Nowoczesne sieci neuronowe w przetwarzaniu danych

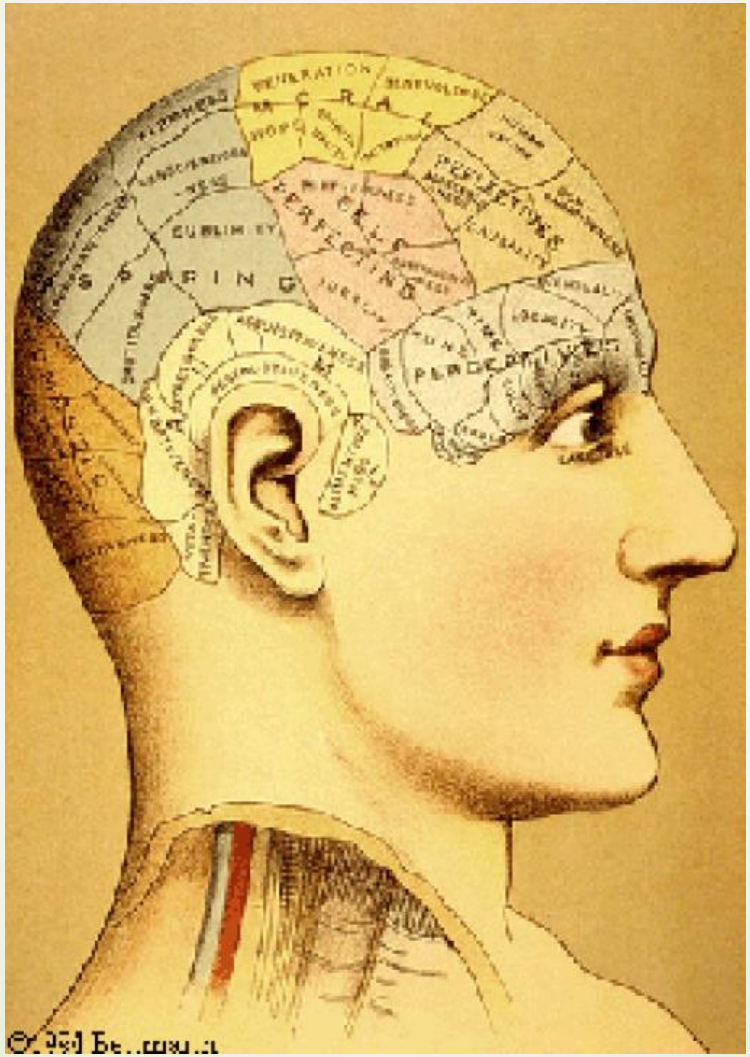
Wykład

dr hab. inż. **Piotr A. Kowalski**, prof. AGH

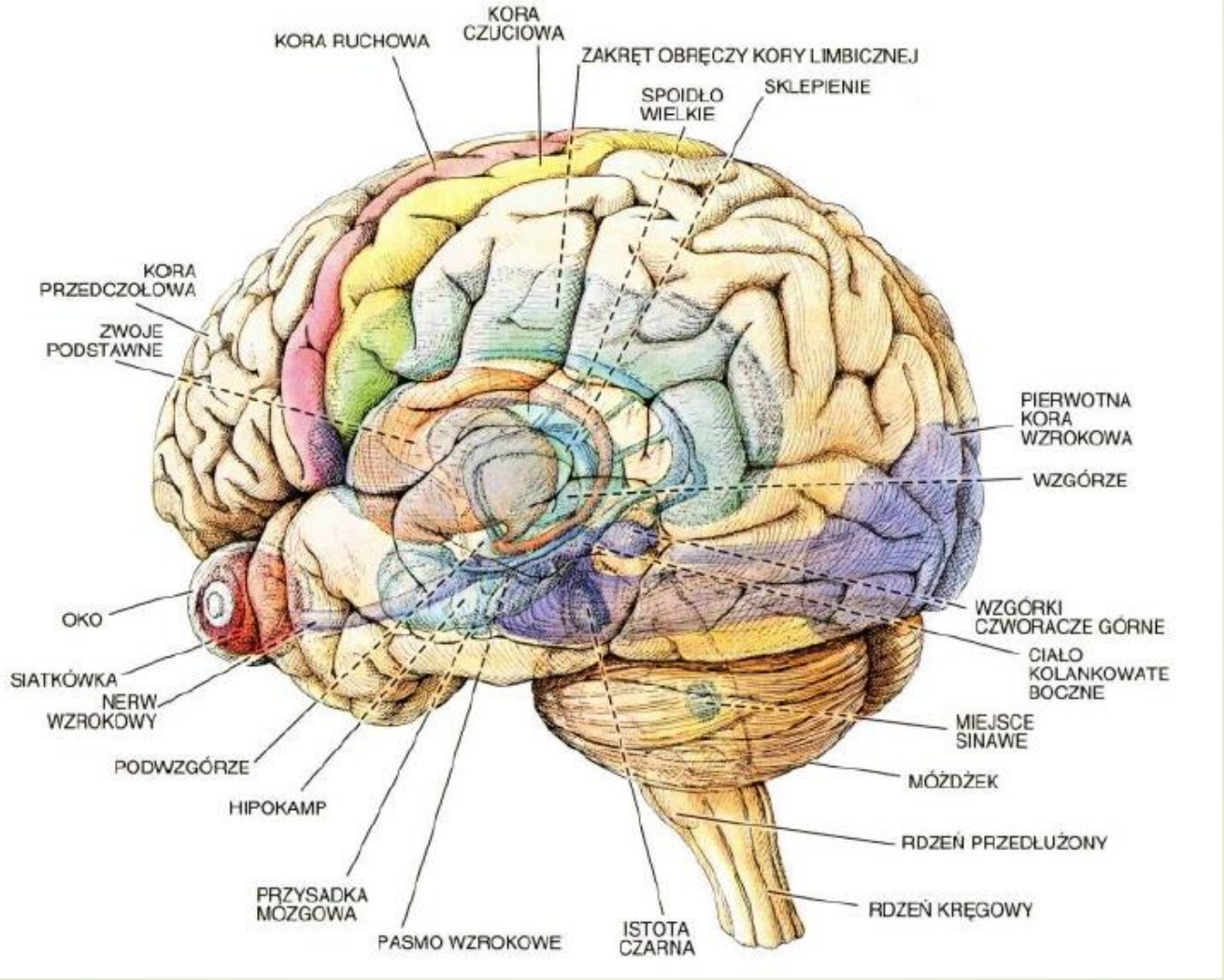
Katedra Informatyki Stosowanej i Fizyki Komputerowej
Wydział Fizyki i Informatyki Stosowanej AGH

Instytut Badań Systemowych PAN

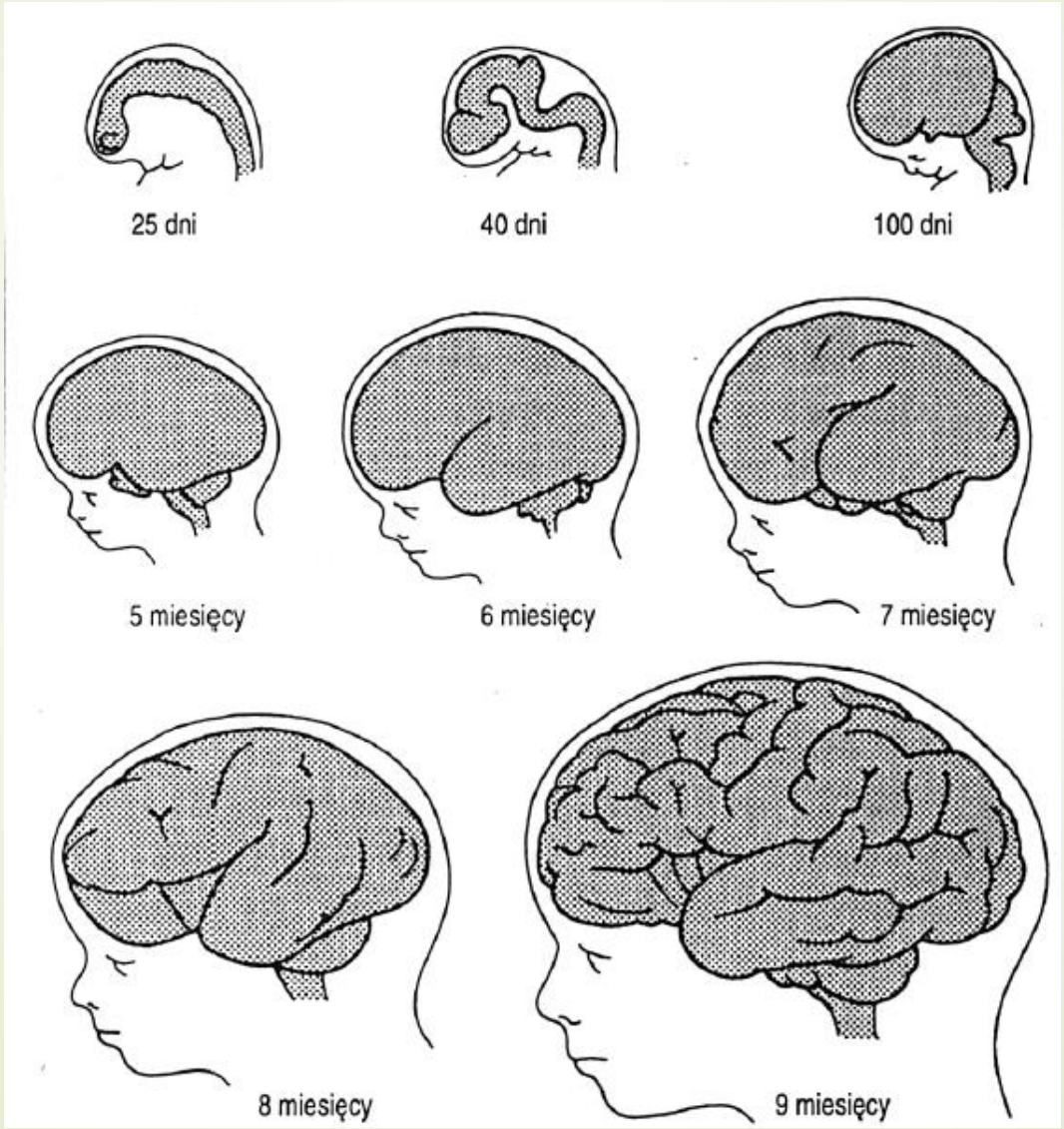
Inspiracja biologiczna



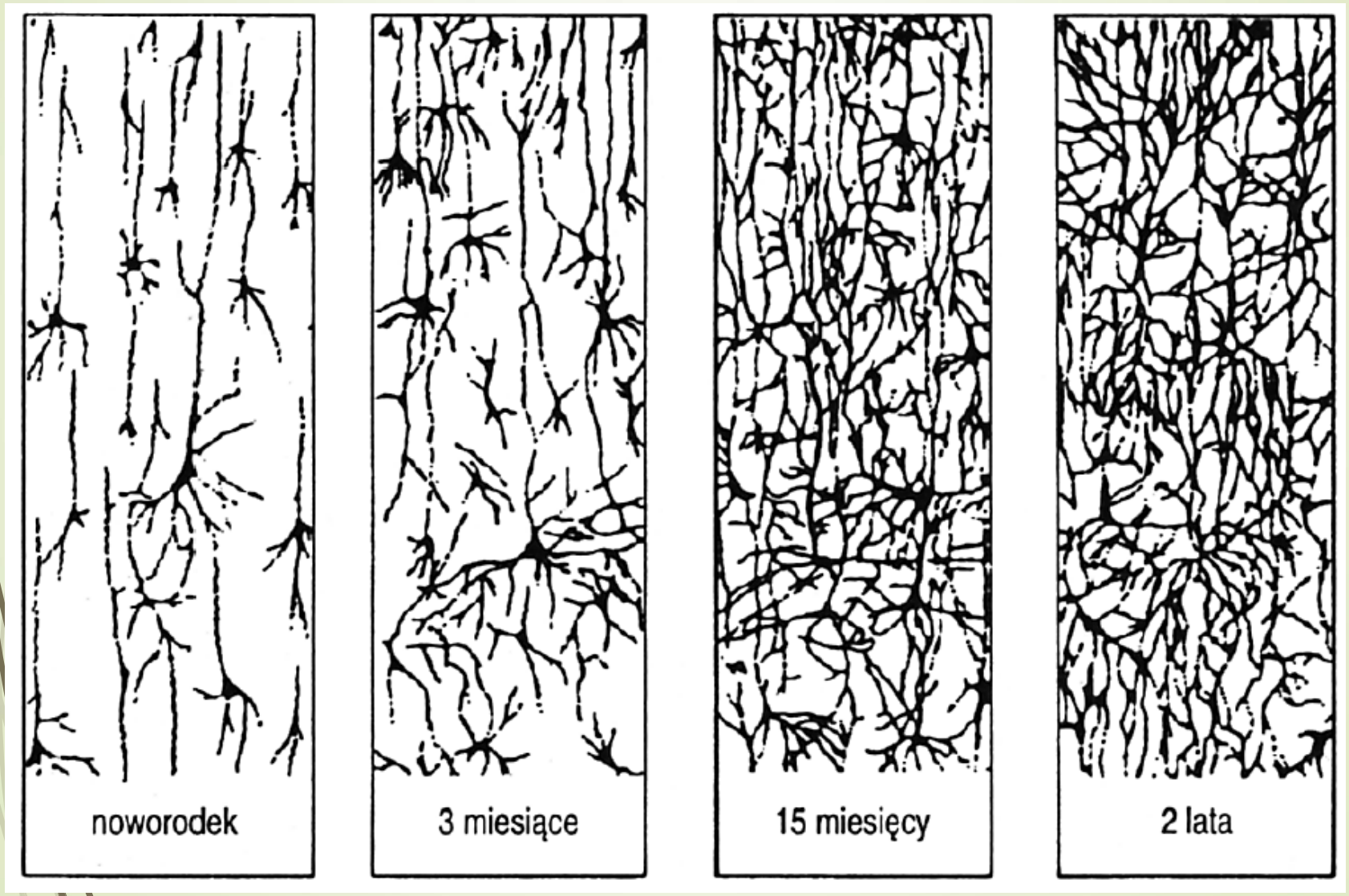
Mózg ludzki – specjalizacja



Mózg ludzki – rozwój

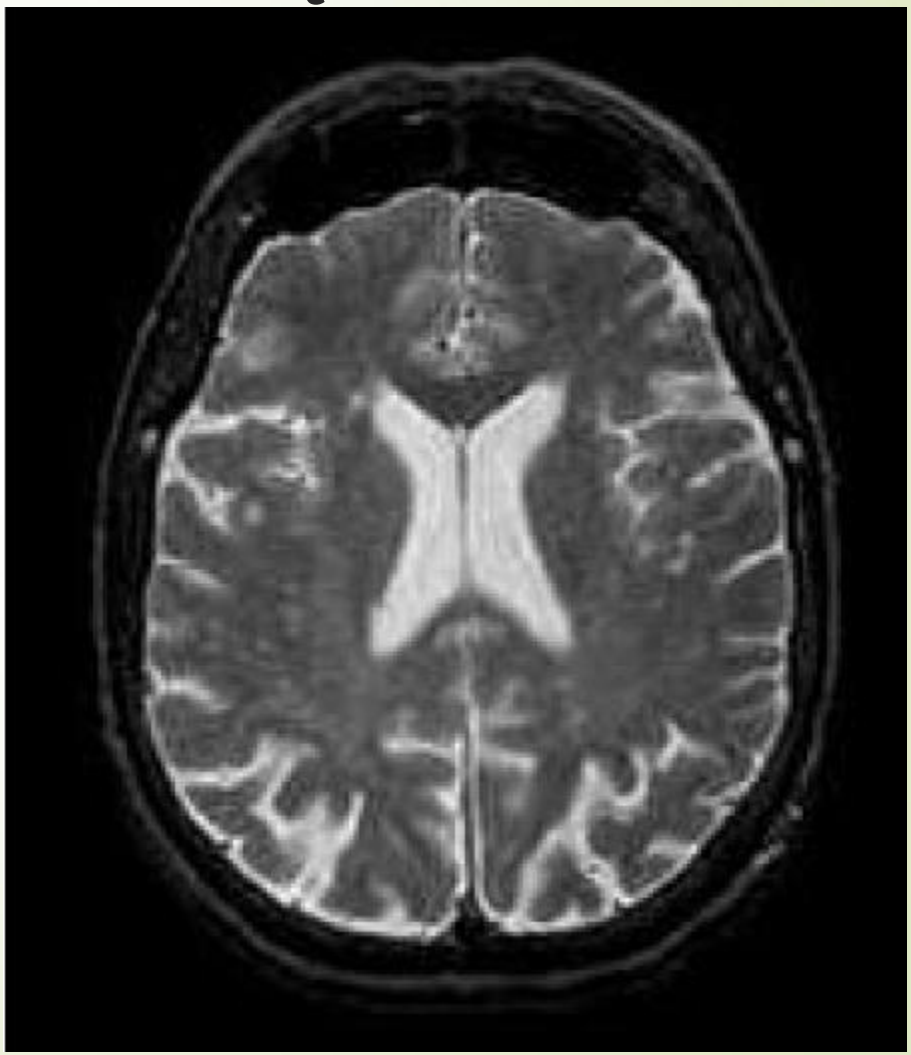


Mózg ludzki – rozwój kory



Mózg ludzki – badanie aktywności metodą RM

- Obrazowanie metodą rezonansu magnetycznego



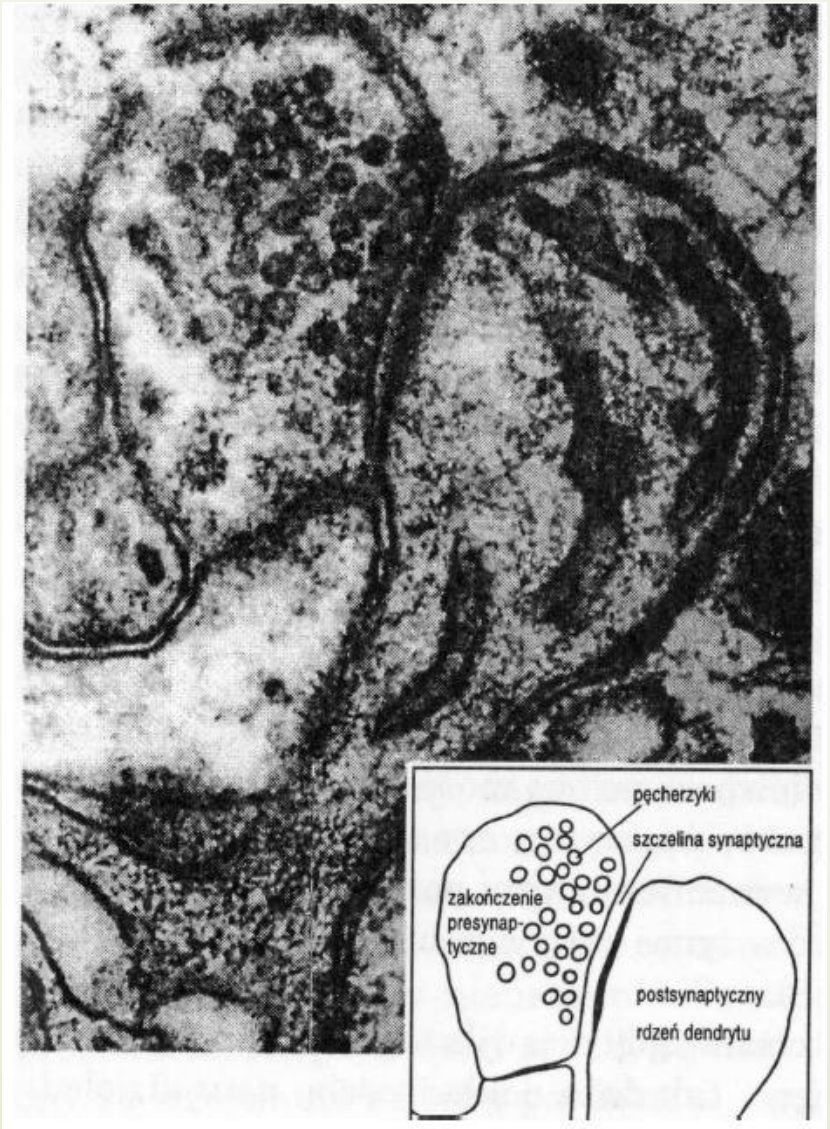
Mózg ludzki – badanie aktywności metodą RM

- Obrazowanie metodą rezonansu magnetycznego

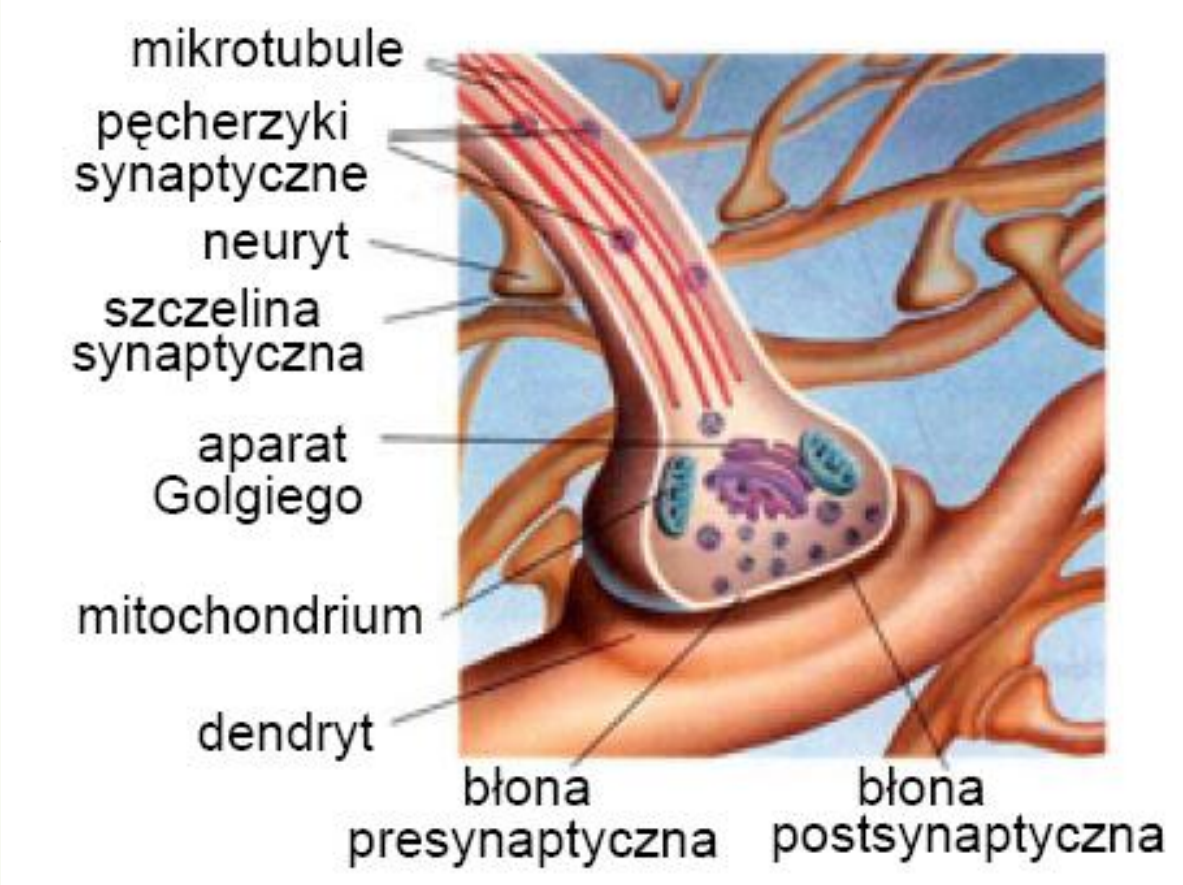
Neuron


400X
Neuron
➔Mózg jako wydajna maszyna licząca.
➔Wzorowanie się na budowie mózgu człowieka.
➔Systemy przetwarzające informację.

