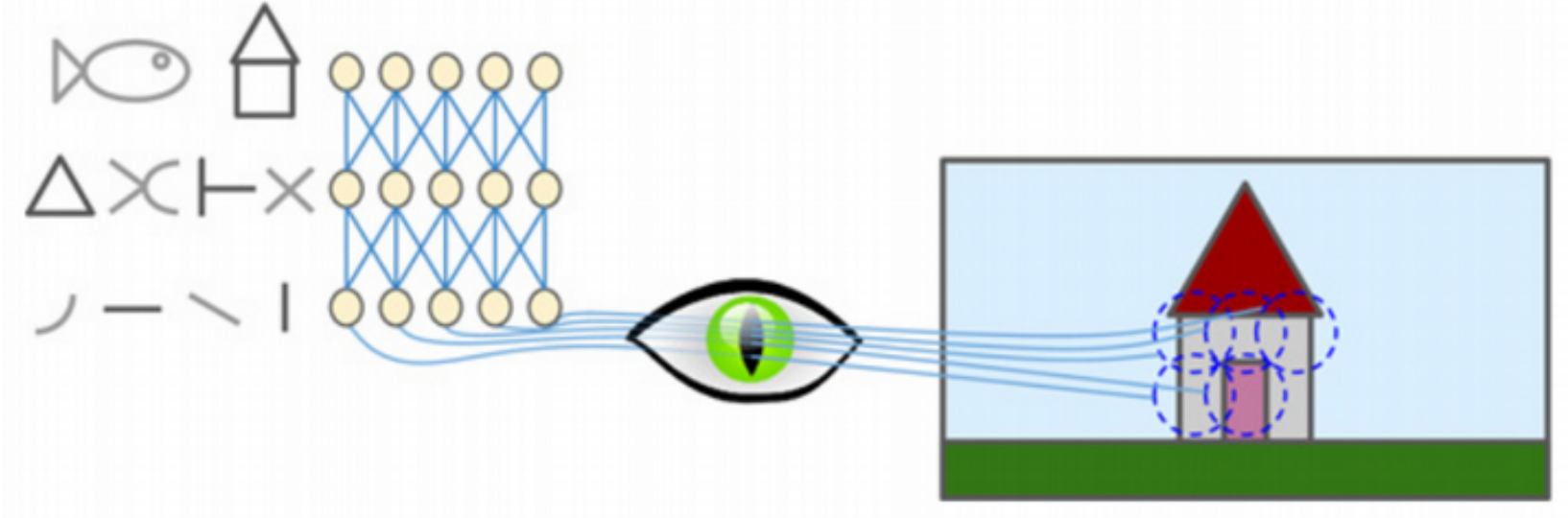
Linear filtration

- Linear is the simplest and most useful
- Turns every pixel into a linear neighbors combination.
- The function (method) for the linear combination is called the convolution kernel.

Cconvolutional Neural Networks - CNN

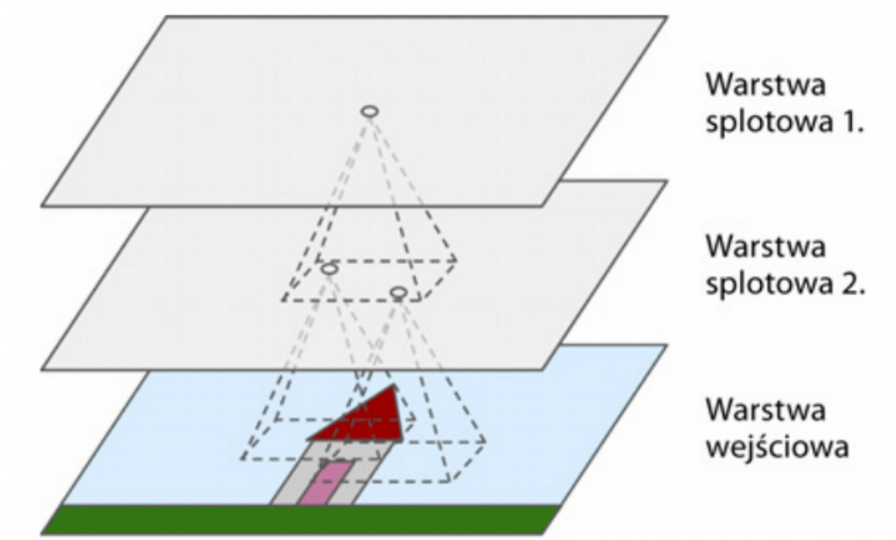
- They are the result of research on the visual cortex and have been used for image recognition since the 1980s.
- They form the basis of image search services, smart cars, automated movie classification systems, etc.
- They are also effective in other tasks, such as voice recognition or natural language processing.

