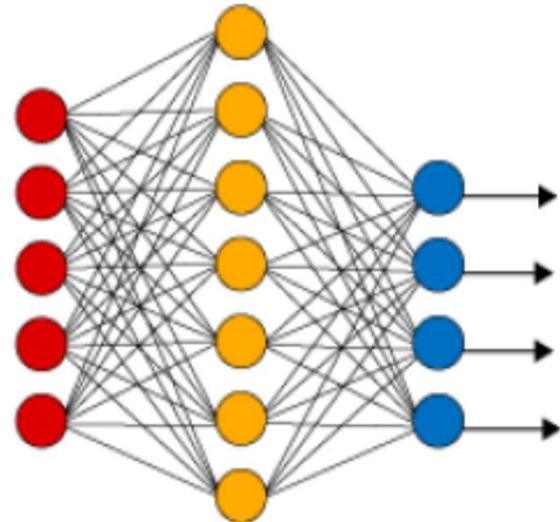


Aplikácie rozpoznávania konvolučné neurónové siete (hlboké siete)

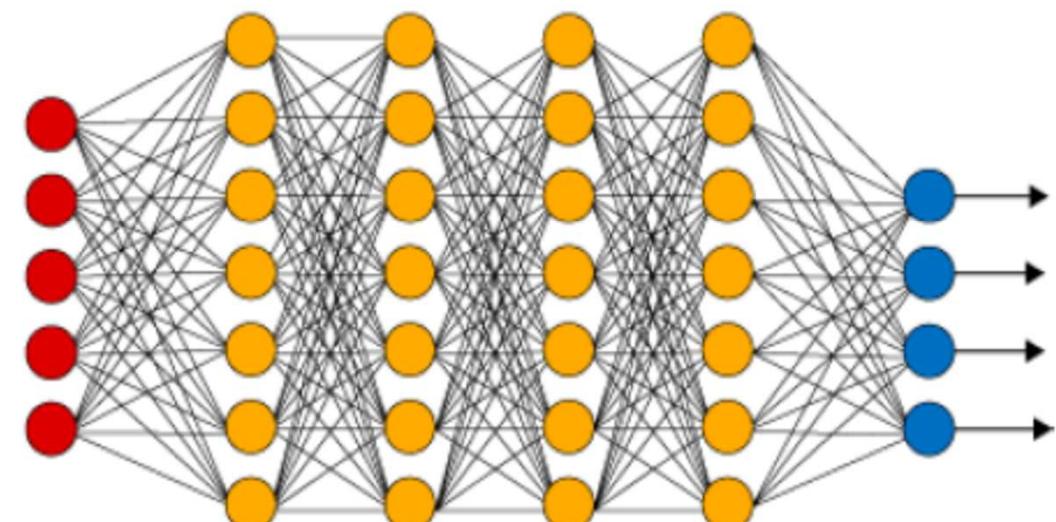
(deep neural network, deep learning)

Plytká vs. hlboká neurónová siet'

Jednoduchá/plytká siet'



Komplexná/hlboká siet'



Vstupná vrstva



Skrytá vrstva



Výstupná vrstva

Hypotéza:

Prehlbovanie architektúry vytvára komplexnejšie mapovanie/príznaky vedúce k zvýšeniu presnosti modelu.

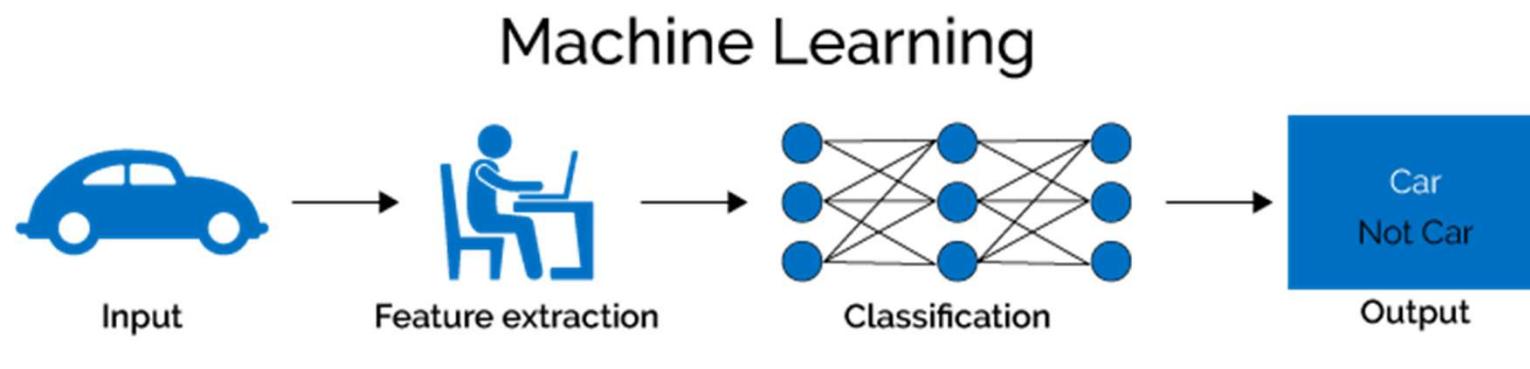
Hlboké učenie vs. Strojové učenie

Hlboké učenie :

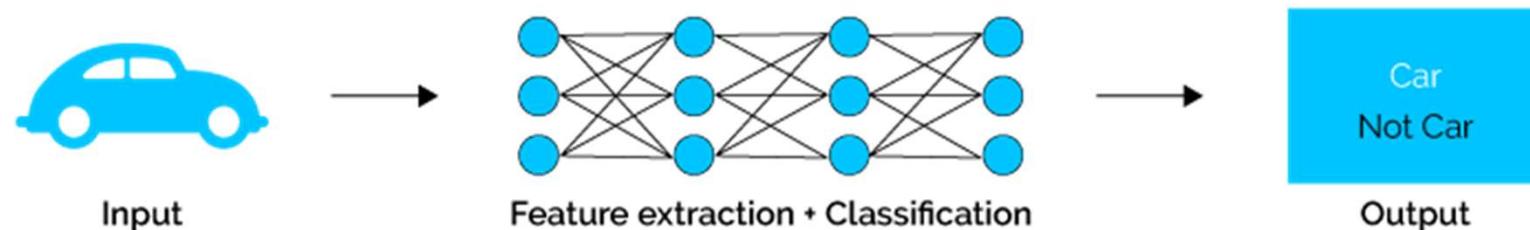
- momentálne „state-of-the-art“
- efektívne škálovateľné s objemom dát
- nie je potrebné ručne extrahovať príznaky (predspracovanie dát)
- prispôsobiteľné a ľahko prenosné na nové problémy

Strojové učenie :

- vyššia úspešnosť pri malom množstve dát
- finančne a výpočtovo nenáročné
- jednoduchšia interpretácia výsledkov



Deep Learning



Konvolučné neurónové siete

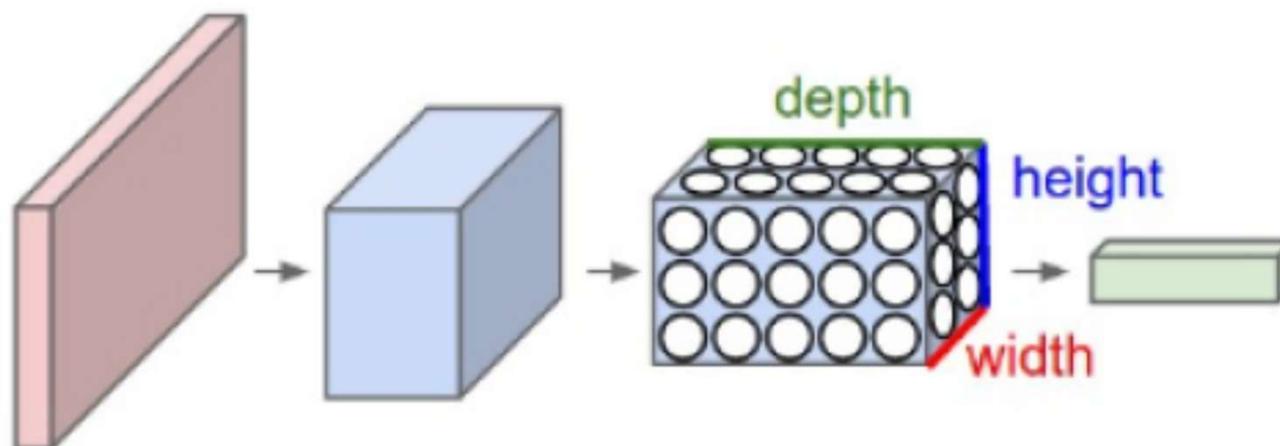
- ❖ Inšpirované biologickou podstatou spracovania vizuálnych vnemov v mozgu cicavcov.
- ❖ Využívajú matematickú operáciu konvolúcie namiesto klasického maticového násobenia aspoň v jednej z jej vrstiev.
- ❖ Architektúra týchto sietí je priamo prispôsobená na spracovanie obrazových informácií.
- ❖ Veľká miera invariantnosti voči posunu, zmene mierky alebo iným formám skreslenia.
- ❖ Nevýhody: zložité nastavenie optimálnej štruktúry siete, veľký počet parametrov siete, dlhý čas učenia, potreba veľkého množstva dát

Architektúra konvolučnej neurónovej siete

Najčastejšie používané vrstvy sú:

Konvolučná , Zhlukovacia (pooling), Plne -prepojené vrstvy

Usporiadaním týchto výpočtových vrstiev vytvoríme celkovú architektúru konvolučnej neurónovej siete.

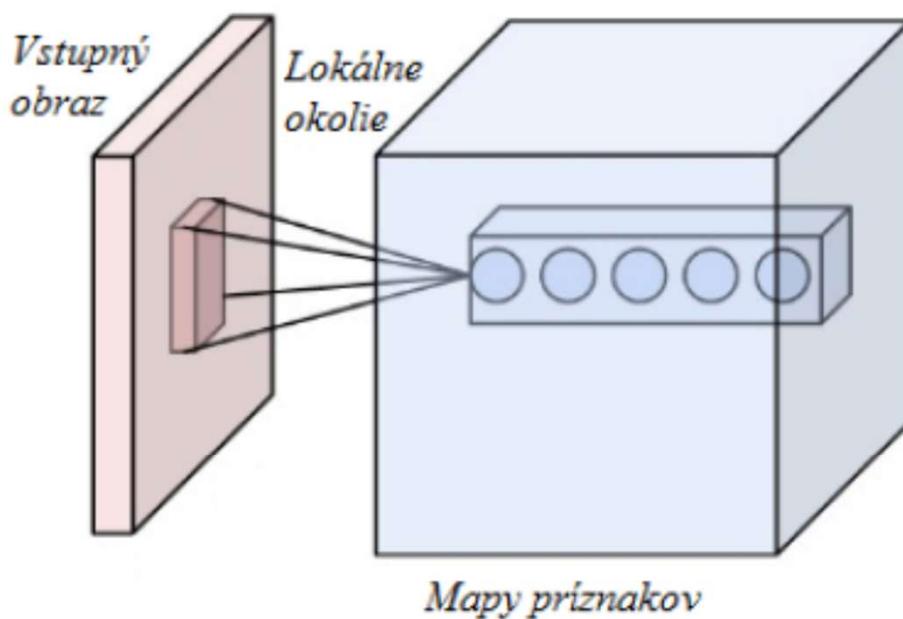


Architektúra konvolučnej neurónovej siete

Konvolučná vrstva

Konvolúcia v dvojrozmernom diskrétnom priestore:

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



Vstup

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

Jadro

1	0	1
0	1	0
1	0	1

Konvolúcia

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

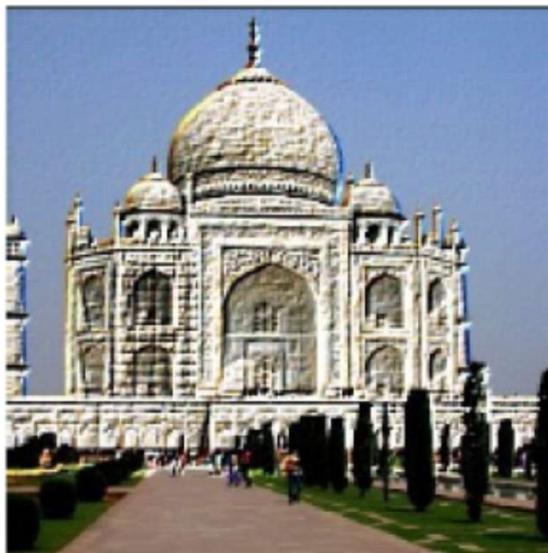
Mapa príznakov

4	3	4
2	4	3
2	3	4

Konvolučná vrstva

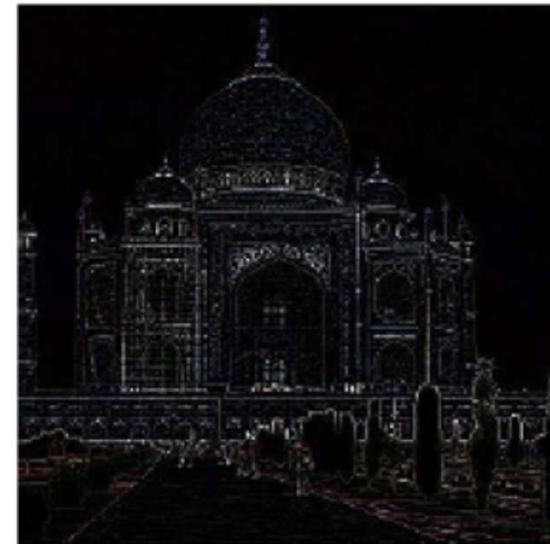
Konvolúcia

- ❖ Môžeme ju vnímať aj ako vyhľadzovanie, rozmazávanie dát zvoleným filtrom.
- ❖ Výstupom konvolúcie je funkcia označovaná ako mapa príznakov
- ❖ Pri použití rôznych jadier získame rôzne mapy príznakov.