Wielka lupa - obraz synapsy



Schemat budowy zakończenia nerwowego i synapsy



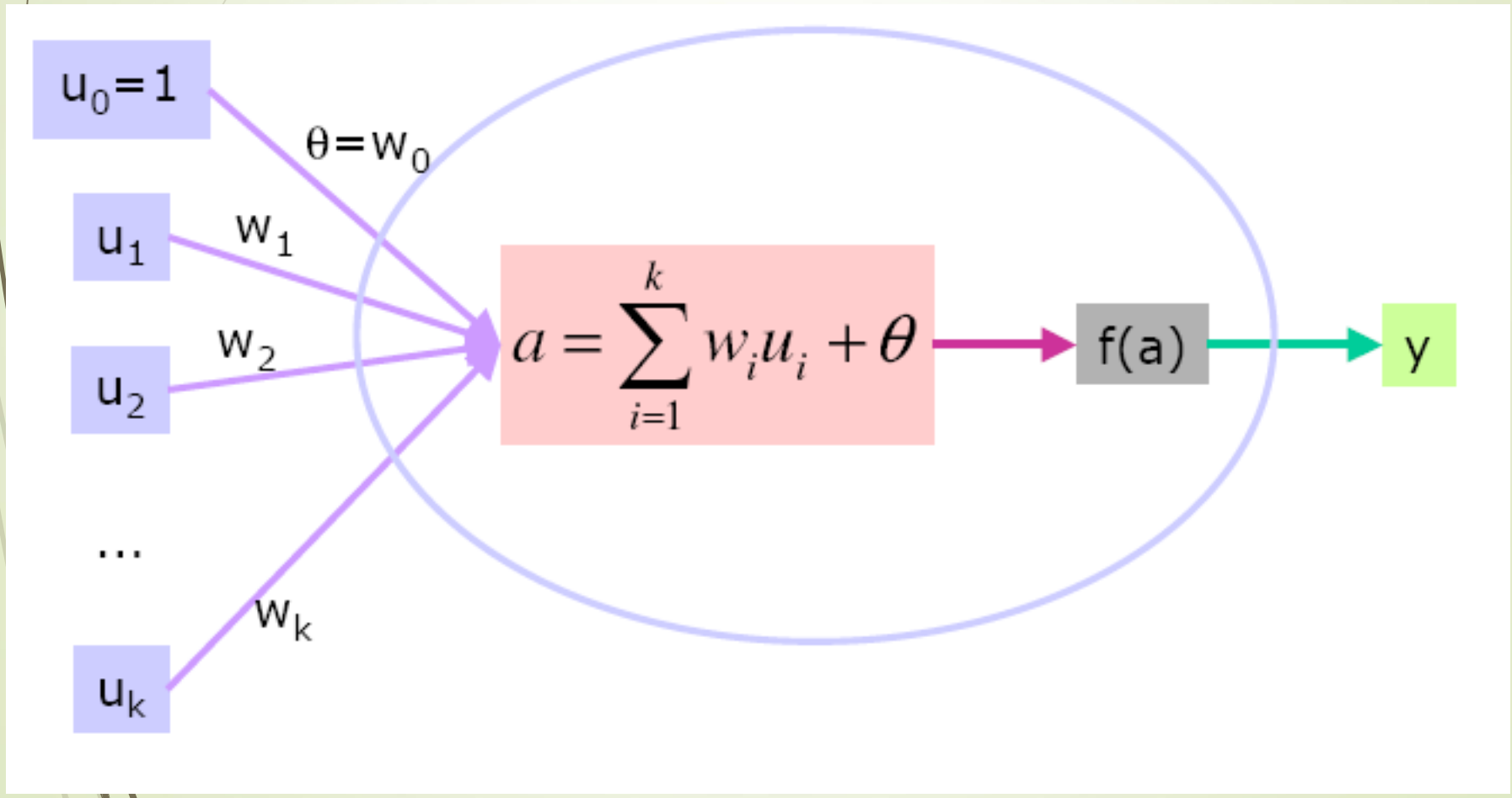
Mózg (bio) – dane podstawowe

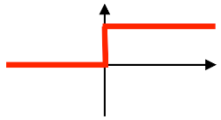
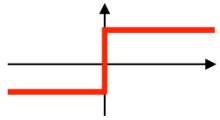
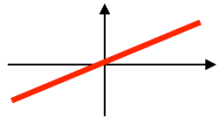
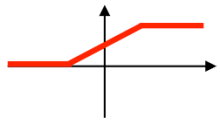
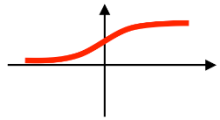
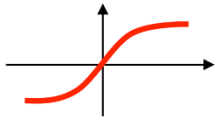
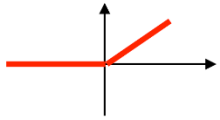
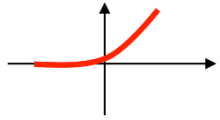
- Mózg składa się z elementarnych komórek nerwowych – **neuronów.**
- Ilość neuronów w mózgu to ok**. 10 miliardów.**
- Każdy neuron jest **połączony z 10^4 innych komórek.**
- W elementarnej komórce (neuron) przeprowadzane są „proste obliczenia” (ich natura nie jest do końca jasna).
- Przez neurony połączone w sieć, przechodzą **impulsy elektryczne** z różną częstotliwością (1- 100 Hz) i o różnej amplitudzie.
- Mózg jest skuteczny dzięki **zrównolegleniu i szeregowaniu zadań**.

Model sztucznego neuronu McCulloch-Pitts'a

- 1943 rok – pierwszy matematyczny opis neuronu.
- Najprostszy neuron można sobie wyobrazić jako przetwornik, który pobiera informację ze wszystkich wejść i na ich podstawie emituje sygnał wyjściowy.
- Każde wejście jest mnożone przez pewną wartość zwaną wagą (wzmocnienie lub osłabienie sygnału).
- Sygnały wejściowe są sumowane, by następnie dopasować odpowiedź za pomocą funkcji aktywacji.

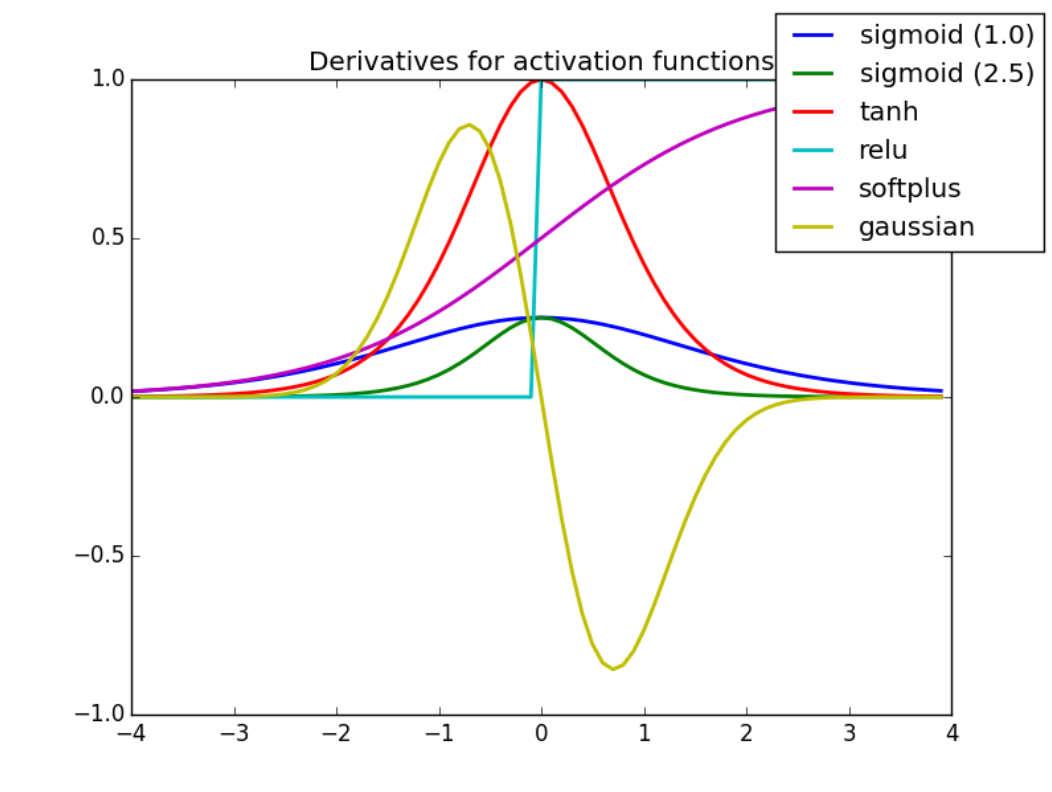
Model sztucznego neuronu McCulloch-Pitts'a



Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^{z})$	Multi-layer Neural Networks	

Dlaczego takie funkcje?

Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^{z})$	Multi-layer Neural Networks	



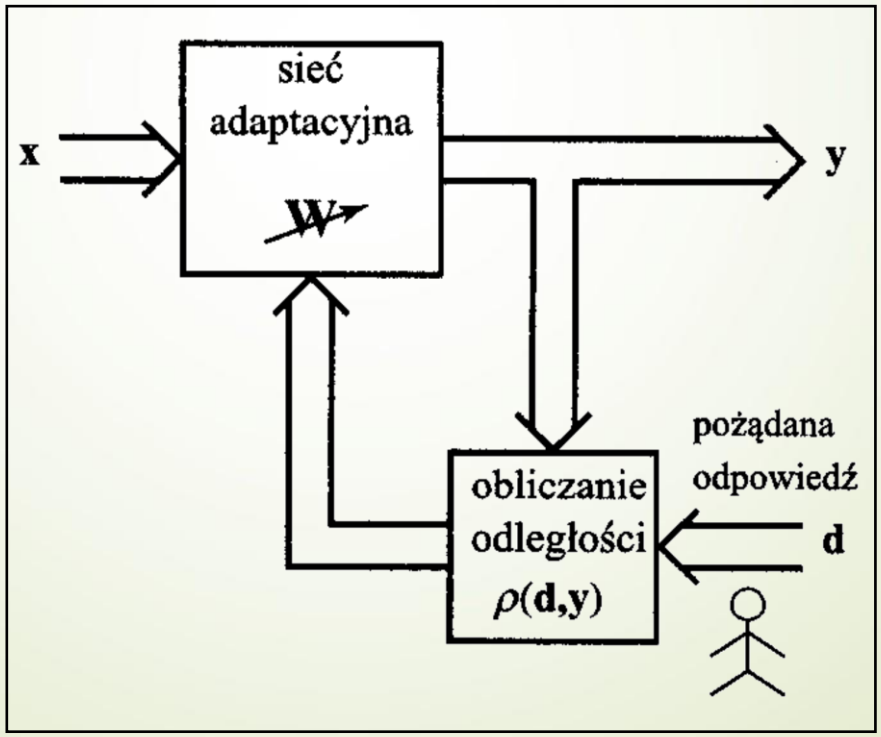
Cechy sztucznego neuronu - podsumowanie

- Wejścia i wagi są **liczbami rzeczywistymi** dodatnimi bądź ujemnymi.
- Jeżeli jakaś cecha (wejście) powoduje **pobudzenie** neuronu, **waga będzie dodatnia**, jeżeli cecha ma charakter **hamujący waga jest ujemna**.
- Neuron dokonuje **sumowania** i **dopasowania do progu** – biasu θ
- Przyjmuje się traktowanie **progu** θ jako wagi w_0, gdzie wejście zawsze jest równe 1.

Uczenie z nauczycielem

Uczenie z nauczycielem – **nadzorowane** - dla każdego wektora wejściowego wchodzącego w skład zbioru uczącego znana jest poprawna odpowiedz.

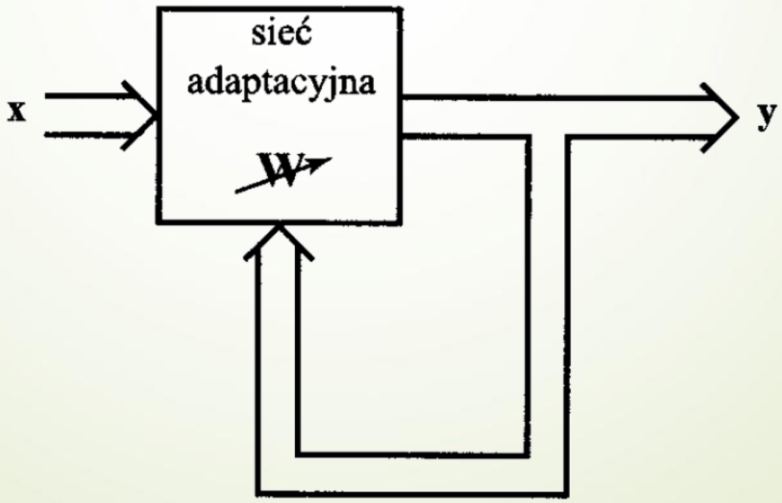
Korekcja wag opiera się na różnicy pomiędzy rzeczywistą a pożądaną odpowiedzią sieci.



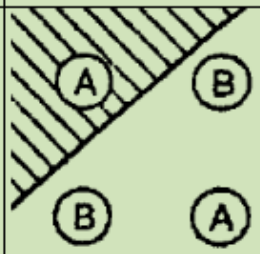
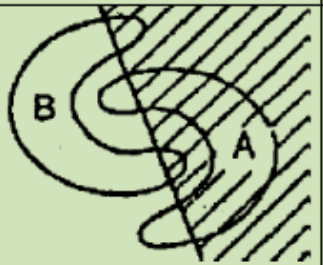
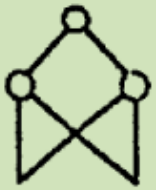
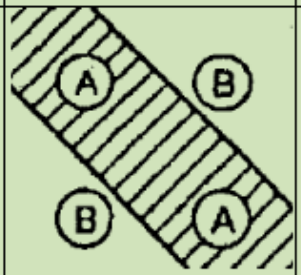
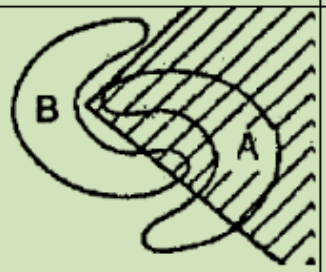
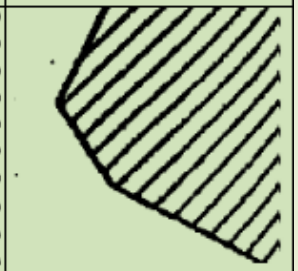
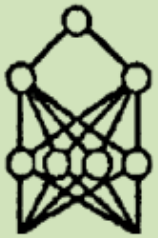
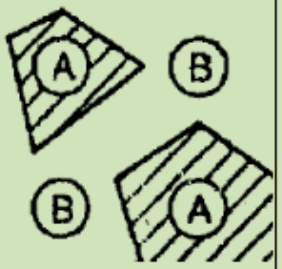
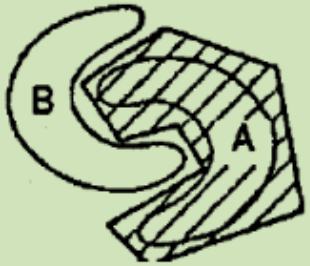
Uczenie bez nauczyciela

Uczenie bez nauczycielem – **nienadzorowane** – pożądana odpowiedź sieci nie jest znana.

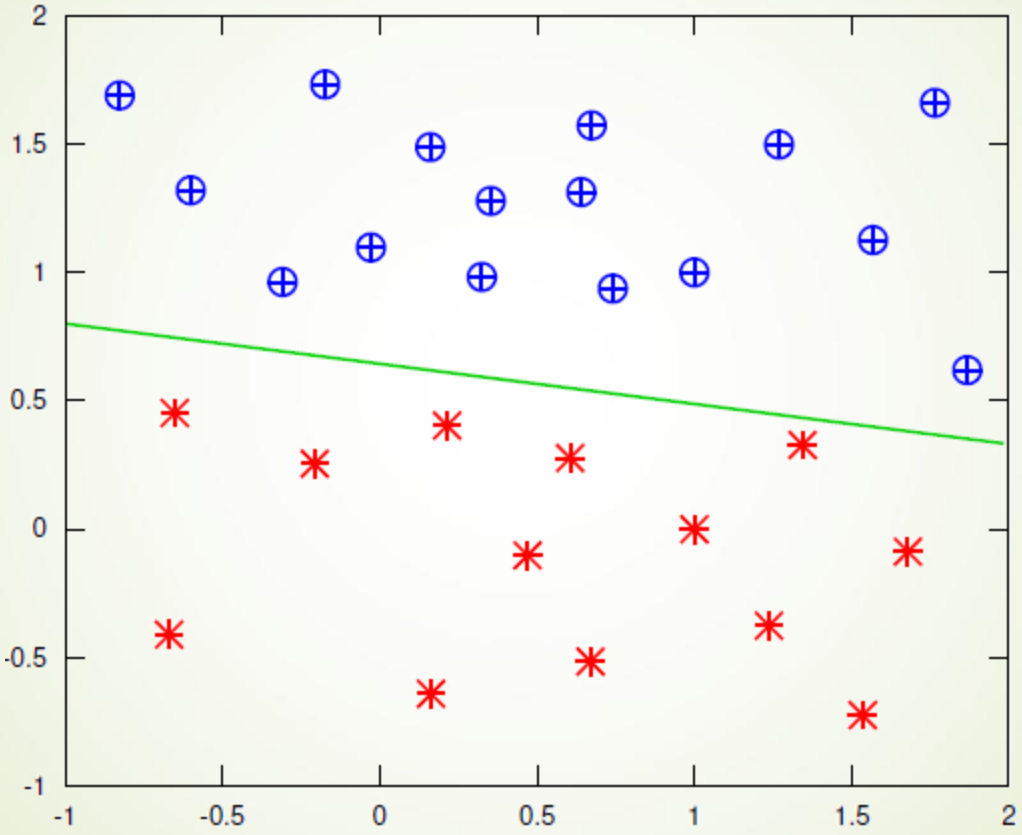
Sieć uczy się poprzez analizę reakcji na pobudzenia. W trakcie wykrywania np. skupisk w danych, parametry sieci podlegają zmianom, co nazywamy samoorganizacją.