Architecture of the visual cortex



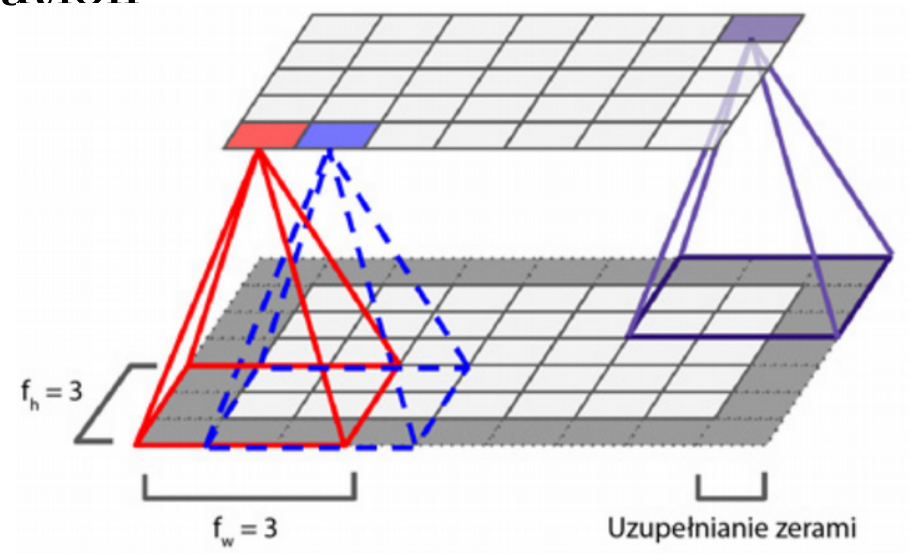
- Neurons form **local reception fields** - they react only to visual stimuli located in a specific region,
- Some neurons may have the same reception field, but they only react to images consisting of horizontal lines, and other vertical ones,
- Neurons that detect more complex shapes (which are a combination of more general patterns) are found on the output of neighboring neurons.

Convolution layer



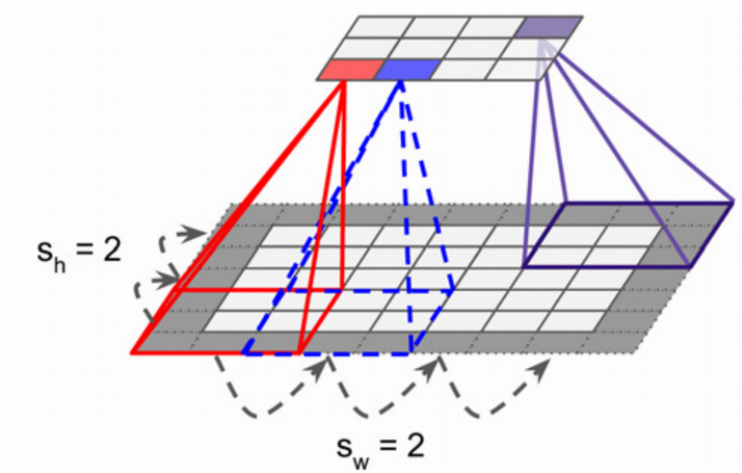
- Neurons in the first convolution layer are not connected to each pixel of the input image, but only to the pixels in their reception field
- In turn, each neuron in the second convolution layer connects only to neurons located in a small area of the first layer.