Jadro

-2	-1	0
-1	1	1
0	1	2

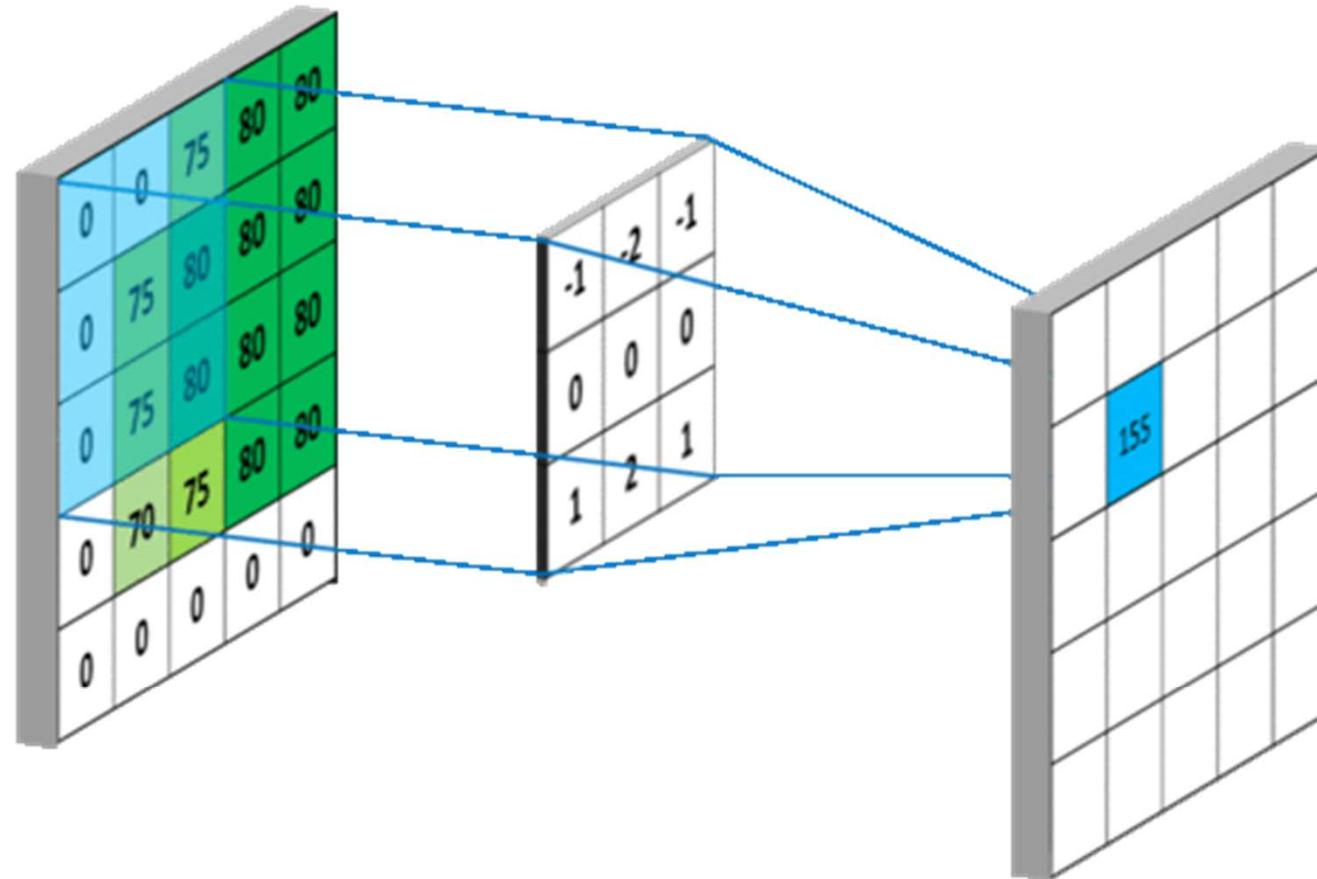


Jadro

0	1	0
1	-4	1
0	1	0

Embosovanie a detekcia hrán zmenou jadra

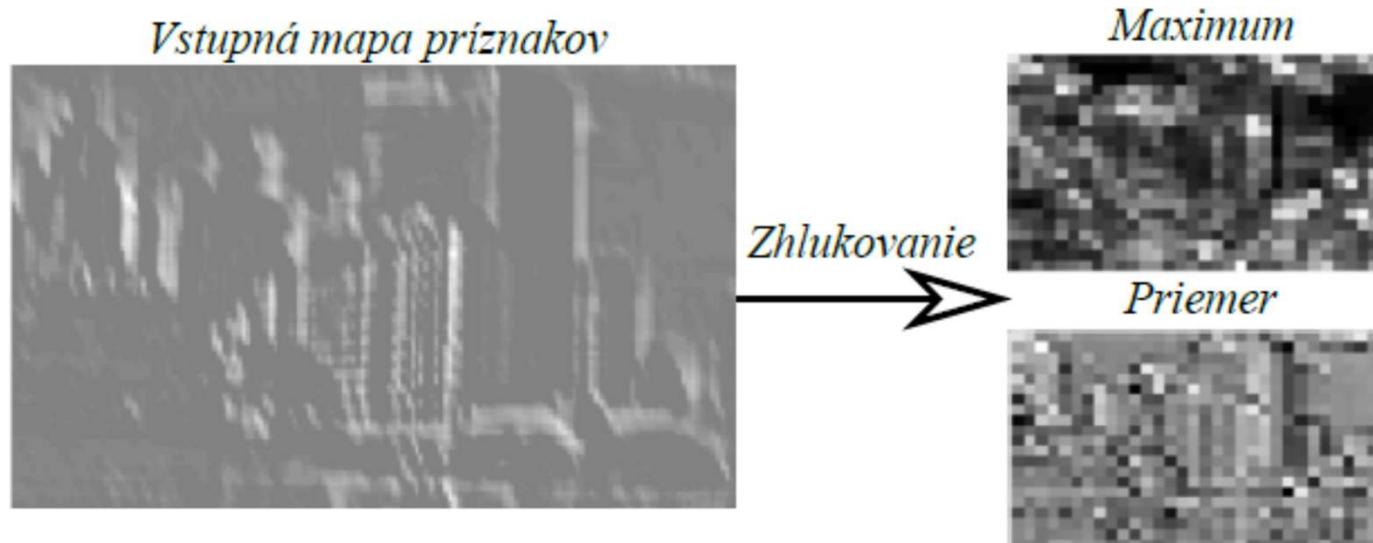
Konvolúcia



Slúži na extrakciu obrazových príznakov

Zhlukovacia vrstva

Redukcia (podvzorkovanie) mapy príznakov, na podvzorkovanie sa používa funkcia maxima alebo priemeru.



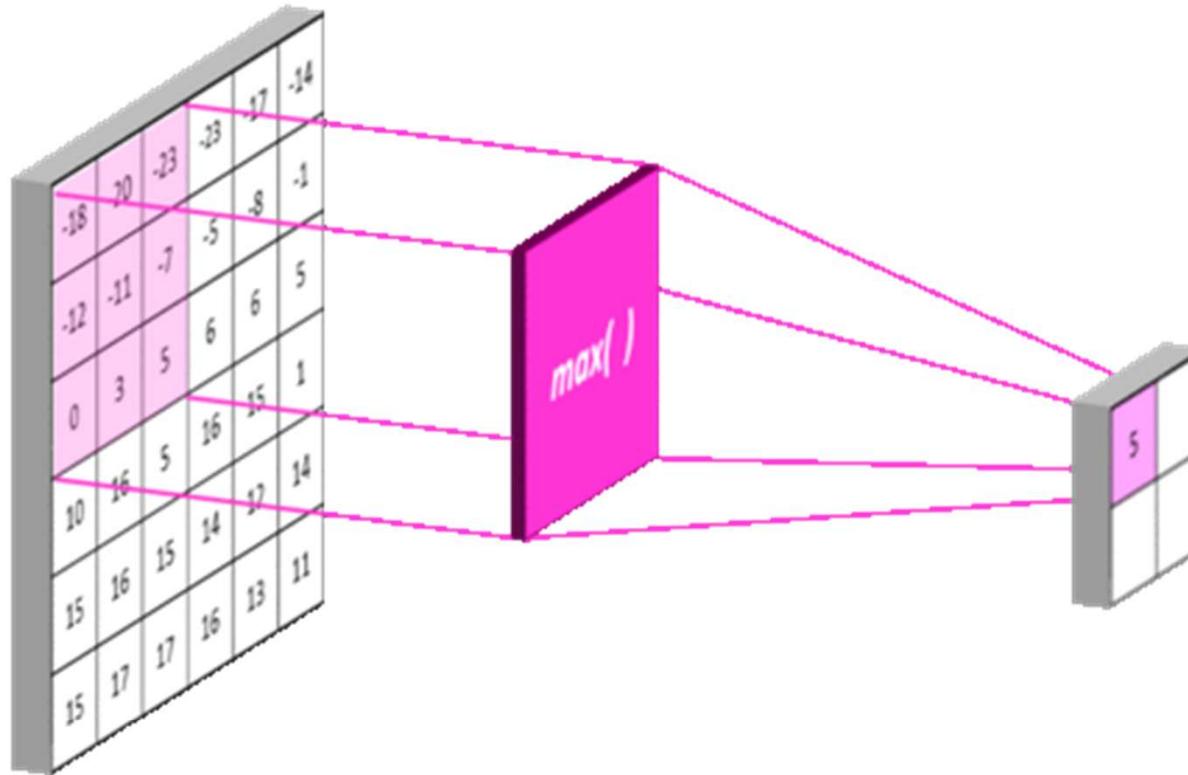
1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

Zhlukovanie (maximum)
Jadro 2x2, krok 2



6	8
3	4

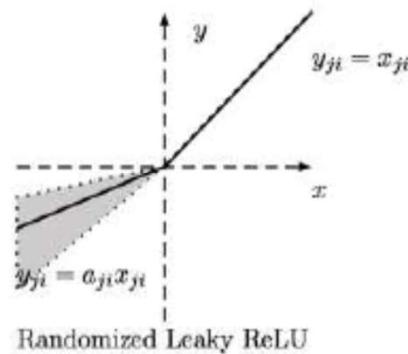
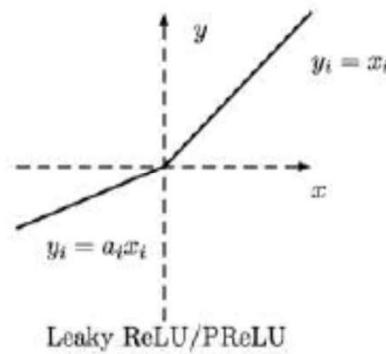
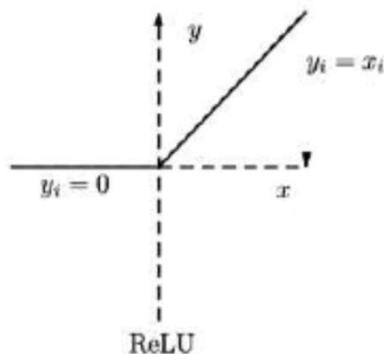
Zhlukovacia (pooling) vrstva



Slúži na redukciu obrazových príznakov

Aktivačná funkcia

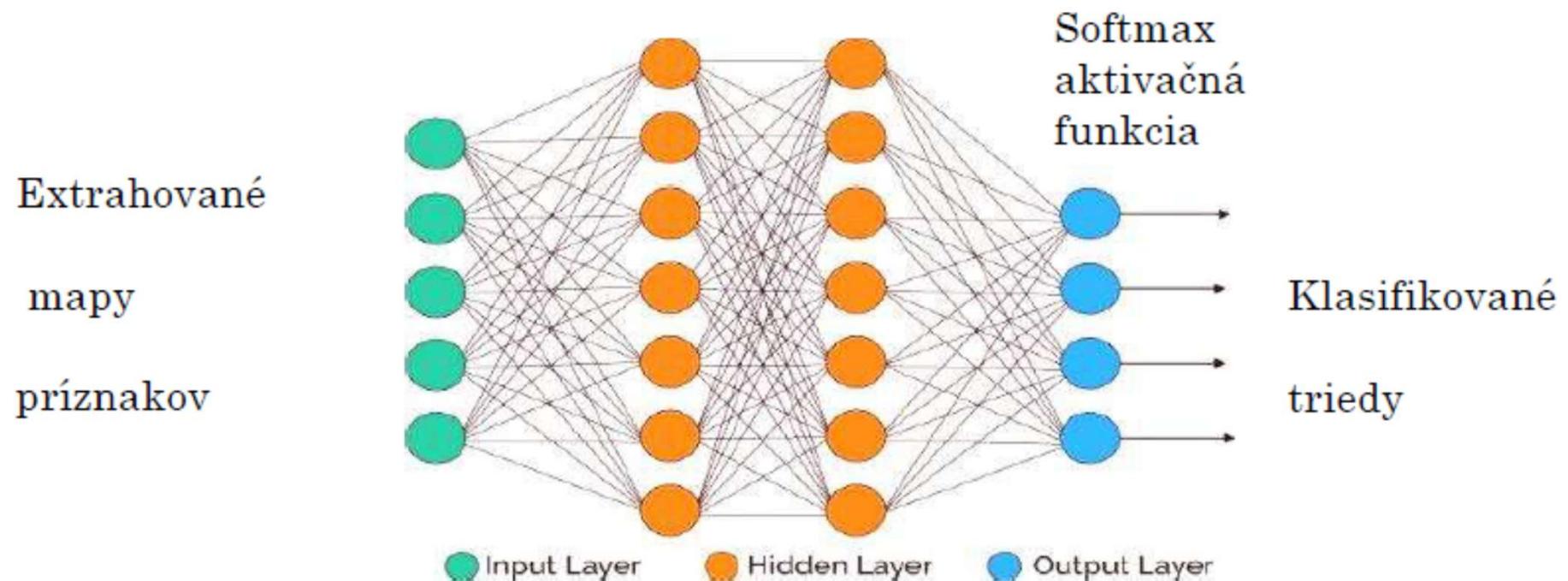
Slúži na zavedenie nelinearity do modelu



Plne prepojená vrstva

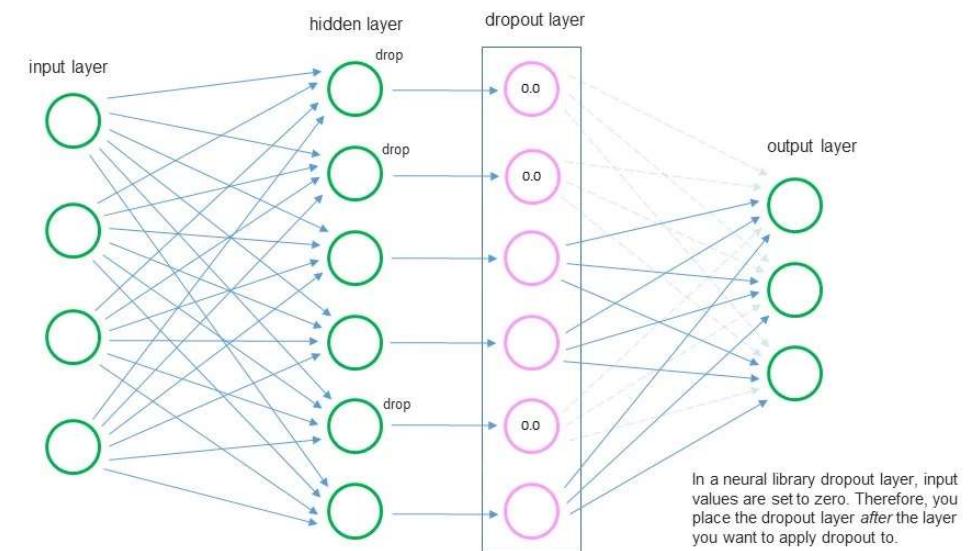
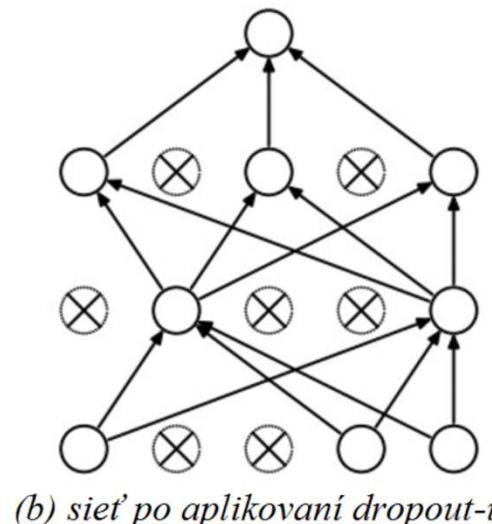
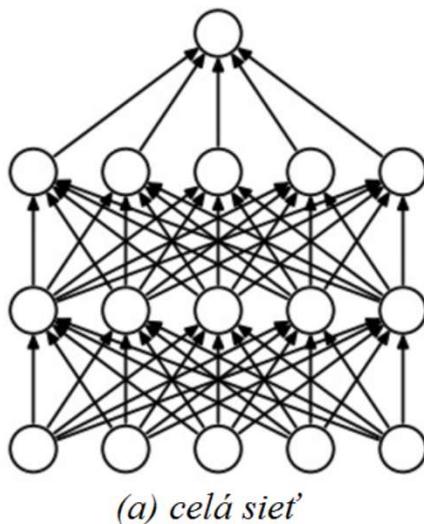
Slúži na klasifikáciu príznakov

