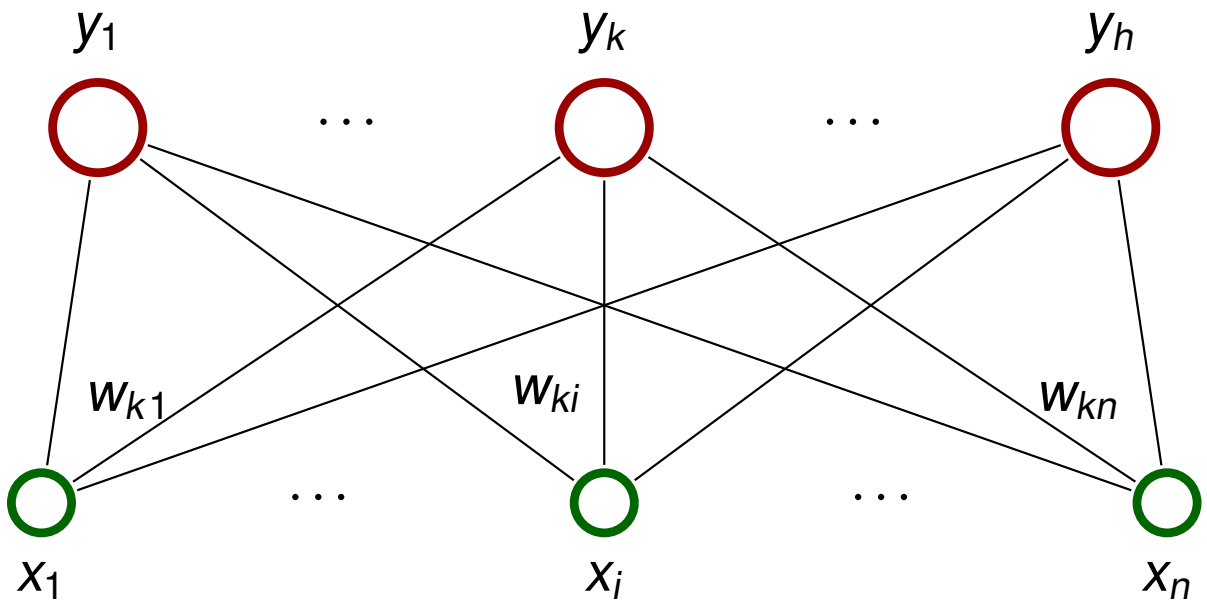
Kohonenova mapa - opakování

Organizační dynamika: Jednovrstvá síť



- Mezi neurony je navíc zavedena **topologická struktura** (tj. neurony tvoří uzly neorientovaného grafu).
- V drtivé většině případů je tato struktura buď jednorozměrná řada jednotek nebo dvojrozměrná mřížka.

Kohonenova mapa

Aktivní dynamika: Pro vstup $\vec{x} \in \mathbb{R}^n$ a $k = 1, \ldots, h$:

$$y_k = \begin{cases} 1 & k = \arg\min_{i=1,\ldots,h} \left\| \vec{x} - \vec{w}_i \right\| \\ 0 & \text{jinak} \end{cases}$$

Adaptivní dynamika: V adaptivním režimu využijeme topologickou strukturu.

- Označme $d(c, k)$ délku nejkratší cesty z neuronu c do neuronu k v topologické struktuře.
- Pro neuron c a dané $s \in \mathbb{N}_0$ definujeme **okolí** neuronu c velikosti s takto: $N_s(c) = \{k \mid d(c, k) \leq s\}$

V kroku t po předložení vstupu $\vec{x}_t$ adaptujeme každé $\vec{w}_k$ takto:

$$\vec{w}_k^{(t)} = \begin{cases} \vec{w}_k^{(t-1)} + \theta \cdot \left(\vec{x}_t - \vec{w}_k^{(t-1)} \right) & k \in N_s(c) \\ \vec{w}_k^{(t-1)} & \text{jinak} \end{cases}$$

kde $c = \arg\min_{i=1,\ldots,h} \left\| \vec{x}_t - \vec{w}_i^{(t-1)} \right\|$ a kde $\theta \in \mathbb{R}$ a $s \in \mathbb{N}_0$ jsou parametry, které se mohou v průběhu učení měnit.

Kohonenova mapa - adaptivní dynamika

Obecnější formulace adaptivní dynamiky:

$$\vec{w}_k^{(t)} = \vec{w}_k^{(t-1)} + \Theta(c,k) \cdot \left(\vec{x} - \vec{w}_k^{(t-1)}\right)$$

kde $c = \arg\min_{i=1,\ldots,h} \left\|\vec{x}_t - \vec{w}_i^{(t-1)}\right\|$. Předchozí případ potom odpovídá

$$\Theta(c,k) = \begin{cases} \theta & k \in N_s(c) \\ 0 & \text{jinak} \end{cases}$$

Obvykle se používá plynulejší přechod mezi nenulovými a nulovými hodnotami, např.

$$\Theta(c,k) = \theta_0 \cdot \exp\left(\frac{-d(c,k)^2}{\sigma^2}\right)$$

kde $\theta_0 \in \mathbb{R}$ určuje maximální míru změny vah a $\sigma \in \mathbb{R}$ je šířka (oba parametry se mohou v průběhu měnit).

LVQ - klasifikace - opakování

Přepodkládejme, že máme náhodně generované vzory tvaru $(\vec{x}_t, d_t)$ kde $\vec{x}_t \in \mathbb{R}^n$ je **vektor vlastností** a $d_t \in \{C_1, \ldots, C_q\}$ určuje jednu z q **tříd**.

Cílem je klasifikovat vstupy do tříd na základě znalosti vlastností, tj. každému $\vec{x}_t$ chceme přiřadit třídu tak, abychom minimalizovali pravděpodobnost chyby.

Př.: Po pásu jede náhodně „naházené“ ovoce dvou druhů: jablka a meruňky. Námi sledovaná data budou $(\vec{x}_t, d_t)$ kde

- $\vec{x}_t \in \mathbb{R}^2$, první vlastnost je hmotnost, druhá je průměr
- d_t je buď J nebo M v závislosti na tom, jestli je daný kus jablko nebo meruňka

Připouštíme možnost, že se najde jablko a meloun se stejnými mírami.

Cílem je třídit ovoce na základě hmotnosti a průměru.

LVQ - klasifikace - opakování

Využijeme vektorovou kvantizaci (Kohonenovu mapu) takto:

1. Natrénujeme mapu na vektorech vlastností $\vec{x}_t$ kde $t = 1, \ldots, \ell$.
2. Jednotlivé neurony označíme třídami. Třídu v_c neuronu c nalezneme takto:

   Pro každý neuron c a třídu C_i spočítáme četnost vzorů třídy C_i, které jsou reprezentovány neuronem c.

   Toto lze provést jedním průchodem přes vzory

   Neuronu c přiřadíme třídu s maximální četností.
3. Doladíme síť pomocí algoritmu LVQ.

Klasifikaci pomocí natrénované sítě provádíme takto: Na vektor vlastností $\vec{x}$ aplikujeme natrénovanou síť v aktivním režimu. Právě jeden neuron, řekněme c, bude mít výstup 1. Potom vstup zařadíme do třídy v_c.

LVQ1 - opakování

Postupně projdi tréninkové vzory. Pro vzor $(\vec{x}_t, d_t)$ urči nejbližší neuron c

$$c = \arg \min_{i=1,\ldots,h} \left\| \vec{x}_t - \vec{w}_i \right\|$$

Potom uprav váhy neuronu c takto:

$$\vec{w}_c^{(t)} = \begin{cases} \vec{w}_c^{(t-1)} + \alpha(\vec{x}_t - \vec{w}_c^{(t-1)}) & d_t = v_c \\ \vec{w}_c^{(t-1)} - \alpha(\vec{x}_t - \vec{w}_c^{(t-1)}) & d_t \neq v_c \end{cases}$$