2D Convolution



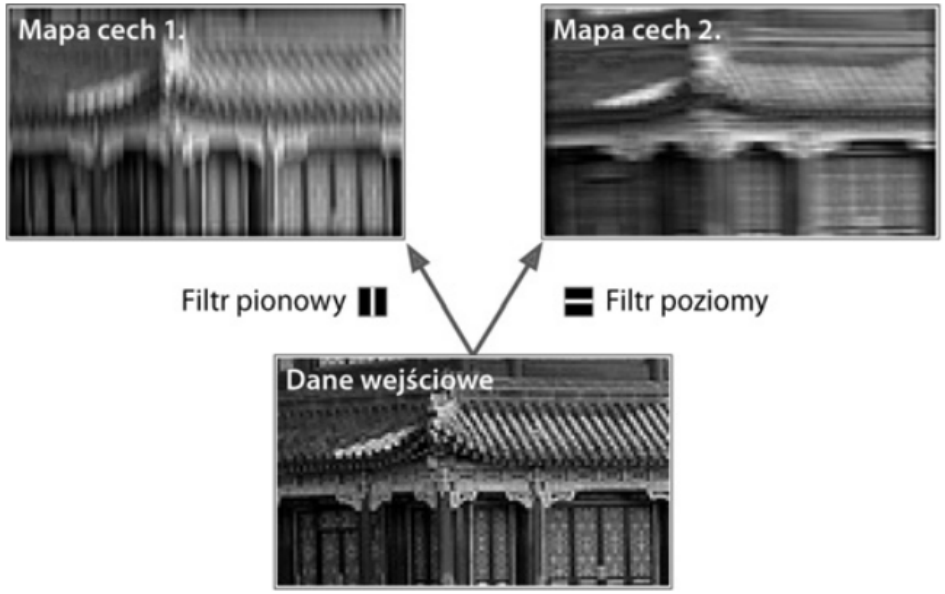
- Connected to the neuron outputs of the previous layer located in rows from i to $i + f_h - 1$ and columns from j to $j + f_w - 1$, where f_h and f_w mean, respectively, the height and width of the reception field
- In order to obtain the same dimensions of each layer, zeros (zero padding) are most often added around the inputs

Separation of reception fields



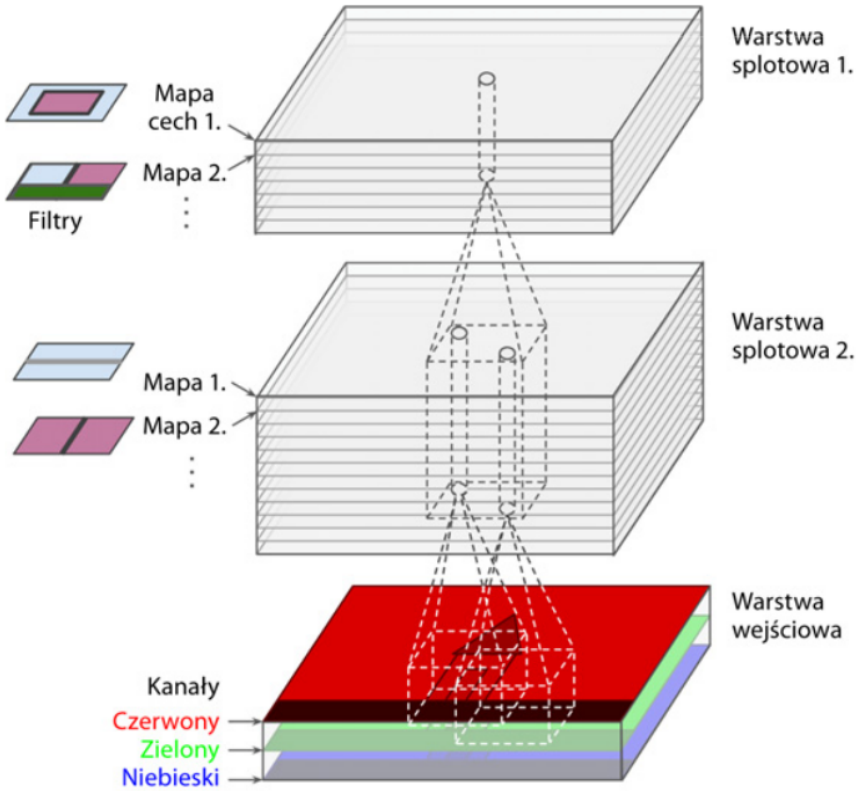
- The distance between two consecutive fields is bf step
- The input layer with dimensions 5×7 (plus zero padding) connects to the size layer 3×4 using the reception fields that are squares 3×3 and a step 2
- The neuron located in the i rows and the j column of the top layer connects to the output of the bottom layer neurons located in rows from $i \times s_h$ to $i \times s_h + f_h - 1$ and in columns from $j \times s_w$ to $j \times s_w + f_w - 1$, where s_h and s_w values of steps in columns and rows.

