POLITECHNIKA GDAŃSKA

Monitorowanie i Diagnostyka w Systemach Sterowania

Wydział Elektrotechniki i Automatyki
Katedra Elektrotechniki, Systemów Sterowania i Informatyki
Dr inż. Michał Grochowski

POLITECHNIKA GDAŃSKA
WYDZIAŁ ELEKTROTECHNIKI I AUTOMATYKI

Monitorowanie i Diagnostyka w Systemach Sterowania

na studiach II stopnia specjalności: Systemy Sterowania i Podejmowania Decyzji

Głębokie sieci neuronowe

w wykładzie wykorzystano materiały z następujących źródeł:

CS231n: Convolutional Neural Networks for Visual Recognition, 2016. Stanford University

Data Science and Robots (http://brohrer.github.io/blog.html)

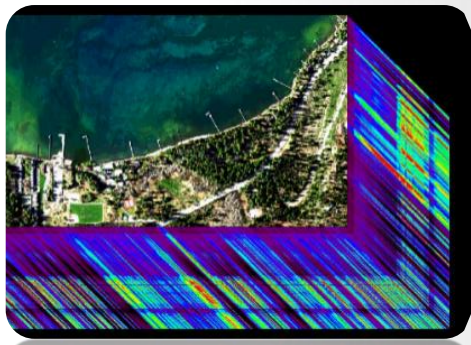
Opracował: dr inż. Michał Grochowski

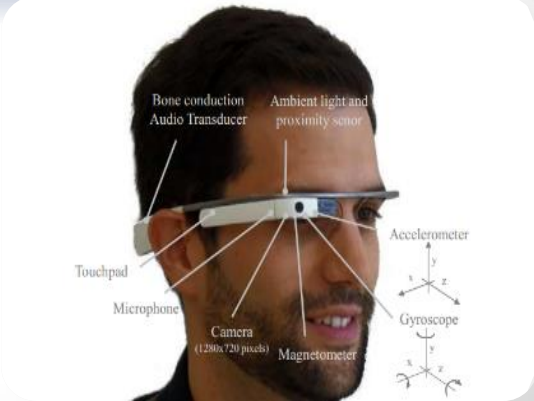
Motywacja

Od ok. 2015 era „ Big data” określana też jako Four V’s:

- Volume (scale of data),
- Velocity (streaming data),
- Variety (different forms of data),
- Veracity (uncertainty of data)

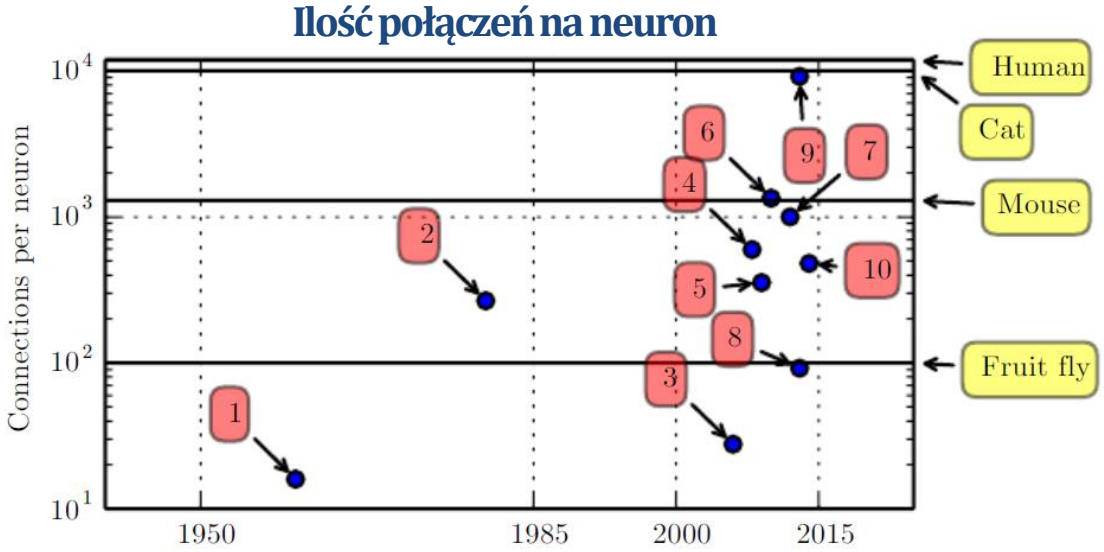
Technologie uczenia maszynowego, w tym ***głębokie sieci neuronowe*** posiadają cechy aby być skutecznym narzędziem do efektywnego przetwarzania tego rodzaju danych





Czym są głębokie sieci neuronowe ?

Rozrost stosowanych architektur sieci neuronowych

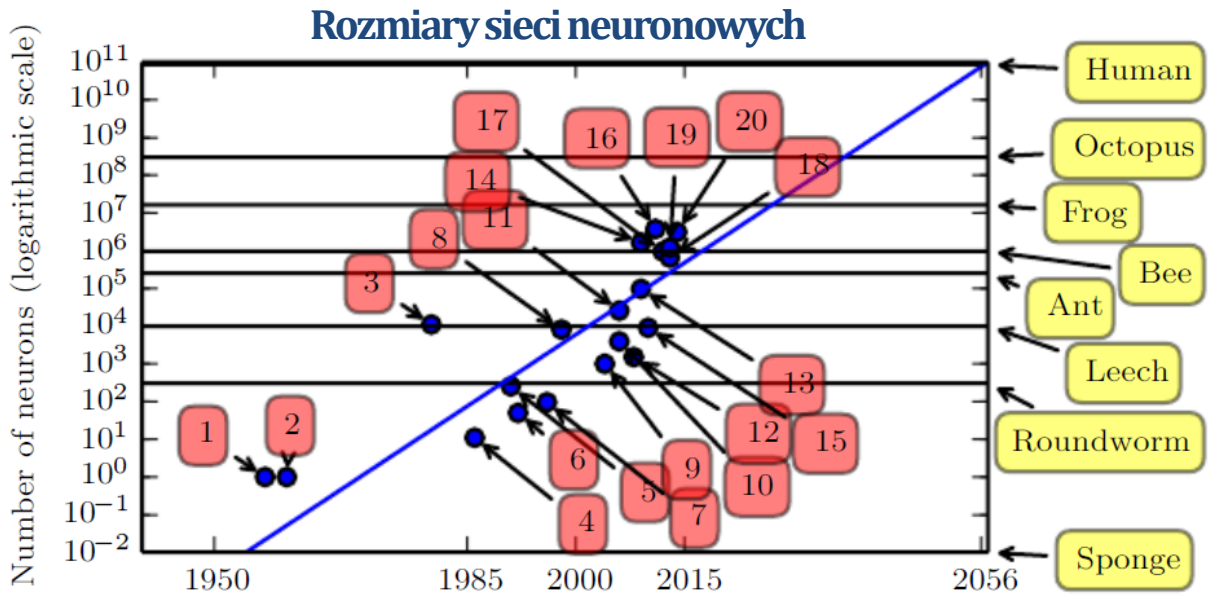


1. Adaptive linear element (Widrow and Hoff, 1960)
2. Neocognitron (Fukushima, 1980)
3. GPU-accelerated convolutional network (Chellapilla *et al.*, 2006)
4. Deep Boltzmann machine (Salakhutdinov and Hinton, 2009a)
5. Unsupervised convolutional network (Jarrett *et al.*, 2009)
6. GPU-accelerated multilayer perceptron (Ciresan *et al.*, 2010)
7. Distributed autoencoder (Le *et al.*, 2012)
8. Multi-GPU convolutional network (Krizhevsky *et al.*, 2012)
9. COTS HPC unsupervised convolutional network (Coates *et al.*, 2013)
10. GoogLeNet (Szegedy *et al.*, 2014a)

źródło: Ian Goodfellow and Yoshua Bengio and Aaron Courville. Deep Learning. MIT Press, 2016

Motywacja

Wielkość architektur sieci neuronowych

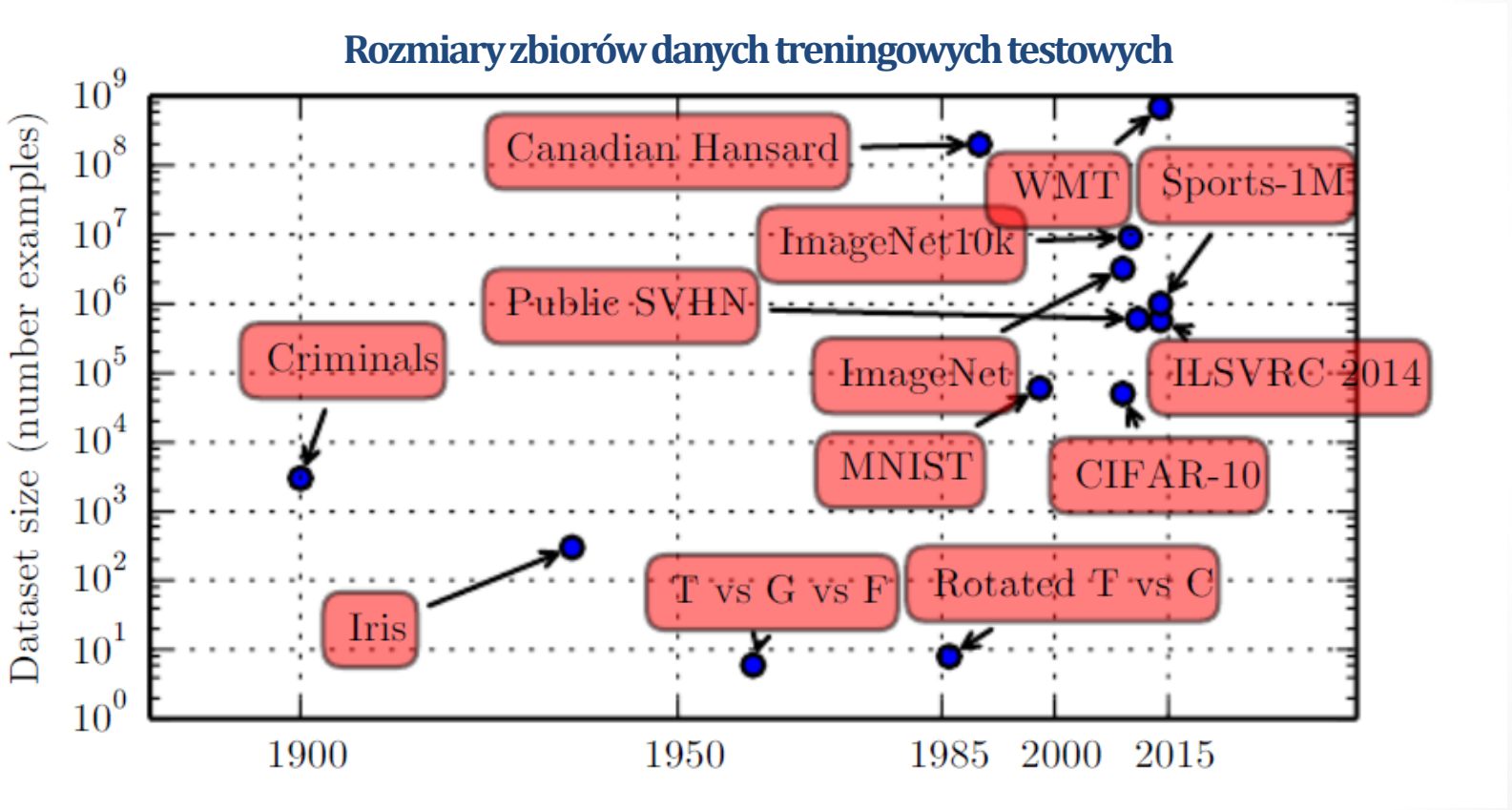


1. Perceptron (Rosenblatt, 1958, 1962)
2. Adaptive linear element (Widrow and Hoff, 1960)
3. Neocognitron (Fukushima, 1980)
4. Early back-propagation network (Rumelhart *et al.*, 1986b)
5. Recurrent neural network for speech recognition (Robinson and Fallsid
6. Multilayer perceptron for speech recognition (Bengio *et al.*, 1991)
7. Mean field sigmoid belief network (Saul *et al.*, 1996)
8. LeNet-5 (LeCun *et al.*, 1998b)
9. Echo state network (Jaeger and Haas, 2004)
10. Deep belief network (Hinton *et al.*, 2006)
11. GPU-accelerated convolutional network (Chellapilla *et al.*, 2006)
12. Deep Boltzmann machine (Salakhutdinov and Hinton, 2009a)
13. GPU-accelerated deep belief network (Raina *et al.*, 2009)
14. Unsupervised convolutional network (Jarrett *et al.*, 2009)
15. GPU-accelerated multilayer perceptron (Ciresan *et al.*, 2010)
16. OMP-1 network (Coates and Ng, 2011)
17. Distributed autoencoder (Le *et al.*, 2012)
18. Multi-GPU convolutional network (Krizhevsky *et al.*, 2012)
19. COTS HPC unsupervised convolutional network (Coates *et al.*, 2013)
20. GoogLeNet (Szegedy *et al.*, 2014a)

źródło: Ian Goodfellow and Yoshua Bengio and Aaron Courville. Deep Learning. MIT Press, 2016

Motywacja

Popularne bazy danych



źródło: Ian Goodfellow and Yoshua Bengio and Aaron Courville. Deep Learning. MIT Press, 2016

Konwolucyjne sieci neuronowe

- W 2012 r, konwolucyjna sieć neuronowa wygrała konkurs ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) i od tej pory corocznie wygrywają go głębokie sieci neuronowe
- Największy „komercyjny” sukces głębokich sieci neuronowych to pokonanie mistrzów świata i Europy Fan Hui (5-0) oraz Lee Sedol (4-1) w grę Go przez DNN AlphaGo (Google DeepMind)

Motywacja

Główni przedstawiciele

Geoffrey Hinton

Emeritus Professor of Computer Science, University of Toronto & Engineering Fellow, Google Inc.
Zweryfikowany adres z cs.toronto.edu - Strona główna
machine learning neural networks artificial intelligence cognitive science cor

Yann LeCun

Director of AI Research at Facebook & Silver Professor at the Courant Institute, New York University
Zweryfikowany adres z cs.nyu.edu - Strona główna
AI machine learning computer vision robotics image compression

Yoshua Bengio

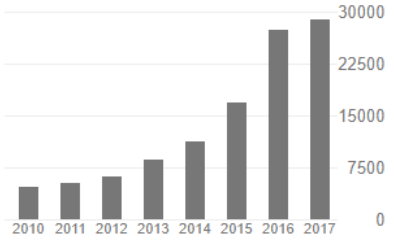
Professor, U. Montreal (Computer Sc. & Op. Res.), MILA, CIFAR, CRM, REPARTI, GRSNC
Zweryfikowany adres z umontreal.ca - Strona główna
Machine learning deep learning artificial intelligence

Zaangażowane badawczo, finansowo, wdrożeniowo firmy:

Google, Microsoft, Facebook, NVIDIA, IBM,
Apple, Adobe, Netflix,Uber, NEC, tysiące mniejszych…

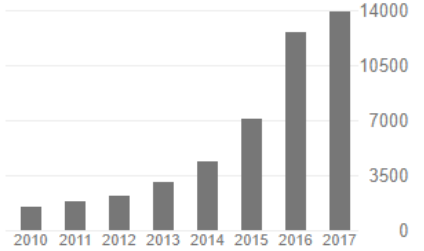
Cytowane przez WYŚWIETL WSZYSTKO

	Wszystkie	Od 2012
Cytowania	191020	99898
h-indeks	135	101
i10-indeks	314	227

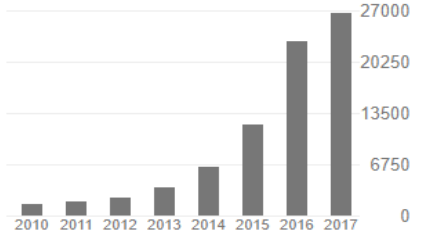


Cytowane przez WYŚWIETL WSZYSTKO

	Wszystkie	Od 2012
Cytowania	59911	43563
h-indeks	97	80
i10-indeks	220	182



	Wszystkie	Od 2012
Cytowania	86444	75065
h-indeks	103	96
i10-indeks	352	298



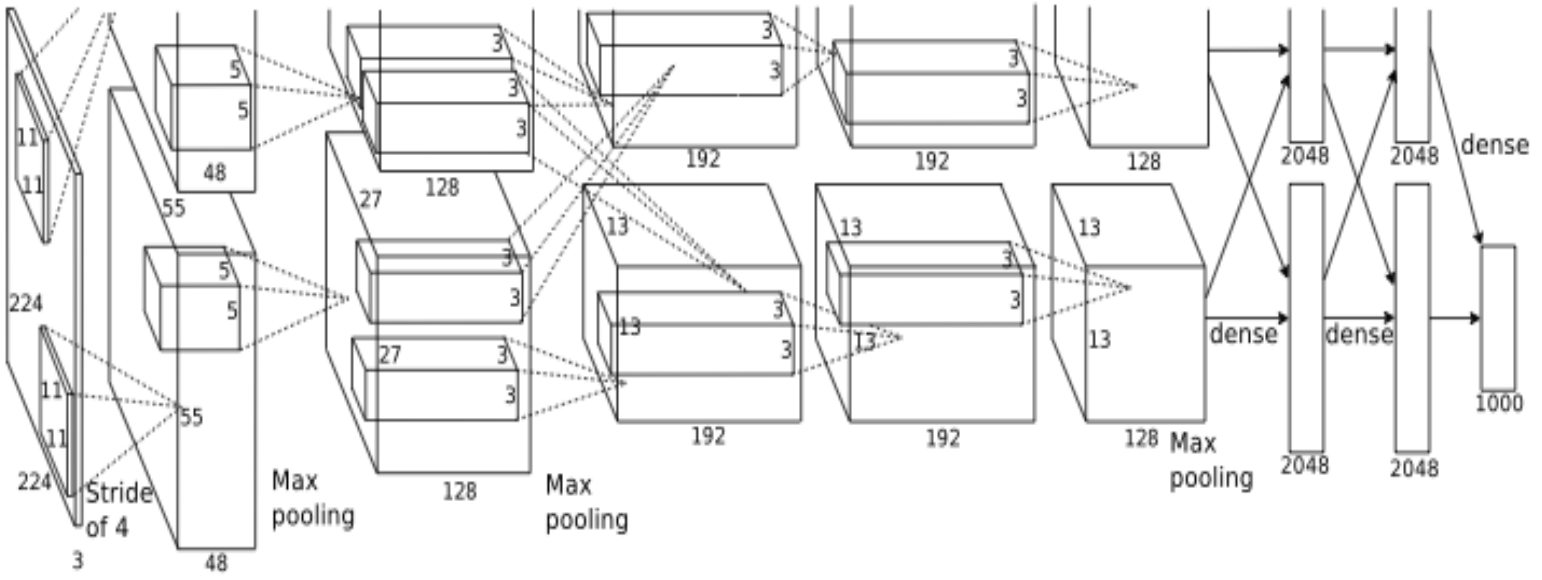
Czym są głębokie sieci neuronowe ?

Głębokie sieci neuronowe to sieci, które osiągają swoją skuteczność i elastyczność, dzięki zagnieżdżonej wielowarstwowej strukturze złożonej z hierarchicznych lokalnych modeli neuronowych, uczących się relacji o coraz wyższym stopniu abstrakcji, bazując na informacji z warstw poprzednich.

Rodzaje głębokich sieci neuronowych

- Wielowarstwowy perceptron
- ***Sieci konwolucyjne***
- Głęboka rekurencyjna sieć neuronowa (np. LSTM - Long short-term memory)
- Głębokie generatywne nienadzorowane sieci neuronowe (np. Deep Belive Boltzman Machines)
- Hybrydowe sieci neuronowe
- Ewolucyjne głębokie sieci neuronowe (Evolving Deep Neural Networks)
- …

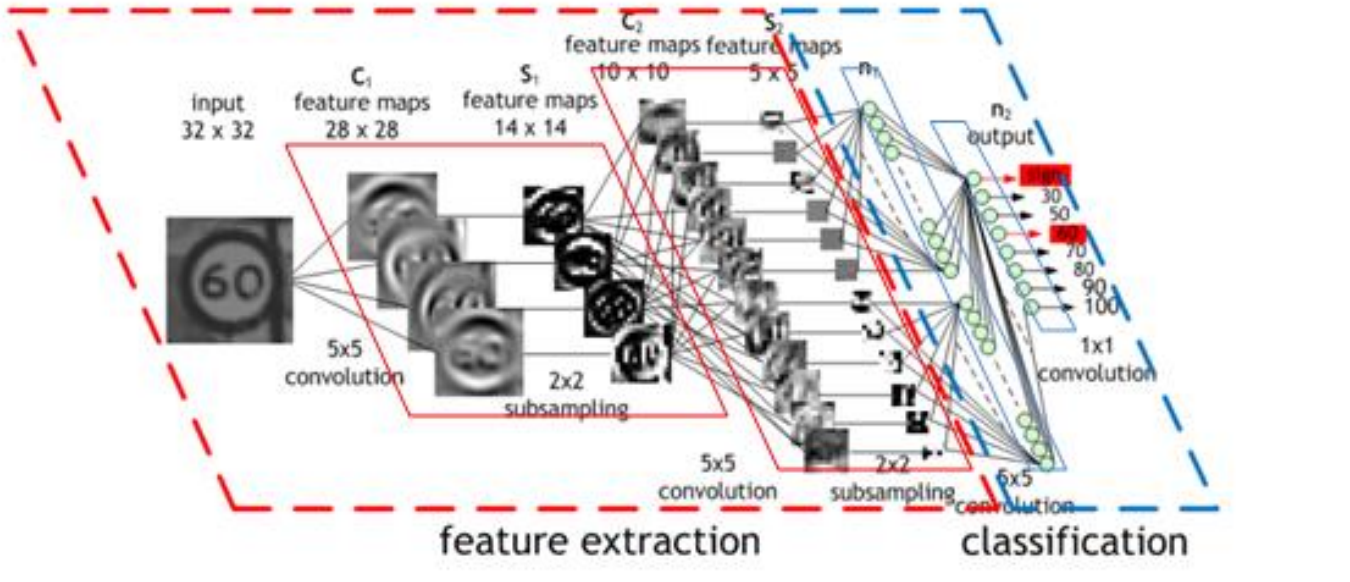
Convolutional Neural Networks (ConvNets, CNNs) - Deep Neural Networks (Splotowe sieci neuronowe – Głębokie sieci neuronowe)



„AlexNet” - Alex Krizhevsky, Ilya Sutskever, Geoffrey E Hinton, 2012

źródło: CS231n: Convolutional Neural Networks for Visual Recognition, 2016. Stanford University

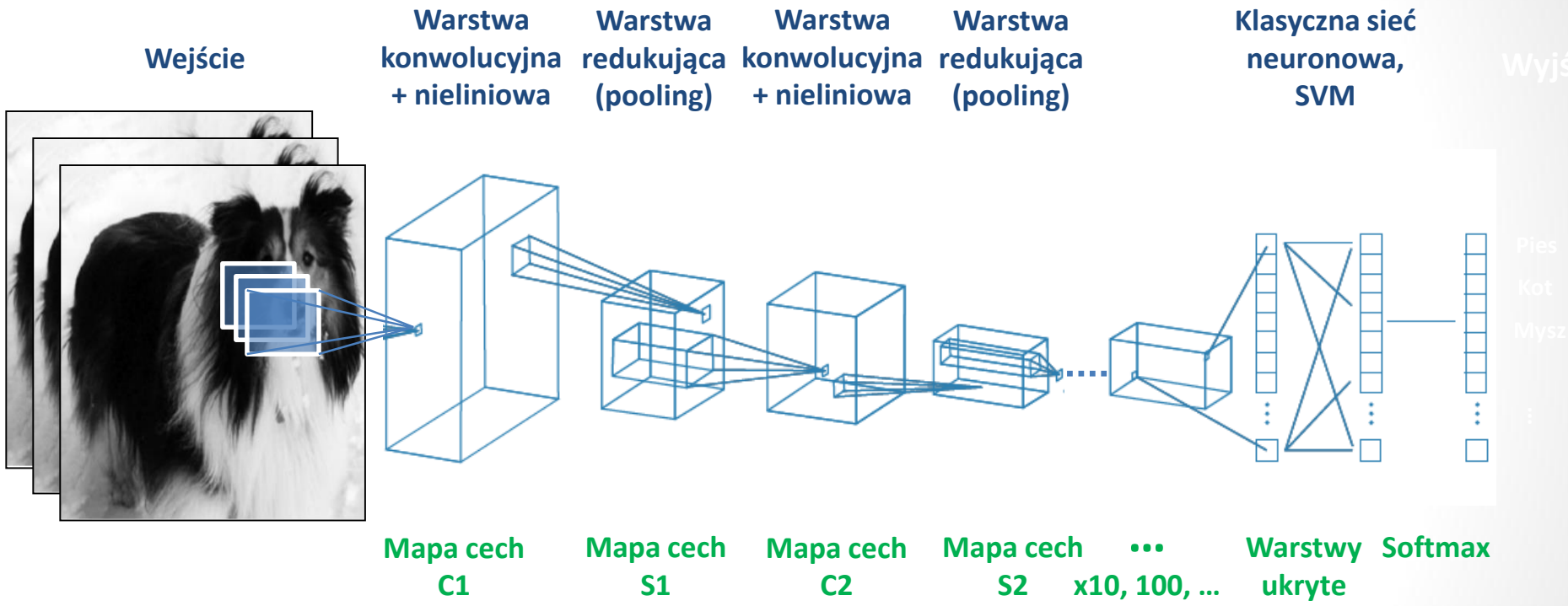
Convolutional Neural Networks (ConvNets, CNNs) - Deep Neural Networks (Splotowe sieci neuronowe – Głębokie sieci neuronowe)



An image of a traffic sign is filtered by 4 5×5 convolutional kernels which create 4 feature maps, these feature maps are subsampled by max pooling. The next layer applies 10 5×5 convolutional kernels to these subsampled images and again we pool the feature maps. The final layer is a fully connected layer where all generated features are combined and used in the classifier (essentially logistic regression). Image by Maurice Peemen.

https://devblogs.nvidia.com/parallelforall/deep-learning-nutshell-core-concepts/

Konwolucyjne sieci neuronowe



Fragmenty prezentacji:

How it Works: Convolutional Neural Networks

źródło:

Blog: Data Science and Robots

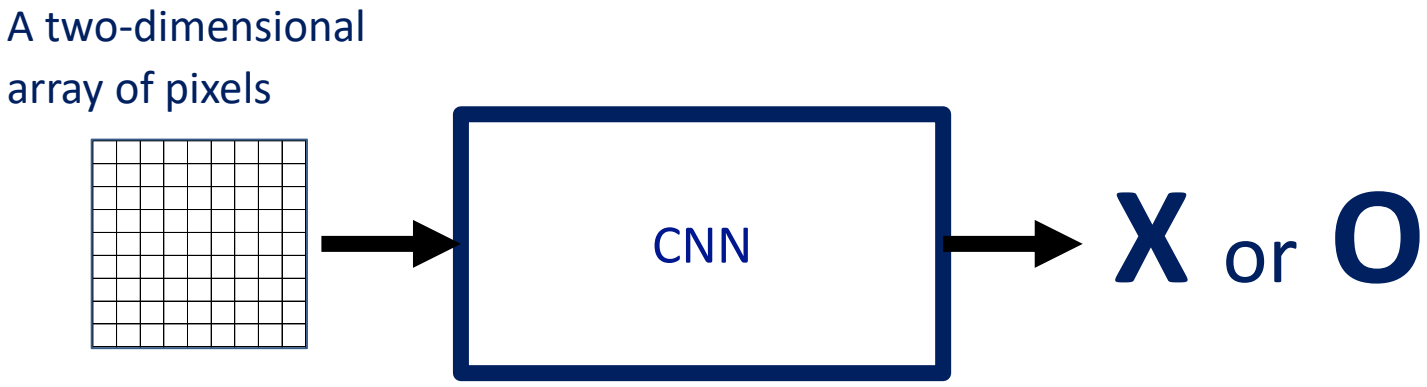
Brandon Rohrer

http://brohrer.github.io/blog.html

brohrer@microsoft.com

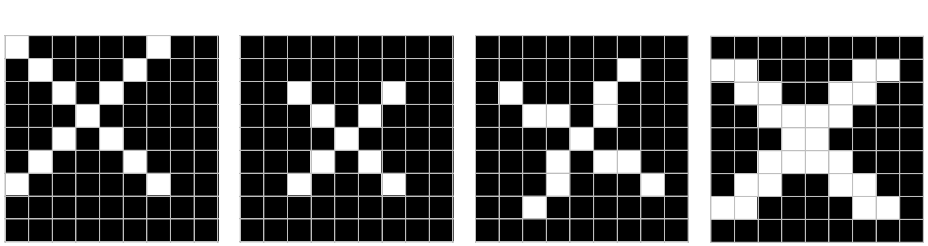
A toy ConvNet: X's and O's

Says whether a picture is of an X or an O

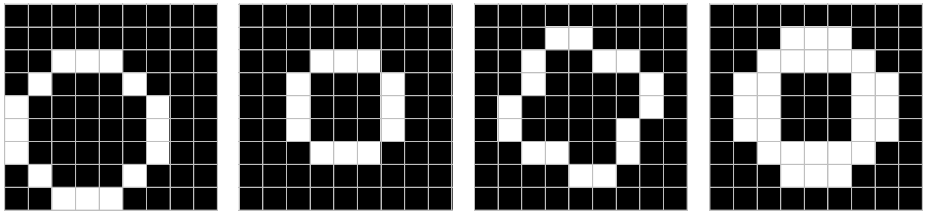


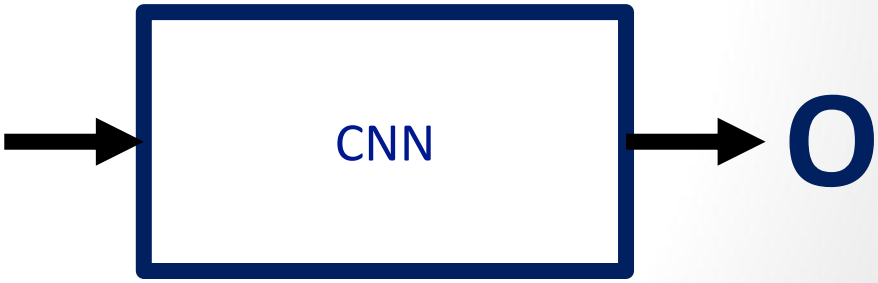
źródło: http://brohrer.github.io/blog.html

Trickier cases



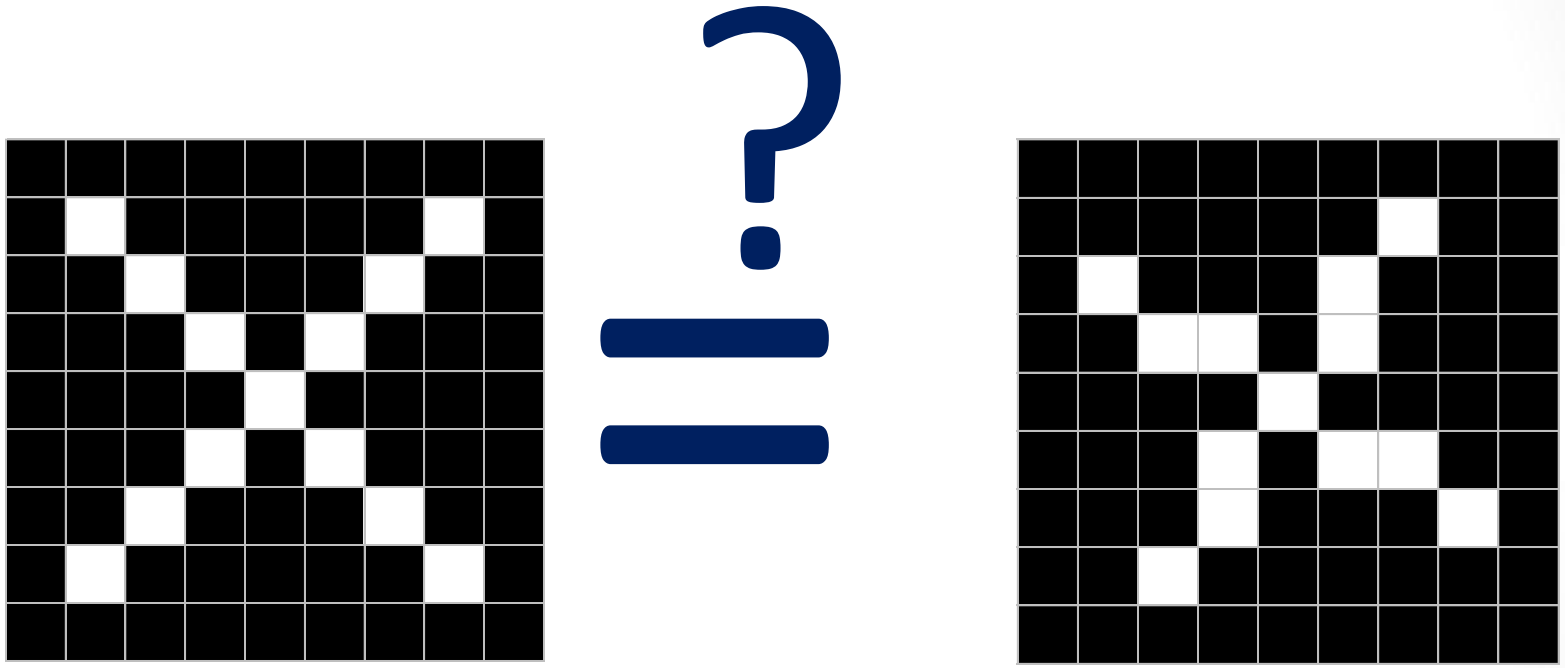
translation scaling rotation weight





źródło: http://brohrer.github.io/blog.html

Deciding is hard



źródło: http://brohrer.github.io/blog.html

What computers see

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

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

źródło: http://brohrer.github.io/blog.html

What computers see

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	X	-1	-1	-1	-1	X	X	-1
-1	X	X	-1	-1	X	X	-1	-1
-1	-1	X	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	X	-1	-1
-1	-1	X	X	-1	-1	X	X	-1
-1	X	X	-1	-1	-1	-1	X	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

