CNN
Convolutional Neural Networks
Sieci splotowe

CNN

Convolutional Neural Networks (CNN, ConvNet) wariant MLP inspirowany biologicznie, gdzie mnożenie macierzy wag i sygnału wejściowego zastąpione jest operacją splotu

Zdolne do generalizacji sygnału posiadającego relacje przestrzenne

- 1D sygnały czasowe
- 2D obrazy
- 3D fMRI, video, obrazy RGB
- sygnały wielokanałowe

Potrafią być odporne na przestrzenne transformacje sygnału (skalowanie, obrót, przesunięcie)

Splot 1D

- **Splot** (*convolution*) funkcji $x(t)$ oraz $w(t)$

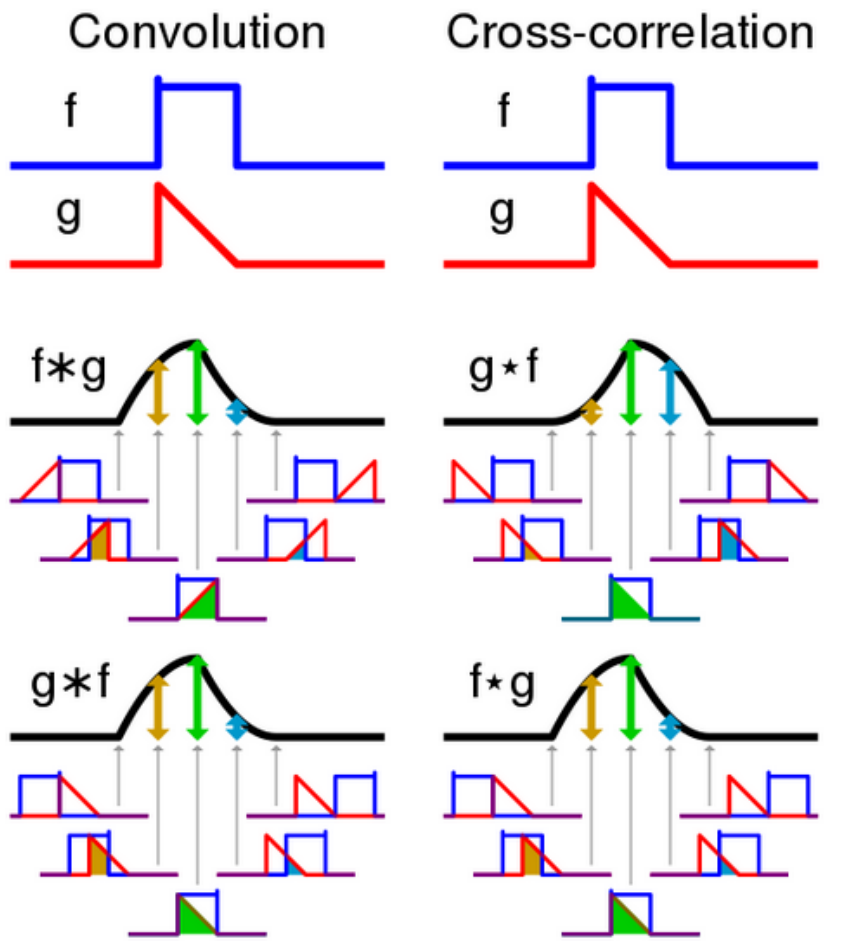
$$(x \star w)(t) = \int x(\tau) w(t - \tau) d\tau$$

- **Korelacja wzajemna** (*cross correlation*)

$$(x \star w)(t) = \int x(\tau + t) w(\tau) d\tau$$

- wynikiem jest funkcja, miara podobieństwa dwóch sygnałów jako funkcja przesunięcia jednego sygnału względem drugiego

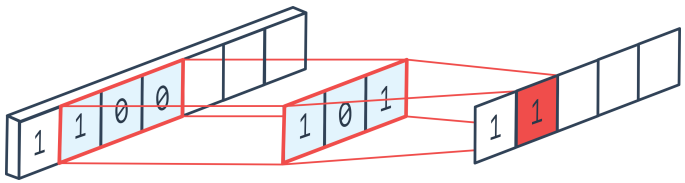
Splot i korelacja wzajemna



https://en.wikipedia.org/wiki/Cross-correlation

Splot 1D w sieciach splotowych

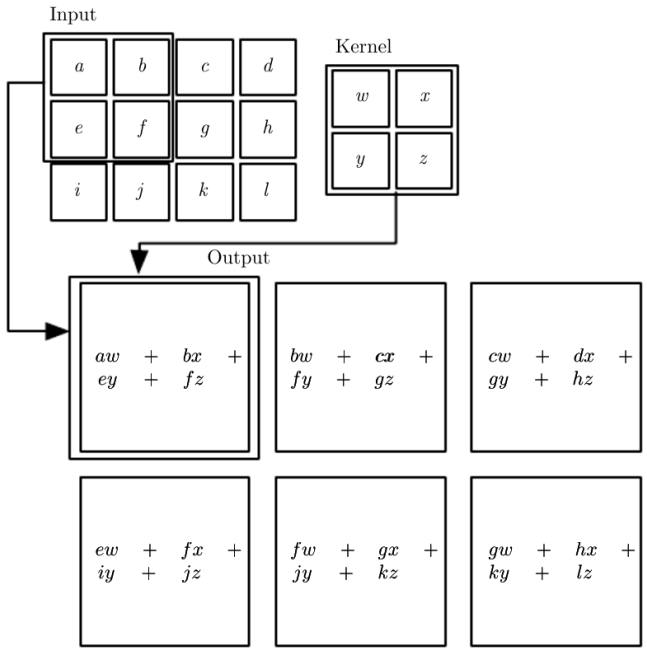
$$s(i) = \sum_m x(m+i)w(m)$$



- w sieciach neuronowych splotowych CNN mamy splot dyskretny
  x - sygnał wejściowy
  w - kernel (filtr), wagi połączeń neuronu, niezerowe tylko w ograniczonym obszarze (pole recepcyjne)
  $s(i)$ - sygnał wyjściowy, aktywacja neuronu i
- wartości wyjściowe $s(i)$ tworzą **mapę cech** (*feature map*)

Splot 2D

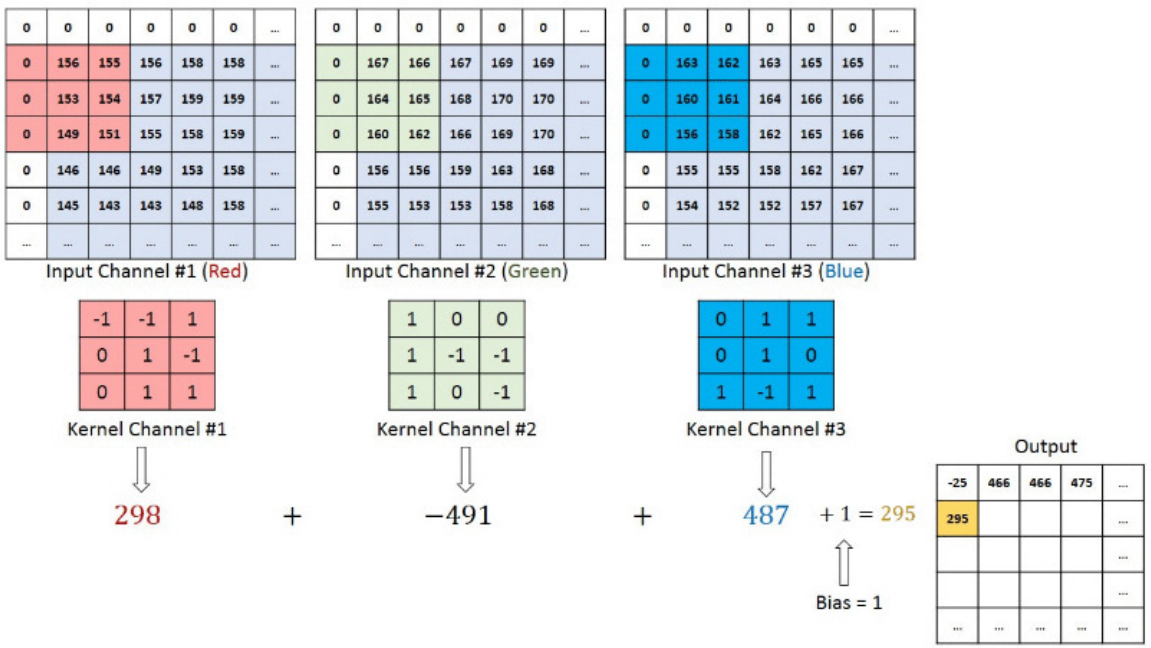
- wejście 2D, np. obraz w skali szarości WxH (jeden kanał)
- filtr 2D (kernel) o mniejszych wymiarach niż wejście (2x2) + ewentualny wyraz wolny (bias)
- wynikiem jest mapa cech 2D, gdzie każdy punkt jest sumą ważoną wartości z obszaru wejściowego (pole recepcyjne) i wag filtra



$$s(i,j) = \sum_m \sum_n w(m,n) \cdot x(i+m, j+n)$$

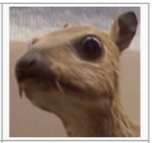
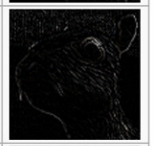
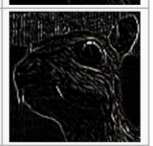
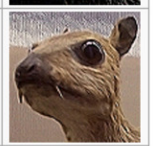
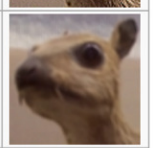
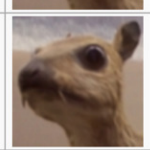
Splot 2D obrazu RGB

- wejście 3D, np. obraz RGB o wymiarach $W \times H \times 3$
- filtr 3D o wymiarach $k \times k \times 3$, gdzie $k \times k$ to rozmiar pola recepcyjnego, a 3 to liczba kanałów (RGB) + ewentualny wyraz wolny (bias)
- wyjście: mapa cech o wymiarach $(W - k + 1) \times (H - k + 1)$



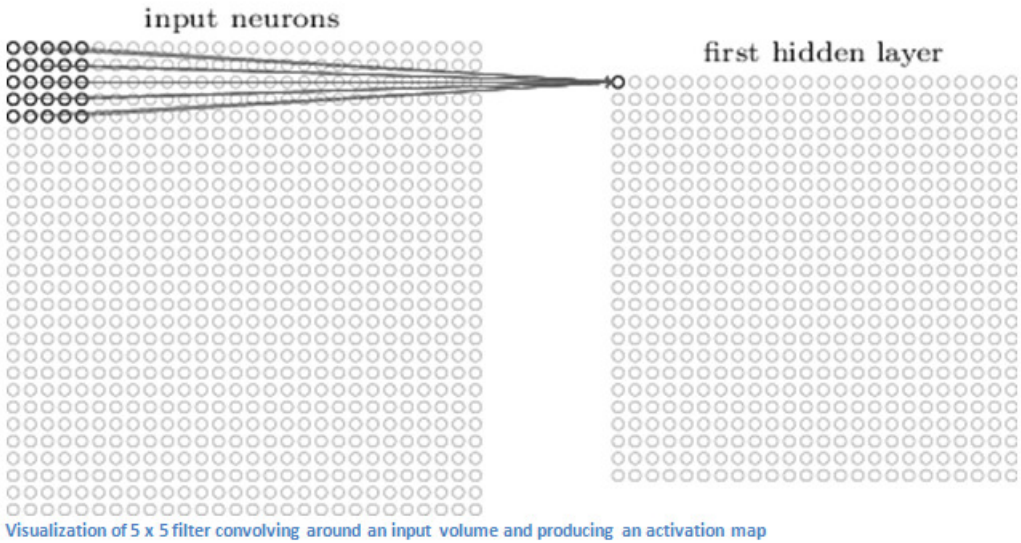
$$s(i,j) = \sum_m \sum_n \sum_c w(m,n,c) \cdot x(i+m, j+n, c) + bias$$

Filtry graficzne 2D - przykłady

Operation	Filter	Convolved Image
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	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

Pola recepcyjne i mapy cech



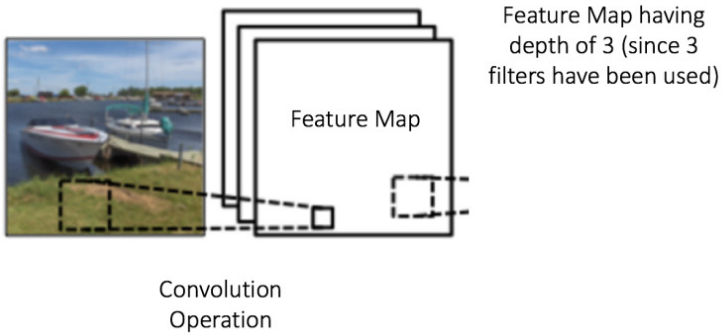
Visualization of 5 x 5 filter convolving around an input volume and producing an activation map

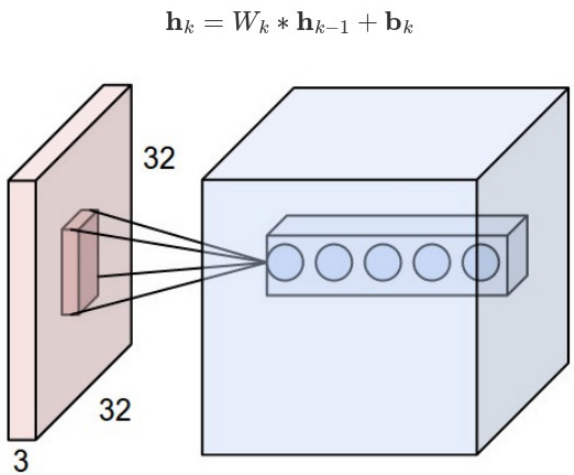
- **Pole recepcyjne** - obszar „widziany” przez neuron (rozmiar filtra)
- wyjście neuronu: splot sygnału i liniowego filtra, ewentualny wyraz wolny (bias) i nieliniowa funkcja wyjściowa (ReLU, tanh), generują **mapę cech** (*feature map)*

$$s(i,j) = \sigma\left(b + \sum_m \sum_n \sum_k w(m,n,k) \cdot x(i+m, j+n, k)\right)$$

Pola recepcyjne i mapy cech

- N neuronów (filtrów) tworzy na wyjściu N map cech, tworząc tensor o wymiarach $M_x \times M_y \times N$, gdzie M_x, M_y - rozmiary mapy wyjściowej, N - liczba filtrów (kanałów wyjściowych)