Filters



- The neuron scales can be presented as a small image with the size of the reception field,
- The first filter (a black square with a white vertical line) passing through its center (this is a 7×7 matrix filled with zeros in addition to the middle column that contains ones). The second filter (the middle line is arranged horizontally),
- A layer filled with neurons using the same filter gives us a feature map,

Stacks of features map



- For one feature map, all neurons have the same parameters,
- Other feature maps may have different parameter values,
- The convolution layer simultaneously applies various filters to the inputs, thanks to which it is able to detect many features.

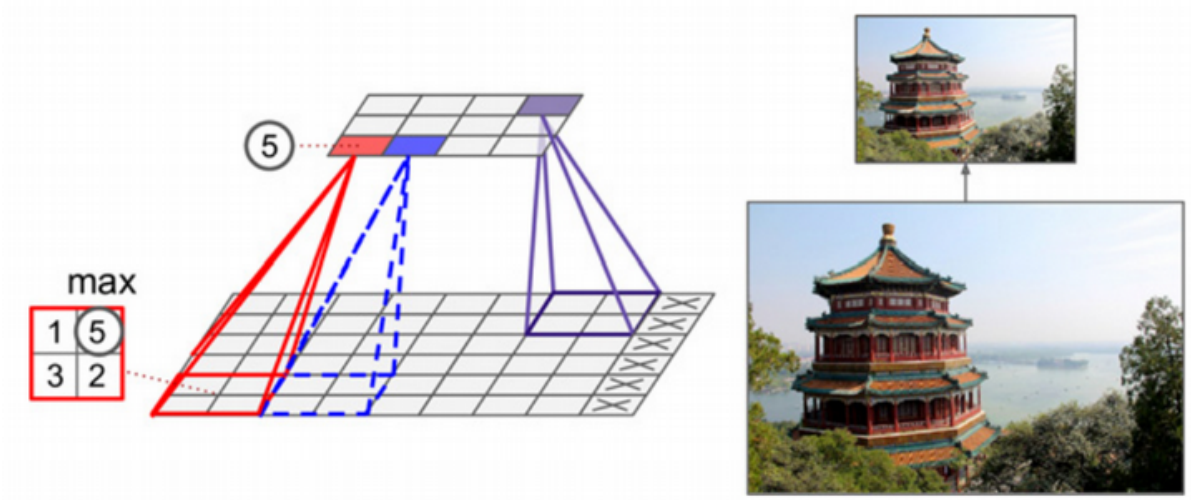
The values of the initial neuron in the convolution layer

$$z_{i,j,k} = b_k \sum_{u=0}^{f_h-1} \sum_{v=0}^{f_w-1} \sum_{k'=0}^{f_{n'}-1} x_{i',j',k'} \cdot w_{u,v,k',k} \tag{1}$$