Obszary decyzyjne klasyfikatorów wielowarstwowych

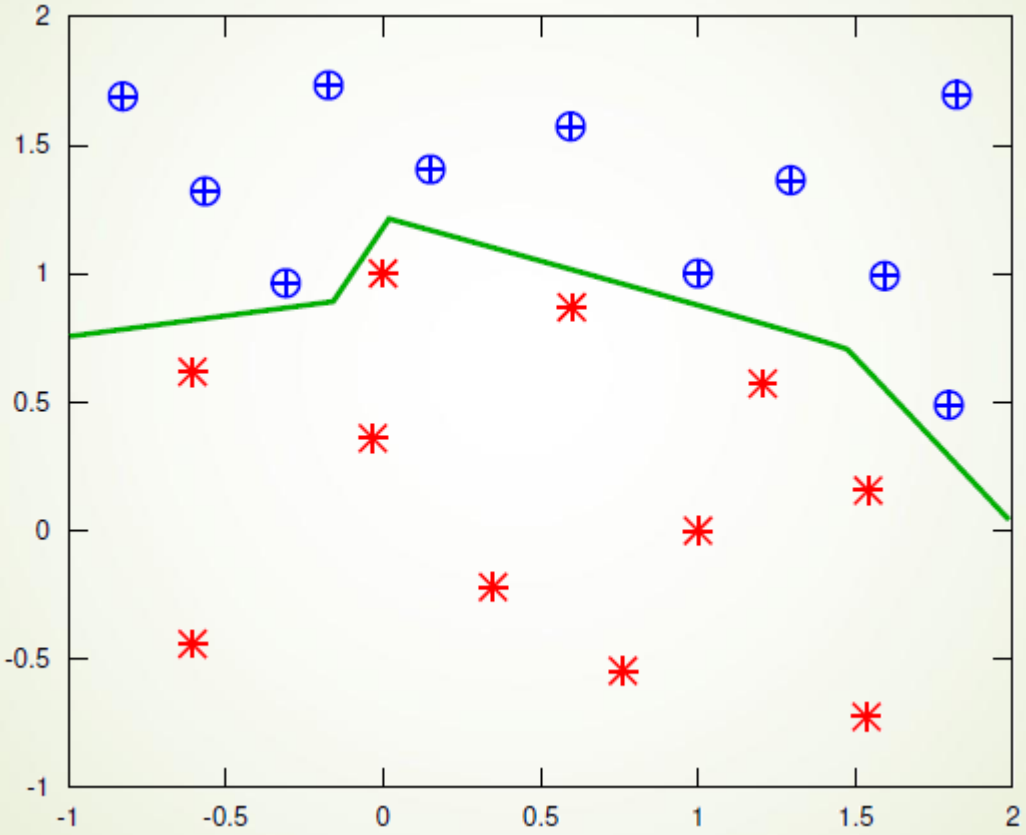
STRUKTURA	REGION DECYZJI	PROBLEM XOR	KLASY I PODZIAŁ	KSZTAŁTY REJONÓW
1 warstwa 	Półpłaszczyzny			
2 warstwy 	Wypukłe otwarte i zamknięte obszary			
3 warstwy 	Wieloboki z ograniczoną liczbą kątów			

Dlaczego sieci a nie perceptron ?



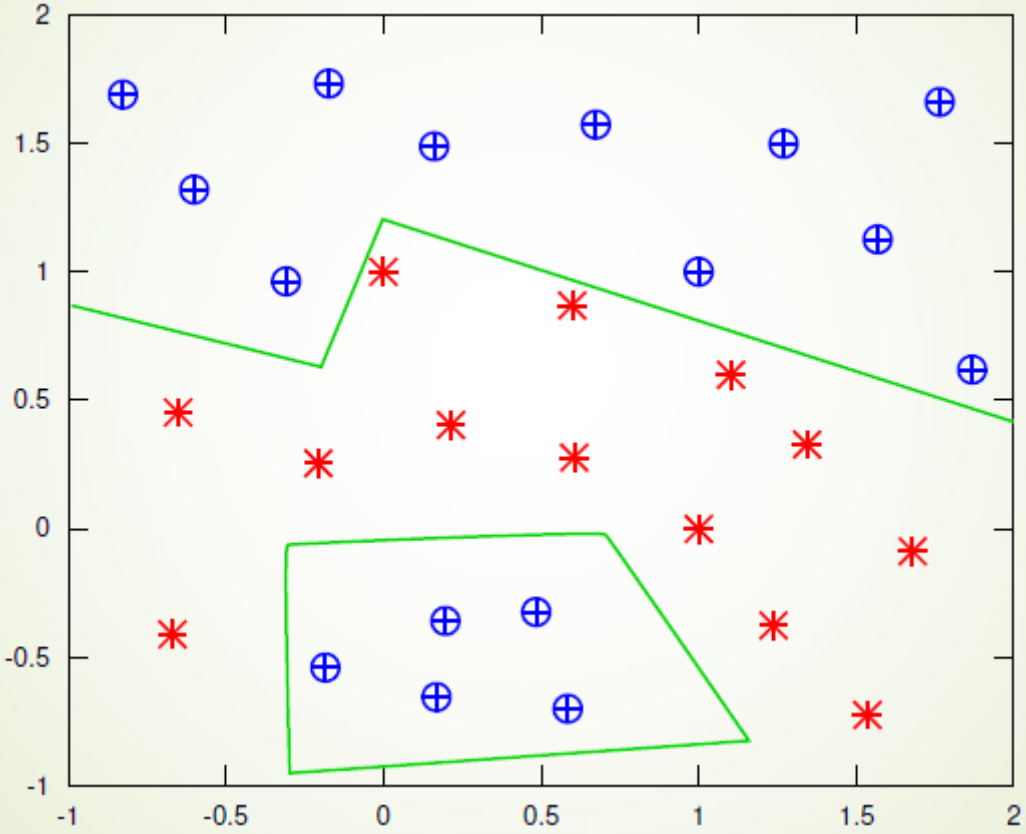
Warstwa neuronów jest w stanie aproksymować **funkcje linowe**

Dlaczego sieci a nie perceptron ?



Dodatkowa warstwa neuronów UKRYTYCH jest w stanie aproksymować funkcje ciągłe

Dlaczego sieci a nie perceptron ?

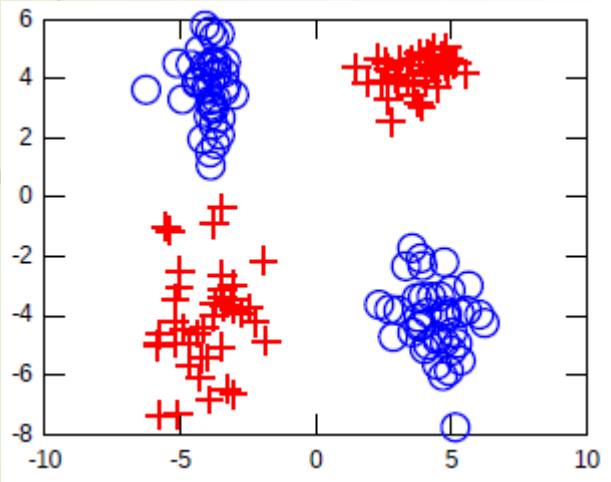


Kolejne warstwy neuronów **ukrytych** pozwalają na aproksymację **funkcje nieciągłych**

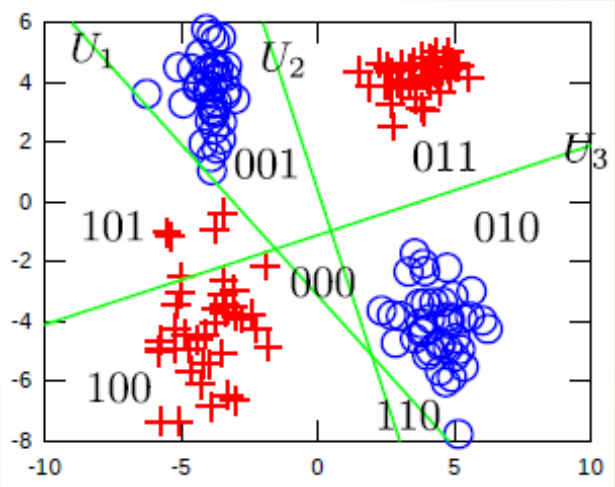
Obszary decyzyjne klasyfikatorów wielowarstwowych

Budujemy SSN

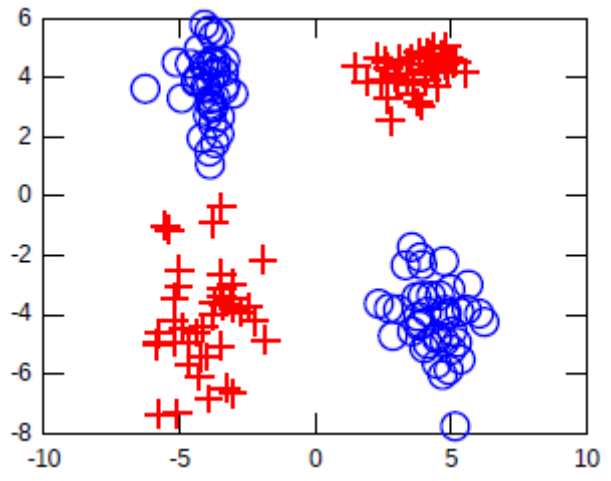
DANE



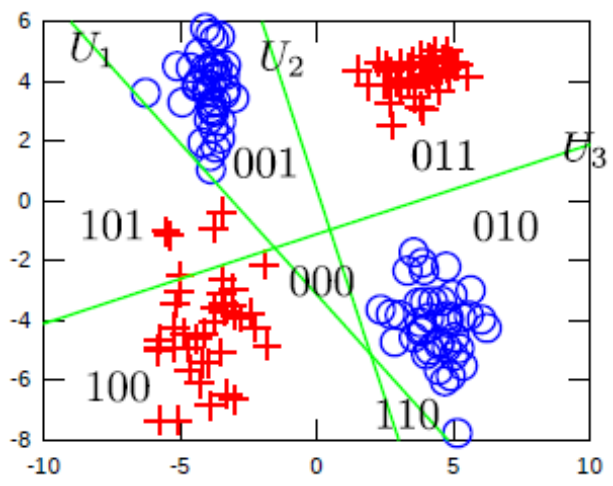
Warstwa neuronów

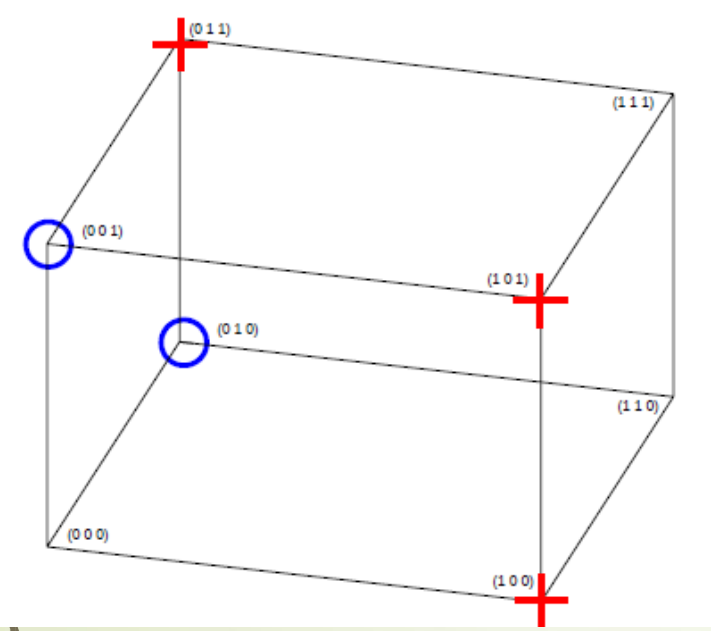


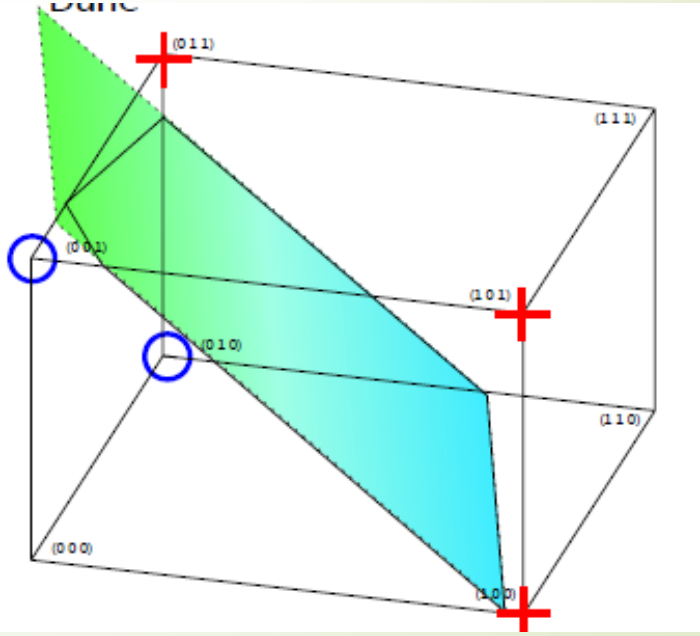
2 neurony wejściowe – 3 neurony w następnej warstwie



decyzyjne kl
rstwowych

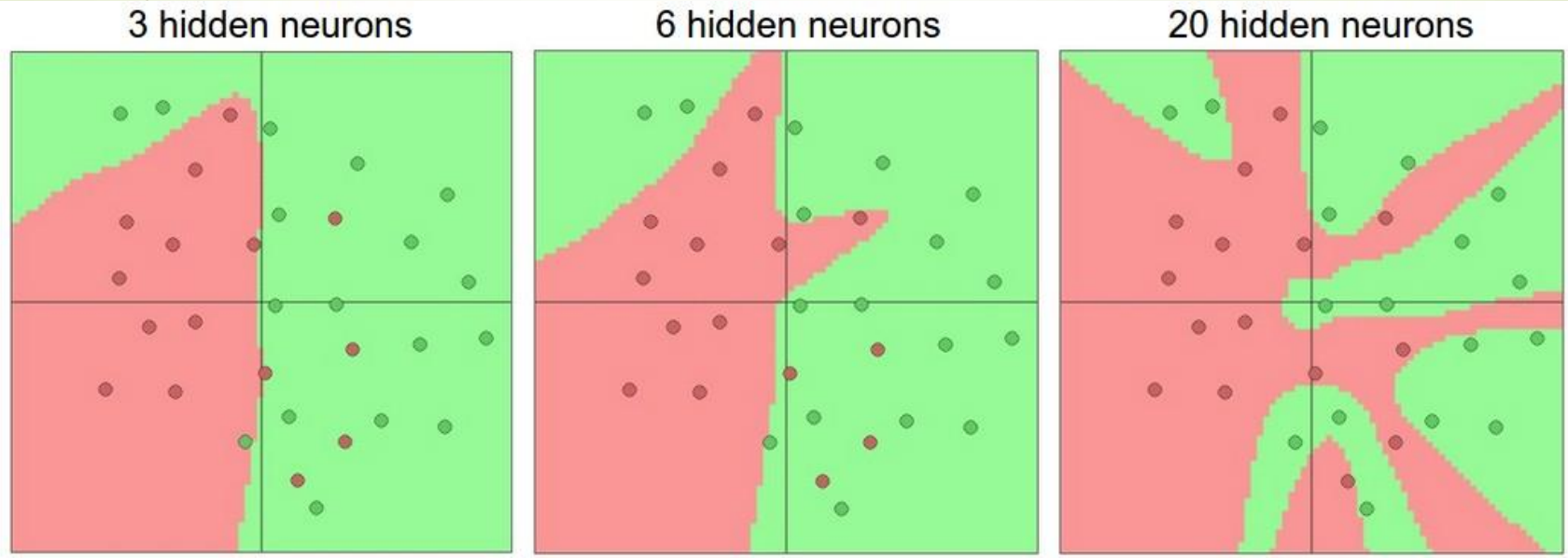






2 neurony wejściowe – 3 neurony w warstwie ukrytej – 1 wyjściowy

To jaka powinna być struktura SSN dla danego zadania?

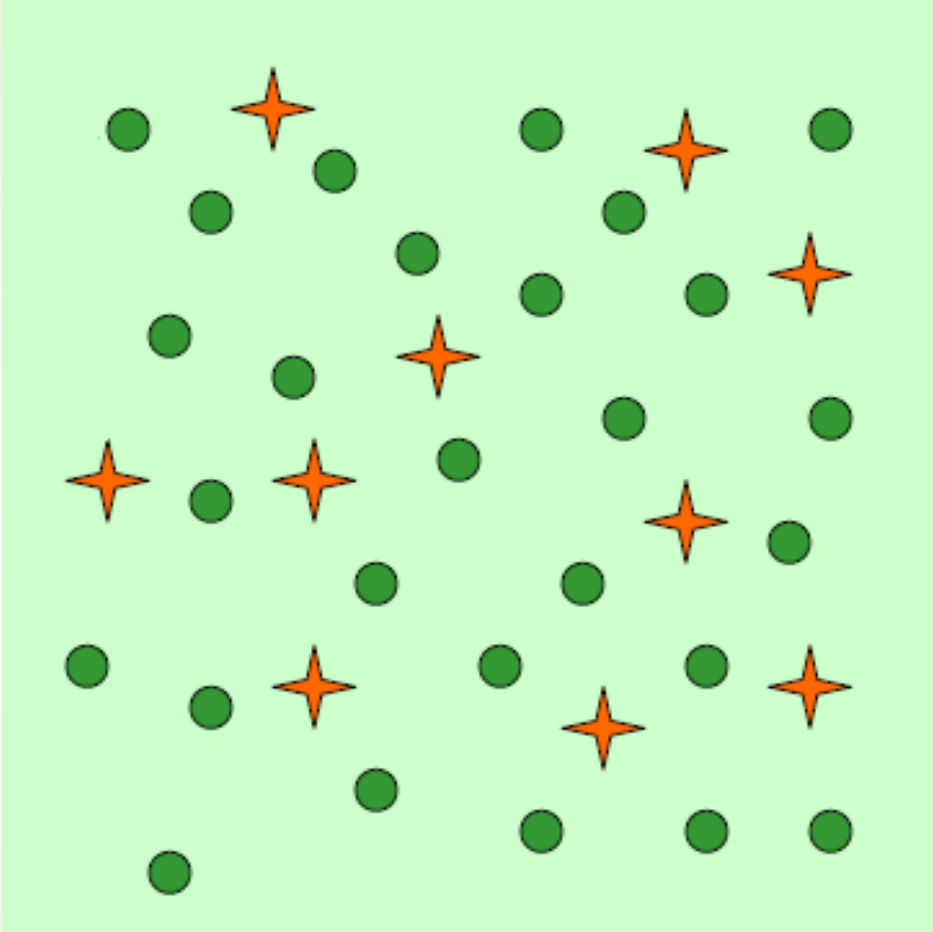


Problem dobrej generalizacji sieci

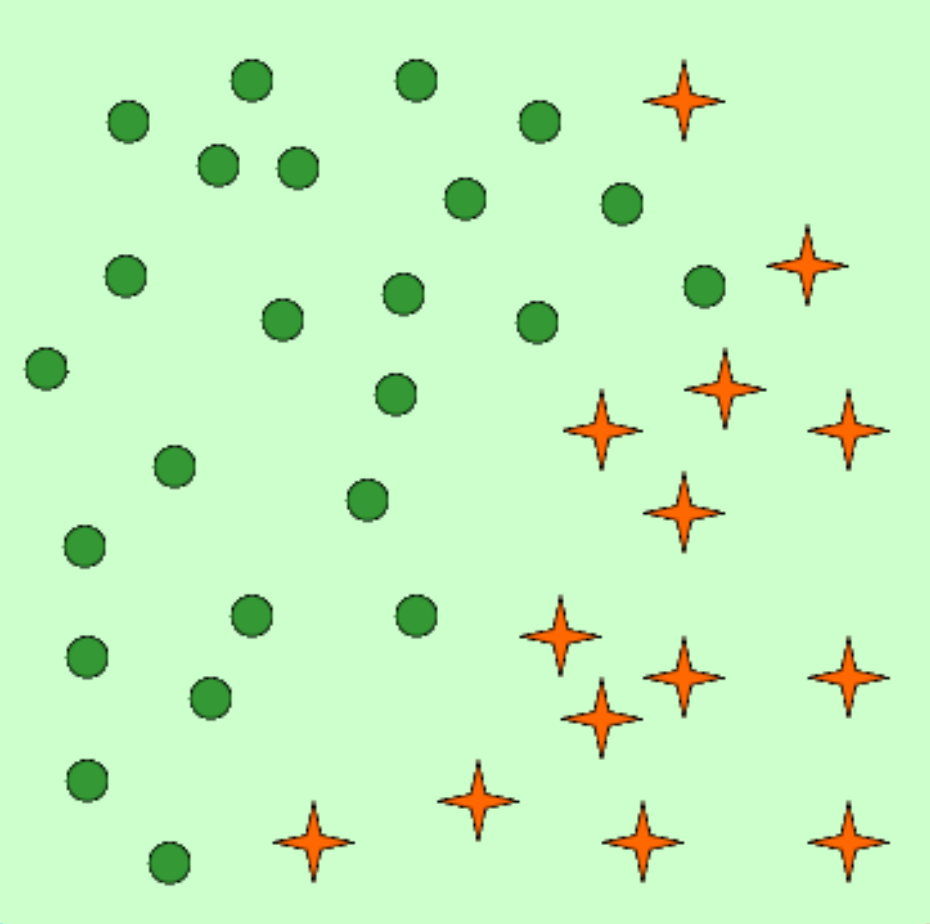
Teoretycznie, można zapewnić strukturę sieci, która pozwoli zminimalizować błąd sieci do wartości bliskich zera. Jednak**, celem uczenia** sieci jest **aproksymacja ogólnej „reguły" wiążącej zależności wejściowo-wyjściowe**, a nie optymalizacja struktury sieci dla, nie zawsze w pełni reprezentatywnej, próby **zbioru trenującego**.