źródło: http://brohrer.github.io/blog.html

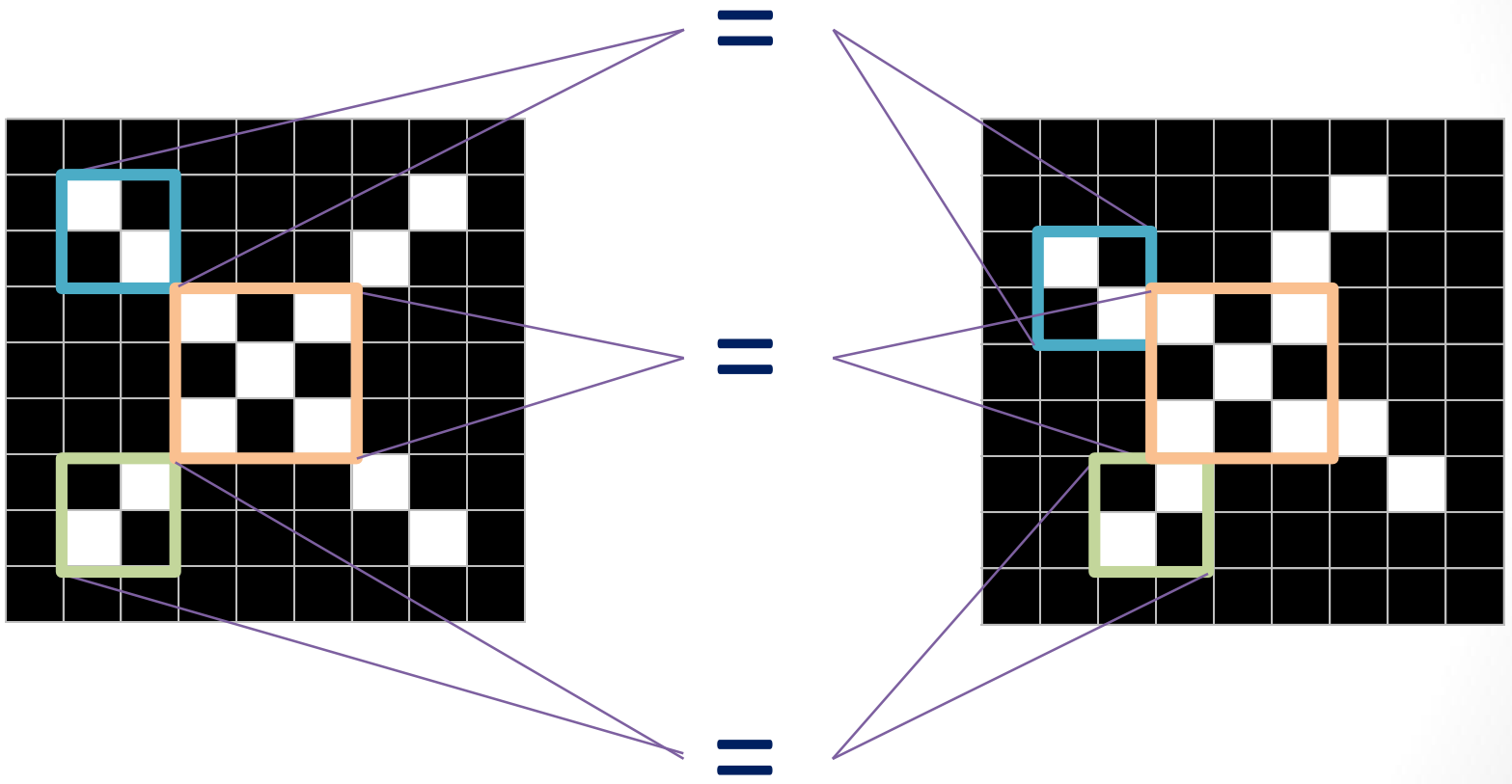
Computers are literal

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

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

źródło: http://brohrer.github.io/blog.html

ConvNets match pieces of the image



źródło: http://brohrer.github.io/blog.html

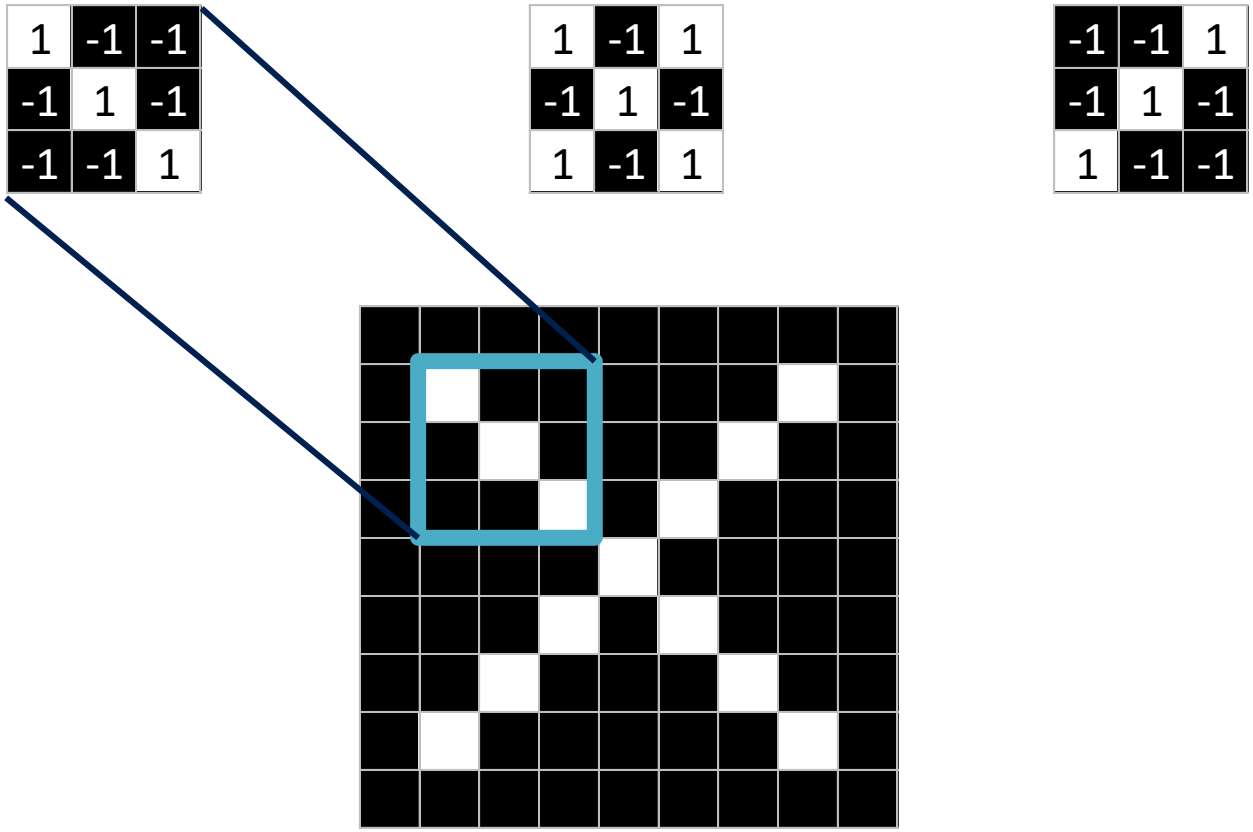
Features match pieces of the image

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

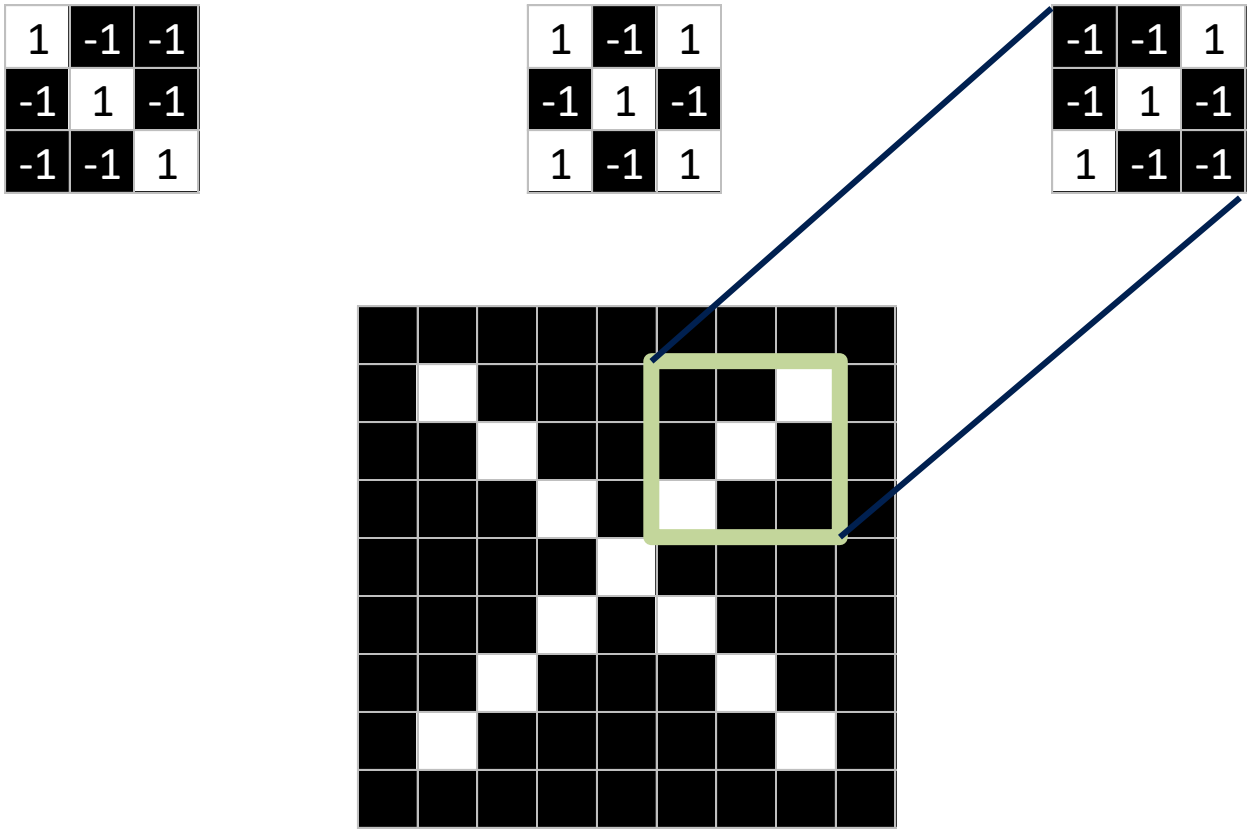
1	-1	1
-1	1	-1
1	-1	1

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

źródło: http://brohrer.github.io/blog.html



źródło: http://brohrer.github.io/blog.html

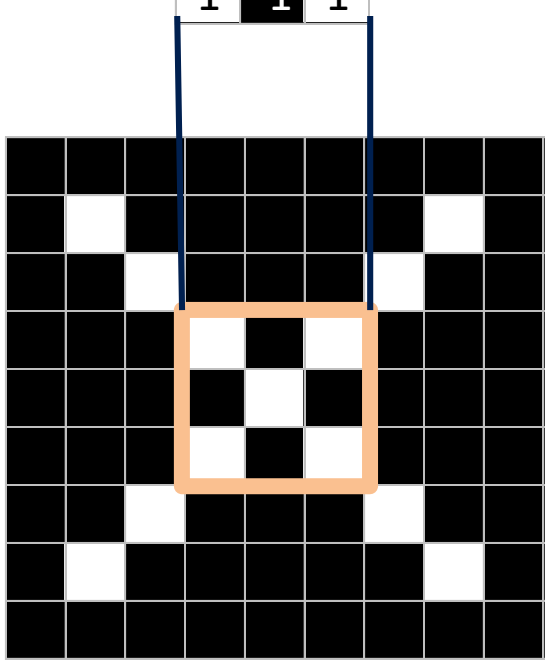


źródło: http://brohrer.github.io/blog.html

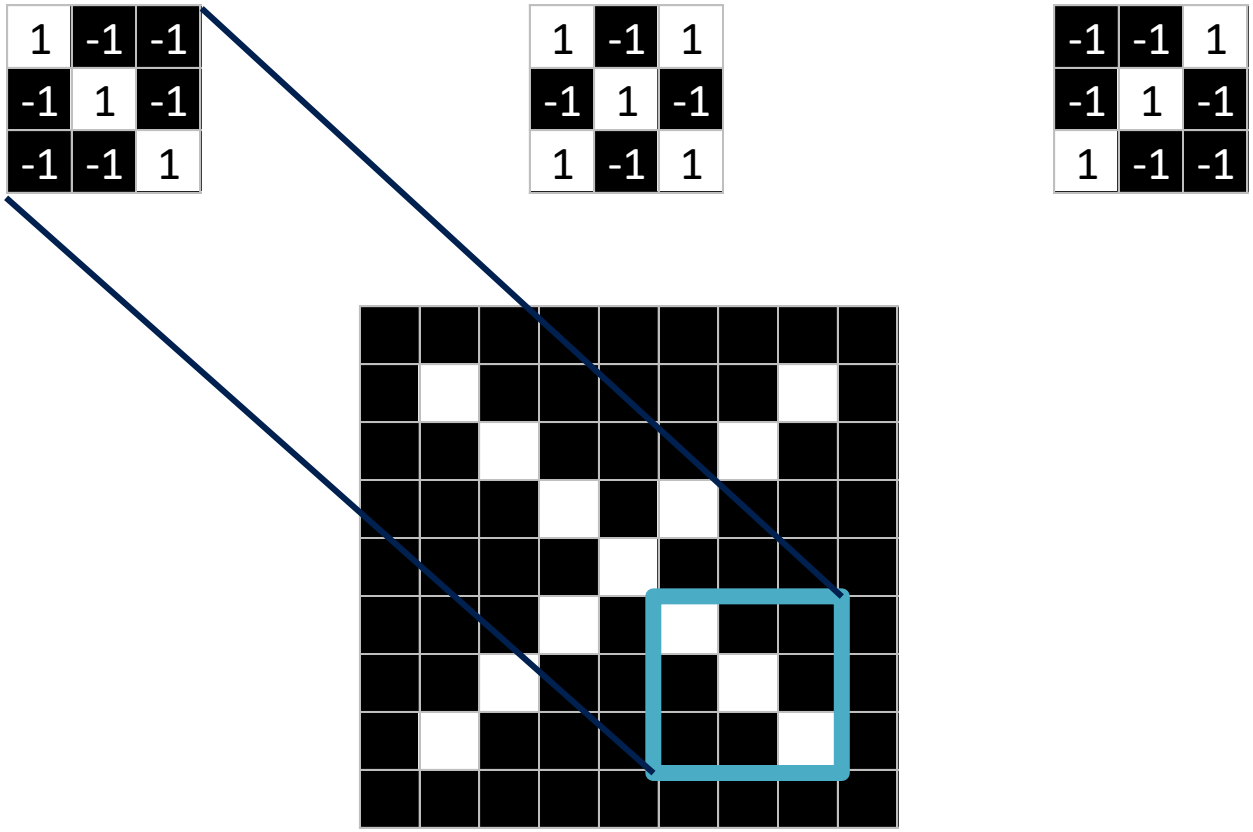
1	-1	-1
-1	1	-1
-1	-1	1

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

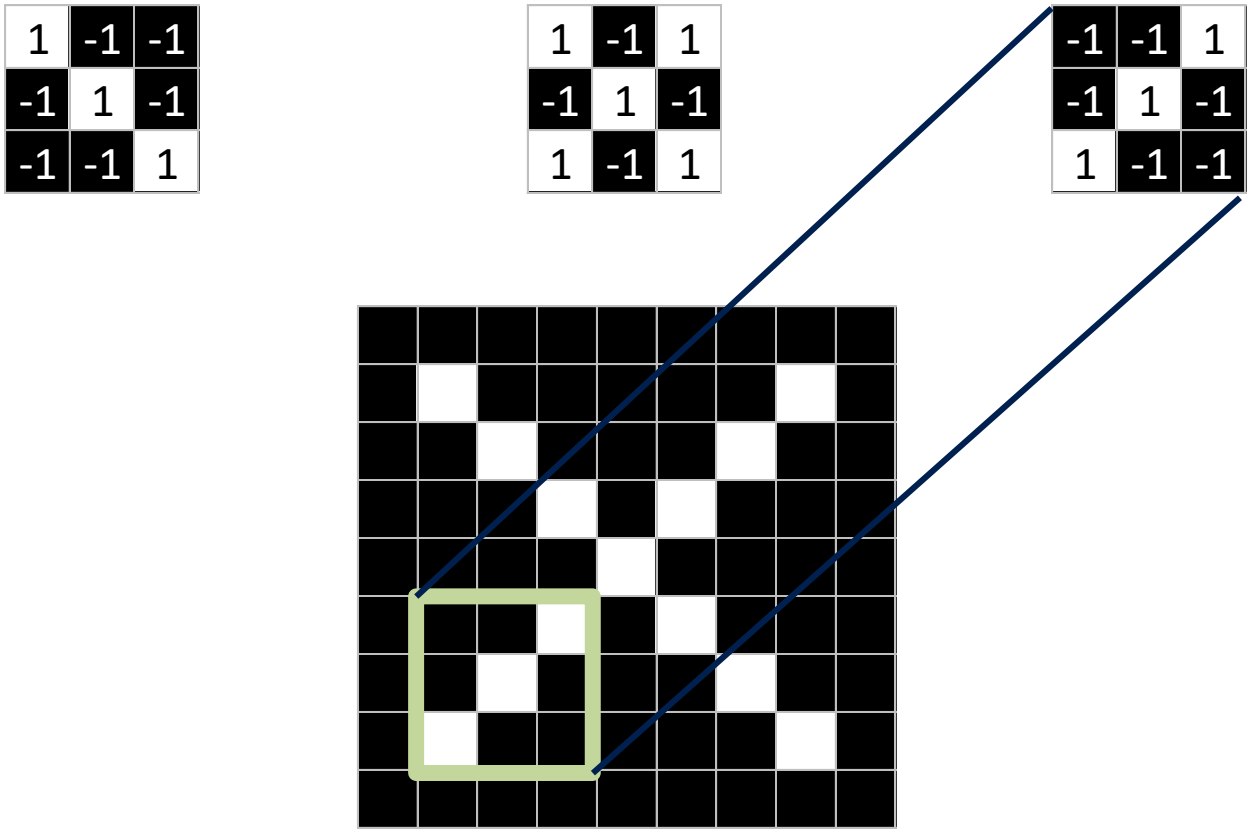
-1	-1	1
-1	1	-1
1	-1	-1



źródło: http://brohrer.github.io/blog.html

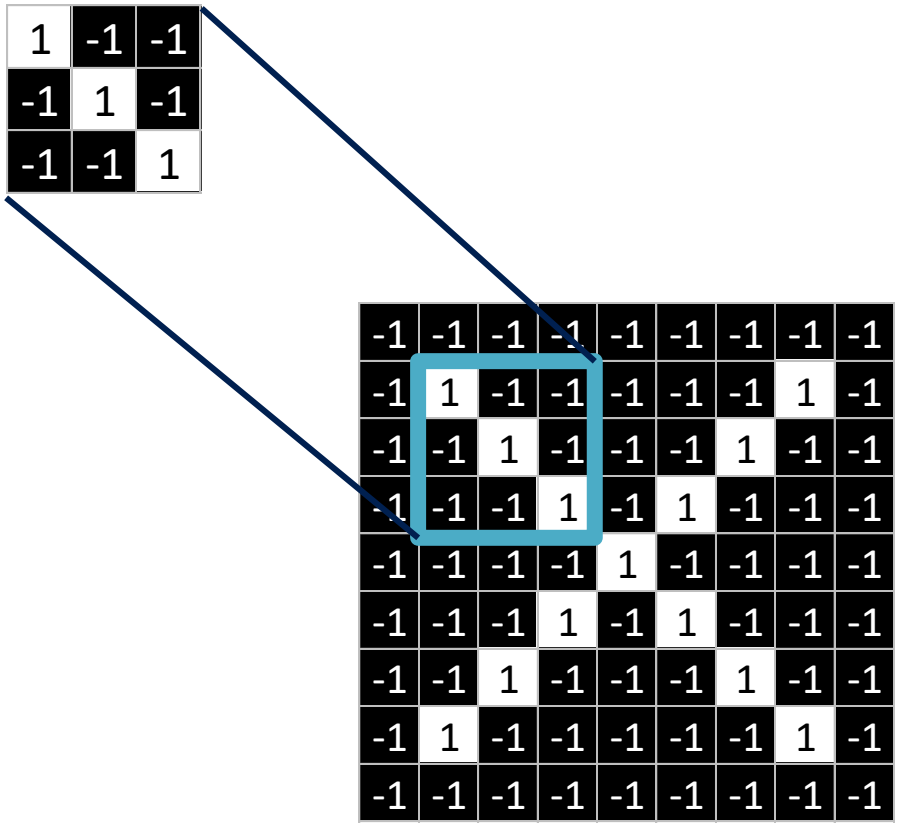


źródło: http://brohrer.github.io/blog.html



źródło: http://brohrer.github.io/blog.html

Filtering: The math behind the match



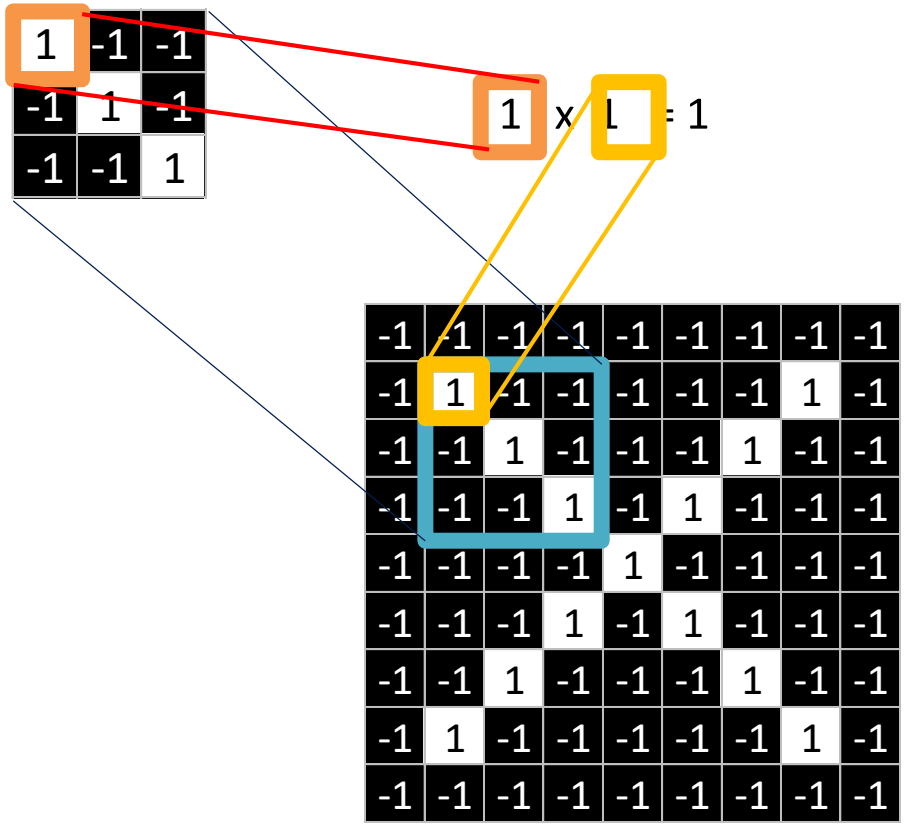
źródło: http://brohrer.github.io/blog.html

Filtering: The math behind the match

1. Line up the feature and the image patch.
2. Multiply each image pixel by the corresponding feature pixel.
3. Add them up.
4. Divide by the total number of pixels in the feature.

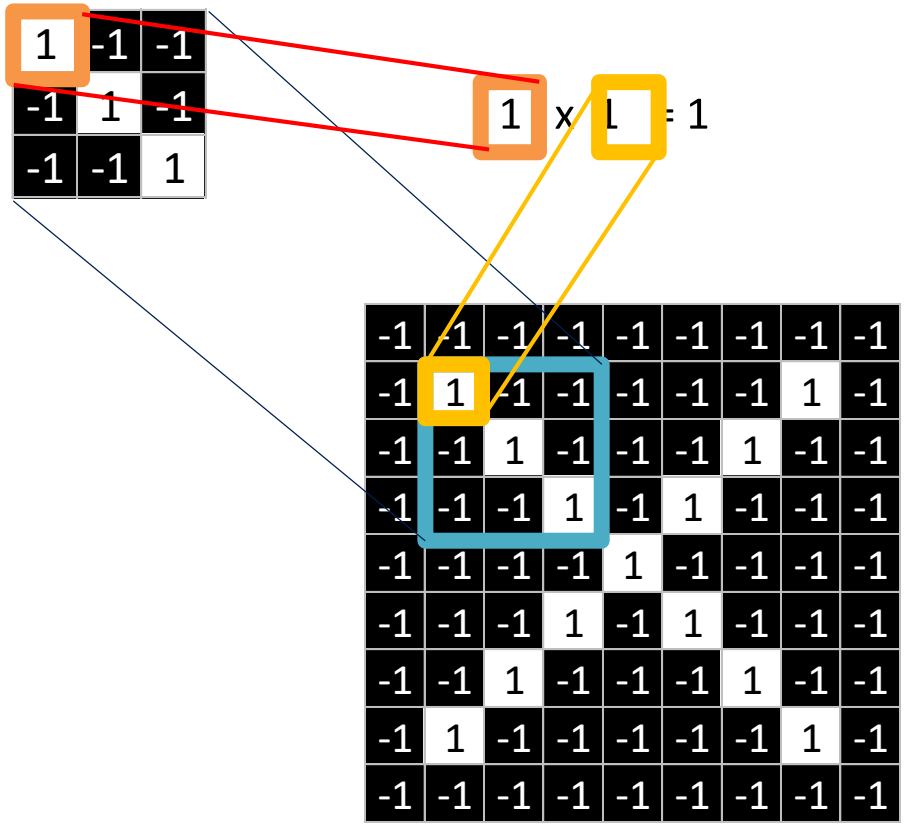
źródło: http://brohrer.github.io/blog.html

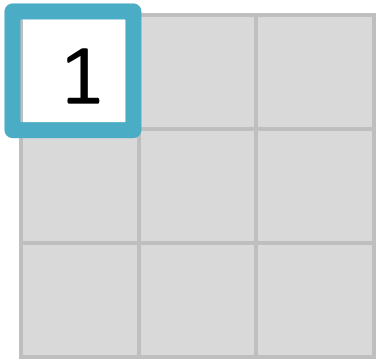
Filtering: The math behind the match



źródło: http://brohrer.github.io/blog.html

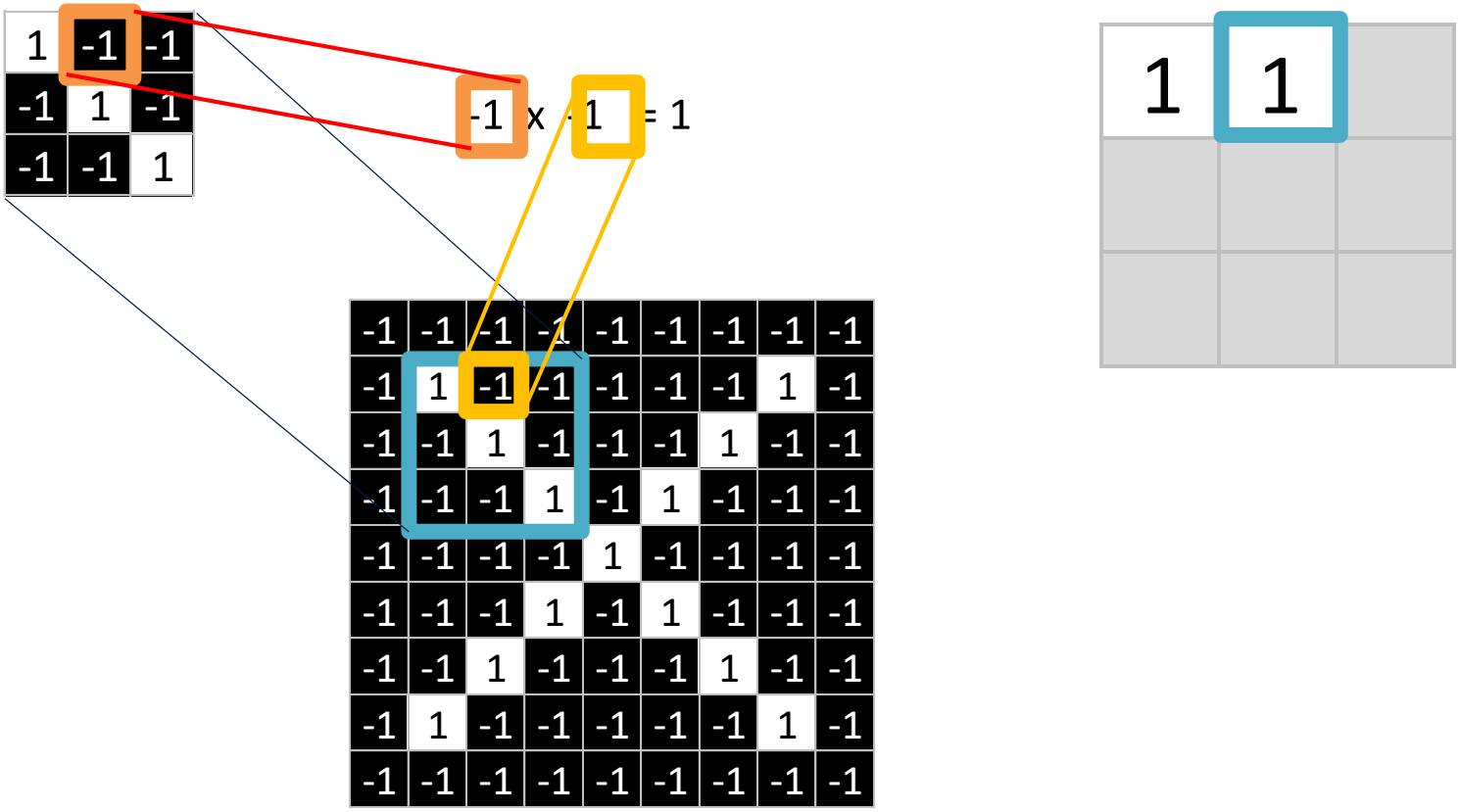
Filtering: The math behind the match





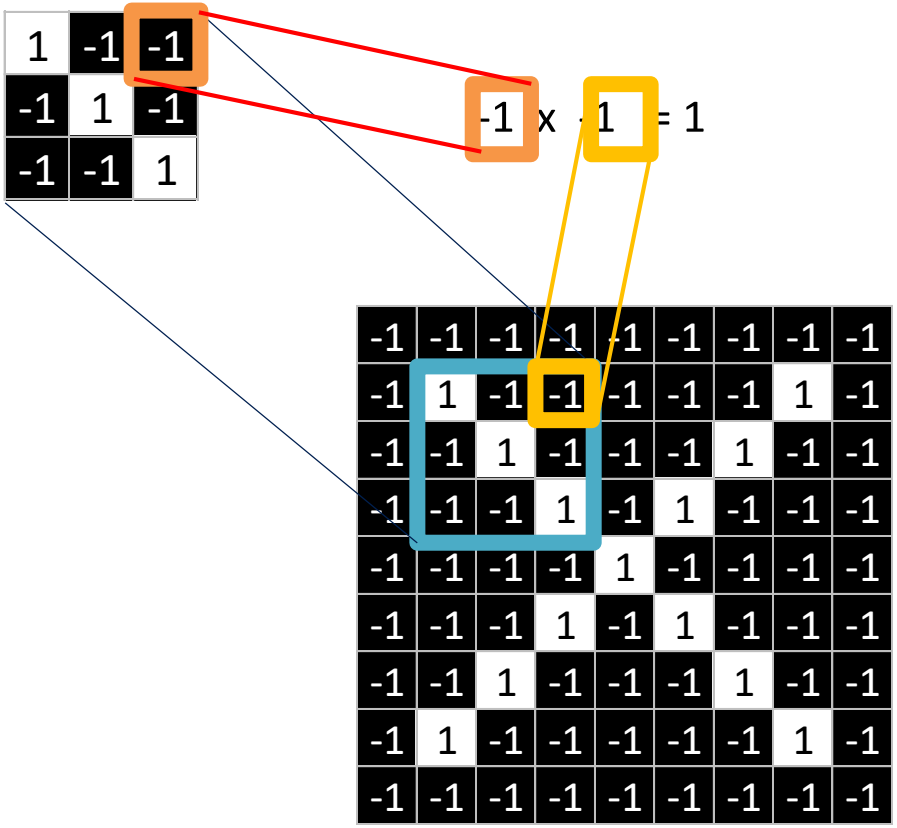
źródło: http://brohrer.github.io/blog.html