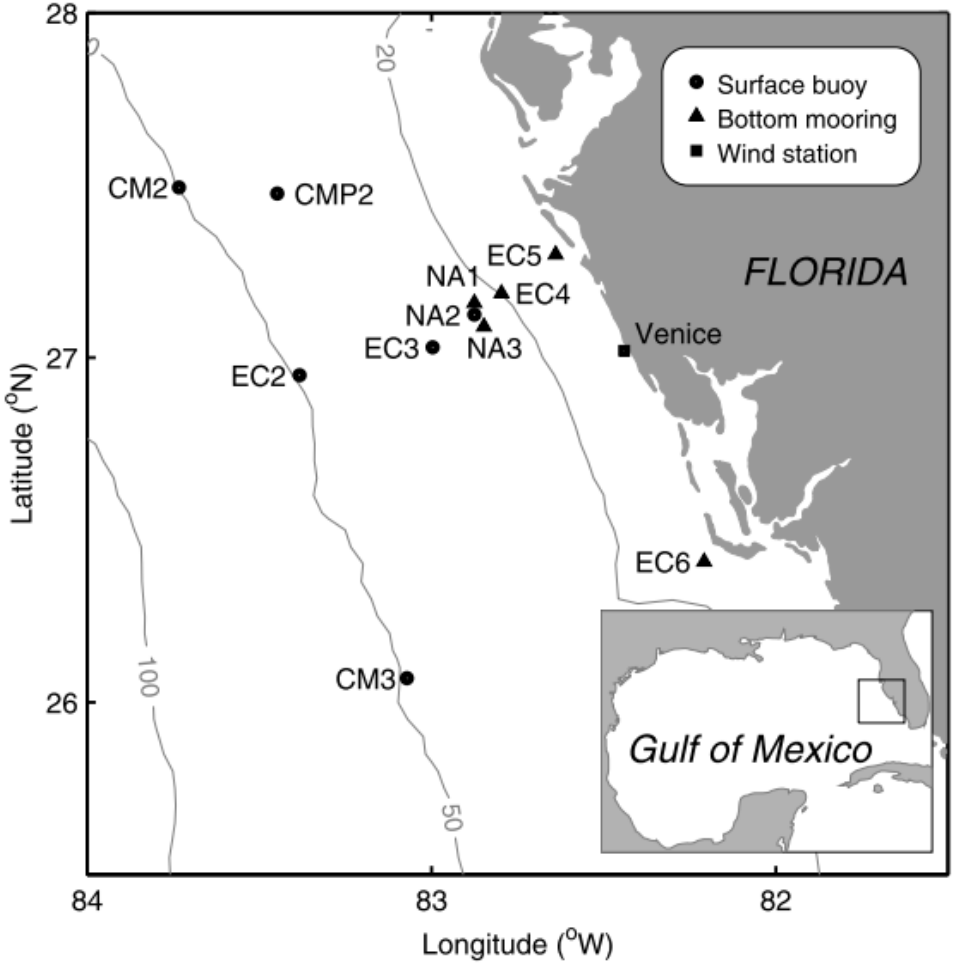
Parametr α by měl být od počátku malý (cca 0.01 – 0.02) a postupně klesnout k 0.

Hranice mezi třídami vytvořená pomocí LVQ1 je poměrně dobrou aproximací *bayesovské rozhodovací hranice*.

Oceánografická data

Zdroj: Patterns of ocean current variability on the West Florida Shelf using the self-organizing map. Y. Liu a R. H. Weisberg, JOURNAL OF GEOPHYSICAL RESEARCH, 2005

Zkoumá se vývoj proudění vod v oceánu kolem pobřeží Floridy



Oceánografická data

- 11 měřicích stanic, 3 hloubky (hladina, dno, mezi)
- data: 2D vektory rychlosti (a směru) proudění
- měřeno po hodinách, 25585 hodin

Celkově tedy 25585 řádků 66 dimenzionálních položek.

Kohonenova mapa:

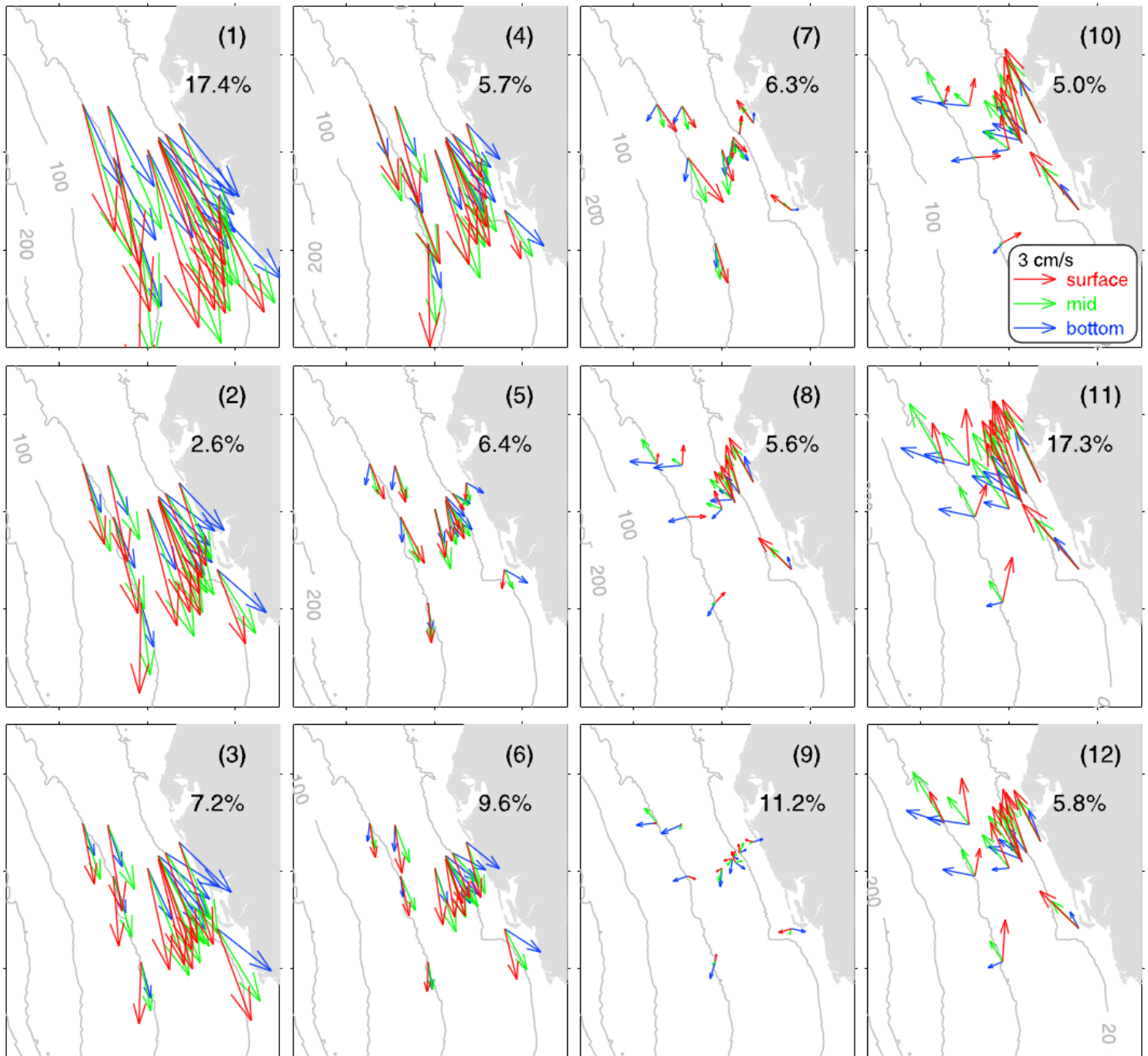
- mřížka 3×4
- okolí dána Gaussovou funkcí

$$\Theta(c,k) = \theta_0 \cdot \exp\left(\frac{-d(c,k)^2}{\sigma^2}\right)$$

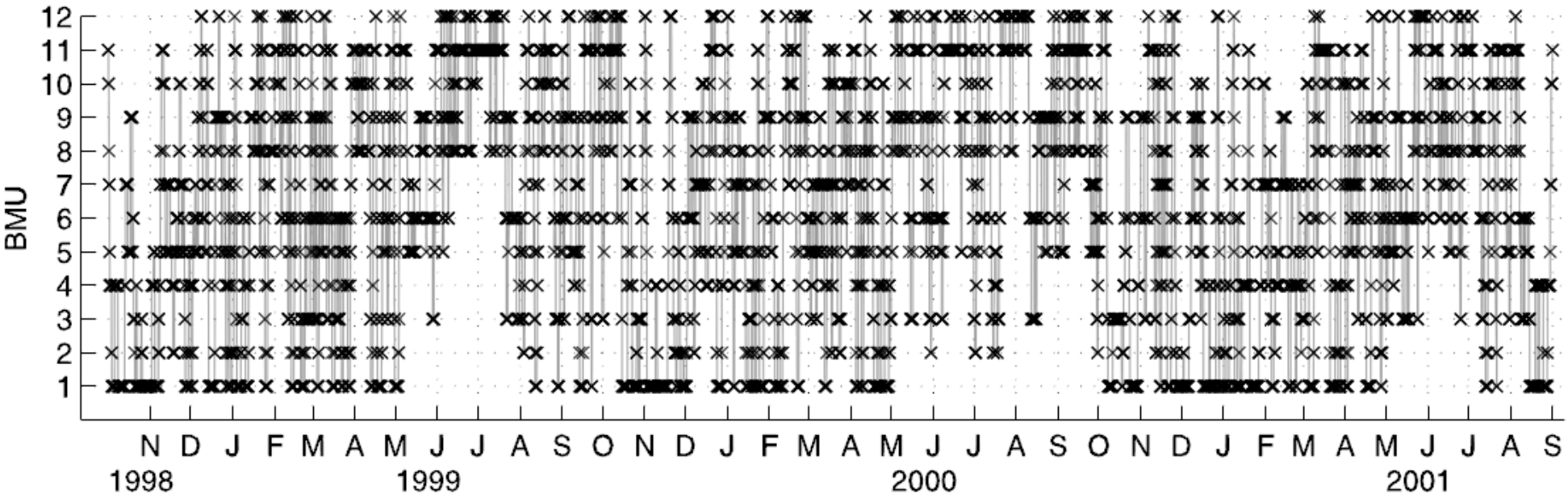
se zmenšující se šířkou

(navíc je tam lineárně se zmenšující rychlost učení, kterou se násobí změna polohy neuronů)