where $i' = i \times s_h + u$ and $j' = j \times s_w + v$

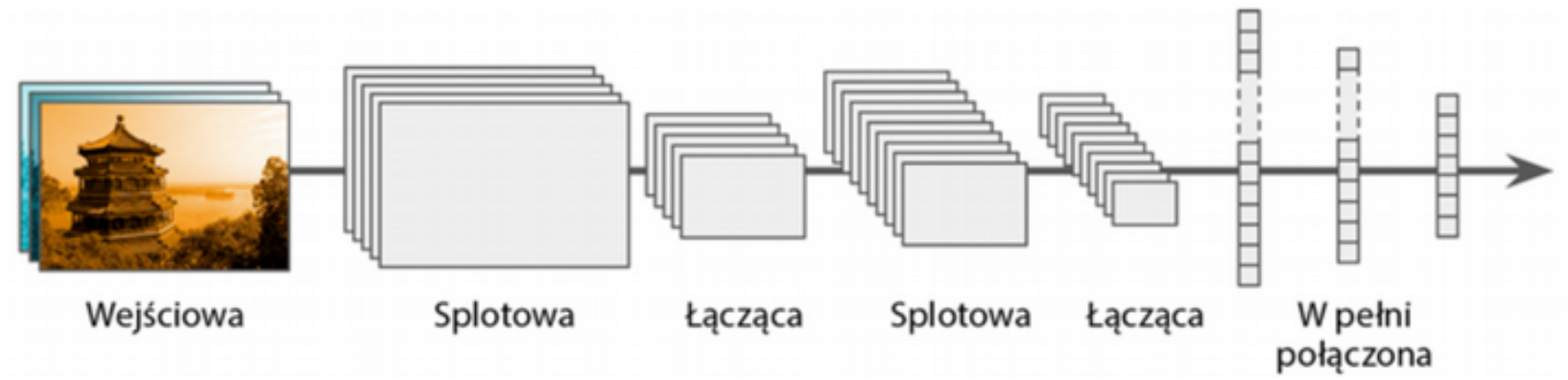
- $z_{i,j,k}$ - neuron output in row i, column j and map feature k convolution layer l,
- s_h and s_w - vertical and horizontal steps,
- f_h and f_w - height and width of the reception field,
- f_n - number of feature maps in the previous layer $(l-1)$,
- $x_{i,j,k}$ - neuron output in $l-1$, row i, column j, feature maps k
- b_k - bias for the features map k (in layer l)
- $w_{u,v,k',k}$ - weight in the features map k layers l and its entrance in the row u, column v and features map k

Connecting layer



- Connection kernel: 2×2, no zeroing
- Does not contain any weights - collects input data using an aggregation function (e.g. maximizing or averaging),
- The purpose of the convolution layer is to subscribe (*subsample*) the input image for optimization purposes,
- Neuron from the connecting layer connects to the outputs of a given number of neurons of the previous layer, in the area of the reception field.

Architecture of convolutional neural networks



- Typical CNN architectures consist of several layers of a convolution, connecting layer, after it several CONVOLUTION layers (+ ReLU), another layer of joining, etc.
- The image gradually decreases, going through subsequent layers of the network, but at the same time its depth increases
- A classic neural network is placed at the top of the network, containing several fully connected layers (+ ReLU), and the last one calculates prediction (eg. softmax logistic regression).

```
# convolution network w keras
from keras import layers
from keras import models
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
model.summary()
```

- The network adopts a tensor with a shape determined by the height of the image, its width and image channels, e.g. (28, 28, 1)
- At the output of each layer *Conv2D* and *MaxPooling2D*, a three-dimensional tensor with shape (height, width, channels) appears.
- As the network sinks, the height and width tend to take smaller values.
- The last step is a dense network with a classifier (tensor (3, 3, 64)).

The result of creating the model

```
Instructions for updating:
Colocations handled automatically by placer.
```

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d_1 (MaxPooling2	(None, 13, 13, 32)	0
conv2d_2 (Conv2D)	(None, 11, 11, 64)	18496
max_pooling2d_2 (MaxPooling2	(None, 5, 5, 64)	0
conv2d_3 (Conv2D)	(None, 3, 3, 64)	36928
flatten_1 (Flatten)	(None, 576)	0
dense_1 (Dense)	(None, 64)	36928
dense_2 (Dense)	(None, 10)	650

```
Total params: 93,322
Trainable params: 93,322
Non-trainable params: 0
```