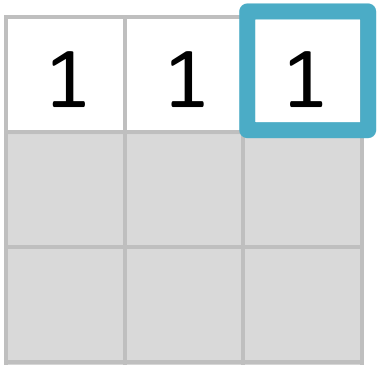
Filtering: The math behind the match



źródło: http://brohrer.github.io/blog.html

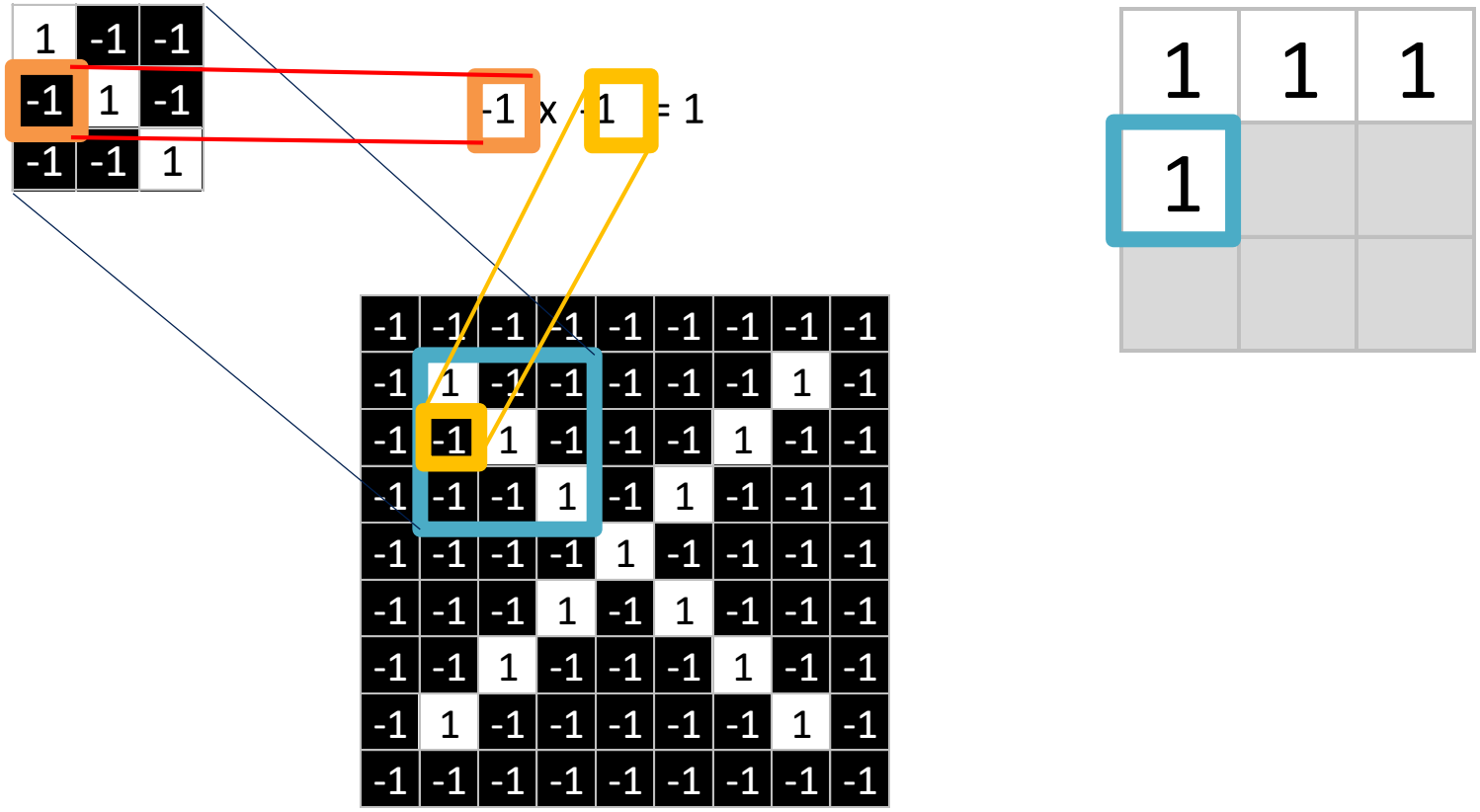
Filtering: The math behind the match





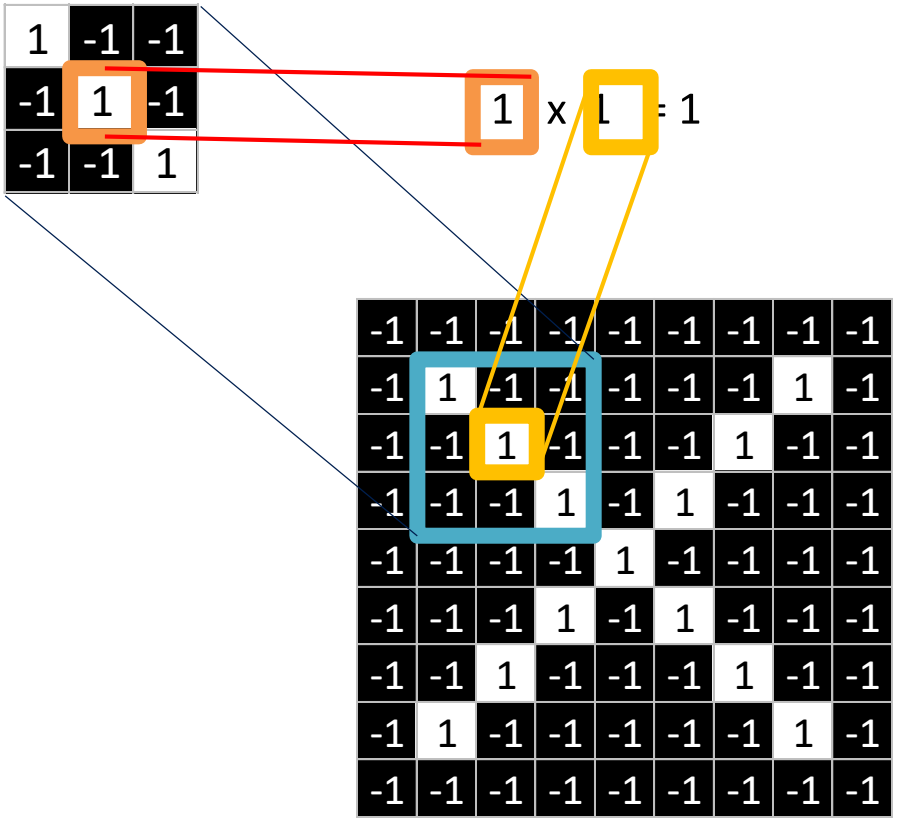
źródło: http://brohrer.github.io/blog.html

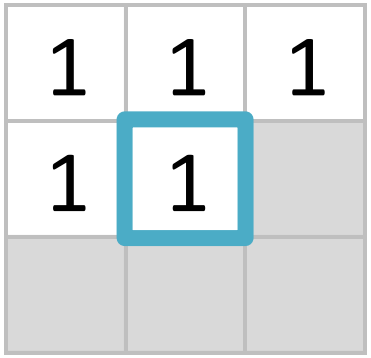
Filtering: The math behind the match



źródło: http://brohrer.github.io/blog.html

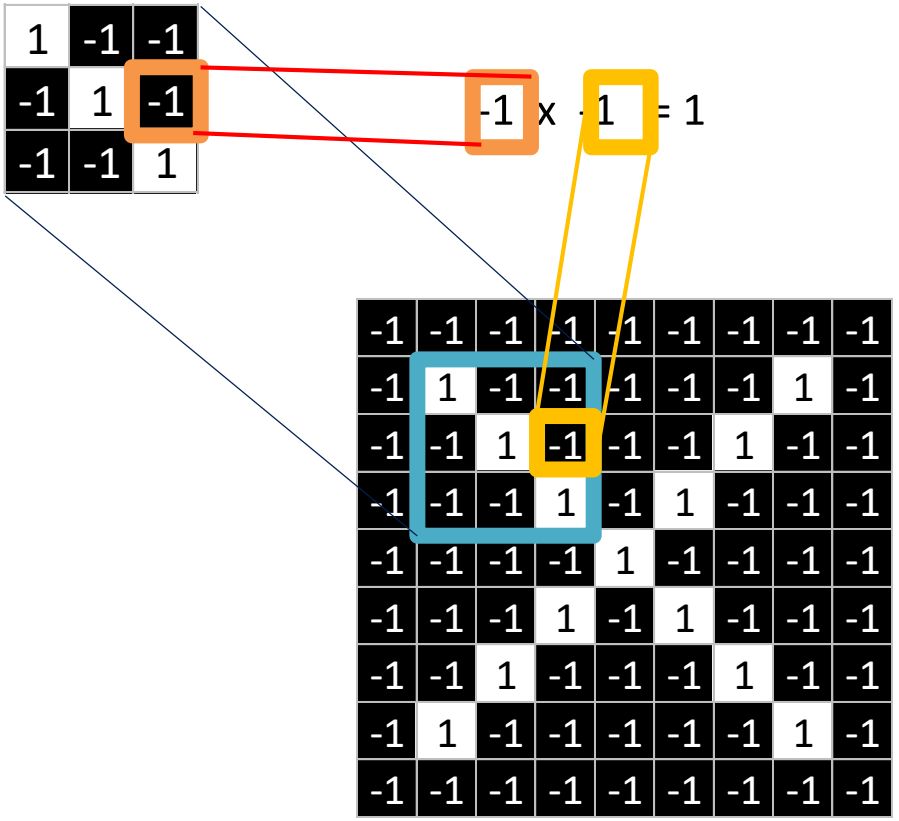
Filtering: The math behind the match

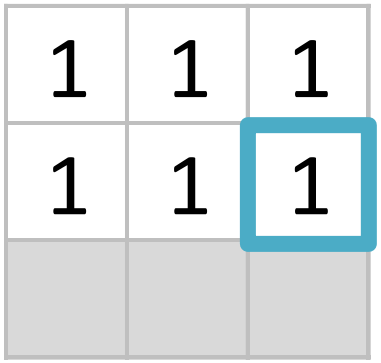




źródło: http://brohrer.github.io/blog.html

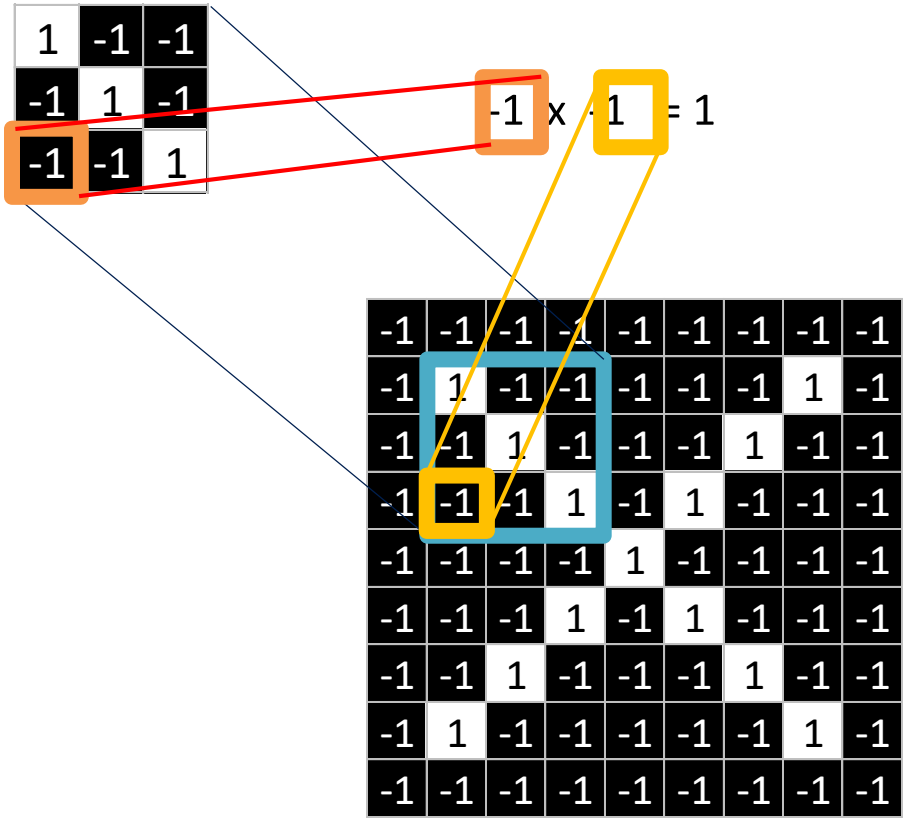
Filtering: The math behind the match

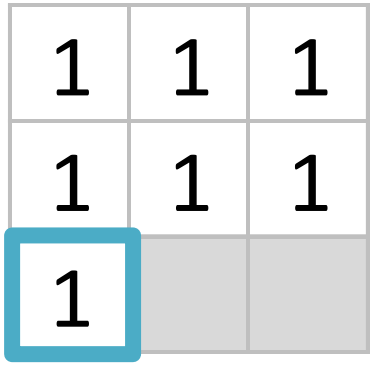




źródło: http://brohrer.github.io/blog.html

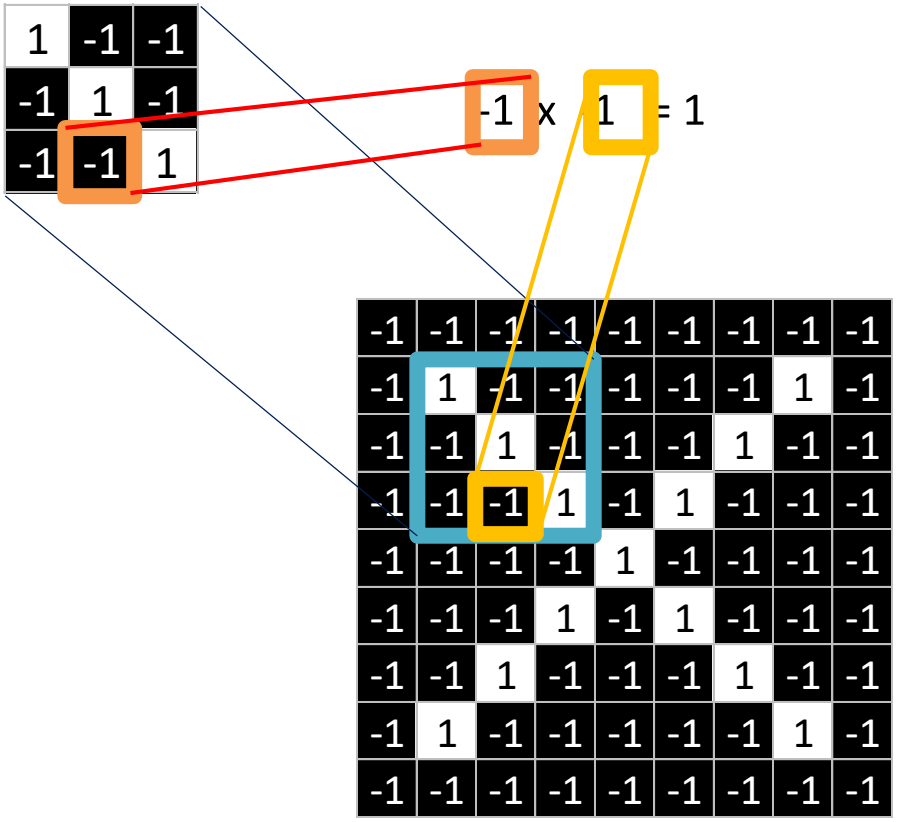
Filtering: The math behind the match

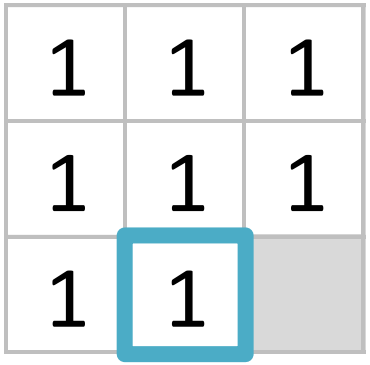




źródło: http://brohrer.github.io/blog.html

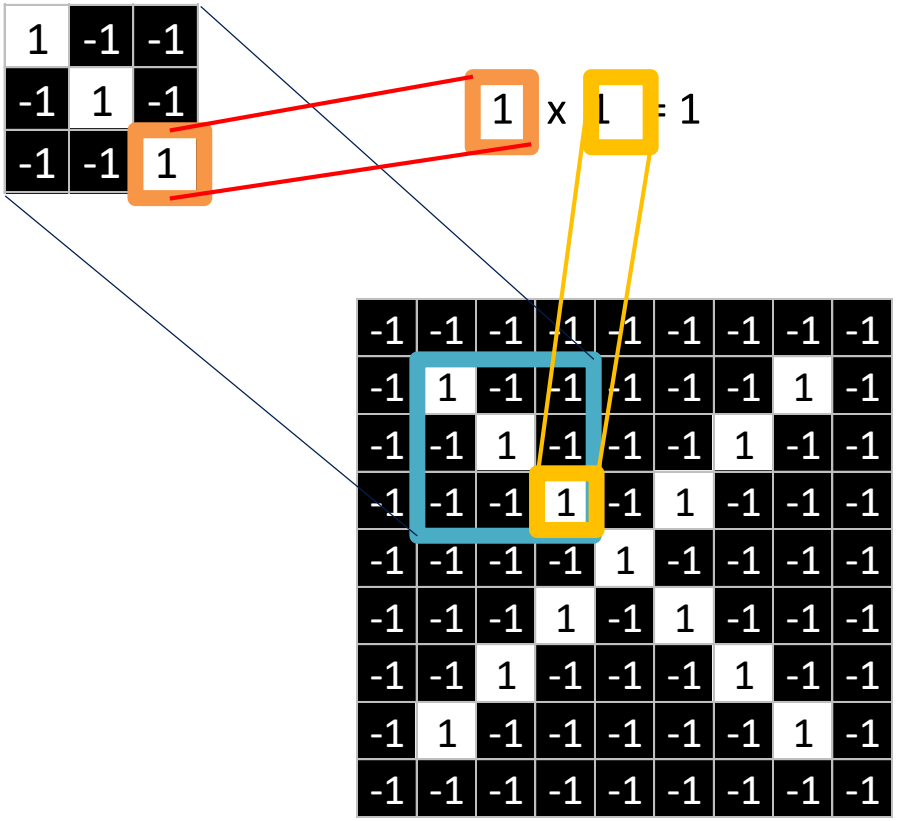
Filtering: The math behind the match

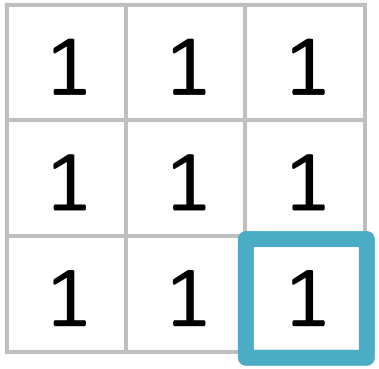




źródło: http://brohrer.github.io/blog.html

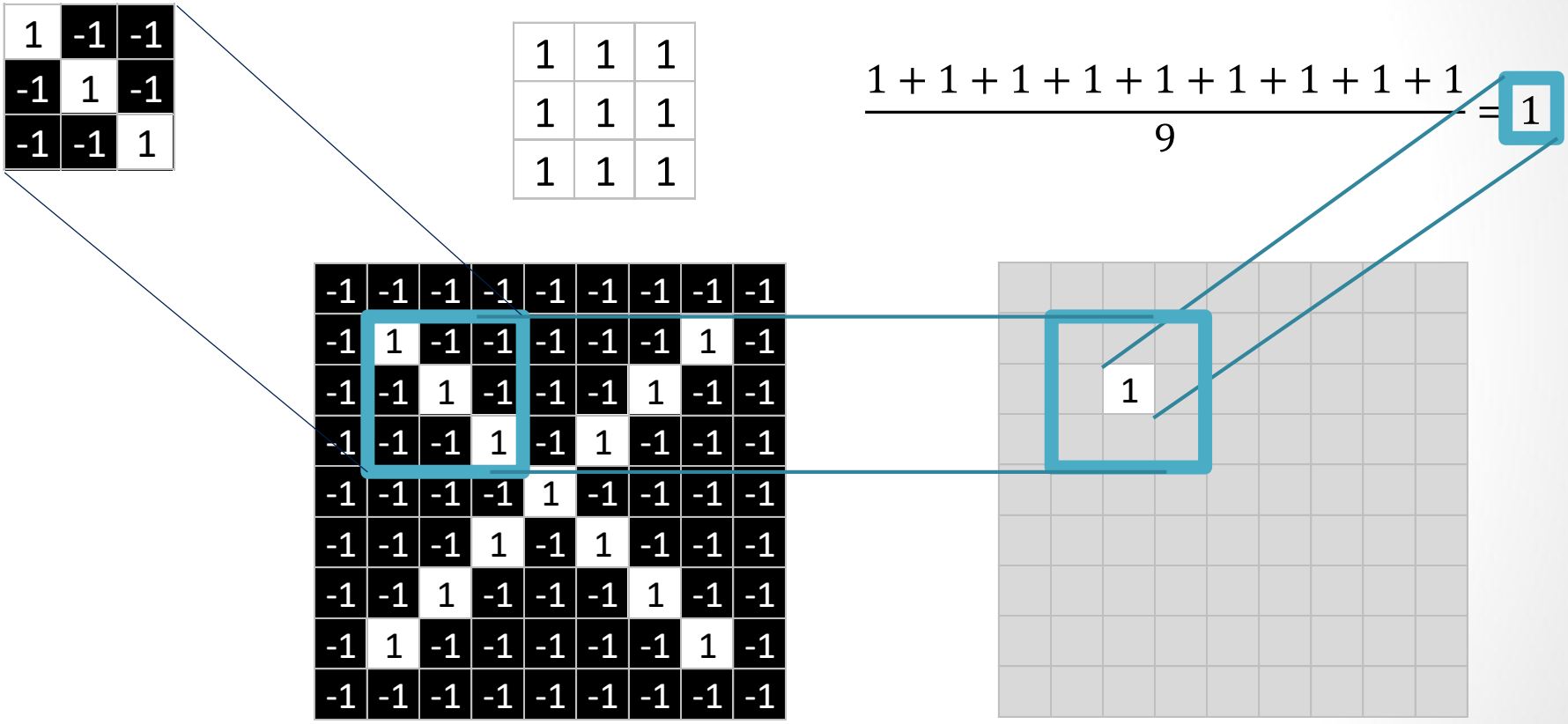
Filtering: The math behind the match





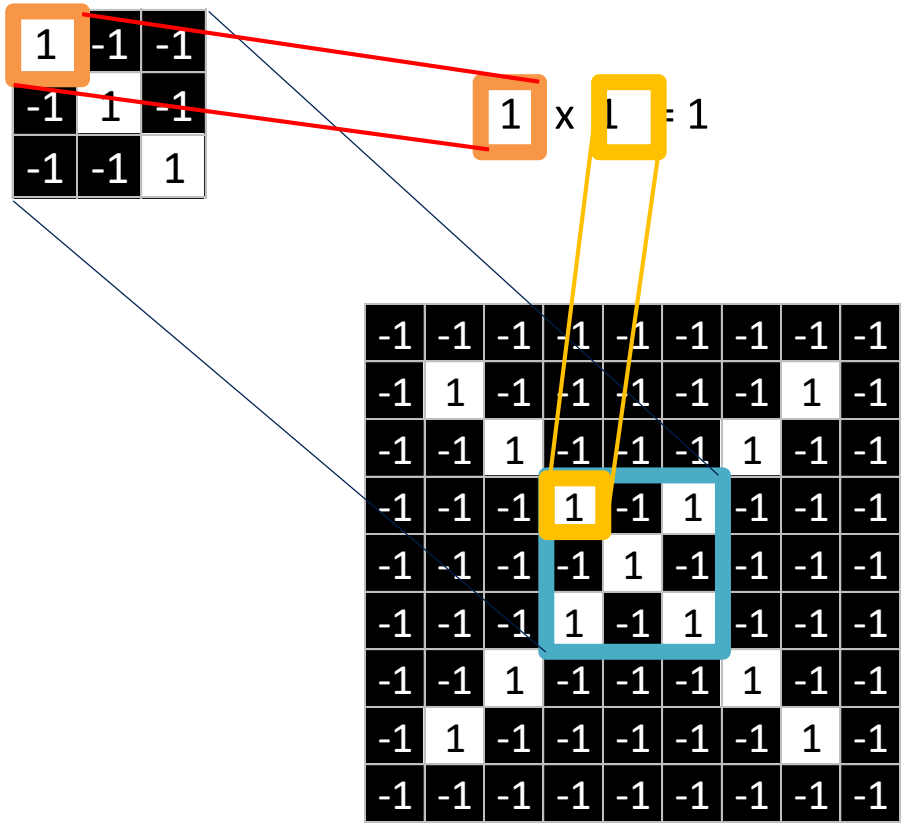
źródło: http://brohrer.github.io/blog.html