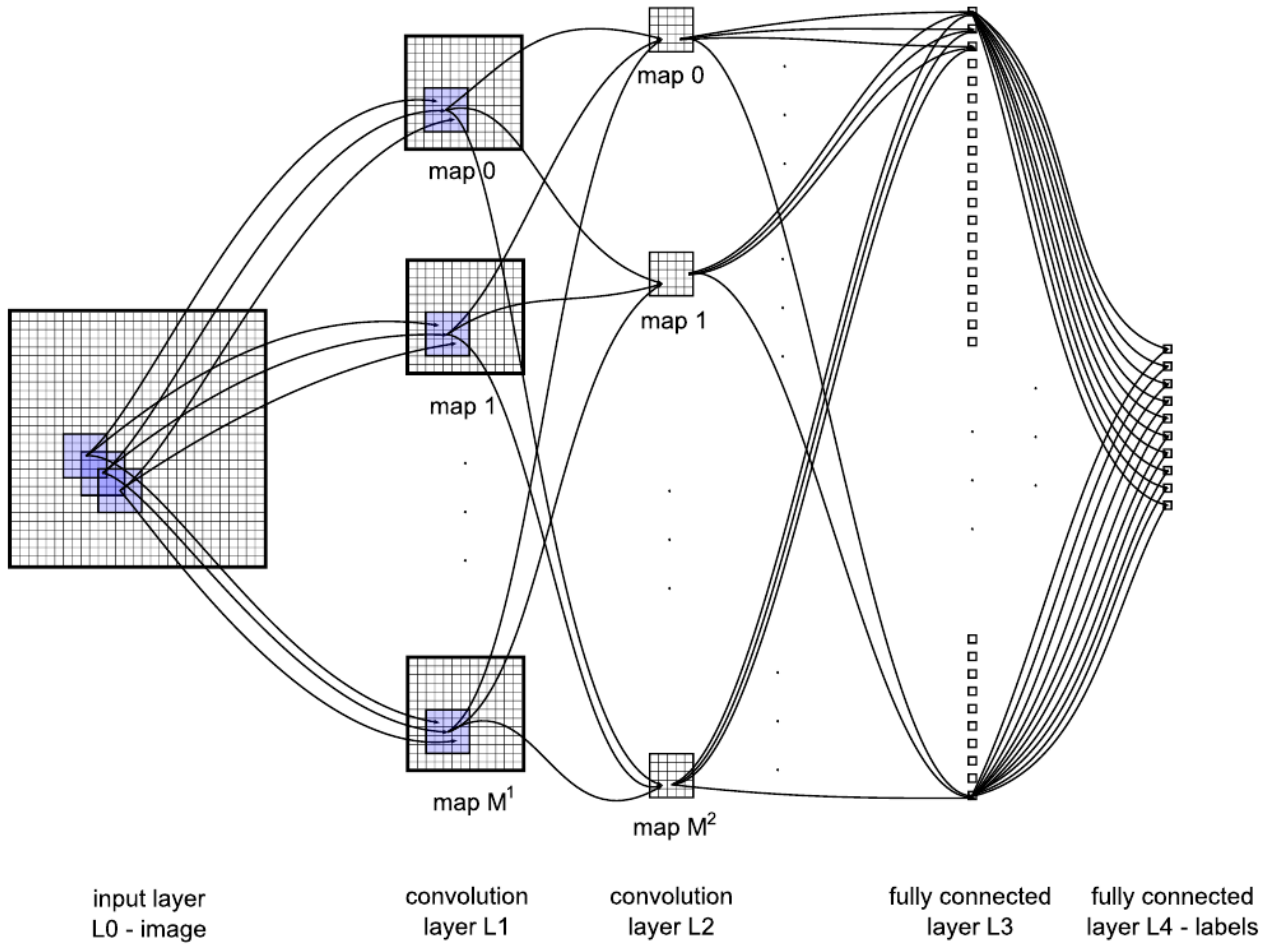
- Przy przetwarzaniu obrazów sygnał wejściowy x to tensor 3D: szerokość, wysokość, kanał (RGB).

$$s(i,j,l) = \sum_{mnk} x(i+m, j+n, k) w(m,n,k,l)$$

gdzie l to indeks filtra (kanału wyjściowego)

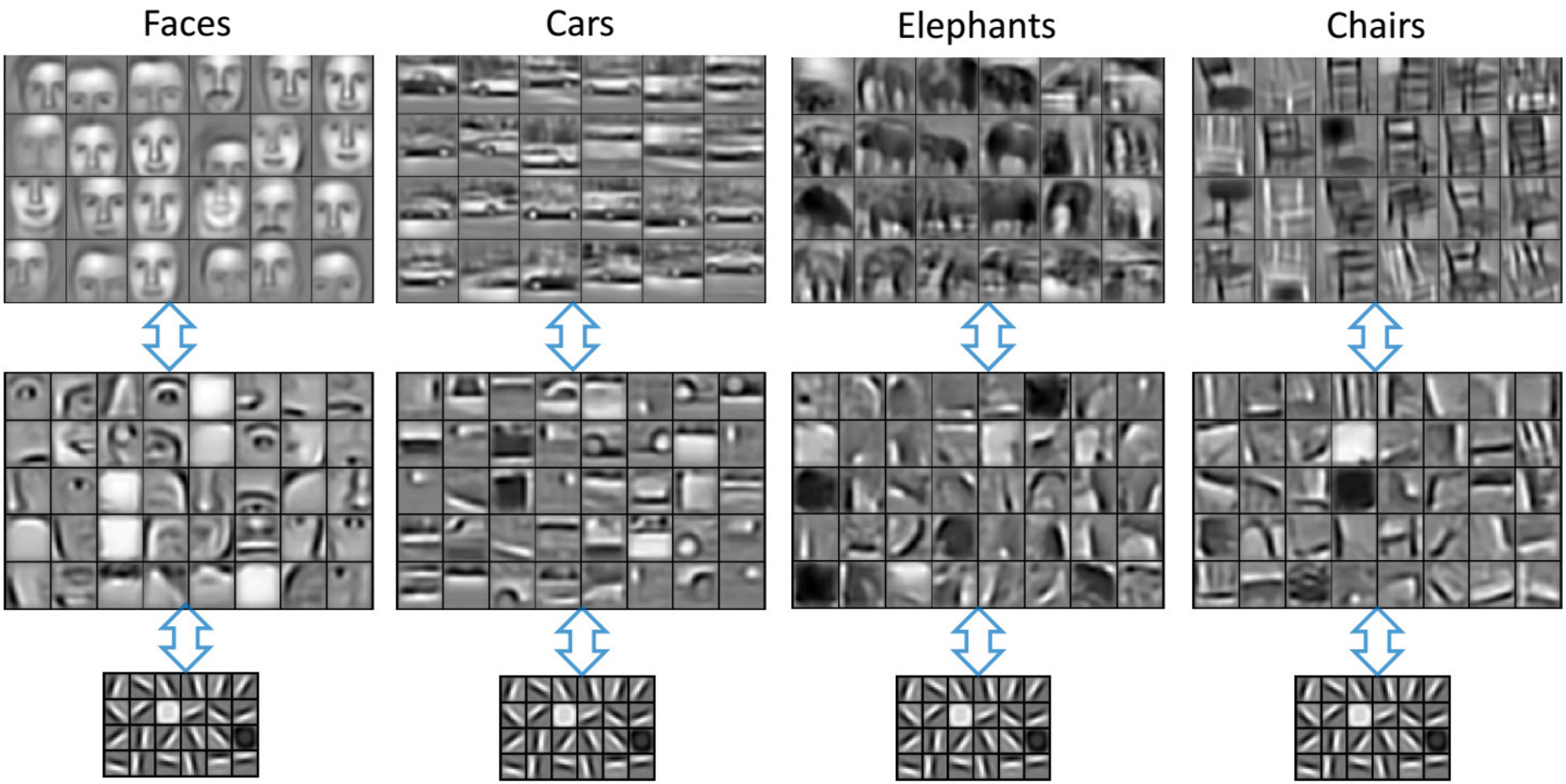
- w praktyce x rozszerzony do 4D o wymiar związany z rozmiarem mini-batcha

Typowa architektura CNN



☞ An Interactive Node-Link Visualization of Convolutional Neural Networks by Adam W. Harley

Przykłady filtrów



Goodfellow, 2016 [?]

Rozmiar mapy cech w splocie 2D

Rozmiar mapy wyjściowej (M_x^n, M_y^n) w warstwie n po wykonaniu splotu 2D z przesunięciem (S_x^n, S_y^n) oraz uzupełnieniem brzegów sygnału wejściowego o P_x^n, P_y^n (zero padding) wynosi

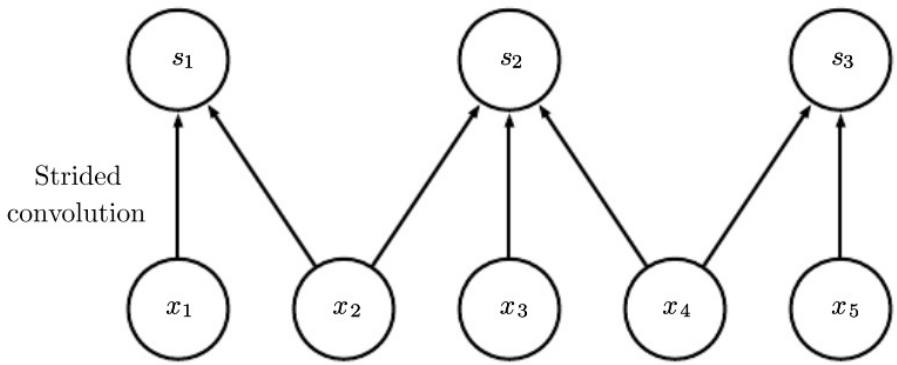
$$M_x^n = \frac{M_x^{n-1} - K_x^n + 2P_x^n}{S_x^n} + 1 \qquad M_y^n = \frac{M_y^{n-1} - K_y^n + 2P_y^n}{S_y^n} + 1$$

gdzie
M_x^{n-1}, M_y^{n-1} to rozmiar mapy cech warstwy $n-1$ (sygnał z warstwy poprzedniej),
K_x, K_y to rozmiar filtra (szerokość i wysokość)
S_x, S_y to przesunięcie (*stride*) wzdłuż osi x i y,
P_x, P_y wielkość rozszerzenia brzegu sygnału wejściowego (*zero padding*)
Splot 2D z przesunięciem

$$s(i,j,l) = \sum_{mnk} x(i \cdot S_x + m, j \cdot S_y + n, k) w(m,n,k,l)$$

Stride - krok przesunięcia

Przykład w 1D: S=2, K=3, P=1

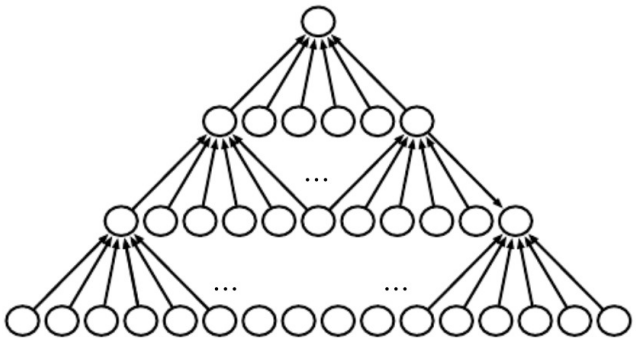


$$M^n = \frac{M^{n-1} - 3 + 2}{2} + 1$$

- redukcja wymiaru - zmniejszenie wymogów obliczeniowych i pamięciowych
- zmniejszenie rozdzielczości sygnału
- kosztem mniej dokładnej reprezentacji

Zero padding

valid zero padding: brak rozszerzenia na brzegach ($P = 0$)



$$M^n = \frac{M^{n-1} - K^n}{S^n} + 1$$

- rozmiar map maleje w kolejnych warstwach
- ogranicza to możliwość budowania głębokich sieci i wymusza stosowanie małych filtrów
- sygnał wejściowy na brzegach mapy jest słabiej reprezentowany w sygnale wyjściowym (mniej aktywacji neuronów w pobliżu brzegów)

Zero padding

same zero padding: brzegi wypełnione dodatkowymi wartościami (zerami) aby zapewniać odtworzenie wymiaru sygnału wejściowego $P = \lfloor \frac{K}{2} \rfloor, S = 1$



- pozwala budować bardzo głębokie sieci o dowolnych wielkościach filtrów $M_i^n = \lceil \frac{M_i^{n-1}}{S_i} \rceil$

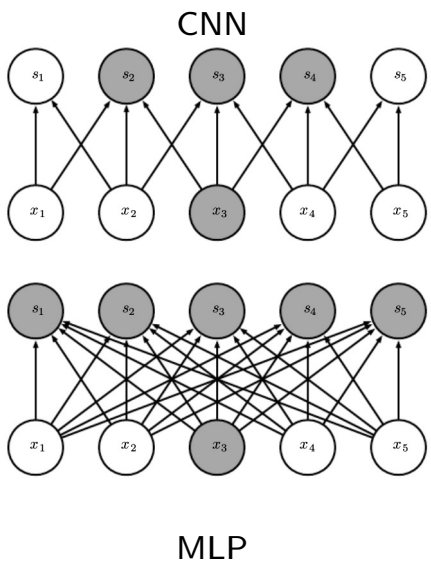
Właściwości CNN

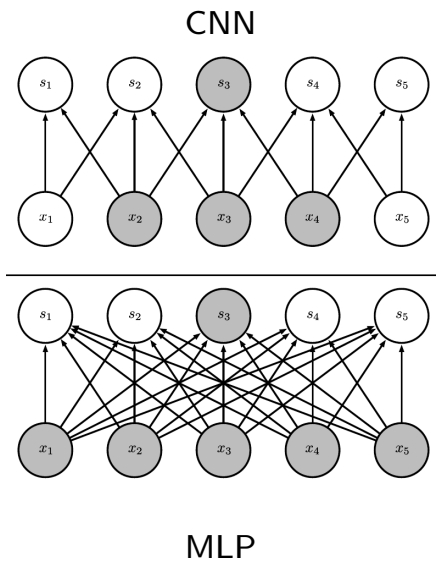
- **rzadka reprezentacja** - rozmiar filtra jest dużo mniejszy od rozmiaru sygnału wejściowego, pojedyncze wejście oddziałuje tylko na grupę neuronów w zasięgu pola recepcyjnego
- **współdzielenie parametrów** - warstwa splotowa może być widziana jako w pełni połączona warstwa ze współdzielonymi wagami (dużo mniejsza liczba parametrów w stosunku do MLP)
- **równoważność względem przesunięcia** sygnału - ta sama cecha znajdująca się w różnych miejscach obrazu będzie aktywowała ten sam filtr
- możliwość zastosowania sygnału wejściowego o nieustalonym rozmiarze - większy obraz wejściowy wygeneruje większe mapy. W przypadku MLP zwiększenie rozmiaru wektora wejściowego wymaga rozbudowy architektury.

Rzadka reprezentacja

pojedyncze wejście aktywuje tylko grupę neuronów w małym obszarze wyjściowej mapy cech, podczas gdy w MLP każdy neuron jest aktywowany przez całe wejście

wyjście neuronu zależy od małego obszaru sygnału wejściowego (pole recepcyjne), podczas gdy w MLP każdy neuron widzi całe wejście





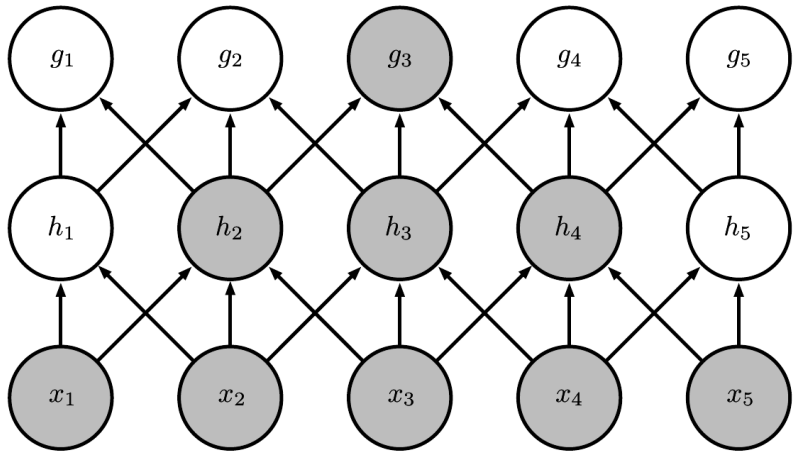
MLP: mnożenie macierzy $O(m \times n)$ połączeń

CNN: warstwa splotowa $O(k \times n)$ połączeń, gdzie rozmiar filtra $k \ll m$

Źródło grafiki: Goodfellow, 2016 [?]