Starting and training the model

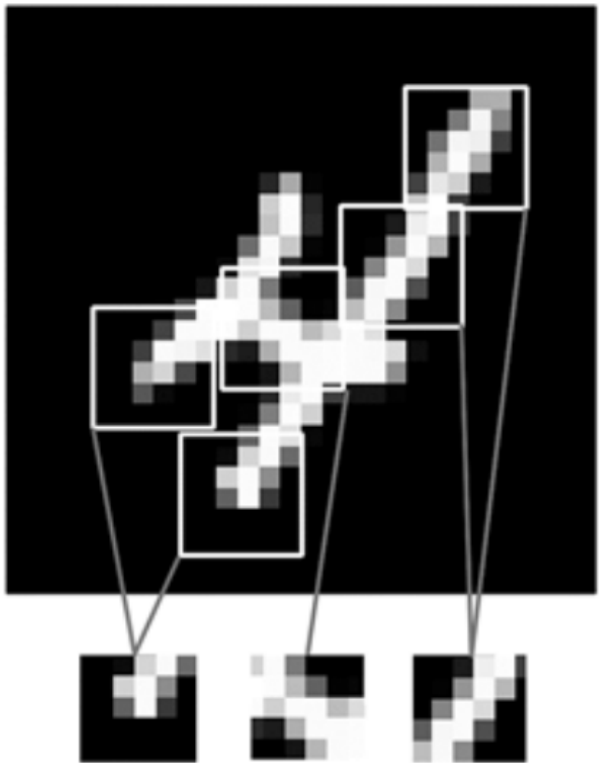
```
from keras.datasets import mnist
from keras.utils import to_categorical
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
train_images = train_images.reshape((60000, 28, 28, 1))
train_images = train_images.astype('float32') / 255
test_images = test_images.reshape((10000, 28, 28, 1))
test_images = test_images.astype('float32') / 255
train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
model.compile(optimizer='rmsprop', loss='categorical_crossentropy',
metrics=['accuracy'])
model.fit(train_images, train_labels, epochs=5, batch_size=64)

test_loss, test_acc = model.evaluate(test_images, test_labels)
print('test_acc =', test_acc)
```

The result of the program:

```
.................................
10000/10000 [==============================] - 1s 72us/step
test_acc =  0.9919
```

Convolutional network

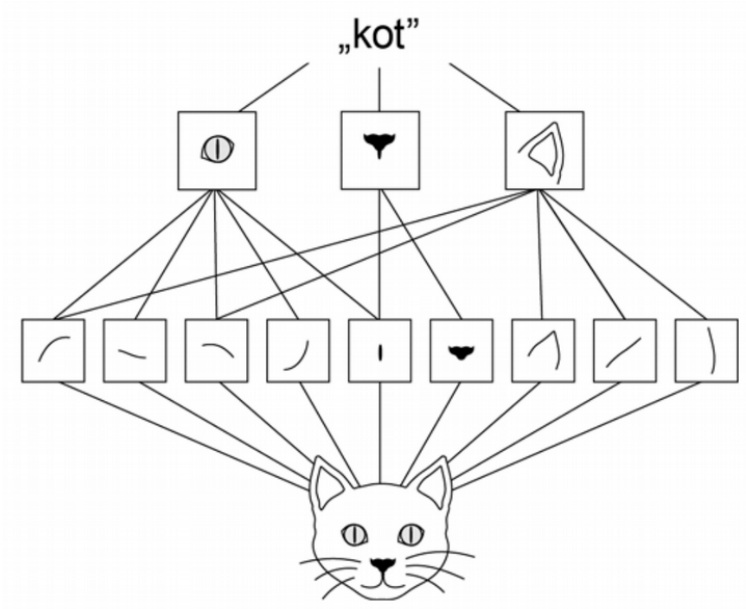


- Images can be broken down into local patterns, e.g. edges and textures,
- The basic difference between the dense connections layer and the convolutional network is that the Dense layers learn the parameters of global parameters in their input spaces,