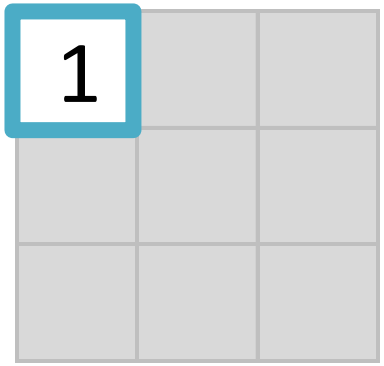
Filtering: The math behind the match



źródło: http://brohrer.github.io/blog.html

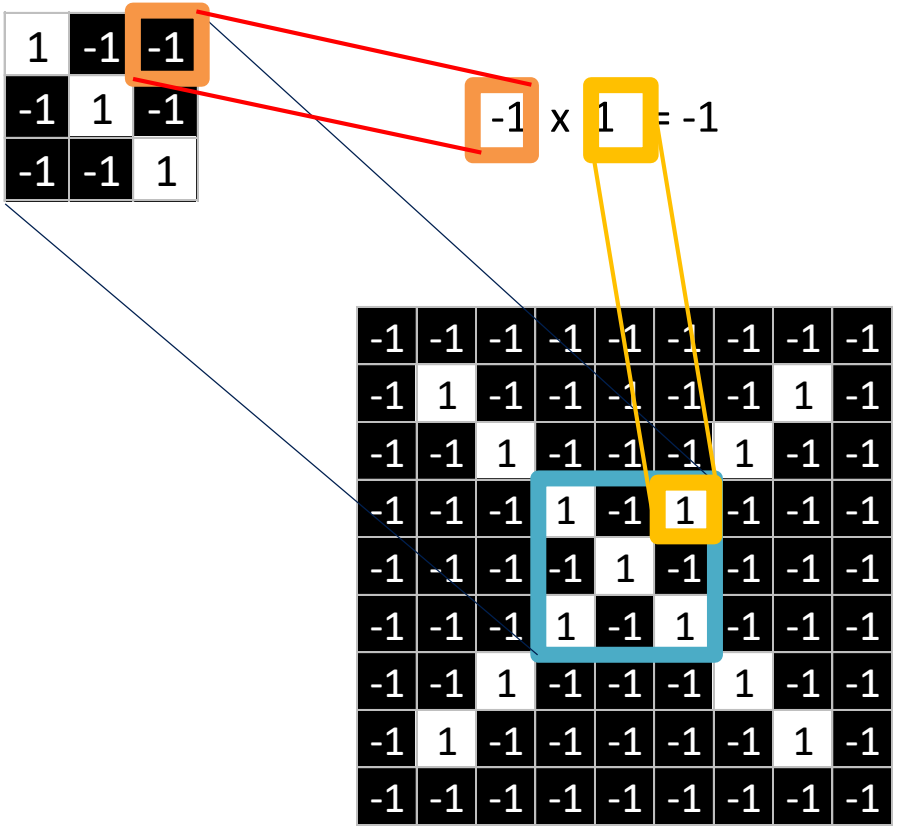
Filtering: The math behind the match

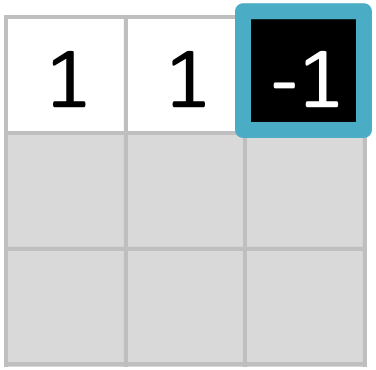




źródło: http://brohrer.github.io/blog.html

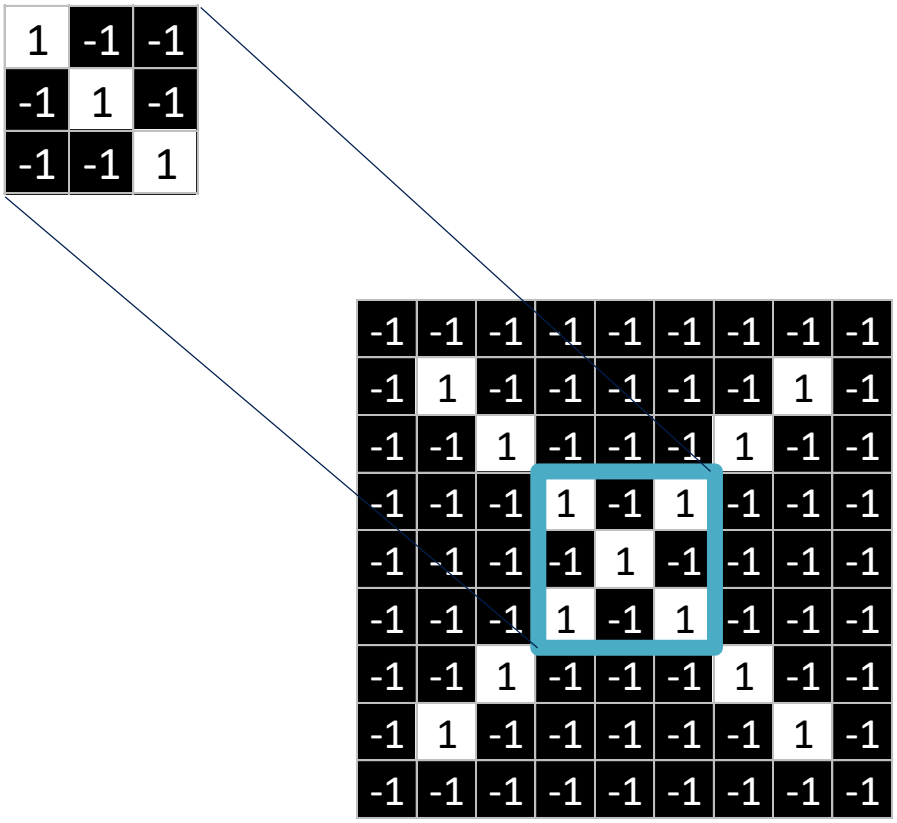
Filtering: The math behind the match





źródło: http://brohrer.github.io/blog.html

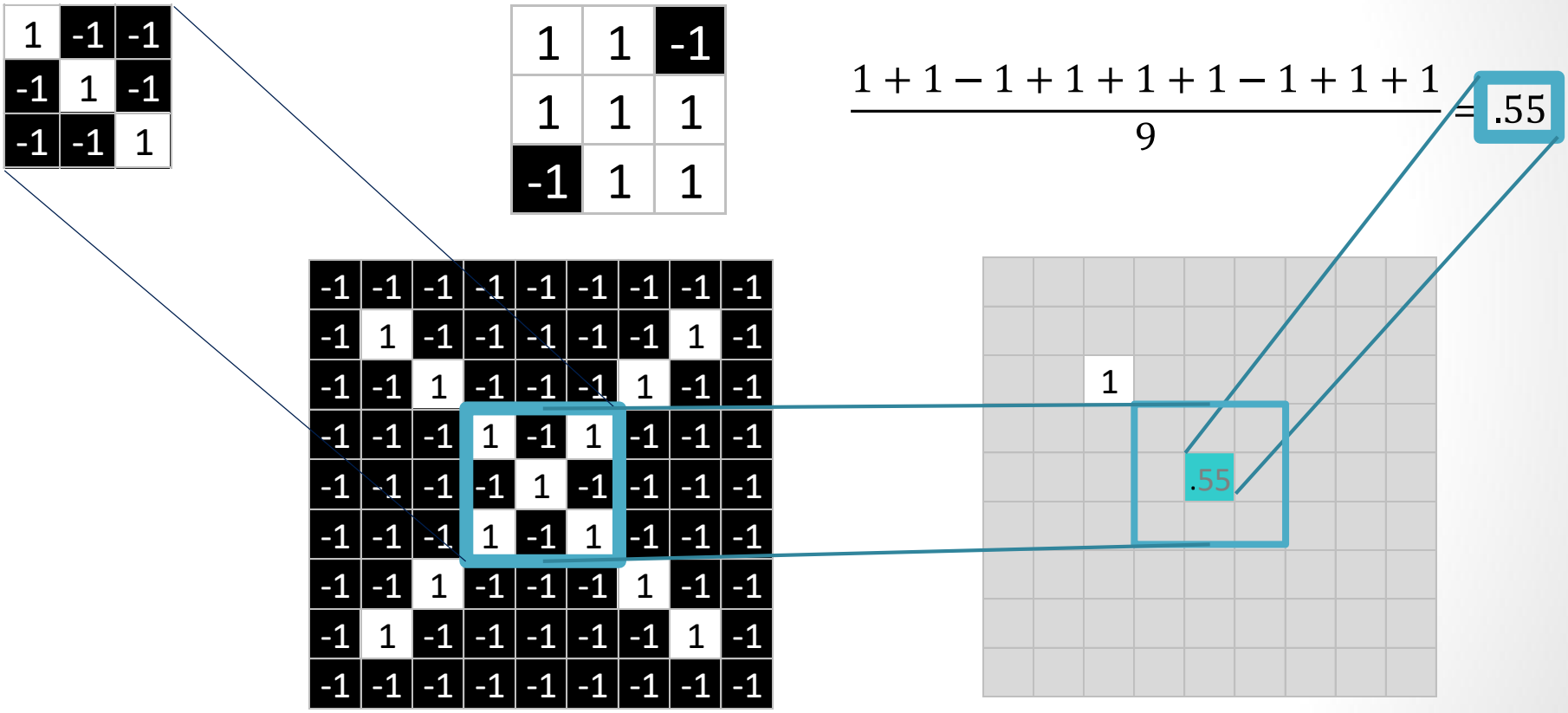
Filtering: The math behind the match



1	1	-1
1	1	1
-1	1	1

źródło: http://brohrer.github.io/blog.html

Filtering: The math behind the match



źródło: http://brohrer.github.io/blog.html

Convolution: Trying every possible match

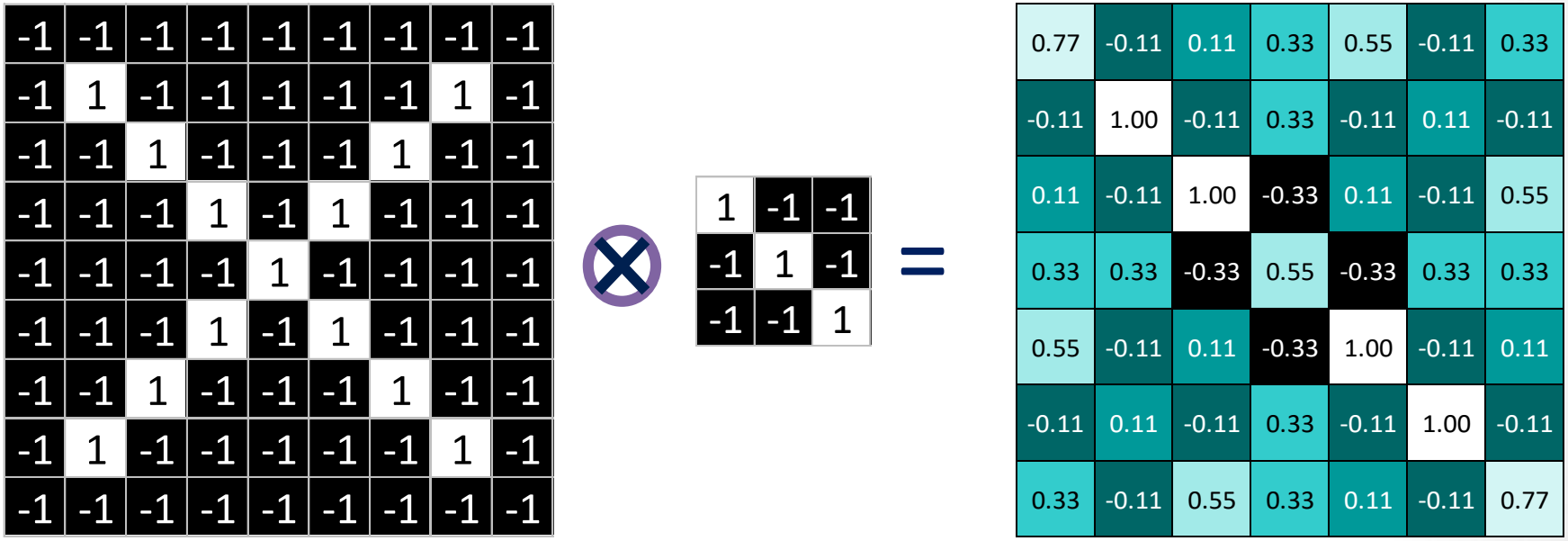
1	-1	-1
-1	1	-1
-1	-1	1

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

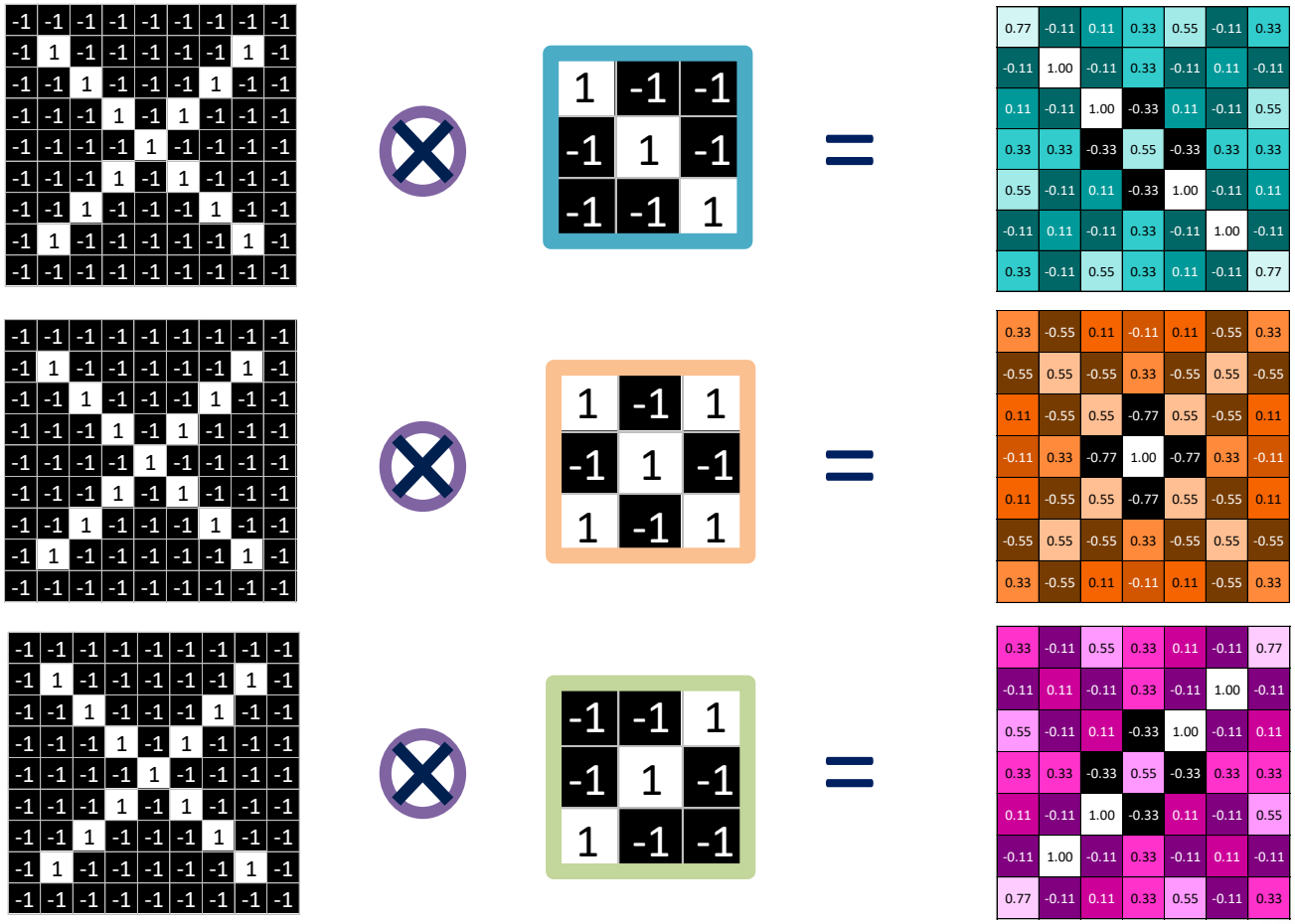
0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

źródło: http://brohrer.github.io/blog.html

Convolution: Trying every possible match



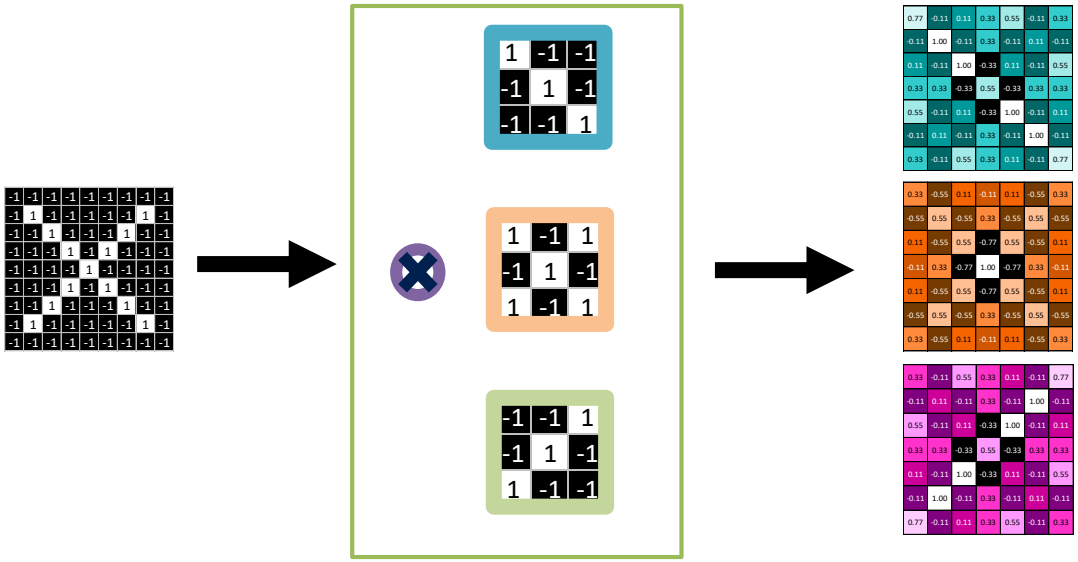
źródło: http://brohrer.github.io/blog.html



źródło: http://brohrer.github.io/blog.html

Convolution layer

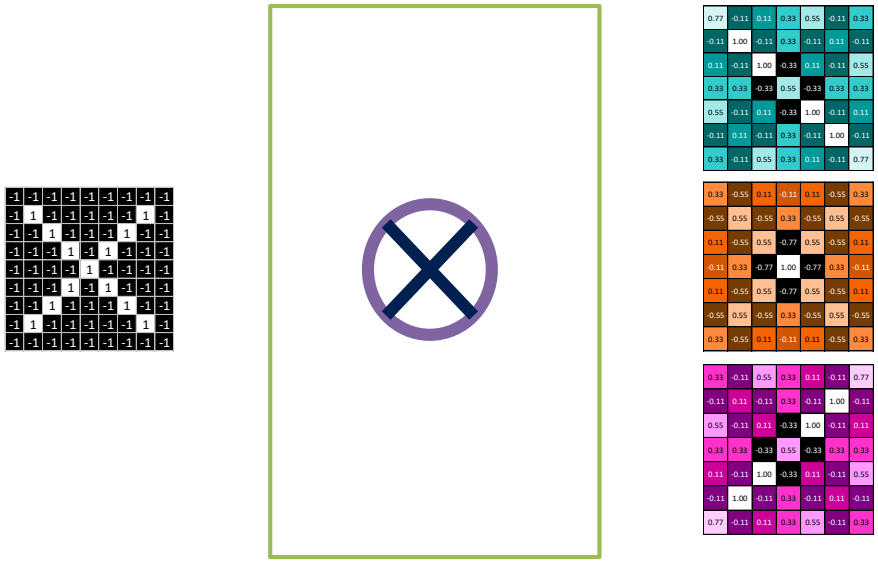
One image becomes a stack of filtered images



źródło: http://brohrer.github.io/blog.html

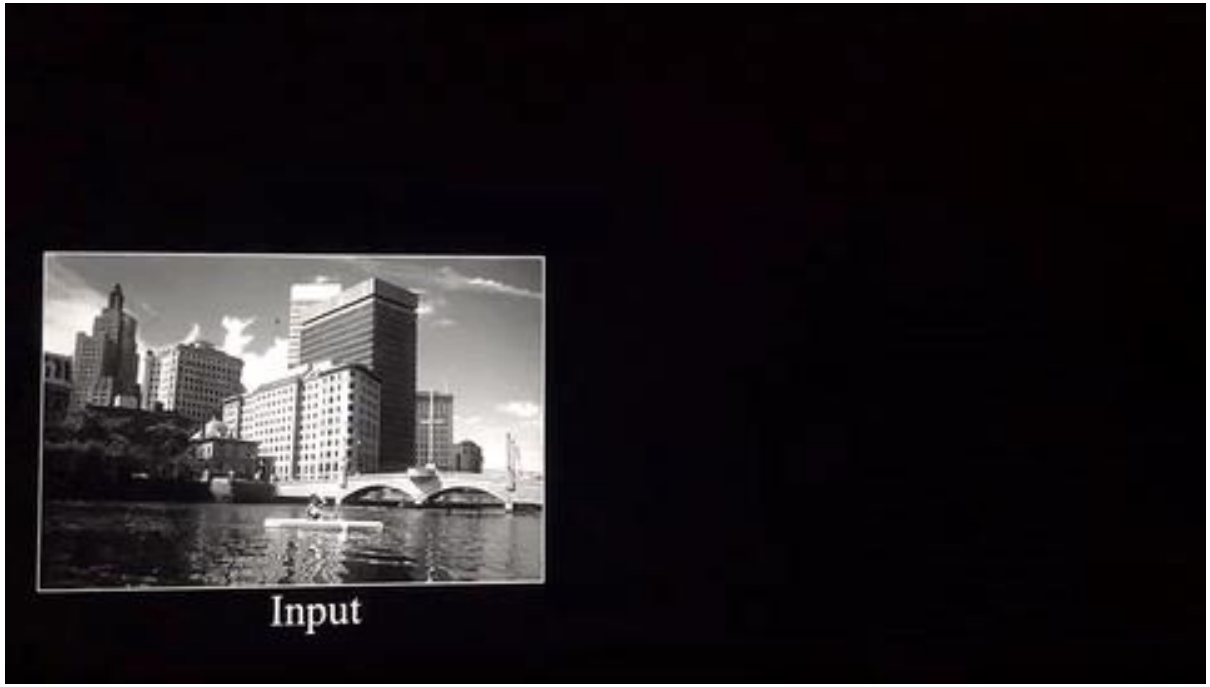
Convolution layer

One image becomes a stack of filtered images



źródło: http://brohrer.github.io/blog.html

Convolution layer

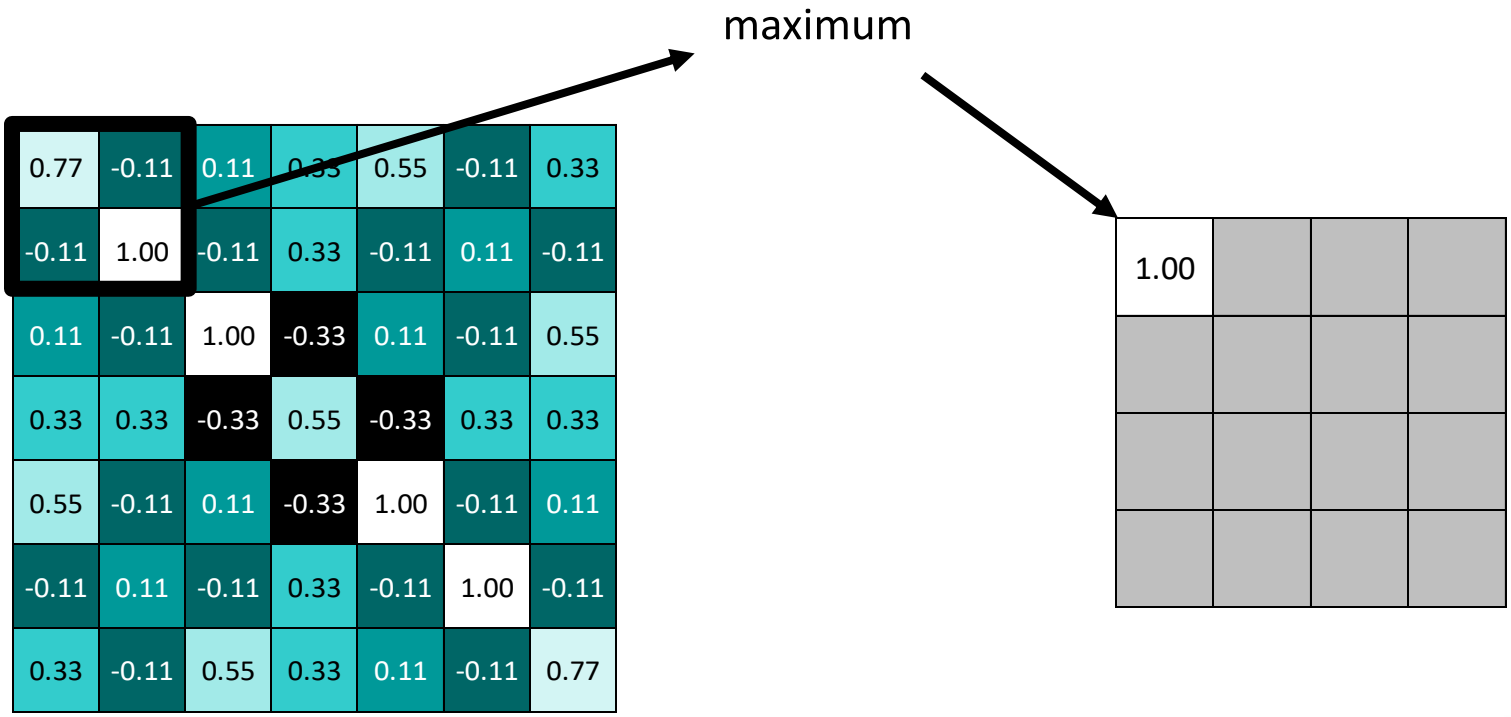


Pooling: Shrinking the image stack

1. Pick a window size (usually 2 or 3).
2. Pick a stride (usually 2).
3. Walk your window across your filtered images.
4. From each window, take the maximum value.

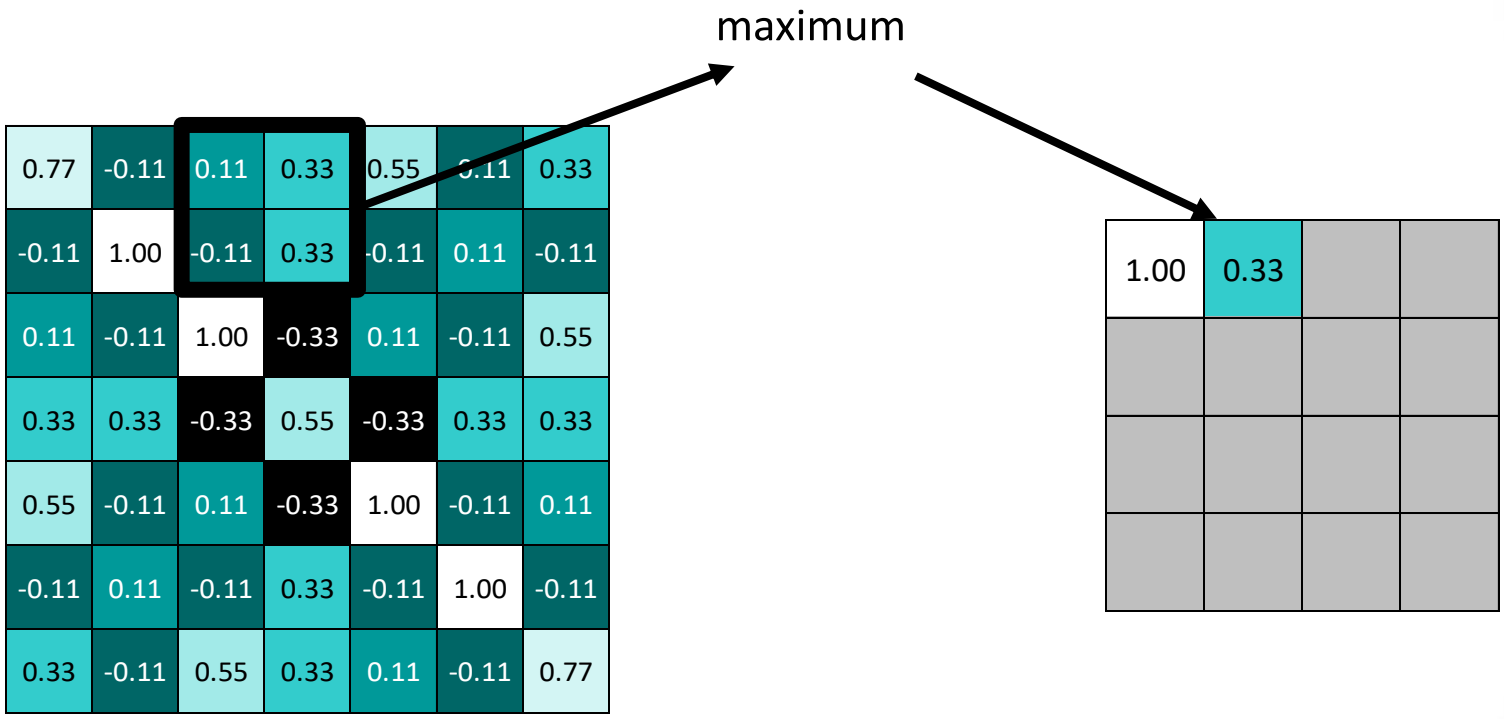
źródło: http://brohrer.github.io/blog.html

Pooling



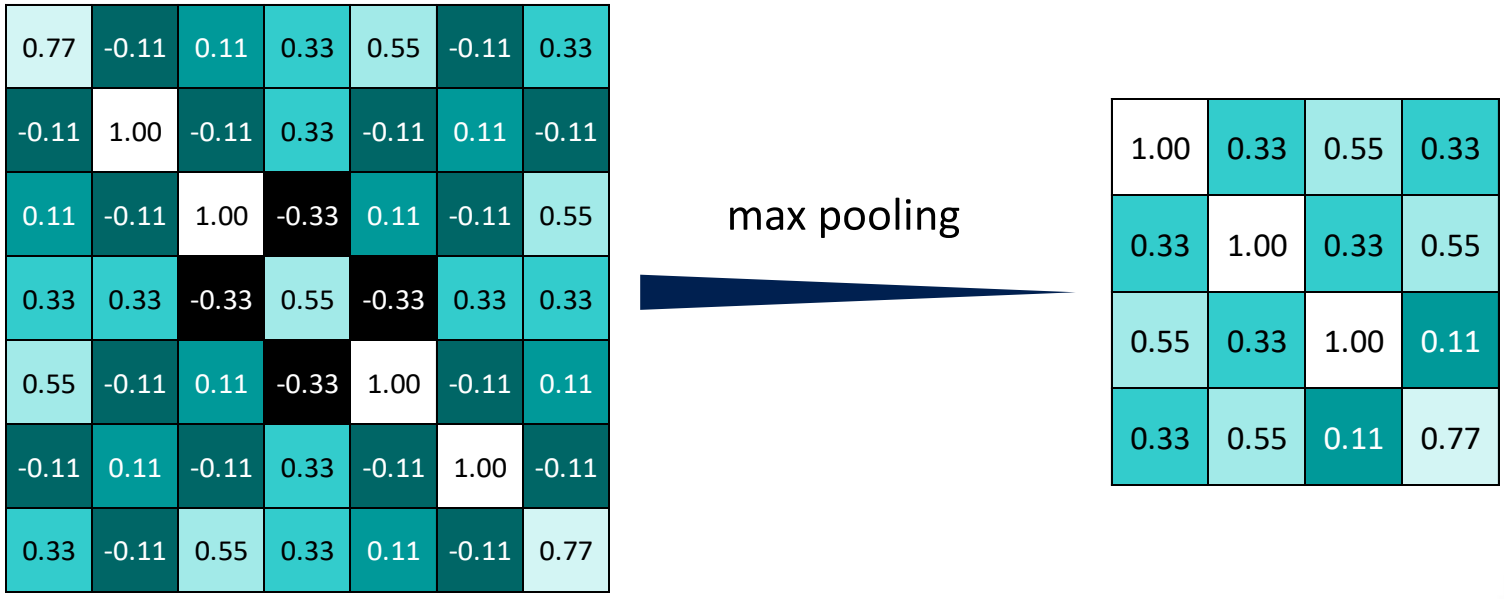
źródło: http://brohrer.github.io/blog.html

Pooling



źródło: http://brohrer.github.io/blog.html

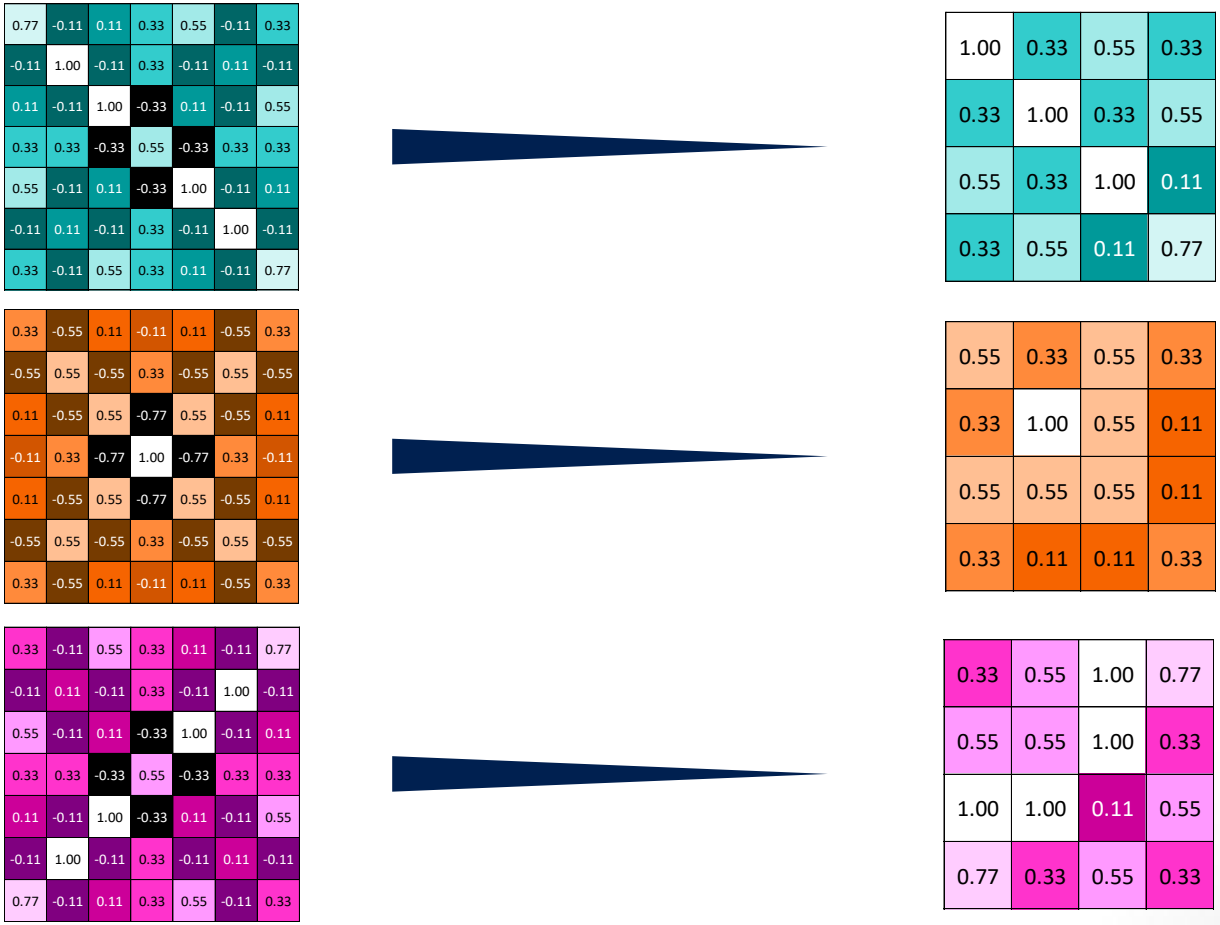
Pooling



źródło: http://brohrer.github.io/blog.html

Pooling layer

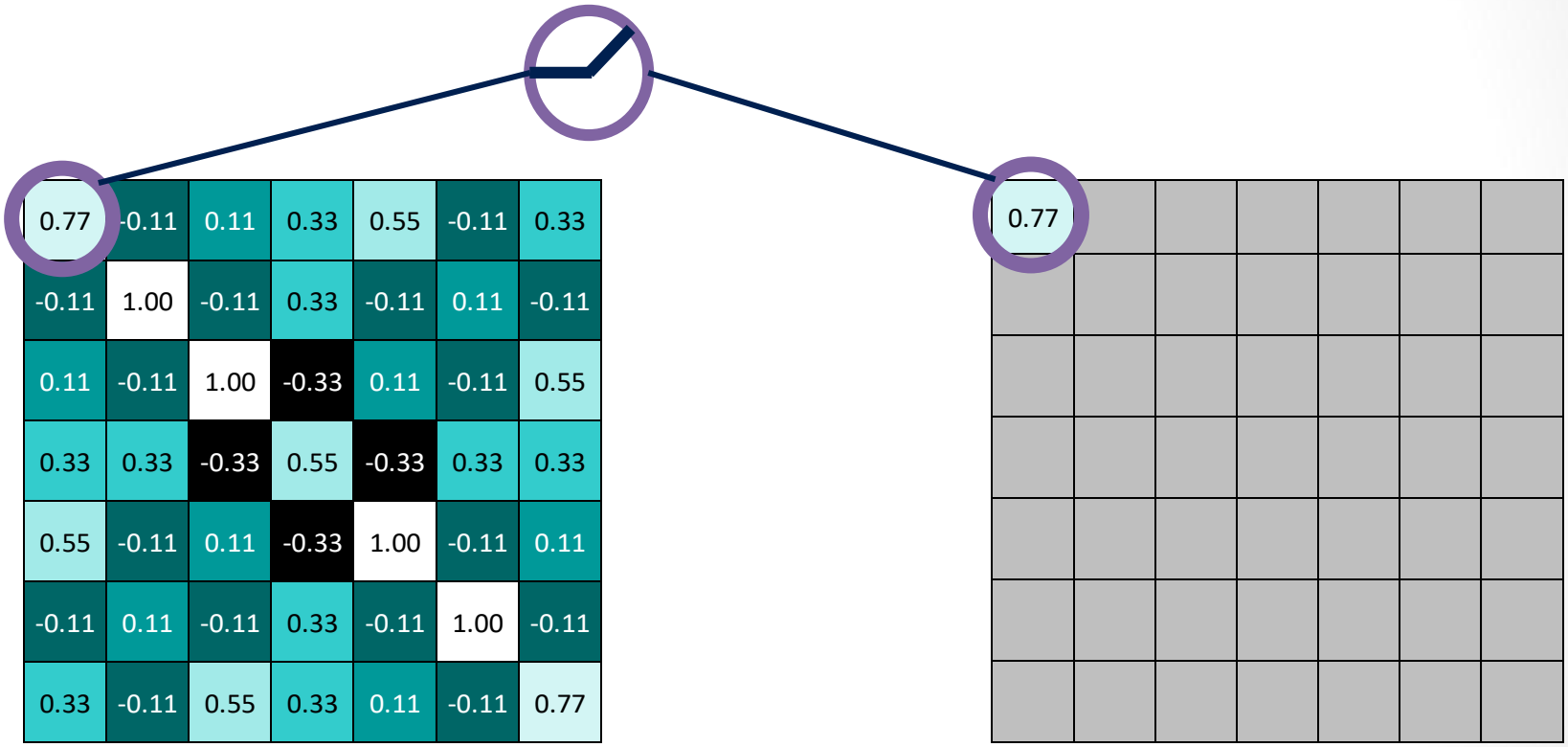
A stack of images becomes a stack of smaller images