Efektywne pole recepcyjne

- **Efektywne pole recepcyjne** - obszar sygnału wejściowego rejestrowany przez neurony w wyższych warstwach rośnie z głębokością

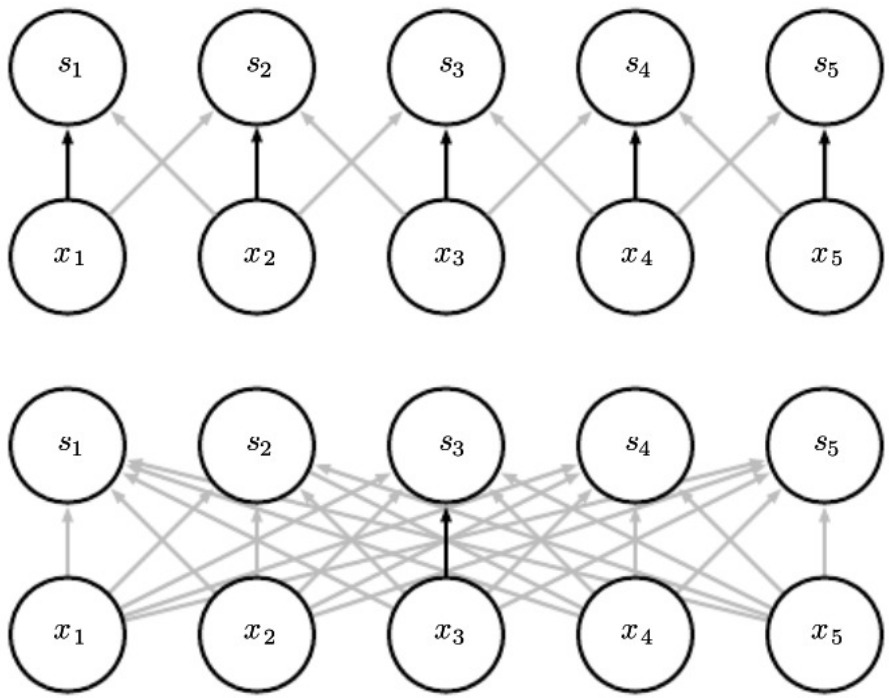


- stride, pooling i dilation (rozrzedzony splot) dodatkowo zwiększają efektywne pole recepcyjne
- pomimo rzadkich połączeń i ograniczonego zasięgu pól recepcyjnych głęboka sieć jest w stanie modelować złożone (odległe) zależności

Współdzielenie wag

- ta sama waga jest używana przy przetwarzaniu każdego punktu wejściowego, gdzie w MLP każda waga jest unikalna dla każdego połączenia
- pojedynczy filtr pozwala wykryć tę samą cechę w różnych położeniach obrazu wejściowego (*equivariance to translation*)

CNN

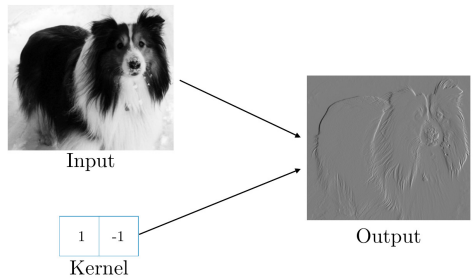


Źródło grafiki: Goodfellow, 2016 [?]

MLP

Rzadkie połączenia i współdzielenie wag

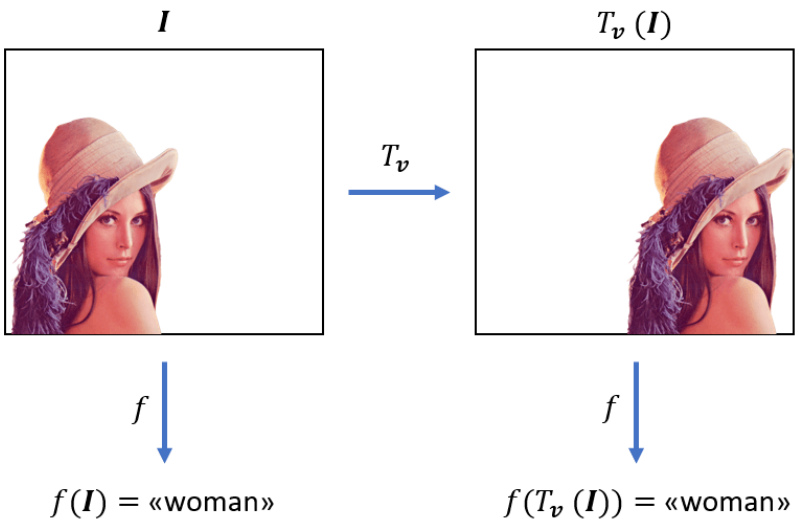
Przykład: wykrywanie pionowych krawędzi obrazu $n \times n$



- MLP: mnożenie pełnej macierzy $n^2 \times n^2 = n^4$ wag, $n^2 \times (n^2 + 1)$ operacji
- CNN: splot filtrem 1x2 (2 wagi) wymaga $n^2 \times 3$ operacji (2 mnożenia i dodawanie)
- warstwa splotowa jest istotnie wydajniejsza w mapowaniu powtarzających się relacji w małych, lokalnych rejonach sygnału wejściowego

Źródło grafiki: Goodfellow, 2016 [?]

Równoważność przy przesunięciu



Przesunięcie sygnału wejściowego $\mathbf{x}$ o wektor $\mathbf{v}$

$$T_{\mathbf{v}}(\mathbf{x})(\mathbf{u}) = \mathbf{x}(\mathbf{u} - \mathbf{v})$$

powoduje identyczne przesunięcie sygnału na mapie cech

$$f(T_{\mathbf{v}}(\mathbf{x})) = T_{\mathbf{v}}(f(\mathbf{x}))$$

Aktywowane są te same filtry wykrywające obiekt.
Równoważność translacji jest pożądaną własnością w rozpoznawaniu obrazów, gdzie lokalne cechy (np. krawędzie) mogą wystąpić w wielu miejscach obrazu, a ich położenie nie jest istotne dla klasyfikacji.
W przypadku sieci MLP przesunięcie sygnału wejściowego powoduje zmianę aktywacji wszystkich neuronów w sieci

Rys. https://www.baeldung.com/cs/translation-invariance-equivariance

Równoważność przy przesunięciu

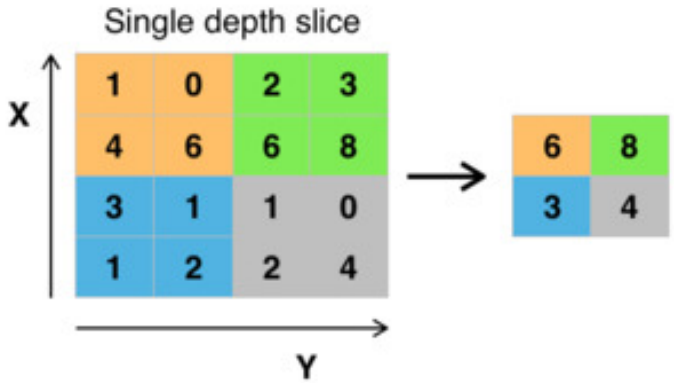
- współdzielenie wag dla całego wejścia nie zawsze jest pożądane, można zastosować różne filtry w różnych obszarach obrazu, np. rozpoznawanie twarzy ze zdjęć paszportowych, które posiadają charakterystyczne cechy występujące wyłącznie w określonych rejonach sygnału wejściowego
- Sieci CNN (operacje splotu) **nie są niezmiennicze** względem zmiany **skali, obrotu oraz przekształceń afinicznych obrazu**. Taką własność osiąga się poprzez rozszerzenie zbioru treningowego o przypadki zdeformowane (szum, skalowanie, obrót, zmiana kontrastu itd.)

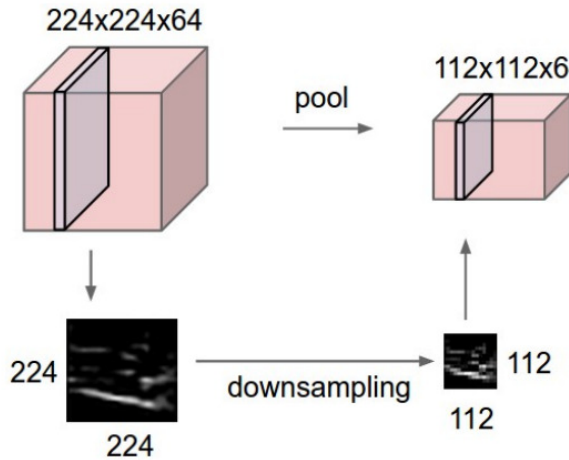
Pooling

W skład typowego bloku realizującego warstwę splotową wchodzi:

- liniowa aktywacja (splot)
- detekcja (nieliniowość np. ReLU)
- redukcja (pooling) - „uogólnienie” wartości sąsiadujących wyjść

Max pooling - maksimum z pewnego podobszaru



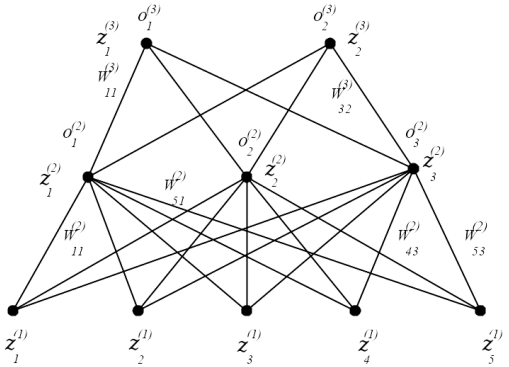


- redukcja wymiarowości - przesunięcie filtra S typowo równe jego wielkości K (maksimum z rozłącznych obszarów)
- inne podejścia: *avg. pooling* (średnia z sąsiedztwa), *norm pooling* (norma z sąsiadujących wyjść), ważona średnia od centrum, niektóre architektury rezygnują z tej warstwy

Wsteczna propagacja w MLP (przypomnienie)

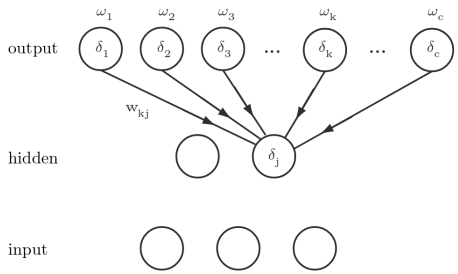
Sygnał propagowany od wejścia do wyjścia

$$o_i^h = f(z_i^h) \qquad z_i = \sum_j w_{ji}^h o_j^{h-1} + b_i^h$$



Sygnał błędu neuronów δ_i^k propagowany od warstwy wyjściowej do wejścia

$$\delta_i^h = \frac{\partial E}{\partial z_i^h} = f'(z_i^h) \sum_j \delta_j^{h+1} w_{ij}^{h+1}$$



Aktualizacja wag w kierunku spadku gradientu

$$\Delta w_{ij}^h = -\eta \frac{\partial E}{\partial w_{ij}^h} = \eta \delta_j^h o_i^{h-1}$$

Wsteczna propagacja błędu w sieciach CNN

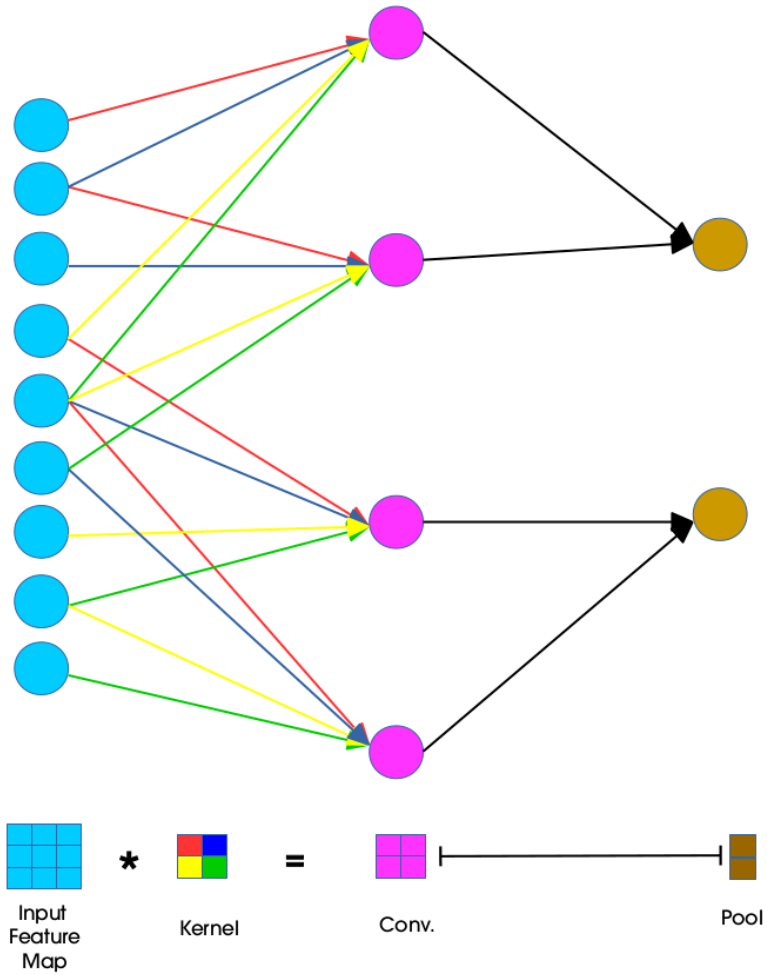
Sygnał propagowany od wejścia do wyjścia (dla jednego kanału $H \times W$ i filtra $k_1 \times k_2$)

$$o_{ij}^h = f(z_{ij}^h)$$

$$z_{ij} = \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} w_{mn}^h o_{i+m,j+n}^{h-1} + b^h$$

oznaczmy $*$ operację splotu

$$o_{ij}^h = f\left((\mathbf{w}^h * \mathbf{o}^{h-1})_{ij} + b^h \right)$$



Źródło
https://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/

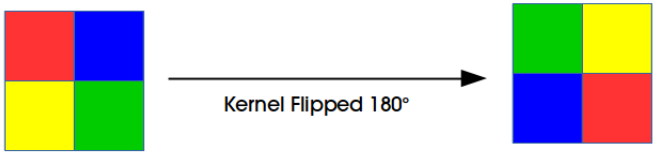
Wsteczna propagacja błędu w sieciach CNN

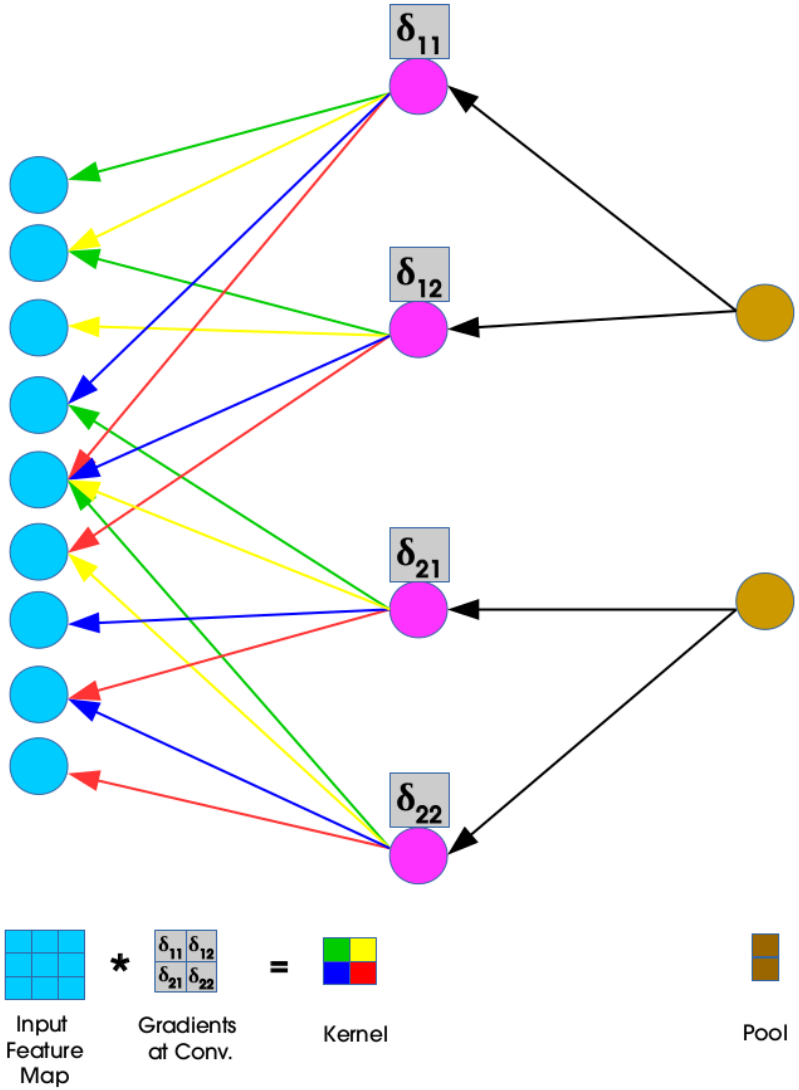
Sygnał błędu neuronów δ_i^k od warstwy wyjściowej do wejścia

$$\delta_{ij}^h = \frac{\partial E}{\partial z_{ij}^h} =$$

$$= f'(z_{ij}^h) \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} \delta_{i-m,j-n}^{h+1} w_{mn}^{h+1}$$

$$= f'(z_{ij}^h) \left(\delta^{h+1} * rot_{180^\circ}(\mathbf{w}^{h+1}) \right)_{ij}$$





Źródło
https://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/

Wsteczna propagacja błędu w sieciach CNN

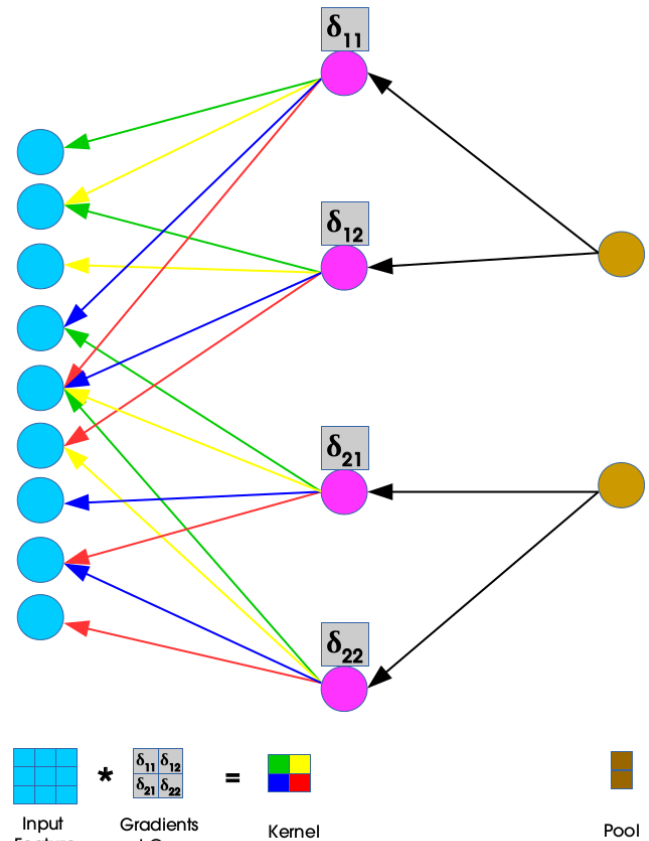
Splot mapy wejściowej o wymiarze $H \times W$ i filtra o wymiarze $k_1 \times k_2$ tworzy mapę wyjściową o wymiarze $(H - k_1 + 1) \times (W - k_2 + 1)$.