Poprawność działania sieci należy zawsze weryfikować dla tzw. zbioru testującego, t.j. wzorców par $\{x_i,d_i\}$, i=1,…,P, **które nie były wykorzystywane w trakcie nauki sieci**.

Zbiór testujący można utworzyć na **drodze losowego wyboru** np. 10% **wzorców** z ogólnej liczby dostępnych danych.

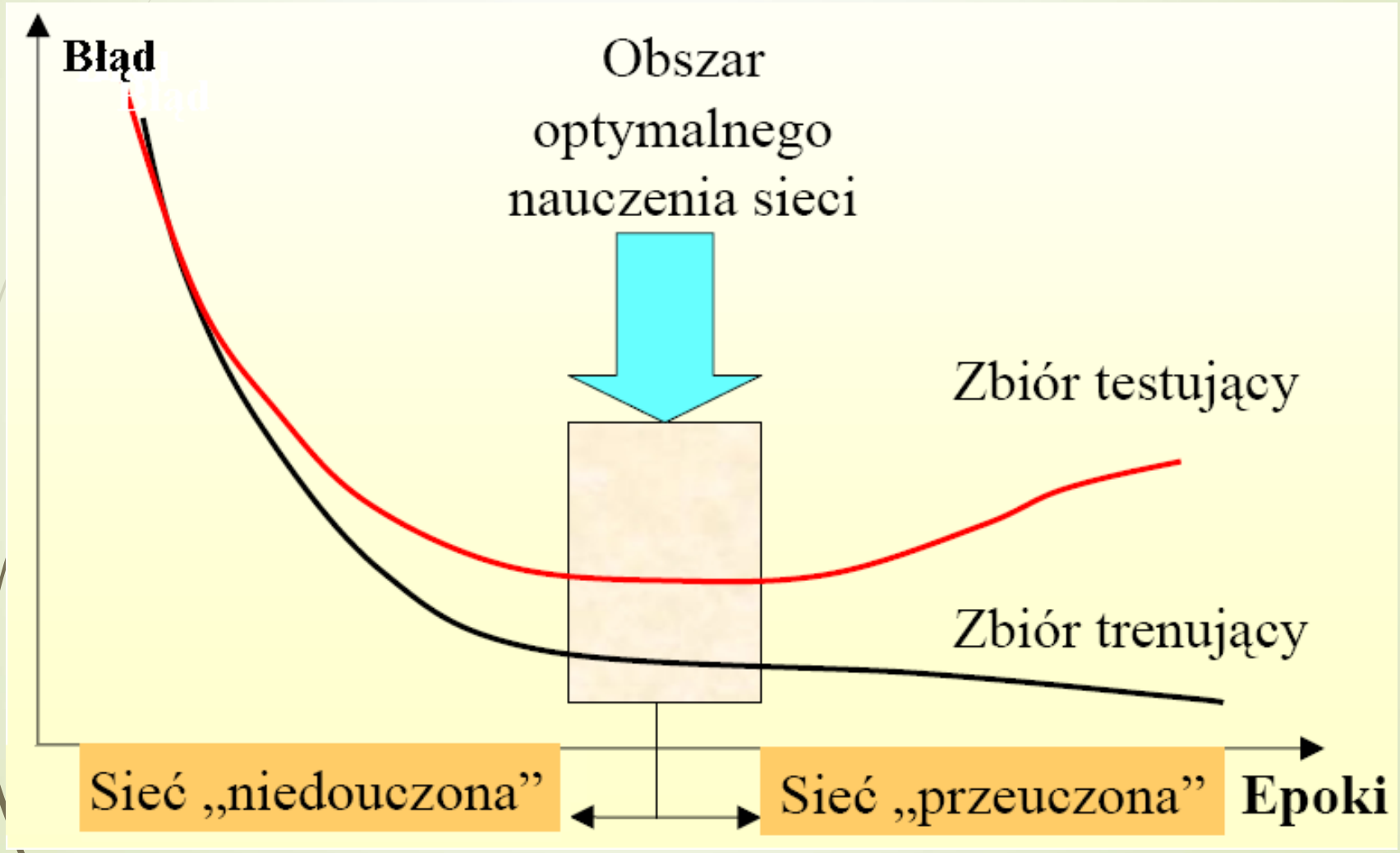


Prawidłowy podział na zbiór testujący i trenujący

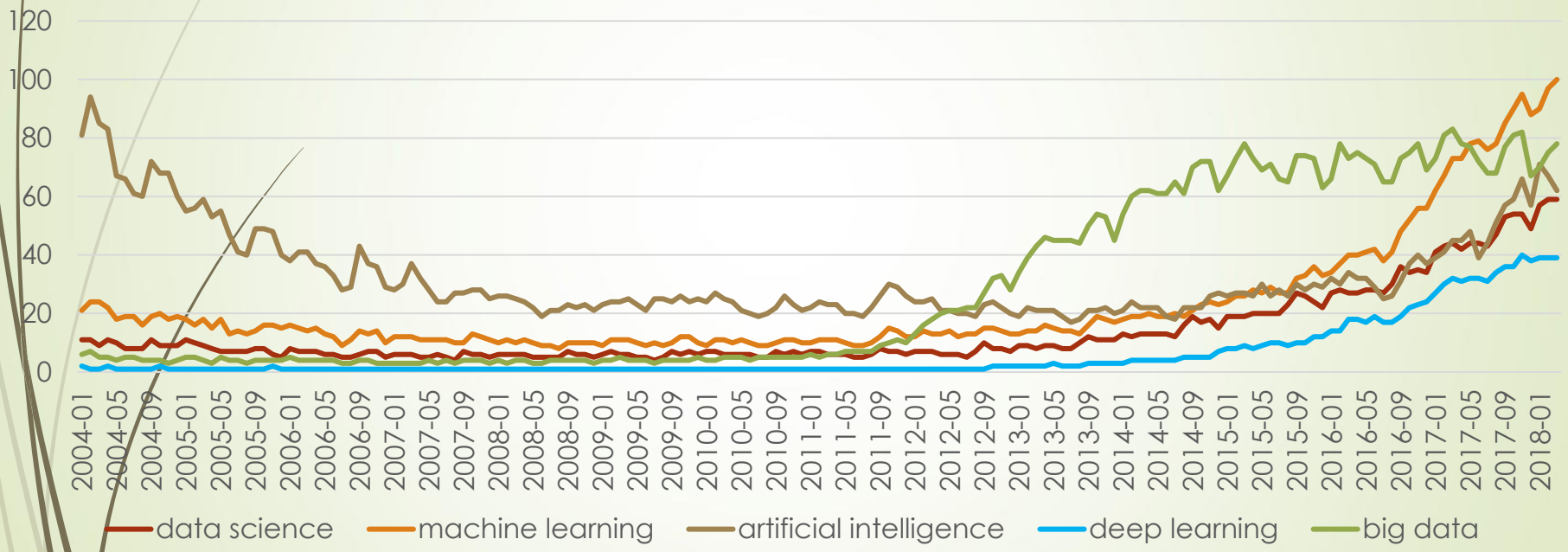


Nieprawidłowy podział na zbiór testujący i trenujący

Problem dobrej generalizacji sieci

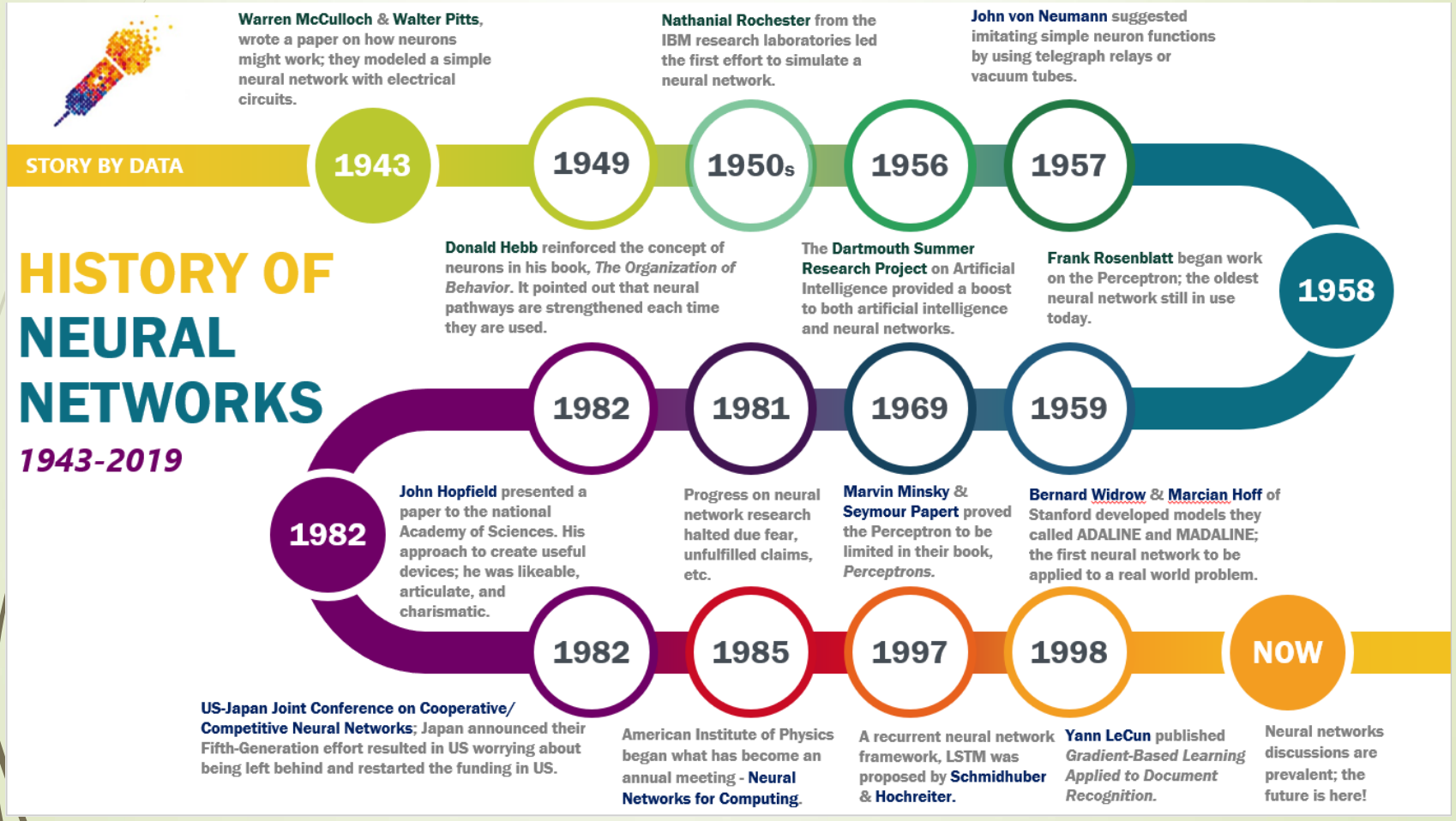


Machine Learning
Big Data
Data Science



Incidence of selected terms in **Google Search**

Zastosowania SSN w pierwszej fazie ich rozwoju



https://medium.com/analytics-vidhya/brief-history-of-neural-networks-44c2bf72eec

Google Trends

https://www.lucidarme.me/popular-neural-networks-years/

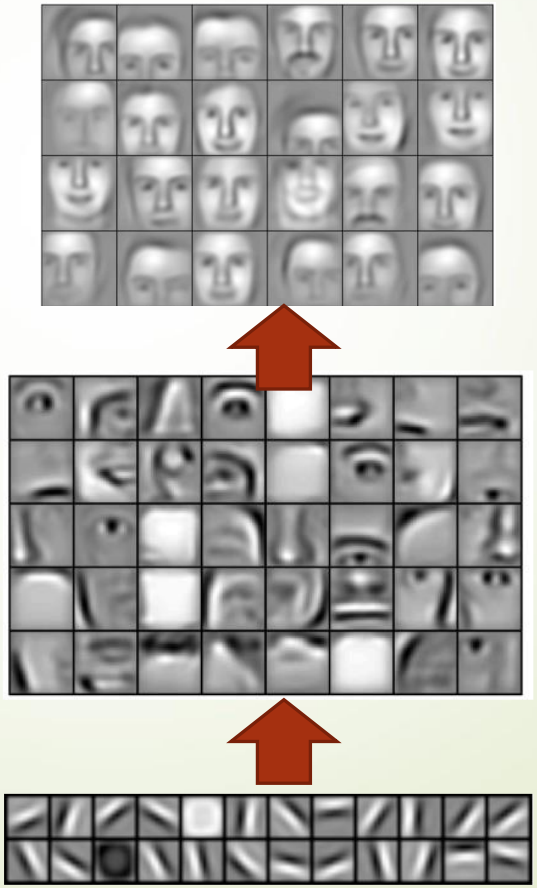
Dlaczego DL ???
Giganci IT rozwijają DL (ścigają się)

Deep Learning podstawowe cechy

- **rozwija strukturę hierarchiczną i reprezentację podstawowych i wtórnych cech**
- wykorzystują **kaskadę wielu warstw neuronów** (lub innych jednostek obliczeniowych) różnych rodzajów w celu **stopniowej ekstrakcji cech** i ich **transformacji** w celu osiągnięcia hierarchii cech wtórnych/pochodnych,
- strategie **uczenia nadzorowanego** i **nienadzorowanego** dla różnych warstw
- Iteracyjnie rozwija strukturę i parametry sieci

Reprezentacja podstawowych i wtórnych cech

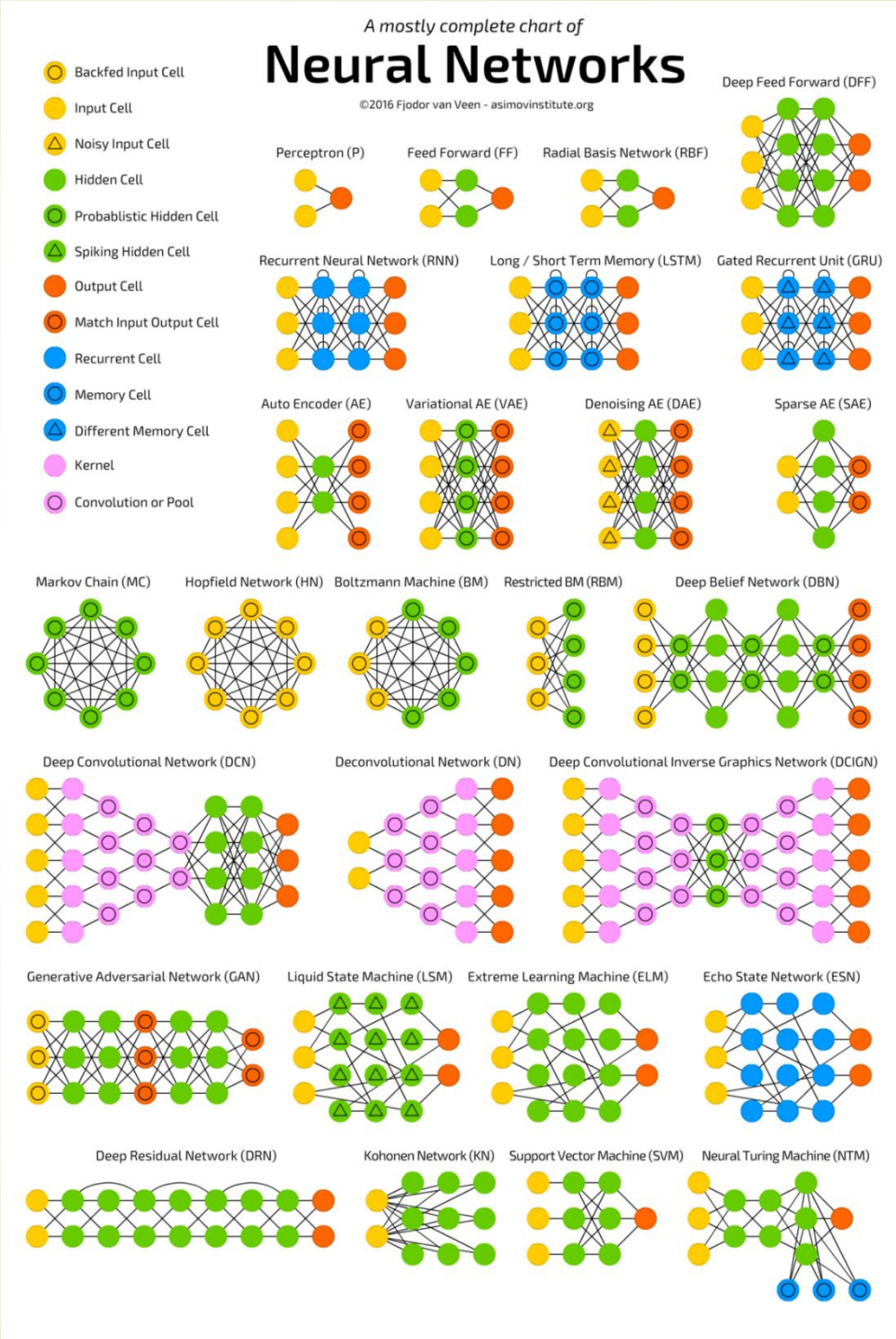
Kolejne warstwy modelu uczą się głębszych reprezentacji pośrednich



warstwa 2

warstwa 1

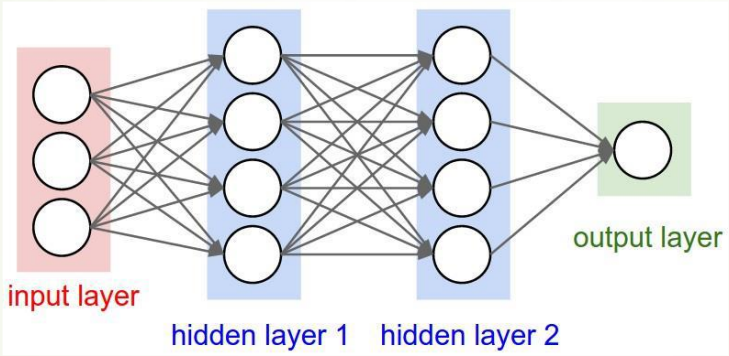
Struktury SSN z uwzględnienim DL



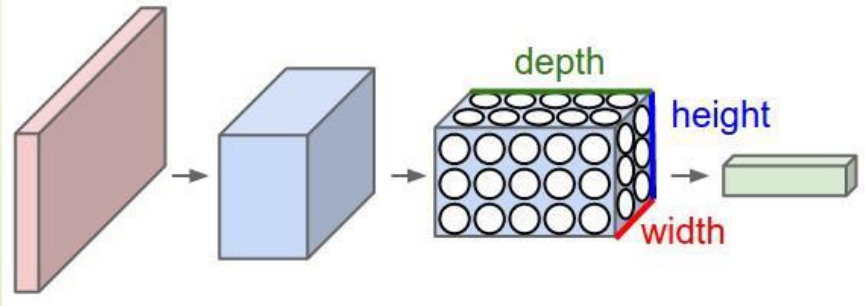
Convolutional Neural Networks (CNN)

- CNN są inspirowane biologiczną **strukturą kory wzrokowej**, gdzie odpowiedzi neuronów są ograniczone do pola receptywnego.
- CNN są stosowane w **rozpoznawaniu obrazów**, **animacji komputerowej**, **przetwarzaniu języka,** ale również do **analizy sygnałów**.
- Aktywacja warstw CNN opiera się na **działaniu splotu**.
- Rzadka reprezentacja, współdzielone wagi,

