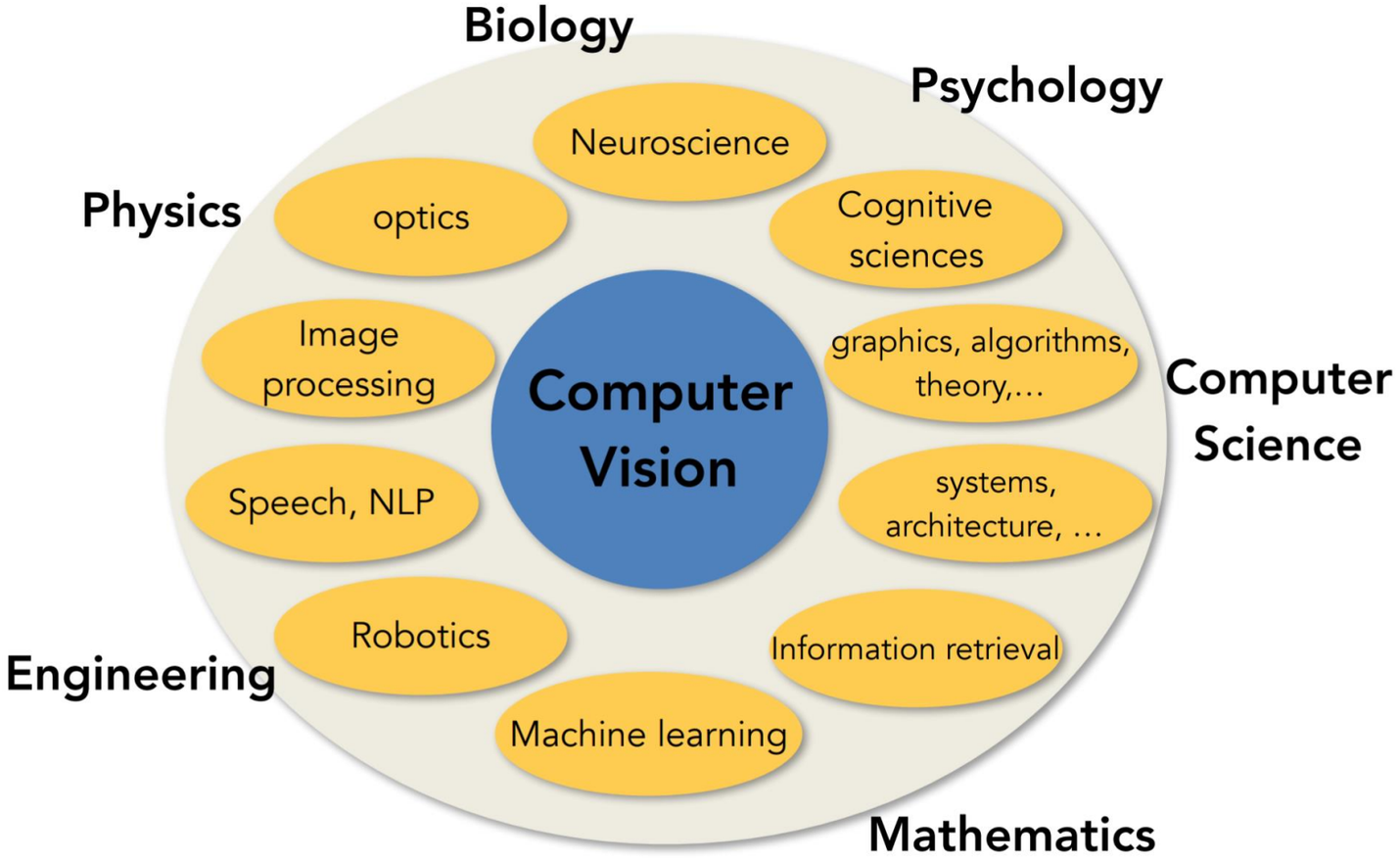
UCZENIE MASZYNOWE

KONWOLUCYJNE SIECI NEURONOWE

Prof. dr hab. inż. Grzegorz Dudek
Wydział Elektryczny
Politechnika Częstochowska

Informacje wstępne

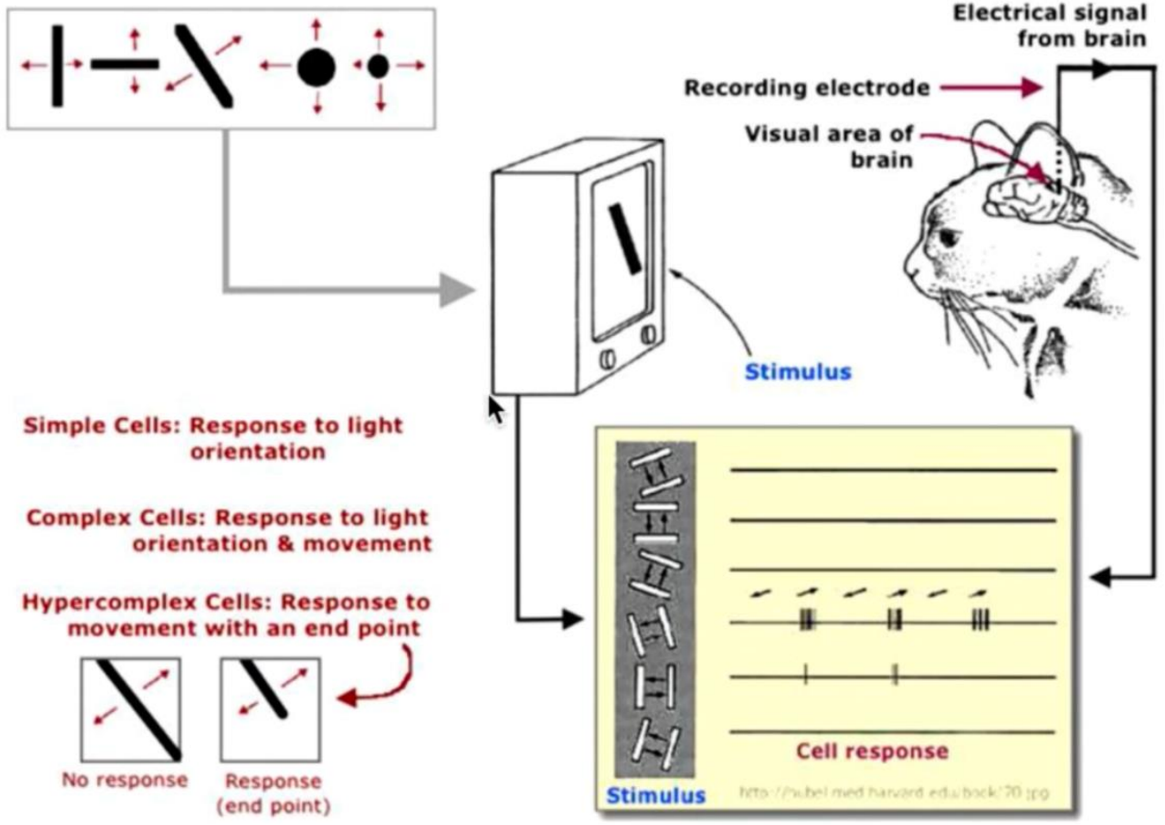
Pole zastosowań wizji komputerowej



Informacje wstępne

Pionierskie prace Hubela i Wiesela

Hubel i Wiesel otrzymali Nagrodę Nobla za prace nt. przetwarzania sygnałów z oka przez mózg - wygenerowania detektorów krawędzi, detektorów ruchu, detektorów głębi stereoskopowej i detektorów koloru.

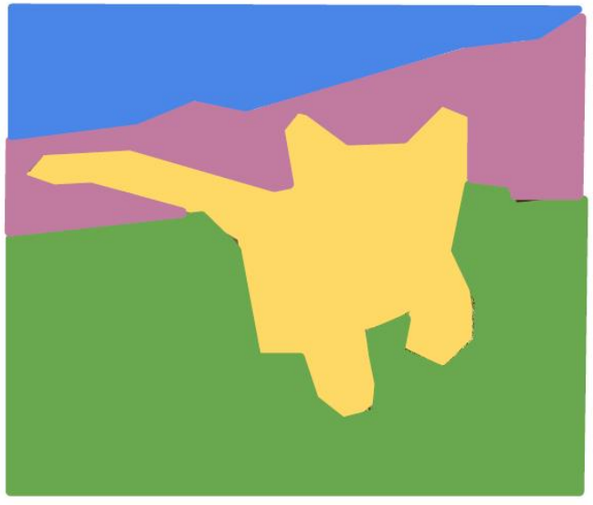


W jednym z eksperymentów, przeprowadzonym w 1959 roku, wprowadzili oni mikroelektrodę do pierwotnej kory wzrokowej znieczulonego kota. Następnie wyświetlali na ekranie przed kotem wzory jasnych i ciemnych linii. Stwierdzili, że niektóre neurony uruchamiały się szybko, gdy wyświetlano linie pod jednym kątem, podczas gdy inne reagowały najlepiej na inny kąt. Niektóre z tych neuronów reagowały inaczej na jasne wzory niż na ciemne. Hubel i Wiesel nazwali te neurony "komórkami prostymi".

Jeszcze inne neurony, które nazwali "komórkami złożonymi", wykrywały krawędzie niezależnie od tego, jak były umiejscowione w polu widzenia, i potrafiły wykrywać ruch w pewnych kierunkach. Badania te pokazały, jak system wizualny konstruuje złożone obrazy informacji wizualnej z prostych cech bodźca.

Problemy wizji komputerowej

Semantic Segmentation



GRASS, CAT, TREE, SKY

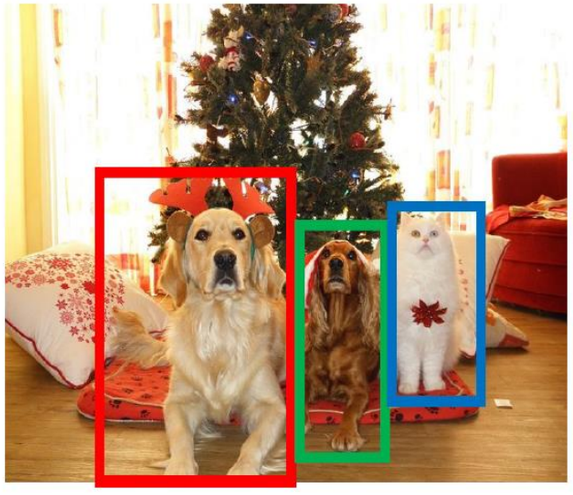
No objects, just pixels

Classification + Localization

CAT

Single Object

Object Detection



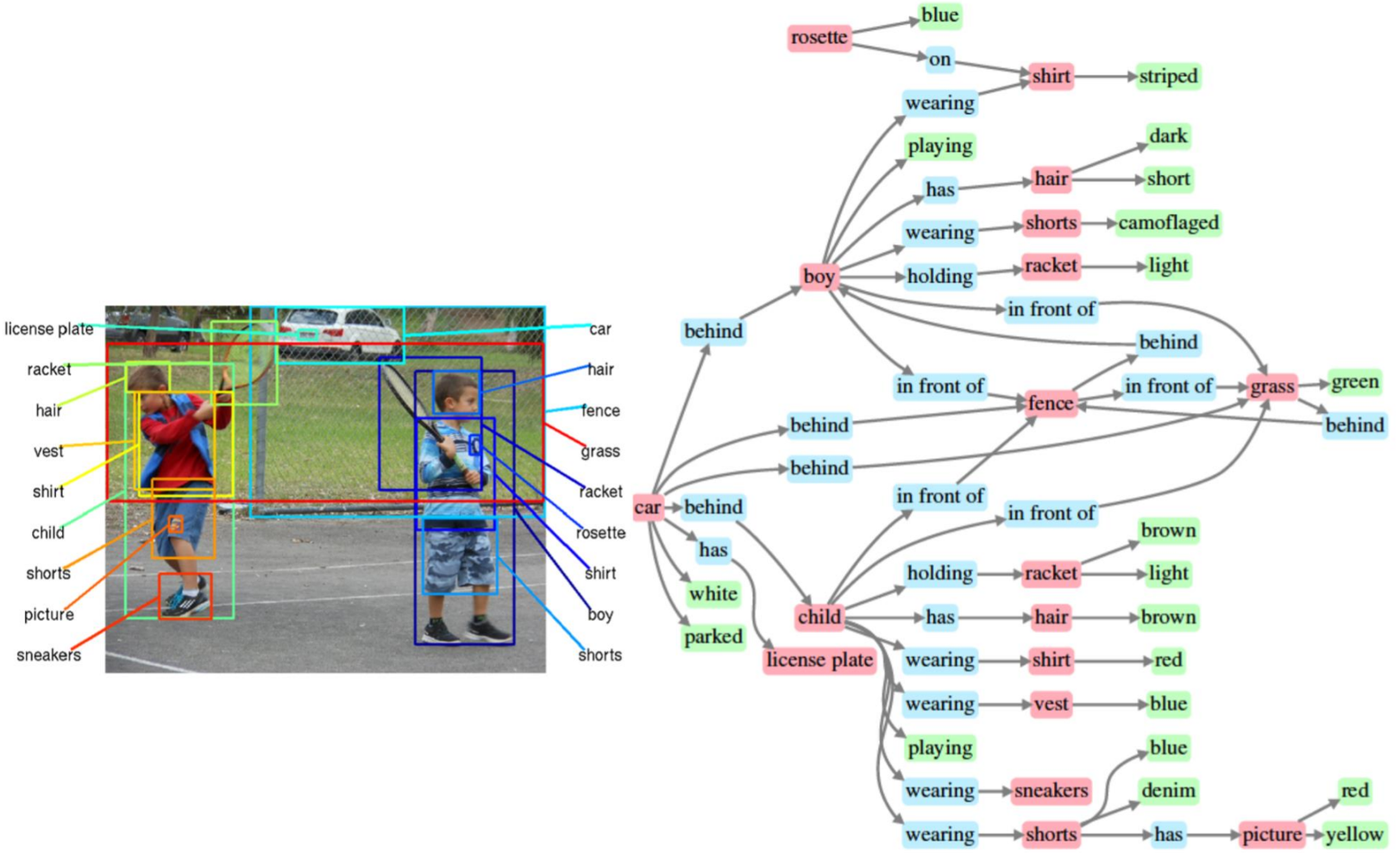
DOG, DOG, CAT

Instance Segmentation

DOG, DOG, CAT

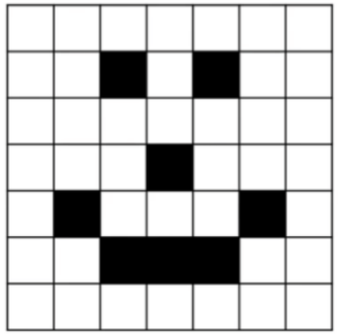
Multiple Object

Problemy wizji komputerowej

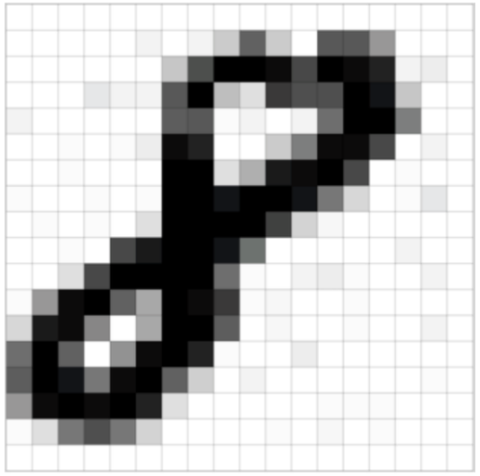


Johnson *et al.*, "Image Retrieval using Scene Graphs", CVPR 2015

Jak komputer „widzi” obraz



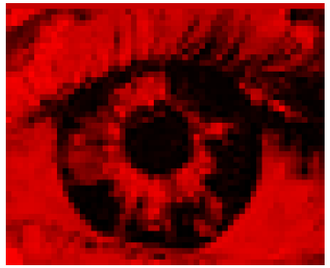
0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

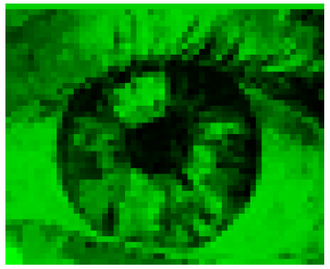


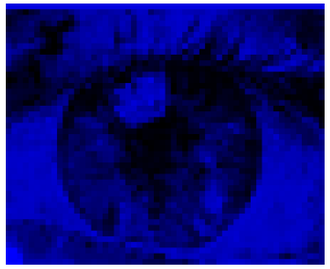
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	12	0	11	39	137	37	0	152	147	84	0	0	0
0	0	1	0	0	0	41	160	250	255	235	162	255	238	206	11	13	0
0	0	0	16	9	9	150	251	45	21	184	159	154	255	233	40	0	0
10	0	0	0	0	0	145	146	3	10	0	11	124	253	255	107	0	0
0	0	3	0	4	15	236	216	0	0	38	109	247	240	169	0	11	0
1	0	2	0	0	0	253	253	23	62	224	241	255	164	0	5	0	0
6	0	0	4	0	3	252	250	228	255	255	234	112	28	0	2	17	0
0	2	1	4	0	21	255	253	251	255	172	31	8	0	1	0	0	0
0	0	4	0	163	225	251	255	229	120	0	0	0	0	0	11	0	0
0	0	21	162	255	255	254	255	126	6	0	10	14	6	0	0	9	0
3	79	242	255	141	66	255	245	189	7	8	0	0	5	0	0	0	0
26	221	237	98	0	67	251	255	144	0	8	0	0	7	0	0	11	0
125	255	141	0	87	244	255	208	3	0	0	13	0	1	0	1	0	0
145	248	228	116	235	255	141	34	0	11	0	1	0	0	0	1	3	0
85	237	253	246	255	210	21	1	0	1	0	0	6	2	4	0	0	0
6	23	112	157	114	32	0	0	0	0	2	0	8	0	7	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

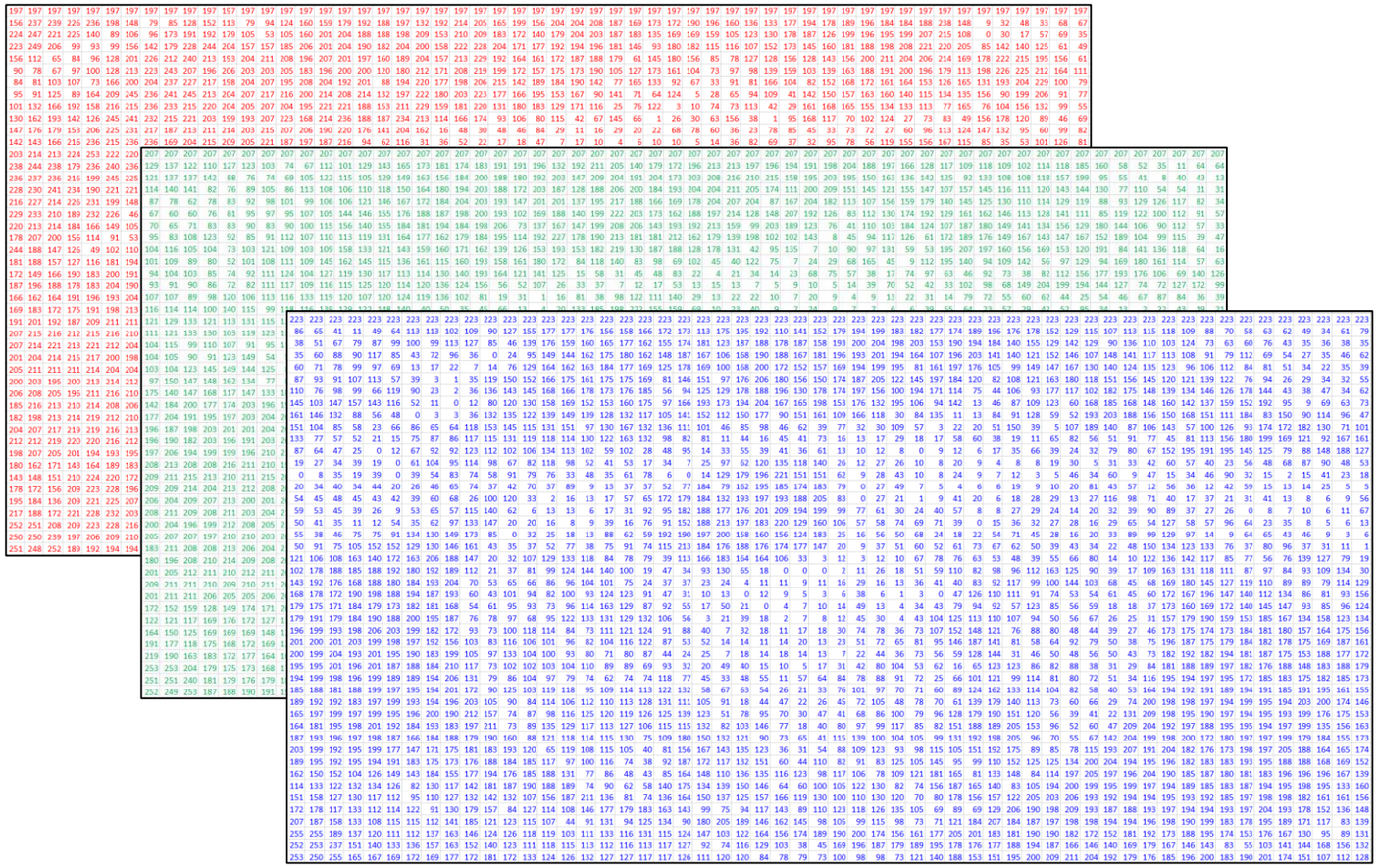
Jak komputer „widzi” obraz

Obraz kolorowy ma trzy kanały – czerwony, zielony i niebieski. Reprezentują go trzy macierze (po jednej dla każdego koloru), z których każda ma wartości pikseli w zakresie od 0 do 255.