Properties of convolutional networks

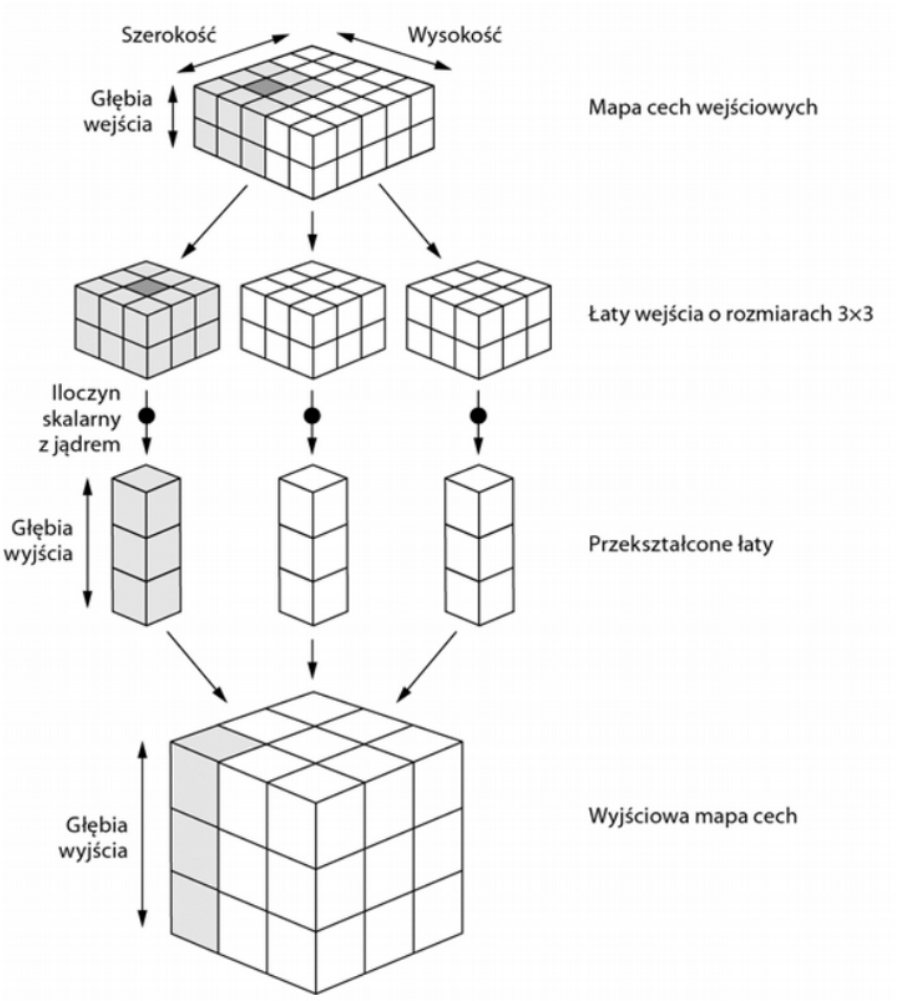
- **Patterns recognized by the network are independent of shifting.** - the image processing algorithm must not bind the shape to its position in the frame.
- **Convolutionary networks can learn spatial hierarchy of patterns**. The first layer of the convolution network is taught small local patterns (eg. edges), the second layer of this network will learn the larger structures consisting of elements recognized by the first layer, etc.

Spatial hierarchy of elements



- They work on three-dimensional tensors known as feature maps, containing two spatial axes defining height and width,
- The third axis is the depth axis, also called the channel axis,
- Extract patches from input features and perform the same operation on all patches to create the output feature map.