CNN - struktura



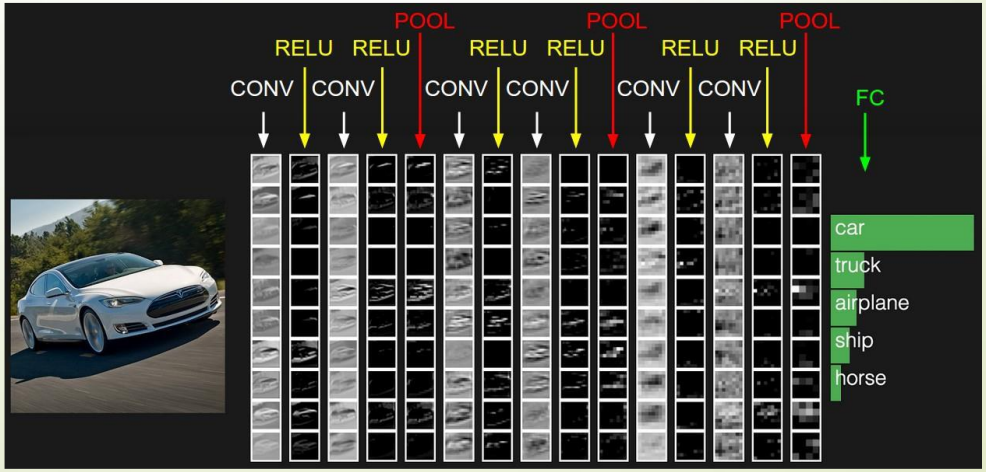
ANN



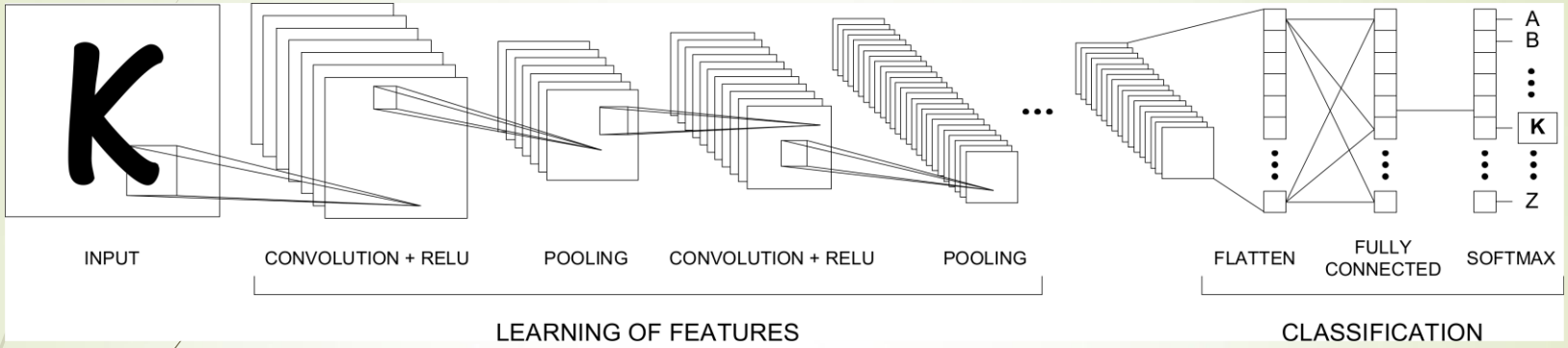
CNN

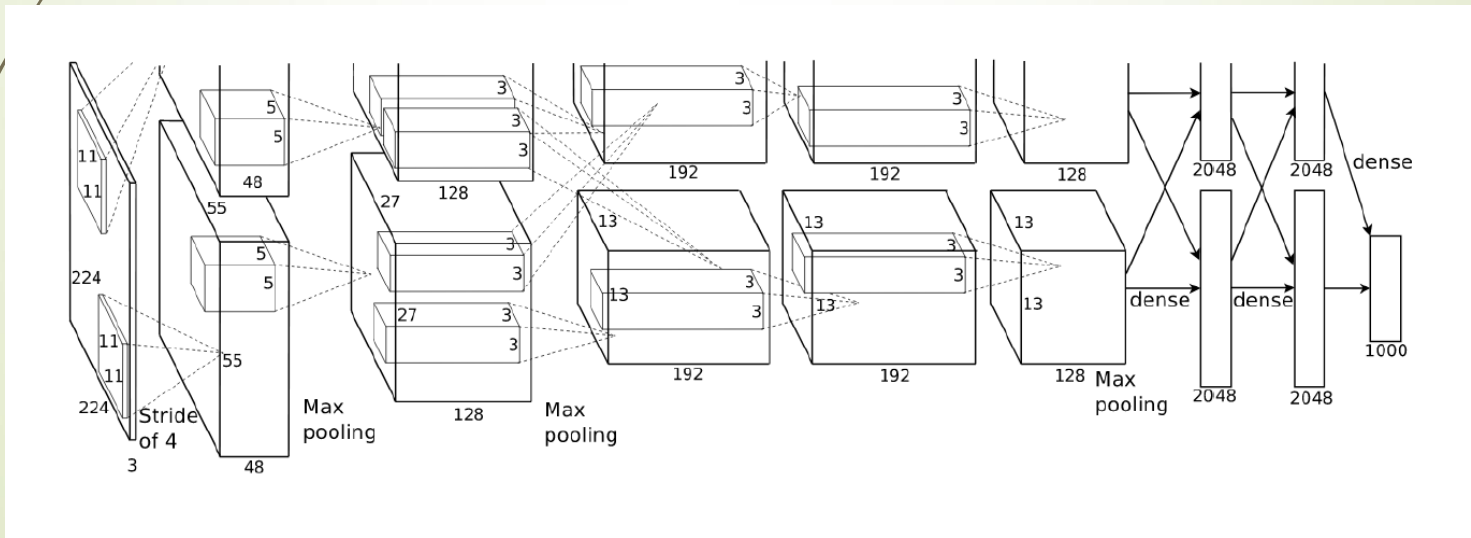
CNN – struktura - warstwy

- **WEJSCIE** [32x32x3] przechowuje surowe wartości pikseli obrazu, w tym przypadku szerokość obrazu 32, wysokość 32 oraz 3 kanały RGB
- **CONV** – obliczająca splot obrazu z filtrem
- **RELU** – funkcja aktywacji
- **POLL** – przekształcenie dla zmniejszenia ilości informacji
- **FC** – klasyczna sieć



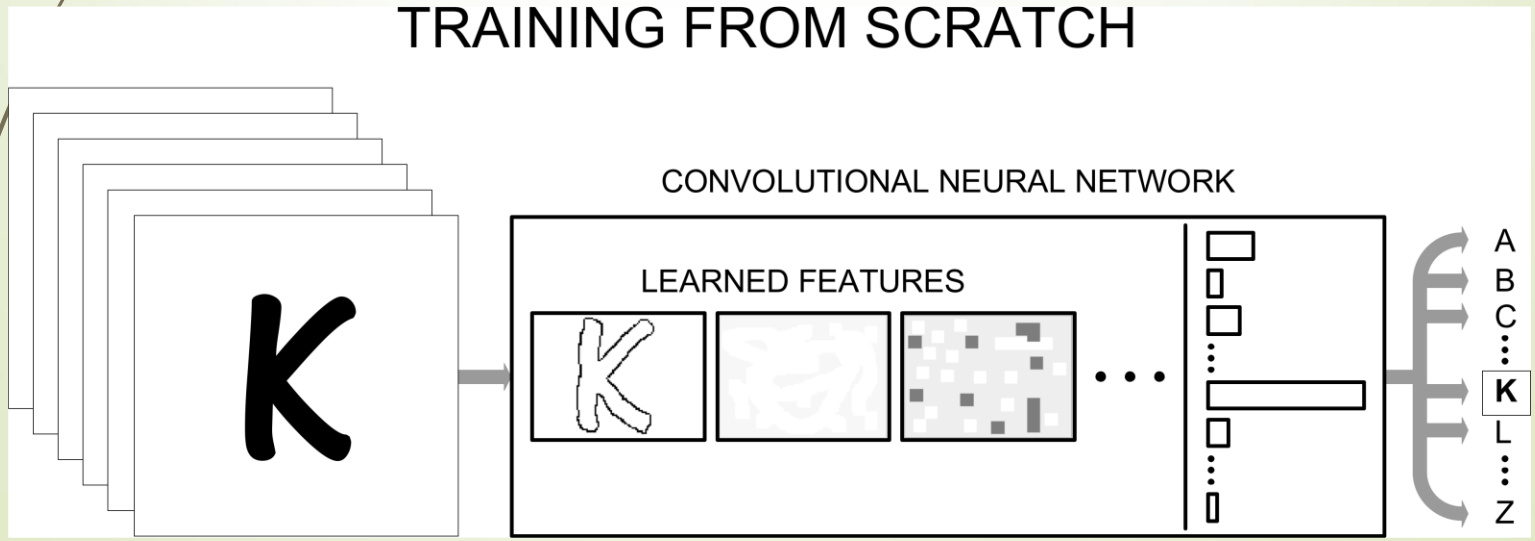
CNN – różne struktury





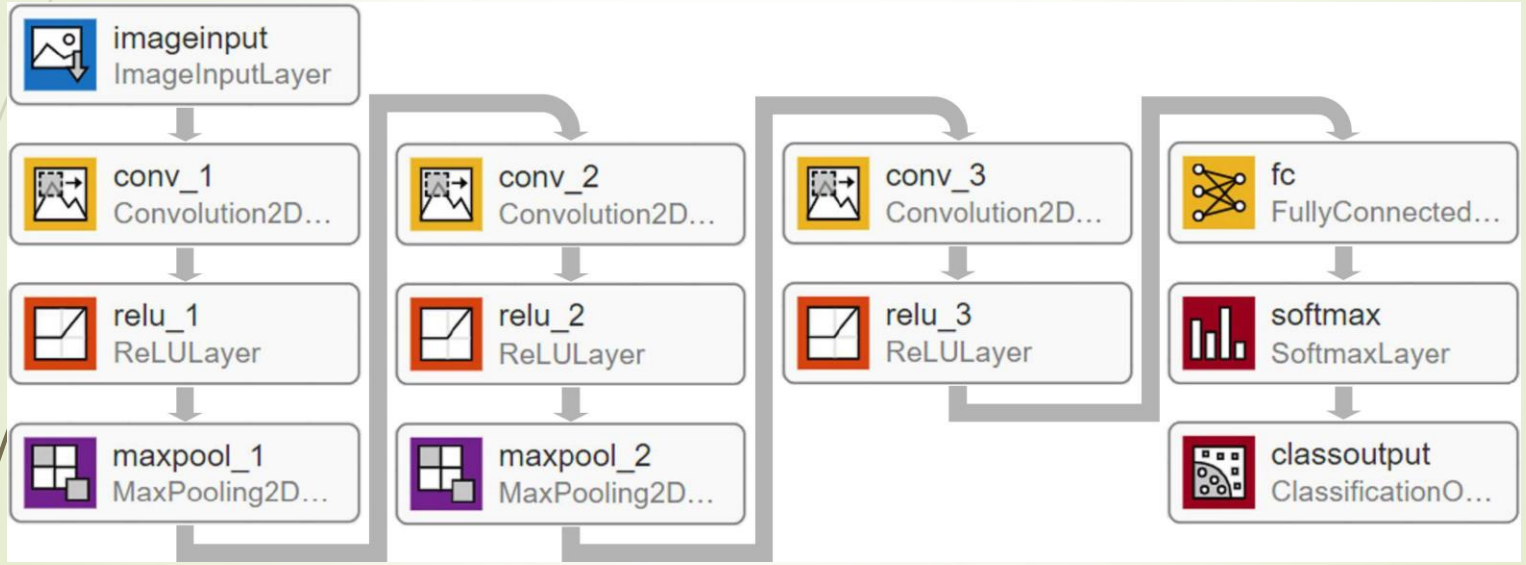
Pierwsze podejście - zaprojektowane CNN

- Możliwe jest stworzenie spersonalizowanej CNN.
- Wymaga to pewnej wiedzy na temat przetwarzania obrazu (filtry, padding), sieci neuronowych (parametry treningowe, funkcje aktywacji, kodowanie wyjściowe) i oszacowania złożoności (głębokość).



Pierwsze podejście - zaprojektowane CNN

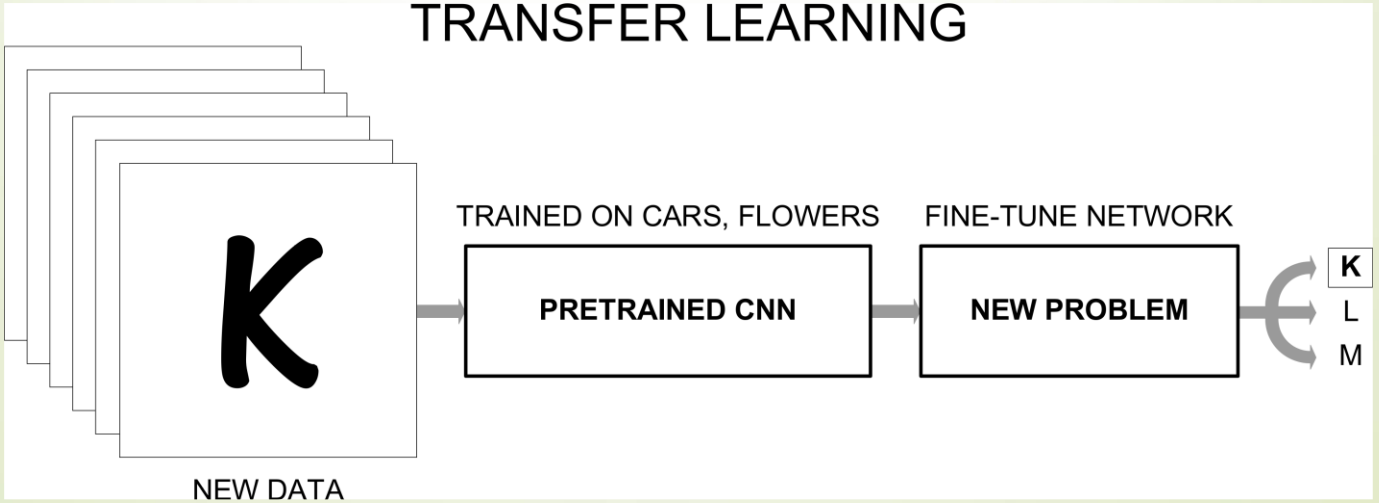
- Uproszczona CNN używana do klasyfikacji obrazów:



Source: Deep Network Designer: built-in environment of Matlab's Deep Learning Toolbox

Drugie podejście - wstępnie wyszkolona CNN (P – CNN)

- Możliwe jest wykorzystanie rozwiniętej i wyszkolonej CNN.
- Wymaga to precyzyjnego dostrojenia sieci: dostosowania głębszych warstw w P – CNN poprzez szkolenie nowego zestawu danych.



Drugie podejście - wstępnie wyszkolona CNN (P – CNN)

- P – CNN nauczyły się **wyodrębniać funkcje informacyjne z naturalnych obrazów** - można to wykorzystać jako punkt wyjścia do nauki nowego zadania czyli rozwiniętej i wyszkolonej CNN.
- P – CNN są szkolone na ponad milionie obrazów i mogą klasyfikować obrazy na 1000 kategorii obiektów (np .: klawiatura, kubek kawy, ołówek i wiele zwierząt).
- Obrazy szkoleniowe są podzbiorem **ImageNet database**, używanej w **ImageNet Large-Scale Visual Recognition Challenge** (ILSVRC).
- Użycie P – CNN jest zazwyczaj dużo szybsze i łatwiejsze niż szkolenie sieci od podstaw.

Drugie podejście - wstępnie wyszkolona CNN (P – CNN)

Network	Depth[1]	Size (MB)	Operations per prediction (billions)	Parameters (millions)	Input size (image)
Alexnet	8	245	0.72	61	227×227
vgg16	16	554	15.5	138	224×224
vgg19	19	575	19.6	144	224×224
Squeezenet	18	6.4	0.39	1.23	227×227
Googlenet	22	29	1.58	7	224×224
inceptionv3	48	96	5.7	23.9	299×299
resnet18	18	47	1.81	11.7	224×224
resnet50	50	104	3.8	25.6	224×224
resnet101	101	180	7.6	45	224×224
inceptionresnetv2	164	225	13.2	56	299×299

Głębokość jest definiowana jako największa liczba kolejnych warstw splotowych oraz w pełni połączonych na ścieżce od warstwy wejściowej do warstwy wyjściowej.

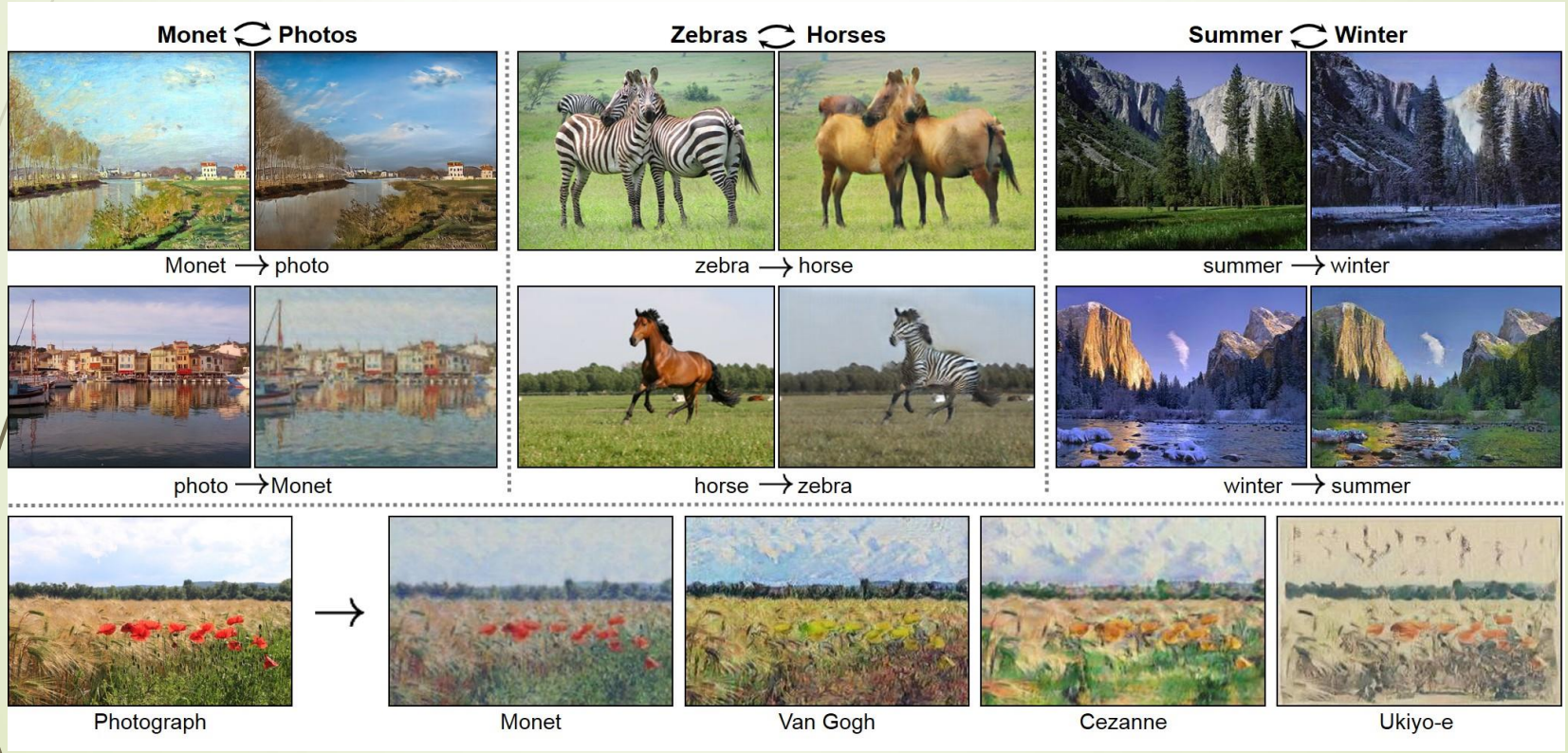
Przykłady

Unpaired Image-to-Image Translation using **Cycle-Consistent Adversarial Networks**