- Klasická dopredná neurónová siet



Dropout vrstva

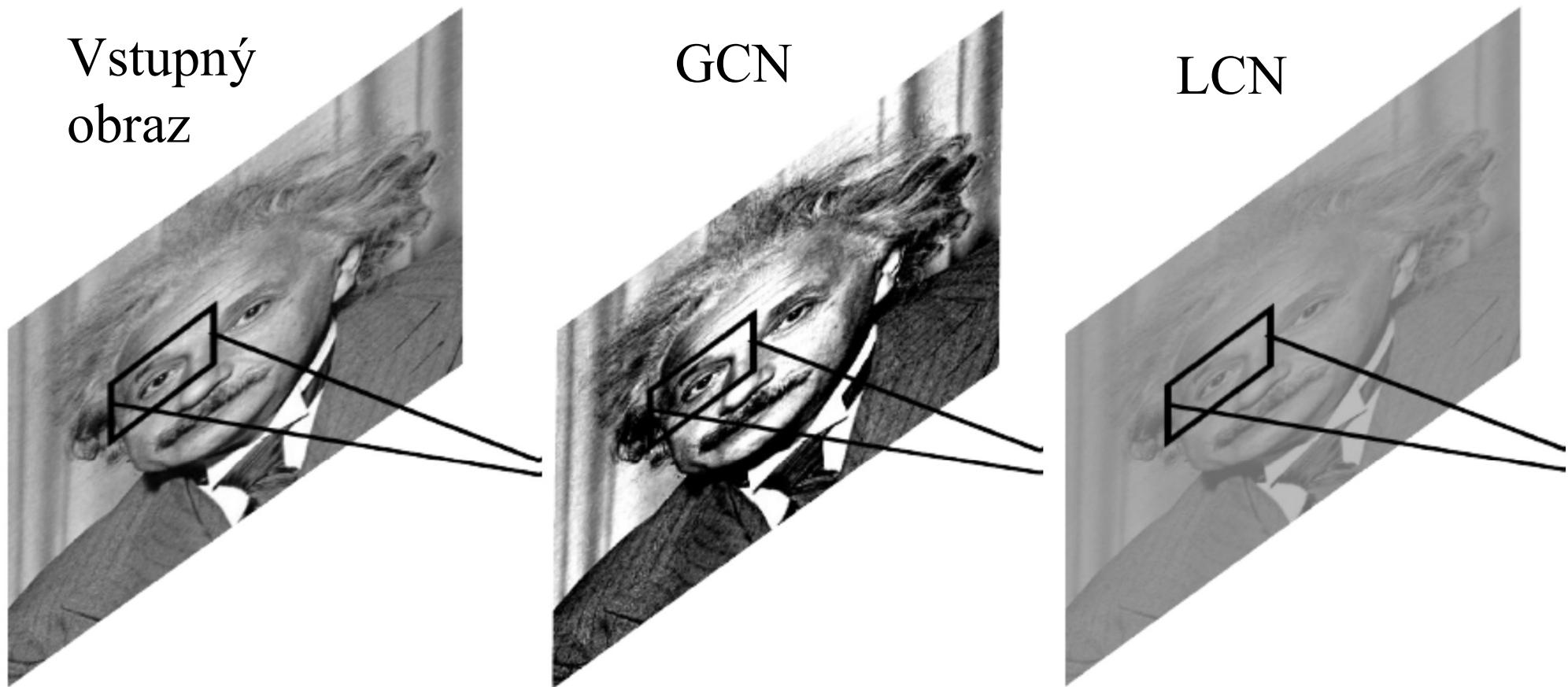
- Opatrenie proti predtrénovaniu, neurpavuje chybovú funkciu ale štruktúru siete.
- Pri trénovaní sa náhodne blokujú určité neuróny, tým je siet nútensá hľadať robustnejšie príznaky. Vo všeobecnosti použitie dropout-u prináša zvýšenie úspešnosti siete vd'aka komplexnejším mapovaniam.
- Dropout sa uplatňuje iba pri trénovaní.



Normalizačná vrstva

Slúži na vyrovnanie kontrastu medzi jednotlivými obrazmi

- Najčastejšie používaná globálna (GCN) a lokálna (LCN) normalizácia kontrastu



Deep learning framework

Caffe



DL4J

Deeplearning4j



MatConvNet

MINERVA

mxnet



theano

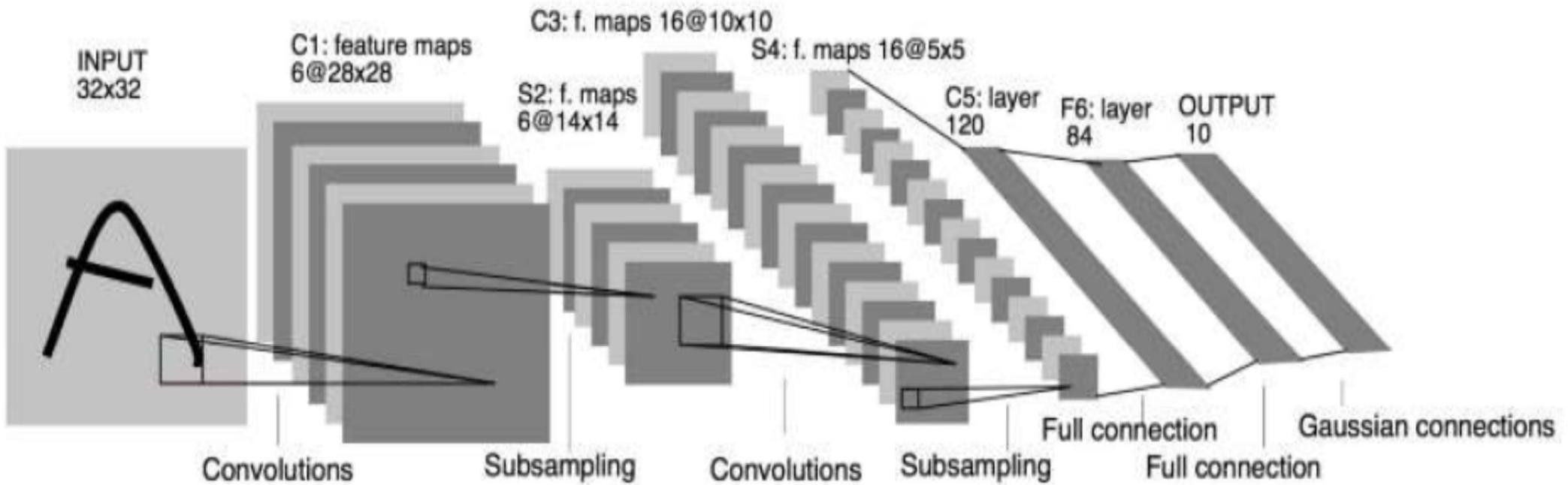


Testované databázy - MNIST

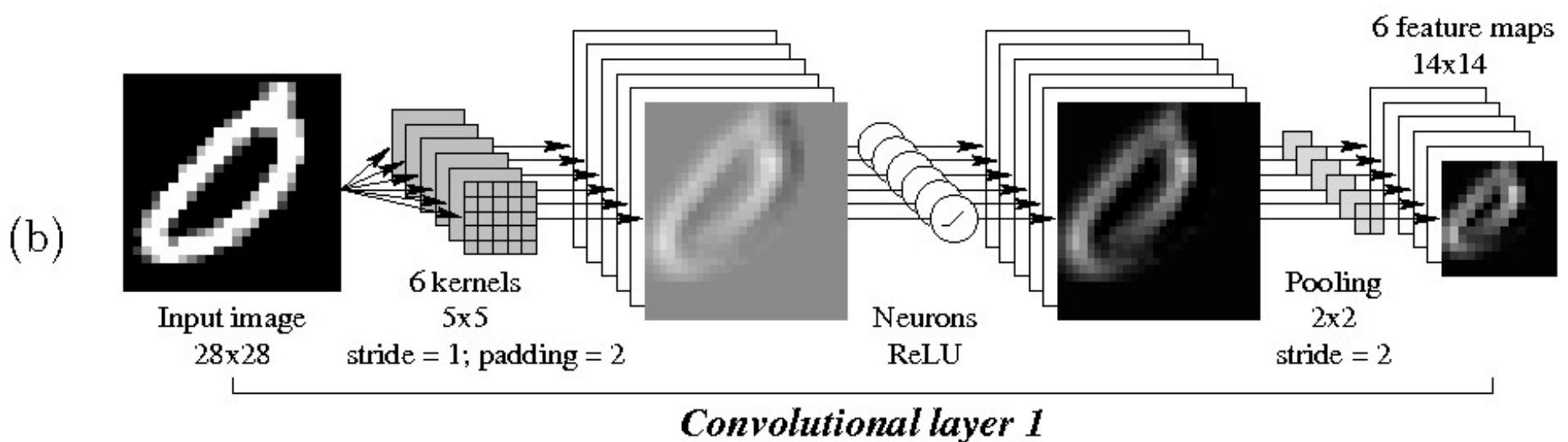
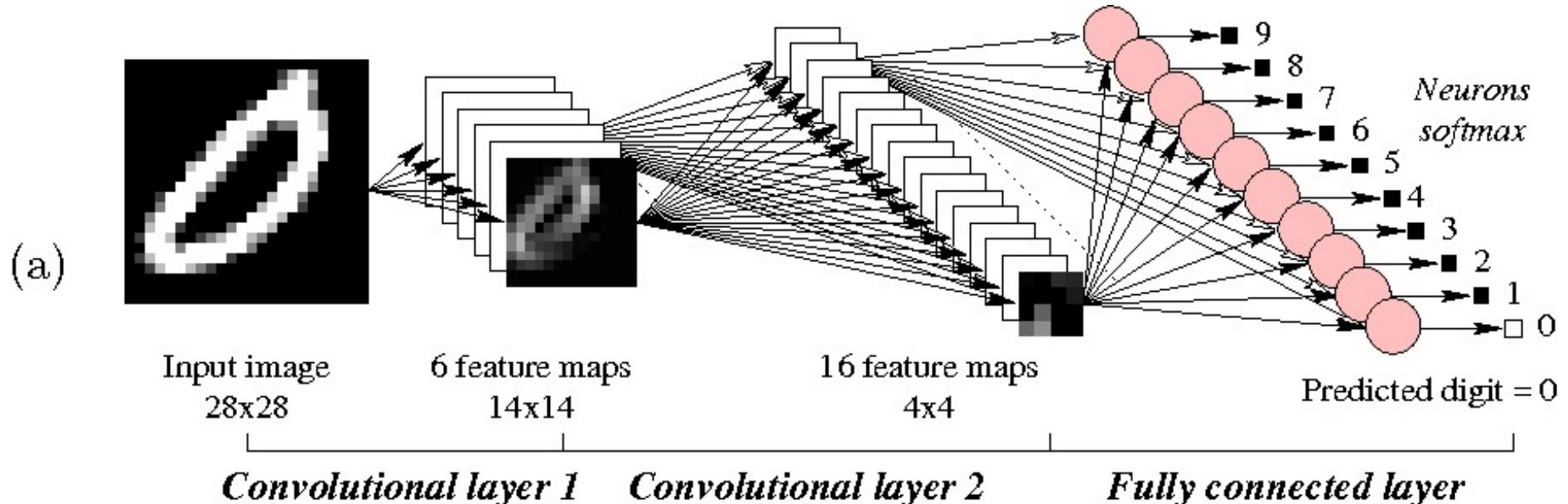
MNIST (Mixed National Institute of Standards and Technology) je databáza ručne písaných číslí. MNIST obsahuje 60 000 trénovacích a 10 000 testovacích dát od približne 250 ľudí, obrázky sú veľkosti 28x28 pixelov.



Rozpoznávanie – LeNet5 (1995)