źródło: http://brohrer.github.io/blog.html

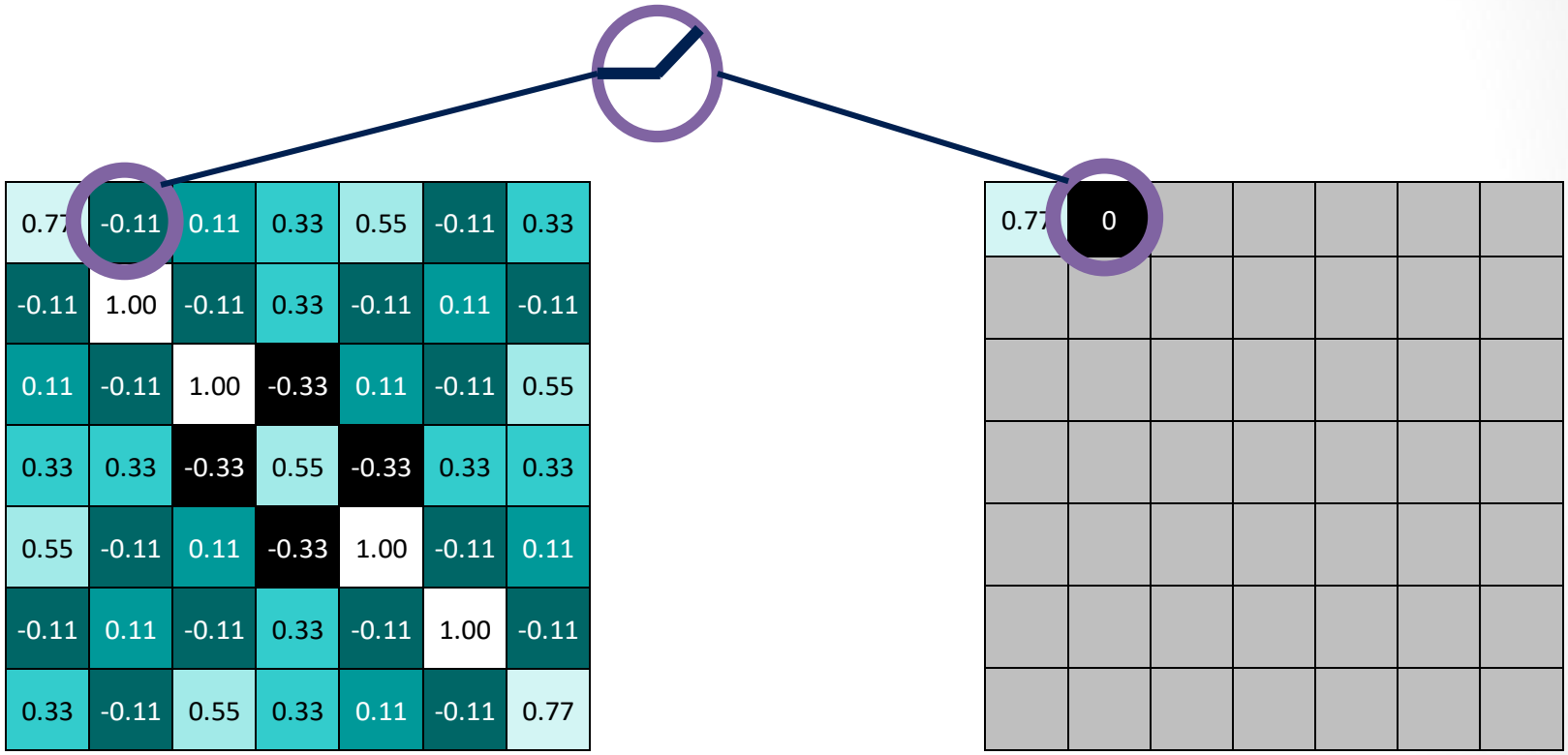
Rectified Linear Units (ReLUs)

Normalization: Change everything negative to zero



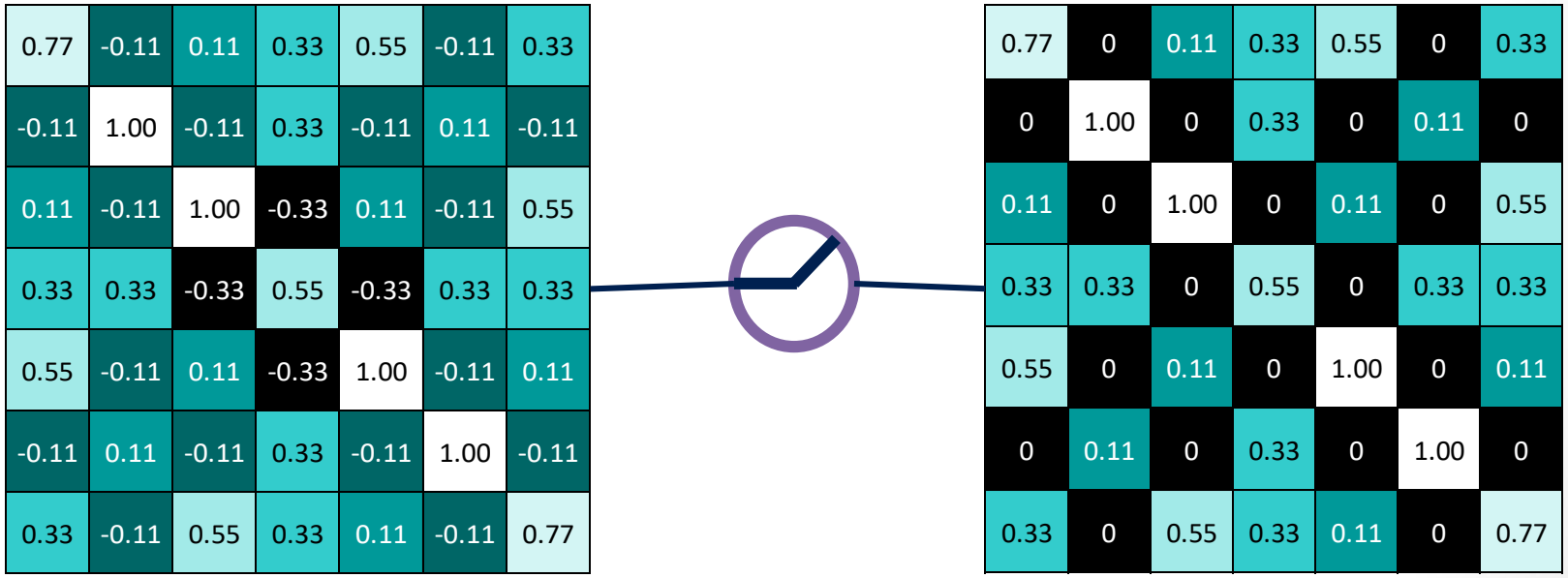
źródło: http://brohrer.github.io/blog.html

Rectified Linear Units (ReLUs)



źródło: http://brohrer.github.io/blog.html

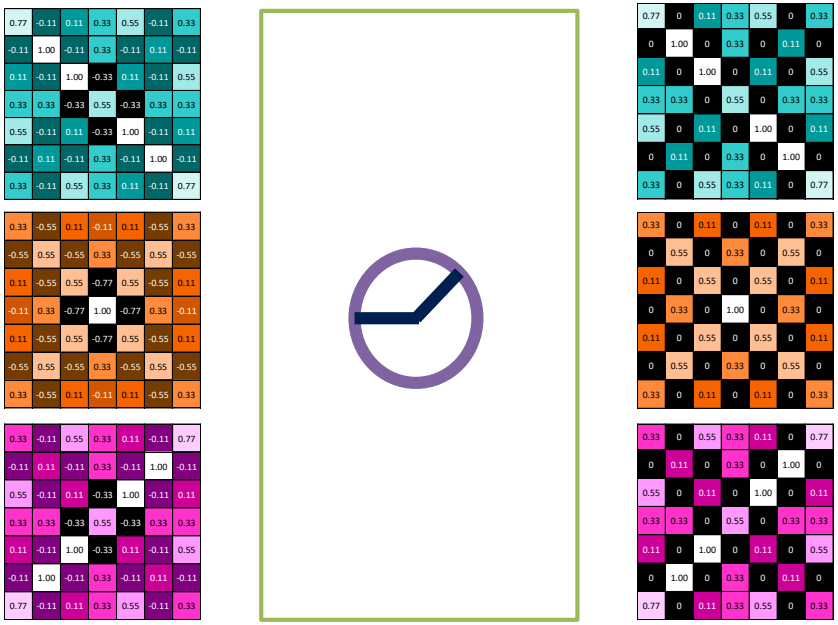
Rectified Linear Units (ReLUs)



źródło: http://brohrer.github.io/blog.html

ReLU layer

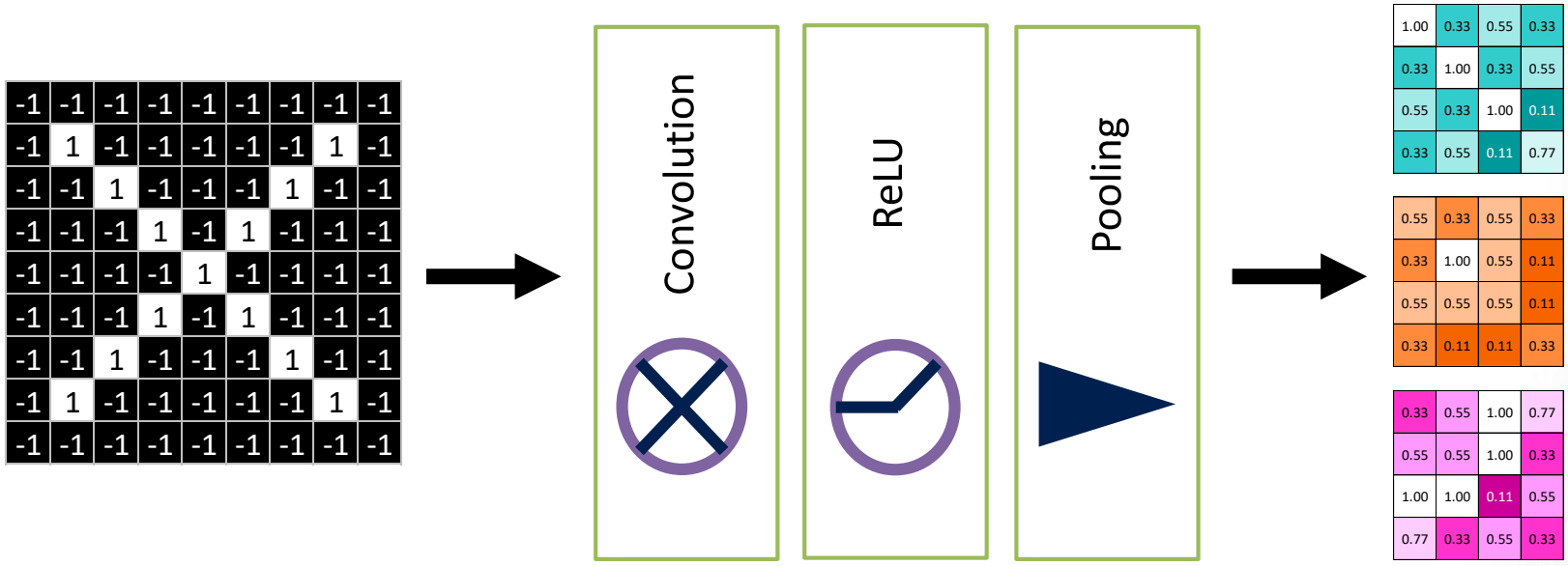
A stack of images becomes a stack of images with no negative values.



źródło: http://brohrer.github.io/blog.html

Layers get stacked

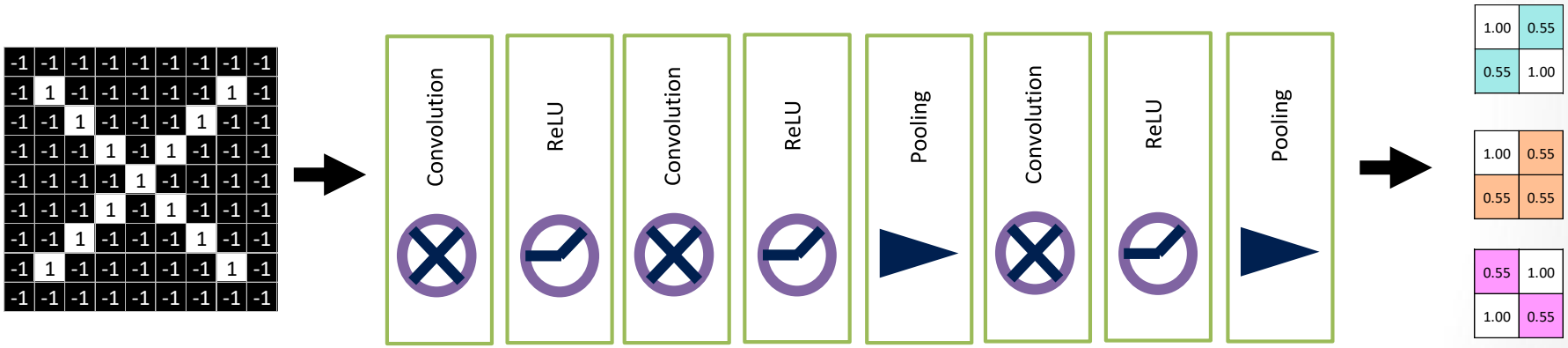
The output of one becomes the input of the next.



źródło: http://brohrer.github.io/blog.html

Deep stacking

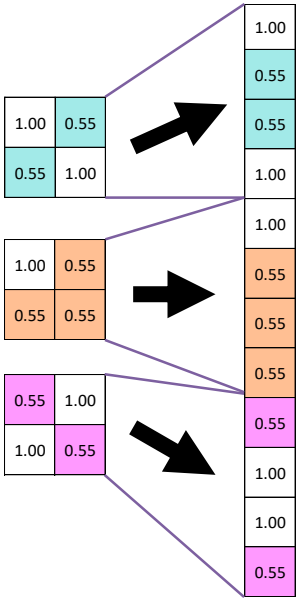
Layers can be repeated several (or many) times.



źródło: http://brohrer.github.io/blog.html

Fully connected layer

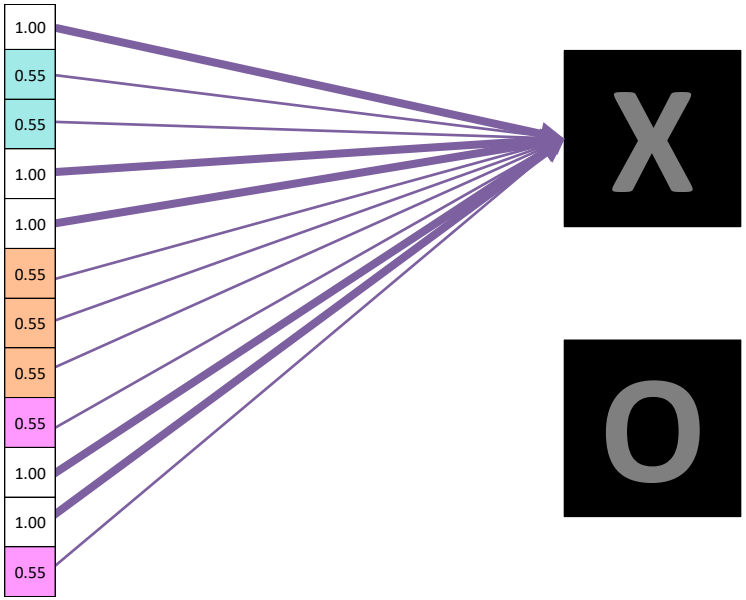
Every value gets a vote



źródło: http://brohrer.github.io/blog.html

Fully connected layer

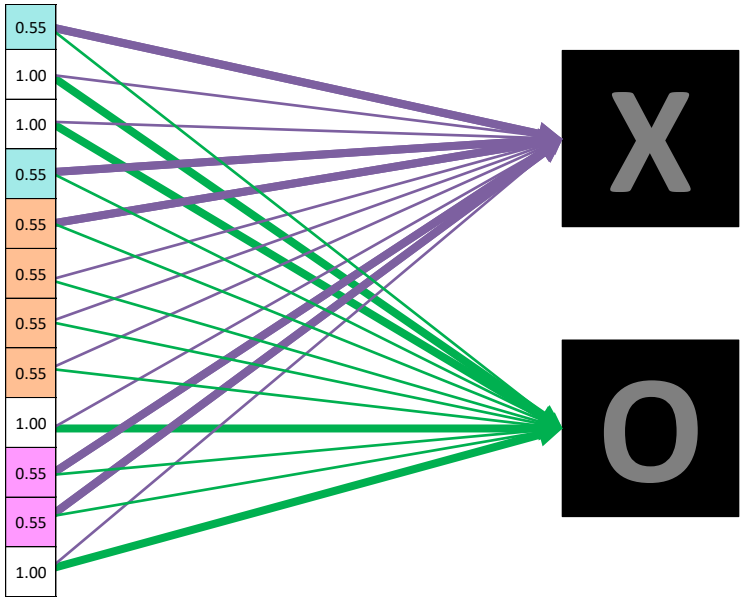
Vote depends on how strongly a value predicts X or O



źródło: http://brohrer.github.io/blog.html

Fully connected layer

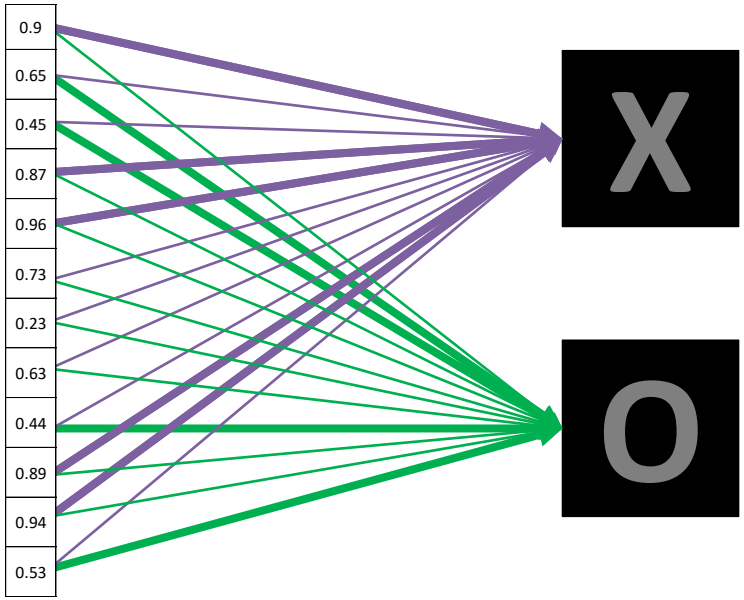
Vote depends on how strongly a value predicts X or O



źródło: http://brohrer.github.io/blog.html

Fully connected layer

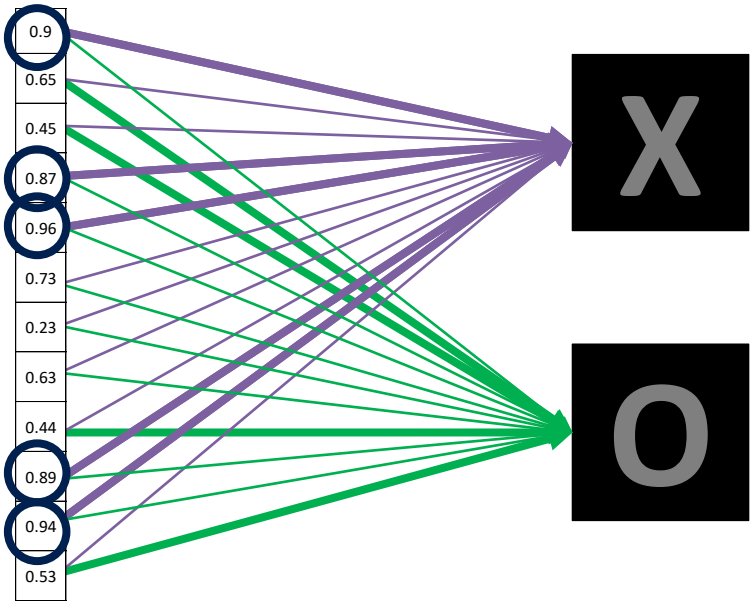
Future values vote on X or O



źródło: http://brohrer.github.io/blog.html

Fully connected layer

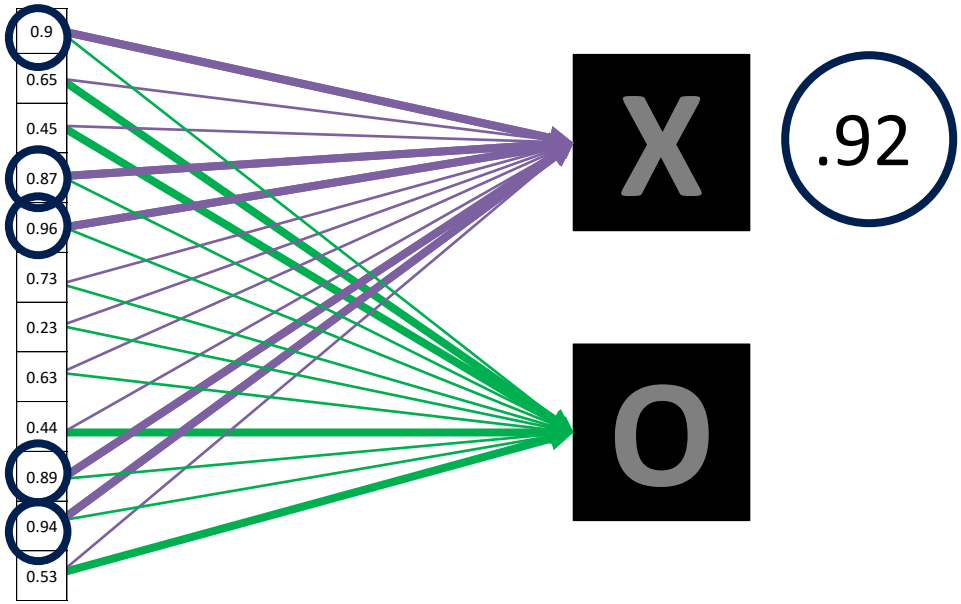
Future values vote on X or O



źródło: http://brohrer.github.io/blog.html

Fully connected layer

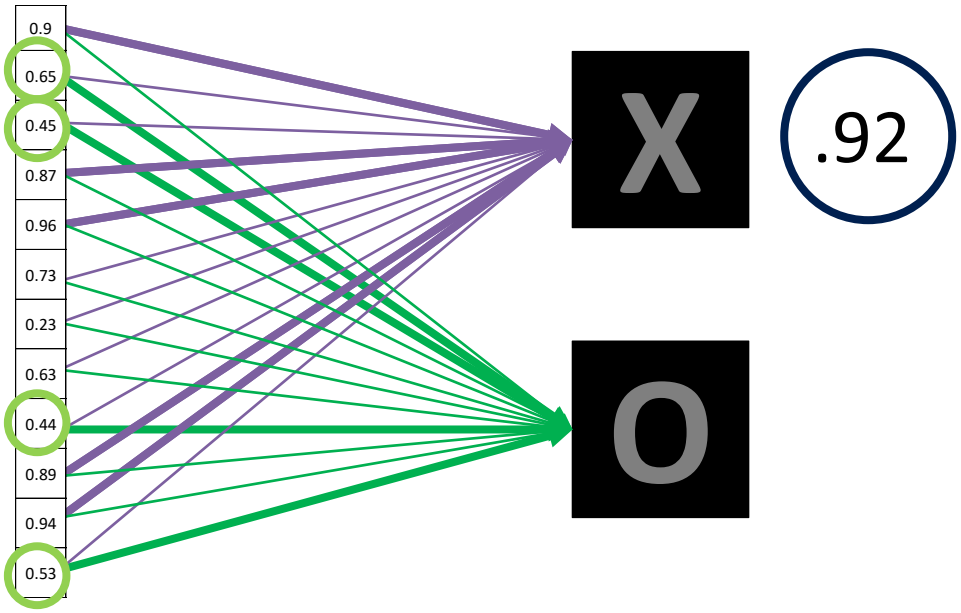
Future values vote on X or O



źródło: http://brohrer.github.io/blog.html

Fully connected layer

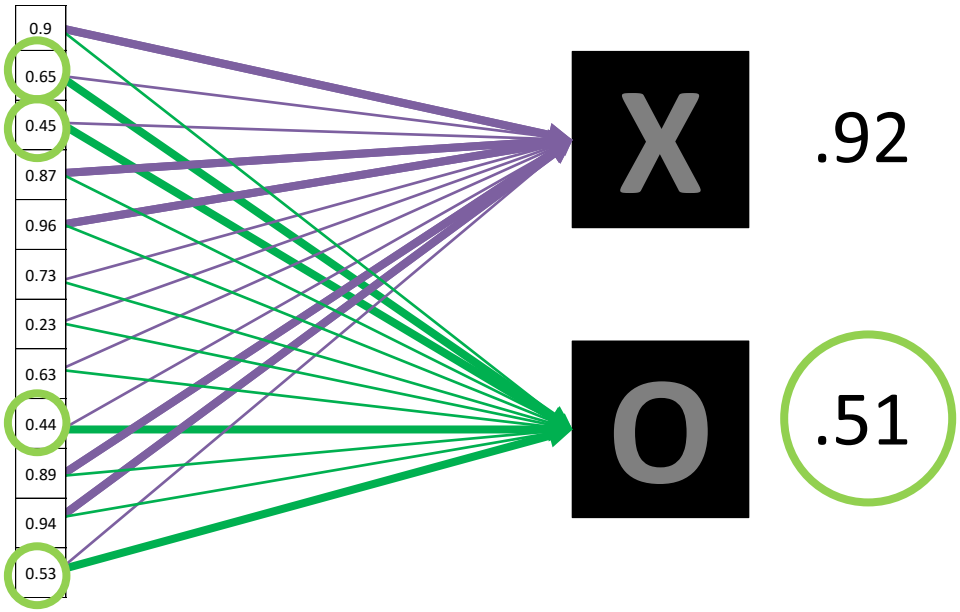
Future values vote on X or O



źródło: http://brohrer.github.io/blog.html

Fully connected layer

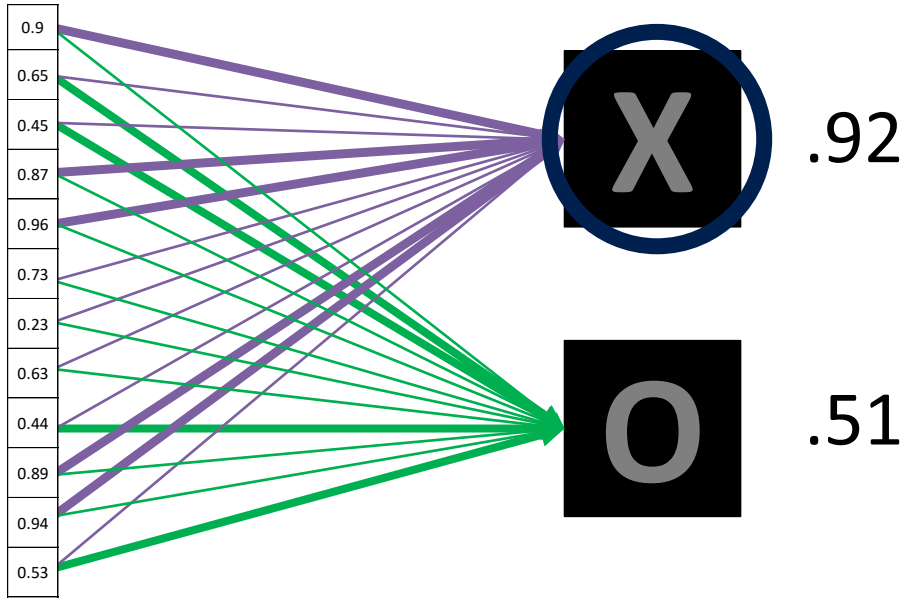
Future values vote on X or O



źródło: http://brohrer.github.io/blog.html

Fully connected layer

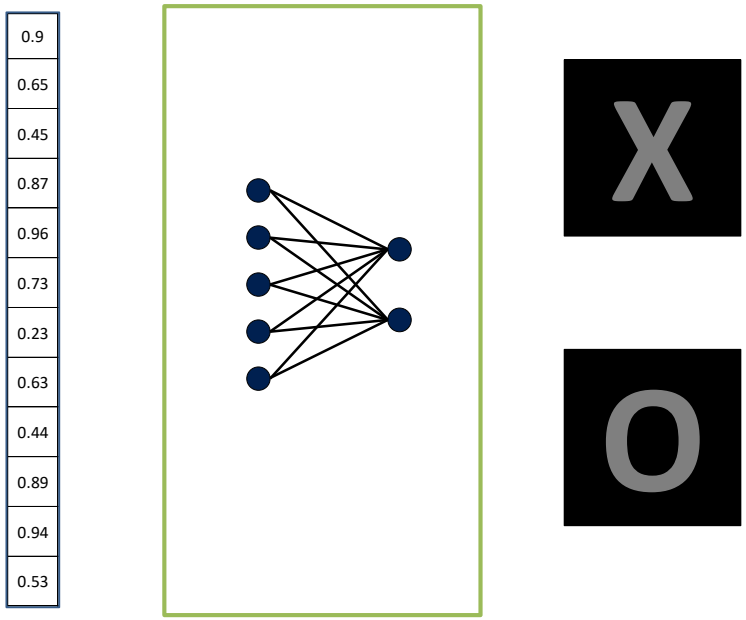
Future values vote on X or O



źródło: http://brohrer.github.io/blog.html

Fully connected layer

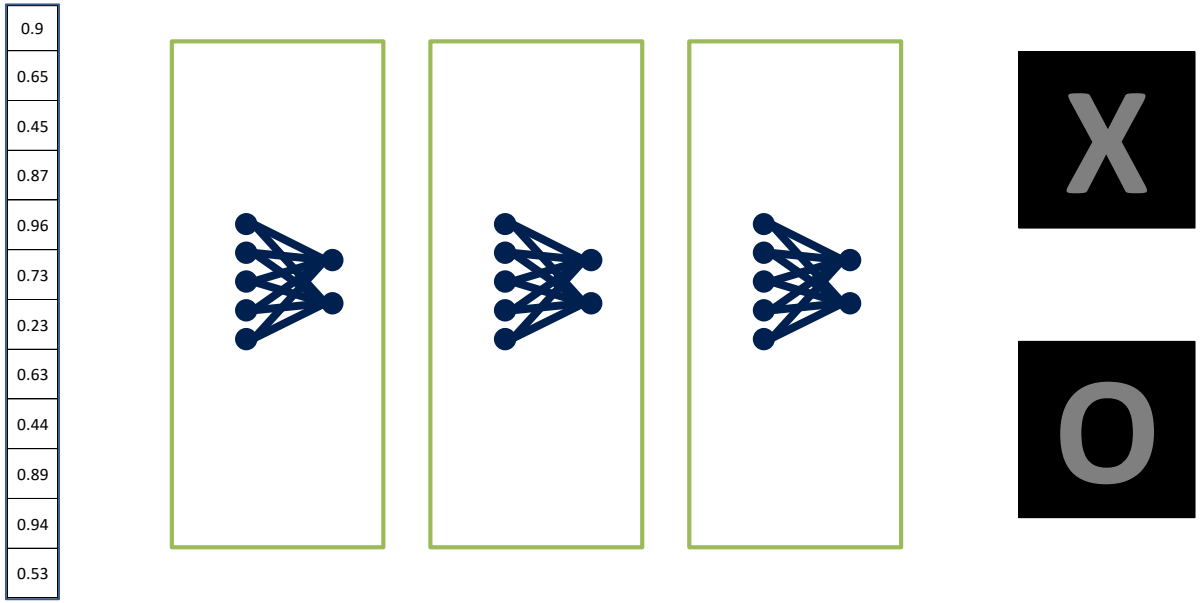
A list of feature values becomes a list of votes.