Jun-Yan Zhu Taesung Park Phillip Isola Alexei A. Efros

UC Berkeley

In ICCV 2017

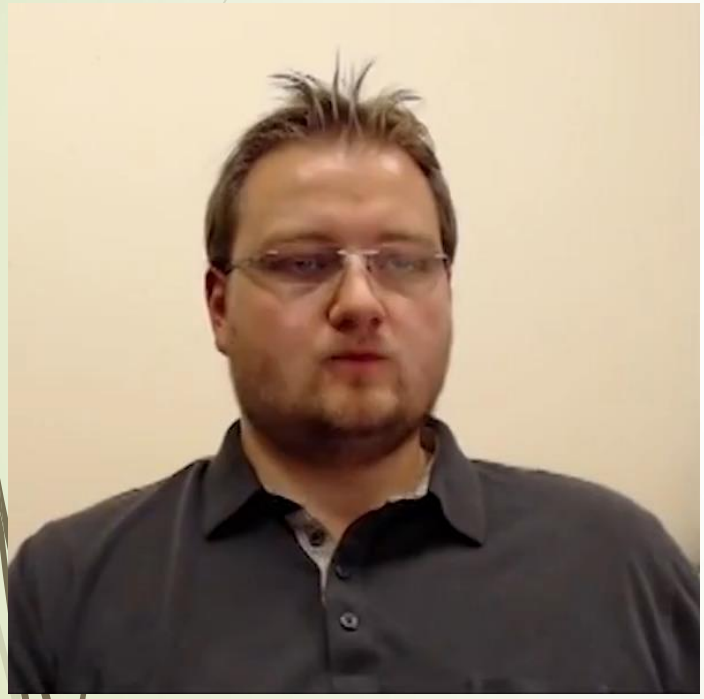


Przykłady

GAN

Przykłady

DeepFake



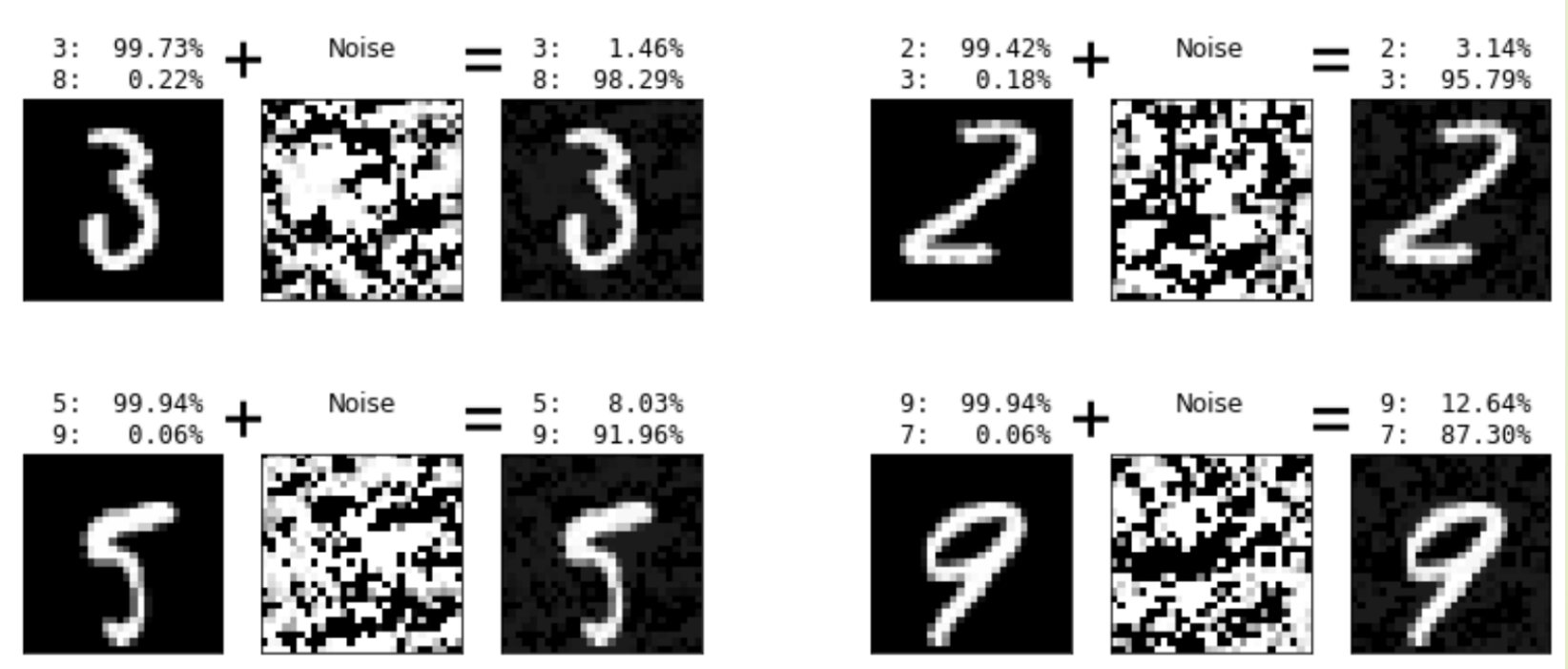
Przykłady

DeepFake

Przykłady

Adversarial Attacks and Defences for Convolutional Neural Networks

Ataki przeciwstawne

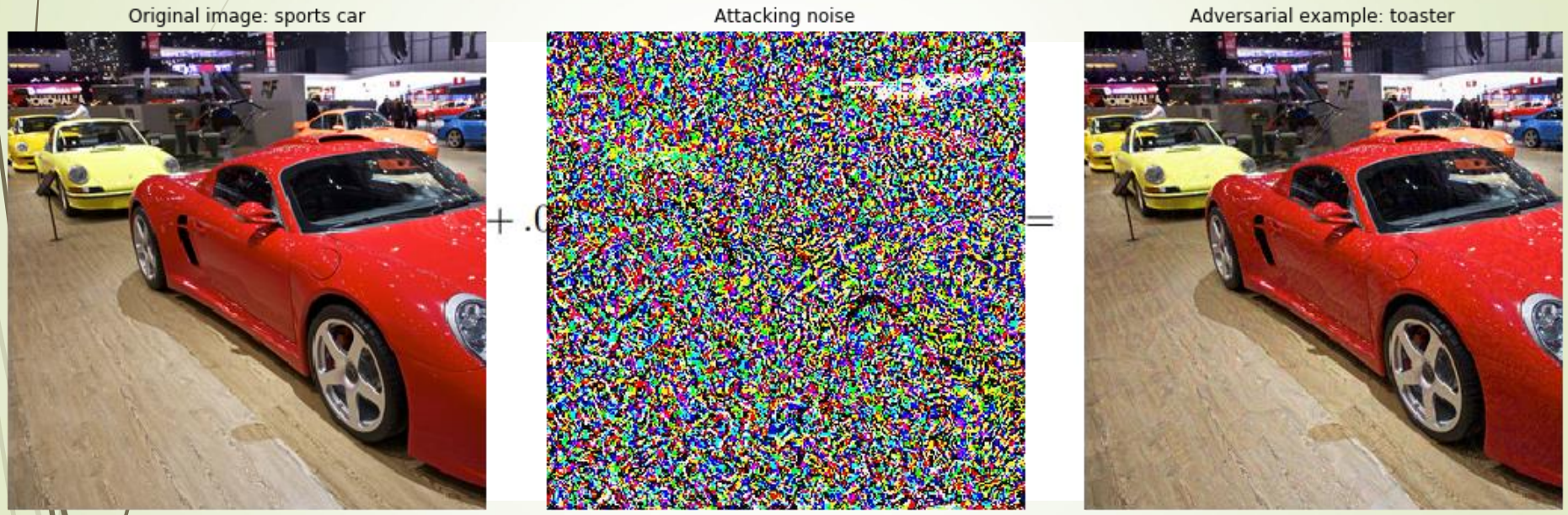


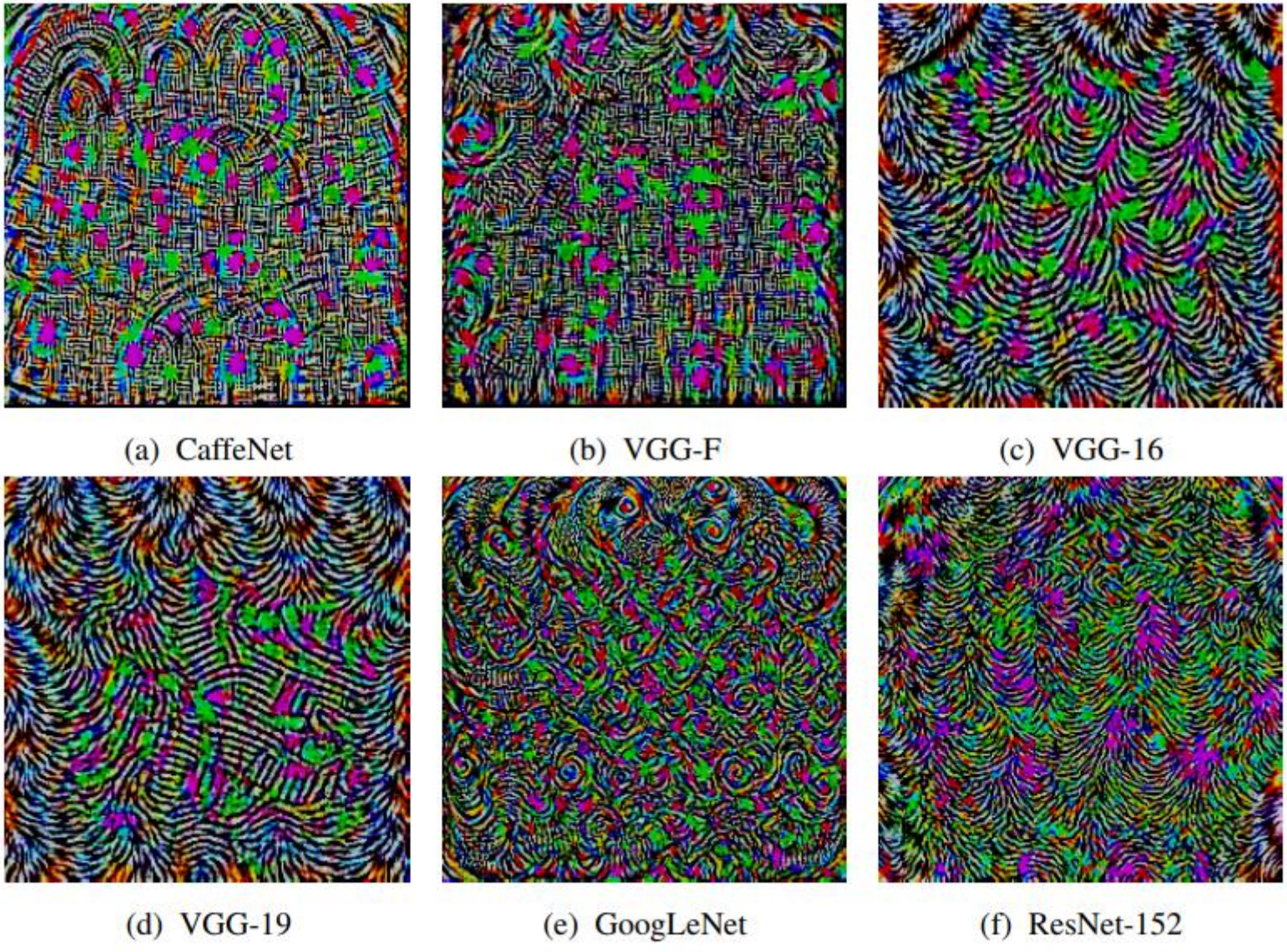
Images on the **left** of each set are original images, **middle** images are amplified depictions of the noise, and images on the **right** are crafted adversarial images (original+noise). Classification probabilities are showed for each image, on a simple conv net which gets greater than **99.25% accuracy on MNIST**. The maximum noise limit here is 0.05.

Przykłady

Adversarial Attacks and Defences for Convolutional Neural Networks

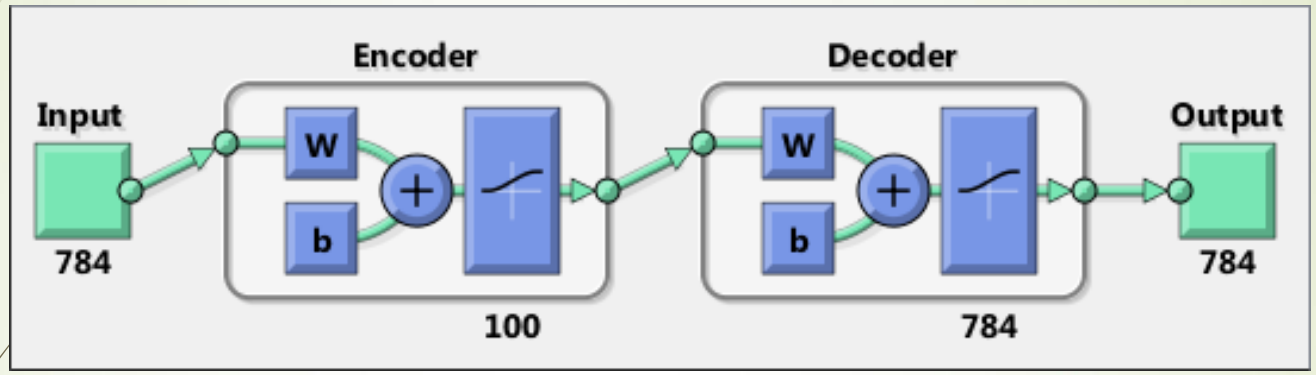
Ataki przeciwstawne (czy panda + nicienie = gibon ???)



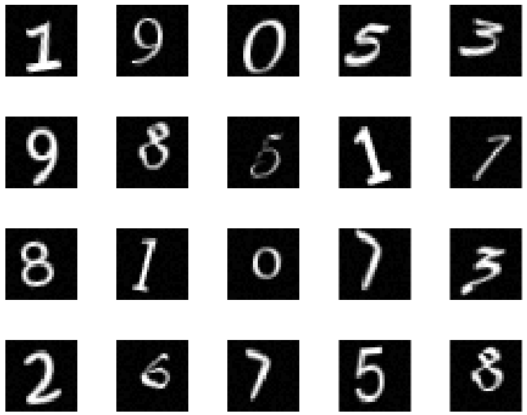


Universal perturbations computed for different deep neural network architectures.

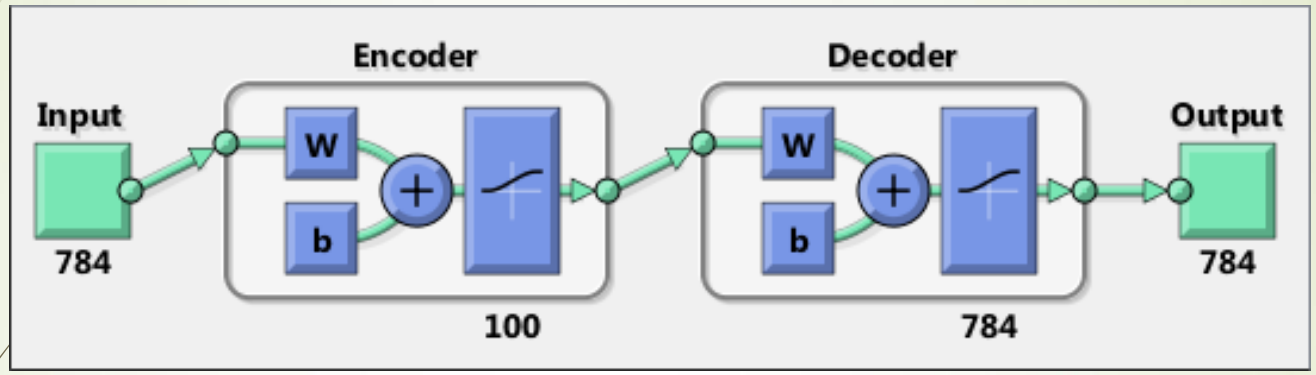
Autoenkoder – etap 1




```
% Load the training data into memory
[xTrainImages, tTrain] = digittrain_dataset;
% Display some of the training images
clf
for i = 1:20
subplot(4,5,i);
imshow(xTrainImages{i});
end
```

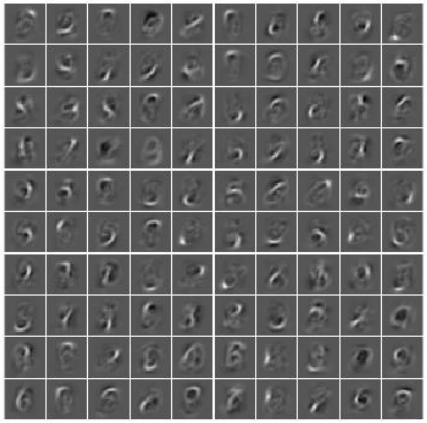


Autoenkoder – etap 1

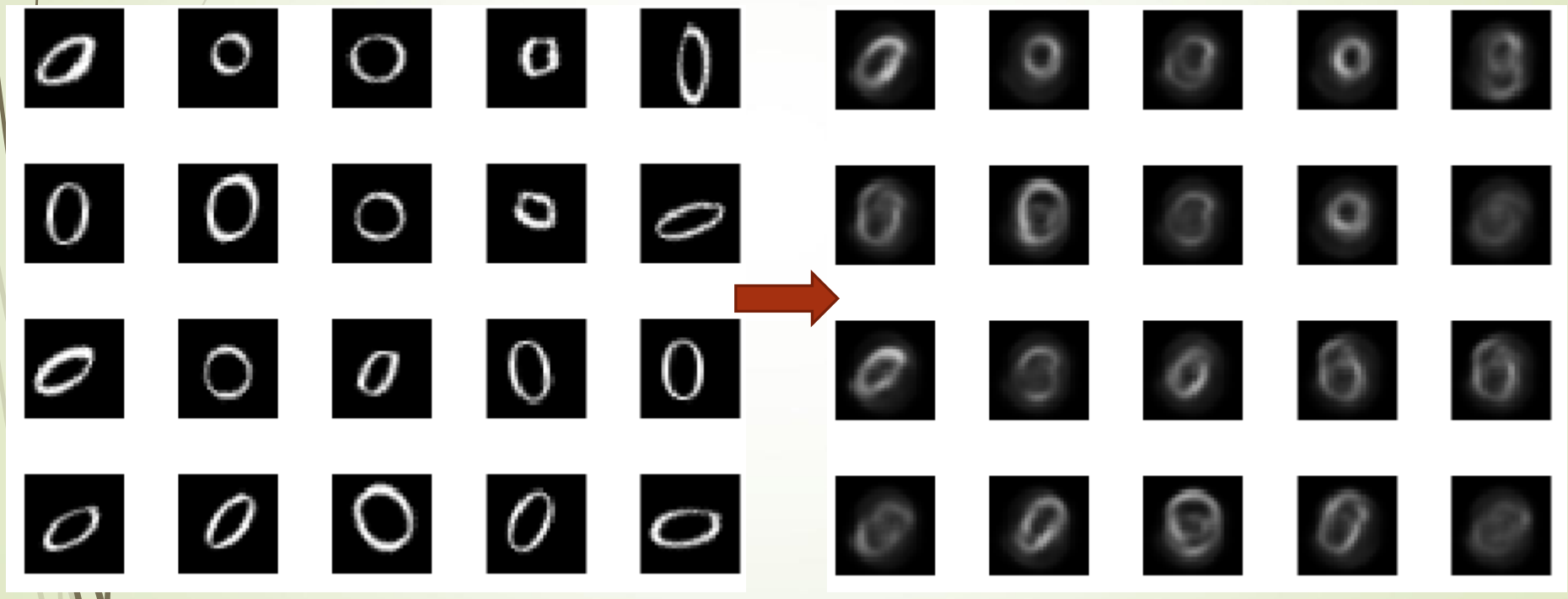



```
rng('default') % set the random number generator seed
hiddenSize1 = 100; % set the number of hidden nodes in Layer 1
autoenc1 = trainAutoencoder(xTrainImages,hiddenSize1, ...
        'MaxEpochs',400, ...
        'L2WeightRegularization',0.004, ...
        'SparsityRegularization',4, ...
        'SparsityProportion',0.15, ...
        'ScaleData', false);
plotWeights(autoenc1);
```

wagi

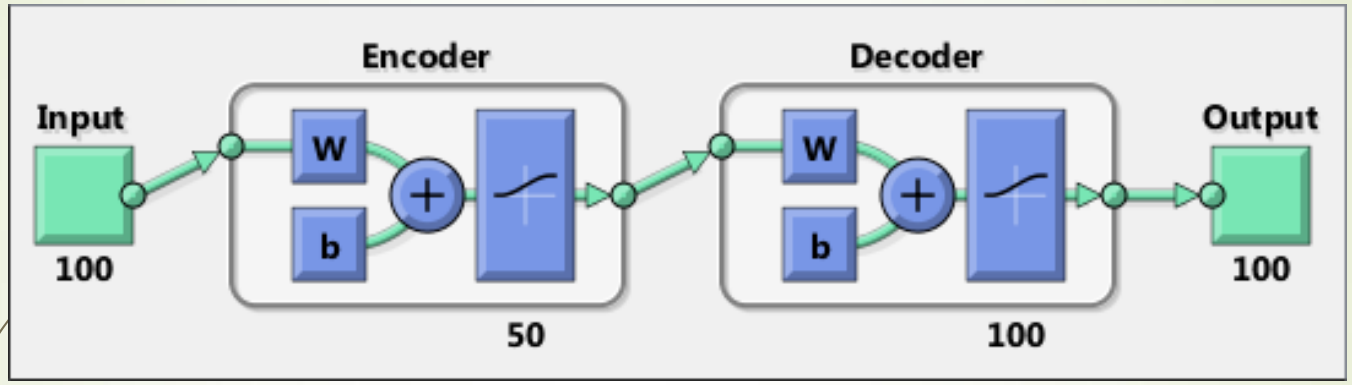


Zadanie autoencodera



xReconstructed = predict(**autoenc1**, XTest_change);