Príklad štruktúry CNN

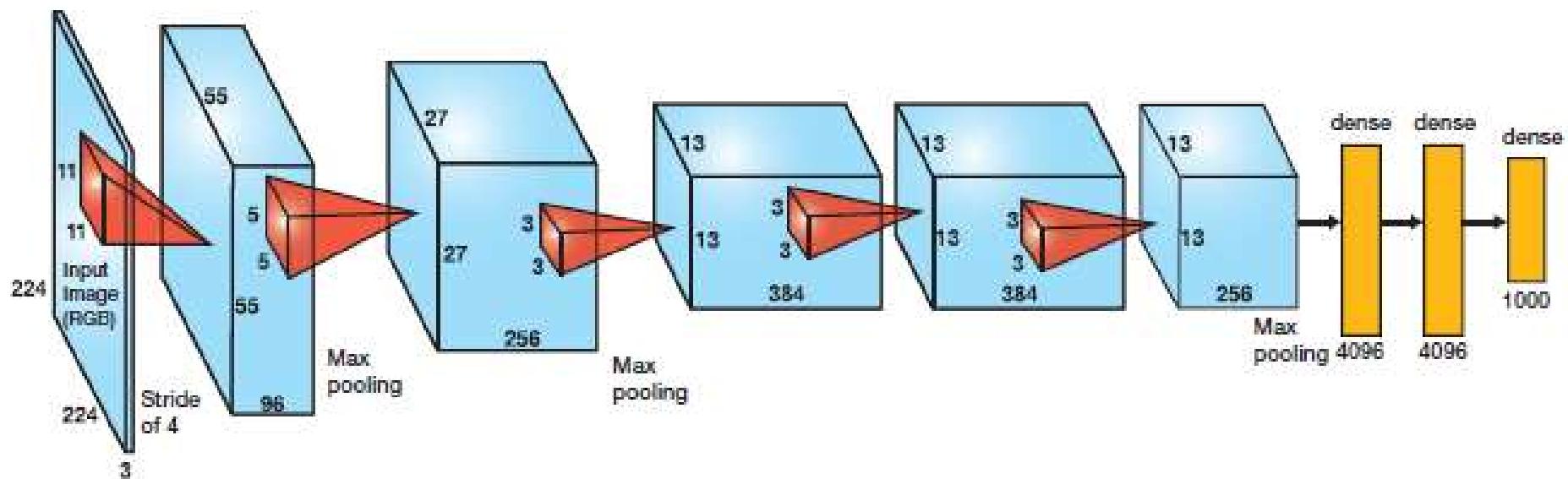


Príklad štruktúry CNN (2 konvolučné vrstvy) v Matlabe

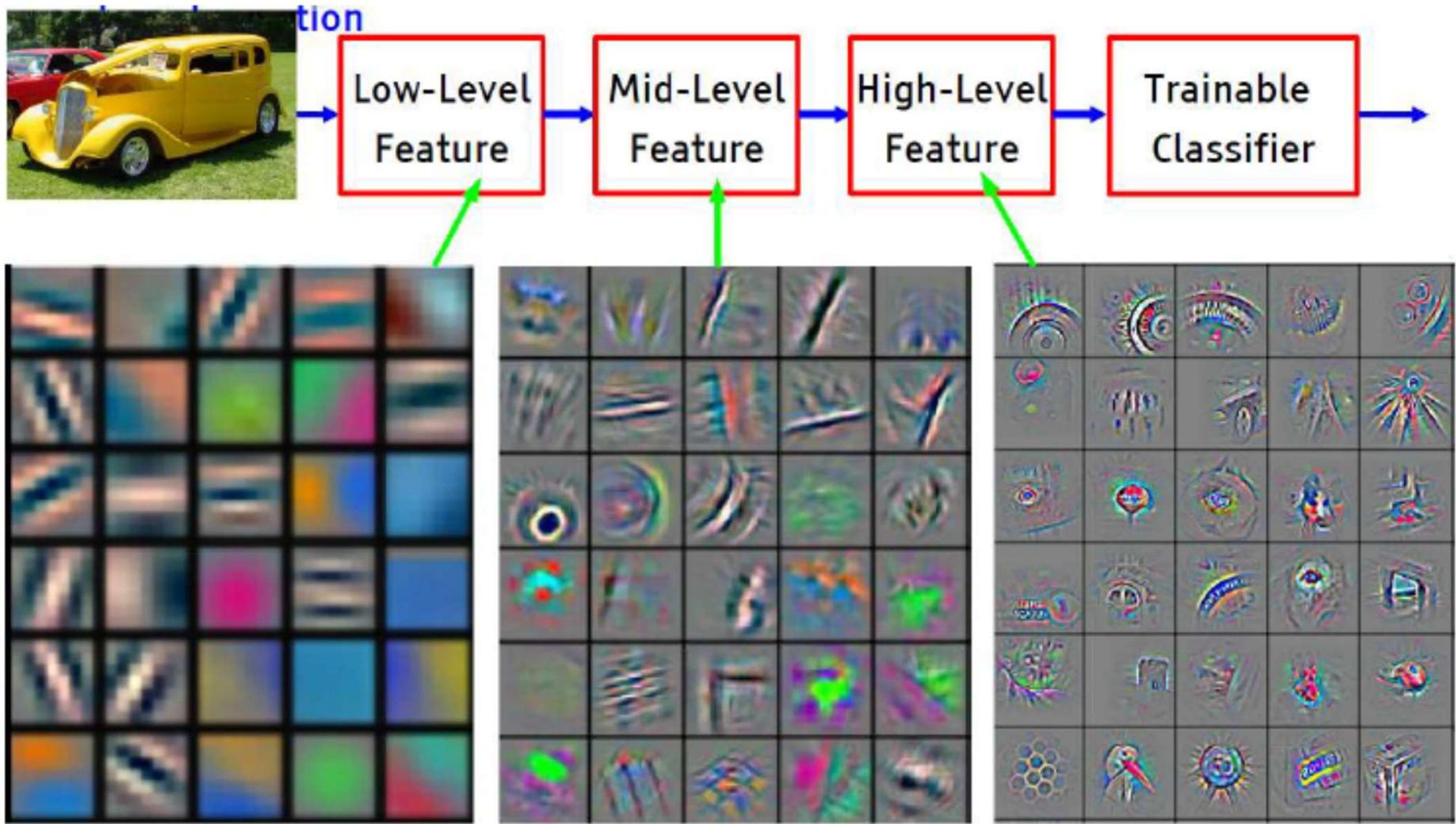
```
>> layers = [imageInputLayer([28 28 1]); % vstupná vrstva – obraz 28x28x1  
    convolution2dLayer(5,6, 'Stride',1, ,Padding',2);  
        % 2D konvolúcia – 6 filtrov, rozmer 5x5, krok 1, doplnie 2 riadkov, stlpov nulami  
    reluLayer(); % relu funkcia  
    maxPooling2dLayer(2,'Stride',2); % max pooling – 2x2, krok 2  
    convolution2dLayer(3,16); % 2D konvolúcia – 16 filtrov, rozmer 3x3  
    reluLayer(); % relu funkcia  
    maxPooling2dLayer(2,'Stride',3); % max pooling – 2x2, krok 3  
    fullyConnectedLayer(10); % plne prepojená vrstva – 10 neurónov  
    softmaxLayer(); % softmax aktivačná funkcia  
    classificationLayer()]; % klasifikačná vrstva – 10 tried
```

Rozpoznávanie objektov na obrázkoch – Alexnet

Alexnet – predtrénovaná siet, na rozpoznávanie 1000 objektov,
Je to hlboká konvolučná siet, má 25 vrstiev, vstupný obrázok má
rozmer 227x227x3, má 5 konvolučných vrstiev, demo ukážky sú aj
Matlabe

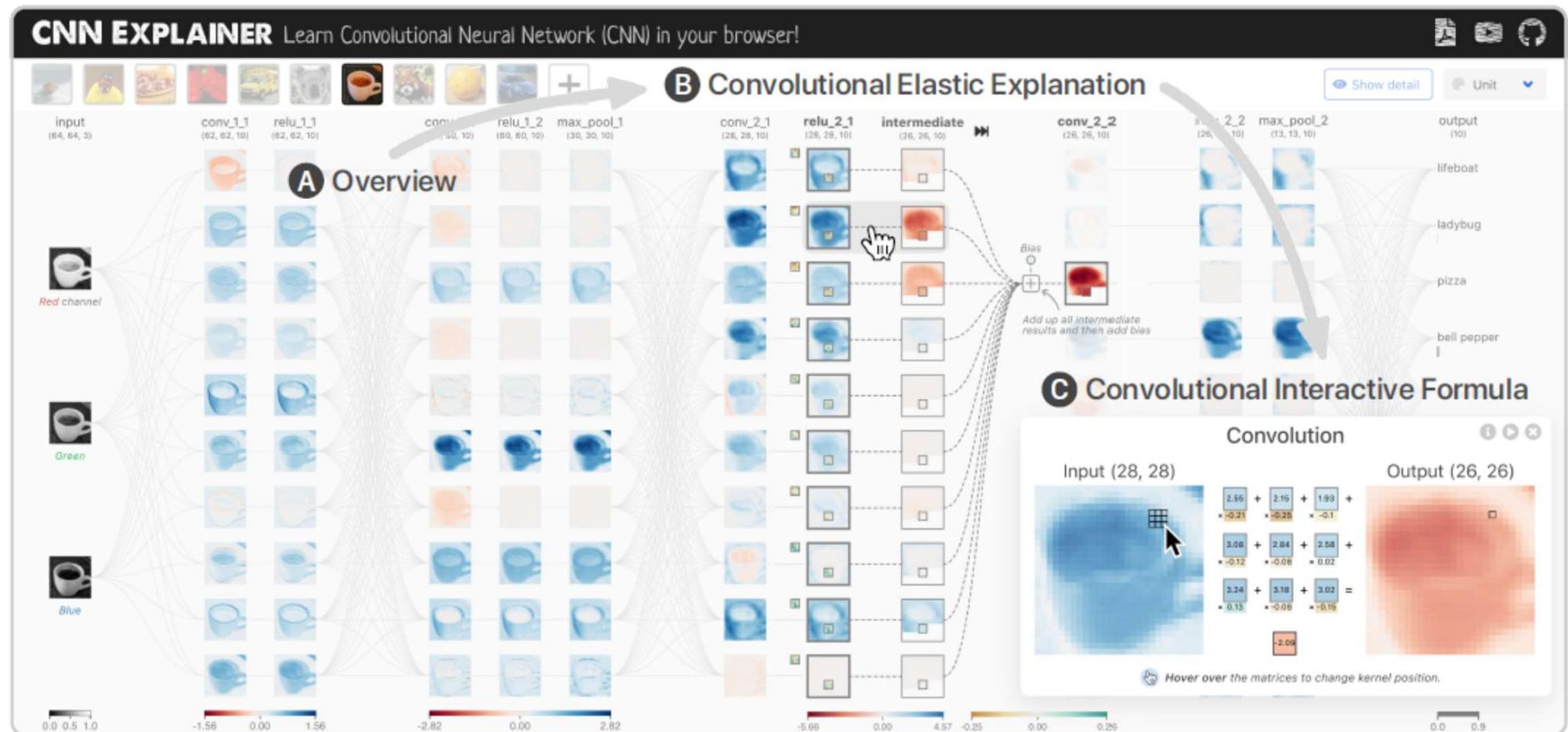


Prezentácia konvolučných vrstiev

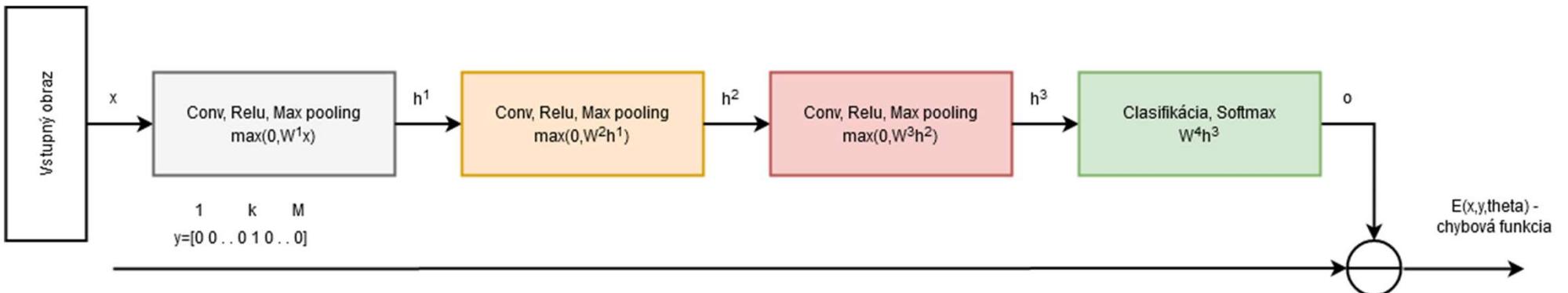


CNN Explainer – interaktívna vizualizácia

Slúži na interaktívnu vizualizáciu a na objasnenie fungovania CNN sietí,
[CNN Explainer \(poloclub.github.io\)](https://poloclub.github.io/CNNExplainer/)



Hlboké učenie (Deep learning)



Pravdepodobnosť klasifikácie do triedy k (softmax funkcia)

$$p(c_k = 1|x) = \frac{e^{o_k}}{\sum_{j=1}^M e^{o_j}}$$

Chybová funkcia – krížová entrópia (cross entropy)

$$E(x, y, W) = - \sum_j y_j \log p(c_j|x) \quad \longrightarrow$$

Stochastic Gradient Descent algorithm

$$W^* = \arg \min_W \sum_{n=1}^N E(x^n, y^n, W)$$

Učenie siete – algoritmus backpropagation

Hlboké učenie (Deep learning)

Stochastic Gradient Descent (najväčšieho spádu)

- Základom metódy najväčšieho spádu je postup po nadrovine chybovej funkcie $E(w)$ s určitým krokom α proti smeru gradientu
- Smer najväčšieho poklesu k stacionárному bodu určuje záporný gradient chybovej funkcie $E(w)$
- Problém s veľkosťou krokom učenia

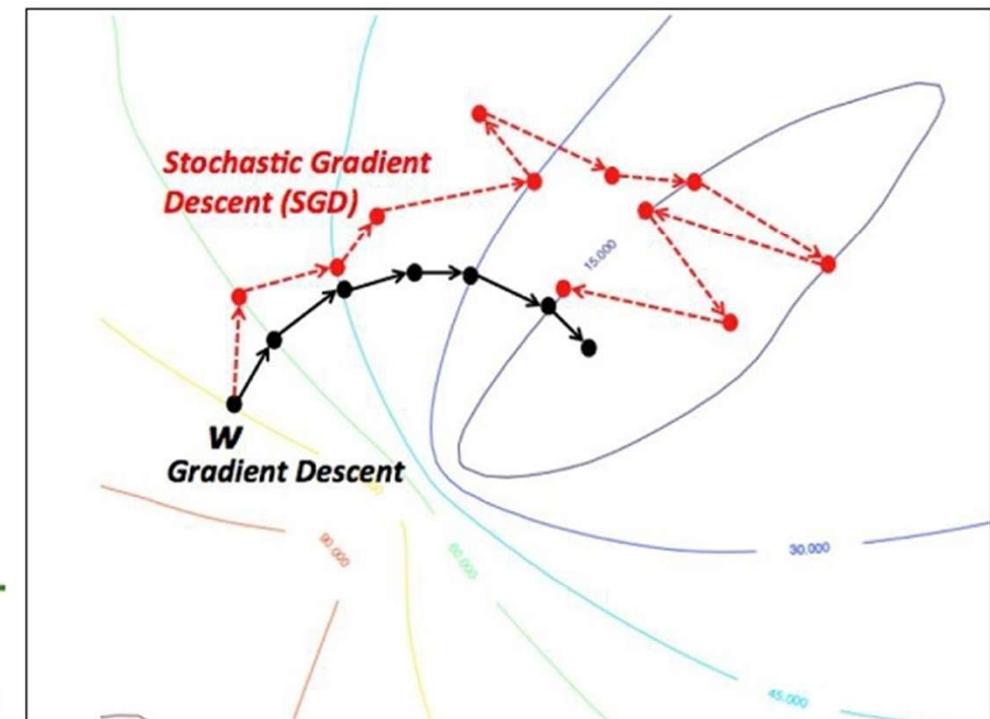
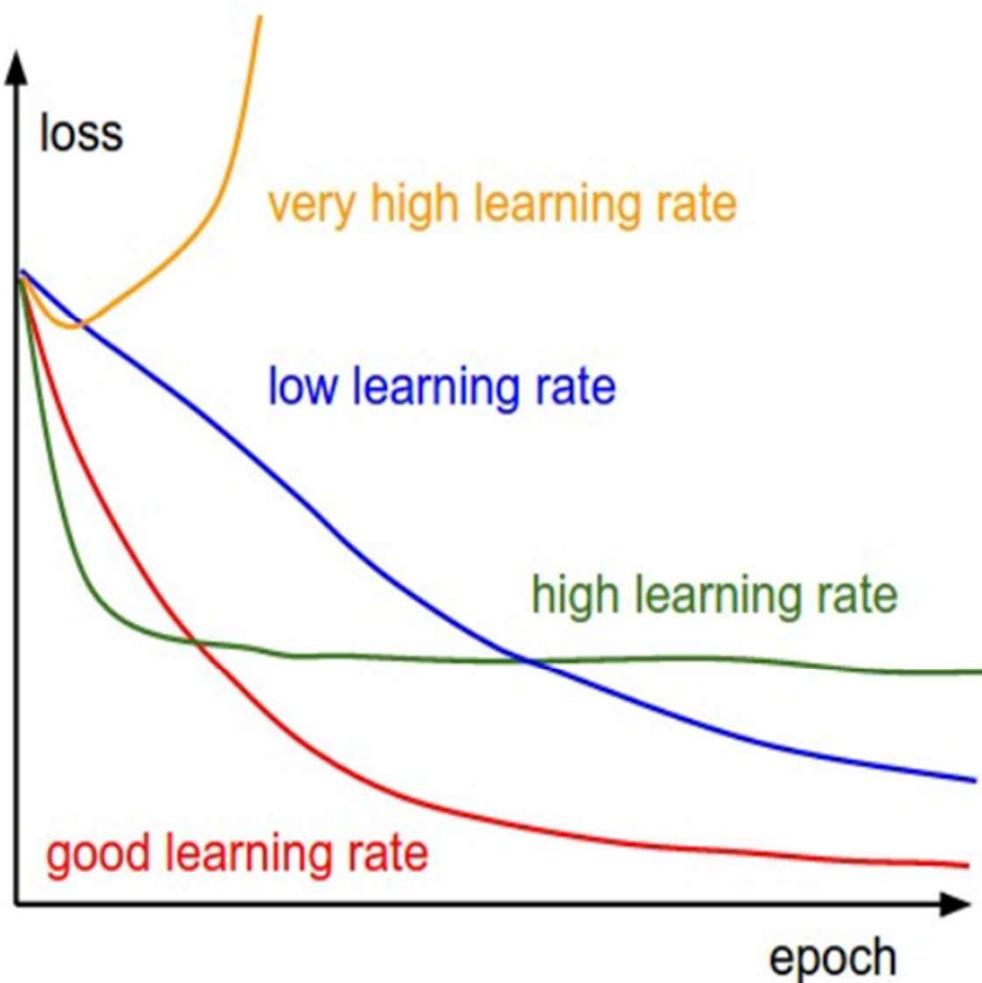
$$w_{k+1} = w_k - \alpha \nabla_w E(w_k)$$