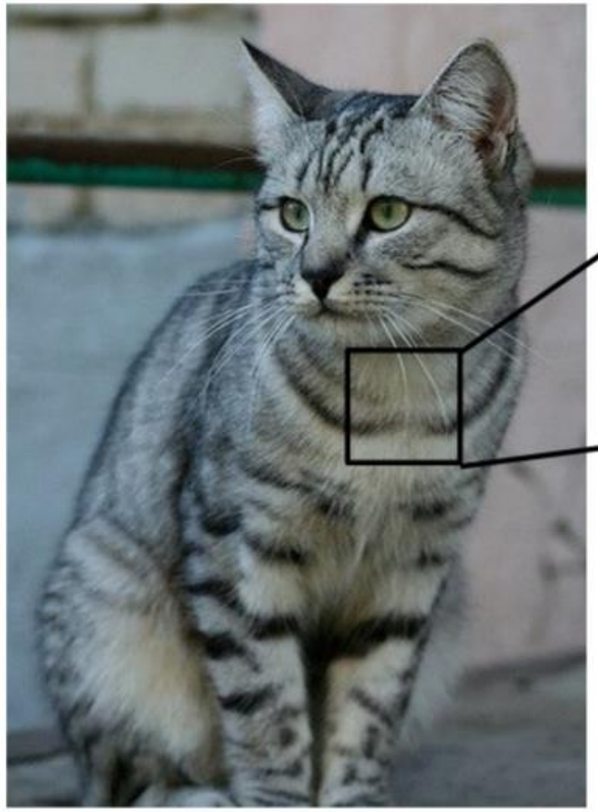


Klasyfikacja obrazów

(assume given set of discrete labels)
{dog, cat, truck, plane, ...}

→ cat

The Problem: Semantic Gap




```
[[105 112 108 111 104  99 106  99  96 103 112 119 104  97  93  87]
 [ 91  98 102 106 104  79  98 103  99 105 123 136 110 105  94  85]
 [ 76  85  90 105 128 105  87  96  95  99 115 112 106 103  99  85]
 [ 99  81  81  93 120 131 127 100  95  98 102  99  96  93 101  94]
 [106  91  61  64  69  91  88  85 101 107 109  98  75  84  96  95]
 [114 108  85  55  55  69  64  54  64  87 112 129  98  74  84  91]
 [133 137 147 103  65  81  80  65  52  54  74  84 102  93  85  82]
 [128 137 144 140 109  95  86  70  62  65  63  63  60  73  86 101]
 [125 133 148 137 119 121 117  94  65  79  80  65  54  64  72  98]
 [127 125 131 147 133 127 126 131 111  96  89  75  61  64  72  84]
 [115 114 109 123 150 148 131 118 113 109 100  92  74  65  72  78]
 [ 89  93  90  97 108 147 131 118 113 114 113 109 106  95  77  80]
 [ 63  77  86  81  77  79 102 123 117 115 117 125 125 130 115  87]
 [ 62  65  82  89  78  71  80 101 124 126 119 101 107 114 131 119]
 [ 63  65  75  88  89  71  62  81 120 138 135 105  81  98 110 118]
 [ 87  65  71  87 106  95  69  45  76 130 126 107  92  94 105 112]
 [118  97  82  86 117 123 116  66  41  51  95  93  89  95 102 107]
 [164 146 112  80  82 120 124 104  76  48  45  66  88 101 102 109]
 [157 170 157 120  93  86 114 132 112  97  69  55  70  82  99  94]
 [130 128 134 161 139 100 109 118 121 134 114  87  65  53  69  86]
 [128 112  96 117 150 144 120 115 104 107 102  93  87  81  72  79]
 [123 107  96  86  83 112 153 149 122 109 104  75  80 107 112  99]
 [122 121 102  80  82  86  94 117 145 148 153 102  58  78  92 107]
 [122 164 148 103  71  56  78  83  93 103 119 139 102  61  69  84]]
```

What the computer sees

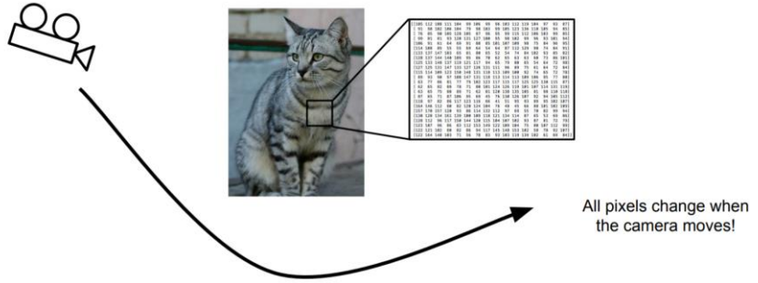
An image is just a big grid of numbers between [0, 255]:

e.g. 800 x 600 x 3
(3 channels RGB)

Klasyfikacja obrazów

Wyzwania

Challenges: Viewpoint variation



Challenges: Illumination

Challenges: Deformation

Challenges: Occlusion

Challenges: Background Clutter

Challenges: Intraclass variation

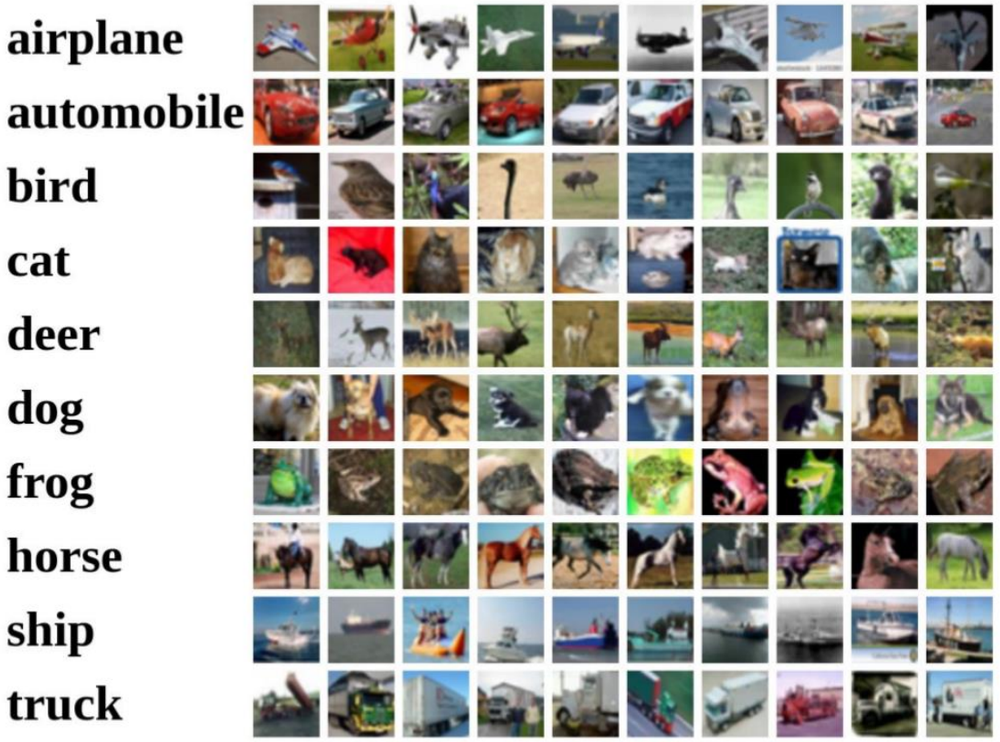
Klasyfikacja obrazów

Procedura

1. Zbierz zbiór danych - obrazów i ich etykiet (zbiór uczący / treningowy)
2. Użyj uczenia maszynowego, aby wyszkolić klasyfikator
3. Oceń klasyfikator na nowych obrazach

Example Dataset: **CIFAR10**

10 classes
50,000 training images
10,000 testing images



Klasyfikacja obrazów

Obraz może być reprezentowany przez wektor z wartościami pikseli:

$$\mathbf{x} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \cdots & x_{m,n} \end{bmatrix} \quad \text{lub} \quad \mathbf{x} = [x_{1,1}, x_{1,2}, \dots, x_{m,n}]$$

Etykieta może mieć postać:

$$y = "kot" \quad \text{lub} \quad y = 1$$

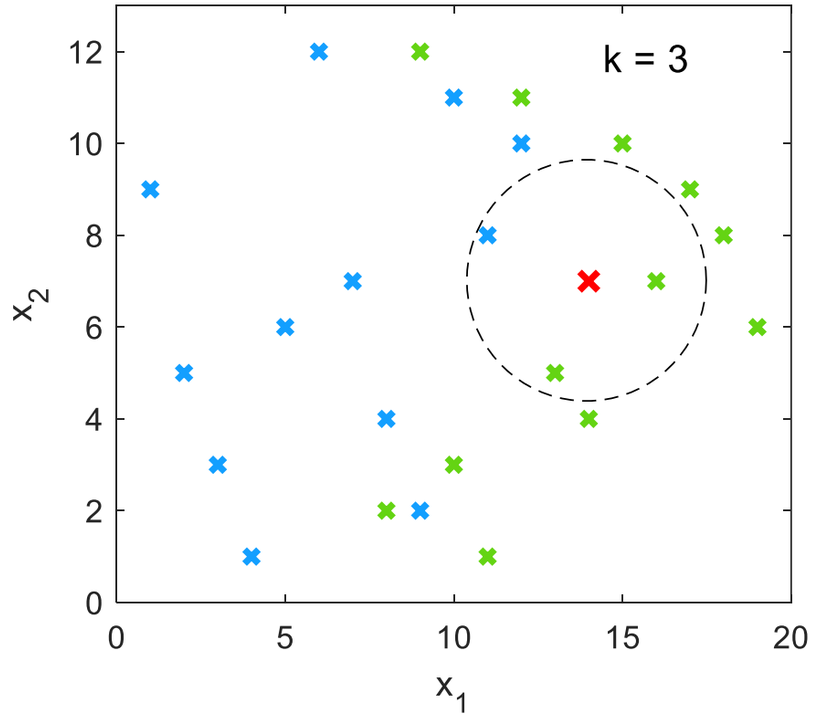
Zbiór uczący:

$$\Phi = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$$

Klasyfikacja obrazów

Metoda najbliższych sąsiadów

W metodzie tej obraz zalicza się do tej klasy, do której należy większość z jego k najbliższych sąsiadów. Klasyfikacja odbywa się poprzez wyznaczenie odległości lub podobieństwa pomiędzy klasyfikowanym obrazem x*, a każdym obrazem ze zbioru trenującego. Klasyfikowanemu obrazowi przypisywana jest klasa reprezentowana przez największą liczbę obrazów spośród k najbliższych sąsiadów.



Klasyfikacja obrazów

Metoda najbliższych sąsiadów

Metoda najbliższych sąsiadów

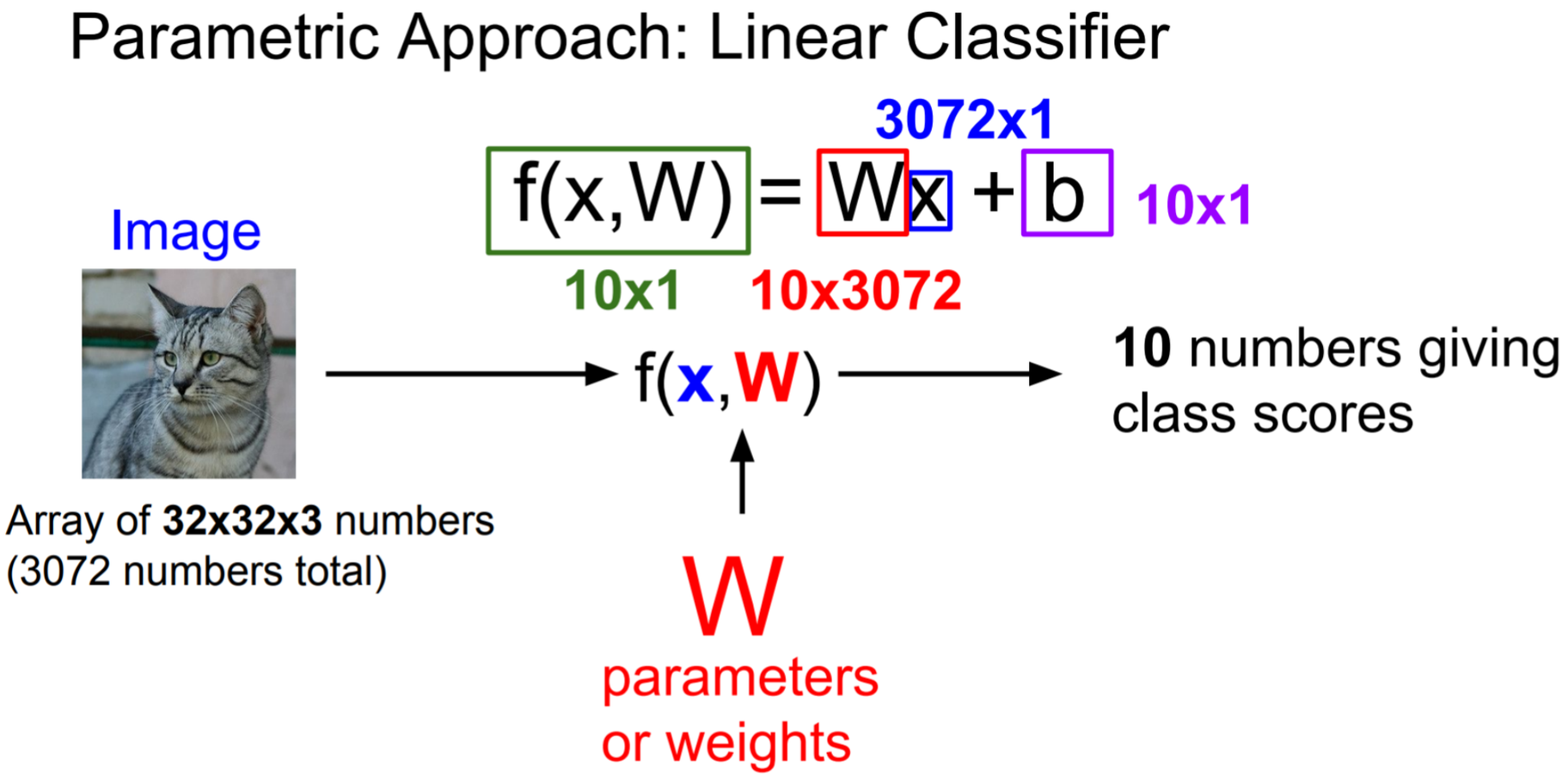
Znajdowanie najbliższych sąsiadów

Distance Metric to compare images

L1 distance: $d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$

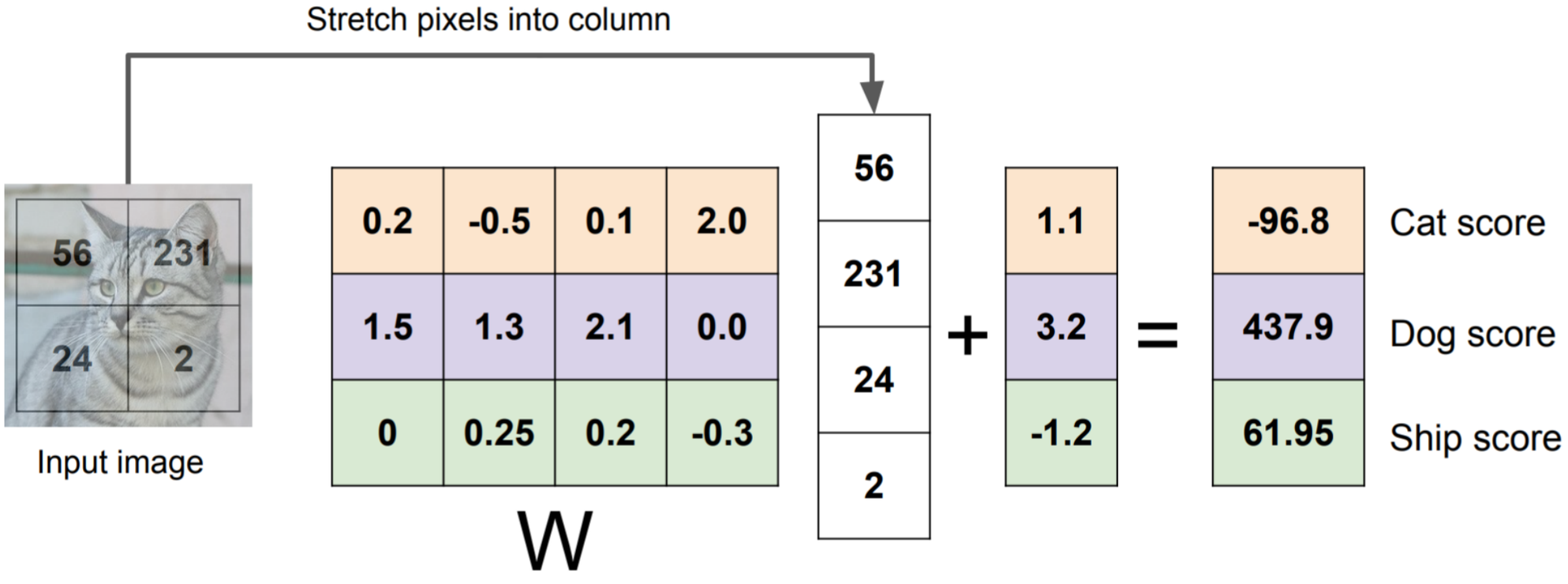
test image					training image					pixel-wise absolute value differences				
56	32	10	18	-	10	20	24	17	=	46	12	14	1	add → 456
90	23	128	133		8	10	89	100		82	13	39	33	
24	26	178	200		12	16	178	170		12	10	0	30	
2	0	255	220		4	32	233	112		2	32	22	108	

Klasyfikator liniowy (maszyna liniowa)