Oceánografická data - mapa

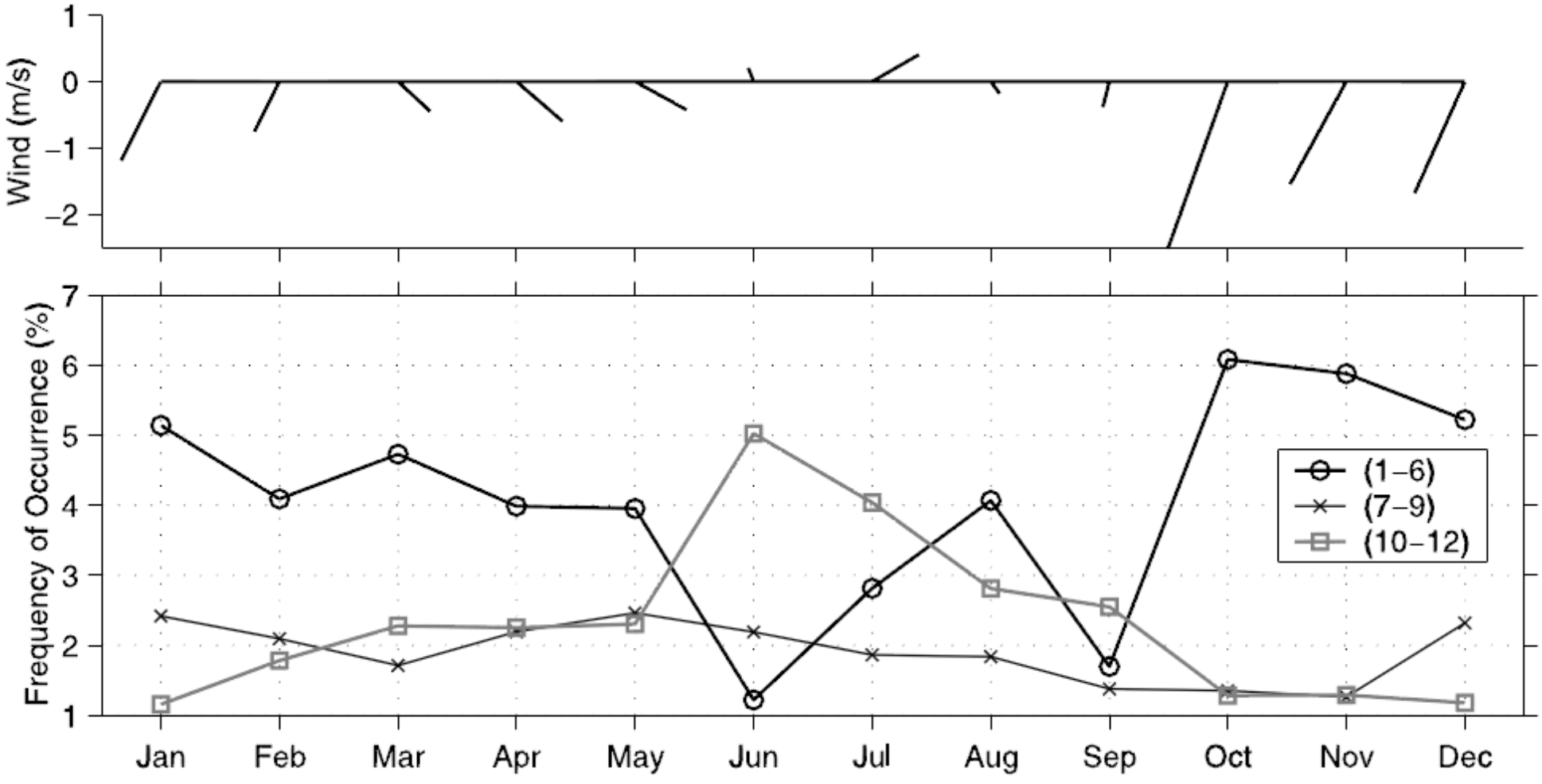


Oceánografická data



- křížky označují „vítězné“ neurony (po hodinách)
- ovlivněno lokálními fluktuacemi
- pozorovatelný trend:
  - v zimě neurony 1-6 (jiho-východ)
  - v létě neurony 10-12 (severo-západ)

Oceánografická data



Srovnání směru větru a proudění vody (znatelná korelace)

- vítr: směr čáry = směr větru ; délka čáry = intenzita
- proudění vody v procentech úspěšnosti skupin neuronů

Analýza pohádek bratří Grimmů

Zdroj: Contextual Relations of Words in Grimm Tales, Analyzed by Self-Organizing Map. T. Kohonen, T. Honkela a V. Pulkki, ICANN, 1995

Cílem je vizualizovat syntaktické a sémantické kategorie slov v pohádkách (v závislosti na kontextu).

Vstup: Pohádky bratří Grimmů (srozumitelně kódované pomocí proudu 270-dimenzionálních vektorů)

- uvažují se trojice slov (předchůdce, klíč, následník)
- každá položka v trojici se zakóduje náhodně vybraným 90-dimenzionálním reálným vektorem (trojice má tedy dimenzi 270)

Síť: Kohonenova mapa, 42×36 neuronů, váhy neuronů jsou tvaru $w = (w_p, w_k, w_n)$ kde $w_p, w_k, w_n \in \mathbb{R}^{90}$.

Analýza pohádek bratří Grimmů

Adaptace:

Síť je natrénována na trojicích po sobě jdoucích slov z pohádek

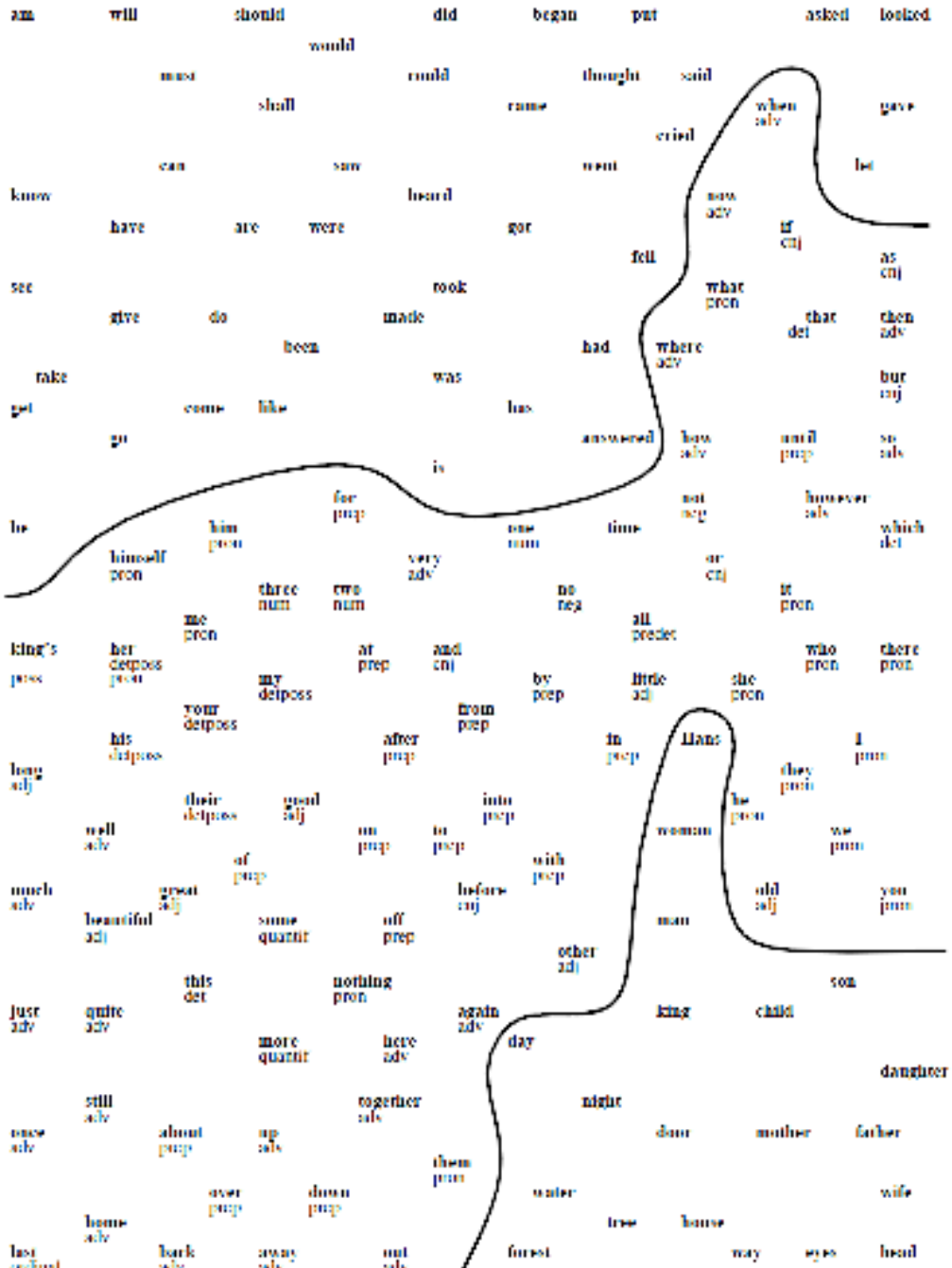
Pozn. Tréninkovou sadu tvořilo 150 nejpoužívanějších slov se “zprůměrovaným” kontextem.