Stochastic Gradient Descent with Momentum

- Modifikácia, pridaním momentovej konštanty
- Na modifikáciu váh vplýva zotrvačnosť predchádzajúcej zmeny váh
- Konštanta zotrvačnosti β (momentum) je z rozsahu 0 až 1, štandardne 0.9

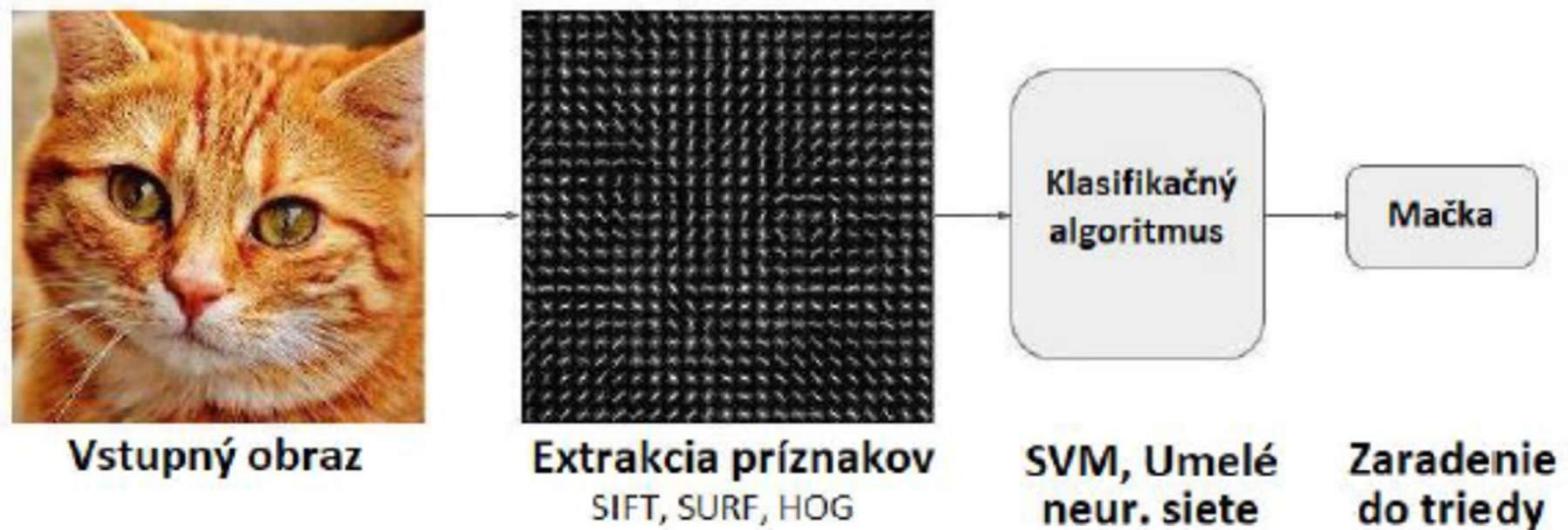
$$w_{k+1} = w_k - \alpha \nabla_w E(w_k) + \beta \Delta w_k$$

Hlboké učenie – SGD – vol'ba kroku učenia

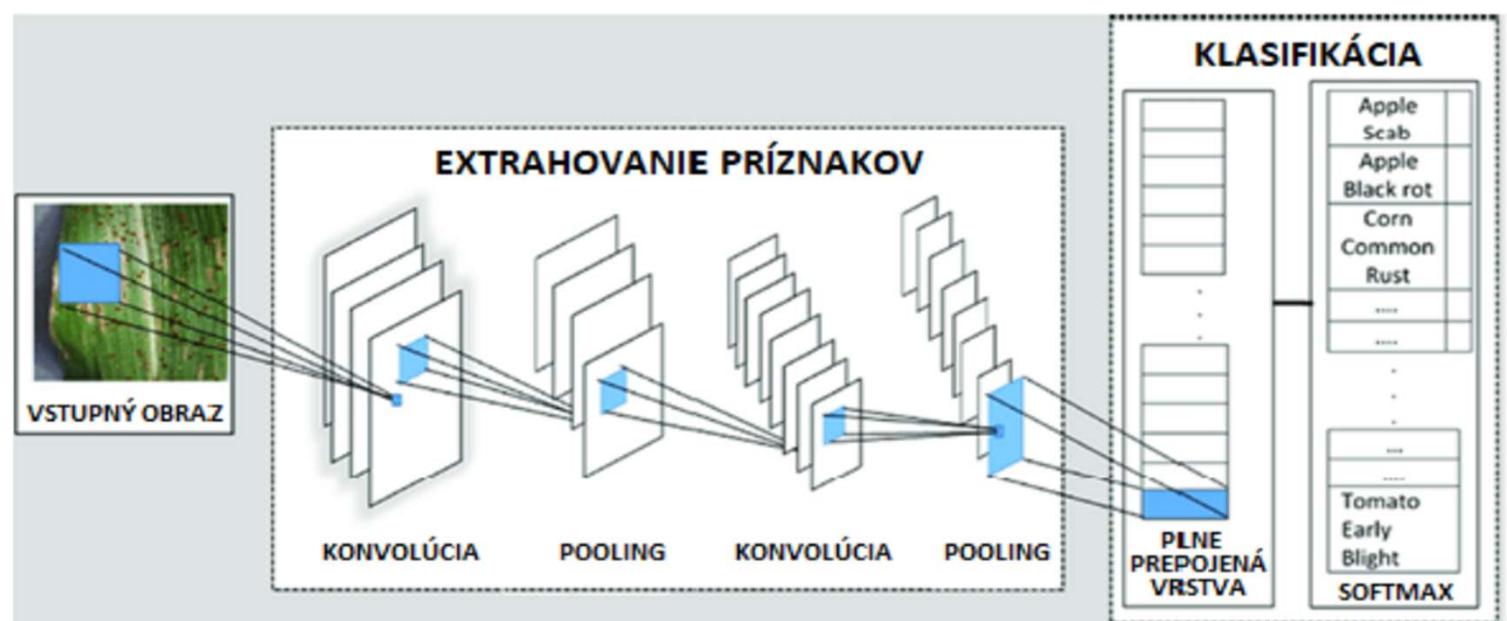


Rozpoznávanie objektov na obrazoch

Klasický
prístup:



Konvolučné
neurónové
siete:



Rozpoznávanie objektov na obrazoch – Caltech Dataset

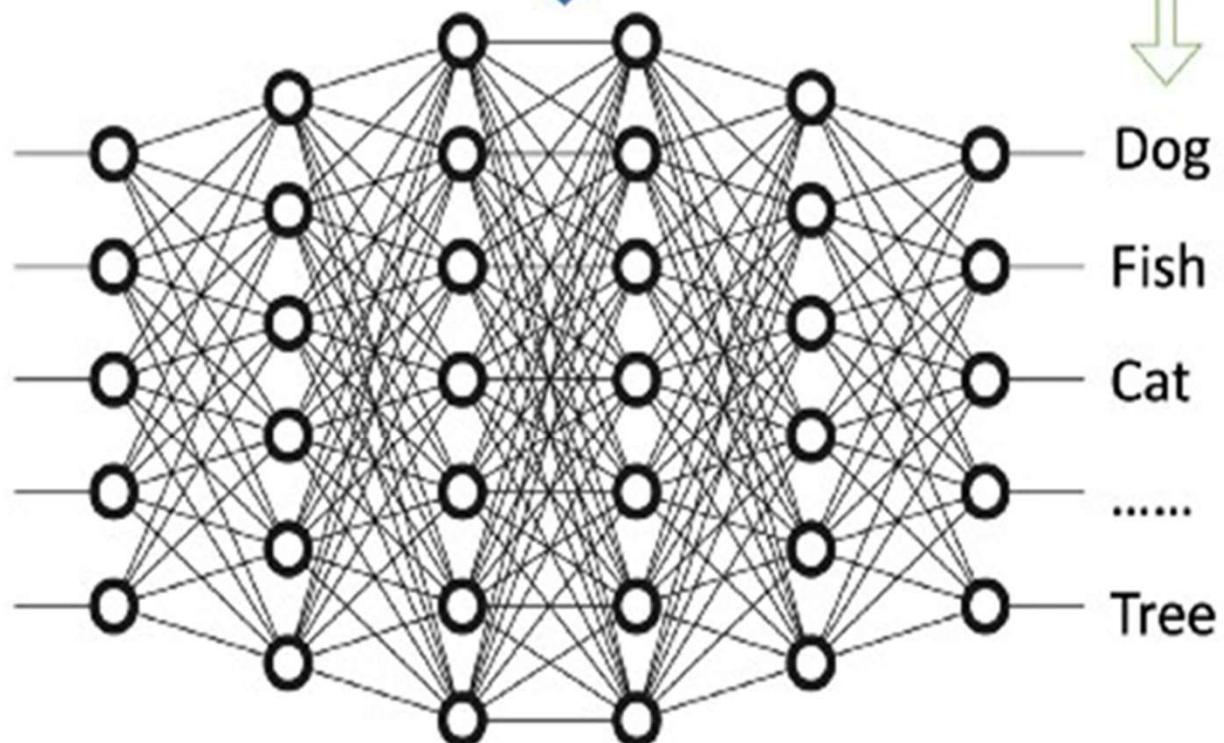
Dog	Fish	Cat	Lion	Bird	Train	Car	Tree
-----	------	-----	------	------	-------	-----	------



pixel features

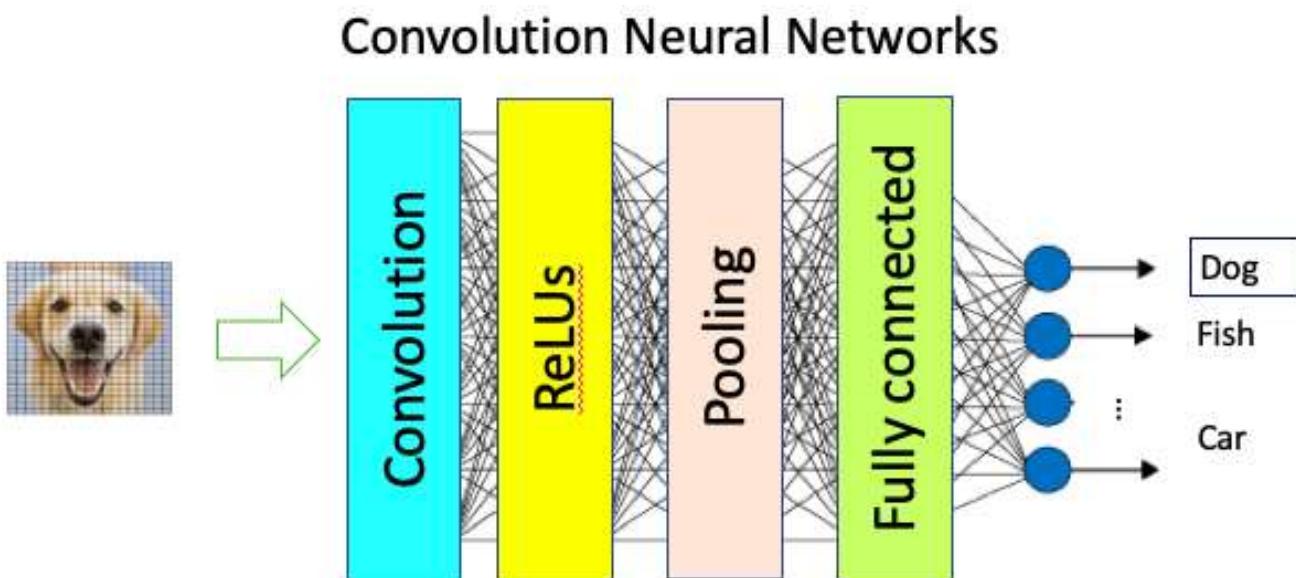
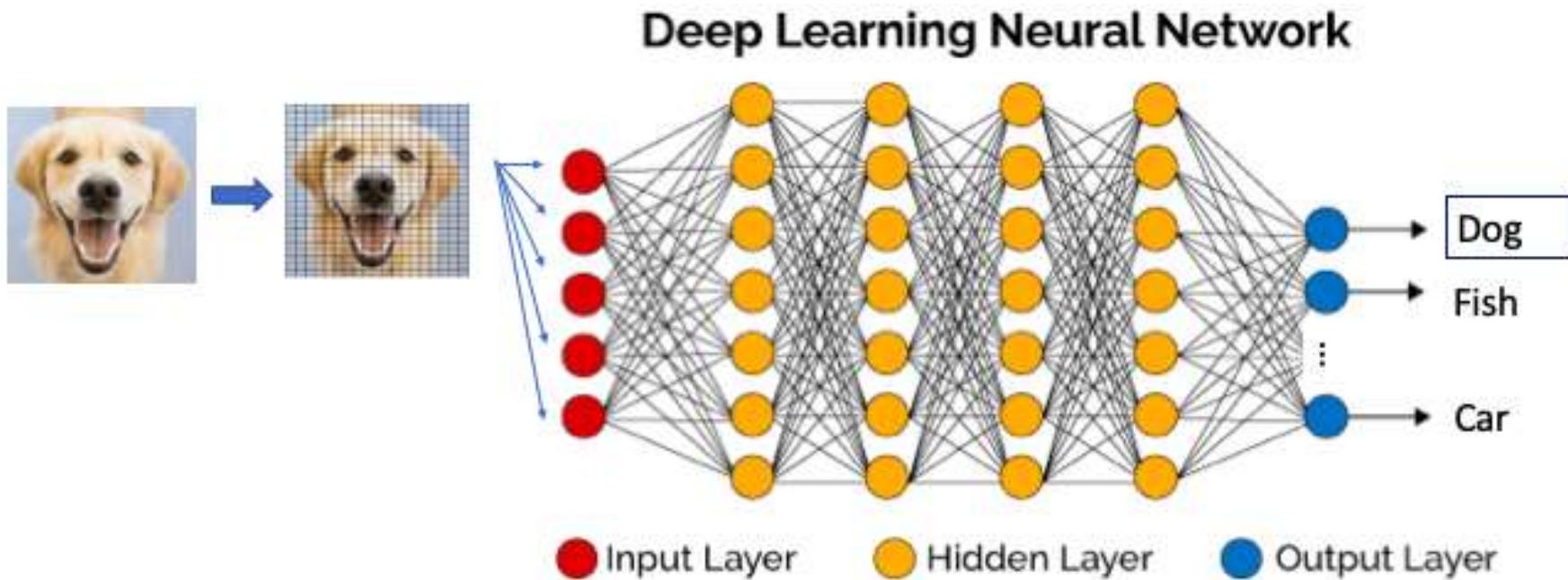