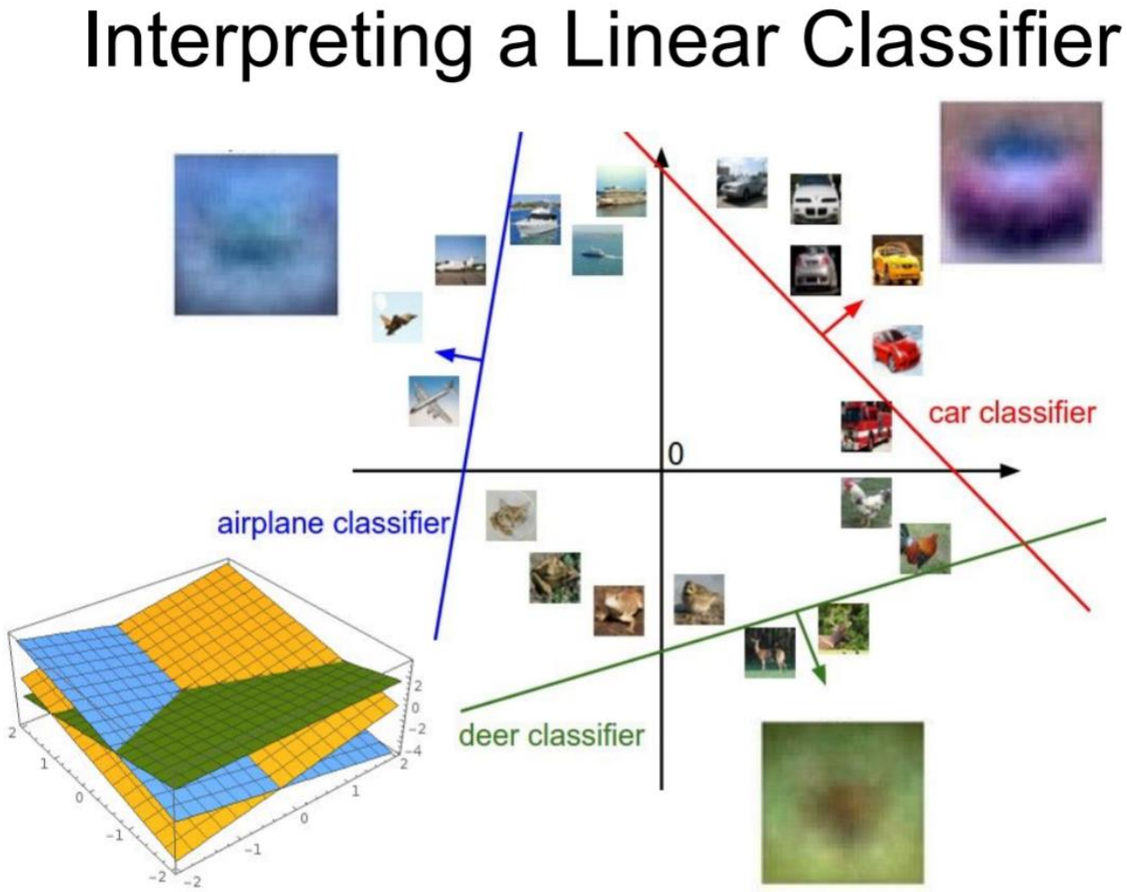


Klasyfikator liniowy (maszyna liniowa)

Example with an image with 4 pixels, and 3 classes (cat/dog/ship)



Klasyfikator liniowy (maszyna liniowa)



$$f(x,W) = Wx + b$$

Array of **32x32x3** numbers
(3072 numbers total)

Klasyfikacja obrazów

Klasyfikator liniowy (maszyna liniowa)

$$f(x,W) = Wx + b$$

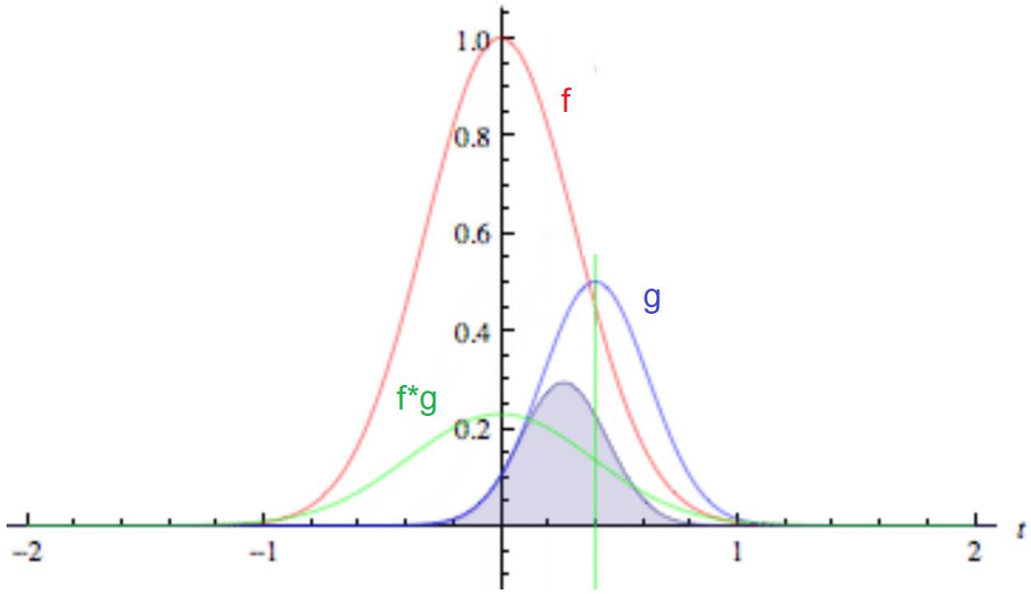


airplane	-3.45	-0.51	3.42
automobile	-8.87	**6.04**	4.64
bird	0.09	5.31	2.65
cat	**2.9**	-4.22	5.1
deer	4.48	-4.19	2.64
dog	8.02	3.58	5.55
frog	3.78	4.49	**-4.34**
horse	1.06	-4.37	-1.5
ship	-0.36	-2.09	-4.79
truck	-0.72	-2.93	6.14

Konwolucja

Konwolucja (splot, splot całkowy, mnożenie splotowe) – działanie określone dla dwóch funkcji (lub opisywanych przez nie sygnałów) dające w wyniku inną funkcję, która może być postrzegana jako zmodyfikowana wersja oryginalnych funkcji. Splot podobny jest do korelacji wzajemnej.

$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau$$



Zobacz symulacje: https://towardsdatascience.com/beginners-guide-for-convolutional-neural-network-cnn-convnets-5a5e725ea581

Konwolucja

Konwolucja obrazu *f* z filtrem ω

$$g(x,y) = \omega * f(x,y) = \sum_{dx=-a}^{a} \sum_{dy=-b}^{b} \omega(dx,dy) f(x+dx, y+dy),$$

where $g(x,y)$ is the filtered image, $f(x,y)$ is the original image, ω is the filter kernel. Every element of the filter kernel is considered by $-a \leq dx \leq a$ and $-b \leq dy \leq b$.

Obraz wejściowy

x →

y ↓

(1,1)	(2,1)	(3,1)	(4,1)	(5,1)	
(1,2)	(2,2)	**(3,2)**	(4,2)	(5,2)	
(1,3)	(2,3)	(3,3)	(4,3)	(5,3)	
(1,4)	(2,4)	(3,4)	(4,4)	(5,4)	
(1,5)	(2,5)	(3,5)	(4,5)	(5,5)	

Filtr

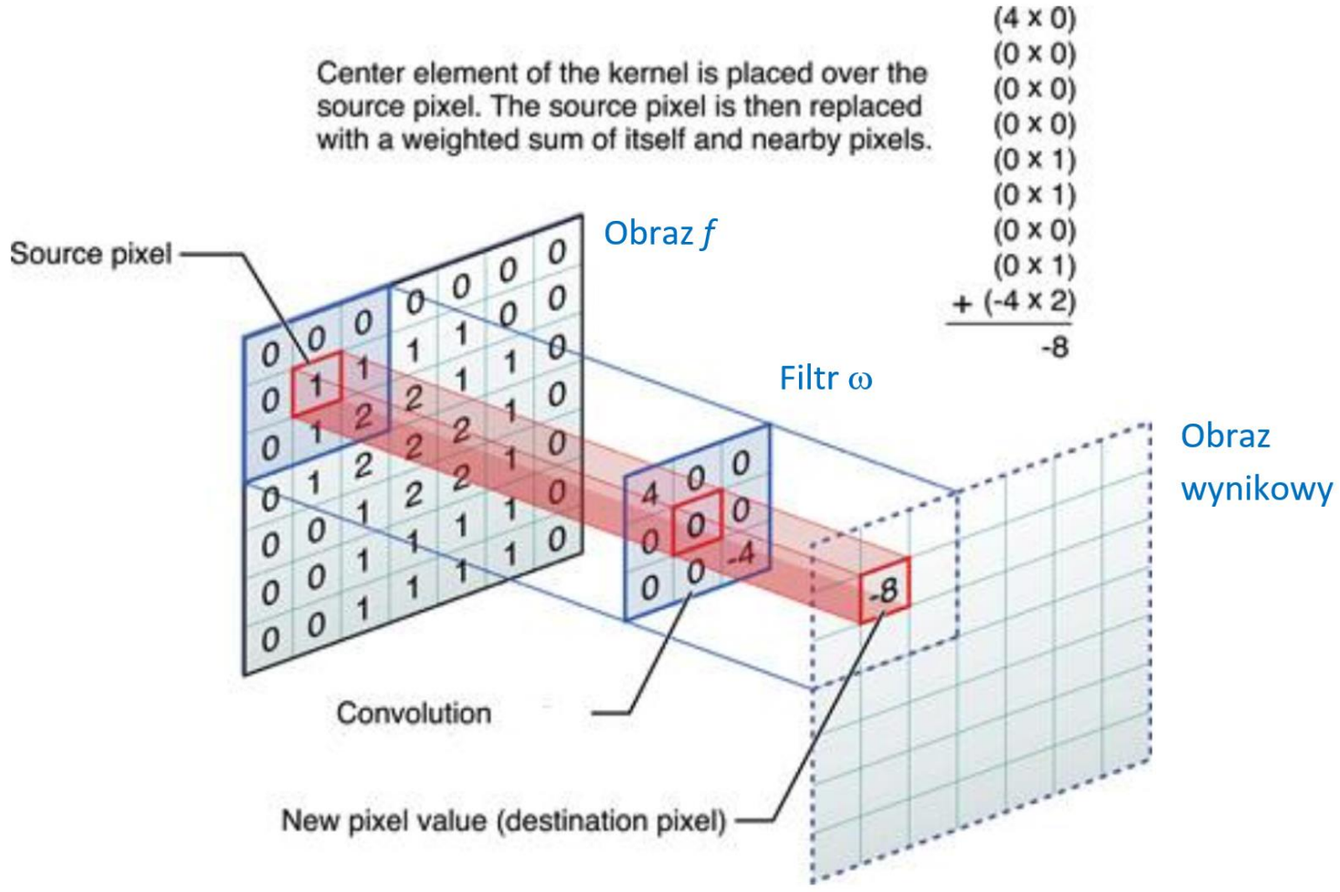
(-1,-1)	(0,-1)	(1,-1)
(-1,0)	**(0,0)**	(1,0)
(-1,1)	(0,1)	(1,1)

Obraz wyjściowy

(1,1)	(2,1)	(3,1)	(4,1)	(5,1)	
(1,2)	(2,2)	**(3,2)**	(4,2)	(5,2)	
(1,3)	(2,3)	(3,3)	(4,3)	(5,3)	
(1,4)	(2,4)	(3,4)	(4,4)	(5,4)	
(1,5)	(2,5)	(3,5)	(4,5)	(5,5)	

Konwolucja obrazu z filtrem

Konwolucja obrazu *f* z filtrem ω (w uproszczeniu – przemnażamy odpowiadające piksele obrazu i filtru i dodajemy wyniki)



Convolution Operation on a 7×7 matrix with a 3×3 kernel

Operację konwolucji można zapisać jako kombinację liniową:

$$\mathbf{w}^T\mathbf{x} = \sum_{i-1}^{n} w_i x_i$$

gdzie $\mathbf{x} = [x_1, x_2, \dots, x_n]$ – piksele obrazu

$\mathbf{w} = [w_1, w_2, \dots, w_n]$ – piksele filtra

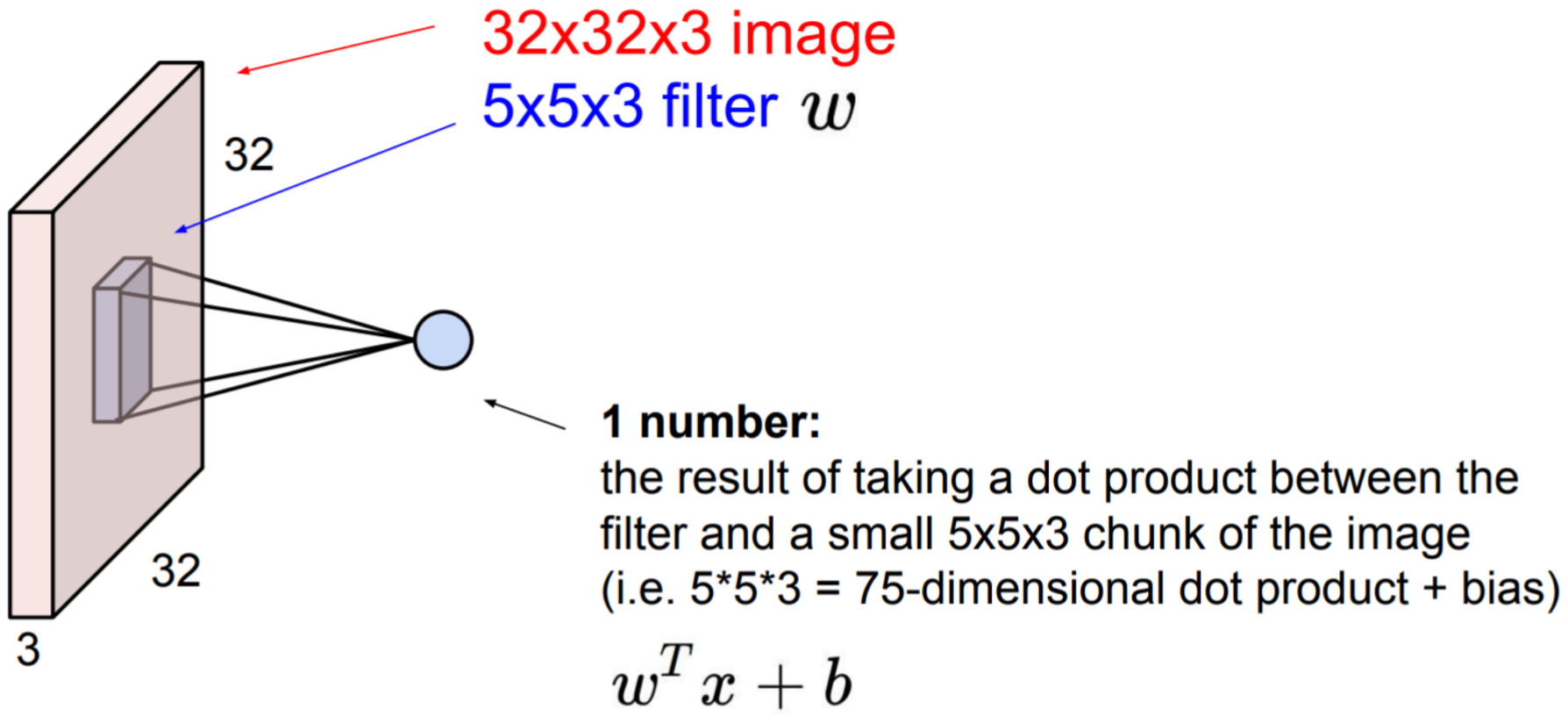
W naszym przypadku:

$$\mathbf{x} = [0,0,0,0,1,1,0,1,2]$$

$$\mathbf{w} = [4,0,0,0,0,0,0,0,-4]$$

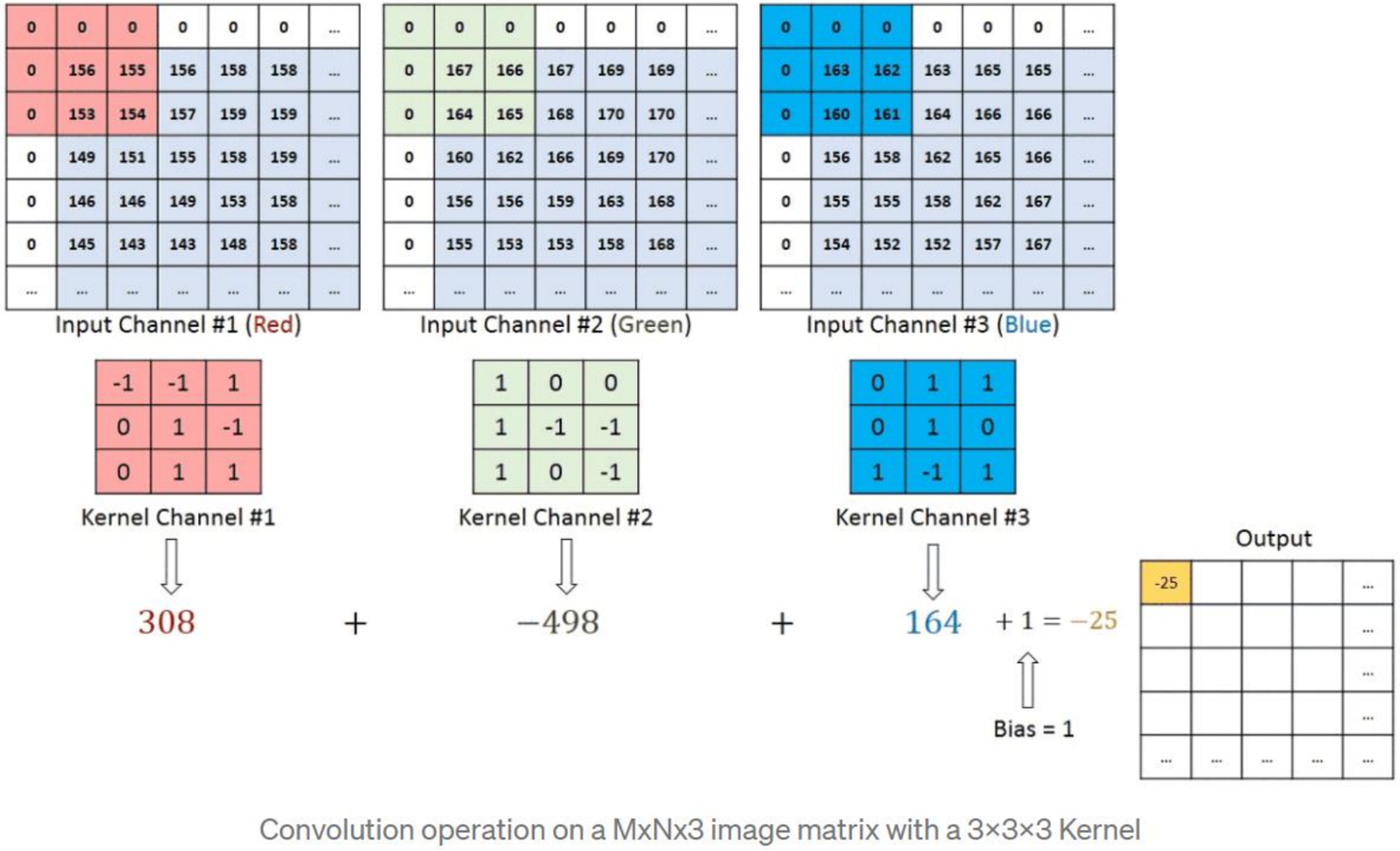
Konwolucja obrazu z filtrem

Trzy kanały RGB



Konwolucja obrazu z filtrem

Trzy kanały RGB



Convolution operation on a MxNx3 image matrix with a 3×3×3 Kernel

Konwolucja obrazu z filtrem

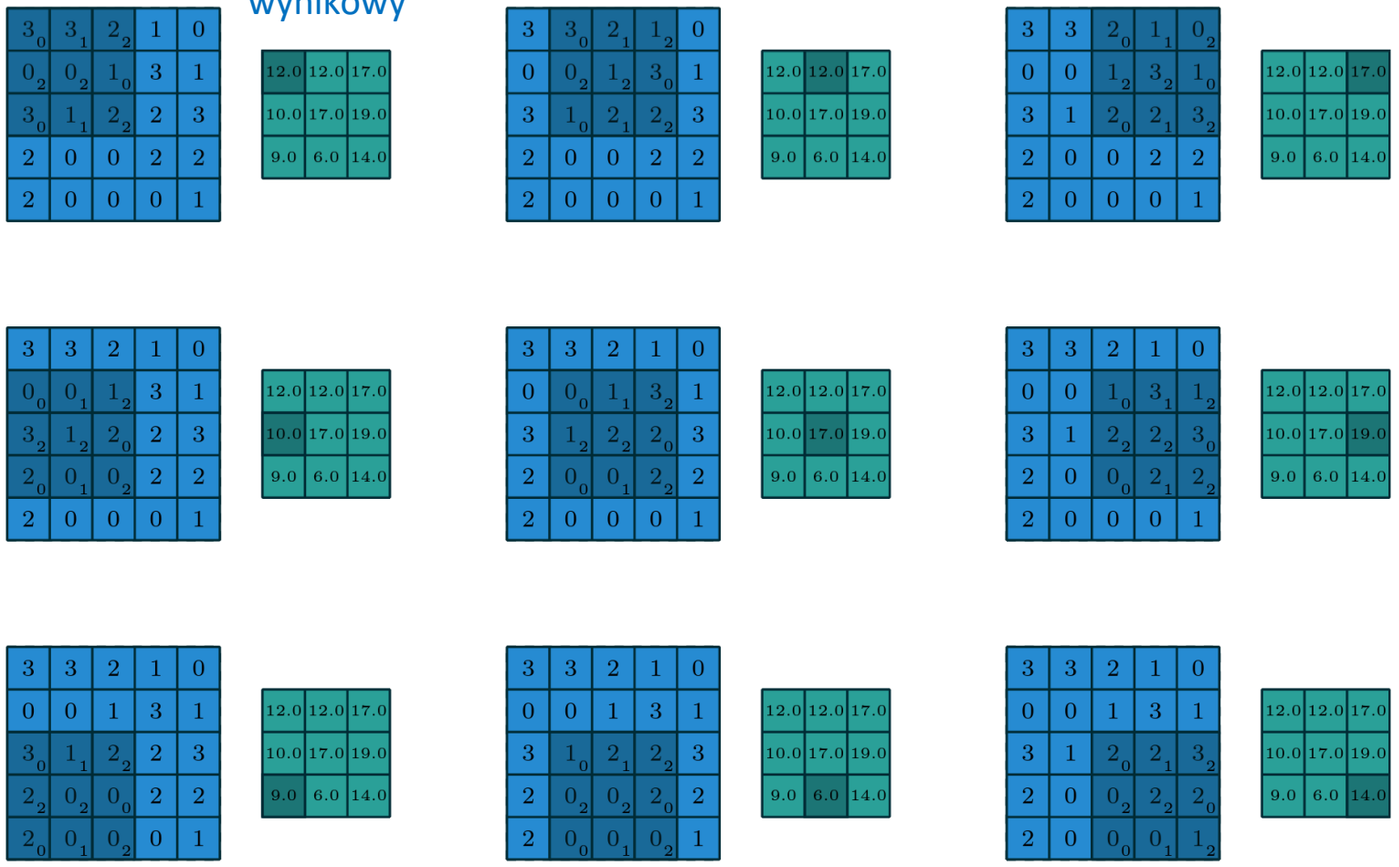
Konwolucja obrazu *f* z filtrem ω - filtr przesuwamy po obrazie, tworząc nowe „obrazy”

Postać filtru:

0	1	2
2	2	0
0	1	2

Obraz wejściowy

Obraz wynikowy

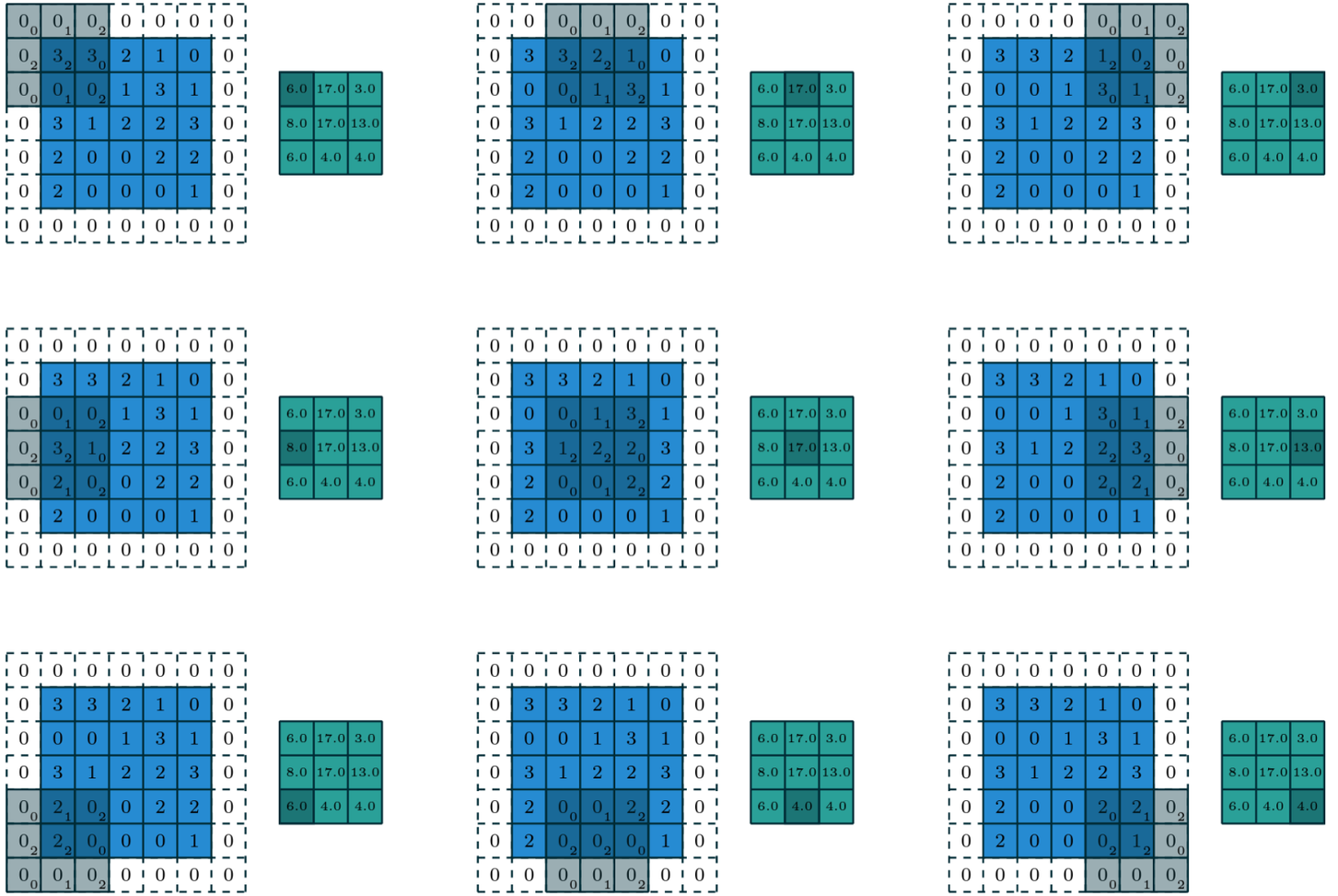


https://arxiv.org/pdf/1603.07285.pdf

Konwolucja obrazu z filtrem

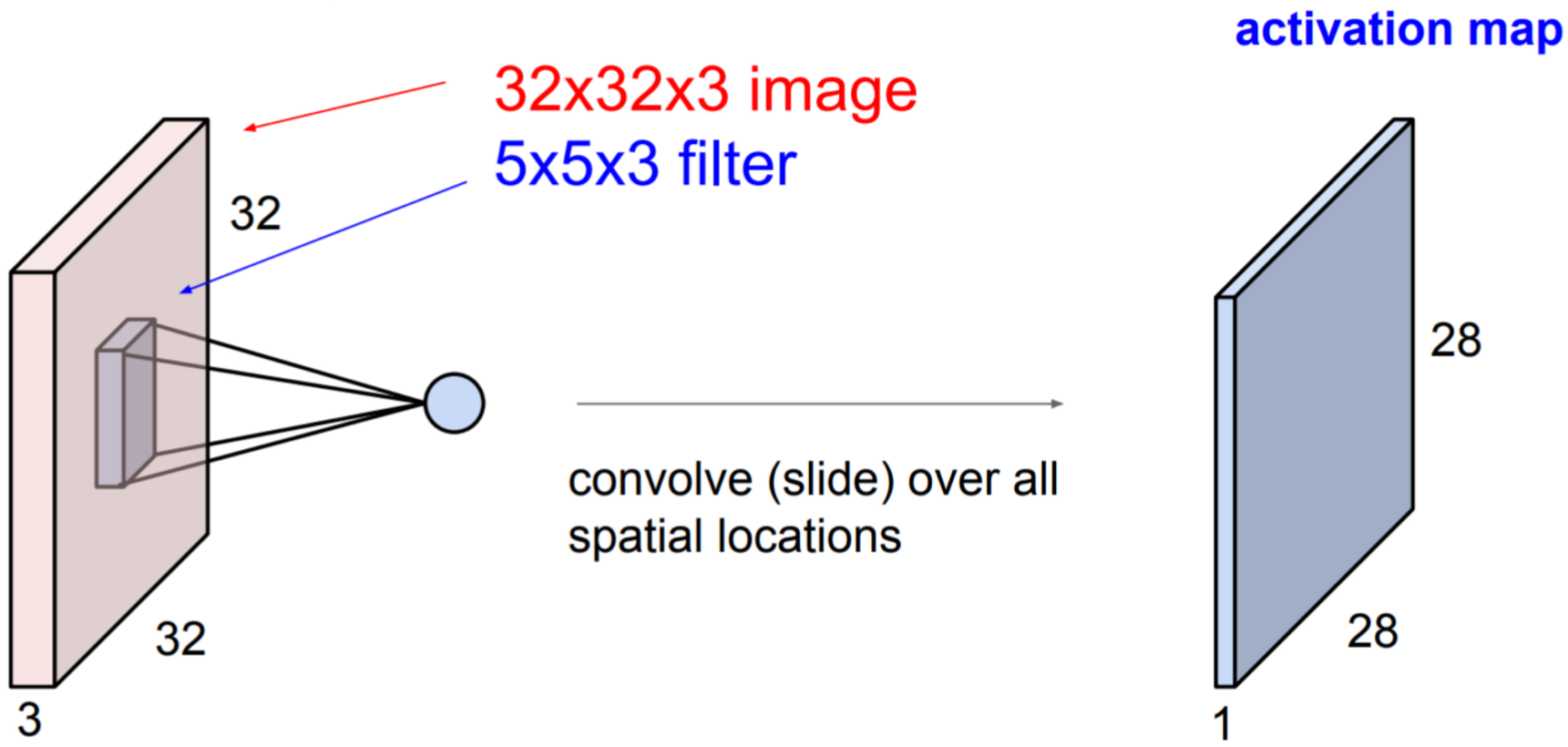
Konwolucja obrazu *f* z filtrem ω - filtr przesuwamy po obrazie, tworząc nowe „obrazy”

Inny wariant: skok przesunięcia filtra (*stride*) równy 2, „rozszerzenie” obrazu (*padding*).



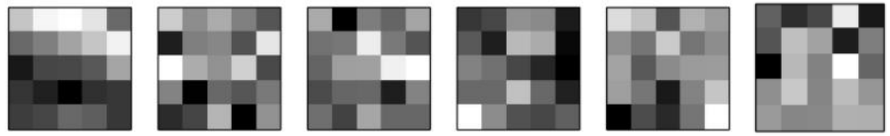
Konwolucja obrazu z filtrem

Konwolucja obrazu f z filtrem ω - trzy kanały RGB, przesuwanie filtra tworzy nowy „obraz”

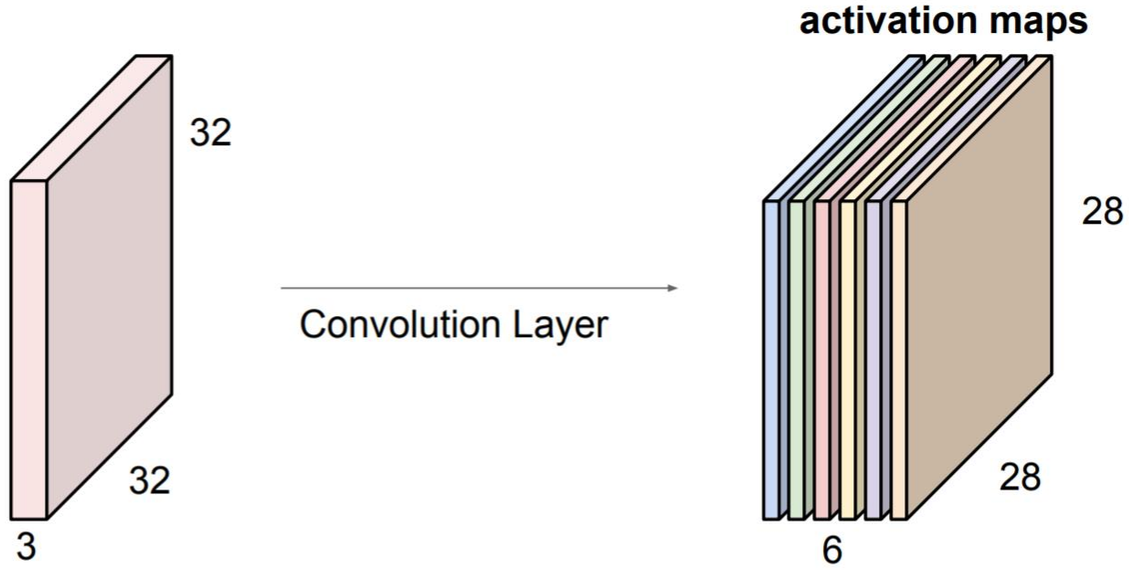


Konwolucja obrazu z filtrem

Kilka filtrów tworzy nowy ”obraz” wielowymiarowy

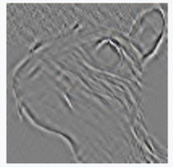
Filtry: 

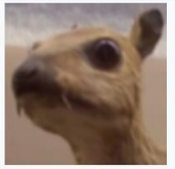
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



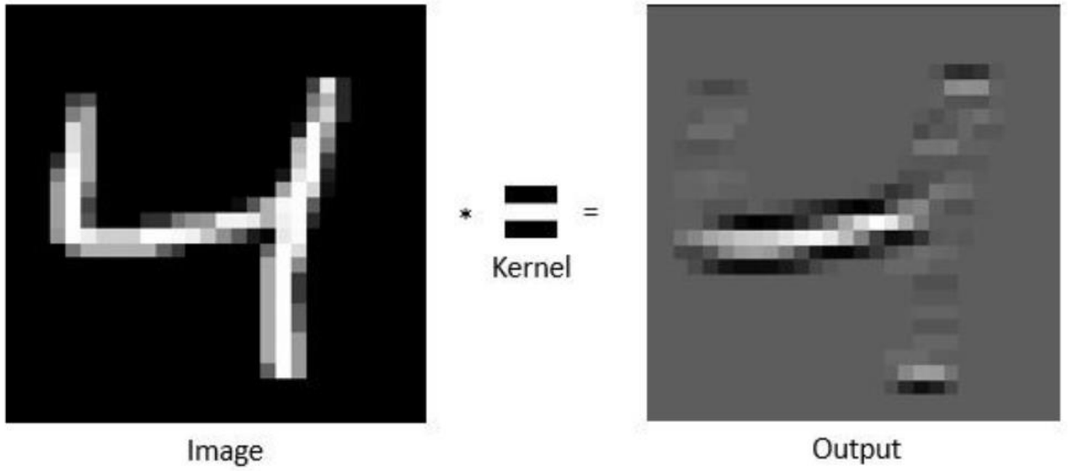
We stack these up to get a “new image” of size 28x28x6!

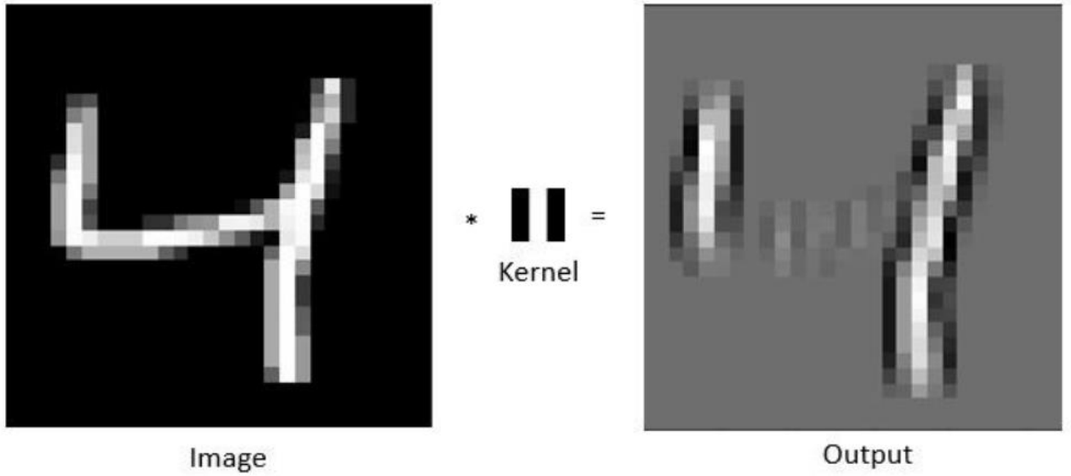
Przykłady filtrów

Operation	Kernel ω	Image result g(x,y)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Ridge detection	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	

Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur 3 × 3 (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	
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)	$\frac{-1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & -476 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	