Hrubé učení: 600 000 iterací; Dolaďování: 400 000

Nakonec 150 nejčastěji použitých slov označuje neurony:

slovem u je označen ten neuron, pro jehož váhy $w = (w_p, w_k, w_n)$ platí, že w_k je nejblíže kódu slova u.

Analýza pohádek bratří Grimmů - výsledek

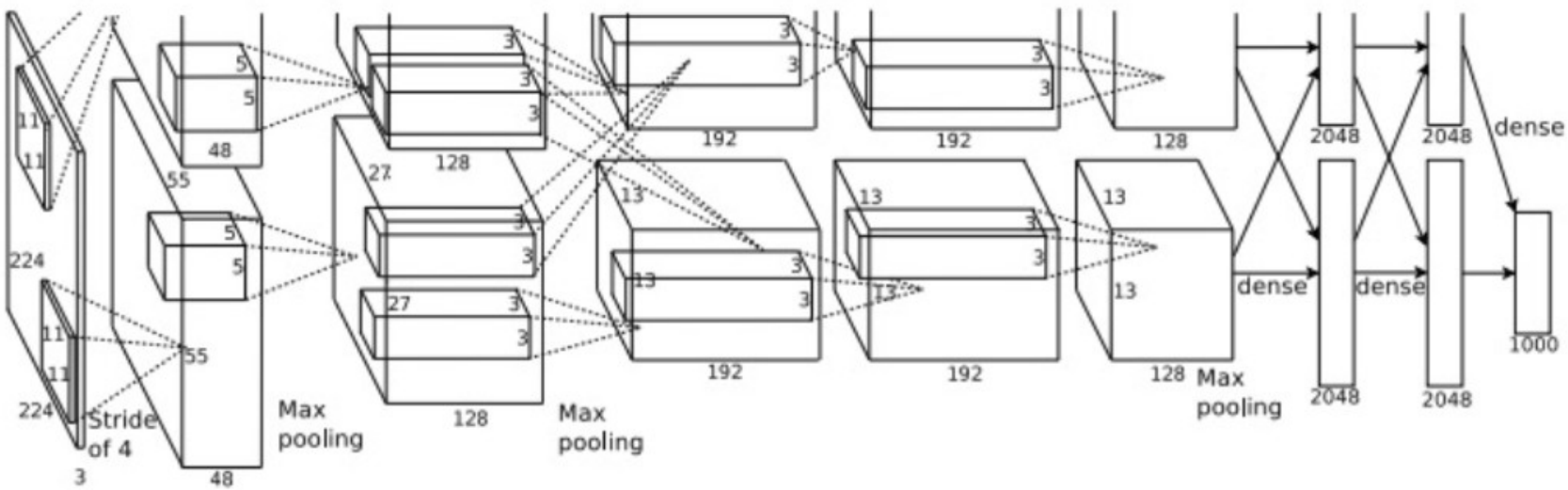


Konvoluční sítě

Zbytek přednášky je založen na nové online knize Neural Networks and Deep Learning, autor Michael Nielsen.
`http://neuralnetworksanddeeplearning.com/index.html`

- Konvoluční sítě jsou v současné době nejlepší metodou pro klasifikaci obrázků z databáze ImageNet.
- Jejich počátky sahají do 90. let – síť LeNet-5

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



Konvoluční sítě vs MNIST (ilustrace)

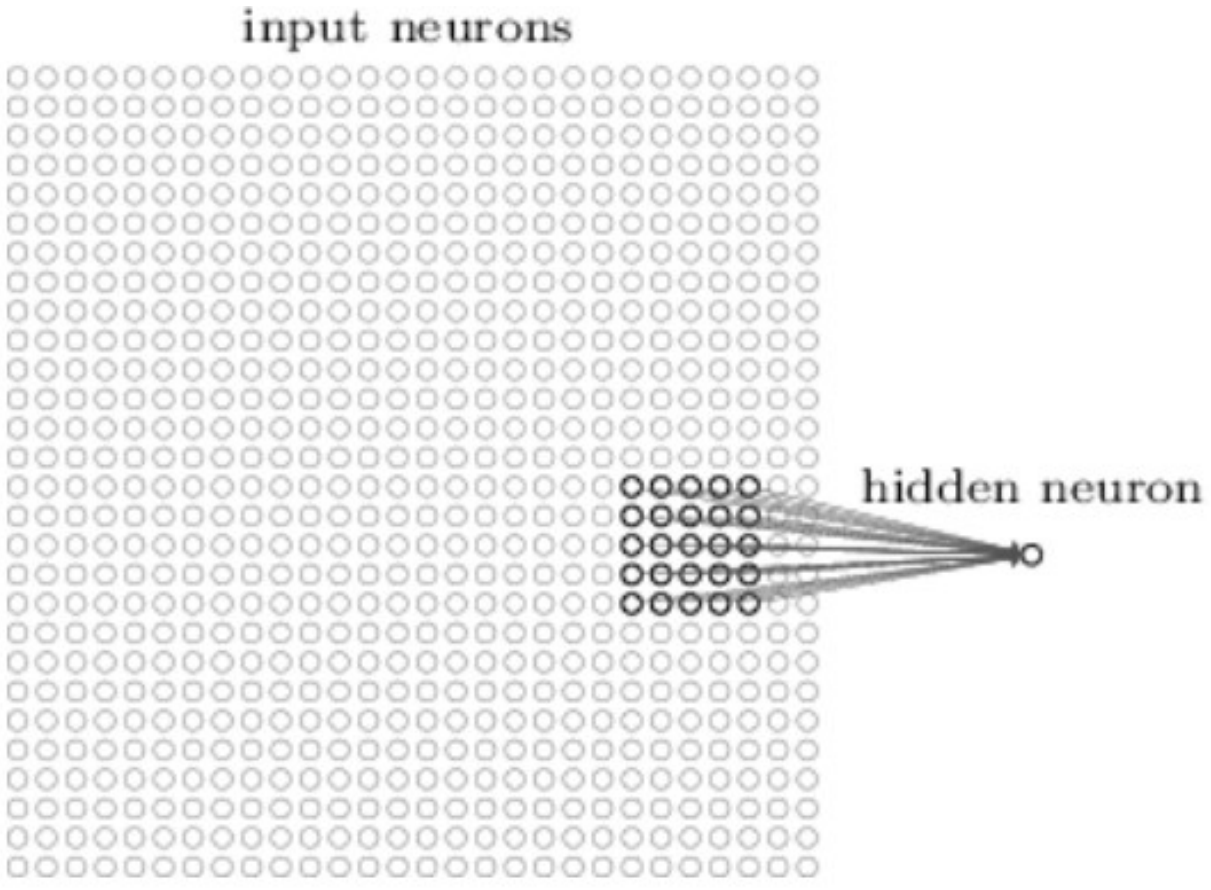
MNIST:

- Databáze (označkovaných) obrázků rukou psaných číslic: 60 000 tréninkových, 10 000 testovacích.
- Dimenze obrázků je 28 x 28 pixelů, jsou vycentrované do "těžiště" jednotlivých pixelů a normalizované na fixní velikost
- Více na `http://yann.lecun.com/exdb/mnist/`

Databáze se používá jako standardní benchmark v mnoha publikacích o rozpoznávání vzorů (nejen) pomocí neuronových sítí.