Aktualizacja wag w kierunku spadku gradientu:

$$\Delta w_{ij}^h = -\eta \frac{\partial E}{\partial w_{ij}^h} =$$

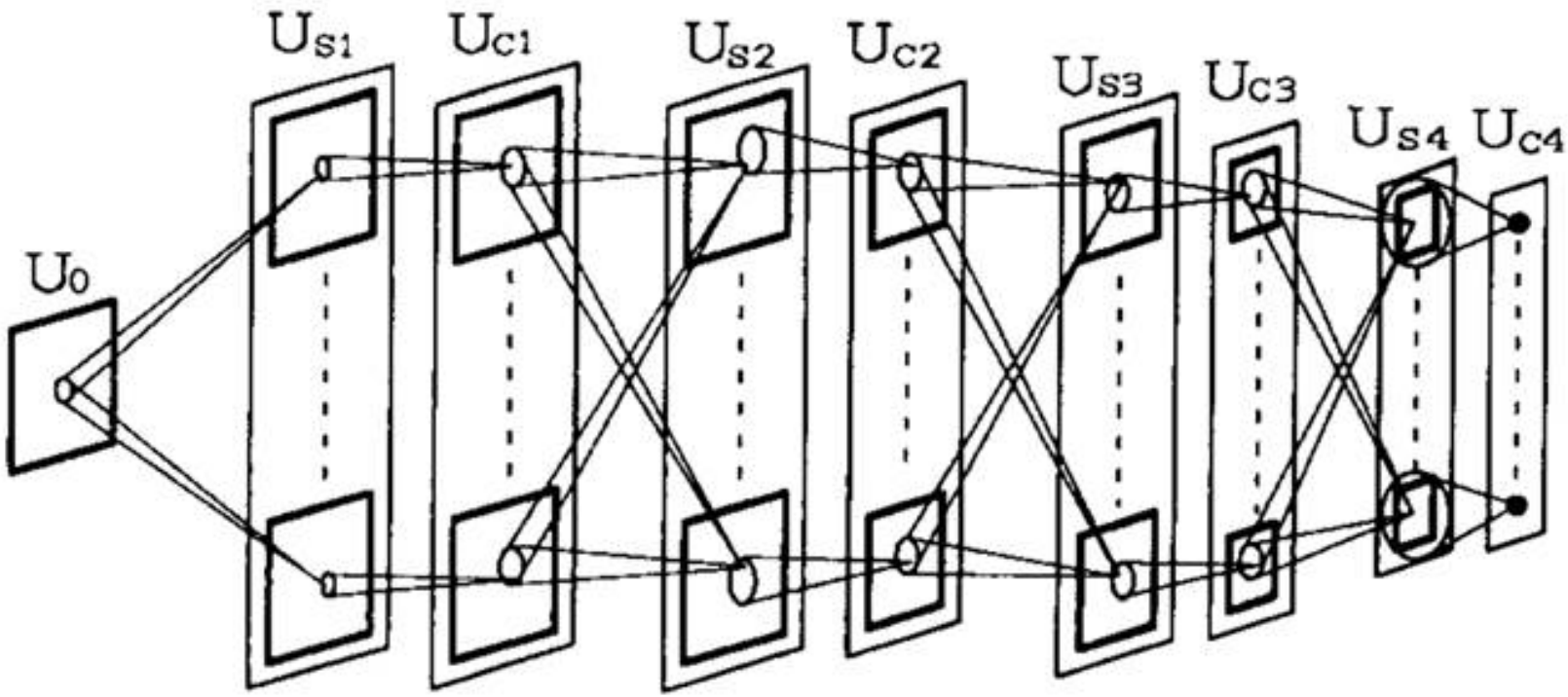
$$= \eta \sum_{m=0}^{H-k_1} \sum_{n=0}^{W-k_2} \delta_{mn}^h o_{m+i,n+j}^{h-1} =$$

$$= \eta (\delta^h * \mathbf{o}^{h-1})_{ij}$$



Źródło
https://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/

Neocognitron (Fukushima, 1980)



- pierwsza sieć CNN do rozpoznawania ręcznie pisanych cyfr
- architektura inspirowana biologicznie ludzkim układem przetwarzania obrazu w mózgu
- uczenie nienadzorowane (bez wstecznej propagacji błędu)

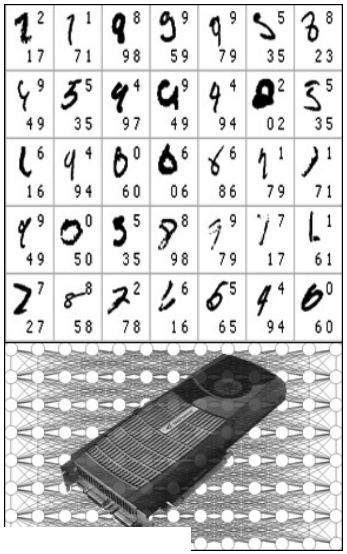
LeNet5 (LeCun 1998)

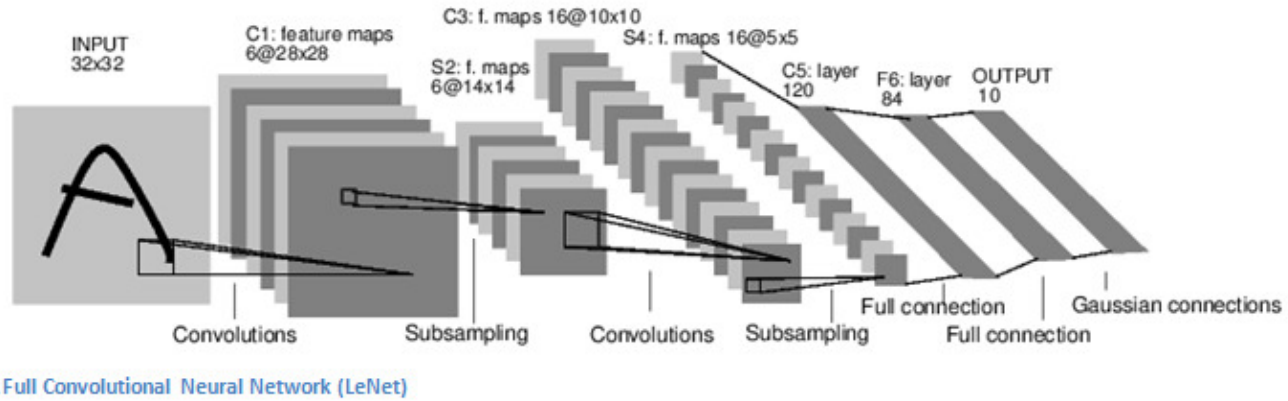
klasyfikacja cyfr pisanych ręcznie
(odczytywanie czeków)

☞ MNIST

trening: 60k cyfr, 250 osób,

test: 10k cyfr



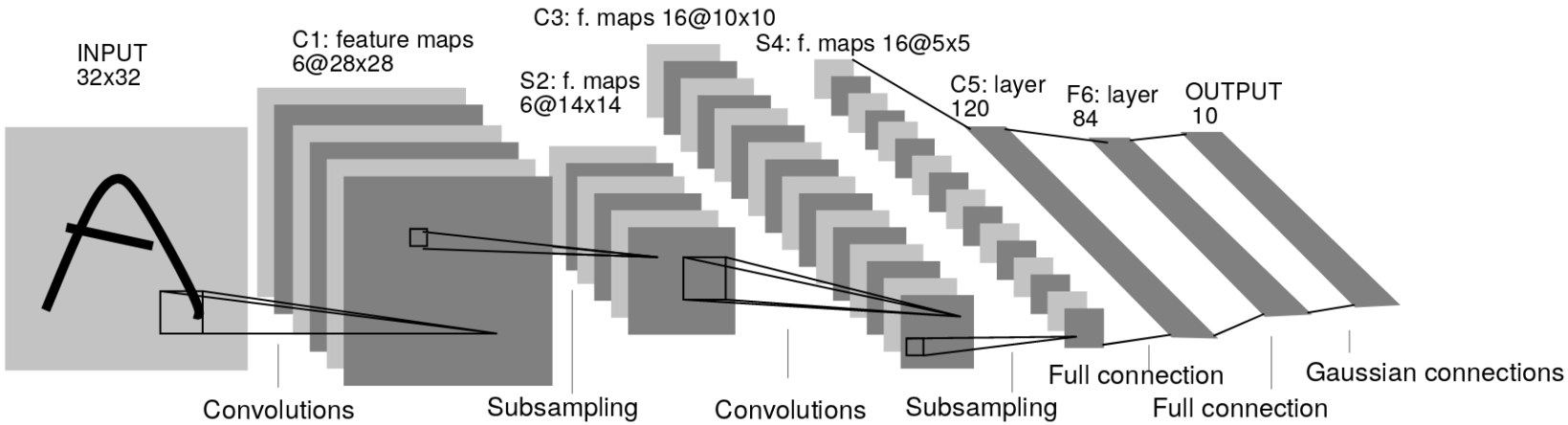


A Full Convolutional Neural Network (LeNet)

☞ LeNet-5, convolutional neural networks

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, november 1998

LeNet-5



Layer	kernels	size	output	connections	parameters
Input			1x32x32		
C1	6	5x5	6x28x28	122304	156
S2		2x2	6x14x14	5880	12
C3	16	5x5	16x10x10	151600	1516
S4		2x2	16x5x5	2000	32
C5	120	5x5	120x1x1	48120	48120
F6	84		84x1x1	10164	10164
Output	10		10x1x1		

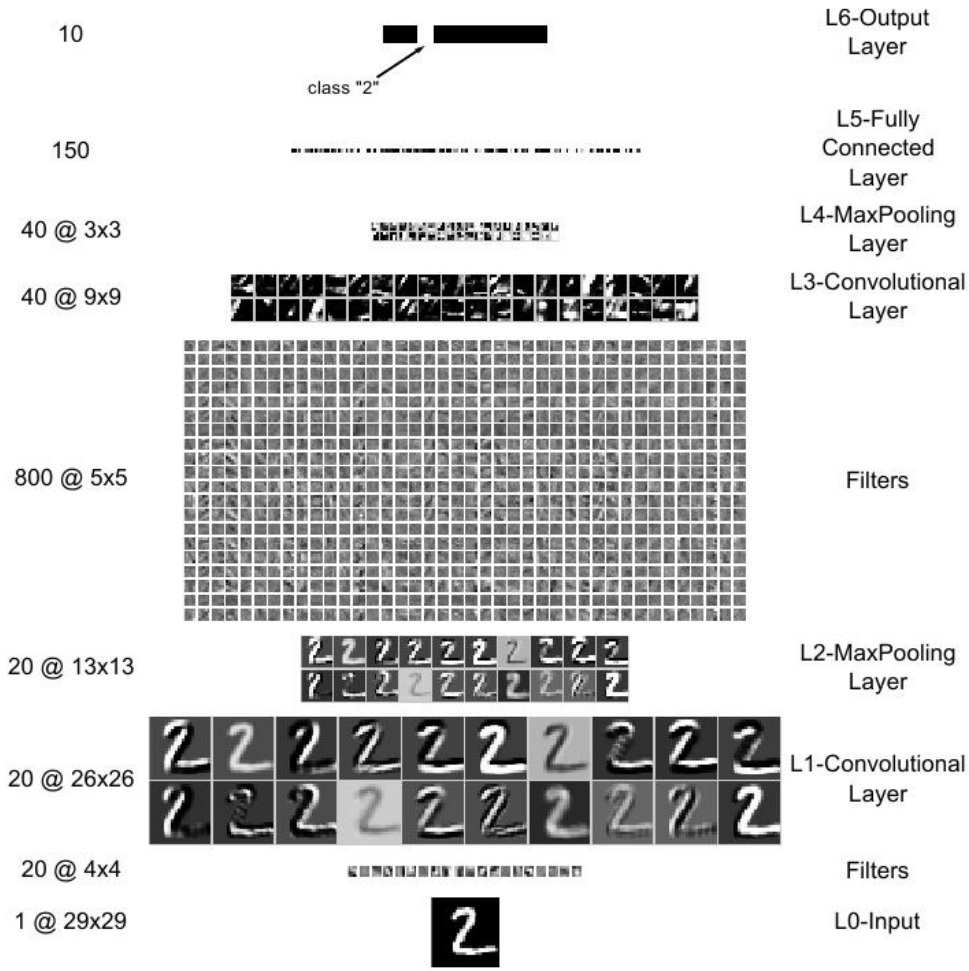
LeNet-5 połączenia S2-C3

Mapy C3 połączone wyłącznie z wybranymi mapami z S2

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

- redukcja liczby połączeń
- przełamanie symetrii sieci - różne sygnały wejściowe pozwalają uzyskać różnorodne (być może komplementarne) zestawy detektorów cech