Przykłady filtrów

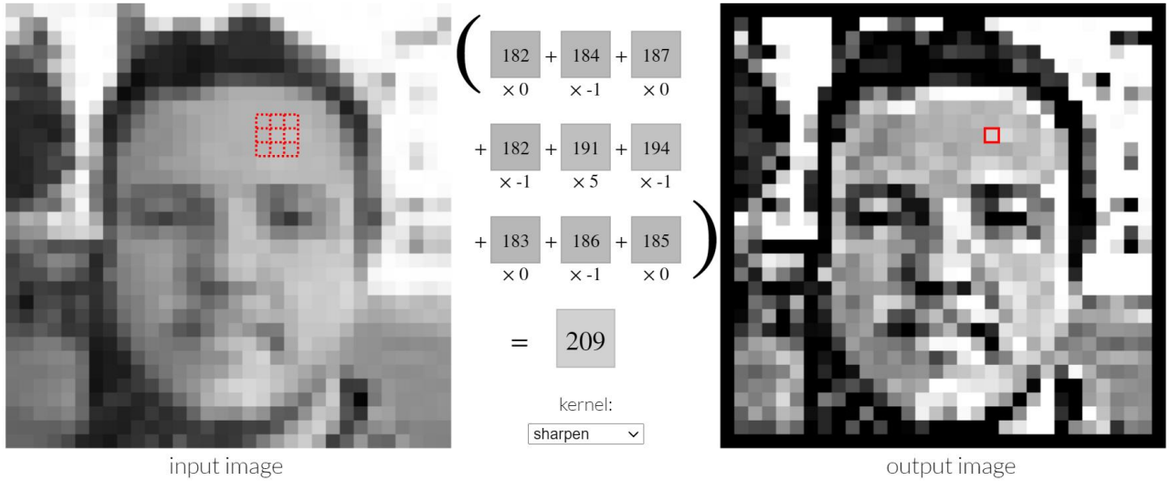




Przykłady filtrów

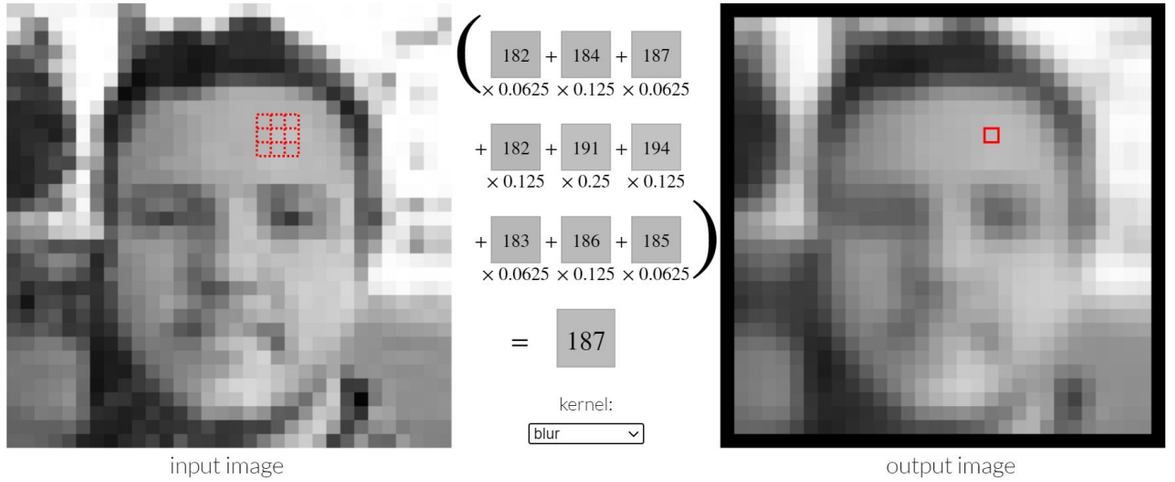
Filtr sharpen

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$



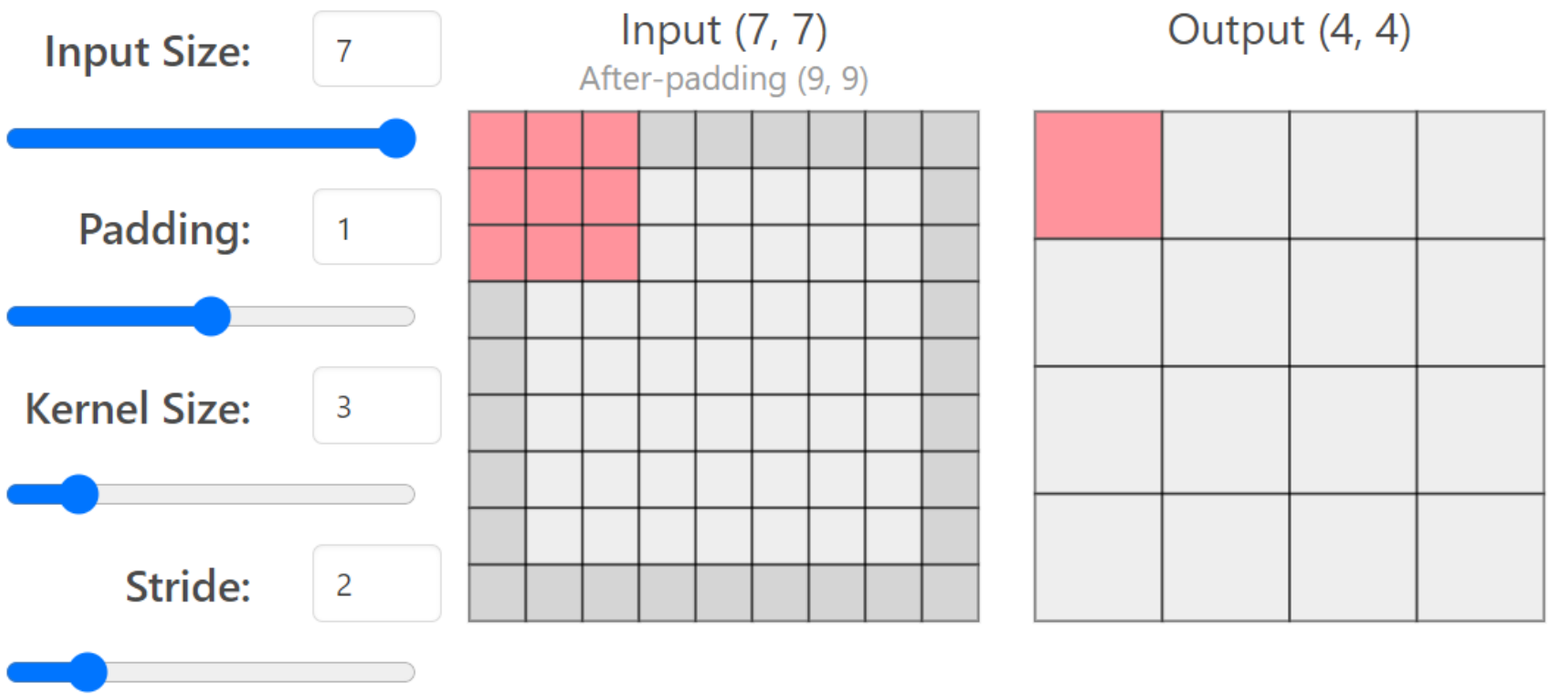
Filtr blur

$$\begin{pmatrix} 0.0625 & 0.125 & 0.0625 \\ 0.125 & 0.25 & 0.125 \\ 0.0625 & 0.125 & 0.0625 \end{pmatrix}$$



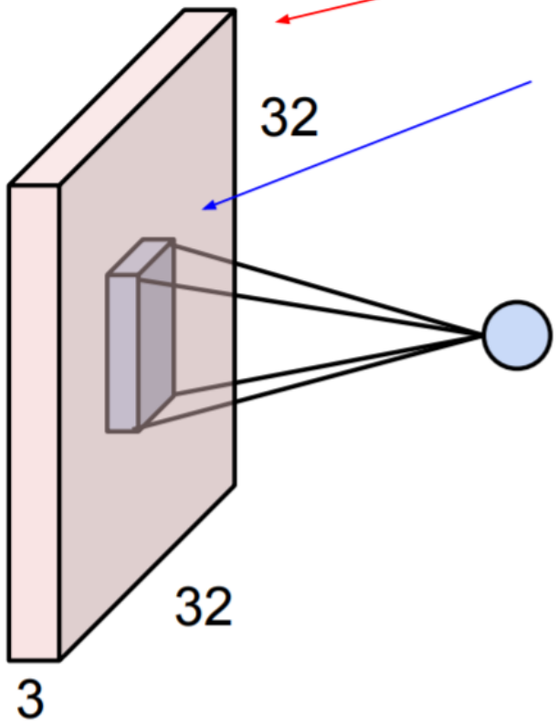
https://setosa.io/ev/image-kernels/

(Hiper)parametry konwolucji



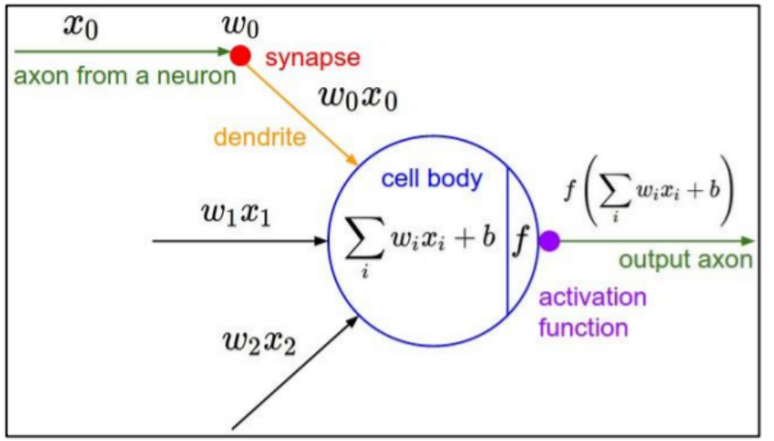
https://poloclub.github.io/cnn-explainer/#article-pooling

Konwolucyjny neuron



32x32x3 image

5x5x3 filter

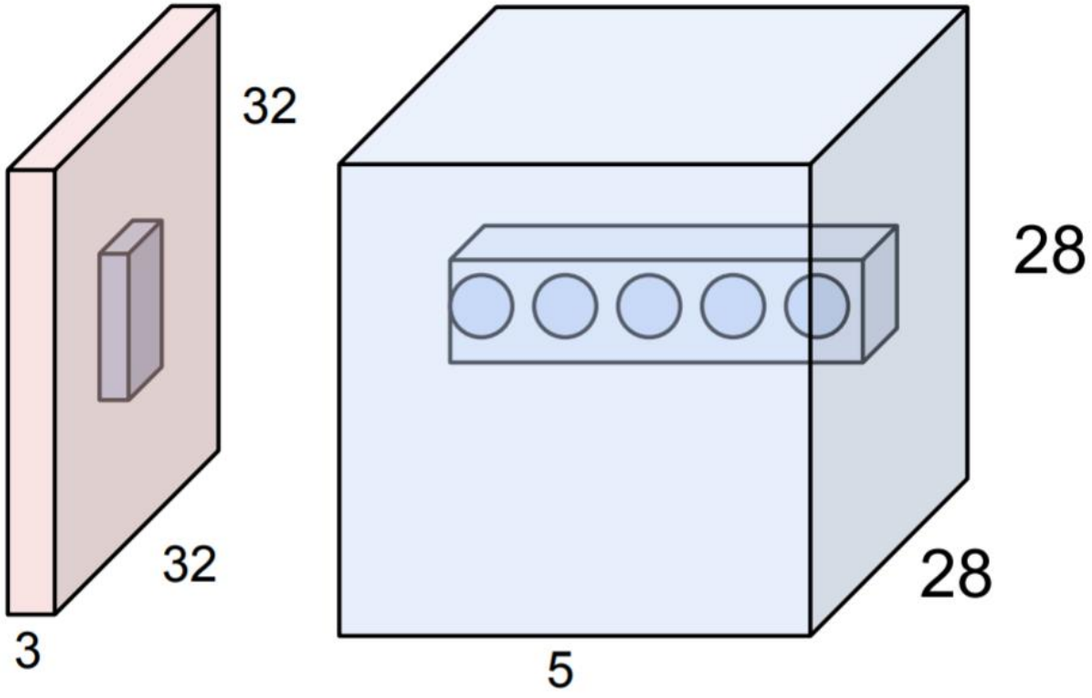


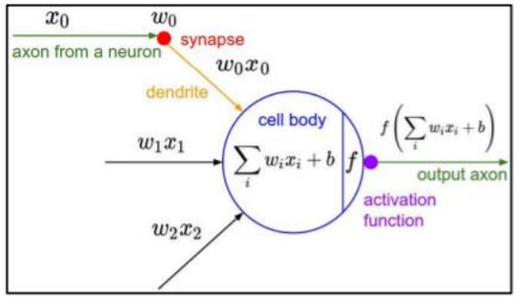
It's just a neuron with local connectivity...

1 number:
the result of taking a dot product between the filter and this part of the image
(i.e. 5*5*3 = 75-dimensional dot product)

Konwolucyjny neuron

The brain/neuron view of CONV Layer





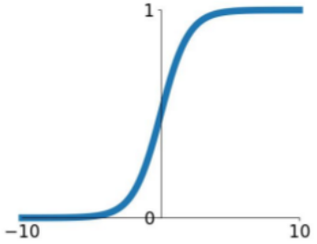
E.g. with 5 filters,
CONV layer consists of neurons arranged in a 3D grid (28x28x5)

There will be 5 different neurons all looking at the same region in the input volume

Funkcje aktywacji

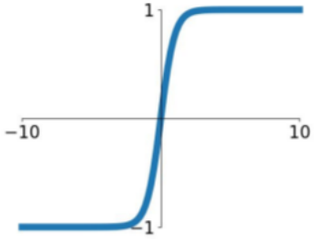
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



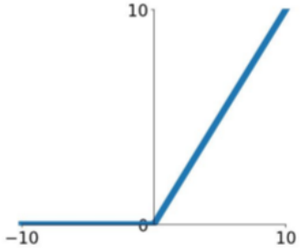
tanh

$$\tanh(x)$$



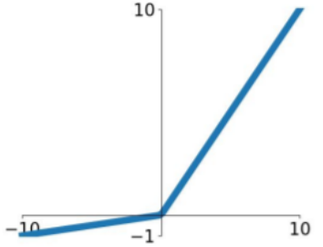
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

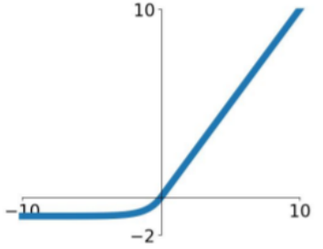


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



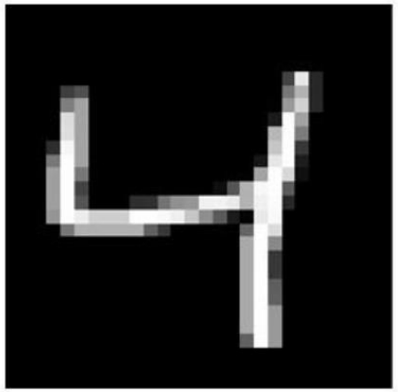
Działanie ReLU

Rectified Linear Unit

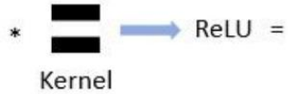
1	14	-9	4
-2	-20	10	6
-3	3	11	1
2	54	-2	80

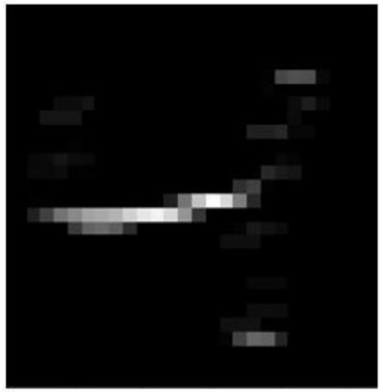
ReLU

1	14	0	4
0	0	10	6
0	3	11	1
2	54	0	80

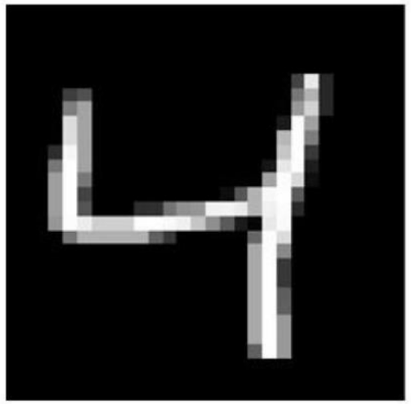


Image

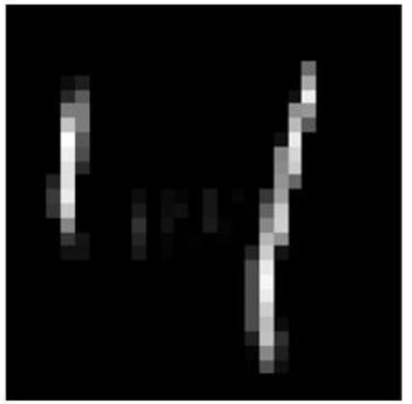




Output



Image

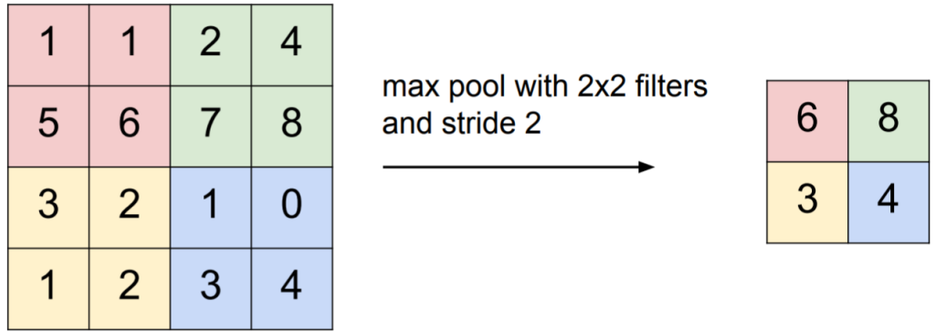


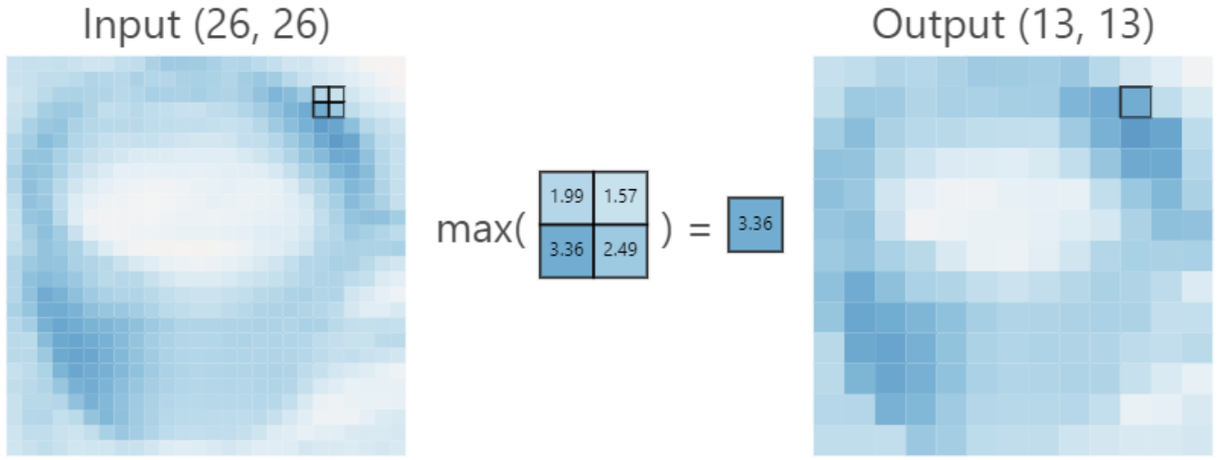
Output

Operacja pooling

Cel – redukcja wymiaru obrazu, wzmocnienie generalizacji

Przesuwamy po obrazie okienko (np. 2x2) ze skokiem np. 2 i notujemy największą wartość w okienku (maxpool)





Operacja pooling

Inne warianty

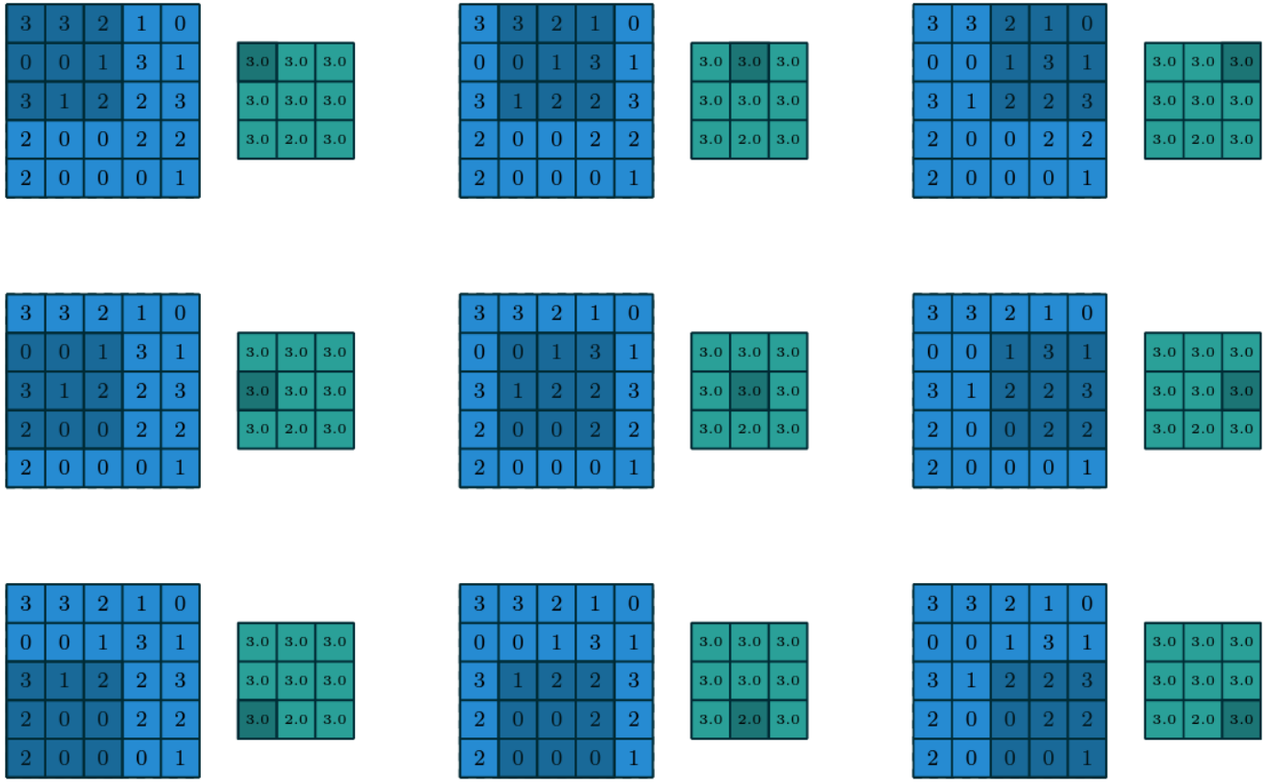


Figure 1.6: Computing the output values of a 3×3 max pooling operation on a 5×5 input using 1×1 strides.