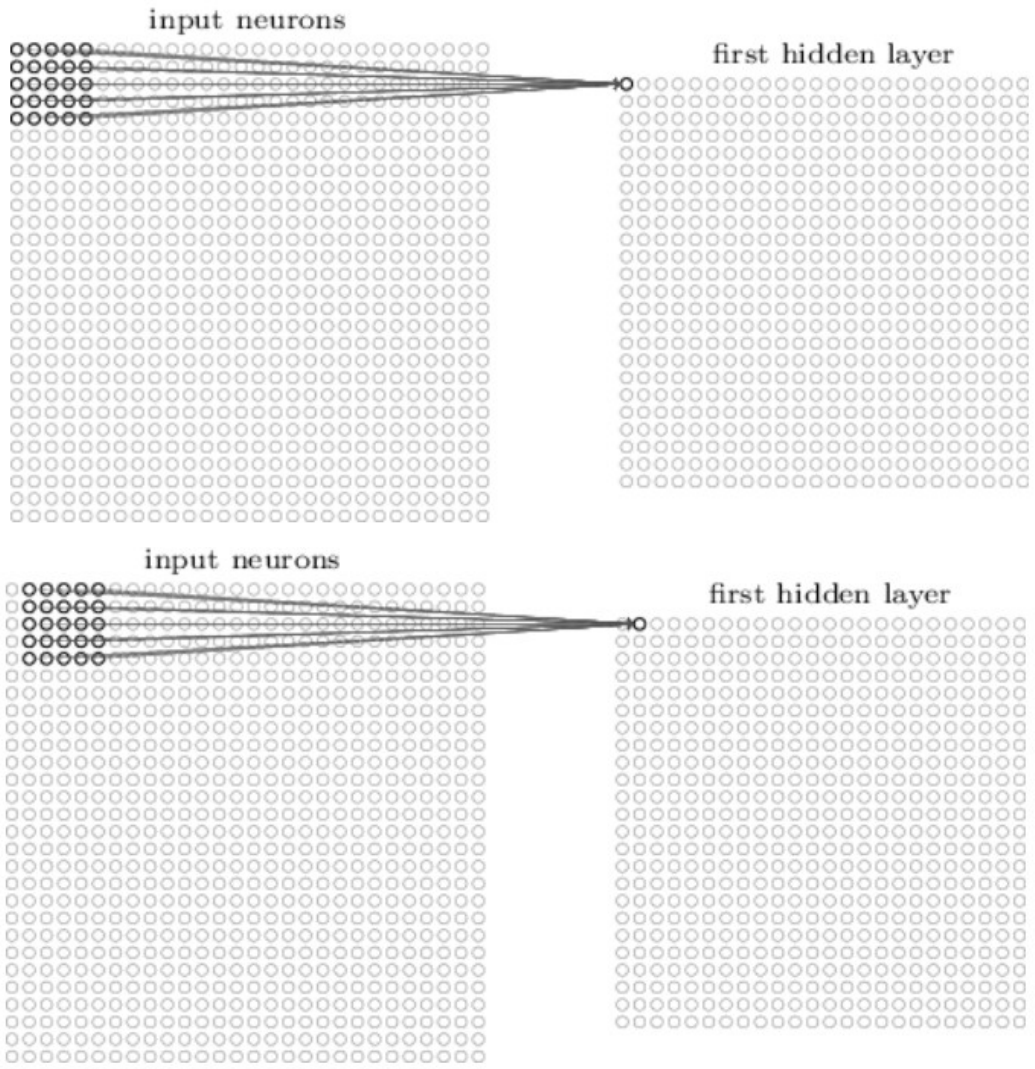
Lze porovnávat přesnost klasifikace různých metod.

Konvoluční sítě - local receptive fields



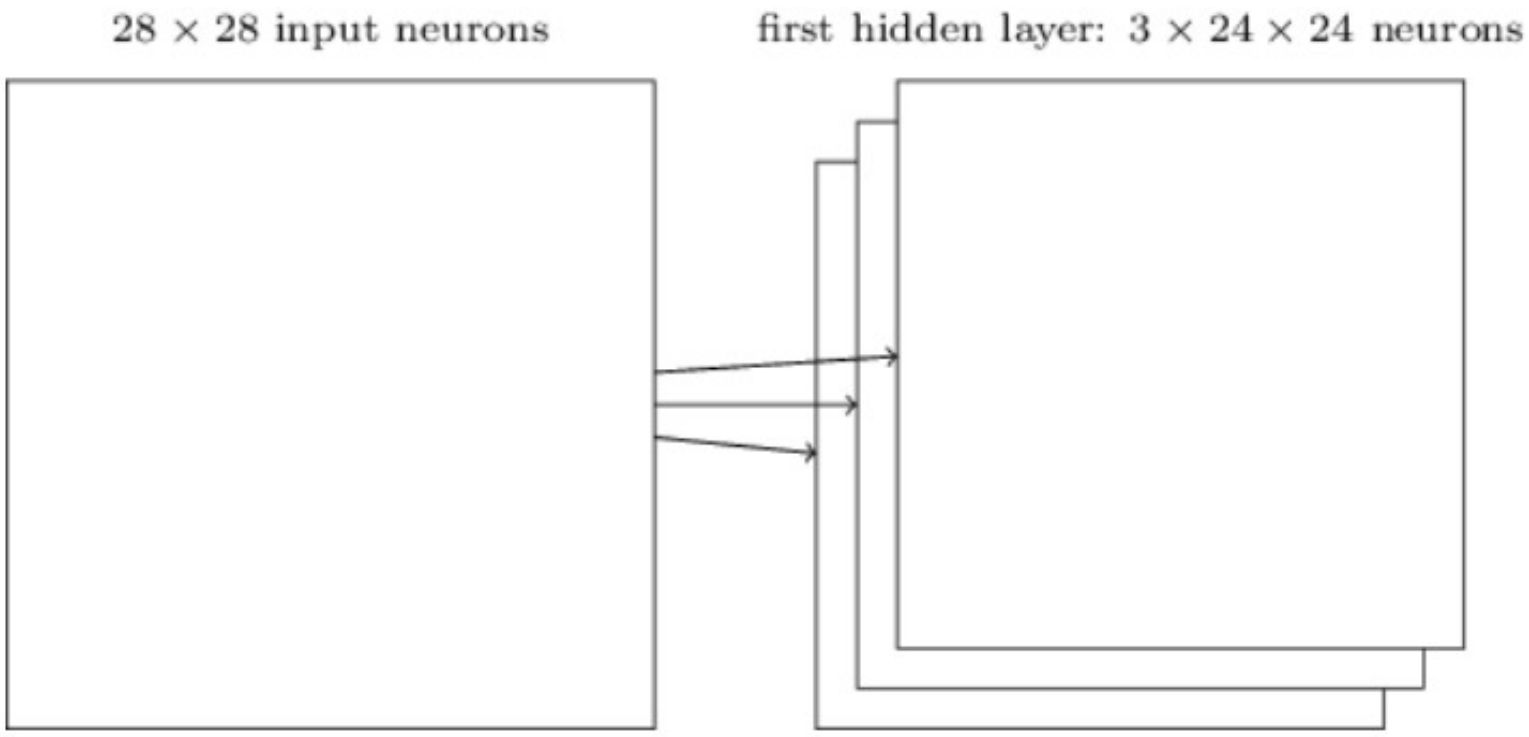
Každý neuron je spojen s polem 5×5 neuronů v nižší vrstvě (toto pole se nazývá *local receptive field*)
Neuron je "standardní": Počítá vážený součet vstupů, na výsledek aplikuje aktivační funkci.

Konvoluční sítě - stride length



Velikost posun vedle sebe ležících polí je *stride length*.
Celá skupina neuronů (tj. všechny polohy) je *feature map*.
Všechny neurony z dané feature map sdílí váhy a bias!

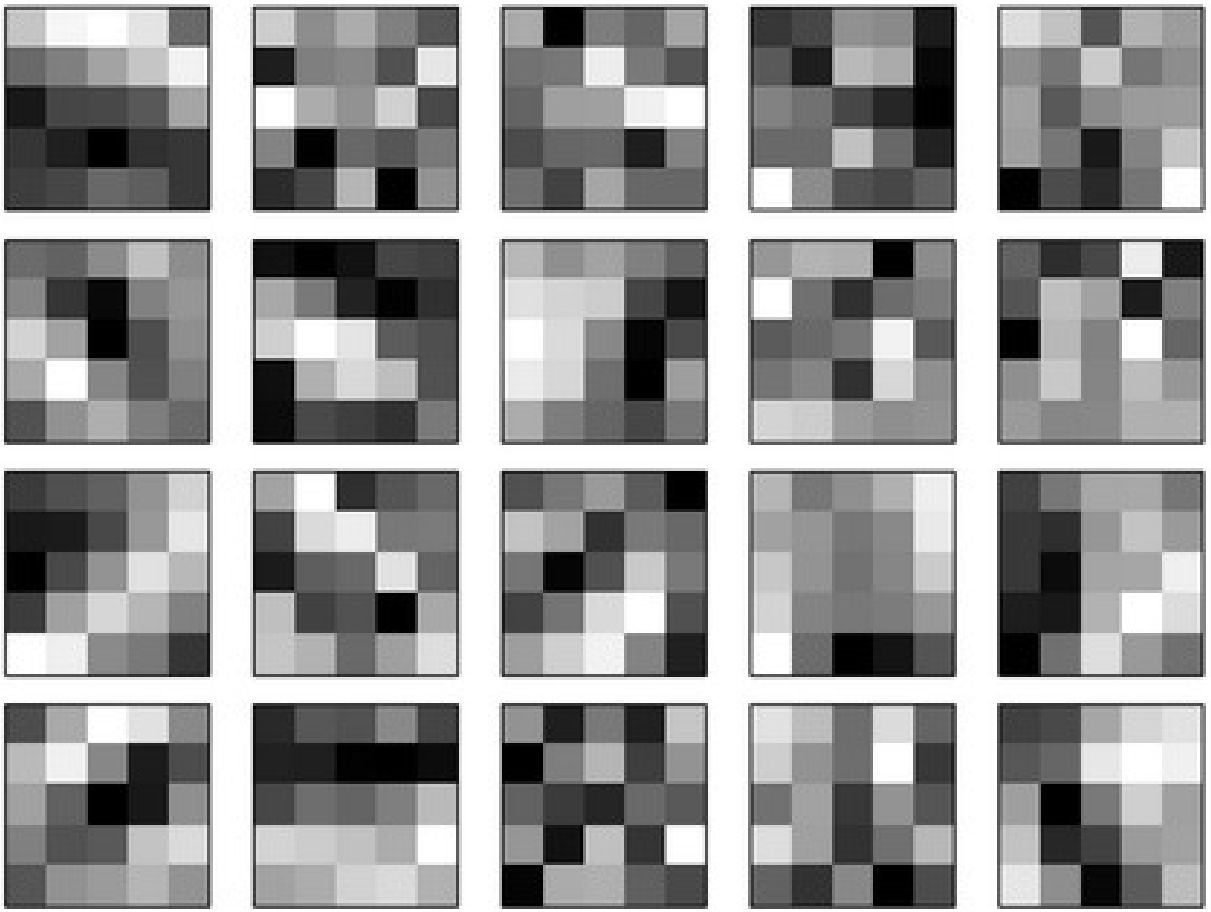
Feature maps



Každá feature map "reprezentuje" nějakou vlastnost vstupu.
(v libovolné poloze ve vstupu)