Train the model using
labeled (known) images



Dog
Fish
Cat
.....
Tree

Rozpoznávanie objektov na obrazoch – Caltech Dataset



Rozpoznávanie objektov na obrazoch – Alexnet

- AlexNet – predtrénovaná hlboká konvolučná siet, na rozpoznávanie 1000 objektov

```
>> net=alexnet;
```

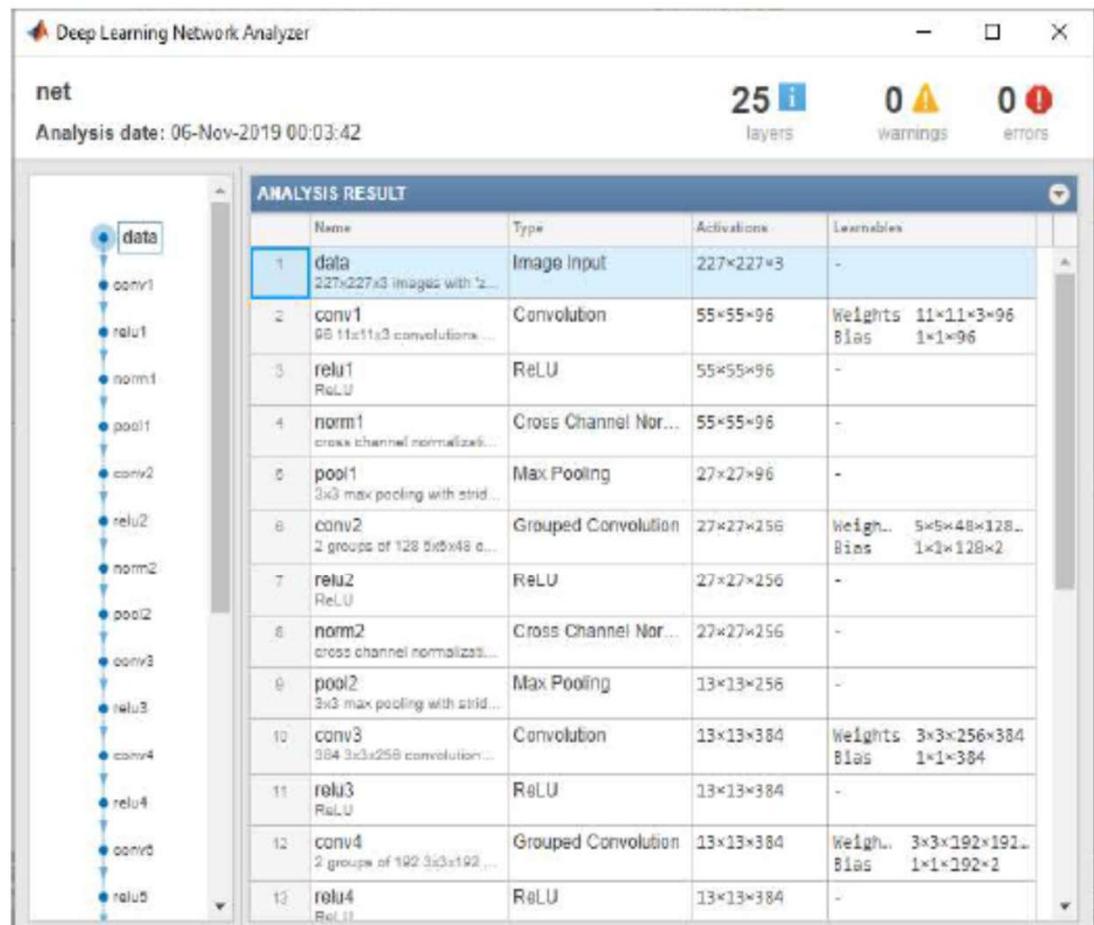
```
>> analyzeNetwork(net)
```

```
>> net.Layers
```

```
ans =
```

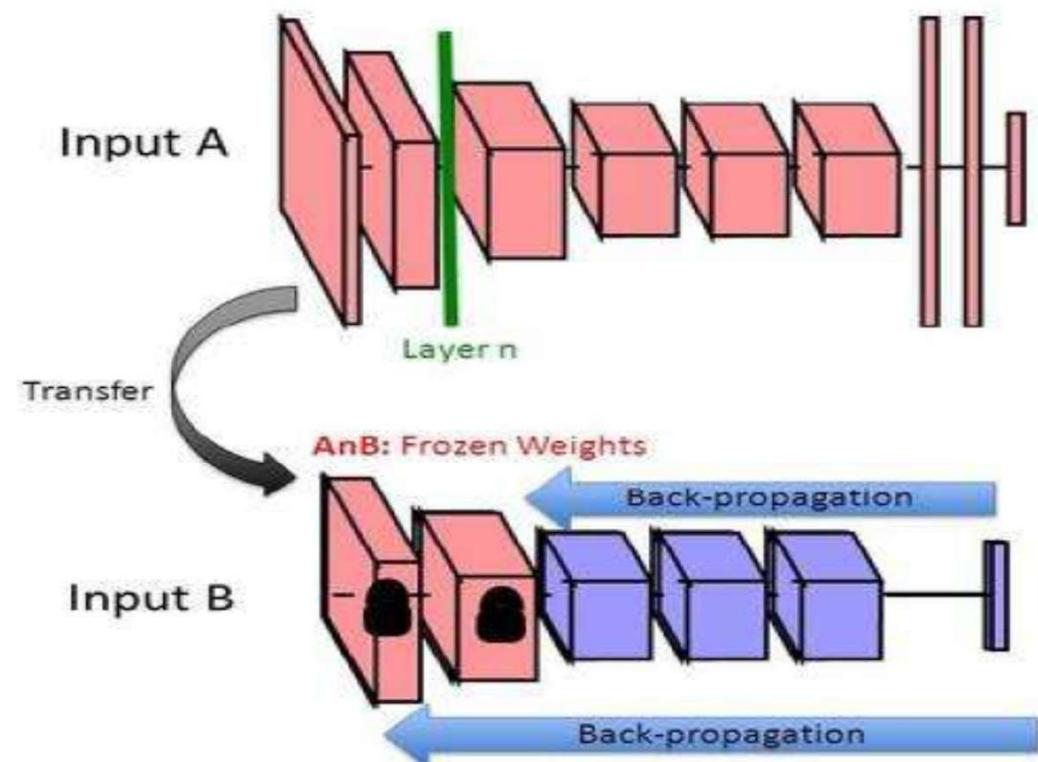
```
25x1 Layer array with layers:
```

1	'data'	Image Input	227x227x3 images with '
2	'convl'	Convolution	96 11x11x3 convolutions
3	'relu1'	ReLU	ReLU
4	'norml'	Cross Channel Normalization	cross channel normaliza
5	'pool1'	Max Pooling	3x3 max pooling with st
6	'conv2'	Grouped Convolution	2 groups of 128 5x5x48
7	'relu2'	ReLU	ReLU



Transferové učenie (Transfer Learning)

- Transfer Learning – využitie časti predtrénovanej sieti a dotrénovanie novej siete
 - Prenos časti natrénovanej siete
 - Zmena zvyšných vrstiev
 - Zvýšený faktor trénovania na nové vrstvy
 - Nové trénovacie data
 - Trénovanie novej siete



Dataset - Matlab

- Vytvorenie datasetu v Matlabe – funkcie s datasetom

```
>> imds = imageDatastore(location) % vytvorenie datasetu  
  
>> [imds1,imds2] = splitEachLabel(imds,p) % rozdelenie datasetu  
  
>> countEachLabel(imds)      % zistenie počtu vzoriek v triedach  
  
>> auimds = augmentedImageDatastore(outputSize,imds)  
% zmena dát (rotácia, posun, veľkosť)  
  
>> aug = imageDataAugmenter
```

Trénovanie / Testovanie siete - Matlab

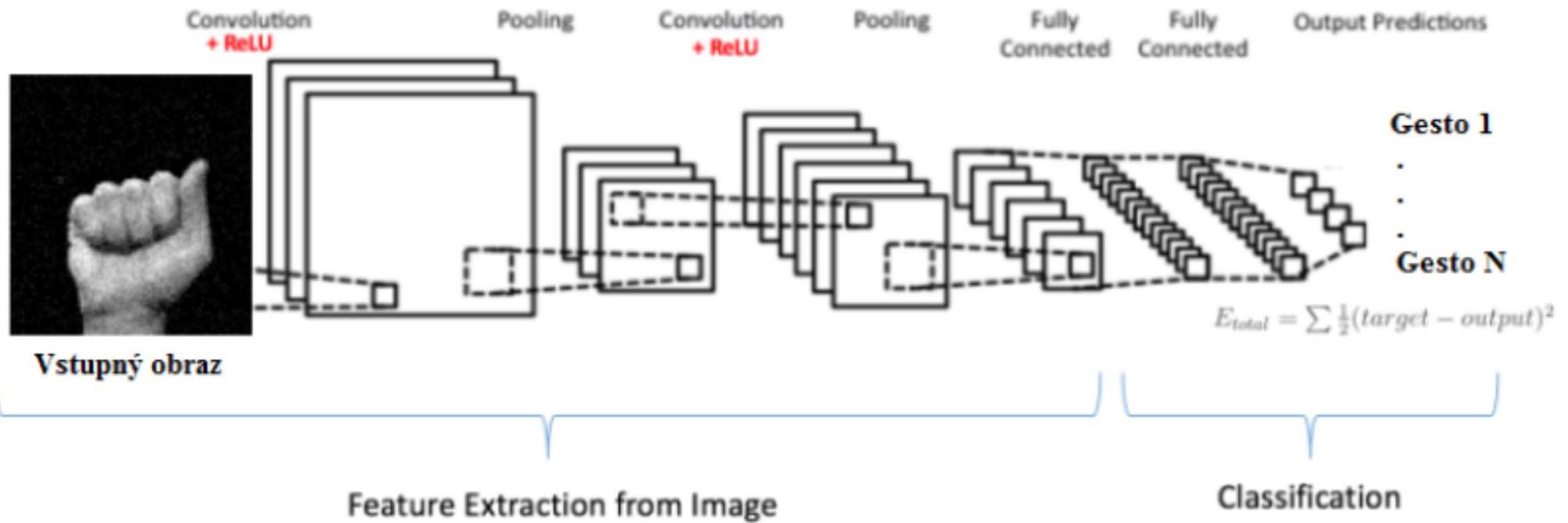
```
>> trainedNet = trainNetwork(imds,layers,options) % trénovanie siete  
  
>> [YPred,scores] = classify(net,X) % testovanie siete - klasifikácia
```

Príklad nastavenia parametrov trénovania v Matlabe

```
>> options = trainingOptions('sgdm',...
    'MaxEpochs',10, ...
    'InitialLearnRate',1e-3, ...
    'ValidationData', {XTest,YTest}, ...
    'ValidationFrequency',50, ...
    'MiniBatchSize',100, ...
    'Shuffle', 'once', ...
    'ExecutionEnvironment', 'cpu', ...
    'Plots','training-progress');
```

% trénovací algoritmus sgdm alebo adam
% počet trénovacích epoch
% počiat. krok učenia
% použitie validačných dát
% pri koľkej iterácii sa vykoná validácia
% veľkosť dávky obrazov pri trénovaní
% premiešanie dát – raz pred trénovaním
% trénovanie na cpu alebo gpu
% zobrazí sa proces trénovania v grafe

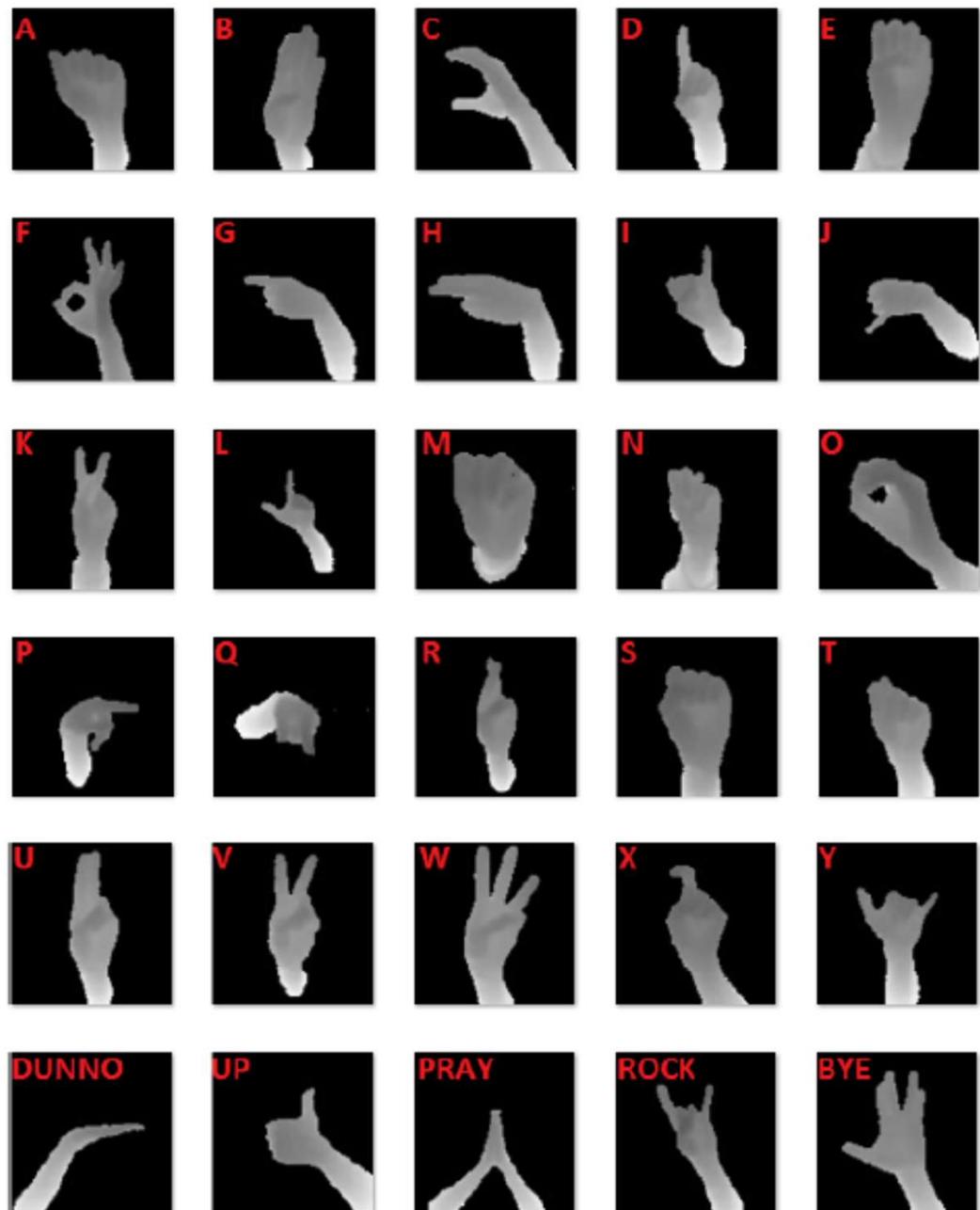
Rozpoznávanie gest pomocou konvolučných sietí



Testované databázy – vlastné izolované gestá

Vlastná databáza 30 gest,
cca 15000 vzoriek, použité
sú hĺbkové dáta s rozmer-
mi 101x101 pixelov.