Operacja pooling

Inne warianty

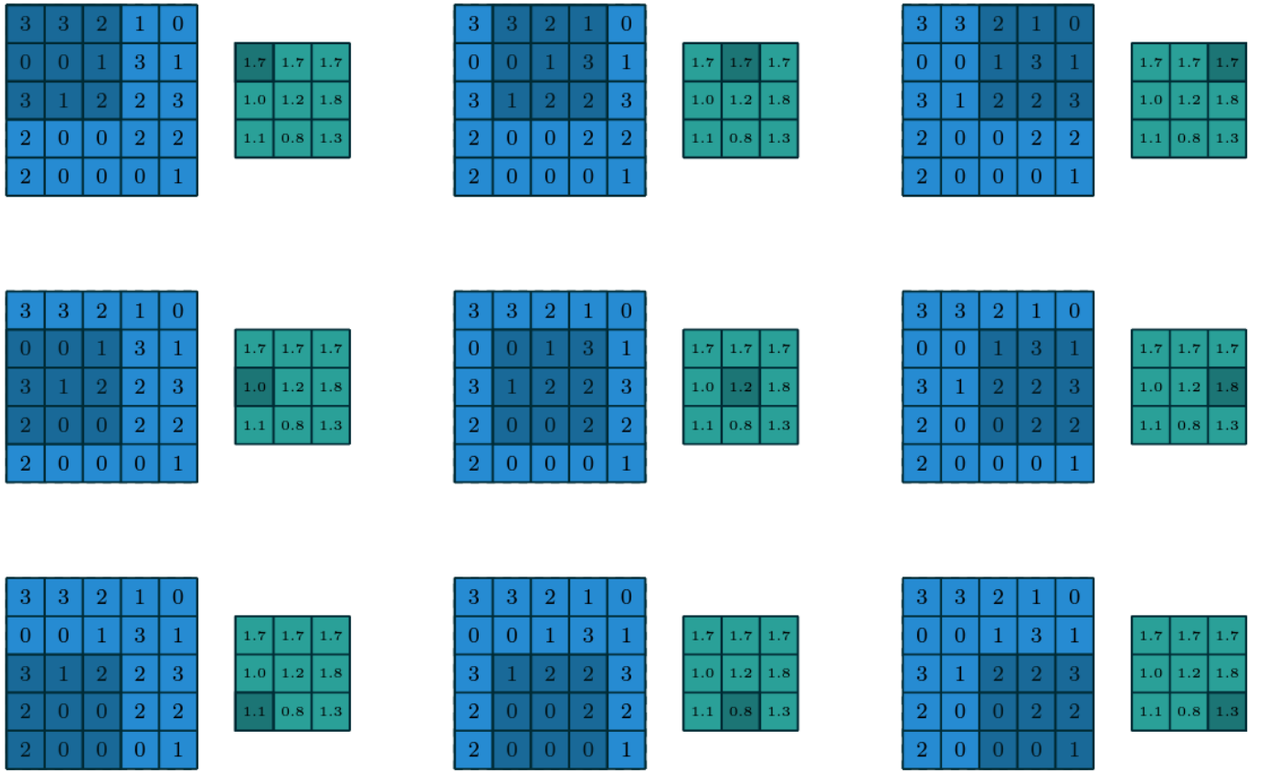
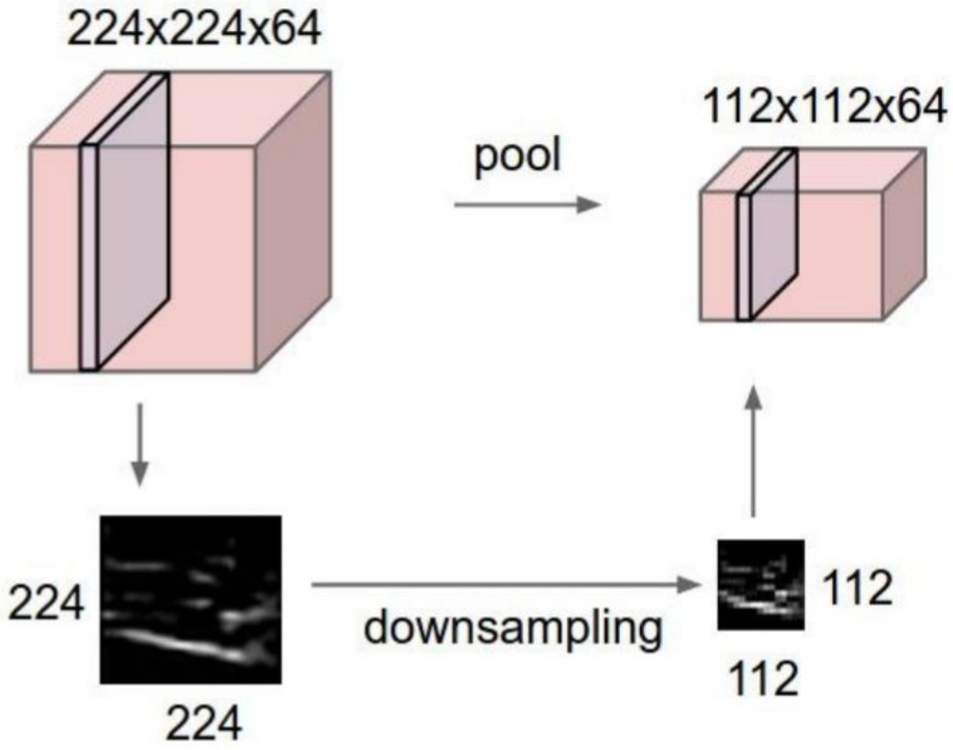


Figure 1.5: Computing the output values of a 3×3 average pooling operation on a 5×5 input using 1×1 strides.

Operacja pooling

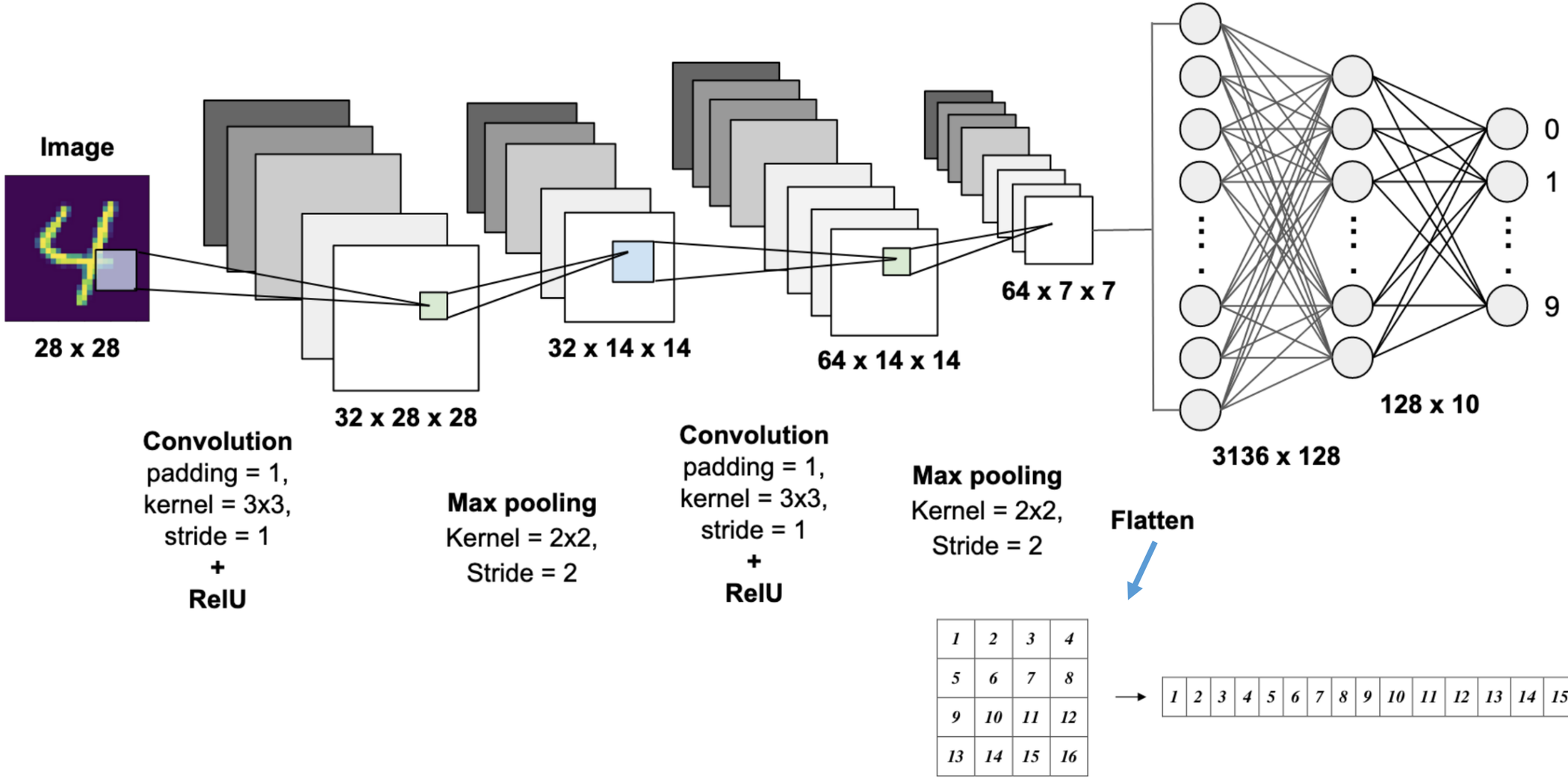
Pooling layer

- makes the representations smaller and more manageable
- operates over each activation map independently:

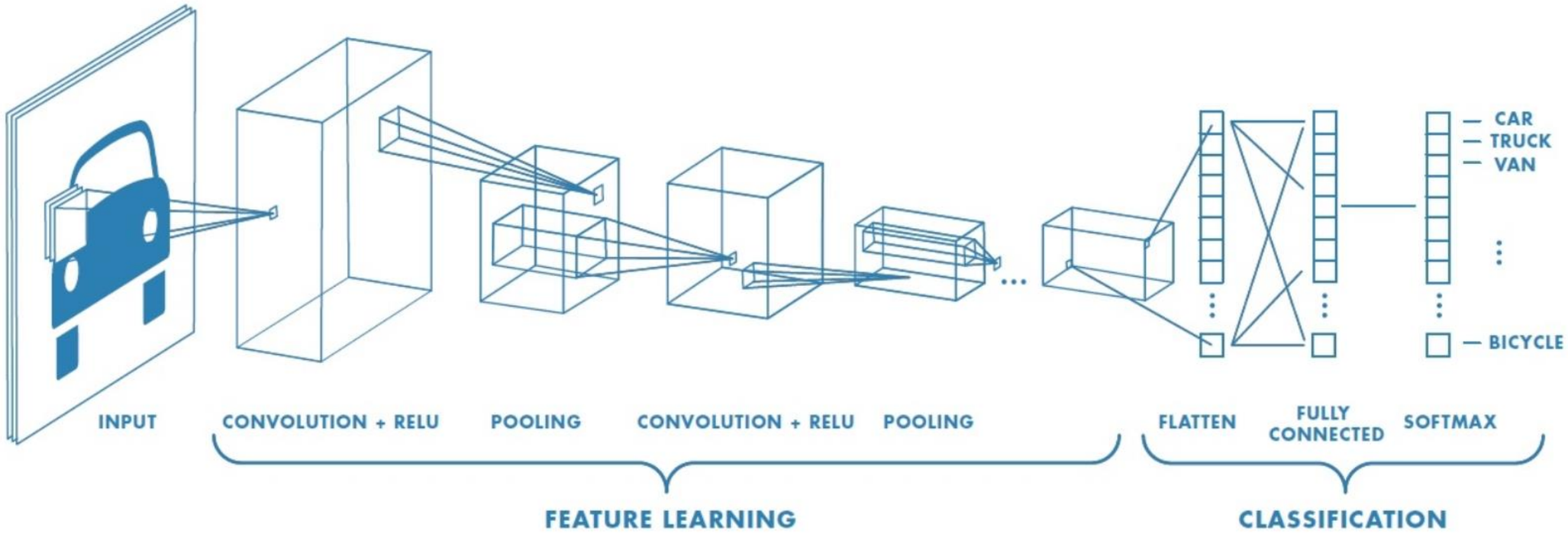


Sieć konwolucyjna

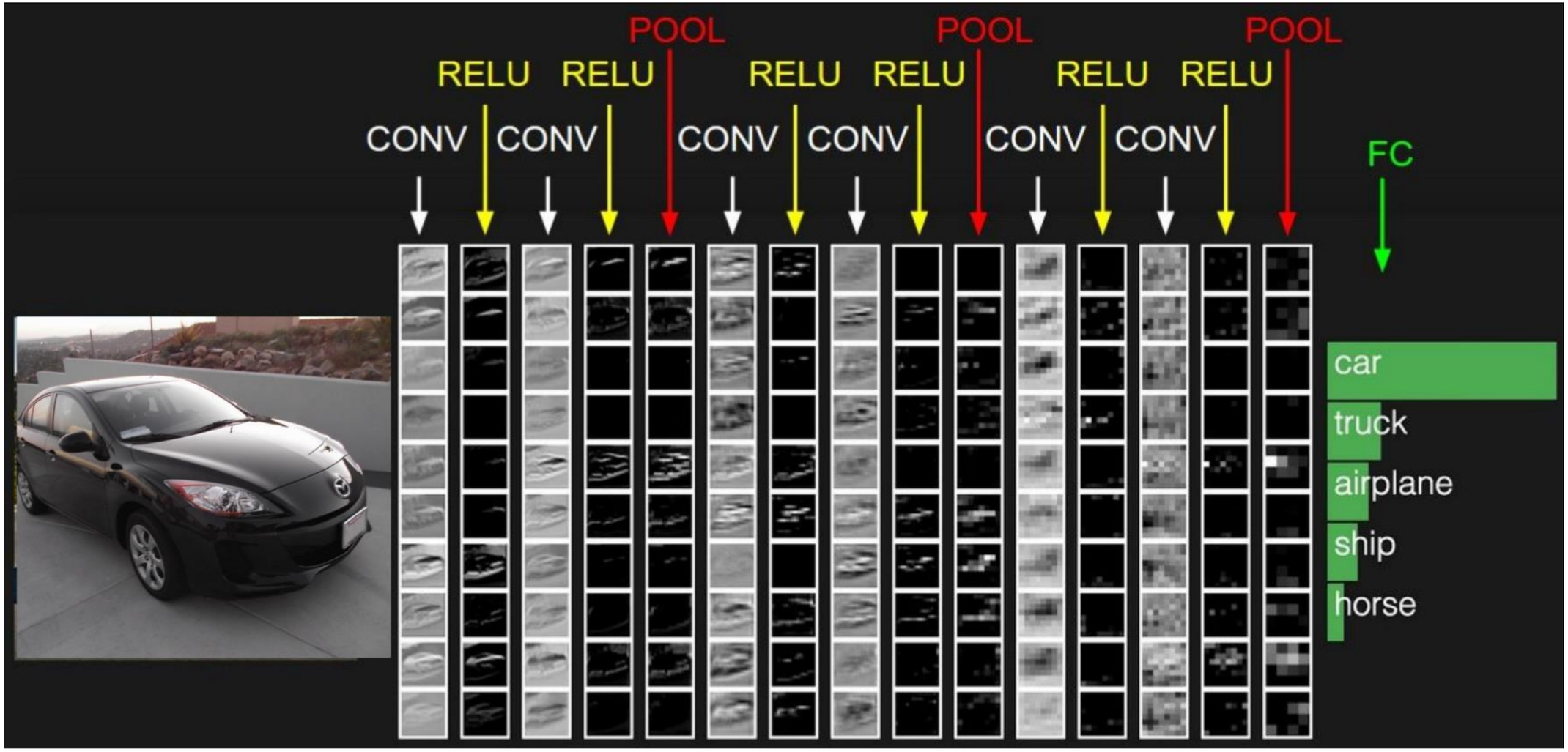
Wiele warstw konwolucyjnych + ReLU + pooling (opcjonalnie) + klasyfikator (np. wielowarstwowy perceptron)