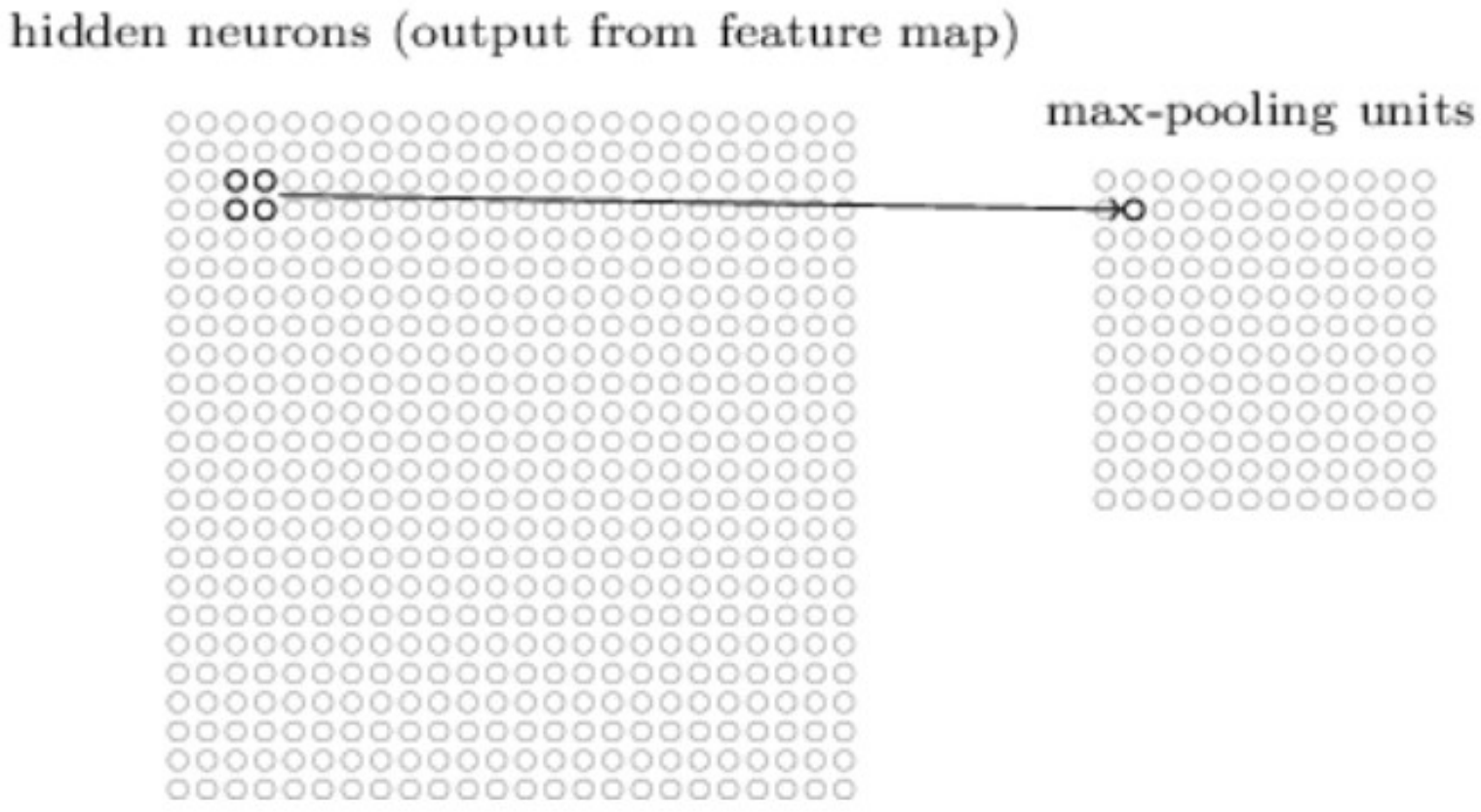
Typicky se uvažuje několik feature maps v jedné vrstvě.

Natrénované feature maps



(zde 20 feature maps, receptive fields 5×5)

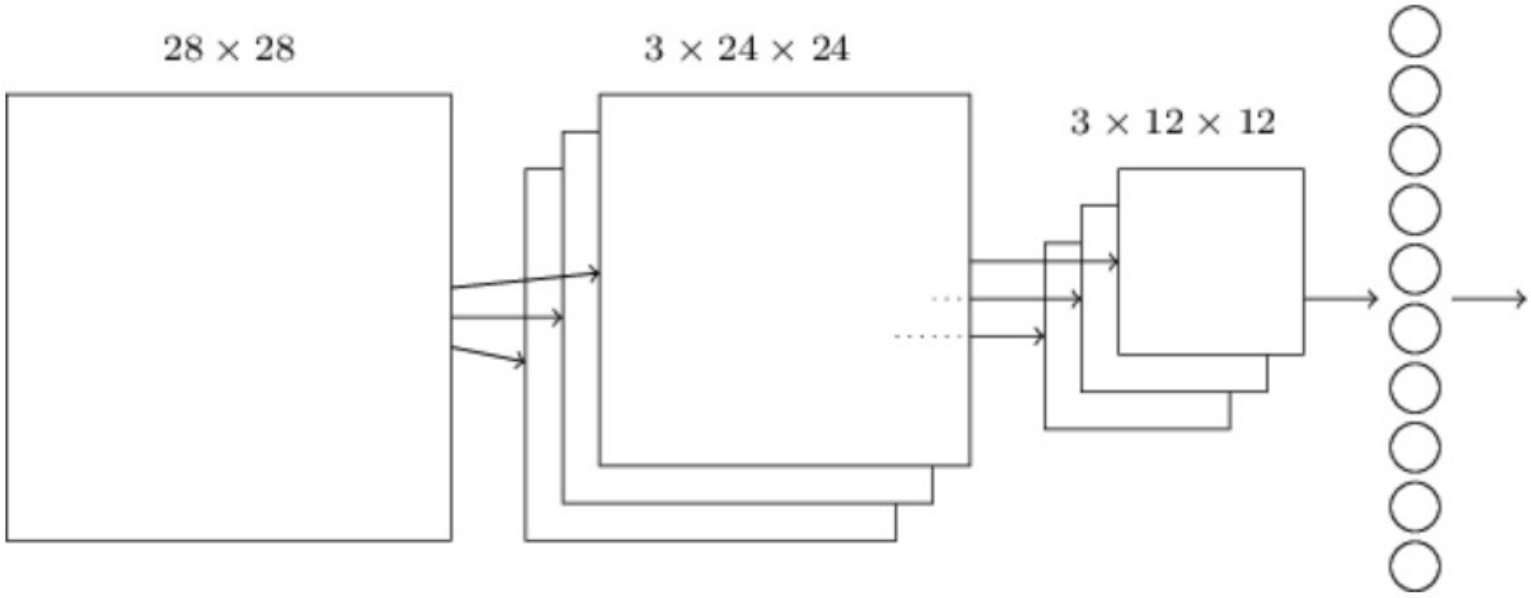
Pooling



Neuron v pooling vrstvě počítá funkci svého pole:

- **Max-pooling** : maximum hodnot neuronů
- **L2-pooling** : odmocnina ze součtu druhých mocnin hodnot neuronů
- **Average-pooling** : aritmetický průměr
- ...