Databázu vytvárali 3 osoby



Testované architektúry CNN

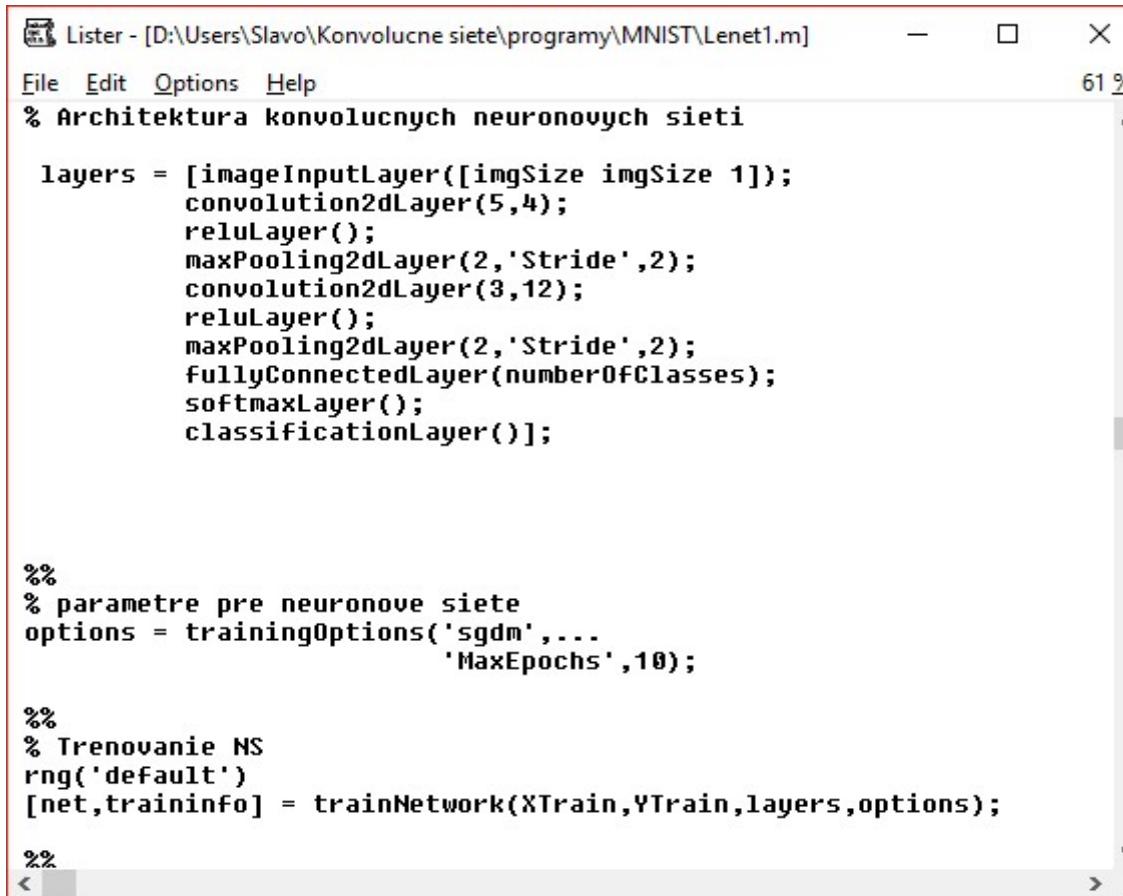
Architektúra 1
vstupná
konvolučná
relu
max pooling
plne- prepojená
výstupná

Architektúra 2
vstupná
konvolučná
relu
max pooling
konvolučná
relu
max pooling
plne- prepojená
výstupná

Architektúra 3
vstupná
konvolučná
relu
max pooling
konvolučná
relu
max pooling
konvolučná
relu
max pooling
plne- prepojená
výstupná

Trénovanie CNN

- ❖ Trénovanie CNN bolo realizované v prostredí Matlab, využitím toolboxu Neural network.
- ❖ Na učenie neurónovej siete bola použitá metóda *stochastic gradient descent*.
- ❖ Učenie CNN bolo realizované na grafickej karte NVIDIA GeForce GTX 1080.



The screenshot shows a Matlab editor window titled 'Lister - [D:\Users\Slavo\Konvolucne siete\programy\MNIST\Lenet1.m]'. The window displays the following MATLAB code:

```
% Architektura konvolucnych neuronovych sieti

layers = [imageInputLayer([imgSize imgSize 1]);
          convolution2dLayer(5,4);
          reluLayer();
          maxPooling2dLayer(2,'Stride',2);
          convolution2dLayer(3,12);
          reluLayer();
          maxPooling2dLayer(2,'Stride',2);
          fullyConnectedLayer(numberOfClasses);
          softmaxLayer();
          classificationLayer()];

%%

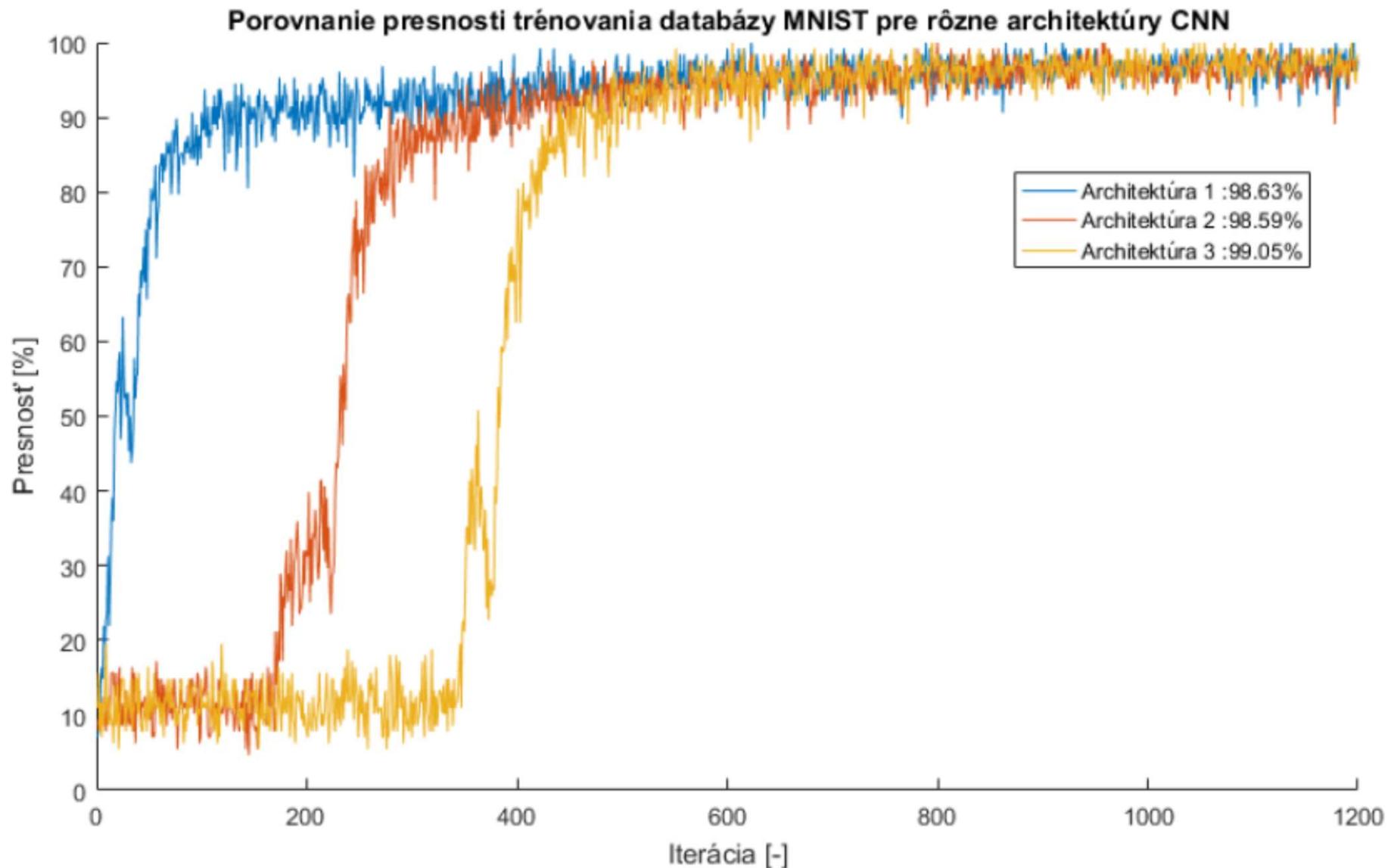
% parametre pre neuronove siete
options = trainingOptions('sgdm',...
                           'MaxEpochs',10);

%%

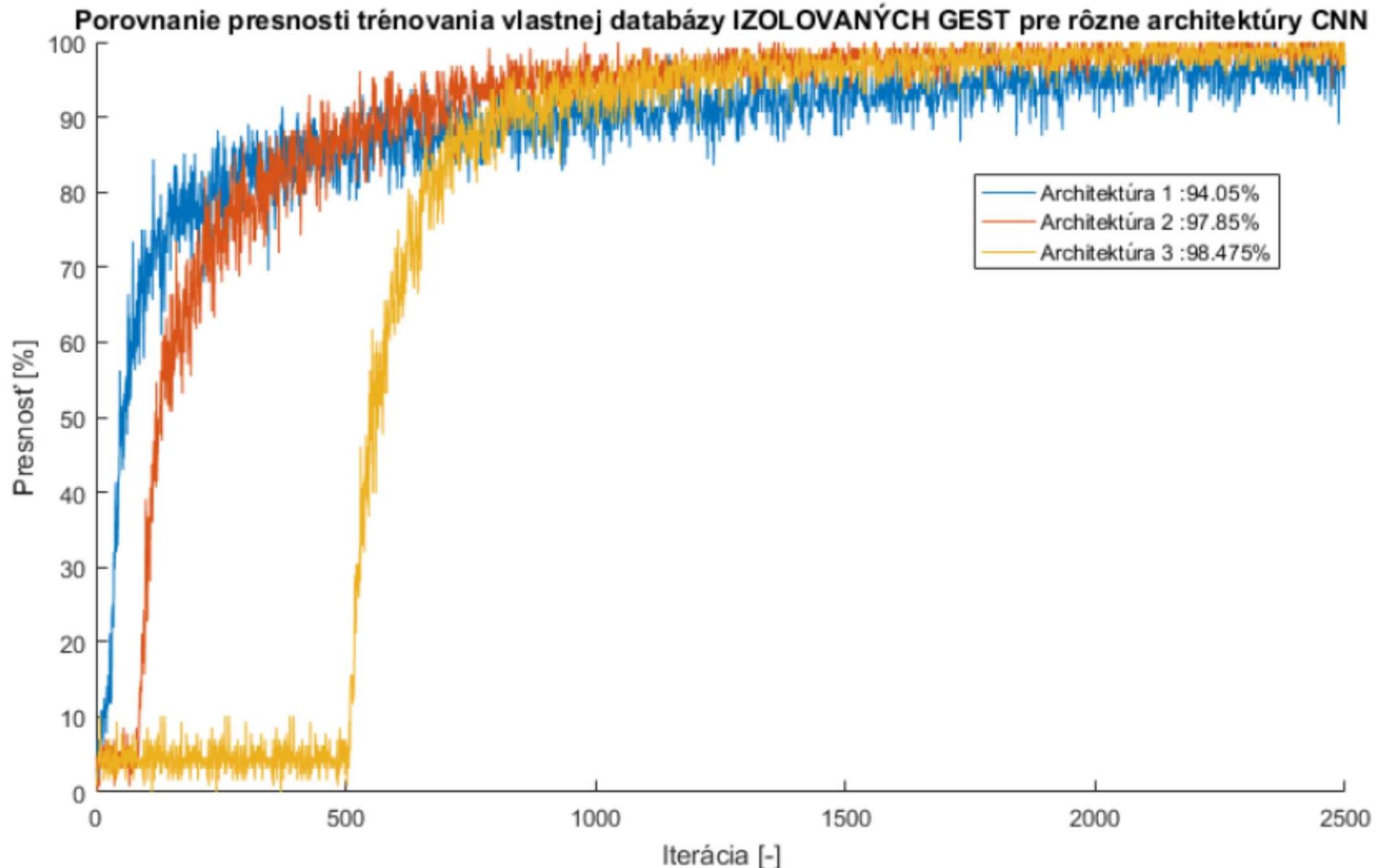
% Trenovanie NS
rng('default')
[net,traininfo] = trainNetwork(XTrain,YTrain,layers,options);

%%
```

Dosiahnuté výsledky – databáza MNIST



Dosiahnuté výsledky – databáza 30 gest



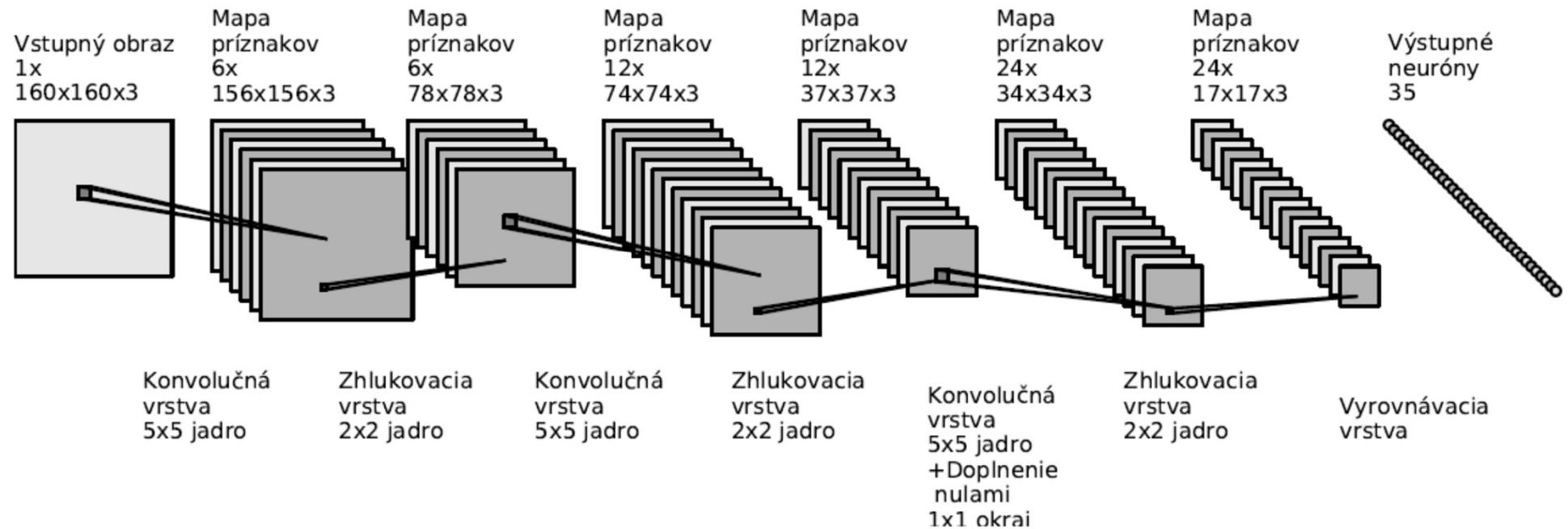
Testované databázy – vlastné ASL gestá

Vlastná databáza 35 gest z americkej znakovnej reči (ASL), databáza je vytvorená od 65 osôb, na jedno gesto 5 snímok, spolu 65×525 , t.j. 34125 snímok.