Sieć konwolucyjna

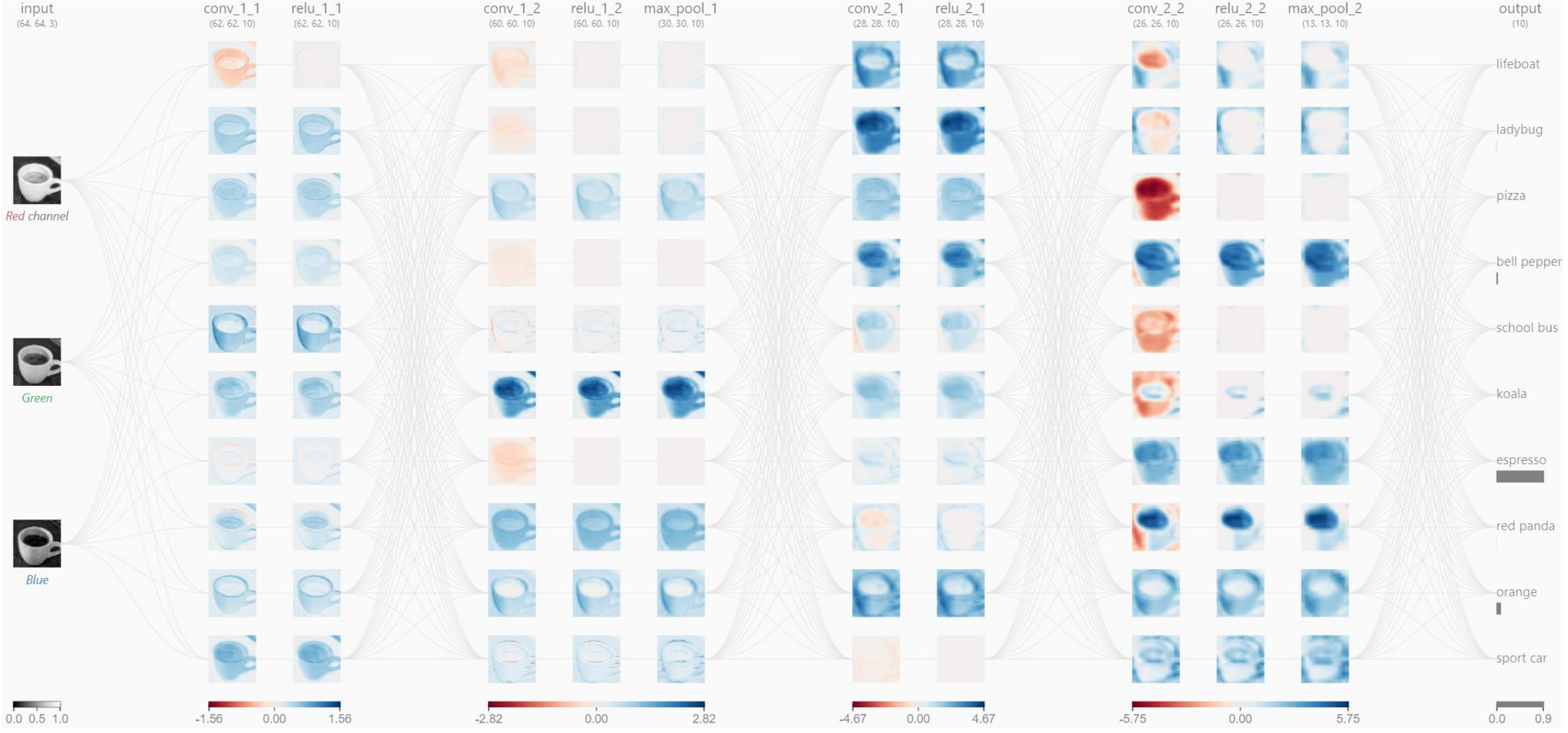


Sieć konwolucyjna



FC – fully connected

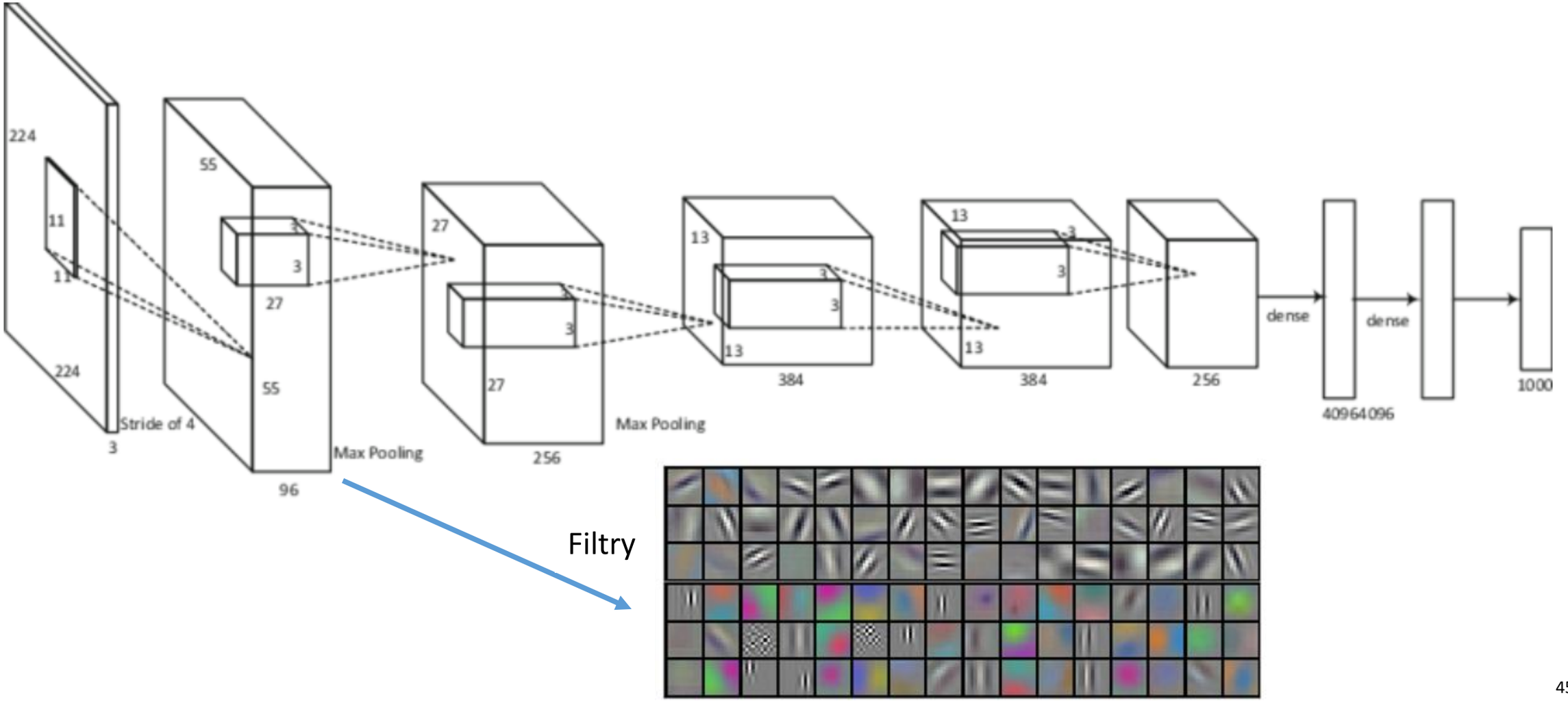
Sieć konwolucyjna



https://poloclub.github.io/cnn-explainer/, https://www.youtube.com/watch?v=HnWIHWFbuUQ

Sieć konwolucyjna

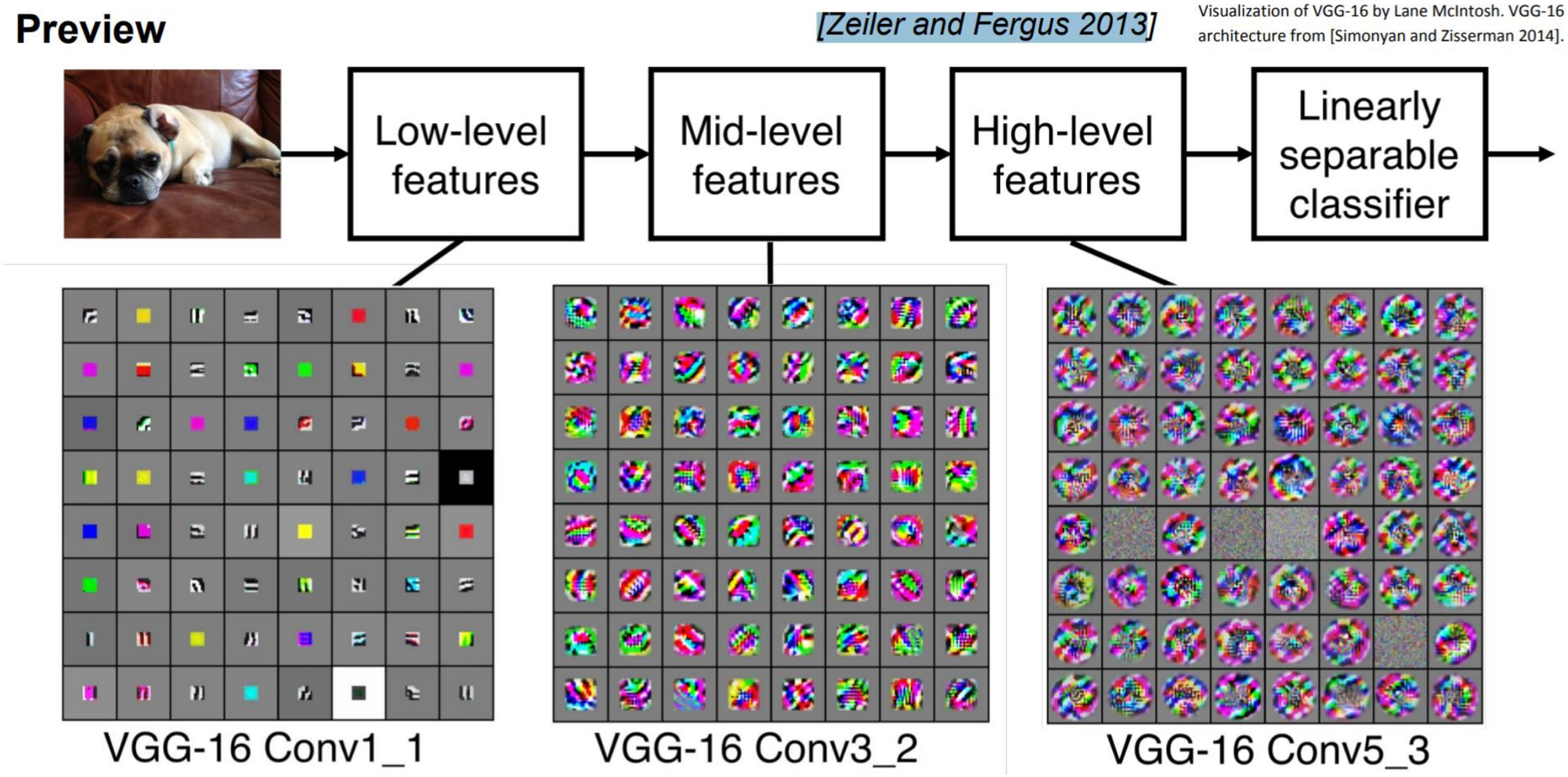
Below a popular CNN that won the 2012 ImageNet competition (AlexNet).



https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html

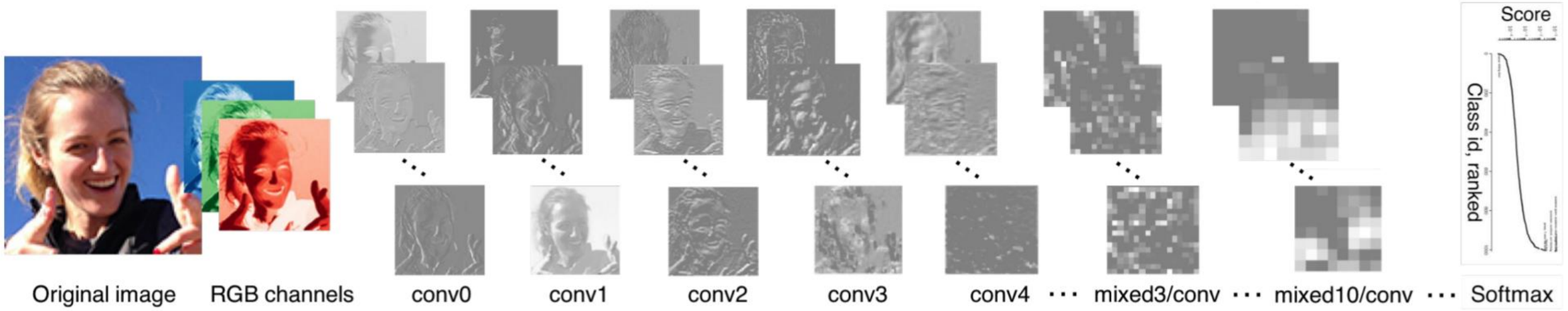
Sieć konwolucyjna

Przetwarzanie obrazu przez sieć konwolucyjną



Sieć konwolucyjna

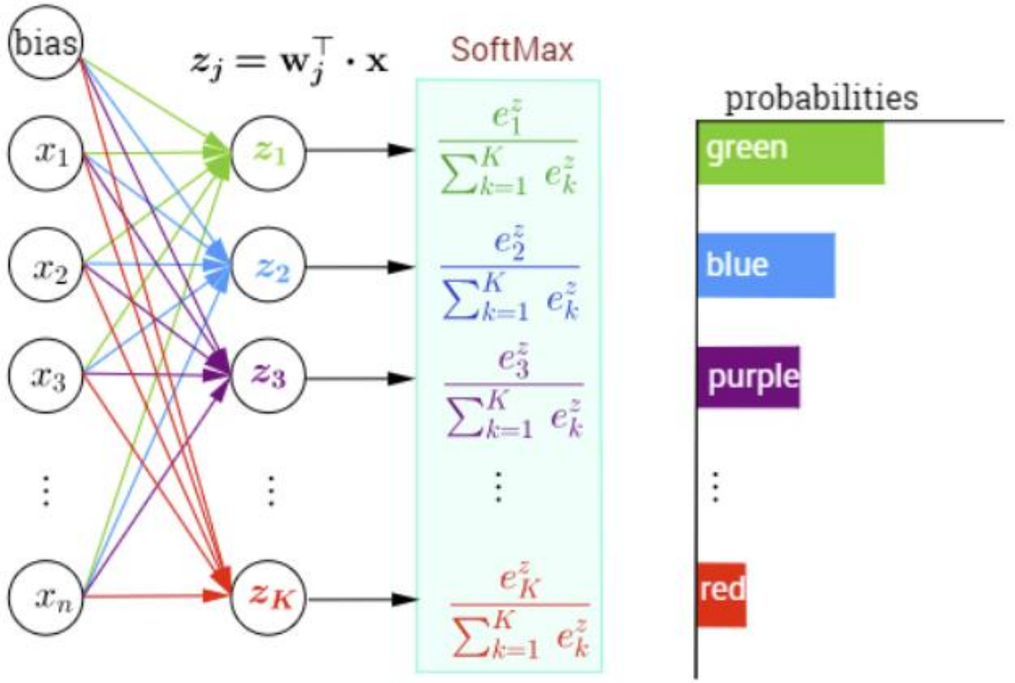
Przetwarzanie obrazu przez sieć konwolucyjną



Sieć konwolucyjna

Klasyfikator (fully connected layer)

$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_K \end{bmatrix} = \begin{bmatrix} \mathbf{w}_1^\top \\ \mathbf{w}_2^\top \\ \mathbf{w}_3^\top \\ \vdots \\ \mathbf{w}_K^\top \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$$



The softmax as

$$\sigma(j) = \frac{\exp(\mathbf{w}_j^\top \mathbf{x})}{\sum_{k=1}^K \exp(\mathbf{w}_k^\top \mathbf{x})} = \frac{\exp(z_j)}{\sum_{k=1}^K \exp(z_k)}$$

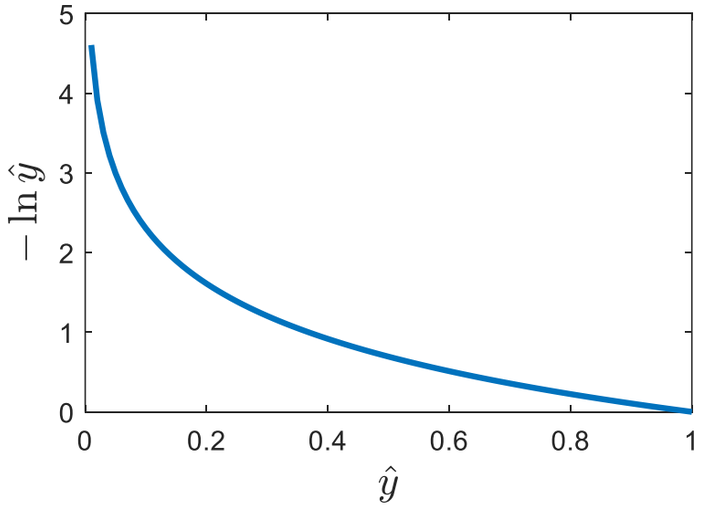
Trening CNN

Cel treningu: wyznaczyć parametry CNN tak, aby wartość funkcji straty (*loss function*) była jak najmniejsza.

Funkcja straty: entropia krzyżowa (*cross entropy*)

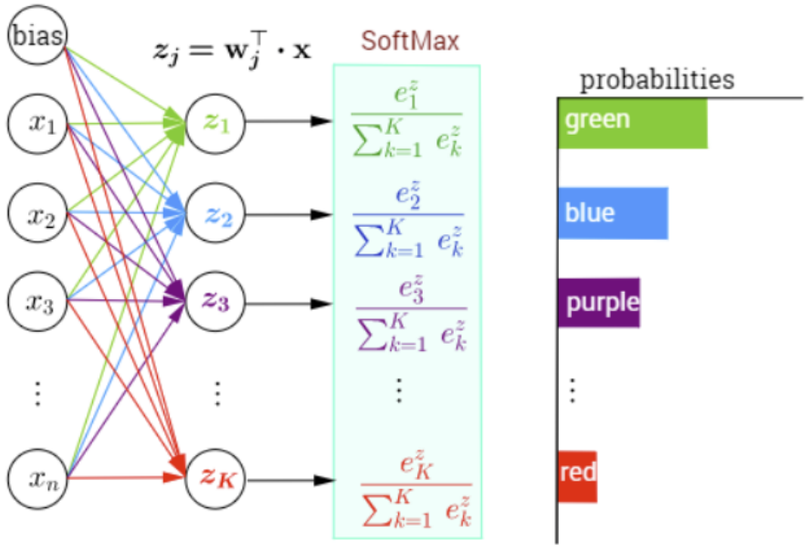
$$L = -\sum_{k=1}^{K} y_k \ln \hat{y}_k$$

gdzie K – liczba klas, $y_k \in \{0, 1\}$ – pożądana odpowiedź dla klasy k, $\hat{y}_k \in [0, 1]$ – odpowiedź dla klasy k.