źródło: http://brohrer.github.io/blog.html

Fully connected layer

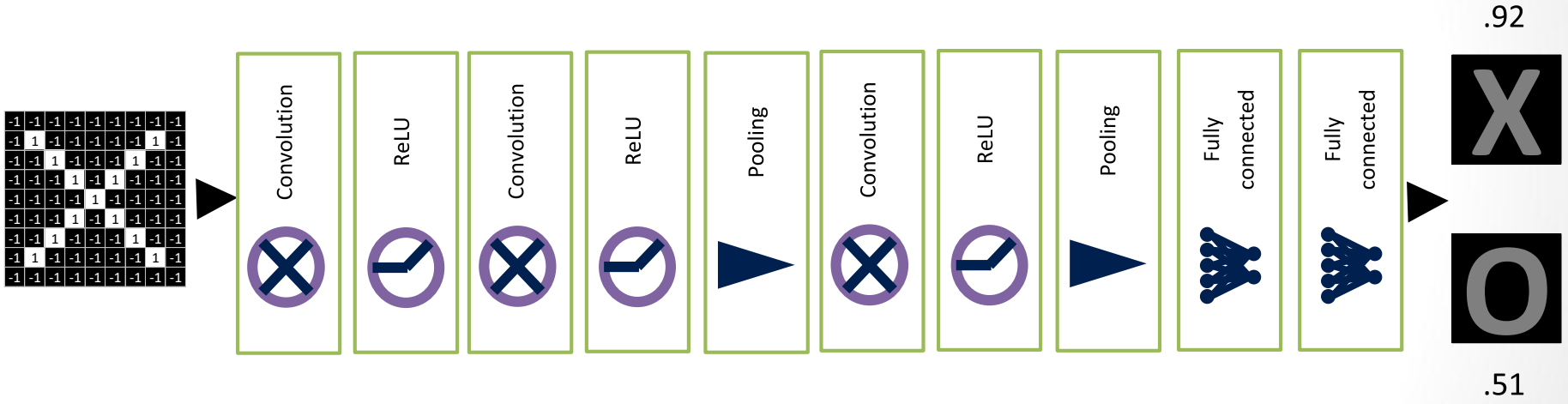
These can also be stacked.



źródło: http://brohrer.github.io/blog.html

Putting it all together

A set of pixels becomes a set of votes.



źródło: http://brohrer.github.io/blog.html

Learning

Q: Where do all the magic numbers come from?

Features in convolutional layers

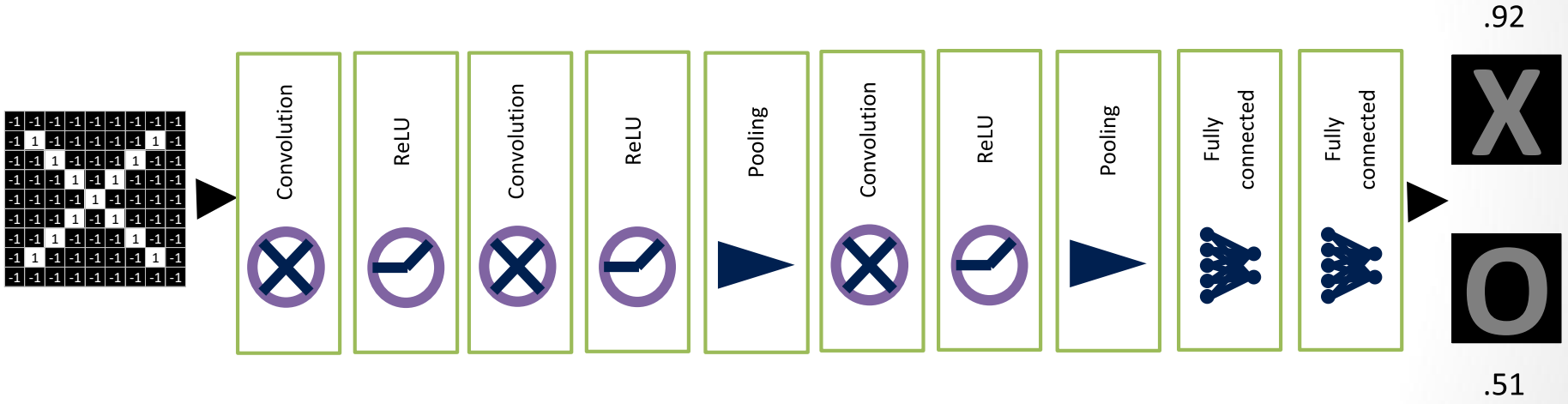
Voting weights in fully connected layers

A: Backpropagation

źródło: http://brohrer.github.io/blog.html

Backprop

Error = right answer – actual answer



źródło: http://brohrer.github.io/blog.html

Podstawowe elementy głębokiej sieci neuronowej

Operacja splotu (konwolucji)

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n) K(m,n)$$

Wejście (do filtru)

Jądro (filtr)

w	*x*
y	*z*

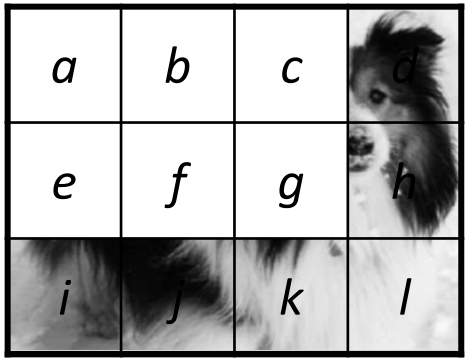
Wyjście (z filtru)

Podstawowe elementy głębokiej sieci neuronowej

Operacja splotu (konwolucji)

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n) K(m,n)$$

Wejście (do filtru)



Jądro (filtr)

w	*x*
y	*z*

Wyjście (z filtru)

Podstawowe elementy głębokiej sieci neuronowej

Operacja splotu (konwolucji)

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n) K(m,n)$$

Wejście (do filtru)

a	*b*	*c*	*d*
e	*f*	*g*	*h*
i	*j*	*k*	*l*

Jądro (filtr)

w	*x*
y	*z*

Wyjście (z filtru)

aw+bx+ ey+fz		

Podstawowe elementy głębokiej sieci neuronowej

Operacja splotu (konwolucji)

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n) K(m,n)$$

Wejście (do filtru)

a	*b*	*c*	*d*
e	*f*	*g*	*h*
i	*j*	*k*	*l*

Jądro (filtr)

w	*x*
y	*z*

Wyjście (z filtru)

aw+bx+ ey+fz	*bw+cx+ fy+gz*	

Podstawowe elementy głębokiej sieci neuronowej

Operacja splotu (konwolucji)

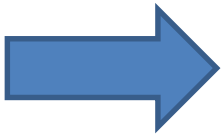
$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n) K(m,n)$$

Wejście (do filtru)

a	*b*	*c*	*d*
e	*f*	*g*	*h*
i	*j*	*k*	*l*

Jądro (filtr)

w	*x*
y	*z*



Wyjście (z filtru)

aw+bx+ ey+fz	*bw+cx+ fy+gz*	*cw+dx+ gy+hz*

Podstawowe elementy głębokiej sieci neuronowej

Operacja splotu (konwolucji)

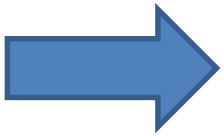
$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n) K(m,n)$$

Wejście (do filtru)

a	*b*	*c*	*d*
e	*f*	*g*	*h*
i	*j*	*k*	*l*

Jądro (filtr)

w	*x*
y	*z*



Wyjście (z filtru)

aw+bx+ ey+fz	*bw+cx+ fy+gz*	*cw+dx+ gy+hz*
ew+fx+ iy+jz		

Podstawowe elementy głębokiej sieci neuronowej

Operacja splotu (konwolucji)

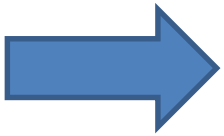
$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n) K(m,n)$$

Wejście (do filtru)

a	*b*	*c*	*d*
e	*f*	*g*	*h*
i	*j*	*k*	*l*

Jądro (filtr)

w	*x*
y	*z*



Wyjście (z filtru)

aw+bx+ ey+fz	*bw+cx+ fy+gz*	*cw+dx+ gy+hz*
ew+fx+ iy+jz	*fw+gx+ jy+kz*	

Podstawowe elementy głębokiej sieci neuronowej

Operacja splotu (konwolucji)

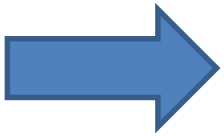
$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n) K(m,n)$$

Wejście (do filtru)

a	*b*	*c*	*d*
e	*f*	*g*	*h*
i	*j*	*k*	*l*

Jądro (filtr)

w	*x*
y	*z*



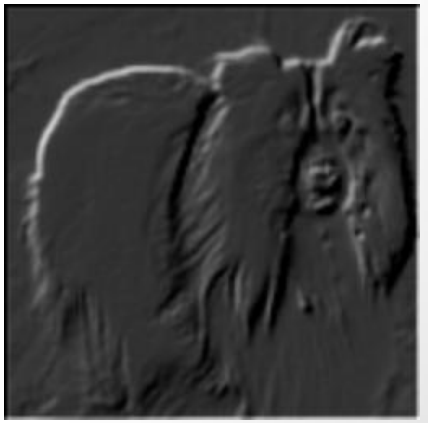
Wyjście (z filtru)

aw+bx+ ey+fz	*bw+cx+ fy+gz*	*cw+dx+ gy+hz*
ew+fx+ iy+jz	*fw+gx+ jy+kz*	*gw+hx+ ky+lz*

Podstawowe elementy głębokiej sieci neuronowej

Operacja splotu (konwolucji)

Przykładowe wyjścia z kolejnych filtrów konwolucyjnych

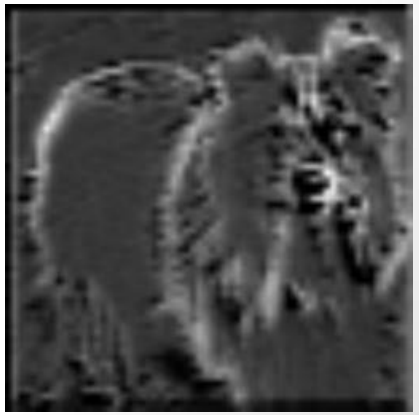


Ilość i rozmiar wyjścia z filtrów: 64; 224x224

Podstawowe elementy głębokiej sieci neuronowej

Operacja splotu (konwolucji)

Przykładowe wyjścia z kolejnych filtrów konwolucyjnych



Ilość i rozmiar wyjścia z filtrów: 128; 112x112

Podstawowe elementy głębokiej sieci neuronowej

Operacja splotu (konwolucji)

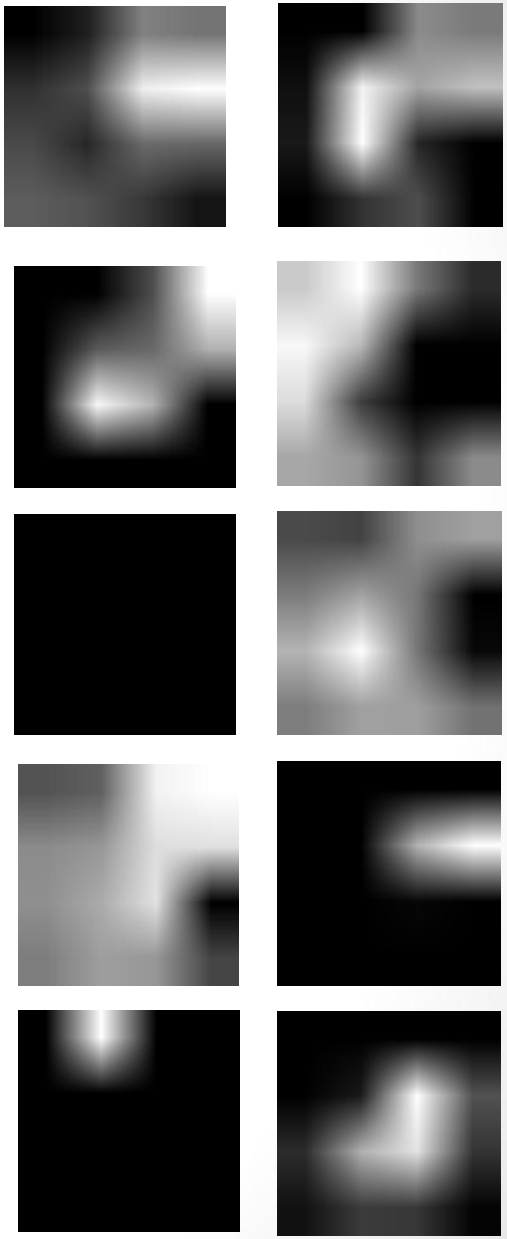
Przykładowe wyjścia z kolejnych filtrów konwolucyjnych

Ilość i rozmiar wyjścia z filtrów: 128; 56x56

Podstawowe elementy głębokiej sieci neuronowej

Operacja splotu (konwolucji)

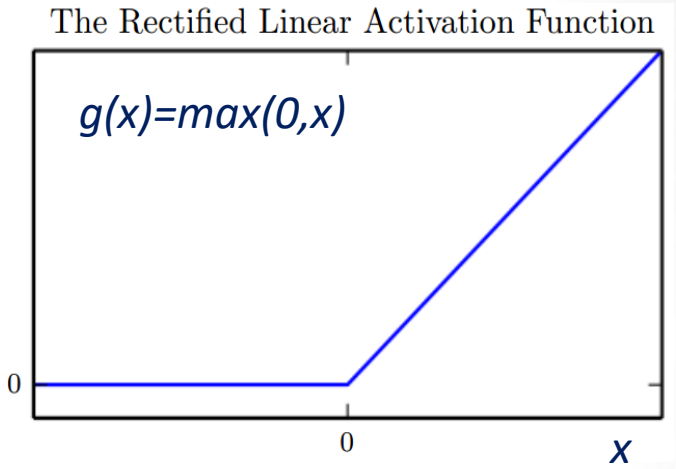
Przykładowe wyjścia z kolejnych filtrów konwolucyjnych



Ilość i rozmiar wyjścia z filtrów: 1024; 7x7

Podstawowe elementy głębokiej sieci neuronowej

Nieliniowa funkcja aktywacji (ReLU)



Najczęściej:

Rectified Linear Unit (ReLU)

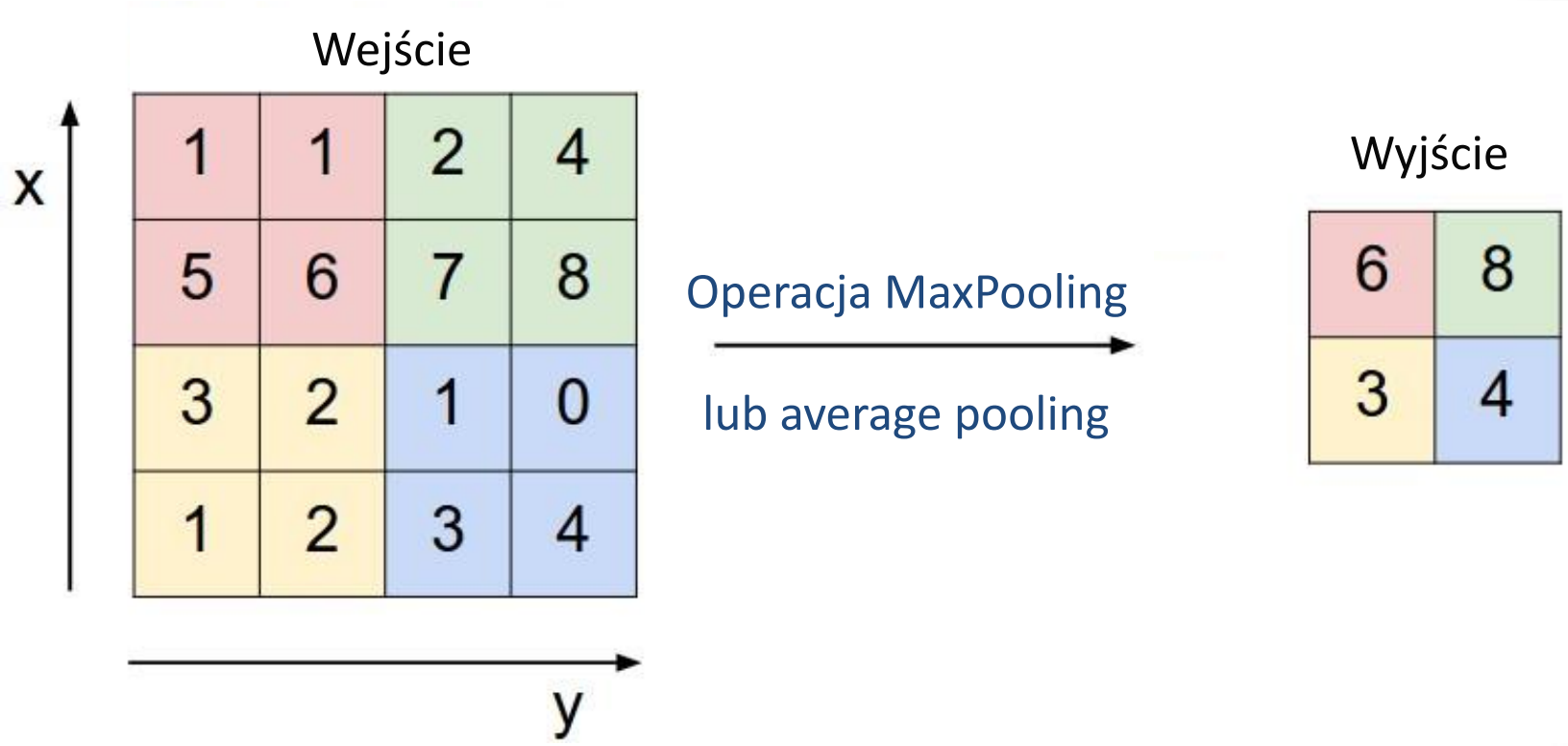
Zalety:

- 2 obszary stałego gradientu
- brak nasycenia funkcji – zapobieganie zjawisku: „zanikania gradientu”
- mniejsze wymagania obliczeniowe funkcji oraz jej gradientu w porównaniu do innych, popularnych funkcji aktywacji (sigmoidalna, tangens hiperboliczny)

Inne: Leaky ReLU (*max(ax,x)*), sigmoidalna, tangens hiperboliczny

Podstawowe elementy głębokiej sieci neuronowej

Warstwa redukująca (subsampling)



Zalety:

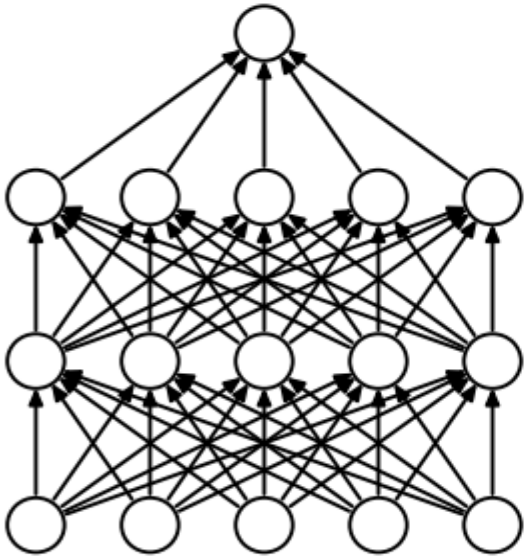
- Redukcja rozmiaru
- Zwiększenie zdolności generalizacji sieci

http://cs231n.github.io/convolutional-networks/

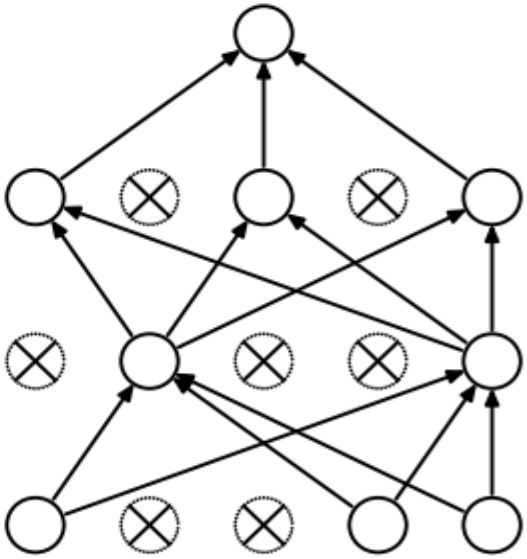
Podstawowe elementy głębokiej sieci neuronowej

„Warstwa" typu dropout

- Polega na usuwaniu losowo wybranych neuronów podczas treningu sieci



Standardowa sieć neuronowa



Sieć po zastosowaniu operacji dropout

- Celem jest poprawa zdolności do generalizacji sieci neuronowej

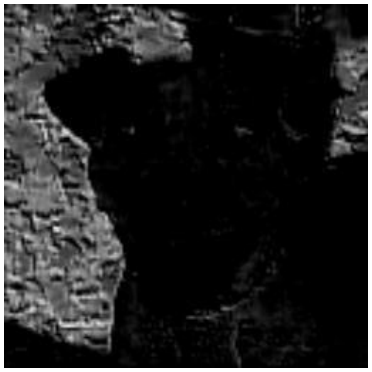
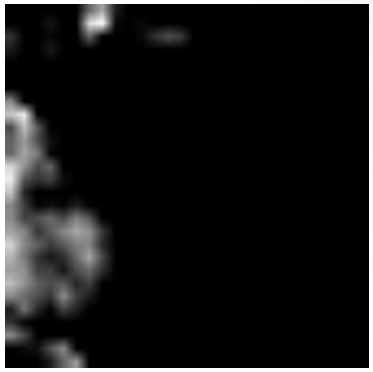
Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, *15*(1), 1929-1958.

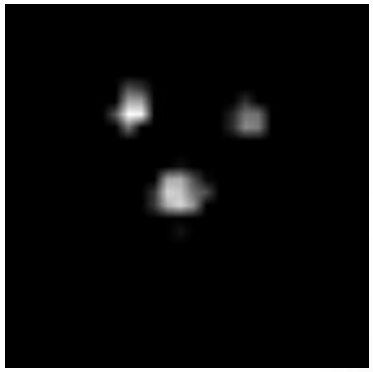
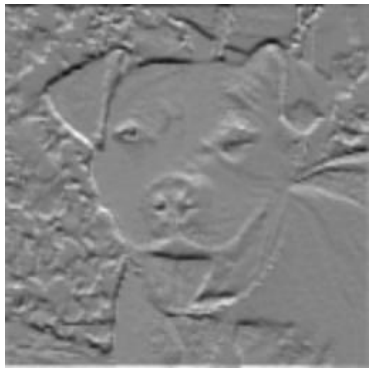
Ekstrakcja cech obrazu

Pierwsza warstwa konwolucyjna

Środkowa warstwa konwolucyjna

Ostatnia warstwa konwolucyjna

Przykładowa głęboka architektura

Wejście (224x224x3)
64 conv
64 conv
maxpooling
128 conv
128 conv
maxpooling
256 conv
256 conv
256 conv
256 conv
maxpooling
512 conv
512 conv
512 conv
512 conv

maxpooling
512 conv
512 conv
512 conv
512 conv
1024 conv
1024 conv
4048 FC
dropout
4048 FC
dropout
Wyjście (neuron sigmoidalny)

- 21 warstw
- 241 mln parametrów
- rozmiar filtrów - 3x3
- rozmiar sieci 1.8 GB

Convolutional Neural Networks (ConvNets, CNNs) - Deep Neural Networks

Jak to policzyć ?

Technologia NVIDIA CUDA - równoległa architektura obliczeniowa wykorzystująca układy GPU (graphics processing unit - jednostka przetwarzania graficznego)

NVIDIA TITAN X
Akcelerator procesu uczenia w głębokim uczeniu do komputerów stacjonarnych.

3584 rdzenie NVIDIA CUDA® taktowane częstotliwością 1,5 GHz, karta TITAN X oferuje wydajność na poziomie 11 TFLOPS.
Dodatkowo wyposażono ją w 12 GB pamięci GDDR5X - jest to jedna z najszybszych technologii pamięci na świecie.

http://www.nvidia.pl/object/deep-learning-pl.html

Convolutional Neural Networks (ConvNets, CNNs) - Deep Neural Networks

Jak to policzyć ?

NVIDIA DGX Station

- 4 karty graficzne Tesla V100 (5120 rdzeni CUDA, 15 TFLOPS, procesory Tensor Core w pełni zoptymalizowane do obliczeń związanych z głębokim uczeniem, 640 rdzeni)
- 500 TFLOPS, (500 000 000 000 000 operacji na liczbach zmiennoprzecinkowych FP16/sek)

Cena ok. 250 tys. PLN

Convolutional Neural Networks (ConvNets, CNNs) - Deep Neural Networks

Jak to policzyć ?

NVIDIA DGX2 Station

- 16 kart graficznych Tesla V100 32GB
- 2-petaFLOPS GPU (2x10e15 operacji zmiennoprzecinkowych/sek)
- 10 – 24 x szybciej niż DX1

Cena ok. 400 tys. $

Convolutional Neural Networks (ConvNets, CNNs) - Deep Neural Networks

Jak to policzyć ?

NVIDIA JETSON TX1
Jetson TX1 to superkomputer w formie modułu o rozmiarach karty kredytowej.

Jetson TX1 wyposażony w 256 rdzeni CUDA, zapewniając ponad 1 TeraFLOP mocy obliczeniowej. 64-bitowym jednostki CPU, kodowanie i dekodowanie wideo 4K oraz interfejs kamer o wydajności na poziomie 1400 MPix/s.

NVIDIA JETSON TX2
Więcej pamięci, lepsze procesory

http://www.nvidia.pl/object/jetson-embedded-systems-pl.html

Convolutional Neural Networks (ConvNets, CNNs) - Deep Neural Networks

Jak to napisać ?

Deep learning software by name [edit]

This list is incomplete; you can help by expanding it.

Software	Creator	Software license[5]	Open source	Platform	Written in	Interface	OpenMP support	OpenCL support	CUDA support	Automatic differentiation[1]	Has pretrained models	Recurrent nets	Convolutional nets	RBM/DBNs	Parallel execution (multi node)
Apache Singa	Apache Incubator	Apache 2.0	Yes	Linux, Mac OS X, Windows	C++	Python, C++, Java	No	Yes	Yes	?	No	Yes	Yes	Yes	Yes
Caffe[2]	Berkeley Vision and Learning Center, community contributors	BSD 2-Clause License	Yes	Ubuntu, OS X, AWS,[3] unofficial Android port,[4] Windows support by Microsoft Research,[5] unofficial Windows port[6]	C++	C++, command line, Python, MATLAB[7]	No	Branch,[8] pull request,[9] third party implementation[10]	Yes	?	Yes[11]	Yes	Yes	No[12]	Yes[13]
Chainer[14]	Preferred Networks, inc., Preferred Infrastructure, inc.[14]	MIT license	Yes	Linux, Mac OS X Windows	Python	Python	No	On roadmap[15]	Yes	Yes	Through Caffe's model zoo[16]	Yes	Yes	Yes	Yes[17]
cnn[18]	Carnegie Mellon School of Computer Science	Apache 2.0	Yes	Linux	C++	C++, Python	No	No	Yes	Yes	No	Yes	Yes	No	No
Deeplearning4j	Skymind engineering team; Deeplearning4j community; originally Adam Gibson	Apache 2.0	Yes	Linux, Ubuntu, Windows, OSX, Android (Cross-platform)	C, C++	Java, Scala, Clojure	Yes	On roadmap[19]	Yes[20]	Computational Graph	Yes[21]	Yes	Yes	Yes	Yes[22]
Microsoft Cognitive Toolkit	Microsoft Research	MIT license[23]	Yes	Windows, Linux[24] (OSX via Docker on roadmap)	C++	Command line,[25] network description language BrainScript[26] (C++, Python and .NET on roadmap[27])	Yes[28]	No	Yes	Yes	No[29]	Yes[30]	Yes[30]	No[31]	Yes[32]
MXNet[33]	Distributed (Deep) Machine Learning Community	Apache 2.0	Yes	Ubuntu, OS X, Windows,[34][35] AWS, Android,[36] iOS, JavaScript[37]	C++, Python, Julia, Matlab, Go, R, Scala	C++, Python, Julia, Matlab, JavaScript, Go, R, Scala	Yes	On roadmap[38]	Yes	Yes[39]	Yes[40]	Yes	Yes	Yes	Yes[41]
Wolfram Mathematica[42]	Wolfram Research	Proprietary	No	Windows, OS X, Linux, Cloud_computing	C++	Command line, Java, C++	No	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Neural Designer	Artelnics	Proprietary	No	Windows, OS X, Linux	C++	Graphical user interface	Yes	No	No	?	?	No	No	No	?
OpenNN	Artelnics	GNU LGPL	Yes	Cross platform	C++	C++	Yes	No	No	?	?	No	No	No	?
Keras	François Chollet	MIT license	Yes	Linux, Mac OS X, Windows	Python	Python	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes[43]
PaddlePaddle[44]	Baidu	Apache 2.0	Yes	Linux, Mac OS X	C++	Python, C++	No	No	Yes	?	Yes	Yes	Yes	?	Yes
TensorFlow	Google Brain team	Apache 2.0	Yes	Linux, Mac OS X (Windows support on roadmap[45][46])	C++, Python	Python, (C/C++ public API only for executing graphs[47])	No	On roadmap[48][49]	Yes	Yes[49]	Yes[50]	Yes	Yes	Yes	Yes
TensorLayer[51]	Imperial College	Apache 2.0	Yes	Linux, Mac OS X (Windows support on roadmap[52][46][53])	Python	Python, (C/C++ public API only for executing graphs[54])	No	On roadmap[55][55]	Yes	Yes[51]	Yes[51]	Yes	Yes	Yes	?
Theano	Université de Montréal	BSD license	Yes	Cross-platform	Python	Python	Yes	Under development[56]	Yes	Yes[57][58]	Through Lasagne's model zoo[59]	Yes	Yes	Yes	Yes[60]
Torch	Ronan Collobert, Koray Kavukcuoglu, Clement Farabet	BSD License	Yes	Linux, Android,[61] Mac OS X, iOS, Windows[62]	C, Lua	Lua, LuaJIT,[63] C, utility library for C++/OpenCL[64]	Yes	Third party implementations[65]	Yes[66][67]	Through Twitter's Autograd[68]	Yes[69]	Yes	Yes	Yes	Yes[70]

https://en.wikipedia.org/wiki/Comparison_of_deep_learning_software

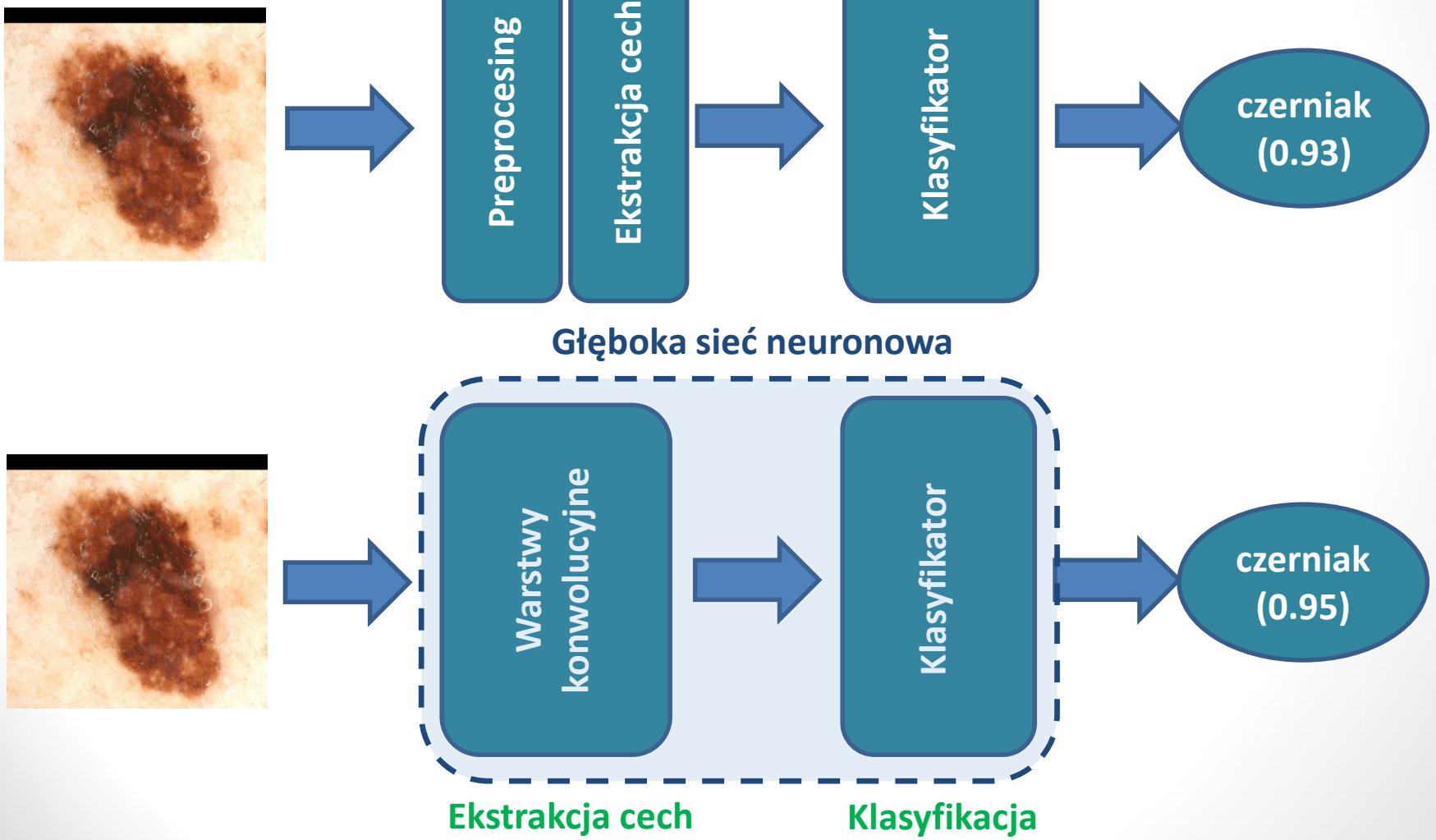
Implementacja głębokich sieci neuronowych:

Oprogramowanie (najpopulariejsze):

- TensorFlow (GoogleBrain),
- Keras (MIT, GoogleBrain),
- Caffe2 (Facebook),
- Theano (Université de Montréal),
- Caffe (Berkeley Vision and Learning Center),
- Microsoft Cognitive Toolkit (Microsoft Research),
- Matlab Machine Learning Toolbox (Mathworks),

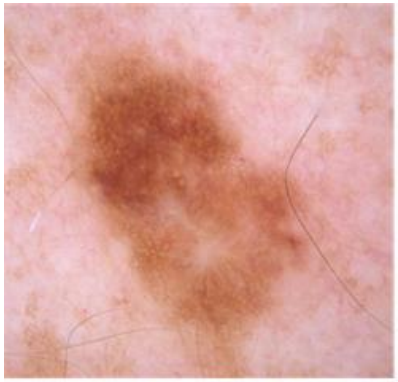
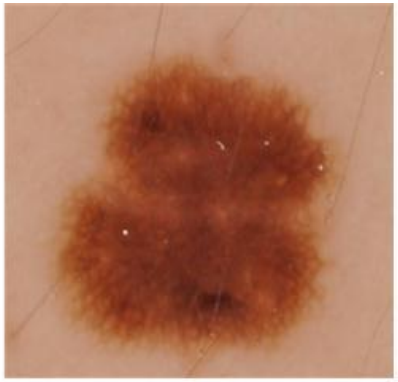
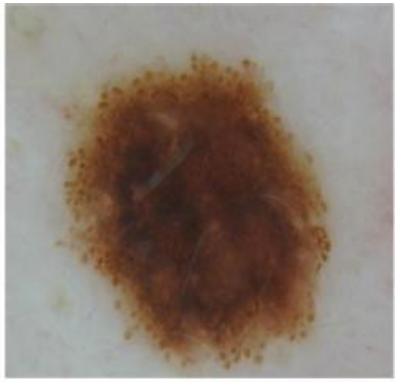
Porównanie działania głębokich i klasycznych sieci neuronowych - analiza znamion skórnych

Różnice w podejściu przy pomocy głębokich i klasycznych sieci neuronowych



Przykład: Analiza znamion skórnych

System wspomagający pracę lekarza w klasyfikacji znamion skórnych na podstawie zdjęć różnej jakości, niekoniecznie dermatoskopowych.