Vyrezaný (segmentovaný) obraz gesta pre RGB dátá je rozmerov 220x220 pixelov.



Testované architektúry CNN

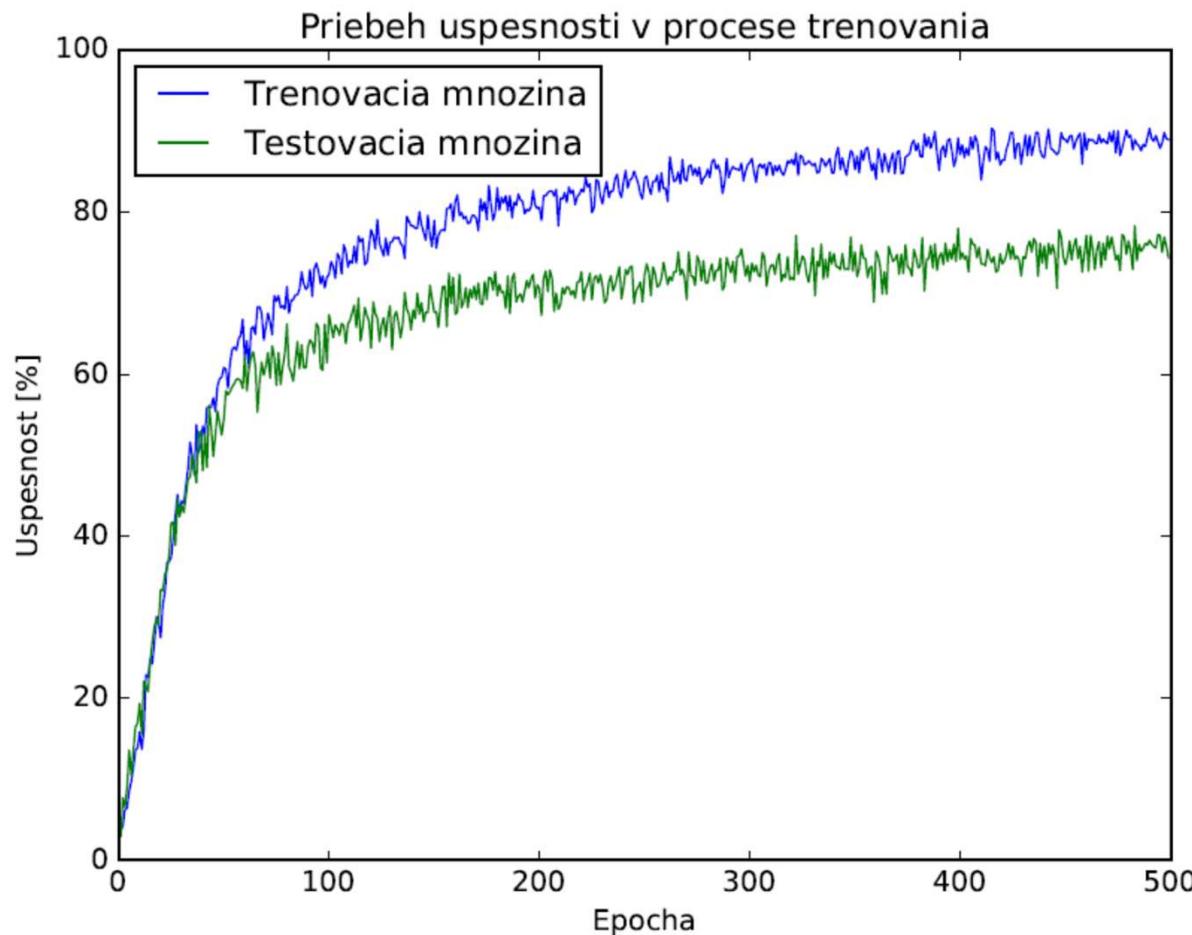


Trénovanie CNN

- ❖ Trénovanie CNN bolo naprogramované v jazyku Python využitím frameworkov TensorFlow, Keras.
 - ❖ Na učenie neurónovej siete bola použitá metóda Adam (adaptive moment estimation).
 - ❖ Učenie CNN bolo realizované na grafickej karte NVIDIA GeForce GTX 960.

The screenshot shows the Spyder Python IDE interface. The top menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Tools, View, Help, and a file path /root. The left sidebar contains various icons for file types like Python, JSON, and CSV. The main area has three panes: 1) A code editor with Python code for gesture recognition, including imports, function definitions, and loops. 2) An Object inspector pane with tabs for Source, Console, and Object, showing help for objects and a 'Usage' section. 3) A large central pane for the IPython console, displaying command history and output. The console output shows training epochs for a neural network, with metrics like loss, accuracy, and validation loss/accuracy.

Dosiahnuté výsledky – databáza ASL

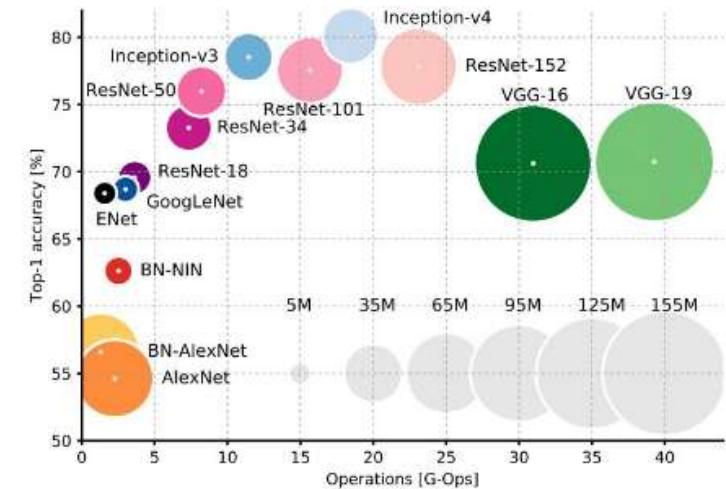
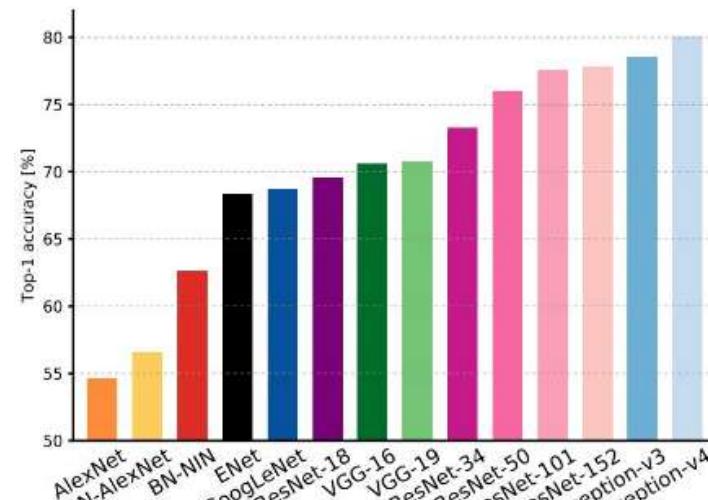


	Priemerná chyba [ACE]		Priemerná úspešnosť [%]	
	Train	Test	Train	Test

	Train	Test	Train	Test
A-2	0.5249	0.4634	80.13	82.44
A-4	0.5134	0.4327	79.72	83.31

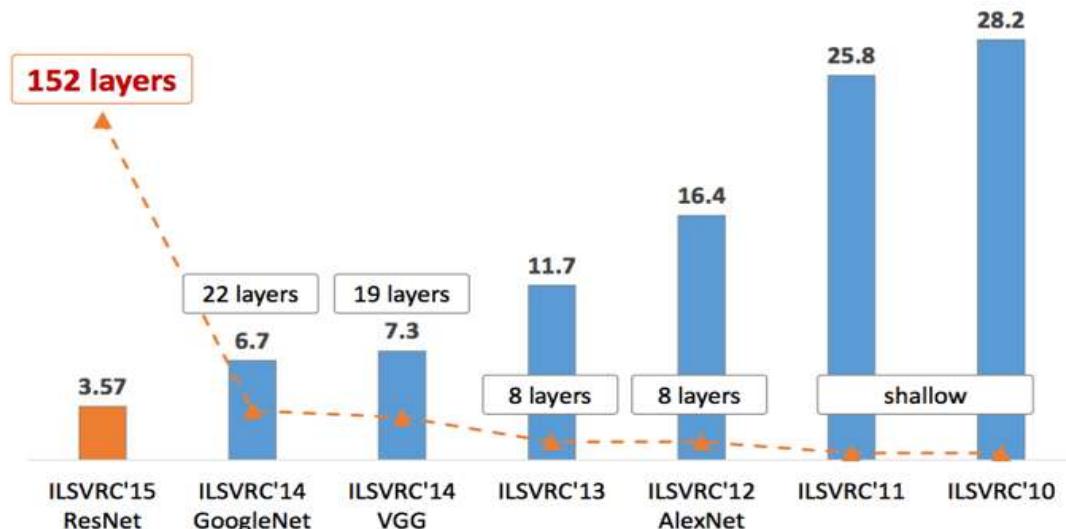
Známe Hlboké Konvolučné neurónové siete

- ❖ LeNet
- ❖ AlexNet
- ❖ VGGnet
- ❖ ResNet
- ❖ GoogLeNet.
- ❖ Inception
- ❖ Mobile Net



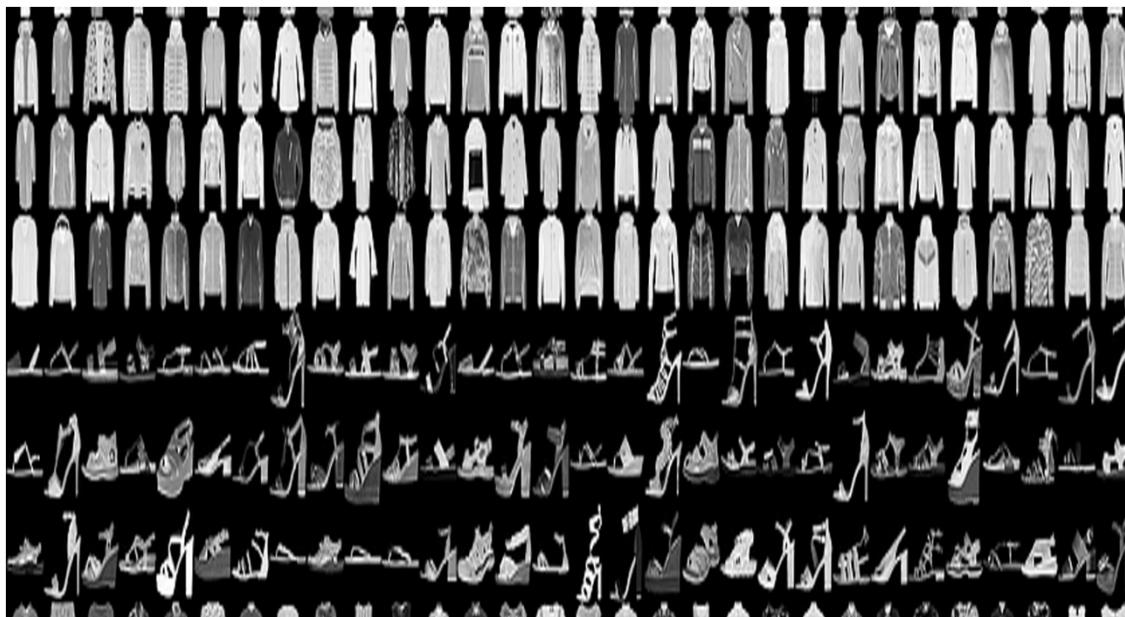
An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Year	CNN	Developed by	Place	Top-5 error rate	No. of parameters
1998	LeNet(8)	Yann LeCun et al			60 thousand
2012	AlexNet(7)	Alex Krizhevsky, Geoffrey Hinton, Ilya Sutskever	1st	15.3%	60 million
2013	ZFNet()	Matthew Zeiler and Rob Fergus	1st	14.8%	
2014	GoogLeNet(19)	Google	1st	6.67%	4 million
2014	VGG Net(16)	Simonyan, Zisserman	2nd	7.3%	138 million
2015	ResNet(152)	Kaiming He	1st	3.6%	



Príklad: FashionMNIST – klasifikácia odevov

FashionMNIST (alternatíva MNIST) je databáza obrazov typov odevov. Tak ako MNIST obsahuje 60 000 trénovacích a 10 000 testovacích dát, obrázky sú veľkosti 28x28 pixelov.



Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

<https://www.kaggle.com/datasets/zalando-research/fashionmnist>

Príklad: CNN štruktúra (2 konvolučné vrstvy)

```
>> layers = [imageInputLayer([28 28 1]); % vstupná vrstva – obraz 28x28x1  
    convolution2dLayer(5,6); % 2D konvolúcia – 6 filtrov, rozmer 5x5  
    batchNormalizationLayer % dávková normalizácia  
    reluLayer(); % relu funkcia  
    maxPooling2dLayer(2,'Stride',2); % max pooling – 2x2, krok 2  
    convolution2dLayer(5,12); % 2D konvolúcia – 12 filtrov, rozmer 5x5  
    batchNormalizationLayer % dávková normalizácia  
    reluLayer(); % relu funkcia  
    maxPooling2dLayer(2,'Stride',2); % max pooling – 2x2, krok 2  
    fullyConnectedLayer(32); % plne prepojená vrstva – 32 neurónov  
    dropoutLayer(0.5); % dropout vrstva  
    fullyConnectedLayer(10); % plne prepojená vrstva – 10 neurónov  
    softmaxLayer(); % softmax aktivačná funkcia  
    classificationLayer()]; % klasifikačná vrstva – 10 tried
```

Príklad: nastavenie parametrov trénovalia

Python ML – klasifikácia FashionMNIST

1. Keras

Klasifikácia (CNN a MLP)

2. PyTorch

Klasifikácia (CNN a MLP)

Keras

https://colab.research.google.com/drive/1BSjDwf_GSLAgTFoRzePJVDbrKEWZ412N

PyTorch

https://colab.research.google.com/drive/1d-2jh1158qSmc01Cd_8aXg229swWe-g