Przykład wizualizacji aktywacji sieci

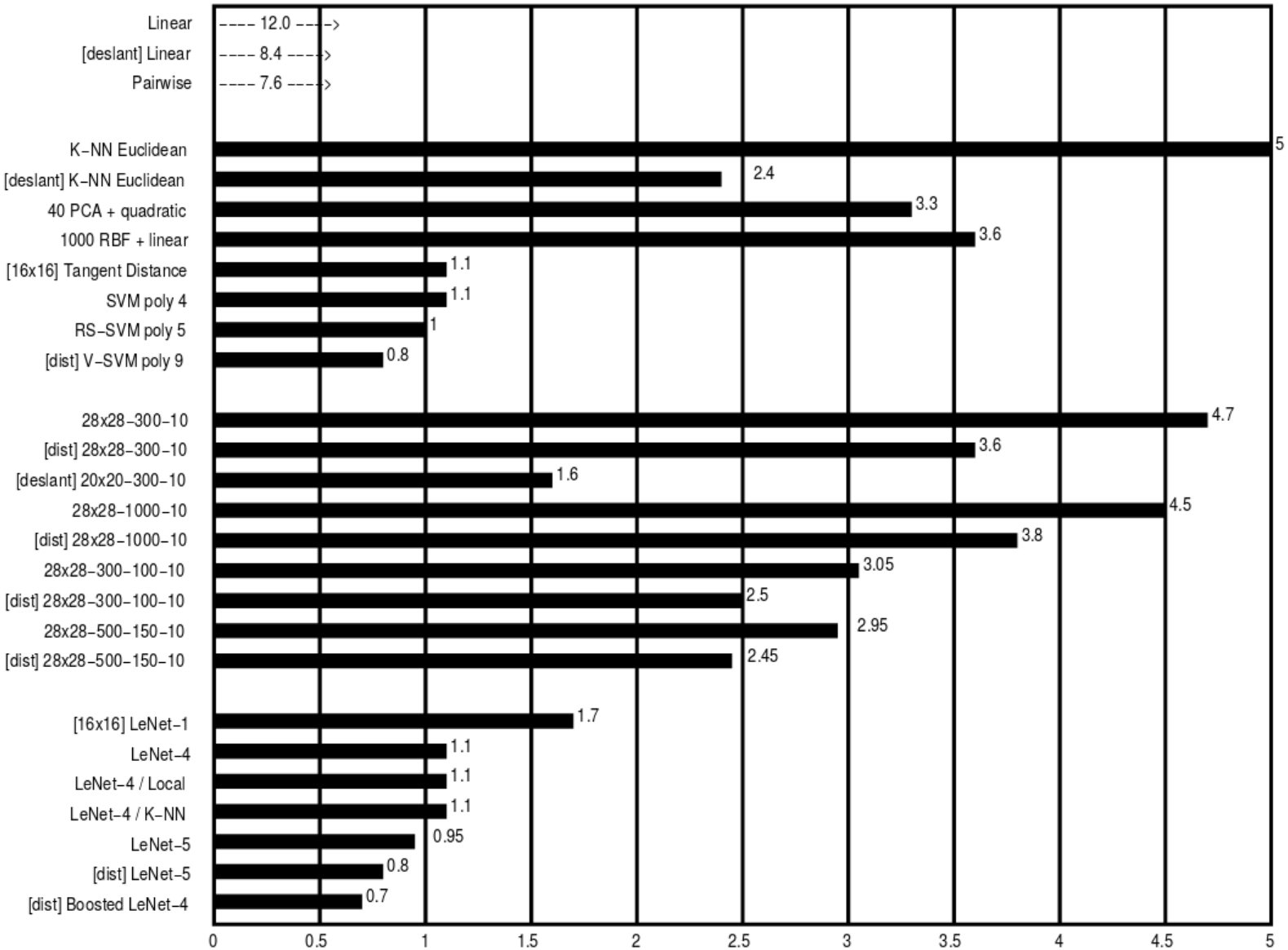


☞ Wizualizacja sieci splotowej

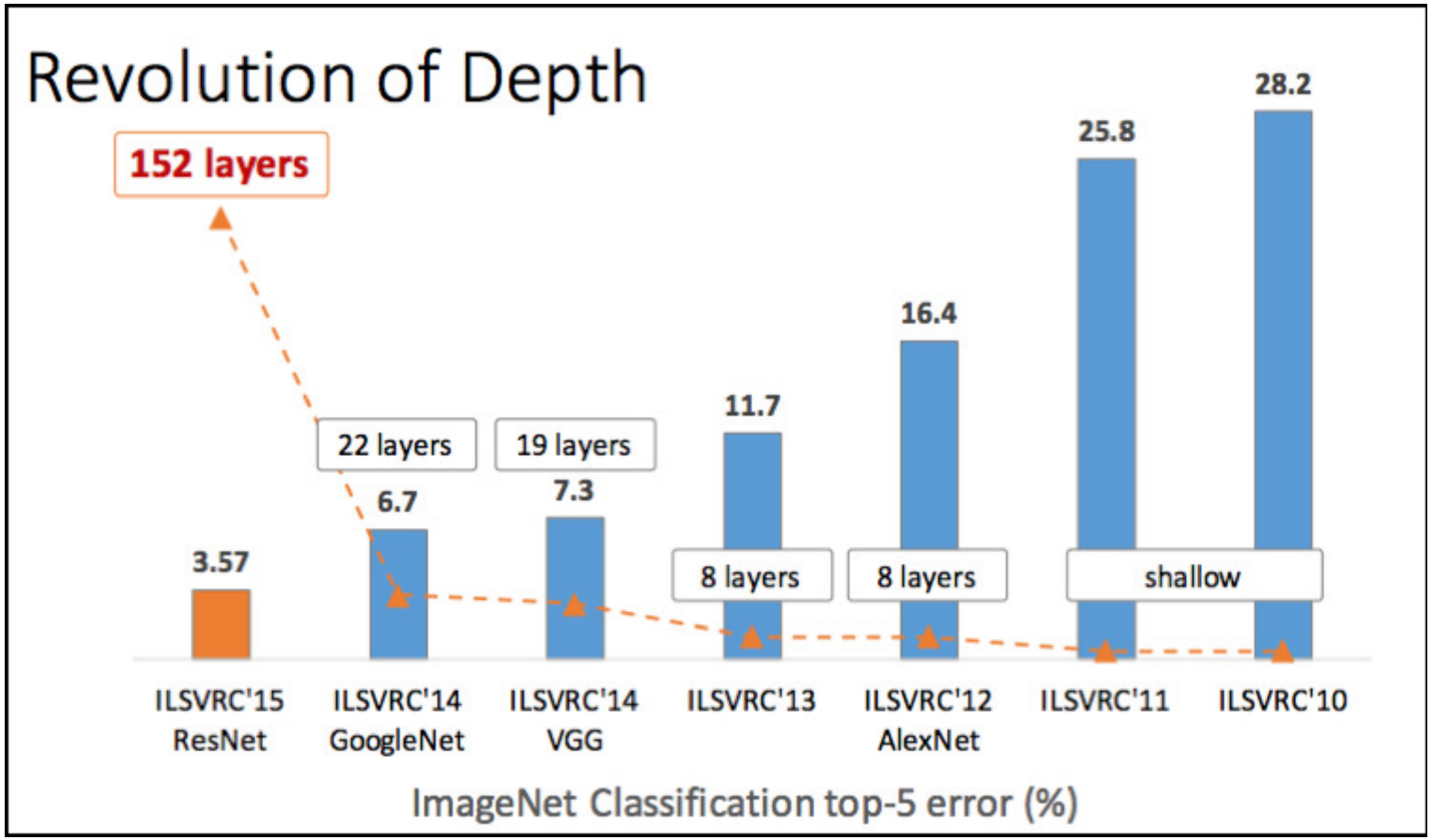
LeNet-5 błędy klasyfikacji



LeNet-5 porównanie z innymi metodami



ImageNet

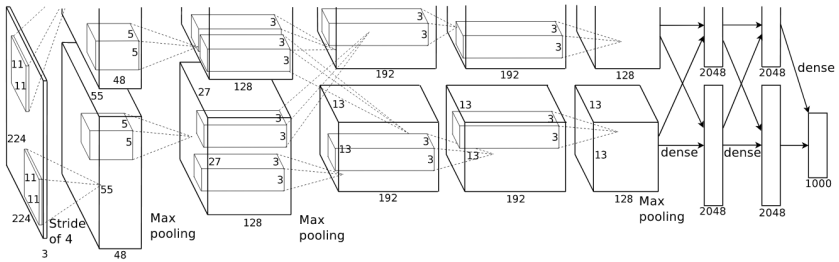


- ☞ CNNs Architectures used on ImageNet

AlexNet (A. Krizhevsky, 2012)

- Zwycięzca ☞ImageNet w 2012, błąd klasyfikacji 15.3% (poprawa z 26%)
- 1.2M obrazów, 1000 klas



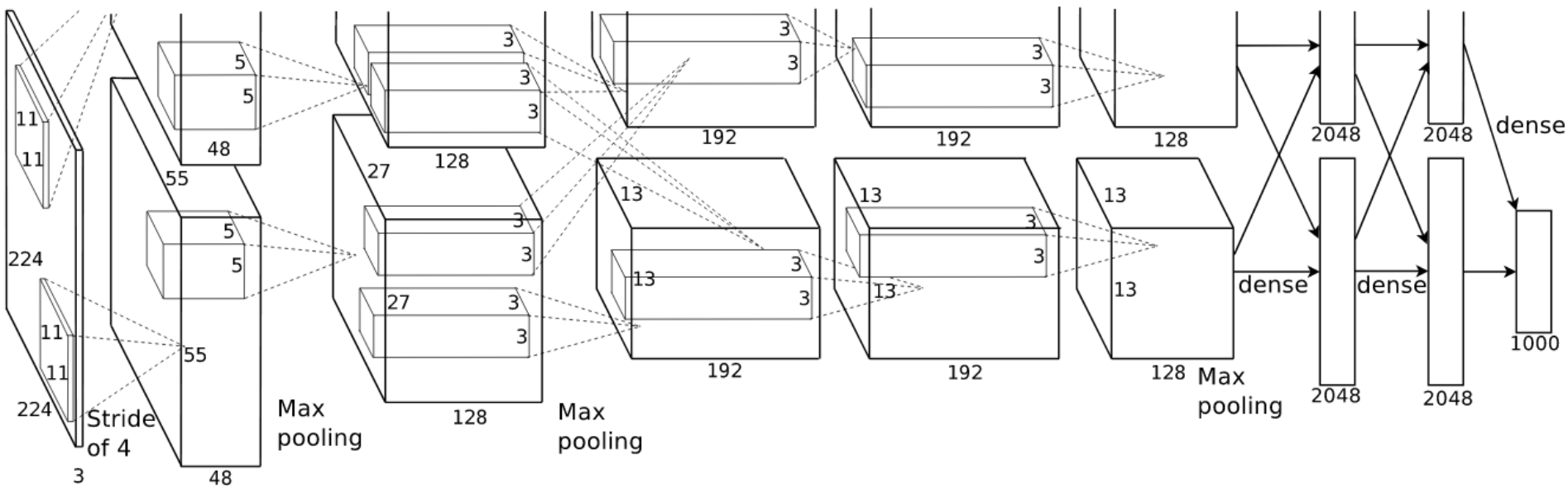


- architektura wzorowana LeNet5 ale głębsza (8 warstw) i więcej filtrów na warstwę, filtry 11x11, 5x5, 3x3
- sekwencje warstw splotowych z jednostkami ReLU
- regularyzacja: dropout, L2, rozszerzanie danych, normalizacja wyjść wybranych warstw
- SGD z momentem, 20 epok, 6 dni treningu (2x NVIDIA GTX 580 3GB GPUs), 60M parametrów

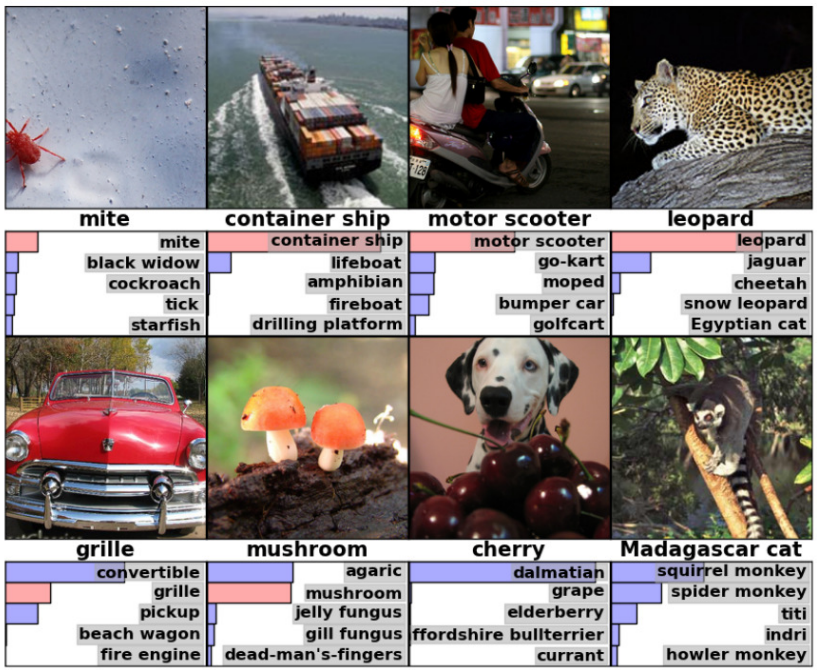
A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, 2012

AlexNet

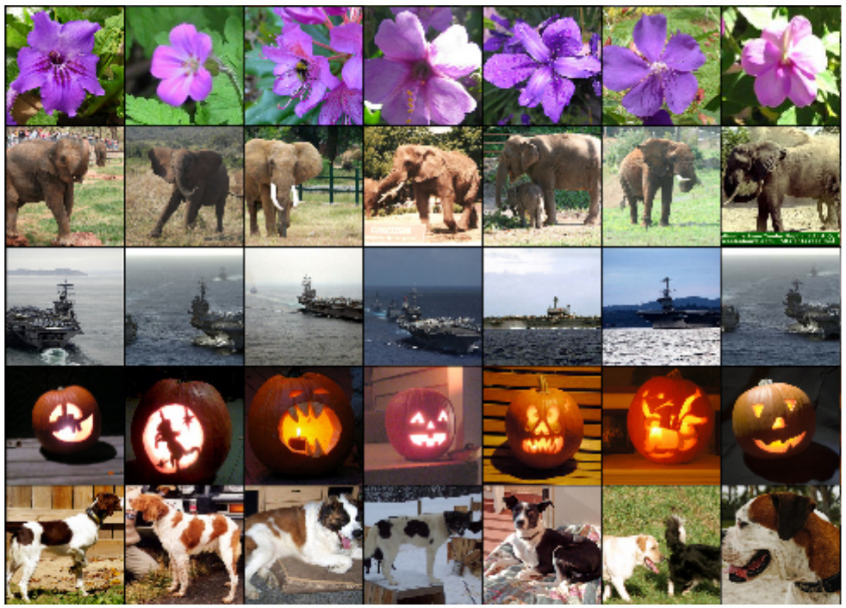
Layer	kernels	size	stride	output
Input				224x224x3
C1 + LRN	96	11x11	4x4	
MaxPooling		3x3	2x2	55x55x96
C2 + LRN	256	5x5		
MaxPooling		3x3	2x2	27x27x256
C3	384	3x3		13x13x384
C4	384	3x3		13x13x384
C5	256	3x3		
MaxPooling		3x3	2x2	13x13x256
F6 + dropout($p = 0.5$)	4096			
F7 + dropout($p = 0.5$)	4096			
Output softmax	1000			



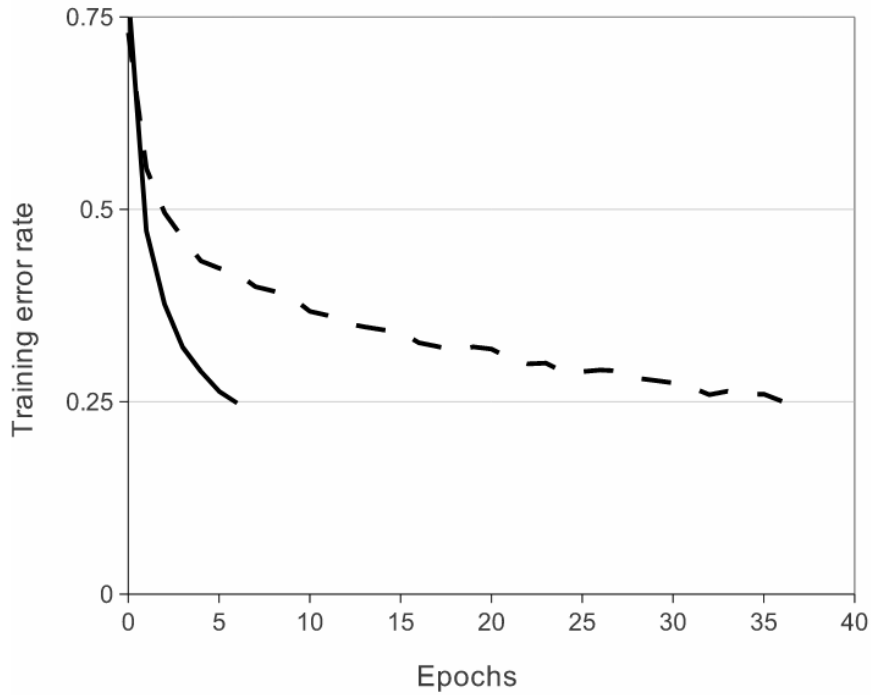
AlexNet



5 najsilniejszych odpowiedzi



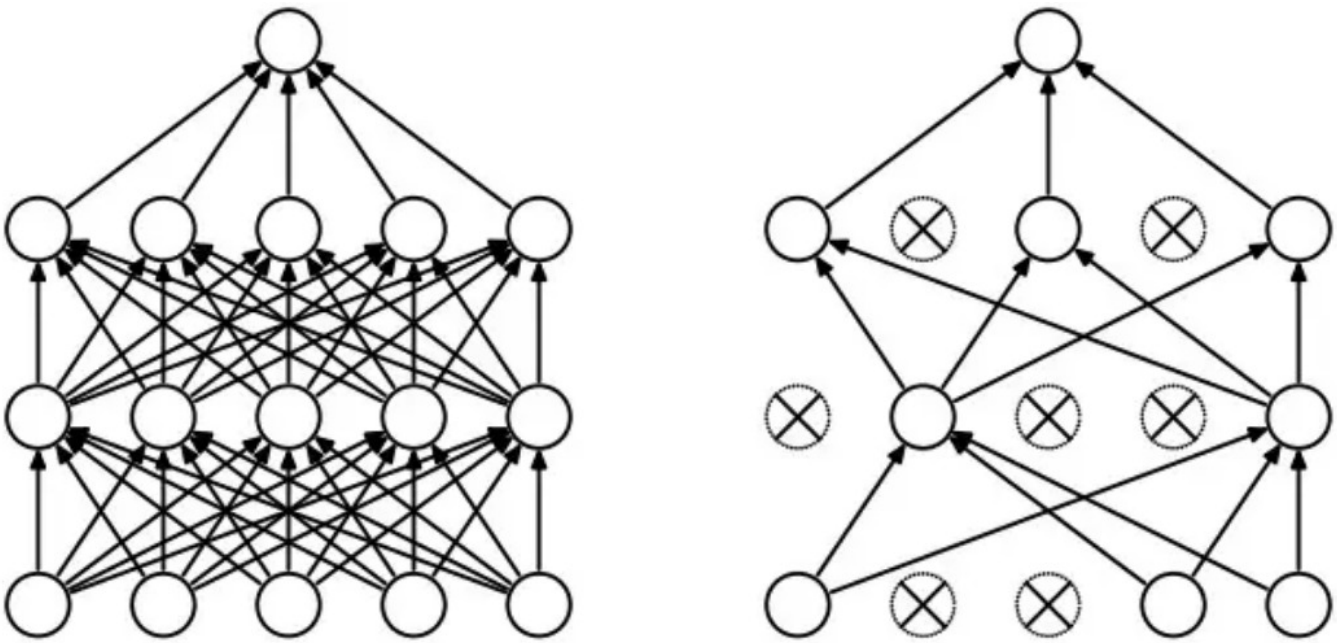
obraz wejściowy i 6 najbliższych wzorców względem odległości Euklidesowej wektora pobudzeń ostatniej warstwy ukrytej