Znamiona zdrowe



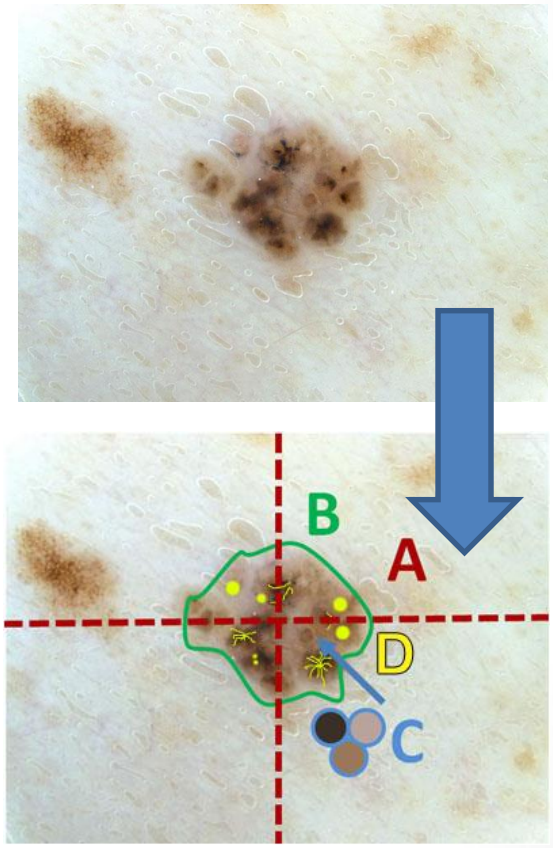
Znamiona chore (czerniak)

Przykład: Analiza znamion skórnych

Najbardziej znana metoda diagnostyczna: ABCD

- **A (Assymetry) – Czy znamię jest symetryczne?**
- **B (Border) – Czy krawędź znamienia jest równa i gładka?**
- **C (Color) – Czy znamię jest jednolitego koloru?**
- **D (Diffrental Structures) – Czy można na nim zauważyć kropki, plamy lub inne nietypowe struktury?**
- **E (Evolution)- Czy znamię zmieniło się w ciągu ostatnich 3 miesięcy?**

NIE!
NIE!
NIE!
TAK!
TAK!



WYNIK: NOWOTWÓR

Przykład: Analiza znamion skórnych

Dwa podejścia:

- Klasyczna sieć neuronowa
- Głęboka sieć neuronowa

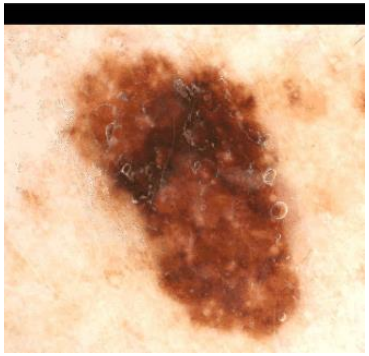
Baza danych: ISIC Archive

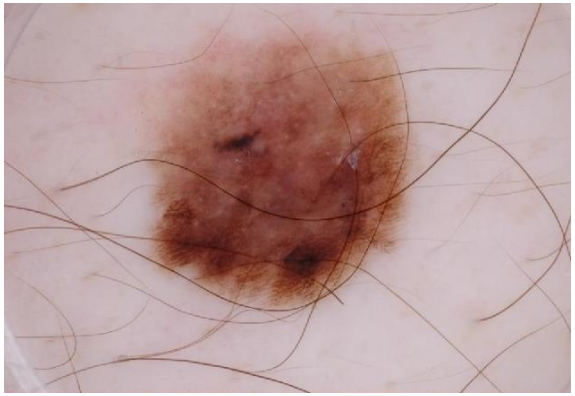
Baza danych stworzona w celu ułatwienia badań nad systemami rozpoznawania obrazów w medycynie.

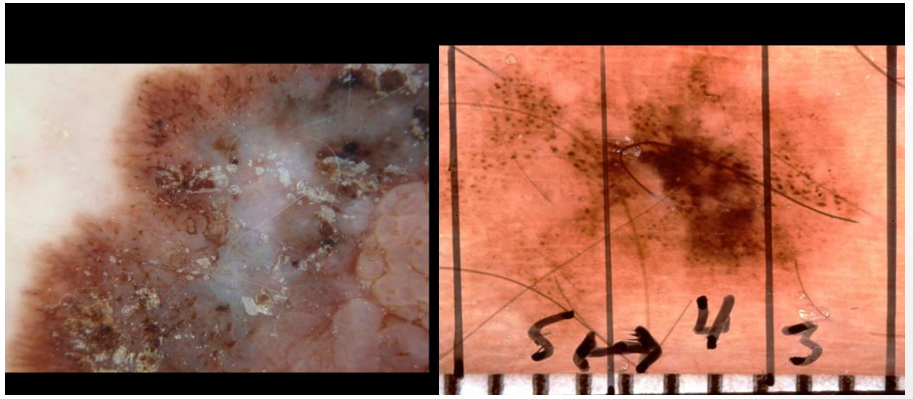
	Zdrowe	Chore
Zbiór uczący	9300	670
Zbiór testowy	100	100

Analiza znamion skórnych: Klasyczna sieć

Przykładowe obrazy z bazy danych

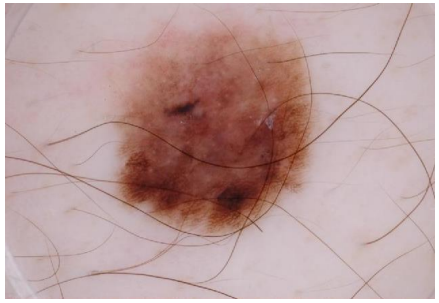






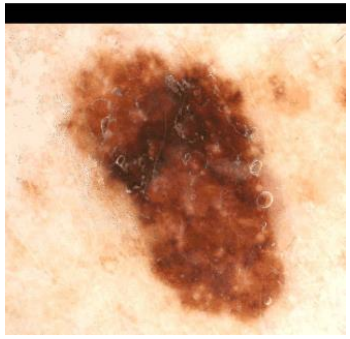
Analiza znamion skórnych: Klasyczna sieć

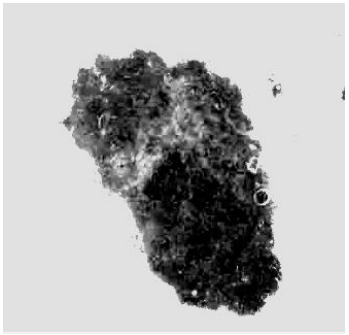
„Ręczna”, automatyczna ekstrakcja cech A, B, C, D



Filtr konwolucyjny

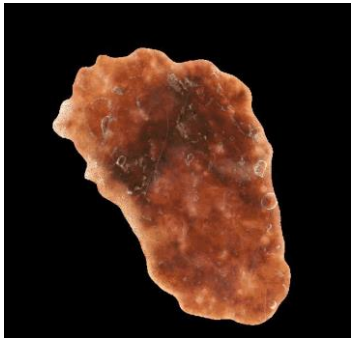
Estymowany obraz



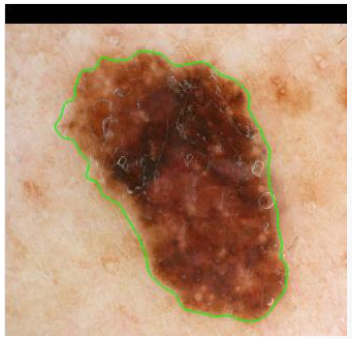


Obraz binarny

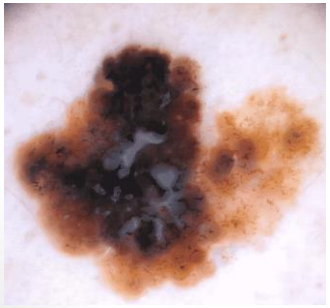
Maska

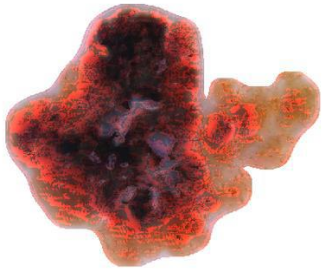


Wycięcie

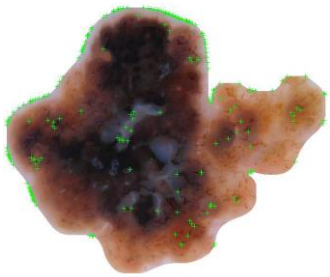


Krawędzie

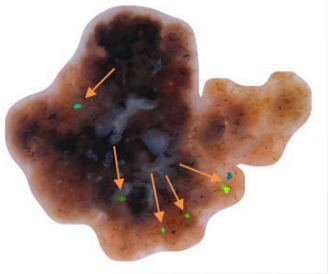




Filtr konwolucyjny



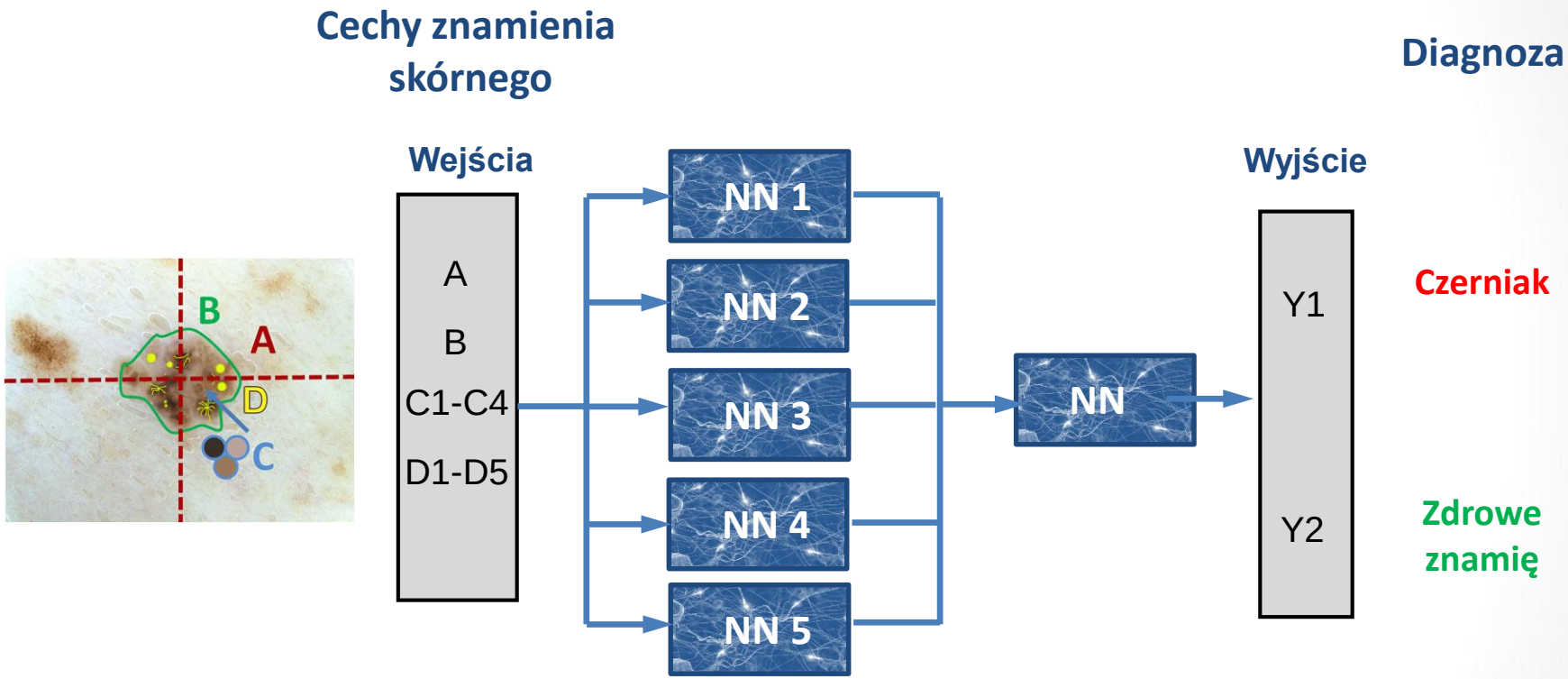
MSER

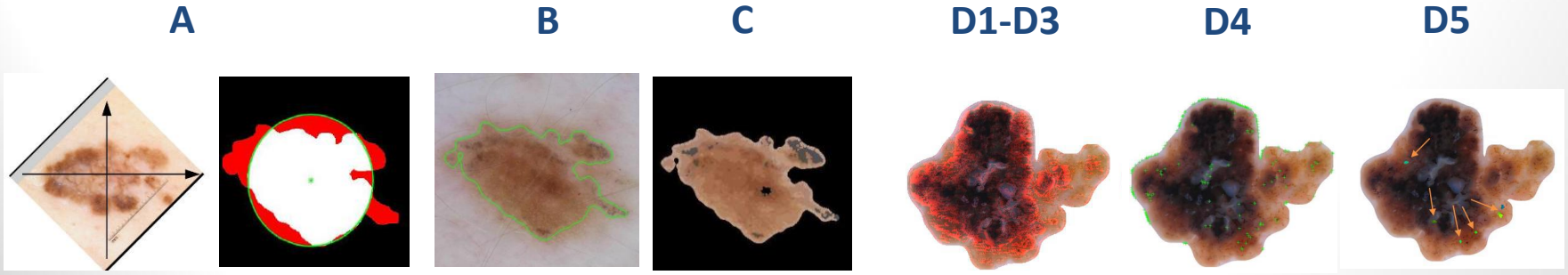


Detektor Harrisa

Analiza znamion skórnych: Klasyczna sieć

Struktura systemu neuronowego





Analiza znamion skórnych: Głęboka sieć

- Brak „ręcznej” ekstrakcji cech i przetwarzania wstępnego obrazów
- Jedyne operacje to zmiana rozmiaru wejść (224x224) oraz normalizacja

Architektura sieci

Wejście (224x224x3)
64 conv
64 conv
maxpooling
128 conv
128 conv
maxpooling
256 conv
256 conv
256 conv
256 conv
maxpooling
512 conv
512 conv
512 conv
512 conv

maxpooling
512 conv
512 conv
512 conv
512 conv
1024 conv
1024 conv
4048 FC
dropout
4048 FC
dropout
Wyjście (neuron sigmoidalny)

- 21 warstw
- 241 mln parametrów
- rozmiar filtrów - 3x3
- rozmiar sieci 1.8 GB

Analiza znamion skórnych: Głęboka sieć

Transfer learning – transfer początkowych wartości wag

- z sieci VGG19 (19 warstw uczonych, 144 mln parametrów)
- VGG19 wytrenowana na milionie bardzo różnych obrazów (1000 klas x 1000 obrazów na klasę)

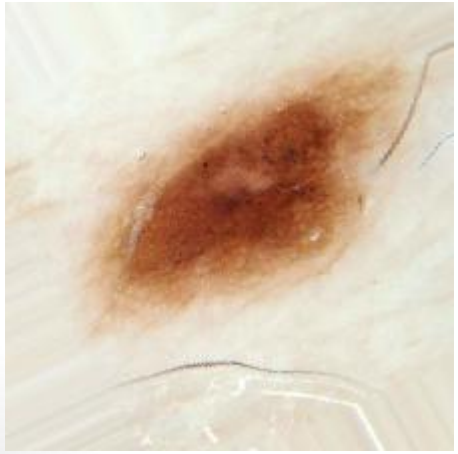
Analiza znamion skórnych: Głęboka sieć

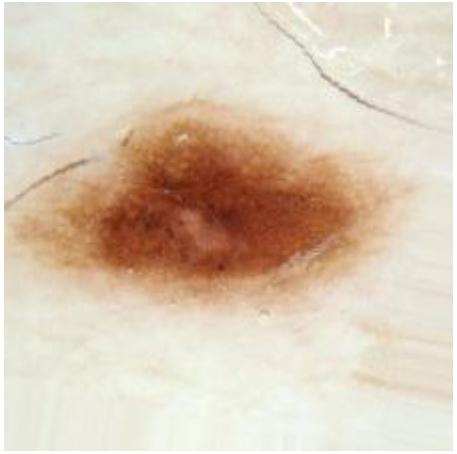
Modyfikacja bazy danych

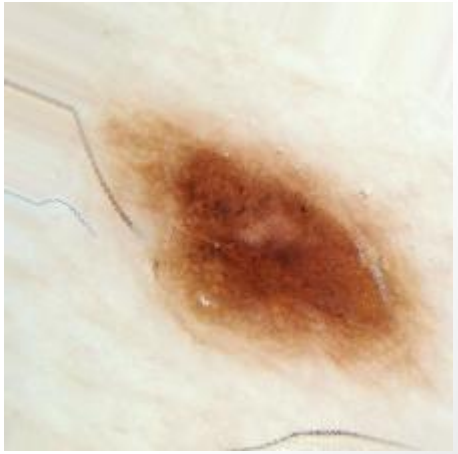
Cel:

- zwiększenie ilości i różnorodności wzorców
- Wyrównanie bazy danych pod względem ilości wzorców pozytywnych i negatywnych

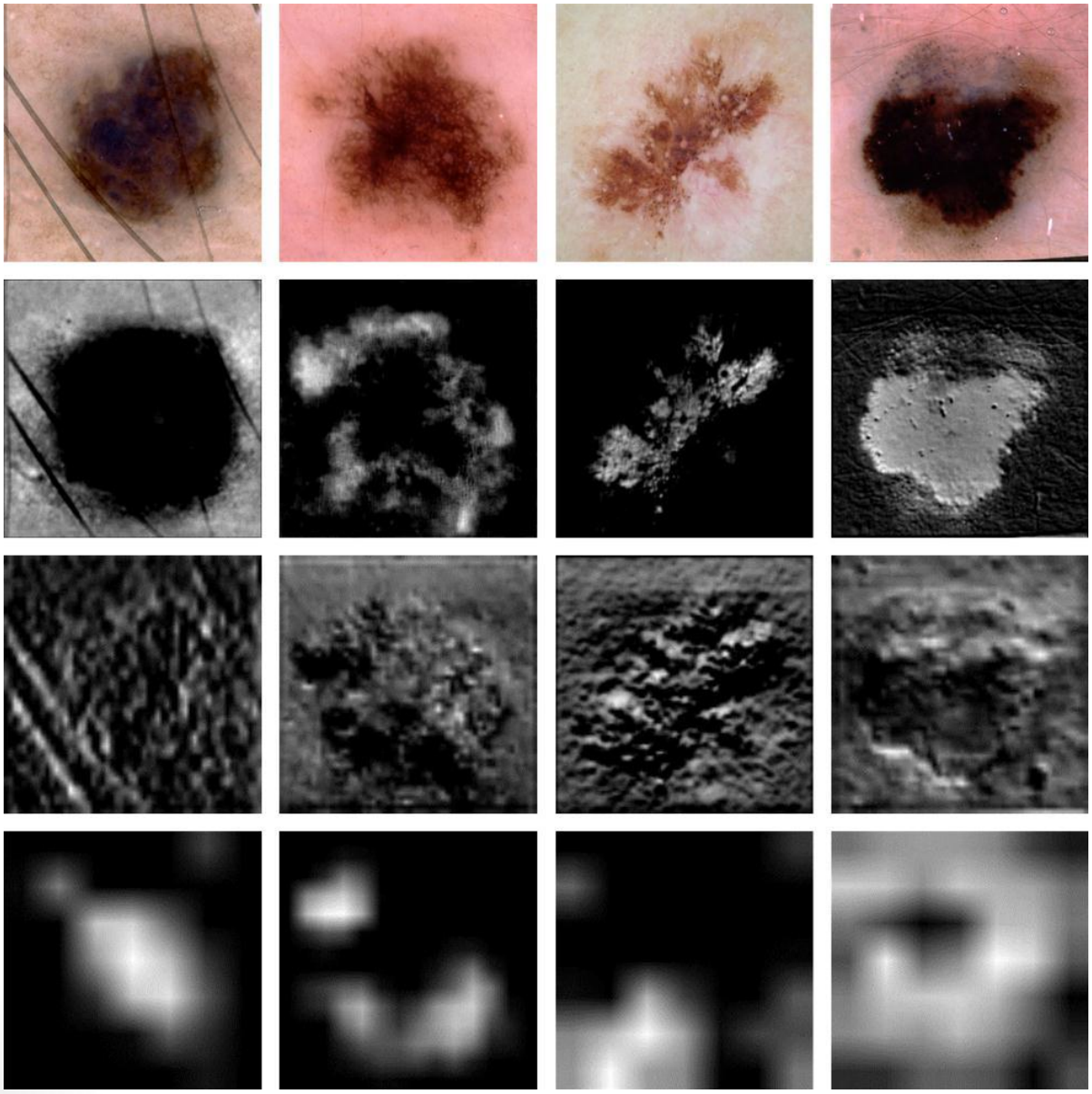
Obroty, rozciąganie, powiększanie i pomniejszanie, odbicia w poziomie i pionie obrazów







Analiza znamion skórnych: Głęboka sieć



Obraz oryginalny

Pierwsza warstwa konwolucyjna

Środkowa warstwa konwolucyjna

Ostatnia warstwa konwolucyjna

Analiza znamion skórnych: Porównanie metod

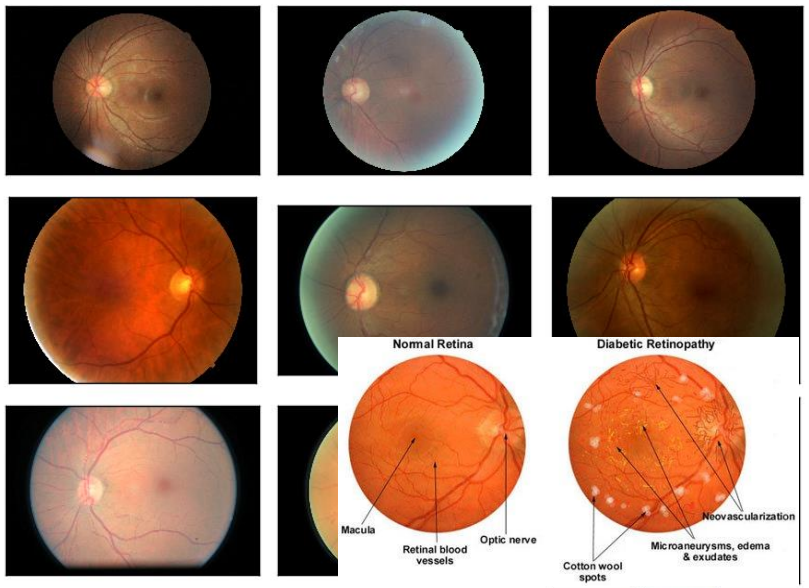
Głębokie uczenie	Metody klasyczne
Ekstrakcja cech, wnioskowanie, klasyfikacja typu black box	Klasyfikacja typu black box
Brak konieczności wstępnego przetwarzania obrazu	Konieczność wstępnego przetwarzania obrazu i zastosowania algorytmów przetwarzania obrazu
Brak konieczności ekstrakcji cech – proces wyboru cech odbywa się w trakcie uczenia	Wymagana wstępna ekstrakcja cech
Wymagana podstawowa znajomość zagadnień medycznych	Wymagana wysoka znajomość zagadnień medycznych
Wyższe dokładności	Niższe dokładności
Wymagana duża ilość obrazów treningowych	Metoda daje dobre efekty na niewielkich ilościach danych treningowych
Kosztowne obliczeniowo – wymaga silnych jednostek graficznych	Obliczenia mogą być wykonywane na CPU
Niższe zaufanie lekarzy	Wyższe zaufanie lekarzy (automatyzacja zadań prowadzonych przez lekarza)
Trudne w analizie wyników	Łatwiejsze w analizie wyników

Zastosowania głębokich sieci neuronowych - przegląd

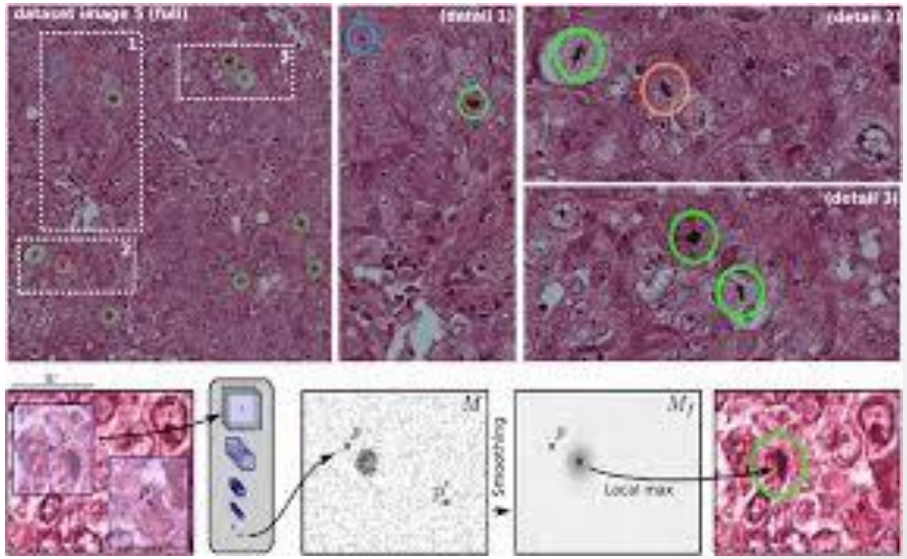
Zastosowania: Systemy wspomagania decyzji

- Wszędzie tam gdzie niezbędna jest analiza obrazów lub analiza dużych zbiorów danych
- Analiza obrazów oraz danych medycznych (np. IBM Watson Health)
- Analizy genetyczne
- Dobór składu i dawkowania leków
- Stawianie diagnoz na podstawie opisu symptomów

Retinopatia cukrzycowa na podstawie analizy gałki ocznej



Wykrywanie raka piersi



Zastosowania: Systemy wspomagania decyzji

- **Rozpoznawanie i analiza sceny:**
  - robotyka, pojazdy bezzałogowe, wspomaganie kierowców, osoby niedowidzące…

Fig. 11. Real-time scene parsing in natural conditions. Training on SiftFlow dataset. We display one label per component in the final prediction.

źródło: Clement Farabet, Camille Couprie, Laurent Najman, Yann LeCun. Learning Hierarchical Features for Scene Labeling

Zastosowania: Systemy wspomagania decyzji

- **Rozpoznawanie i analiza sceny:**
  - robotyka, pojazdy bezzałogowe, wspomaganie kierowców, osoby niedowidzące…

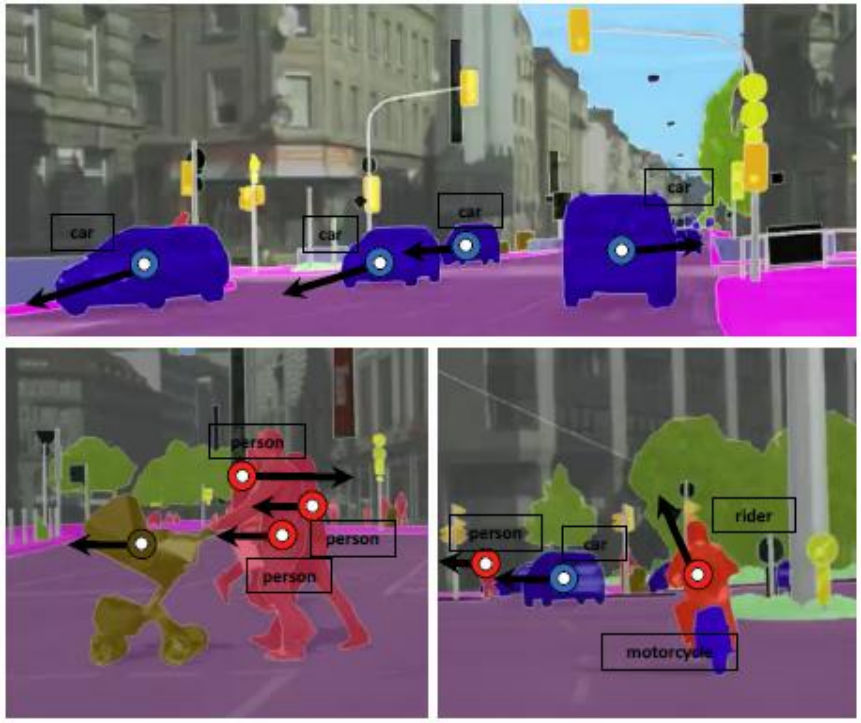


Figure 1: Our models learn semantic-level scene dynamics to predict semantic segmentations of unobserved future frames given several past frames.

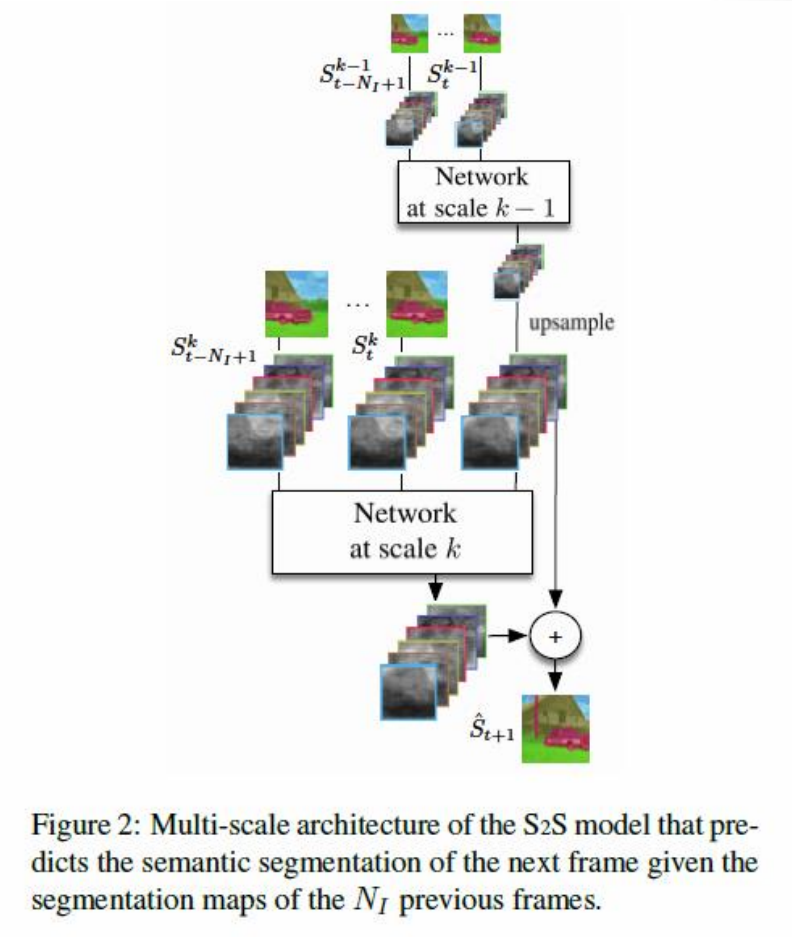


Figure 2: Multi-scale architecture of the S2S model that predicts the semantic segmentation of the next frame given the segmentation maps of the N_I previous frames.