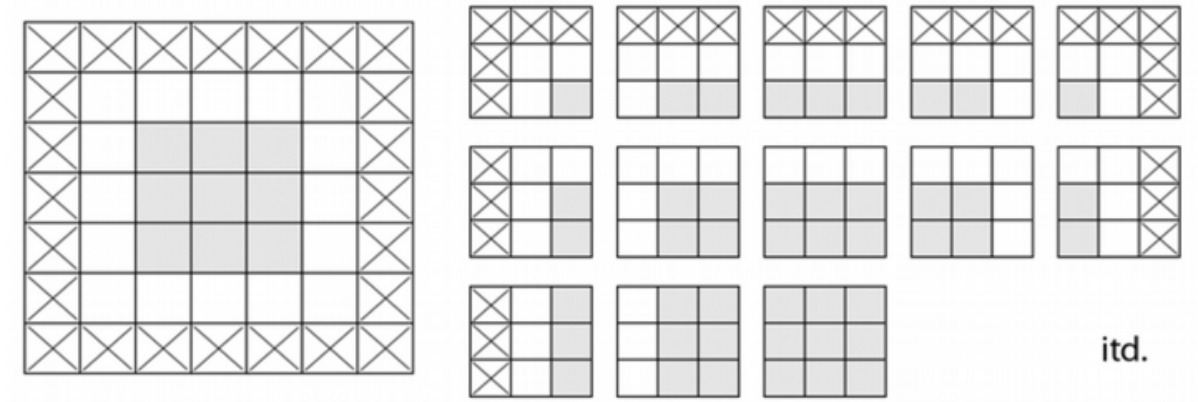
The operation of convolution



The operation of convolution

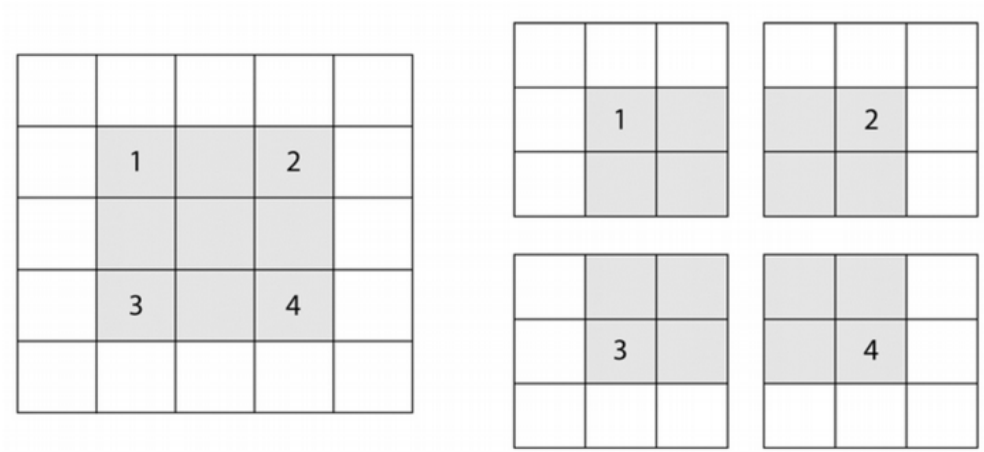
- It involves moving windows with dimensions 3×3 or 5×5 on a three-dimensional map of features.
- Extracts the three-dimensional patch of the surrounding features *(window height, window width, entry depth)*).
- Each such three-dimensional patch is then transformed (through a tensor product with the convolution kernel) to obtain a one-dimensional vector with the shape *(depth of the output)*.
- All these vectors are then rebuilt spatially to create a three-dimensional output map with the shape *(height, width, depth of the output)*.

Border effects and complement



- An example of a feature map with dimensions 5×5 - it contains only 9 fields on which you can set the center of the window with dimensions 3×3
- The feature map will be 3×3.
- If we wanted to get the output map of features with the same dimensions as the input map, then we would have to use the padding technique.
- It's about adding the right number of rows and columns on each side of the input feature map

Steps of the convolution process



Convolution patches 3×3 at the 2×2 step (without compliement).

- The distance between two windows is a convolution parameter - **step** (stride).
- By default, it takes 1, but you can assign a higher value to it, which will lead to the so-called convolutions.
- A step with a value of 2 means that the width and height of the feature map are scaled (they are reduced twice).
- Instead of steps, the **max-pooling** scaling operation is typically used.

Max-pooling operation

- This operation aggressively reduces the resolution of feature maps,
- The *max-pooling* scaling operation is performed using the extraction windows transforming feature maps.
- These windows return the maximum values of each channel - tensor operation max (maximum),
- The *max-pooling* operation is usually performed using windows 2×2 at a step equal to 2 (this is to reduce the feature map twice).
- Convulsions are performed using windows 3×3 with a step parameter equal to 1