- Porównanie szybkości zbieżności 4 warstwowej sieci splotowej z jednostkami ReLU i tanh na danych ImageNet
- Zastosowanie ReLU do kilkukrotnego przyspieszenia zbieżności
- Inicjalizacja: wagi z rozkładu Gaussa $N(0, 0.01)$, obciążenia (bias) $b = 1$, stąd większość ReLU aktywnych na początku treningu

Dropout

Dropout (Srivastava et al., 2014) dla każdego wzorca treningowego z prawdopodobieństwem p „wyrzuca” jednostki (zeruje ich aktywacje), odrzucone jednostki nie biorą udziału w treningu



(a) Standard Neural Net

(b) After applying dropout.

Rys: ☞ ResNet, AlexNet, VGGNet, Inception: Understanding various architectures of Convolutional Networks by Koustubh Sinhal

Dropout

- Silnie zapobiega przeuczeniu w warstwach w pełni połączonych
- Używany wyłącznie w czasie treningu. Podczas predykcji (*inference*) dropout jest wyłączany, czyli $p = 0$
- Każdy krok uczenia odbywa się ze zmienioną losowo architekturą sieci, uczenie przebiega innymi ścieżkami grafu połączeń
- Dla n jednostek mamy 2^n możliwych architektur, trening w mini-batchu uśrednia gradient względem różnych, wylosowanych sieci
- Przeciwdziała powstawaniu złożonych relacji pomiędzy wieloma neuronami, stąd pojedynczy neuron jest zmuszony wykrywać bardziej wartościowe cechy, niezależne od wpływu innych neuronów
- Wolniejsza zbieżność uczenia (AlexNet 2x wolniej z dropoutem)

Local response normalization

Normalizacja wartości $a^i_{x,y}$ pikseli w pozycji x, y i kanału (mapy cech) i wejściowych do warstwy splotowej

$$b^i_{x,y} = a^i_{x,y} \Big/ \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} \left(a^j_{x,y}\right)^2 \right)^{\beta}$$

- $k = 2, n = 5, \alpha = 10^{-4}, \beta = 0.75$ dobrane heurystycznie ze zbioru walidacyjnego
- realizuje normalizację jasności dla n sąsiadujących kolejnych map cech
- w AlexNet poprawa o 1.4% (top 1) oraz 1.2% (top 5)

Dodatkowa regularyzacja

Sztuczne rozszerzenie zbioru danych (*Data augmentation*)

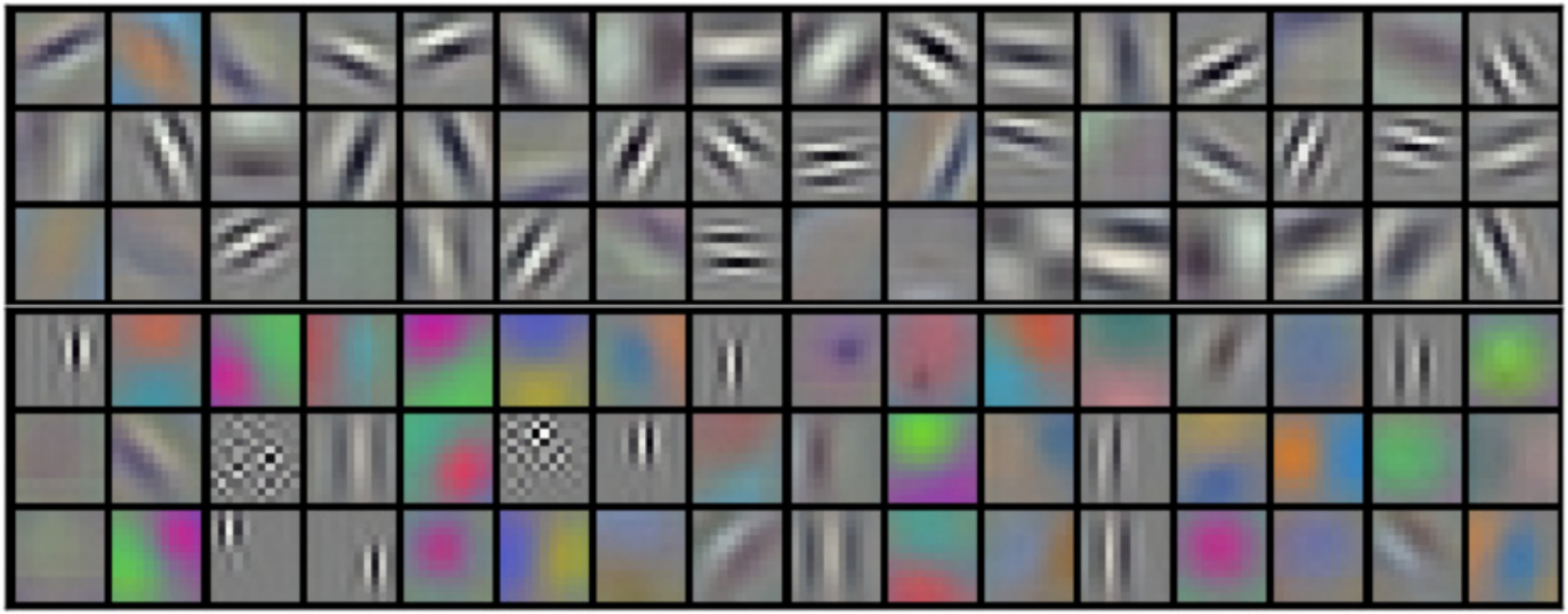
- przesunięcie i odbicia: losowanie obrazków 224x224 z 256x256, zwiększenie zbioru 2048 razy niezbędne do uniknięcia przeuczenia przy tak dużej sieci
- losowa modyfikacja intensywności kanałów RGB
- w AlexNet poprawa ponad 1% (top 1, top 5)

MaxPooling z nakrywaniem

- kernel 3x3, stride 2x2
- poprawa 0.4% (top 1) i 0.3% (top 5) względem architektury bez nakrywania (kernel 2x2)
- obserwacja: AlexNet rzadziej ulegał przeuczeniu

Zastosowanie 2 jednostek GPU pozwoliło na szkolenie sieci o większej liczbie filtrów co zaowocowało poprawą 1.7% (top-1) oraz 1.2% (top-5) w porównaniu z mniejszą siecią uczoną na pojedynczym GPU

Filtry w pierwszej warstwie

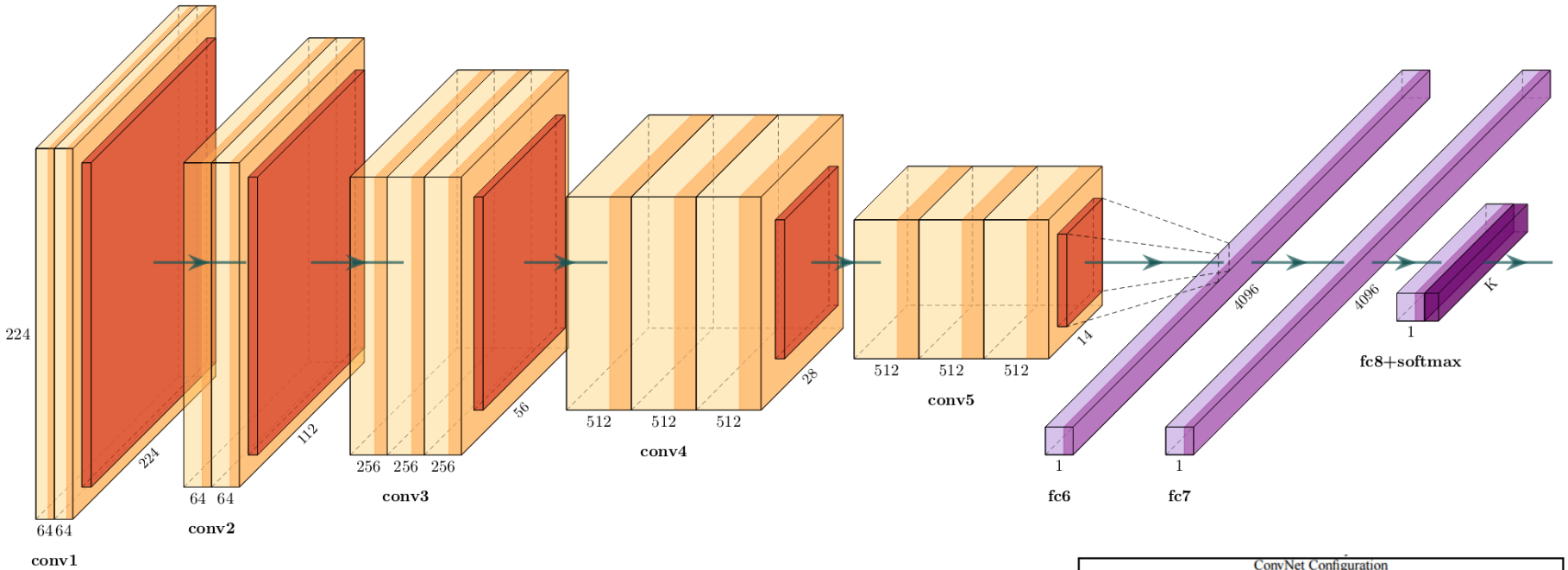


96 filtrów pierwszej warstwy splotowej, osobne kolumny warstw splotowych uczą się dwóch rodzin cech niskiego poziomu (obserwowane przy każdym treningu)

- filtry kolorów
- filtry krawędzi

VGGNet (Simonyan and Zisserman, 2014)

Visual Geometry Group, University of Oxford



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

- bloki VGG: sekwencja warstw splotowych z *padding same* + max pooling
- wielkość map zmniejszana o połowę po każdym bloku
- liczba filtrów zwiększana dwukrotnie po każdym bloku
- małe filtry (3×3), gęsty splot (stride=1)

VGGNet

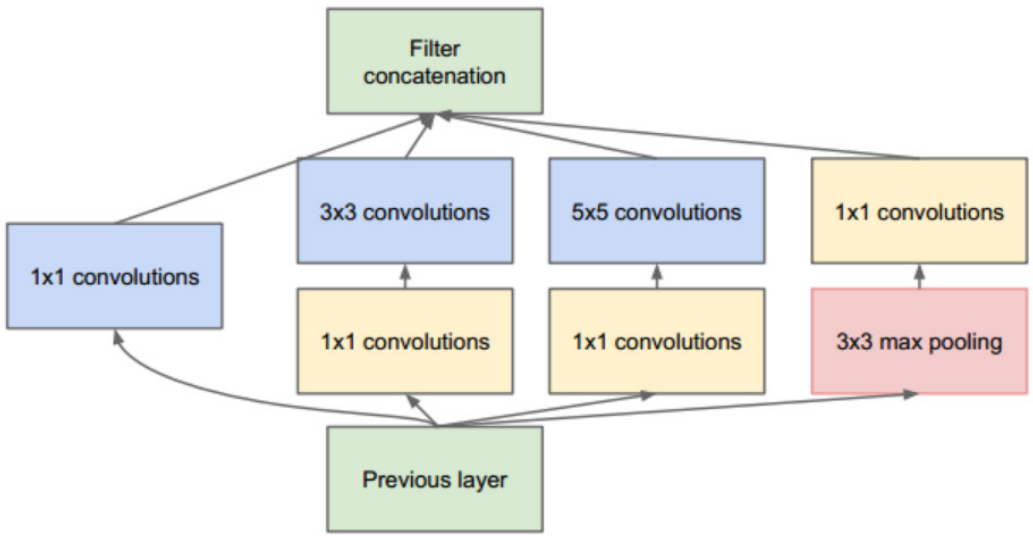
- zamiast dużych filtrów (w AlexNet 11x11, 7x7, 5x5) VGG stosuje we wszystkich warstwach filtry 3x3 zwiększając ich efektywne pola recepcyjne sekwencjami warstw splotowych
- małe filtry 3x3 zdolne do wykrycia bardziej subtelnych relacji w obrazach
- zwiększenie głębokości pozwala trenować bardziej złożone cechy
- stride=1, brak utraty informacji, gęsta konwolucja
- kilka architektur o różnej głębokości od 11 do 19 warstw
- sieć VGG16 138M parametrów, VGG19 143M parametrów
- trening 2-3 tygodnie (4x GPU), duże wymagania pamięciowe i obliczeniowe

K. Simonyan, A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, arXiv technical report, 2014

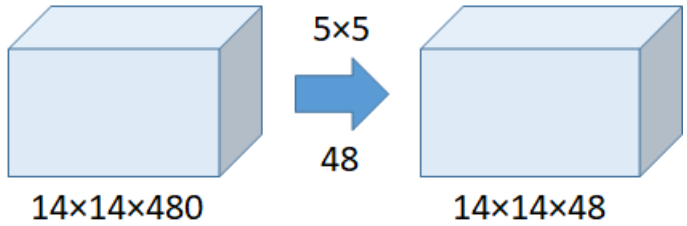
GoogLeNet/Inception

Sieć złożona z wielu bloków incepcyjnych (*inception module*) skonstruowanych według schematu:

- rozmiar filtru odwrotnie proporcjonalny do liczby filtrów, np. 128 filtrów 3x3 i 32 filtry 5x5
- równoległe przetwarzanie w kilku warstwach splotowych o zróżnicowanej wielkości filtrów 1x1, 3x3, 5x5 - detektory cech w różnej skali
- duże filtry poprzedzone splotem 1x1 w celu redukcji liczby parametrów. Splot 1x1 redukuje liczbę kanałów do zadanej liczby f