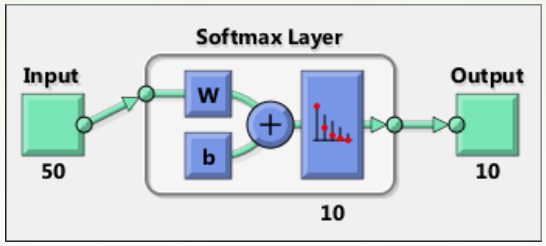
Autoenkoder – etap 2 rozbudowa




```
feat1 = encode(autoenc1,xTrainImages);
hiddenSize2 = 50; % set the number of hidden nodes in Layer 2

autoenc2 = trainAutoencoder(feat1,hiddenSize2, ...
        'MaxEpochs',100, ...
        'L2WeightRegularization',0.002, ...
        'SparsityRegularization',4, ...
        'SparsityProportion',0.1, ...
        'ScaleData', false);
```

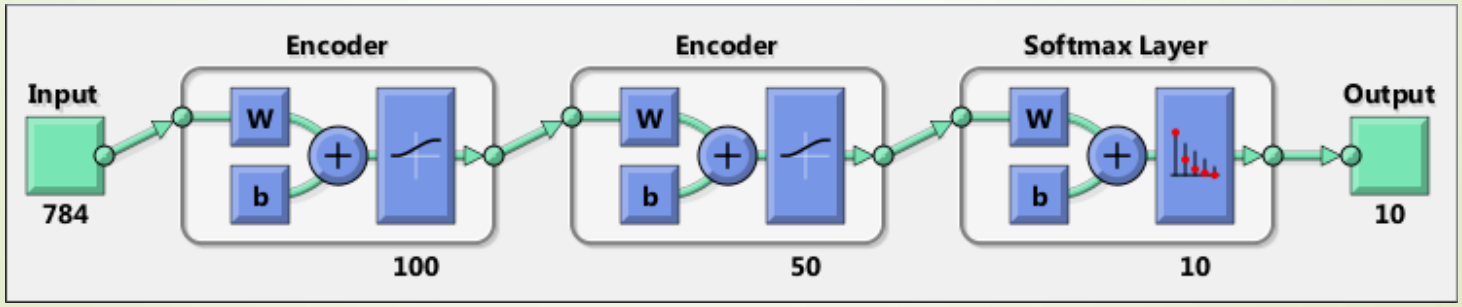
Autoenkoder - rozbudowa



feat2 = encode(autoenc2,feat1);

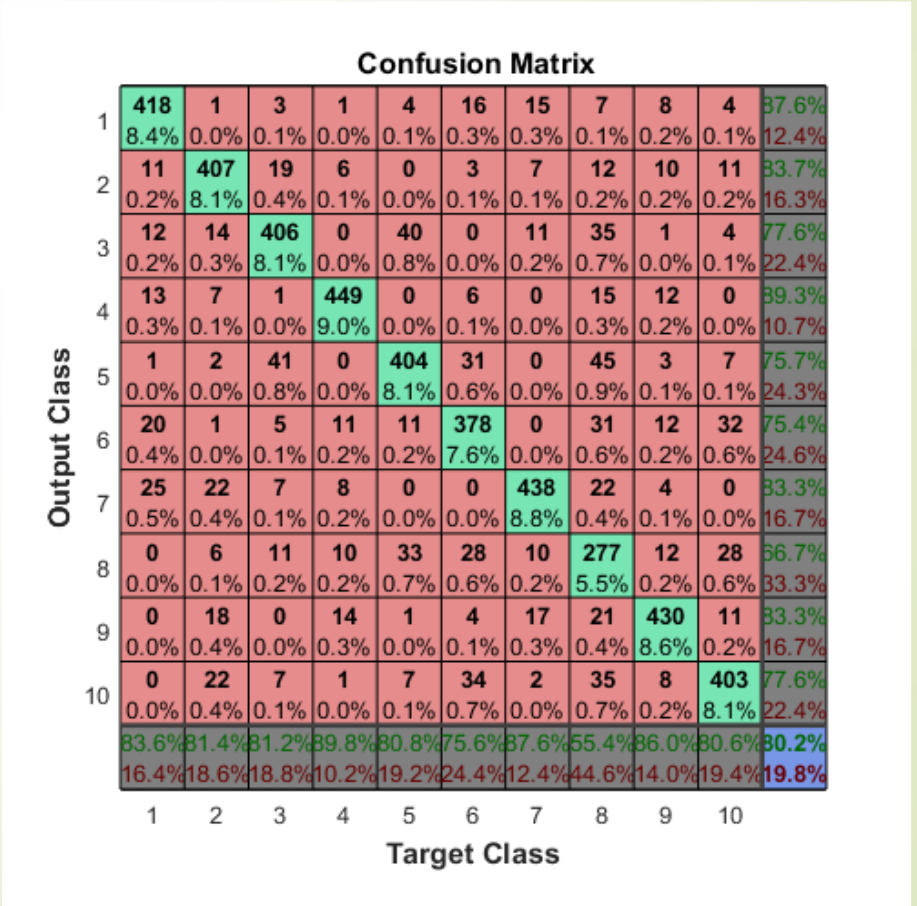
softnet = trainSoftmaxLayer(feat2,tTrain,'MaxEpochs',400);

deepnet = **stack**(autoenc1,autoenc2,softnet); % stack all layers
view(**deepnet**)



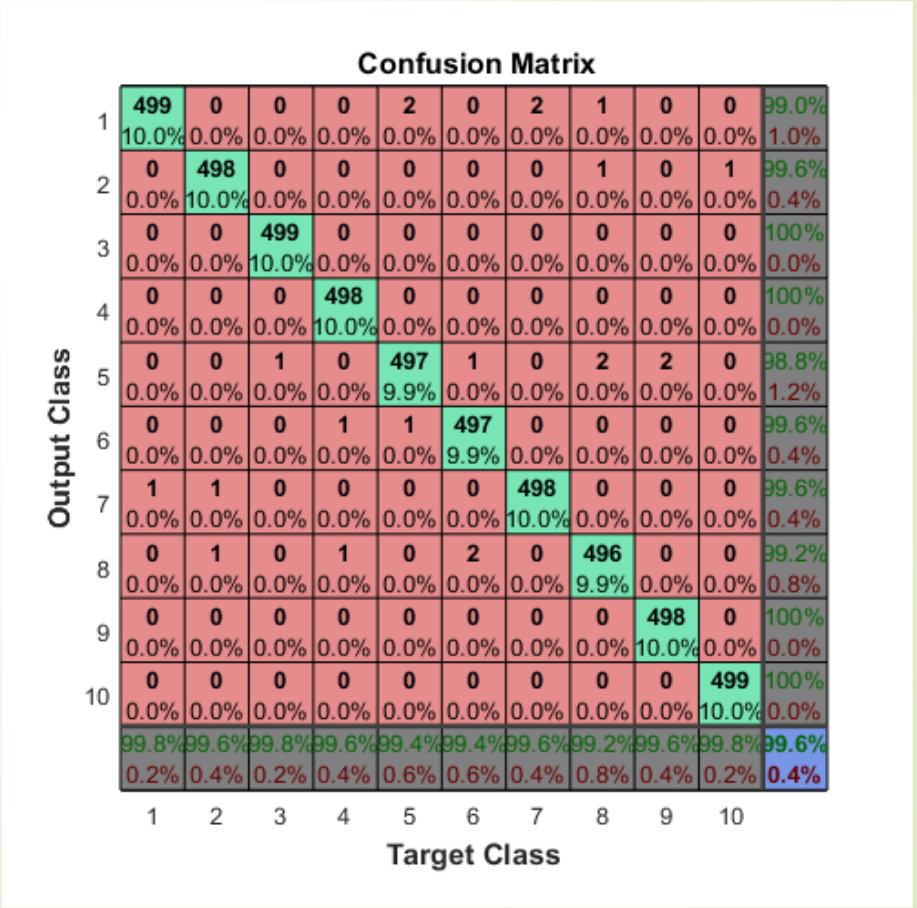
Klasyfikacja kodera, przy uczeniu cząstkowym

```
% Load the test images
[xTestImages, tTest] = digittest_dataset;
y = deepnet(xTest);
plotconfusion(tTest,y);
```

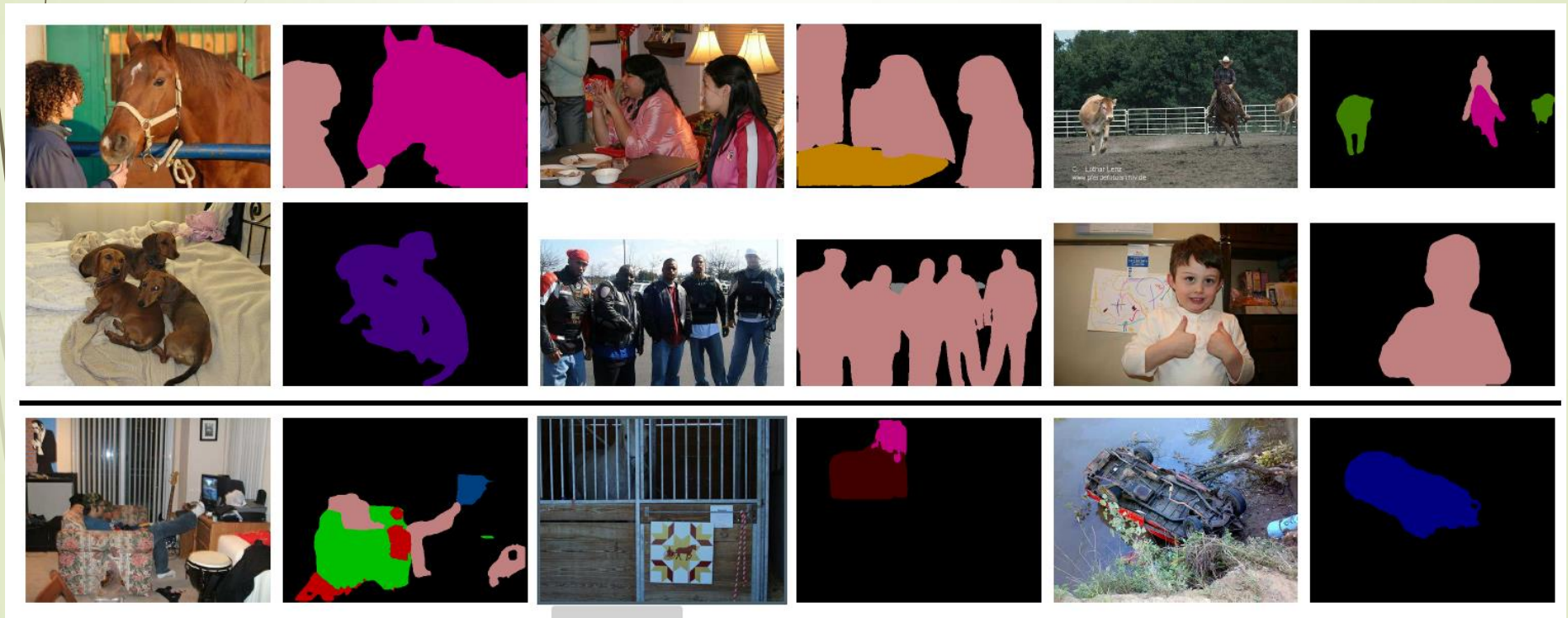


Klasyfikacja kodera, przy uczeniu całościowym

```
% Perform fine tuning
deepnet = train(deepnet,xTrain,tTrain);
y = deepnet(xTest);
plotconfusion(tTest,y);
```



Zastosowanie Encoder-Decoder wraz z CNN do semantycznej segmentacji obrazu



DeepLabv3 + ResNet-101 as Network Backbone

The last row shows a **failure** mode.


Bicyclist
Pedestrian
Car
Fence
SignSymbol
Tree
Pavement
Road
Pole
Building
Sky

Pamięć asocjacyjna

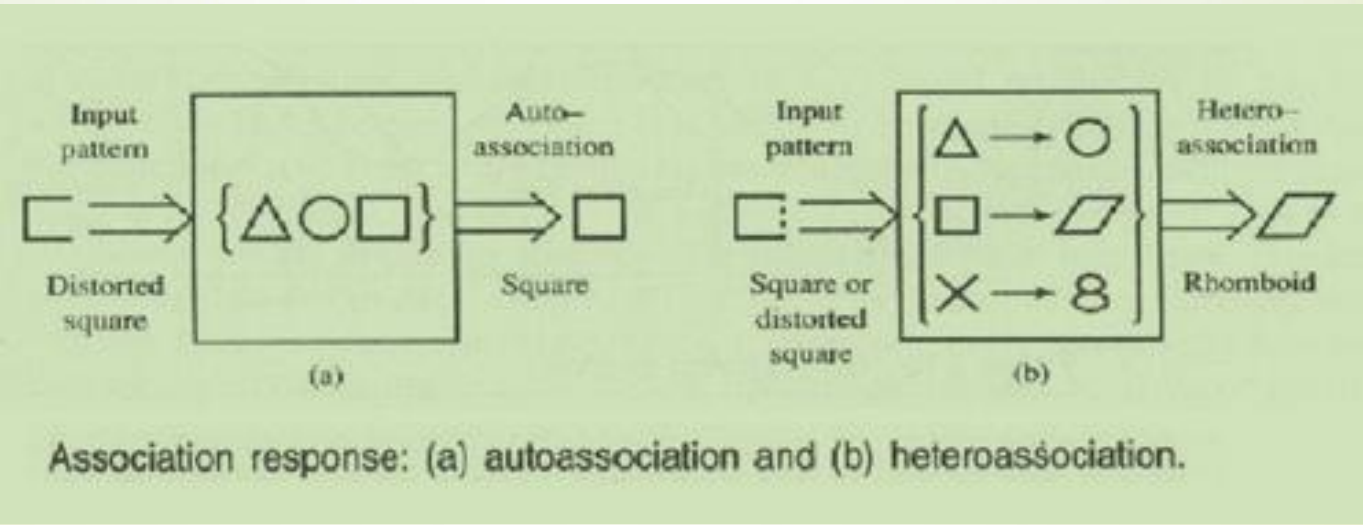
Dwa typy pamięci asocjacyjnej

- **Pamięć autoasocjacyjna**

  Wzorzec $\mathbf{X_i}$: $\mathbf{X_i + \Delta \rightarrow X_i}$

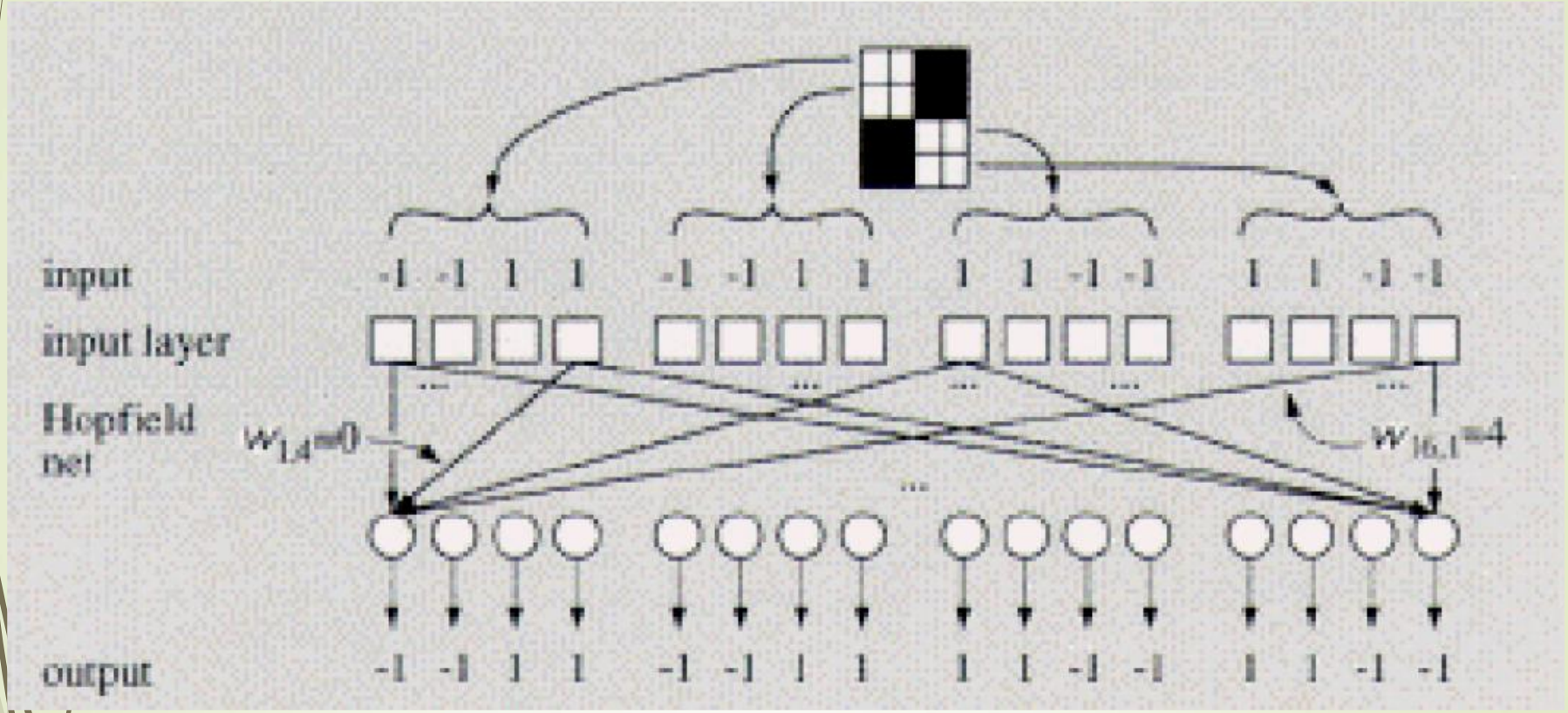
- **Pamięć heteroasocjacyjna**

  Wzorzec $\mathbf{X_i \rightarrow Y_i : X_i + \Delta \rightarrow Y_i}$



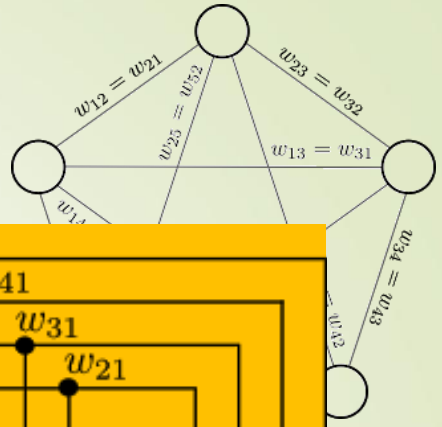
Association response: (a) autoassociation and (b) heteroassociation.

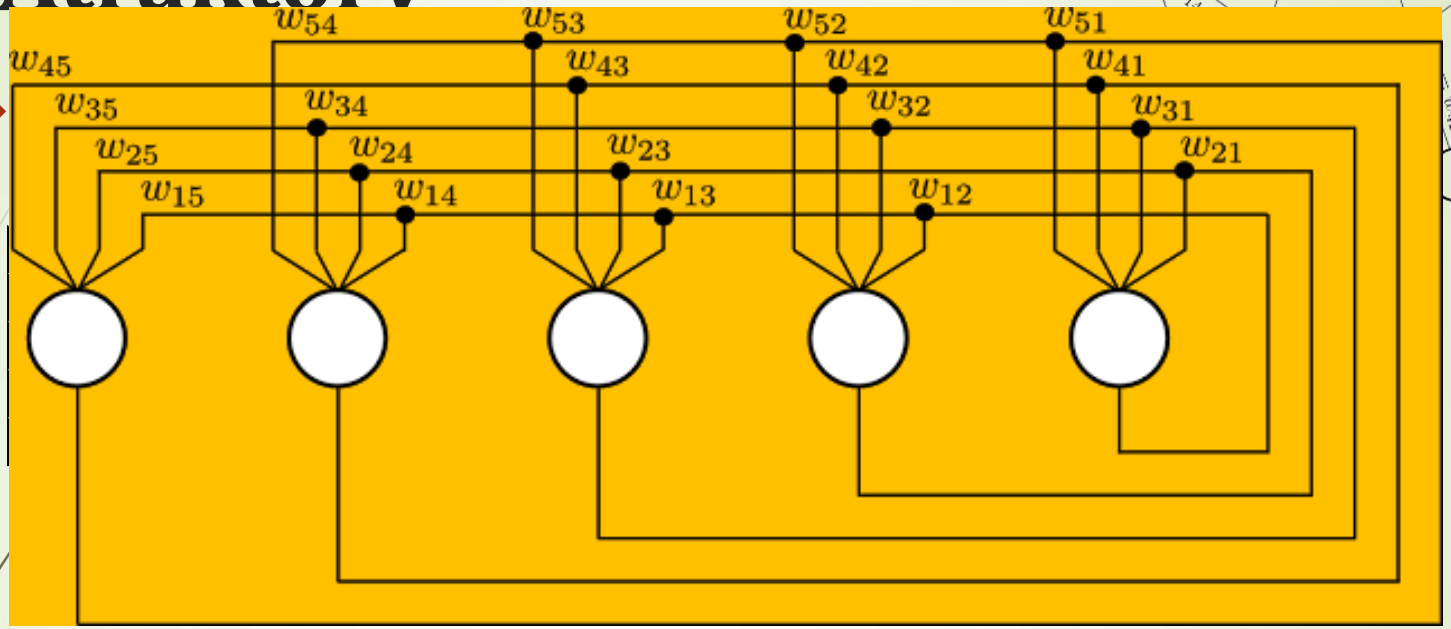
Sieci Hopfielda - uczenie

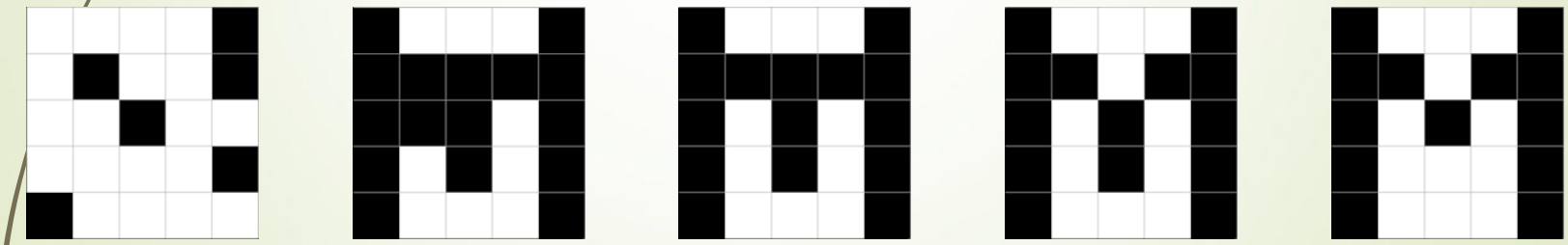


WEJŚCIE = WYJŚCIU !!!