Jednoduchá konvoluční síť

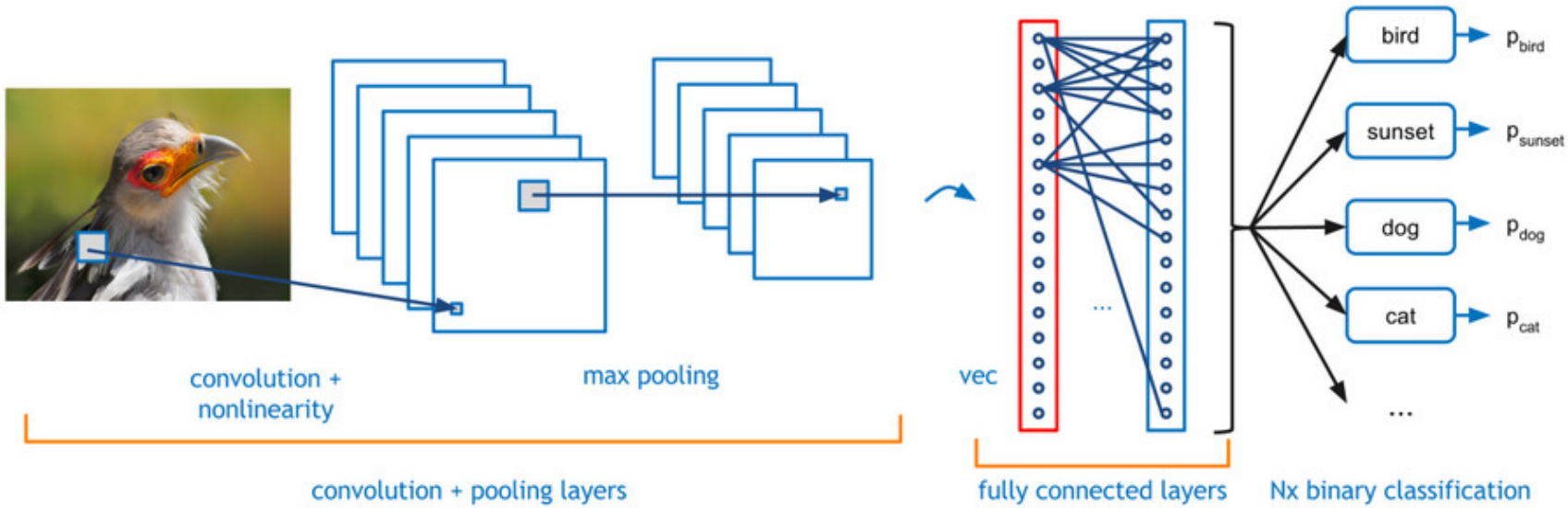


28×28 obraz na vstupu, 3 feature maps, každá má svůj max-pooling (pole 5×5, stride = 1), 10 výstupních neuronů.

Každý neuron ve výstupní vrstvě bere vstup z každého neuronu v pooling layer.

Trénuje se typicky zpětnou propagací, kterou lze snadno upravit pro konvoluční síť (přepočítají se derivace).

Konvoluční síť



Jednoduchá konvoluční síť vs MNIST

dvě konvoluční-pooling vrstvy, jedna 20, druhá 40 feature maps, dvě úplně propojené vrstvy (1000-1000), výtupní vrstva (10)

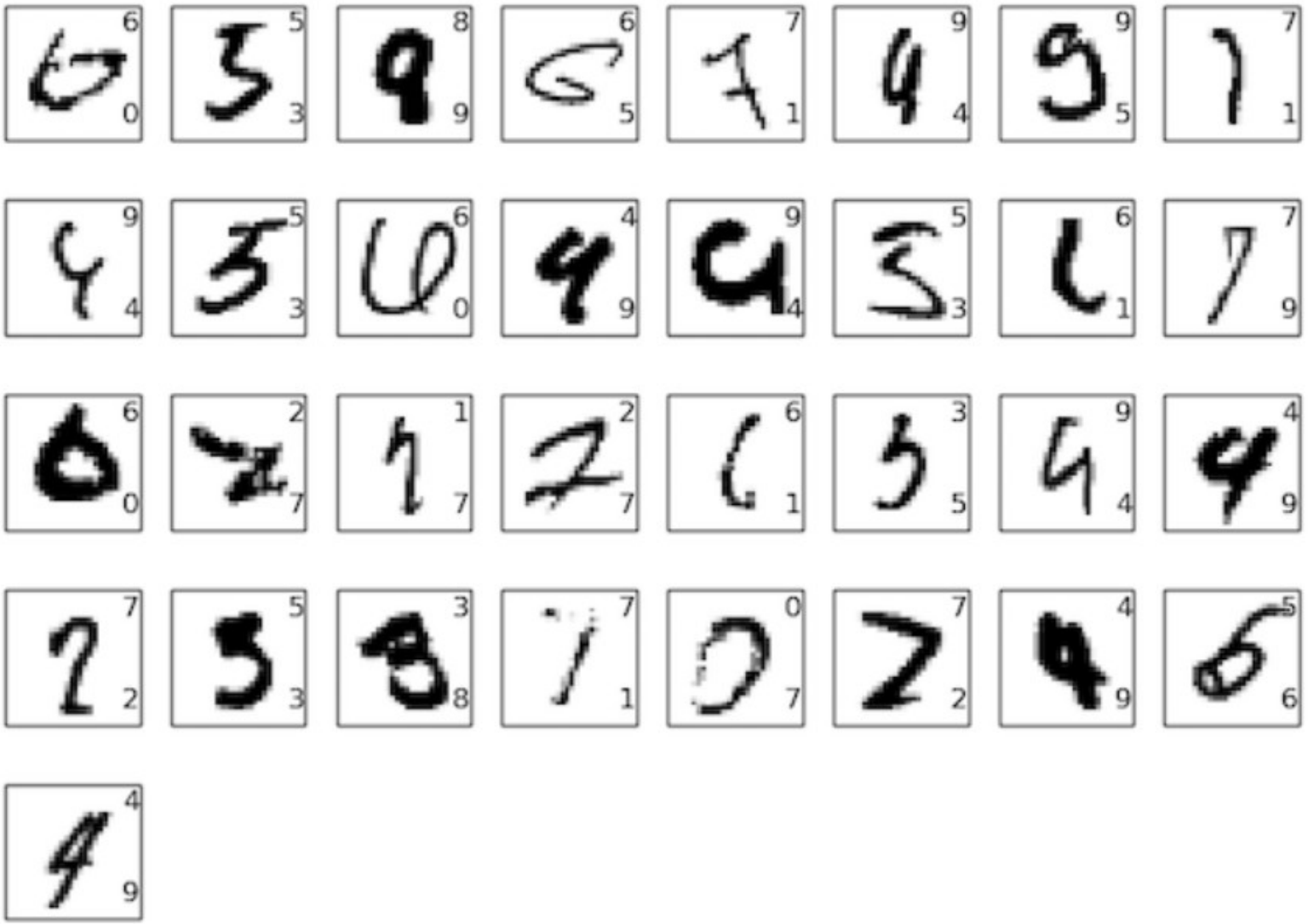
- Aktivační funkce feature maps a plně propojených: ReLU
- max-pooling
- výstupní vrstva: soft-max
- Chybová fce: negative log-likelihood
- V podstatě stochastický gradientní sestup (ve skutečnosti mini-batch velikosti 10)
- rychlost učení 0.03
- L2 regularizace s "váhou"$\lambda = 0.1$ + dropout s pravd. $1/2$
- trénink 40 epoch (tj. vše se projde 40 krát)
- Datová sada upravena posouváním o jeden pixel do libovolného směru.
- Nakonec použito hlasování pěti sítí.

Konvoluční síť - Theano

```
>>> net = Network([
        ConvPoolLayer(image_shape=(mini_batch_size, 1, 28, 28),
                      filter_shape=(20, 1, 5, 5),
                      poolsize=(2, 2),
                      activation_fn=ReLU),
        ConvPoolLayer(image_shape=(mini_batch_size, 20, 12, 12),
                      filter_shape=(40, 20, 5, 5),
                      poolsize=(2, 2),
                      activation_fn=ReLU),
        FullyConnectedLayer(
            n_in=40*4*4, n_out=1000, activation_fn=ReLU, p_dropout=0.5),
        FullyConnectedLayer(
            n_in=1000, n_out=1000, activation_fn=ReLU, p_dropout=0.5),
        SoftmaxLayer(n_in=1000, n_out=10, p_dropout=0.5)],
        mini_batch_size)
>>> net.SGD(expanded_training_data, 40, mini_batch_size, 0.03,
            validation_data, test_data)
```