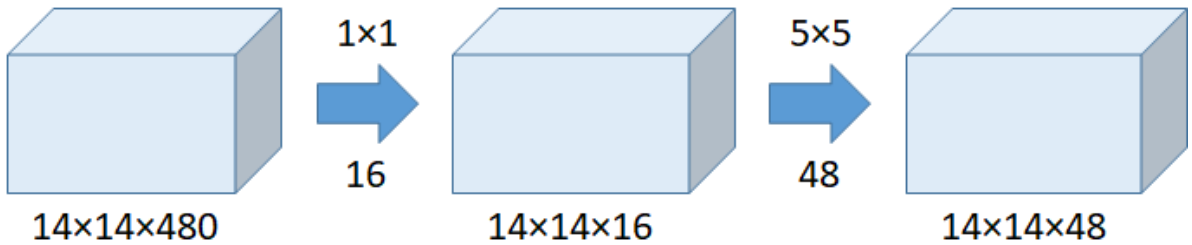


Redukcja złożoności splotem 1x1



ilość parametrów $5 \times 5 \times 48 \times 480 = 576K$

ilość operacji $(14 \times 14 \times 48) \times (5 \times 5 \times 480) = 112.9\text{M}$



ilość parametrów $1 \times 1 \times 16 \times 480 + 5 \times 5 \times 48 \times 16 = 26.88K$

ilość operacji

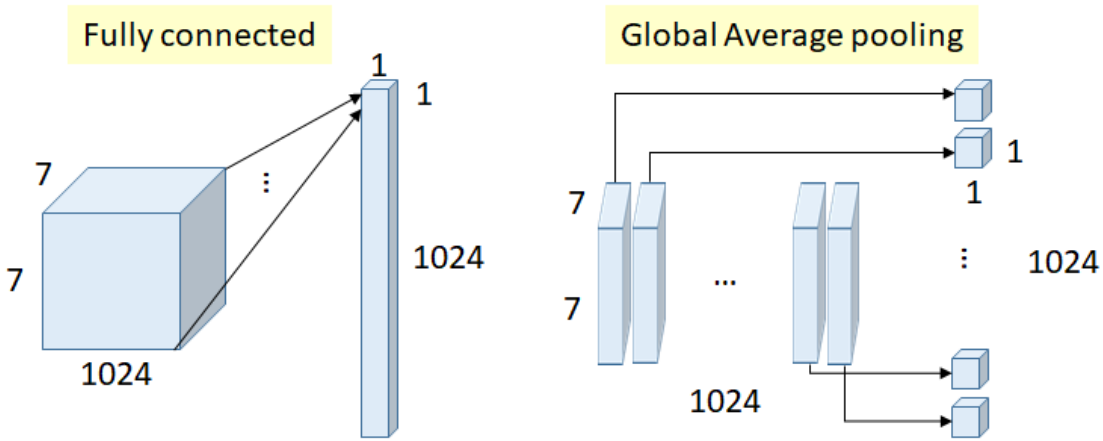
$(14 \times 14 \times 16) \times (1 \times 1 \times 480) + (14 \times 14 \times 48) \times (5 \times 5 \times 16) =$

$1.5\text{M} + 3.8\text{M} = 5.3\text{M}$

Rys: ☞ Review: GoogLeNet (Inception v1)— Winner of ILSVRC 2014 (Image Classification) by Sik-Ho Tsang

Global average pooling

Globalne uśrednienie (*global average pooling*) uśrednia wartości przestrzenne dla poszczególnych kanałów, np. dla mapy cech 7x7x1024 tworzy wektor 1024 elementowy, który jest bezpośrednio podawany do warstwy softmax

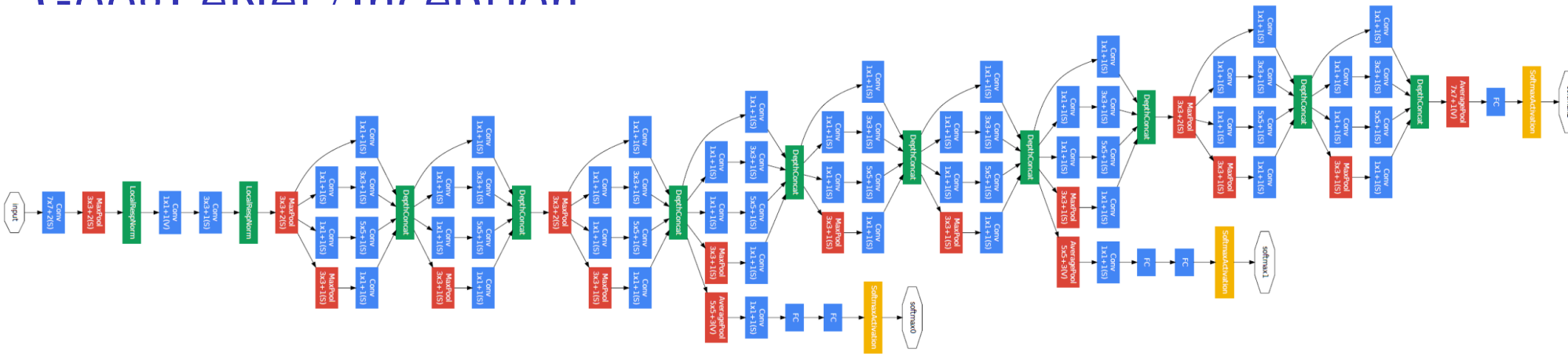


$7 \times 7 \times 1024 \times 1024 = 51.3\text{M}$

- Warstwy w pełni połączone zawierają najwięcej parametrów (w AlexNet 90% parametrów) w sieci
- global average pooling zastępuje warstwy w pełni połączone, redukując liczbę parametrów i zapobiegając przeuczeniu
- w GoogleNet poprawiło to wynik top-1 o 0.6%

Rys: ☞ Review: GoogLeNet (Inception v1)— Winner of ILSVRC 2014 (Image Classification) by Sik-Ho Tsang

GoogLeNet /Inception



- 22 warstwy
- warstwy przewężające (bottleneck) dodatkowo redukują złożoność sieci
- kilka wyjść softmax na różnych poziomach sieci, stabilizują trening i przyspieszają zbieżność, zapobiegają zanikaniu gradientu. Całkowita funkcja kosztu:

$$J(\theta) = \mathbb{E}_D[-\mathbf{t} \log \mathbf{y}_1 - \mathbf{t} \log \mathbf{y}_2 - \mathbf{t} \log \mathbf{y}_3]$$

- poprawność 93.3% top-5 na ImageNet, dużo szybszy w treningu od VGG
- kolejne iteracje: Inception v2, v3, InceptionResNet, Xception

ResNet (Residual Neural Network

Bloki ze skrótami (residual module)

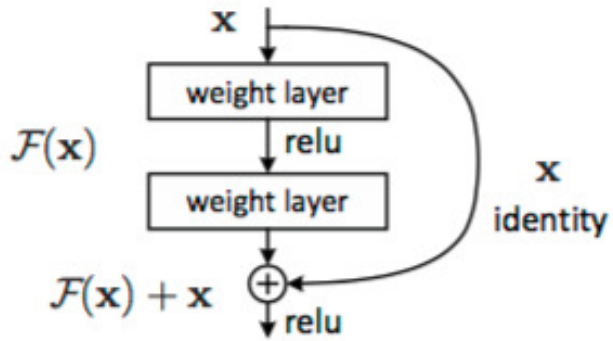
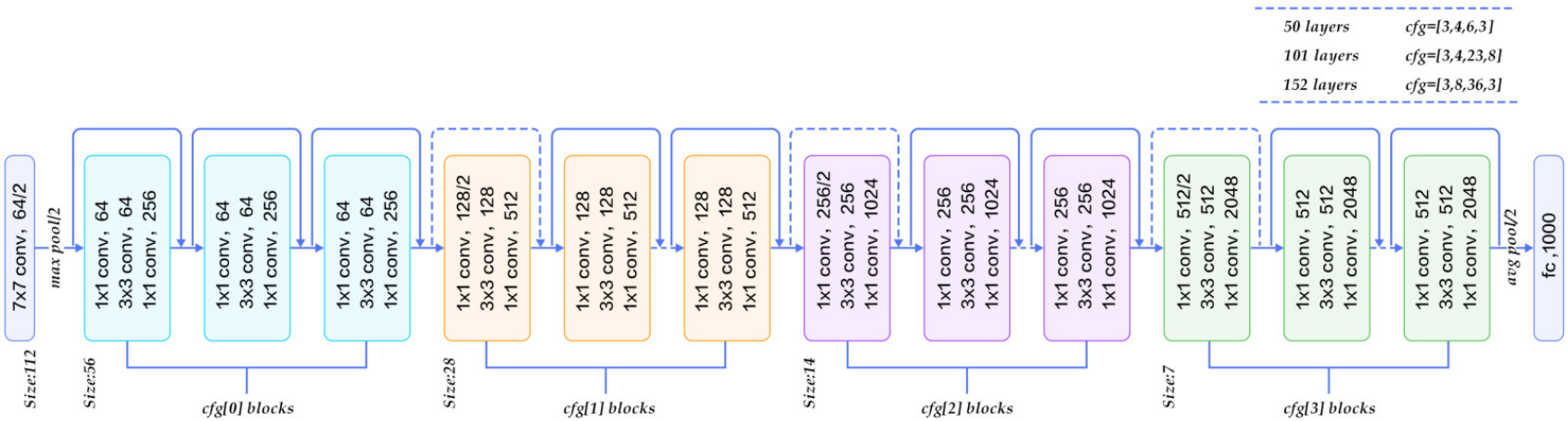


Figure 2. Residual learning: a building block.

- „skróty" łączą wejścia bloku z wyjściem poprzez odwzorowanie jednostkowe
- przejścia „skrótowe" pomagają uczyć bardzo głębokie sieci (152 warstwy)
- zapobiegają zanikaniu gradientu (sygnał może być propagowany skrótami)

ResNet



- wiele bloków zawierających sploty 1x1, 3x3, 1x1
- regularyzacja: batch normalization

Batch normalization

Batch normalization (Ioffe, 2015) normalizuje wejścia $x_i^{(k)}$ warstwy względem wartości średniej $\mu^{(k)}$ i wariancji $\sigma^{(k)}$ wzdłuż wymiaru k dla paczki danych w danej iteracji

$$y_i^{(k)} = \gamma^{(k)} \hat{x}_i^{(k)} + \beta^{(k)}$$

gdzie γ i β podlegają adaptacji w czasie treningu (uczenie normalizacji).
Czynnik normalizujący każde wejście

$$\hat{x}_i^{(k)} = \frac{x_i^{(k)} - \mu_B^{(k)}}{\sqrt{\sigma_B^{(k)^2} + \epsilon}}$$

średnia i odchylenie dla minipakietu o rozmiarze m

$$\mu_B = \frac{1}{m} \sum_{i=1}^{m} x_i, \qquad \sigma_B^2 = \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_B)^2$$

Ioffe, Sergey; Szegedy, Christian (2015). "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift".

Batch normalization

- **przyspieszenie zbieżności**, poprawia stabilność treningu, zmniejsza liczbę epok potrzebnych do osiągnięcia dobrych wyników
- **redukuje problem wewnętrznego przesunięcia kowariancji** - gdy zmienia się wyjście warstwy poprzedniej wówczas warstwa następna musi dopasować się do nowego rozkładu danych
- można stosować **wyższe współczynniki uczenia** bez obawy znikającego lub eksplodującego gradientu
- **zmniejsza zależność od inicjalizacji wag**, zwiększa odporność na złe ustawienia początkowe
- **poprawa generalizacji**: modele często lepiej generalizują na danych testowych
- wygładza powierzchnię błędu (Santurkar, 2018)
- **uniwersalność**: można stosować w różnych typach sieci

Batch normalization w ResNet

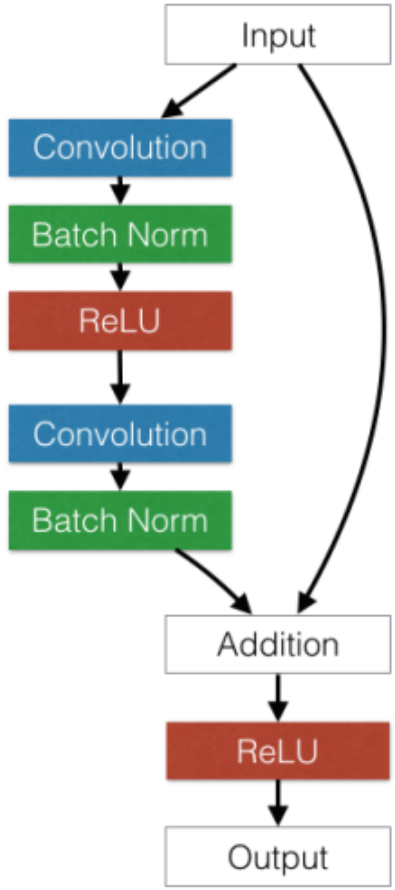
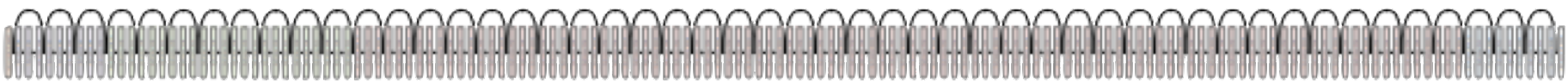


Figure 1. A RestNet basic block

HighNets: Highway networks

Sieci rezydualne, które starają się balansować wpływ głównej ścieżki (primary pathway) oraz ścieżki skrótowej (skip pathway)

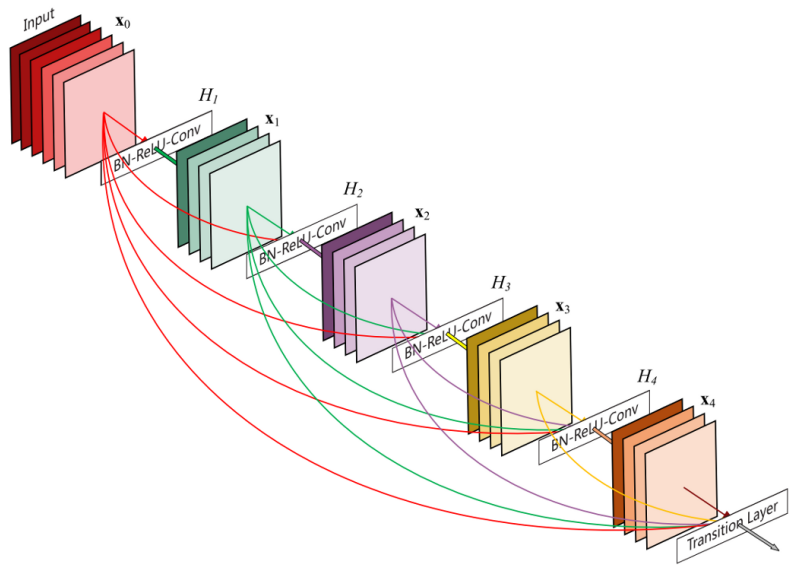
$$\mathbf{h}_n = T_{W'} f_W(h_{n-1}) + (\mathbf{1} - T_{W'}) h_{n-1}$$