Atraktory







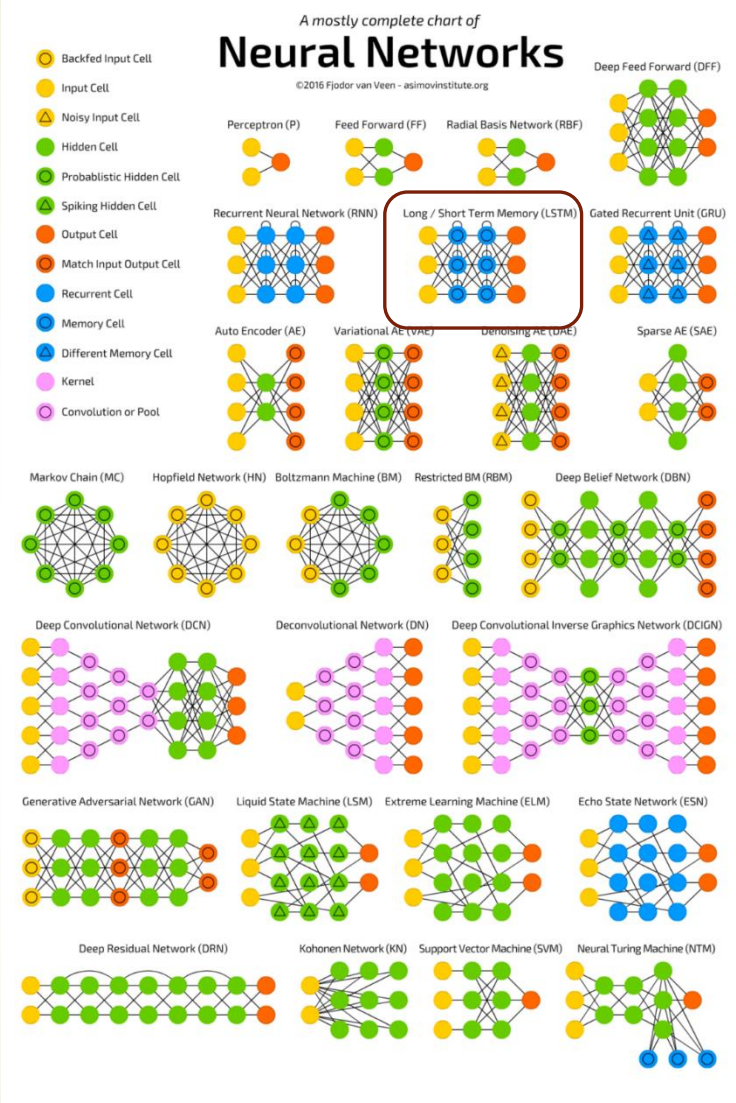
wejście

czas (iteracje SSN)

http://gorayni.blogspot.com/2013/08/hopfield-networks_6.html

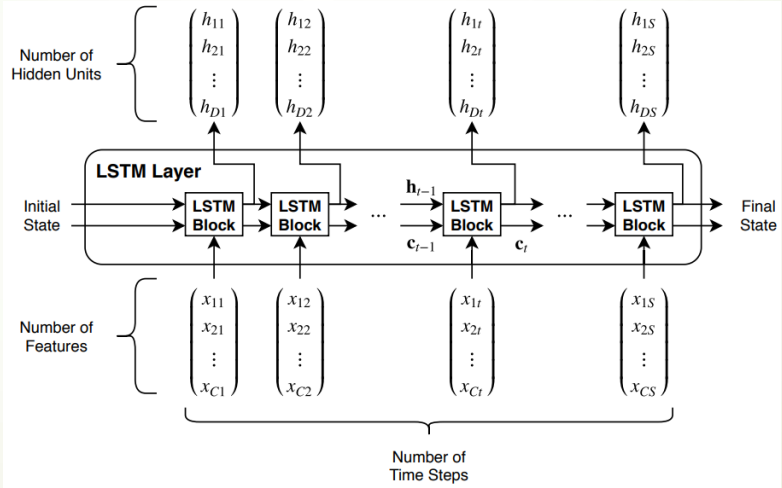
Long Short-Term Memory Networks (LSTM)

Sieć LSTM jest rodzajem rekurencyjnej sieci neuronowej (RNN), która może nauczyć się **długoterminowych zależności między krokami czasowymi danych sekwencji**.



Architektura sieci LSTM

Pierwszy blok LSTM wykorzystuje **stan początkowy sieci** i **pierwszy krok sekwencji** do obliczenia pierwszego wyjścia i zaktualizowanego stanu komórki. W kroku t czas wykorzystuje bieżący stan sieci (c_{t-1}, h_{t-1}) i następny krok sekwencji, aby **obliczyć wyjście i zaktualizowany stan komórki c_t**.



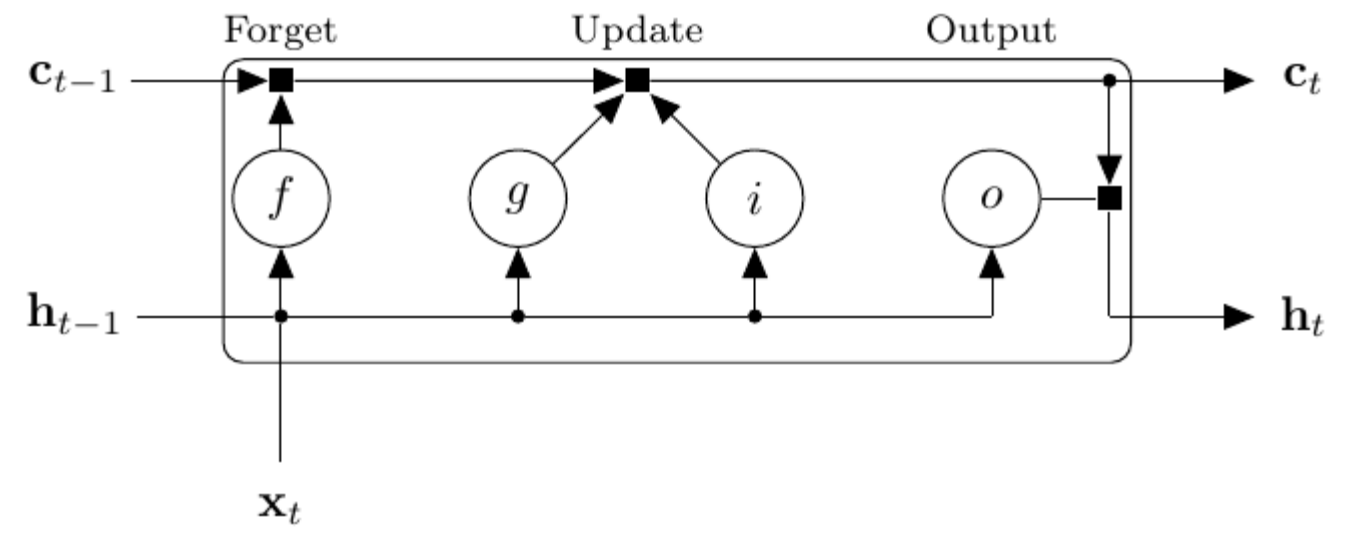
Stan warstwy składa się ze **stanu ukrytego** (znanego również jako stan wyjściowy) **i stanu komórki (c_t)**.

W każdym kroku warstwa dodaje informacje do lub usuwa informacje ze stanu komórki. Warstwa kontroluje te aktualizacje za pomocą **bramek**.

Architektura sieci LSTM

Składniki kontrolują stan komórki i stan ukryty warstwy.

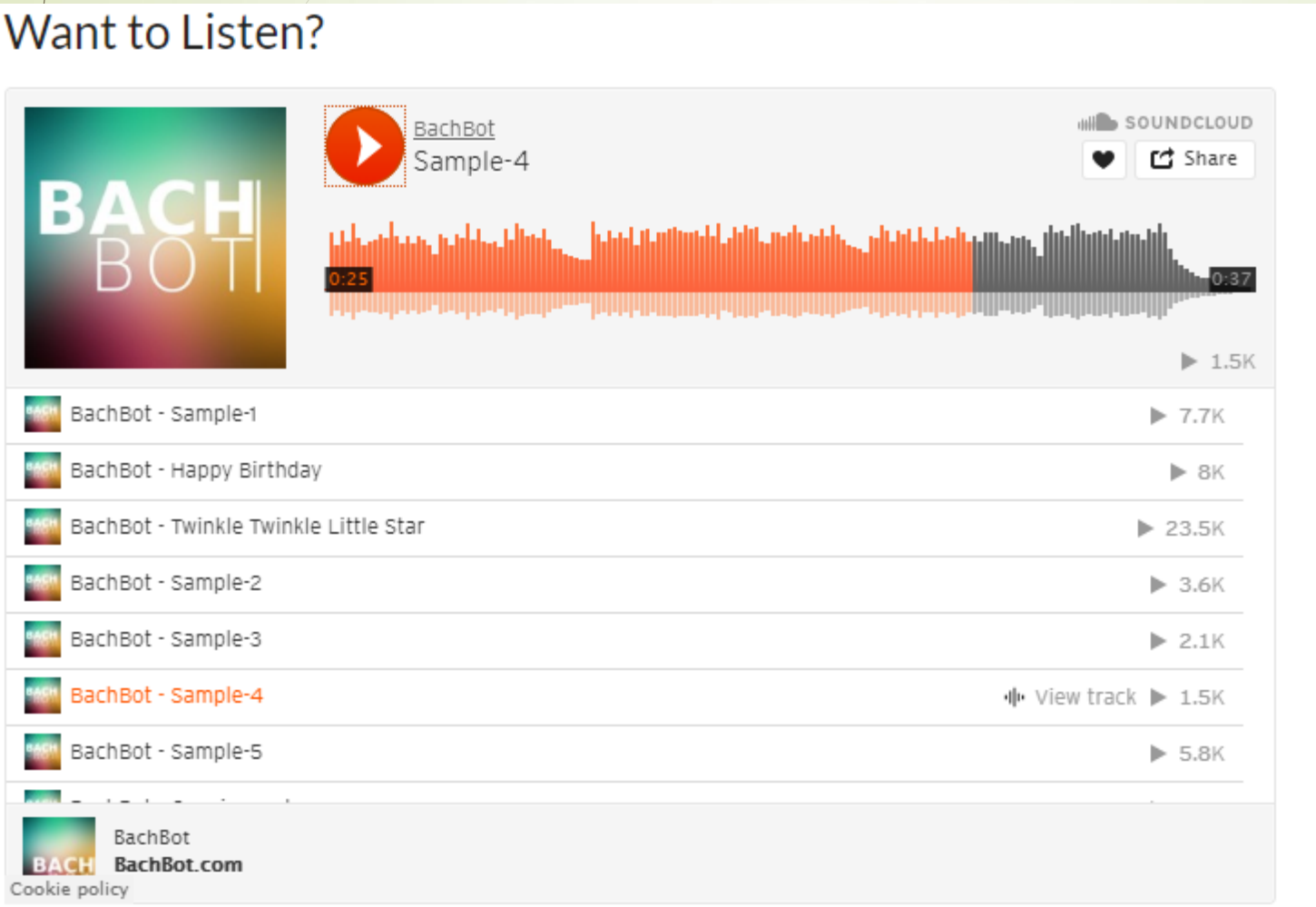
Komponent	**Zadanie**
Bramka wejściowa (i)	Kontrola poziomu aktualizacji stanu komórki (input gate)
Bramka zapomnienia (f)	Kontrola poziomu resetowania stanu komórki (forget gate)
Kandydat na komórkę (g)	Dodaj informacje do stanu komórki (cell candidate)
Brama wyjściowa (o)	Poziom kontroli stanu komórki dodany do stanu ukrytego (output gate)



Przepływ danych dla kroku czasu t.

The BachBot Challenge

Want to Listen?



Japanese Vowels Test LSTM Network

Load the Japanese Vowels **test data**.

XTest is a cell array containing
370 sequences of dimension 12 of varying length.

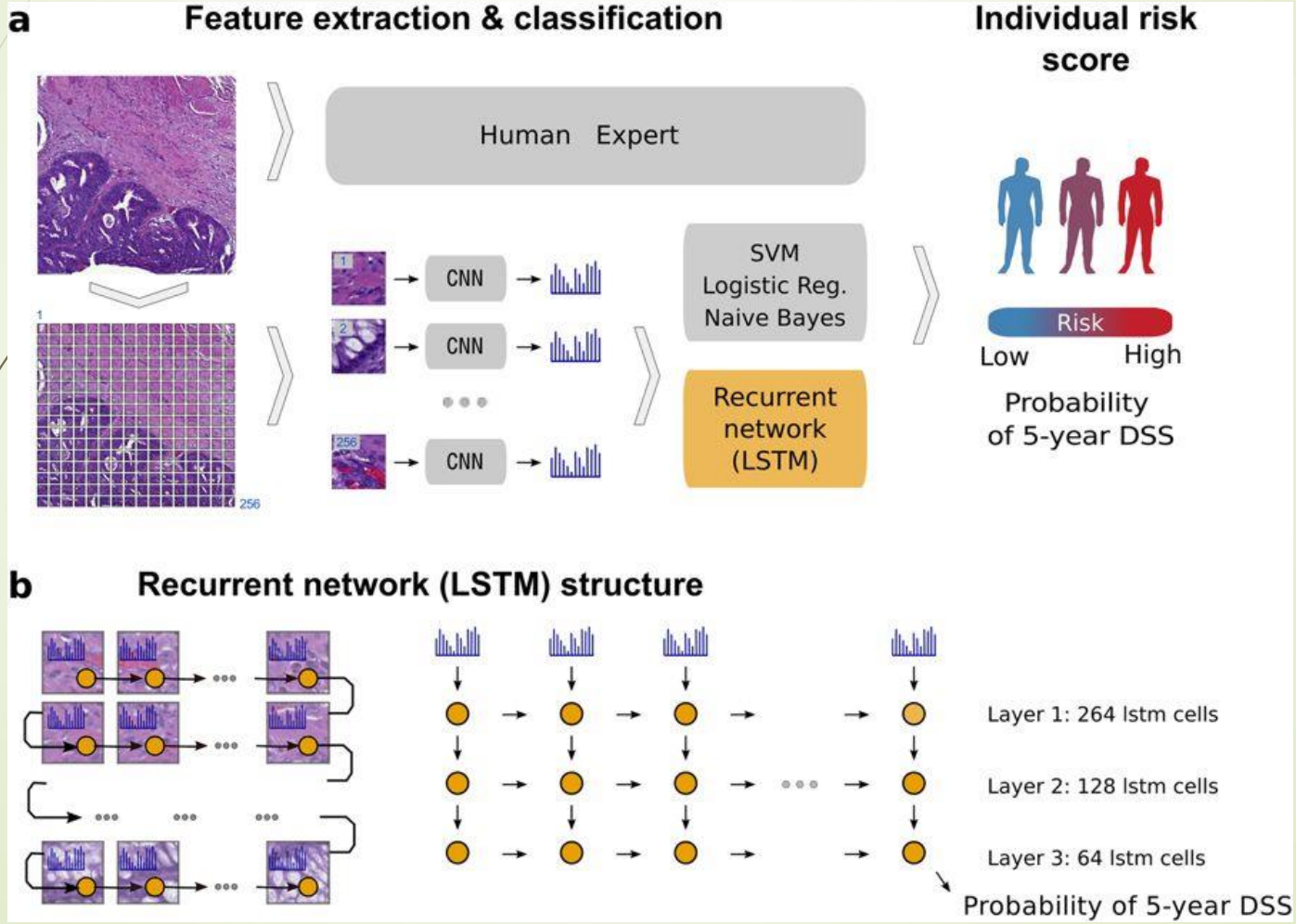
YTest is a categorical vector of labels "1","2",..."9", which correspond to the nine speakers.

```
miniBatchSize = 27;
YPred = classify(net,XTest, ...
    'MiniBatchSize',miniBatchSize, ...
    'SequenceLength','longest');
```

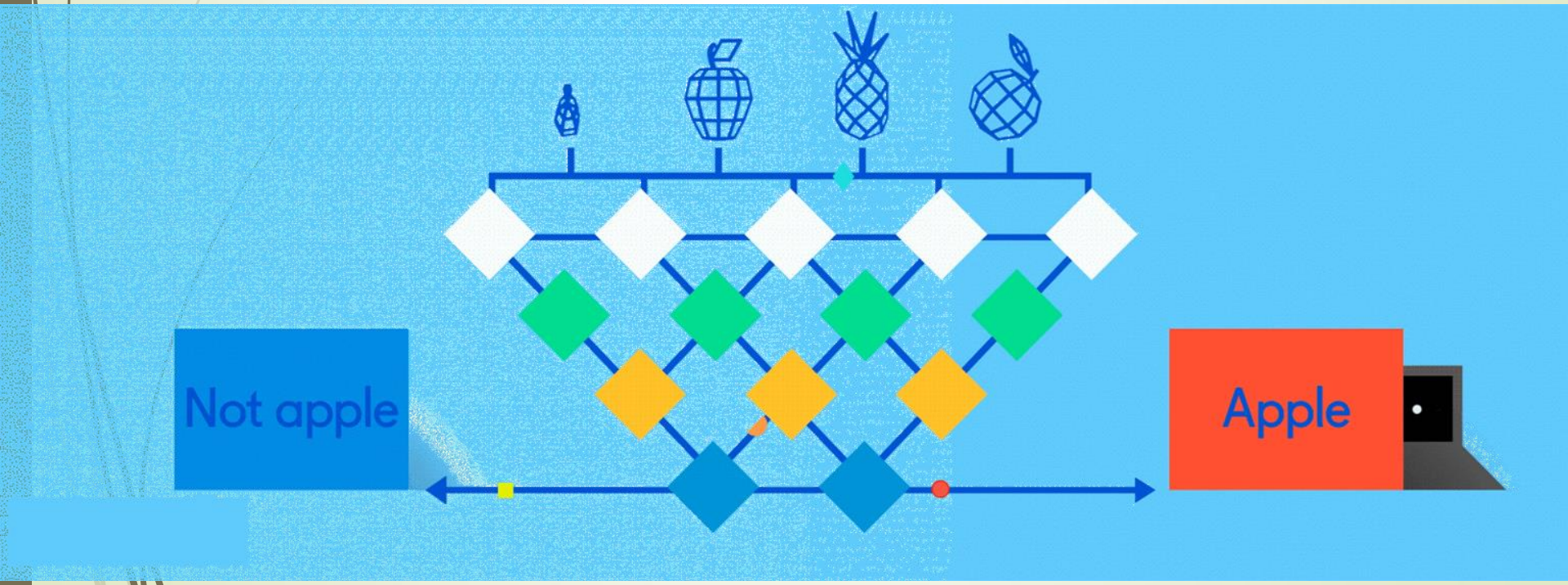
acc = sum(YPred == YTest)./numel(YTest)

acc = 0.9703

Predykcja raka jelita grubego



Dziękuję za uwagę !



https://towardsdatascience.com