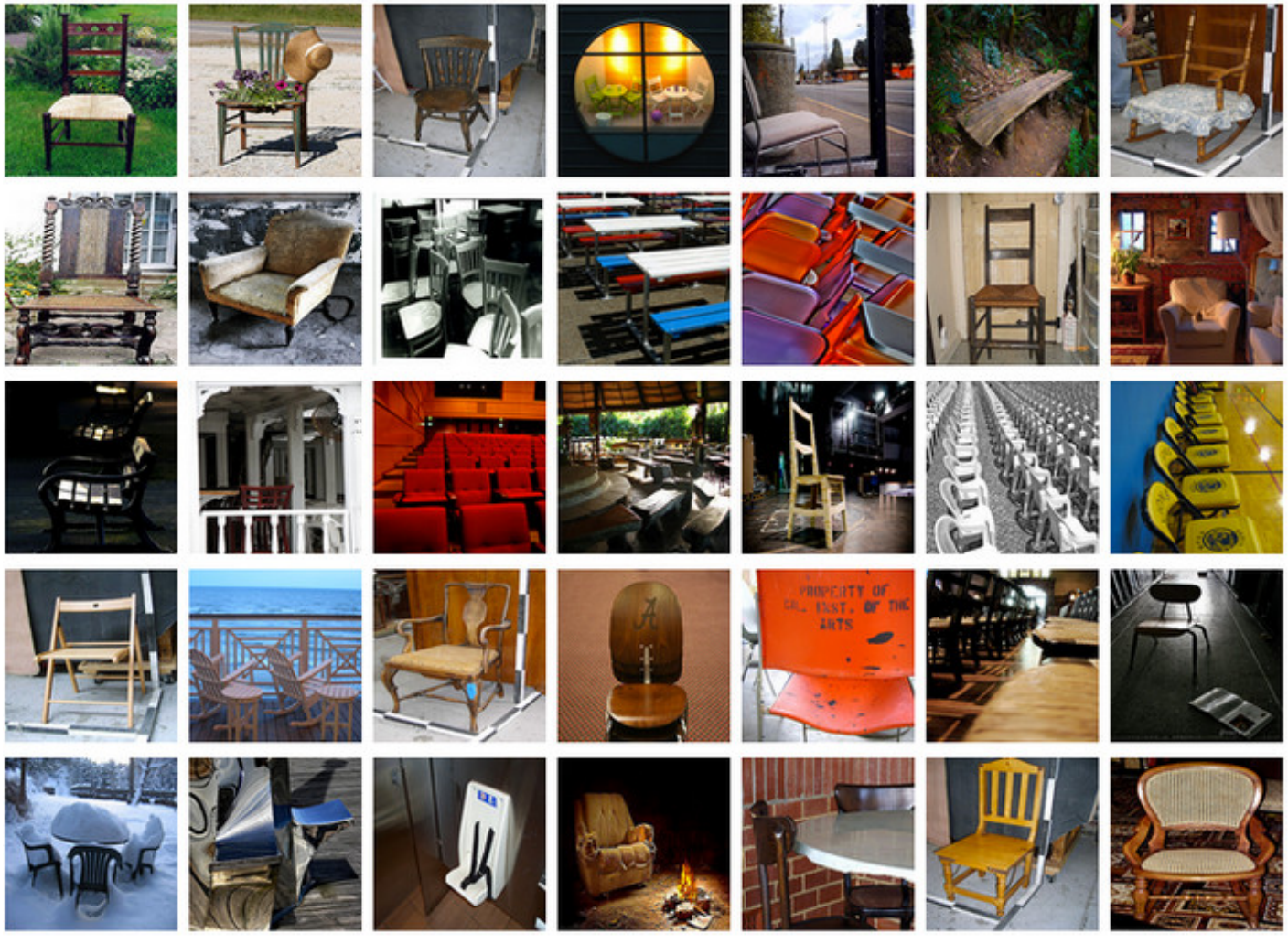
MNIST

Z 10 000 obrázků databáze MNIST, pouze těchto 33 bylo špatně klasifikováno:



Složitější konvoluční síť

Konvoluční sítě byly použity pro klasifikaci obrázků z databáze ImageNet (16 milionů barevných obrázků, 20 tisíc kategorií)



ImageNet Large-Scale Visual Recognition Challenge (ILSVRC)

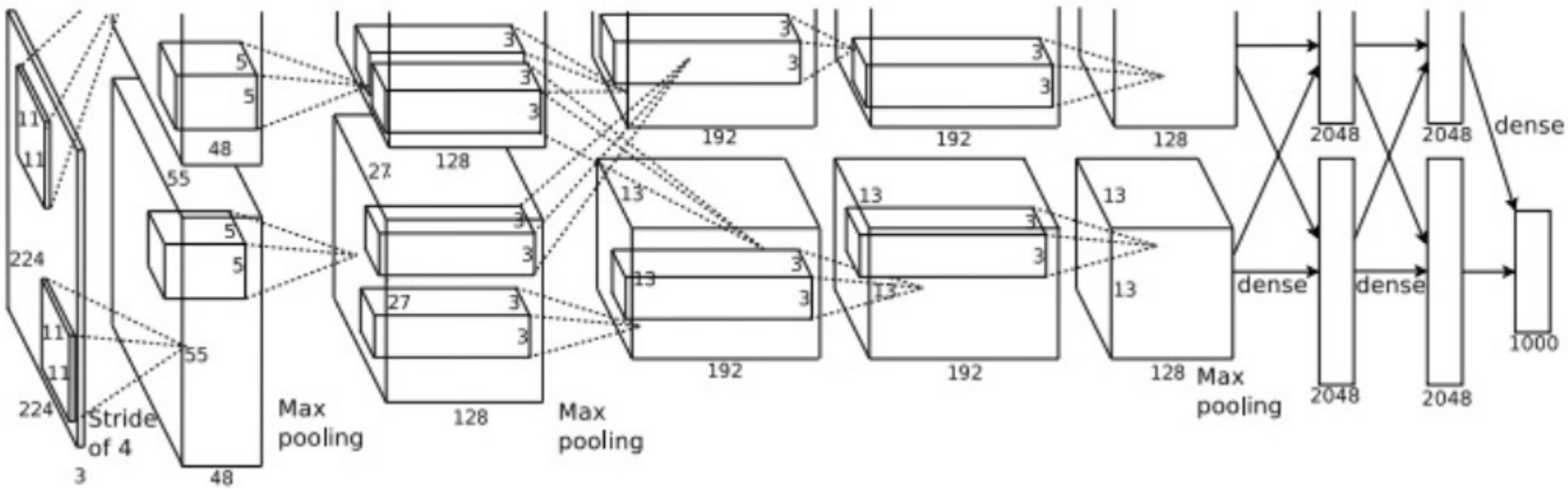
Soutěž v klasifikaci nad podmnožinou obrázků z ImageNet.

V roce 2012: tréninková množina 1.2 milionů obrázků z 1000 kategorií. Validační množina 50 000, testovací 150 000.

Mnoho obrázků obsahuje více objektů → za správnou odpověď se typicky považuje, když je správná kategorie mezi pěti, jimž síť přiřadí nejvyšší pravděpodobnost (top-5 kritérium).

KSH síť

ImageNet classification with deep convolutional neural networks, by Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton (2012).



Trénováno na dvou GPUs (NVIDIA GeForce GTX 580)

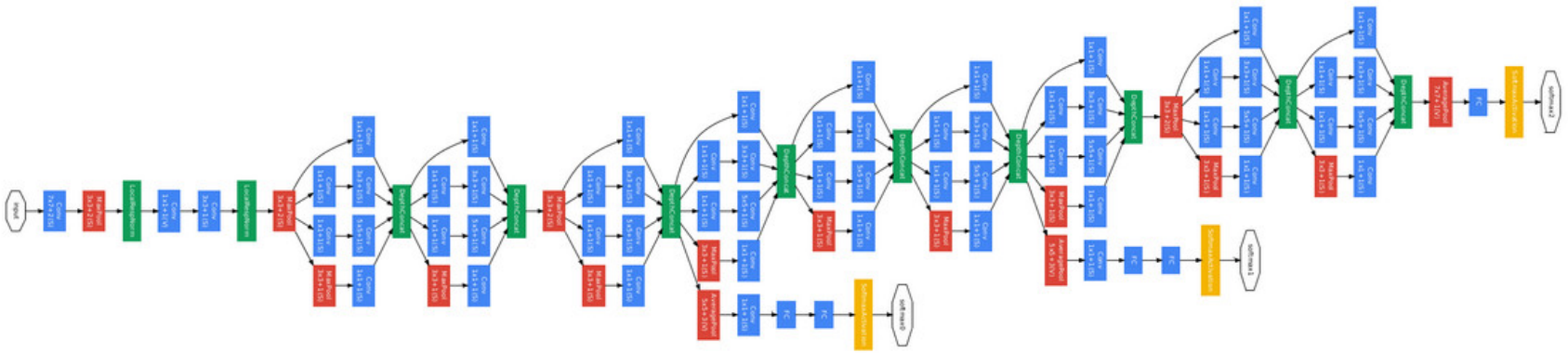
Výsledky:

- úspěšnost 84.7 procenta v top-5 (druhý nejlepší alg. 73.8 procent)
- 63.3 procenta v "absolutní" klasifikaci

ILSVRC 2014

Stejná sada jako v roce 2012, top-5 kritérium.

Síť GoogLeNet: hluboká konvoluční síť, 22 vrstev



Výsledky:

- 93.33 procent top-5

Jsou lidé stále schopni obstát proti tomuhle?

Superhuman GoogLeNet?!

Andrej Karpathy: ...the task of labeling images with 5 out of 1000 categories quickly turned out to be extremely challenging, even for some friends in the lab who have been working on ILSVRC and its classes for a while. First we thought we would put it up on [Amazon Mechanical Turk]. Then we thought we could recruit paid undergrads. Then I organized a labeling party of intense labeling effort only among the (expert labelers) in our lab. Then I developed a modified interface that used GoogLeNet predictions to prune the number of categories from 1000 to only about 100. It was still too hard - people kept missing categories and getting up to ranges of 13-15% error rates. In the end I realized that to get anywhere competitively close to GoogLeNet, it was most efficient if I sat down and went through the painfully long training process and the subsequent careful annotation process myself... The labeling happened at a rate of about 1 per minute, but this decreased over time... Some images are easily recognized, while some images (such as those of fine-grained breeds of dogs, birds, or monkeys) can require multiple minutes of concentrated effort. I became very good at identifying breeds of dogs... Based on the sample of images I worked on, the GoogLeNet classification error turned out to be 6.8%... My own error in the end turned out to be 5.1%, approximately 1.7% better.