źródło: arXiv:1703.07684v3 [cs.CV] 8 Aug 2017

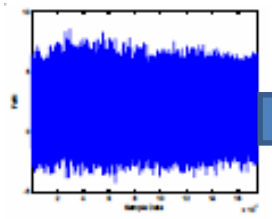
Zastosowania: Systemy wspomagania decyzji

- **Podejmowanie decyzji:**
  - Wypracowywanie strategii biznesowych,
  - Prognozowanie wskaźników giełdowych,
  - Analiza i prognozowanie zachowań konsumenckich …

Głównie systemy hybrydowe głębokich sieci neuronowych z uczeniem ze wzmocnieniem i/lub podejściem ewolucyjnym

Zastosowania: Diagnostyka

Metody klasyczne



Preprocesing (analizy Fouriera, falkowe, transformacje itp.)

Ekstrakcja cech (PCA, ICA, LDA ...)

Klasyfikator

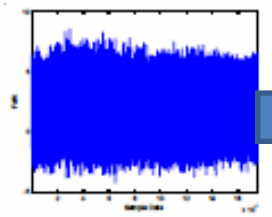
Diagnoza

Ekstrakcja cech

Klasyfikacja

Głębokie uczenie

Akwizycja danych pomiarowych



Głęboka sieć neuronowa (modele hierarchiczne cech)

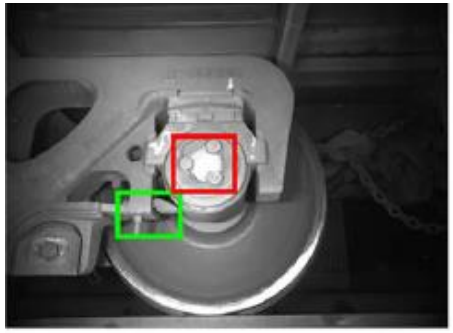
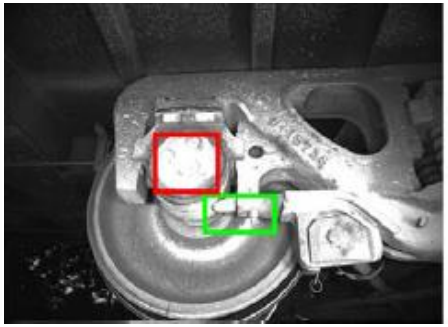
Klasyfikator

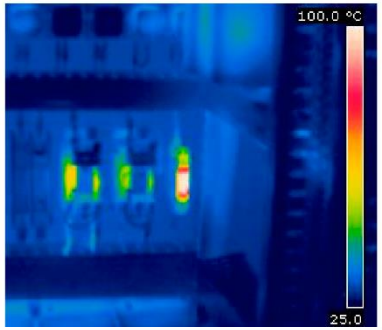
Diagnoza

Zastosowania: Diagnostyka

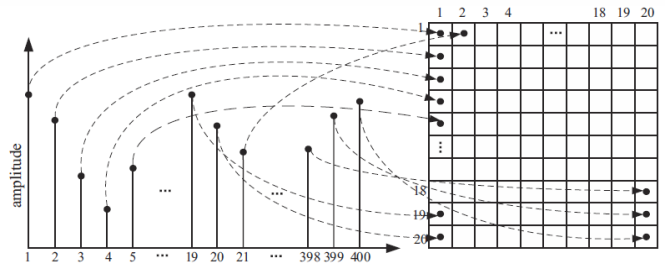
Najczęściej stosowane podejścia:

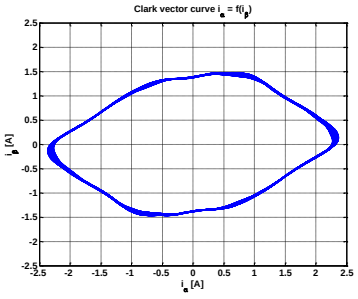
- Analiza obrazów (zdjęcia, klatki z kamery, termowizja …)

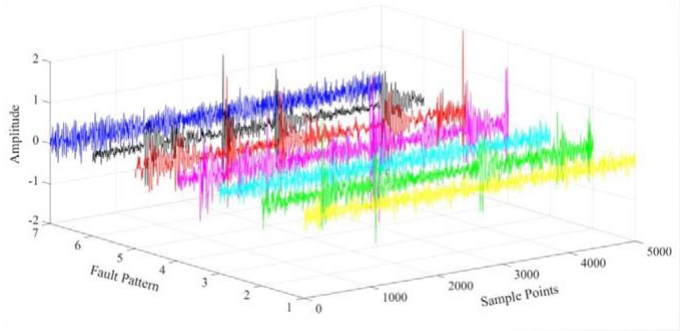



- Analiza sygnałów transformowanych do postaci „obrazów”







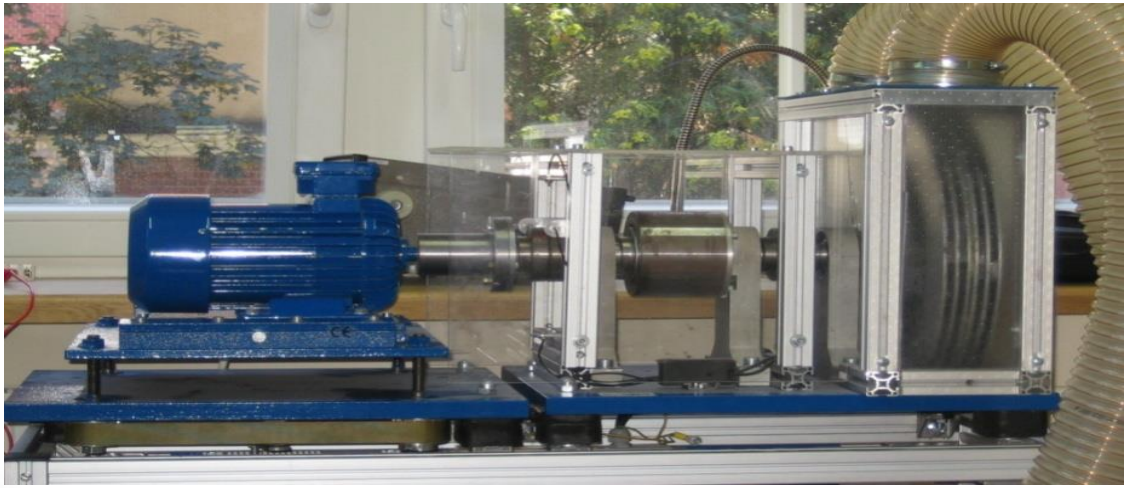
- „Bezpośrednia” analiza sygnałów pomiarowych
  – głębokie sieci rekurencyjne (np. LSTM - Long short-term memory)

Zastosowania: Diagnostyka

Przykład: diagnostyka łożysk

Przykładowe uszkodzenia łożysk silnika indukcyjnego

Przygotowano uszkodzenia o różnych długościach, szerokościach i głębokościach

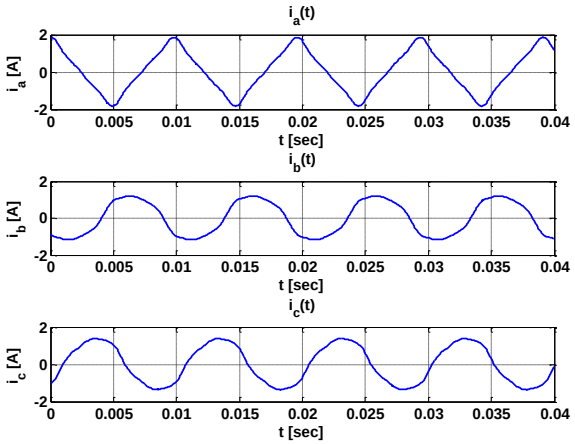


Silnik: STg80X-4C:
Pn = 1,1 kW,
U_n = 380 V,
n_n = 1400 RPM,
In = 2,9 A

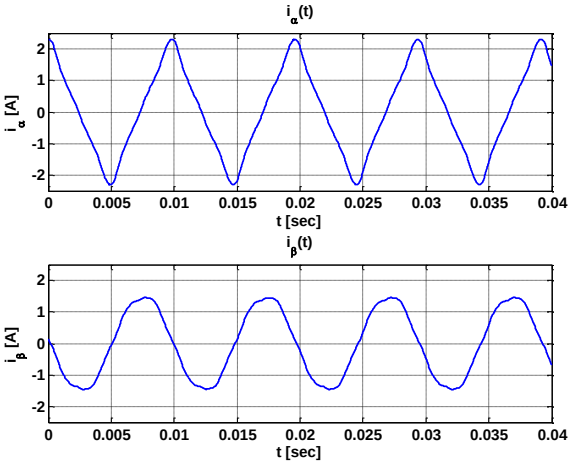
źródło: Use of neural networks in diagnostics of rolling-element bearing of the induction motor. L.Swędrowski, K.Duzinkiewicz, **M.Grochowski**, T.Rutkowski. V Międzynarodowy Kongres Diagnostyki Technicznej 2012 - Kraków

Zastosowania: Diagnostyka

Przykład: diagnostyka łożysk

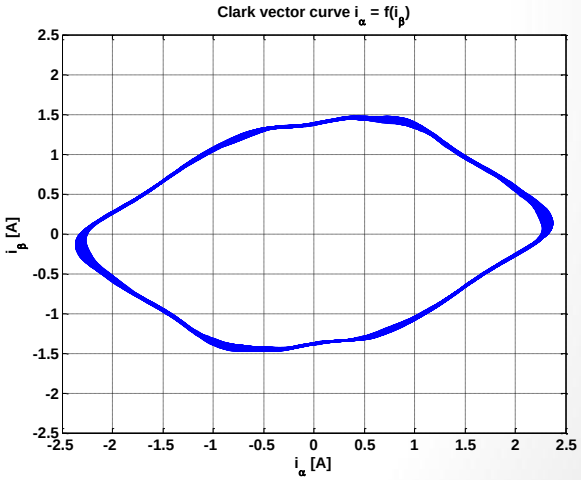


Prądy zasilania
silnika trójfazowego



Transformacja Clarka

$$\begin{cases} i_\alpha = \frac{\sqrt{6}}{2} \cdot I \cdot sin(2\pi f t) \\ i_\beta = \frac{\sqrt{6}}{2} \cdot I \cdot sin\left(2\pi f t - \frac{\pi}{2}\right) \end{cases}$$

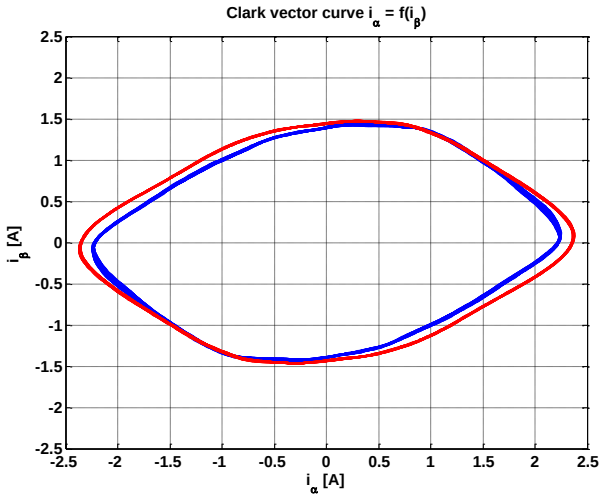


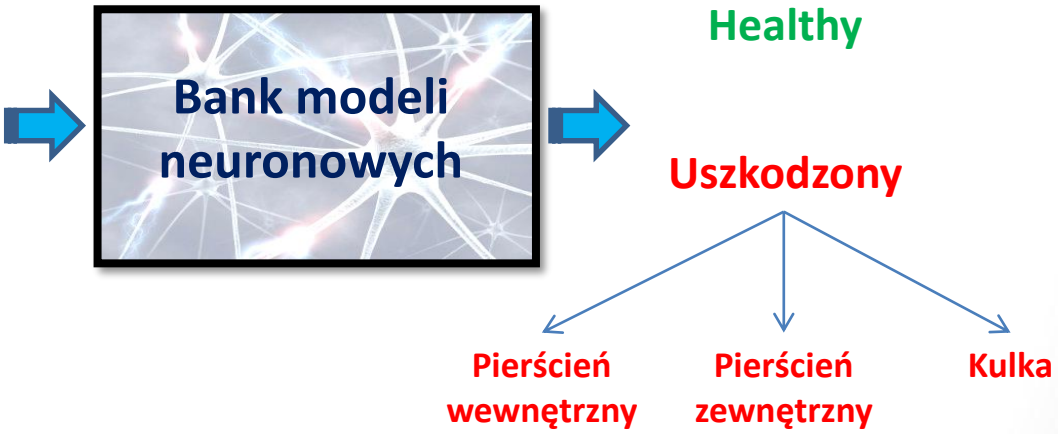
Wektory Clarka

źródło: Use of neural networks in diagnostics of rolling-element bearing of the induction motor. L.Swędrowski, K.Duzinkiewicz,M.Grochowski,T.Rutkowski. V Międzynarodowy Kongres Diagnostyki Technicznej 2012 - Kraków

Zastosowania: Diagnostyka

Przykład: diagnostyka łożysk



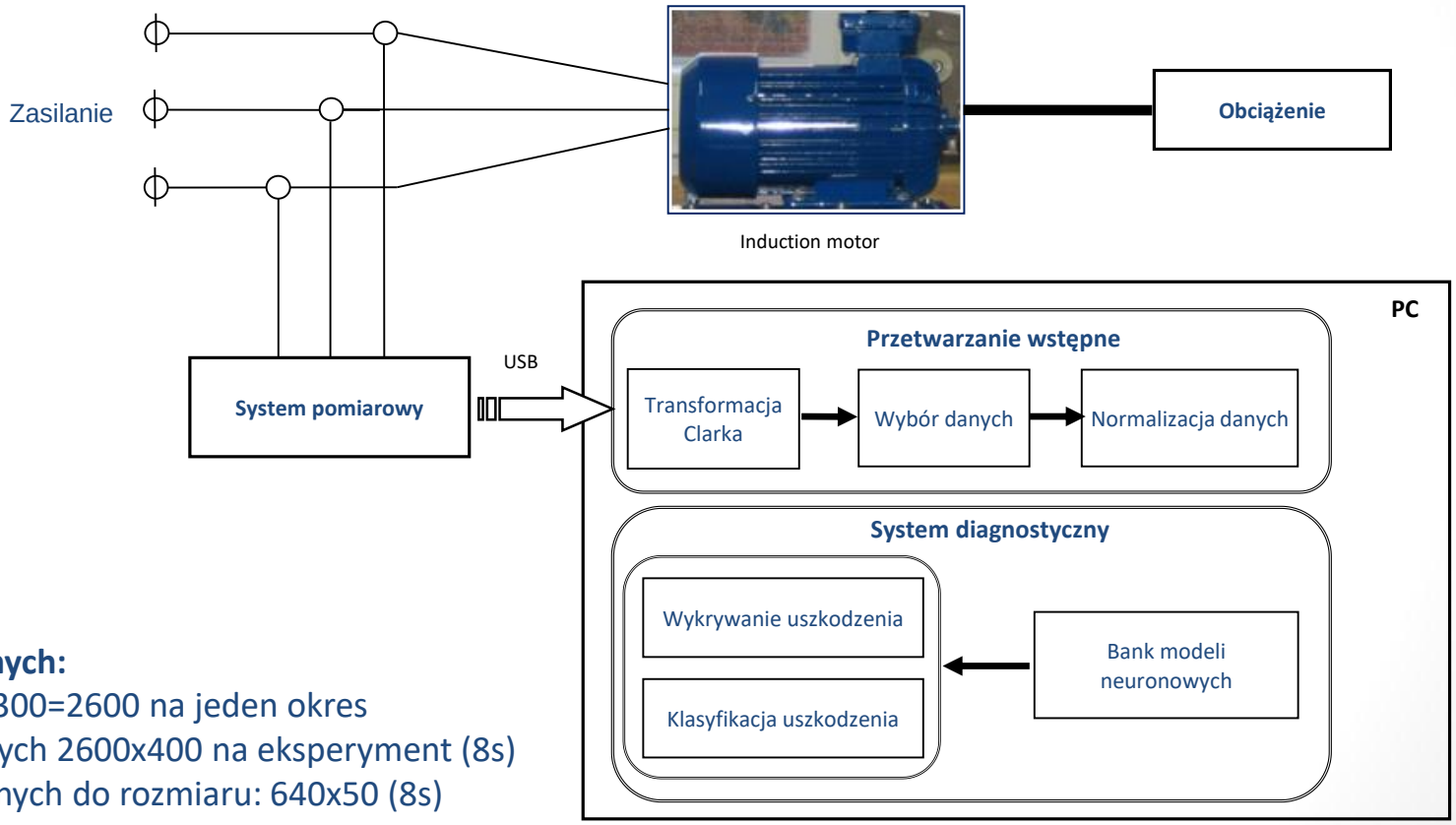


źródło: Use of neural networks in diagnostics of rolling-element bearing of the induction motor. L.Swędrowski, K.Duzinkiewicz,M.Grochowski,T.Rutkowski. V Międzynarodowy Kongres Diagnostyki Technicznej 2012 - Kraków

Zastosowania: Diagnostyka

Przykład: diagnostyka łożysk

Klasyczna sieć neuronowa



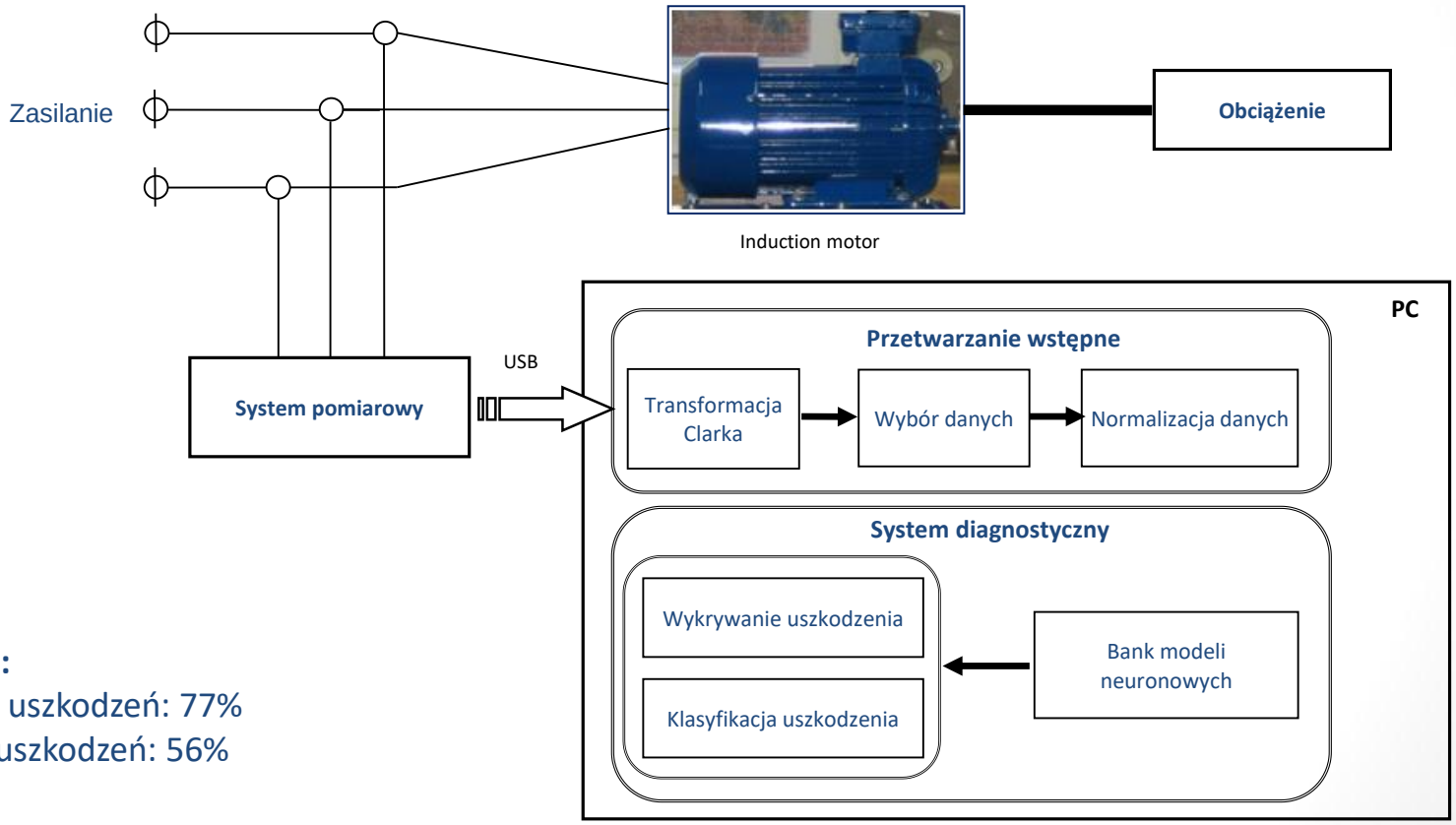
Rozmiar danych:
Wejście 2x1300=2600 na jeden okres
Macierz danych 2600x400 na eksperyment (8s)
Redukcja danych do rozmiaru: 640x50 (8s)

źródło: Use of neural networks in diagnostics of rolling-element bearing of the induction motor. L.Swędrowski, K.Duzinkiewicz,M.Grochowski,T.Rutkowski. V Międzynarodowy Kongres Diagnostyki Technicznej 2012 - Kraków

Zastosowania: Diagnostyka

Przykład: diagnostyka łożysk

Klasyczna sieć neuronowa



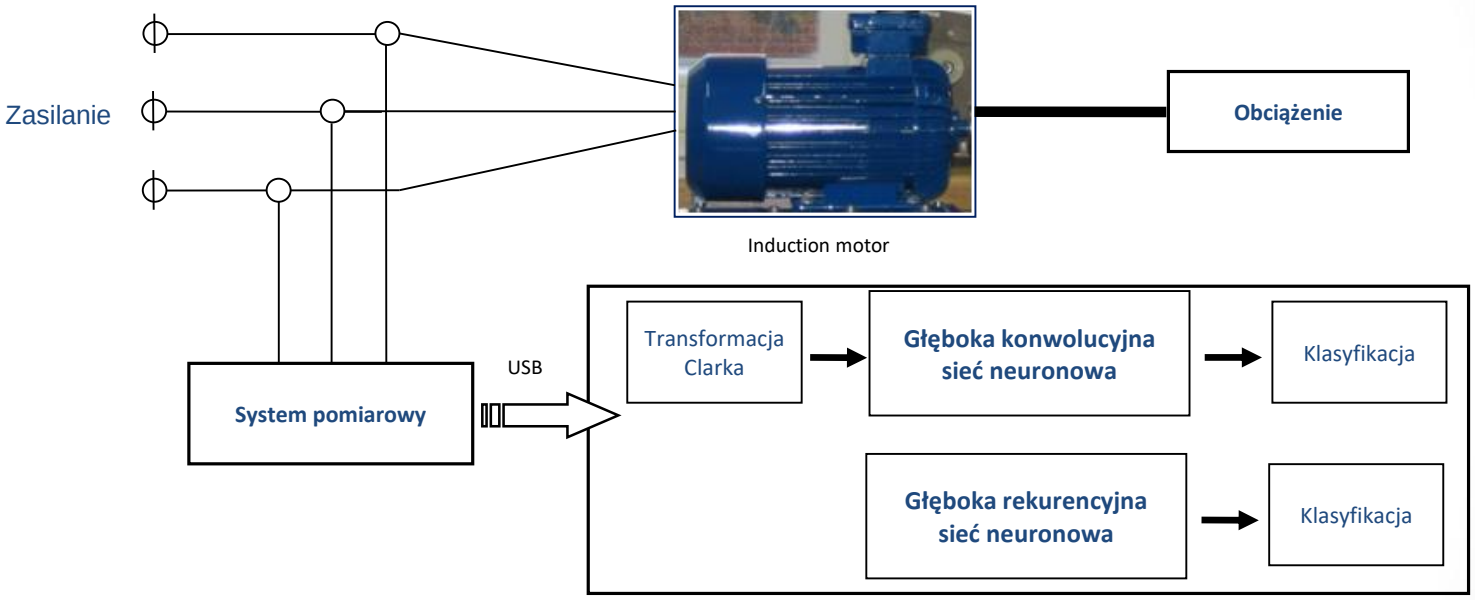
Skuteczność:
Wykrywanie uszkodzeń: 77%
Klasyfikacja uszkodzeń: 56%

źródło: Use of neural networks in diagnostics of rolling-element bearing of the induction motor. L.Swędrowski, K.Duzinkiewicz,M.Grochowski,T.Rutkowski. V Międzynarodowy Kongres Diagnostyki Technicznej 2012 - Kraków

Zastosowania: Diagnostyka

Przykład: diagnostyka łożysk

Głęboka sieć neuronowa



Zastosowania: Sterowanie

- Jako narzędzia do analizy informacji w postaci obrazu
- Modelowanie nieliniowych, złożonych procesów lub obiektów
- Prognozowanie
- Opracowywanie optymalnych strategii poruszania się przez pojazdy autonomiczne, roboty przemysłowe (DNN + RL)

Zalety głębokich sieci neuronowych:

- Zdolność do gromadzenia dużej ilości „wiedzy”
- Wysoka skuteczność działania (szczególnie w analizie obrazów)
- Zdolność rozpoznawania i klasyfikacji wielu klas (nawet tysiące)
- Duża elastyczność
- Zdolności do samouczenia (sieci ewolucyjne, RL, podejścia hybrydowe)
- Aspekty sprzyjające rozwojowi głębokich sieci neuronowych
  - Olbrzymie nakłady finansowe
  - Otwarte środowiska programistyczne
  - Otwarte bazy danych (choć z pewnym opóźnieniem …)
  - Otwarte kody
  - Bardzo szybka prezentacja wyników (tygodnie)
  - Olbrzymia konkurencja
  - Niemal natychmiastowe wprowadzanie rozwiązań do „biznesu”

Wady głębokich sieci neuronowych:

- Struktura głębokiej sieci jest typu „very dark black box”,
  - Nie do końca wiadomo czego sieć się tak naprawdę uczy

Przykłady błędnie sklasyfikowanych

Kotów






Psów






Wady głębokich sieci neuronowych:

- Struktura sieci typu „very dark” black box,
  - Nie do końca wiadomo czego sieć się tak naprawdę uczy
  - Trudno wyjaśnić osiągnięty rezultat czy diagnozę
    - Kłopoty prawne, np. dopuszczanie/certyfikacja systemów wspomagania decyzji w medycynie, finansach, pojazdach bezzałogowych itp.,
- Potrzebują olbrzymich zbiorów danych (zbiory rzędu nawet kilku TB) do efektywnej nauki oraz stosunkowo dużej mocy obliczeniowej,
- W niektórych przypadkach łatwo je „oszukać”,
- Brak klasycznych podstaw teoretycznych typu dowody stabilności, zbieżności, krzepkości …

Wyzwania, otwarte problemy:

- Głębokie sieci neuronowe bardzo często są przewymiarowane i często źle ustrukturyzowane:
  - Wiele problemów jest rozwiązywanych (skutecznie) tą samą siecią,
  - Często sposobem na zwiększenie dokładności jest: *lets go deeper!*
  - Skuteczność „transfer learning” zdaje się potwierdzać ten fakt
- Istnieje silna potrzeba znalezienia algorytmu/metody efektywnego doboru struktury sieci do rozwiązywanego konkretnego problemu (podejścia ewolucyjne, reinforcement learning, heurystyki)
- Opracowanie skutecznych metod inicjalizacji sieci (mogą okazać się pomocne metody klasycznej teorii sterowania, optymalizacji, programowania dynamicznego ...)
- Opracowanie metod efektywnego uczenia w obliczu deficytu i/lub braku balansu klas w danych uczących
- Opracowanie metod tłumaczenia („wizualizacji”) sposobu dojścia głębokiej sieci do danego rozwiązania
- Nauka wnioskowania zamiast zapamiętywania

Zastosowania: Systemy wspomagania decyzji

- **Rozpoznawanie i analiza sceny:**
  - robotyka, pojazdy bezzałogowe, wspomaganie kierowców, osoby niedowidzące…

Real-time event detection for video surveillance applications

https://www.youtube.com/watch?v=QcCjmWwEUgg

https://www.youtube.com/watch?v=Cgxsv1riJhI

https://www.youtube.com/watch?v=opsmd5yuBF0

Zastosowania: Sterowanie

- **Sterowanie predykcyjne MPC**

Aggressive Deep Driving: Combining Convolutional Neural Networks and Model Predictive Control

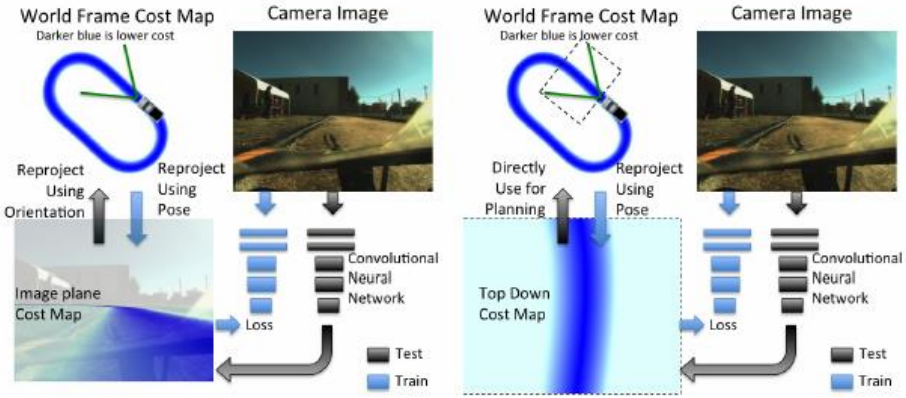


Figure 1: (left) Image plane cost map regression. Camera image and position on a world map are combined to label driveability of image pixels. (right) A top down projection of the cost map can be used as a training target.

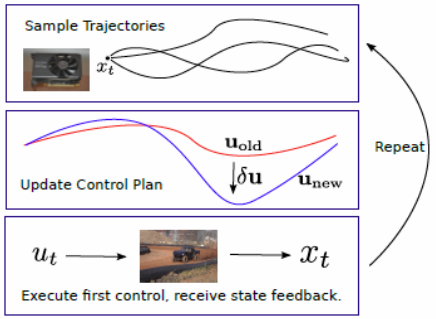
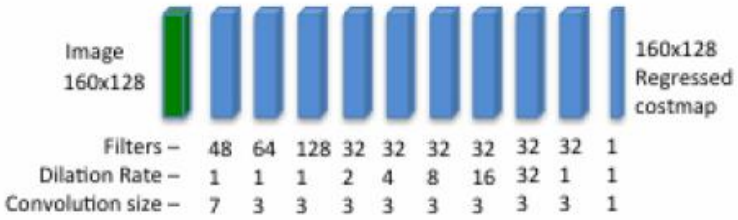
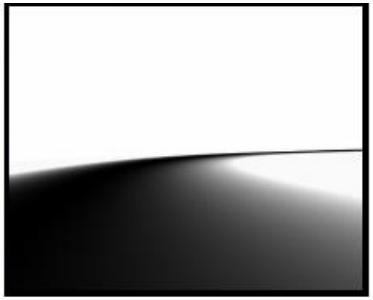
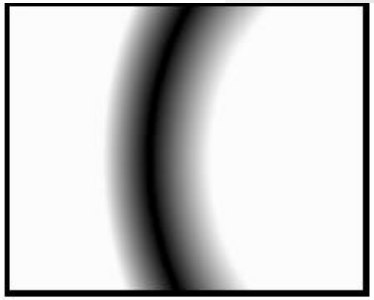


Figure 3: Model Predictive Path Integral Control algorithm. Trajectories are sampled and updated using a GPU, and the first control executed.







DeepDream

źródło: CS231n: Convolutional Neural Networks for Visual Recognition, 2016 – Lecture6. Stanford University

Dziękuję za uwagę