Trening CNN

Przykład dla trzech klas



Wektor pożądanych odpowiedzi dla klasy 1 (*one-hot encoding*): $\mathbf{y} = [1, 0, 0]$

Odpowiedź sieci (*SoftMax*): $\hat{\mathbf{y}} = [0.6, 0.1, 0.3]$

Wartość funkcji starty:

$$L = -[1 \cdot \ln(0.6) + 0 \cdot \ln(0.1) + 0 \cdot \ln(0.3)] = -[-0.51 + 0 + 0] = 0.51$$

Trening CNN

Minimalizacja funkcji straty:

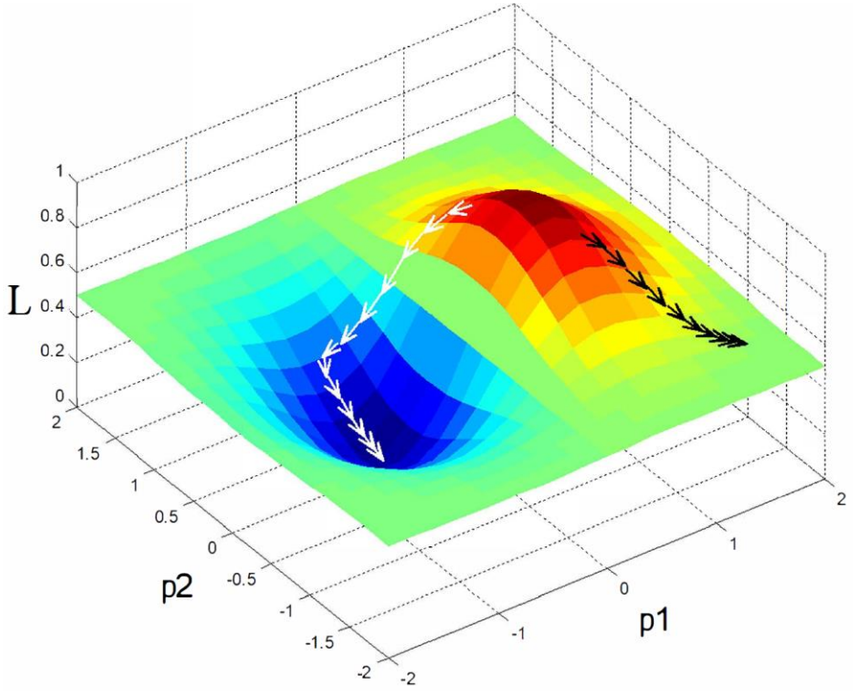
$$L = -\frac{1}{N}\sum_{j=1}^{N}\sum_{k=1}^{K} y_{j,k}\ln\hat{y}_{j,k}$$

Uczenie metodami gradientowymi (*gradient descent , gradient-based learning*)

$$\nabla L(\mathbf{p}) = \left[\frac{\partial L}{\partial p_1}, \frac{\partial L}{\partial p_2}, \ldots, \frac{\partial L}{\partial p_m}\right]$$

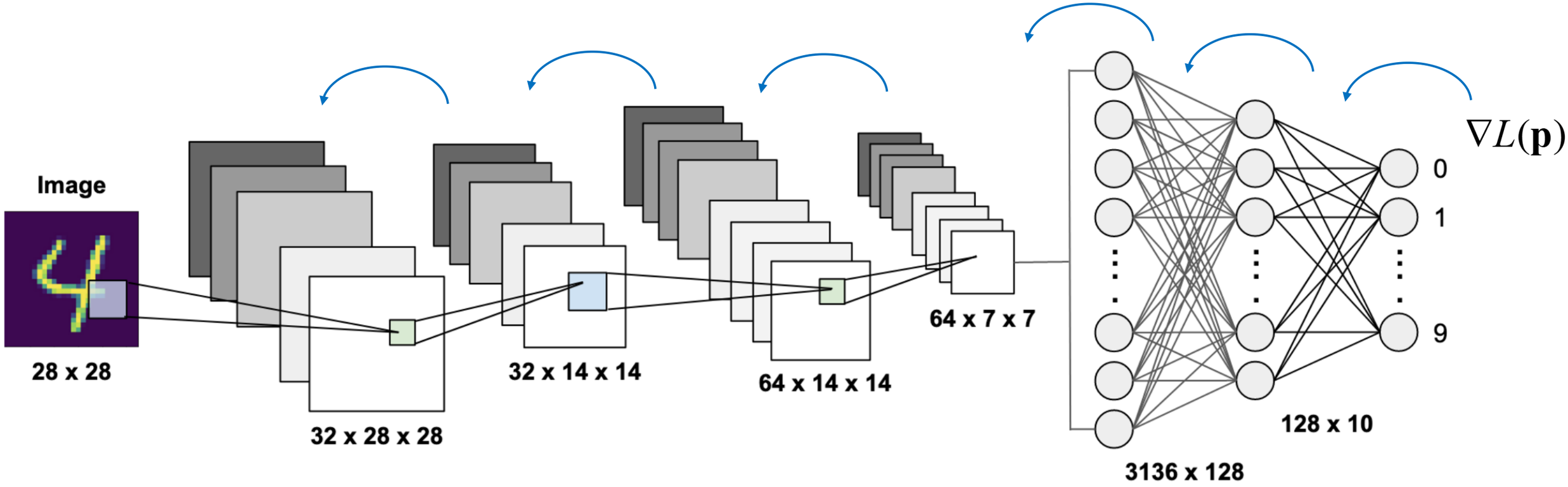
$$\mathbf{p}' = \mathbf{p} - \eta \nabla L(\mathbf{p})$$

$\eta > 0$ – współczynnik uczenia (*learning rate*)

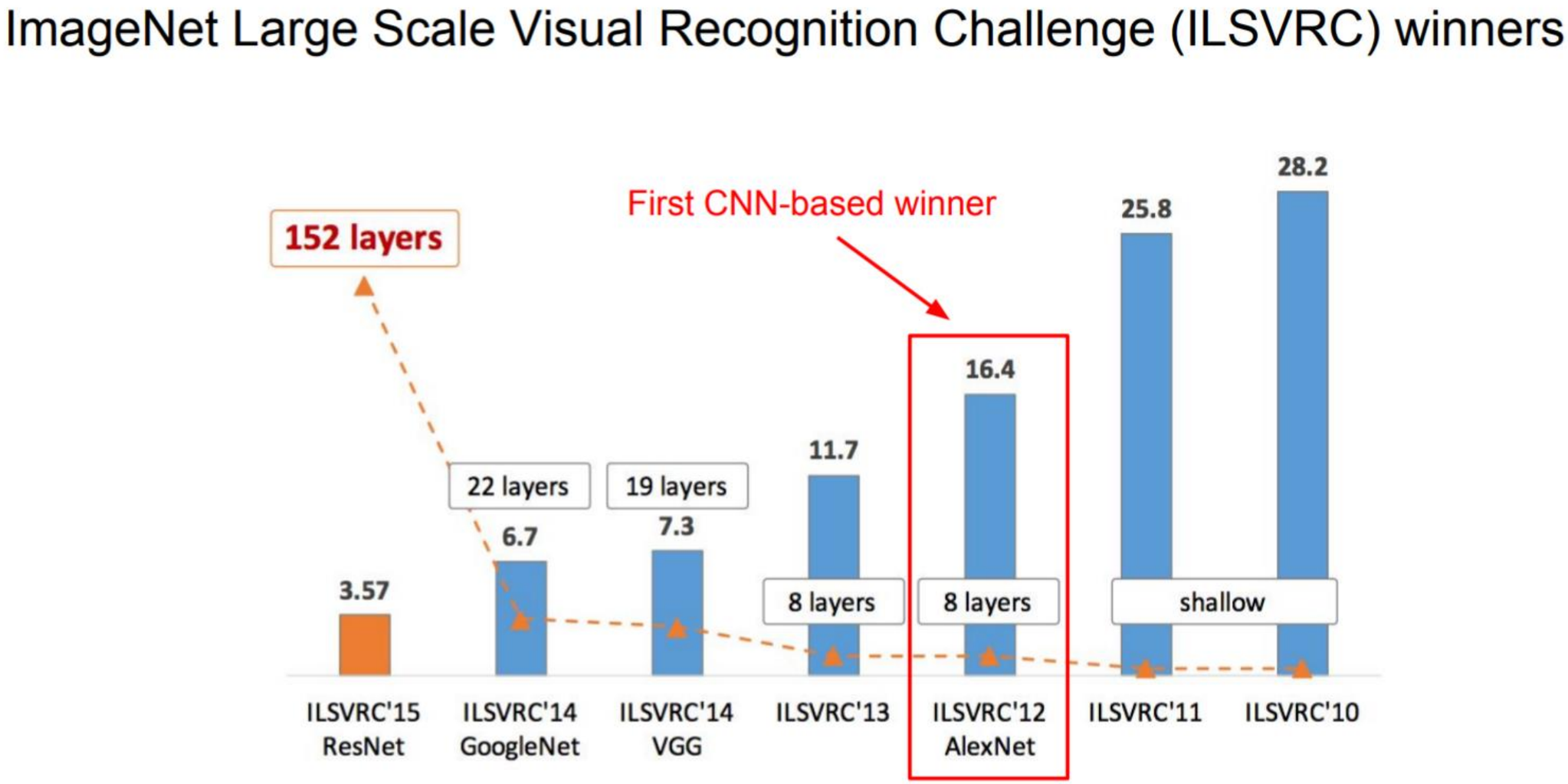


Trening CNN

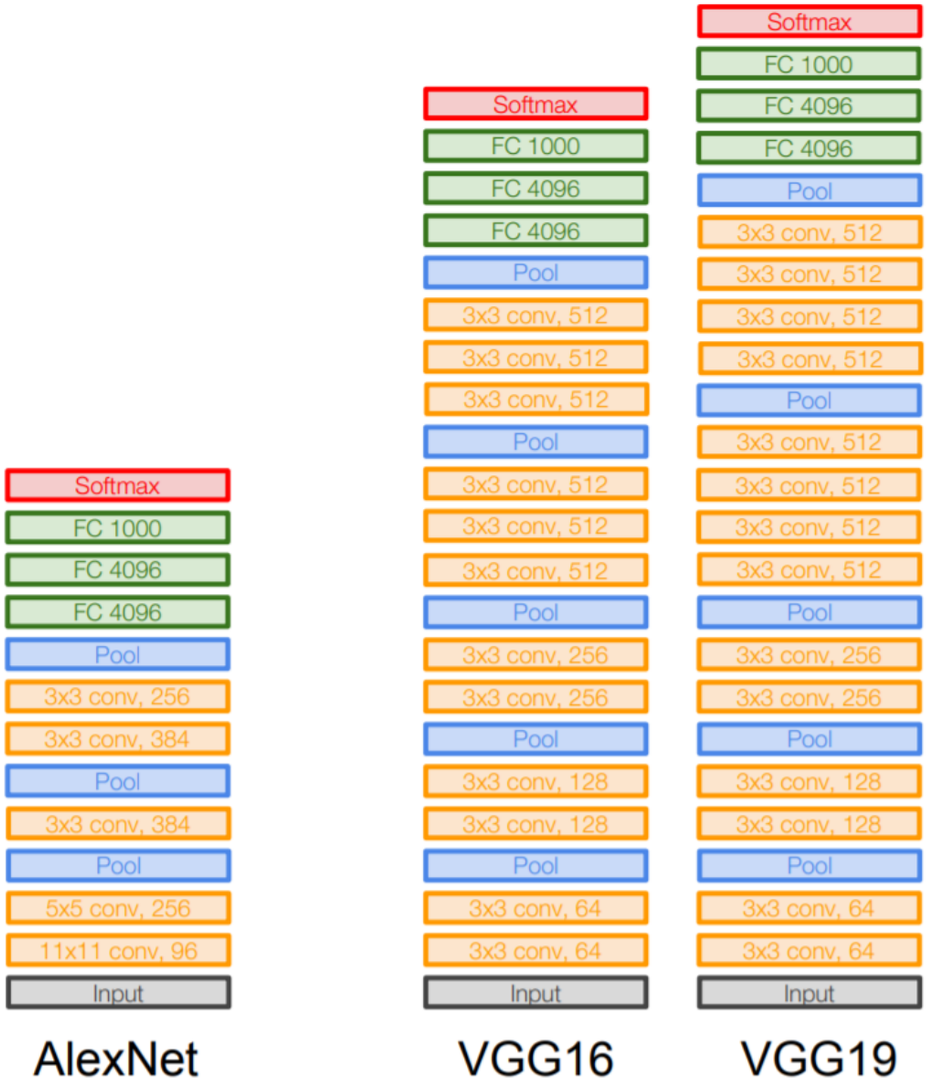
Wsteczna propagacja błędu (*backpropagation*)



Architektury CNN



Architektury CNN

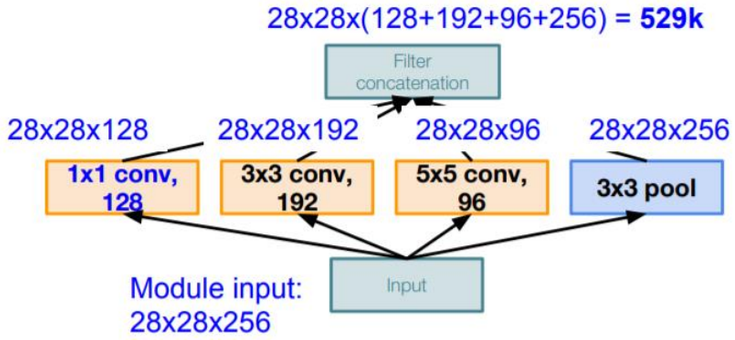


Case Study: GoogLeNet

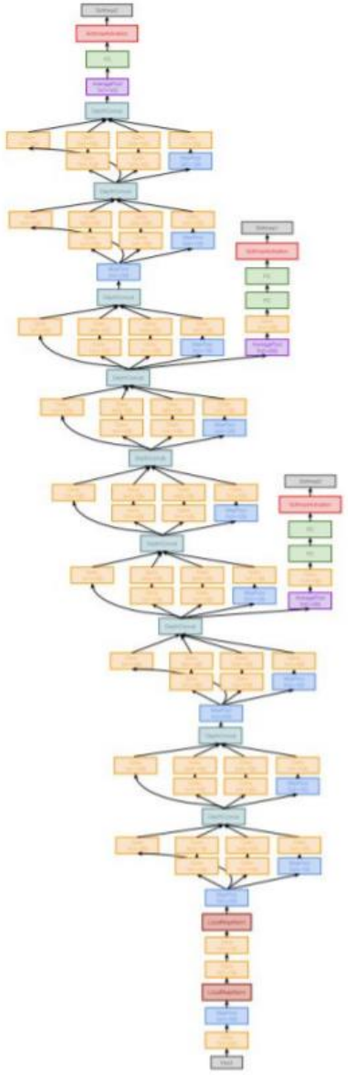
[Szegedy et al., 2014]

Deeper networks, with computational efficiency

- 22 layers
- Efficient “Inception” module
- No FC layers
- Only 5 million parameters!
  12x less than AlexNet
- ILSVRC’14 classification winner
  (6.7% top 5 error)



Inception module

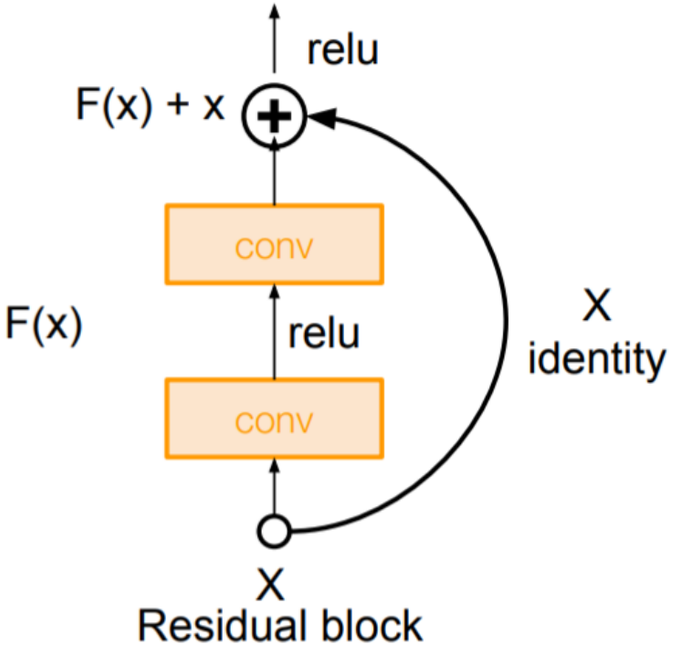


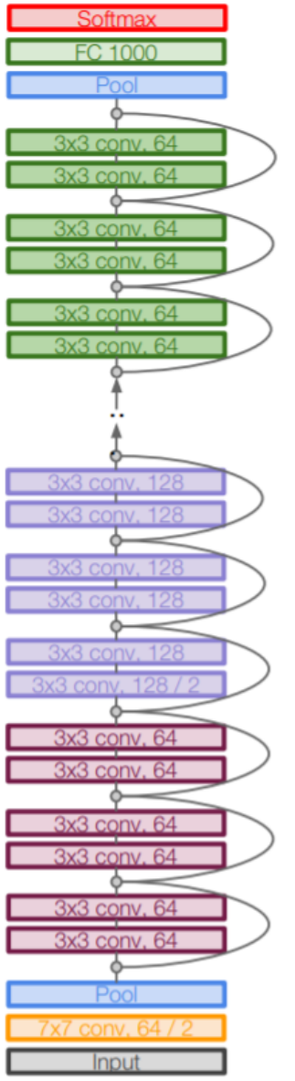
Case Study: ResNet

[He et al., 2015]

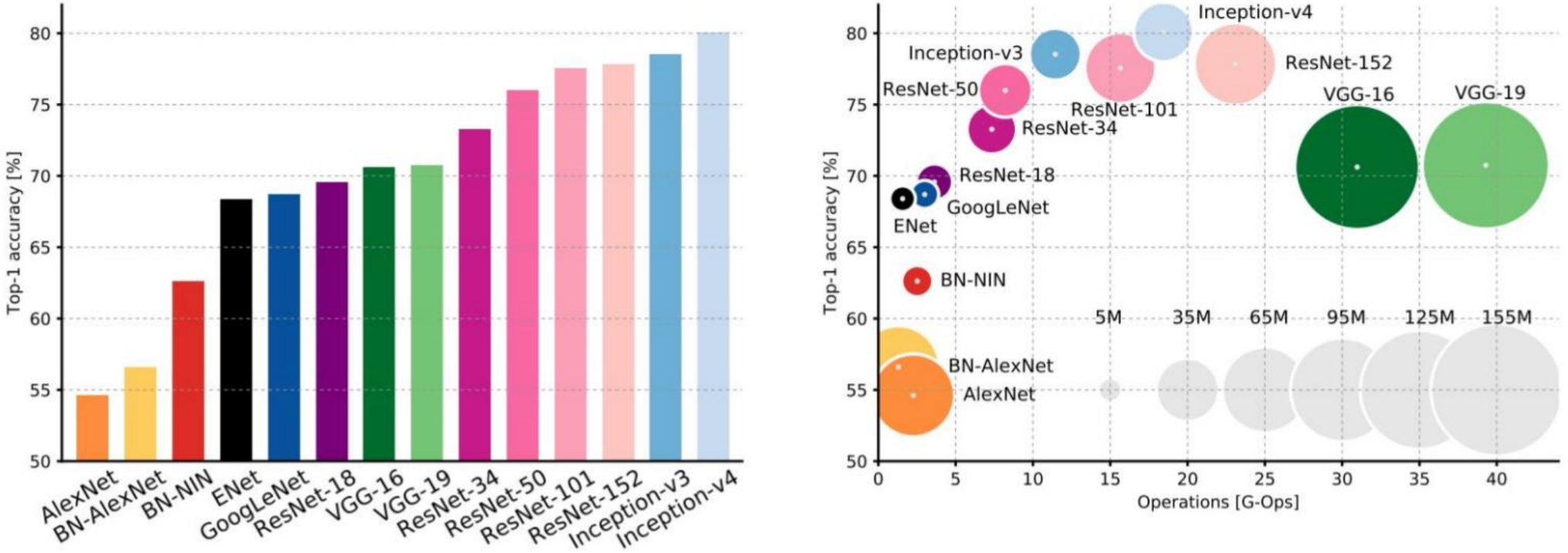
Very deep networks using residual connections

- 152-layer model for ImageNet
- ILSVRC'15 classification winner (3.57% top 5 error)
- Swept all classification and detection competitions in ILSVRC'15 and COCO'15!





Comparing complexity...



An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Podsumowanie

1. Konwolucyjne sieci neuronowe zostały zaprojektowane do rozpoznawania obrazów wizualnych.
2. CNN składa się z dwóch części: konwolucyjnej, zmieniającej reprezentacje obrazów i klasyfikacyjnej.
3. Podstawowymi blokami konstrukcyjnymi części konwolucyjnej są warstwy konwolucyjne (filtry) i warstwy redukujące (*pooling*), a podstawową funkcją aktywacji ReLU. W części klasyfikacyjnej stosuje się warstwy o pełnych połączeniach i funkcję aktywacji softmax.
4. CNN mają strukturę hierarchiczną, w kolejnych warstwach tworzą reprezentacje obrazów o różnych poziomy abstrakcji.
5. CNN uczą się optymalizować filtry, podczas gdy w tradycyjnych algorytmach filtry te tworzone są „ręcznie”.
6. CNN dzięki współdzielonym parametrom (neurony w jednej warstwie mają te same wagi), ograniczają liczbę parametrów w stosunku do sieci o pełnych połączeniach.
7. CNN uczone są metodami gradientowymi, funkcją straty jest entropia krzyżowa.
8. Podstawowe hiperparametry CNN to: rozmiar filtra, *padding*, krok przesunięcia filtra (*stride*), liczba filtrów, rodzaj i rozmiar *poolingu*, współczynnik uczenia.
9. CNN wykazują dużą odporność na przesunięcia obiektów na obrazach i wymagają stosunkowo niewielkiej obróbki wstępnej danych w porównaniu z innymi algorytmami klasyfikacji obrazów.
10. CNN mają szerokie zastosowania w rozpoznawaniu obrazów i wideo, systemach rekomendujących, segmentacji obrazów, analizie obrazów medycznych, przetwarzaniu języka naturalnego, interfejsach mózg-komputer i w analizie i predykcji szeregów czasowych.