- nawet do 1000 warstw
- wyniki state-of-the-art na MNIST i CIFAR 2015

DenseNets: Dense networks

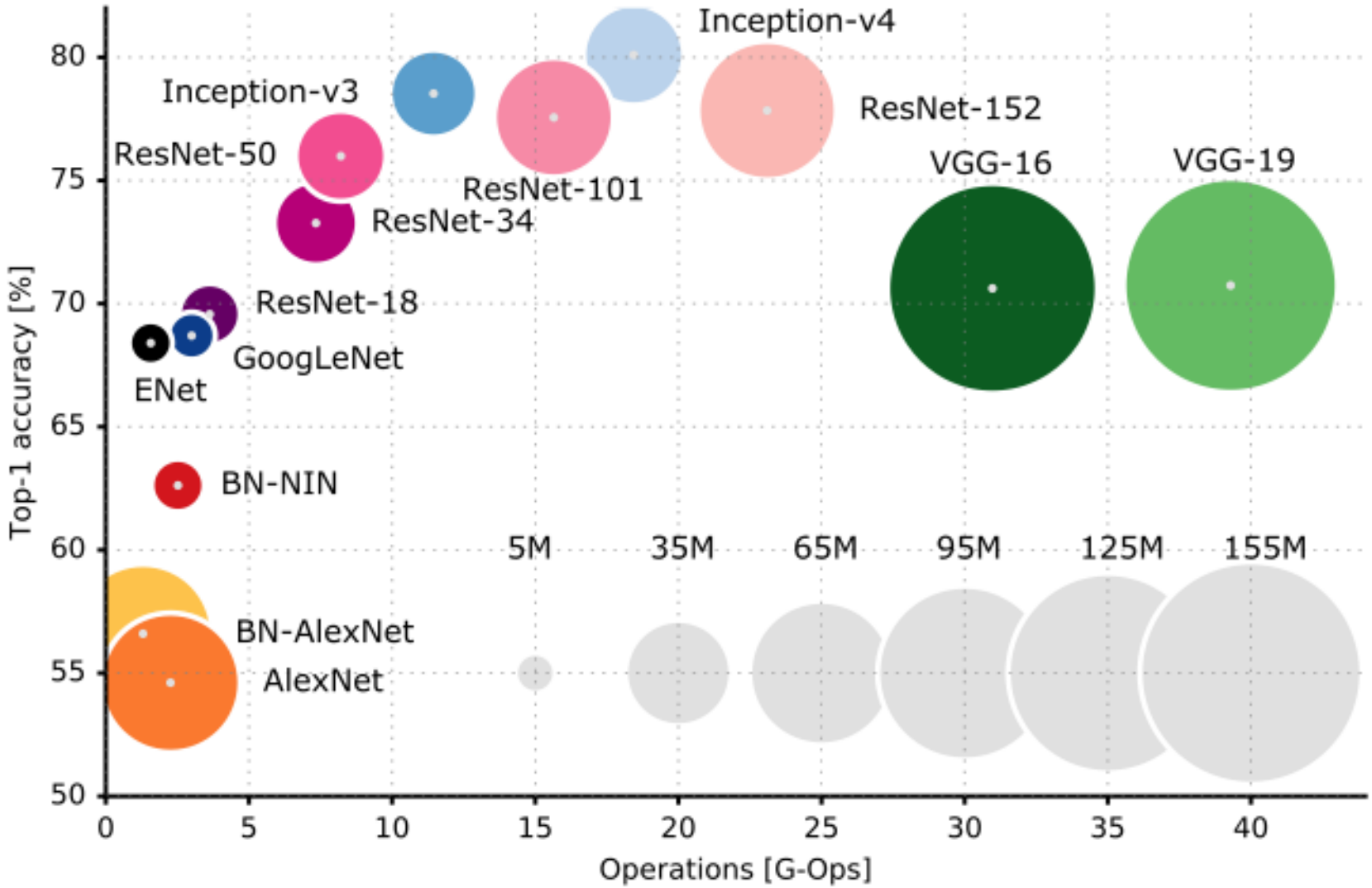
Sieci rezydualne, które posiadają połączenia skrótowe między wszystkimi blokami warstw sieci (do 5 bloków)



- do 100 warstw
- wyniki state-of-the-art na wielu danych benchmarkowych 2018

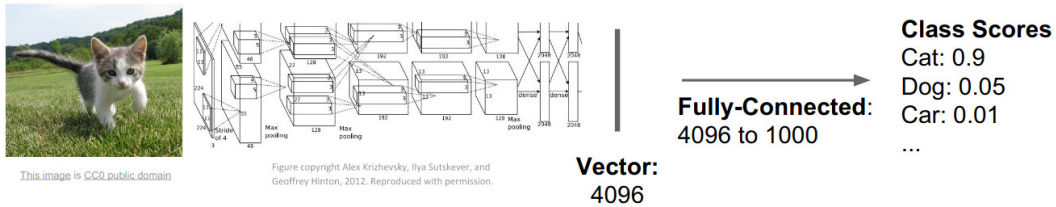
Porównanie popularnych sieci CNN

Poprawność klasyfikacji (top-1) zbioru ImageNet, liczba parametrów (zużycie RAM), szybkość działania (flops)

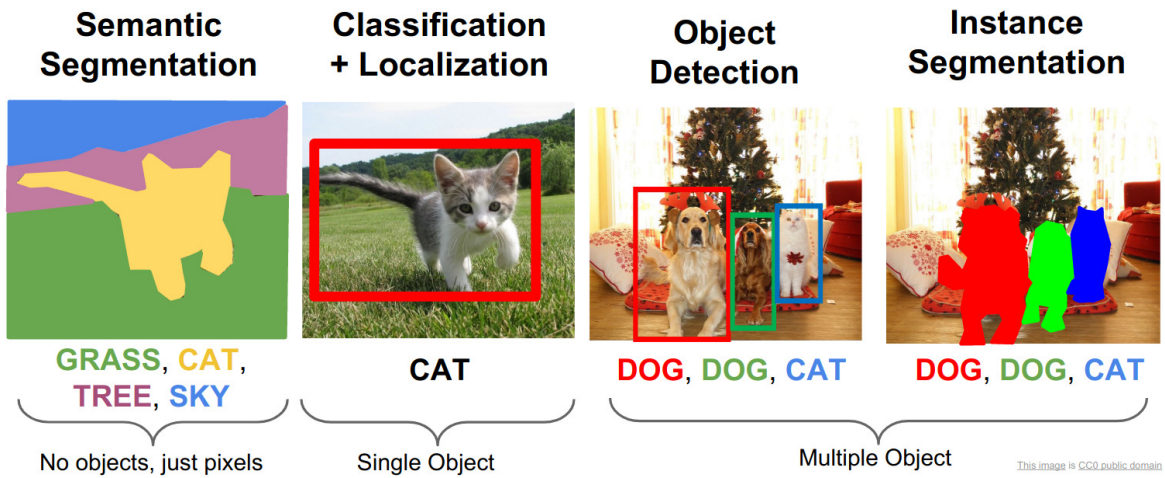


Analiza obrazów

- rozpoznawanie obiektów (klasyfikacja obrazów)



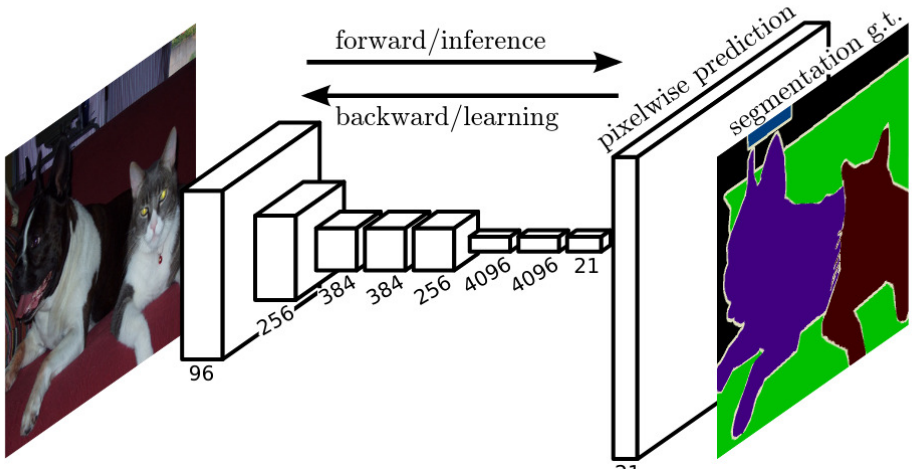
- detekcja i lokalizacja obiektów - określenie położenia obiektów na obrazie
- segmentacja semantyczna (klasyfikacja pikseli) i segmentacja obiektów



Rys: Fei-Fei Li, et al., "Detection and Segmentation", lecture

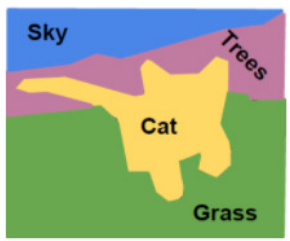
Fully Convolutional Network (FCN)

Segmentacja semantyczna - predykcja obiektów na poziomie pikseli. Wyjście sieci jest rozmiaru WxHxK, gdzie WxH - rozmiar obrazu wejściowego, K - ilość obiektów do segmentacji (tło, kot, pies, itd...)





Input Image

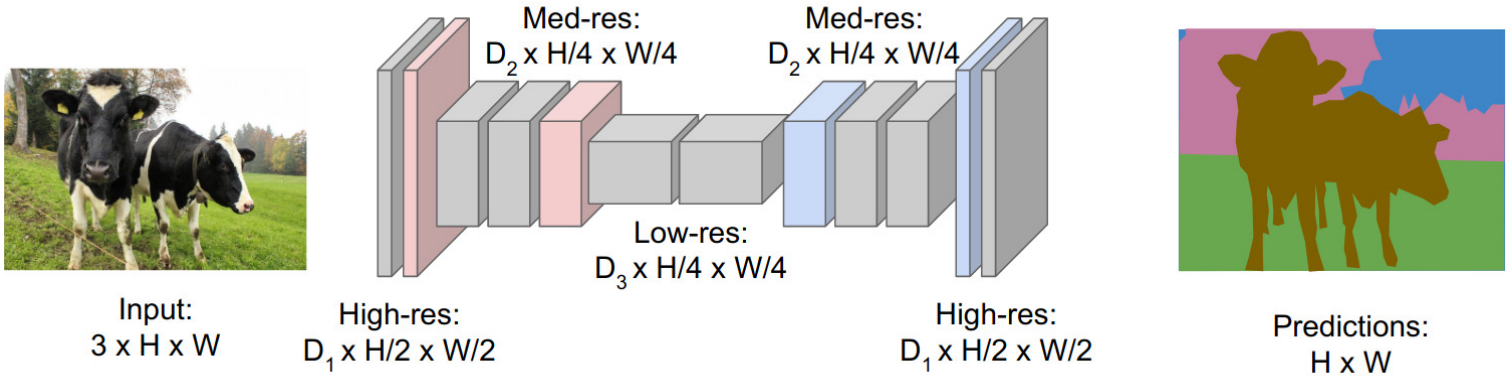


Expected Output (Source – Ref. 3)

Backbone network (kręgosłup) - wykorzystanie sieci do klasyfikacji obrazów (AlexNet, Googlenet, ResNet, VGG) jako część składowa FCN. Kręgosłup może być wstępnie wytrenowany na innym problemie analizy obrazów (np. ImageNet).

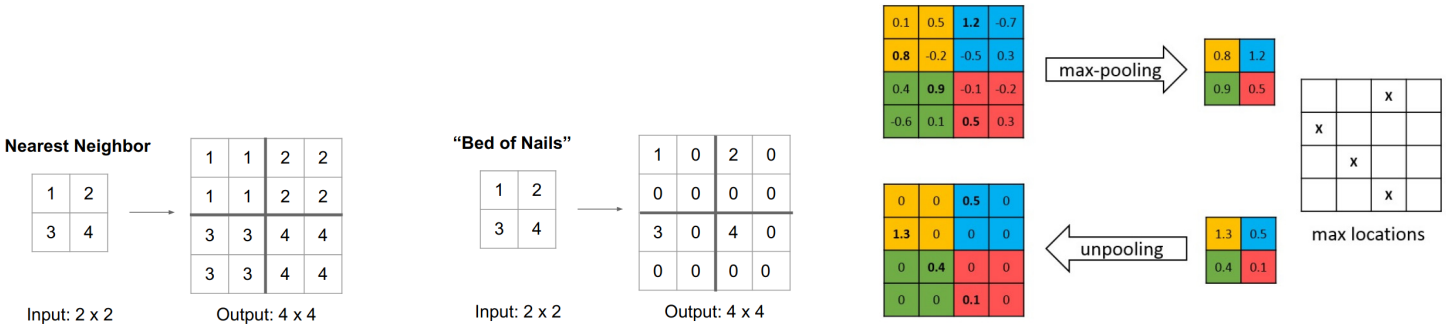
FCN: Downsampling and upsampling

- głębokie przetwarzanie obrazu o dużej rozdzielczości bardzo kosztowne
- FCN używa upsamplingu do odtworzenia oryginalnego rozmiaru obrazu

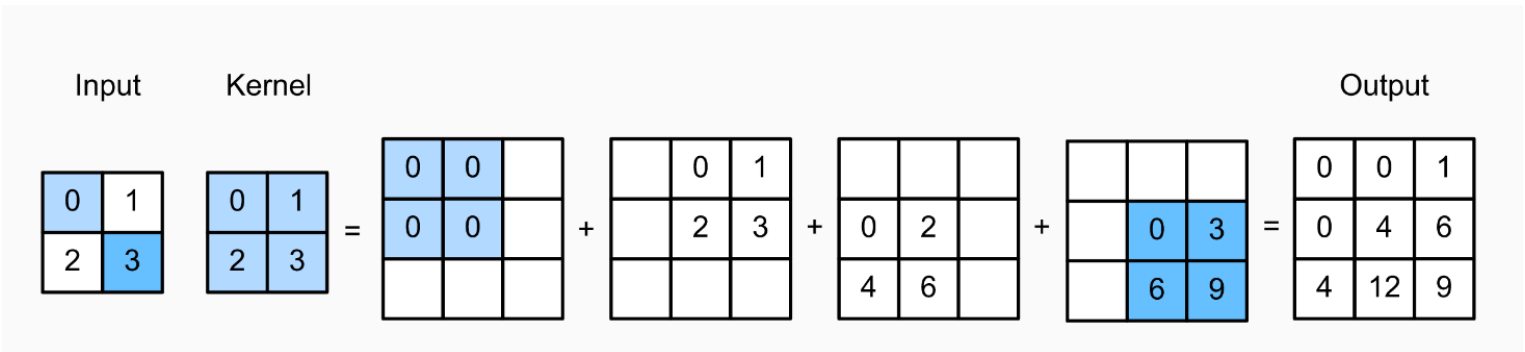


Unpooling i dekonwolucja

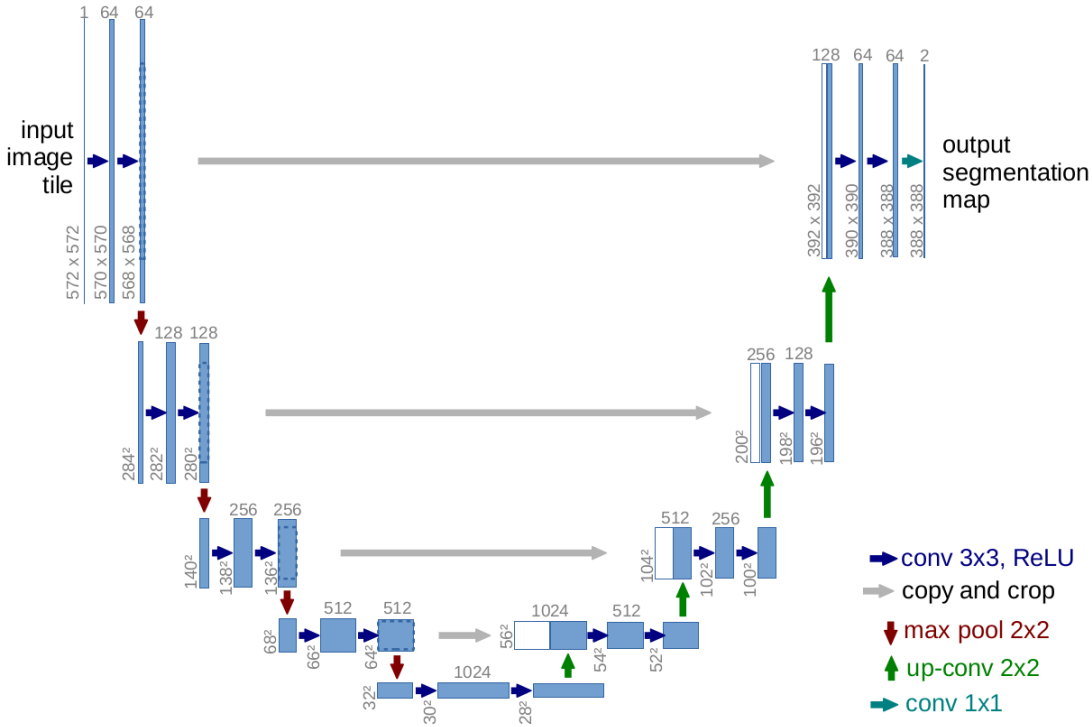
- unpooling



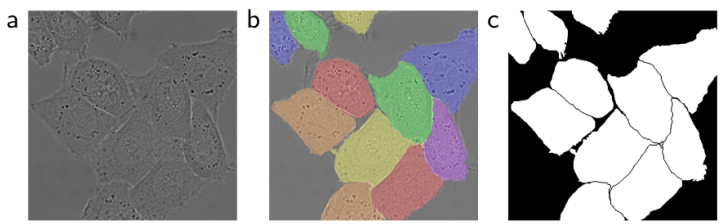
- deconvolution (transpozycja splotu)



U-Net (2015)



Przykład: segmentacja obrazów medycznych



☞ *O. Ronneberger, P. Fischer, . Brox U-Net: Convolutional Networks for Biomedical Image Segmentation, 2015*