

Rozpoznávání snímků, detekce objektů, moderní trendy

Strojové vidění a zpracování obrazu (BI-SVZ)

Úlohy v oboru počítačového vidění

- Klasifikace obrázků
- Lokalizace objektů
- Detekce objektů
- Sémantická segmentace
- Segmentace instance
- Textový popis obrázků

- Stovky dalších...

Nejčastější úlohy v počítačovém vidění

Classification



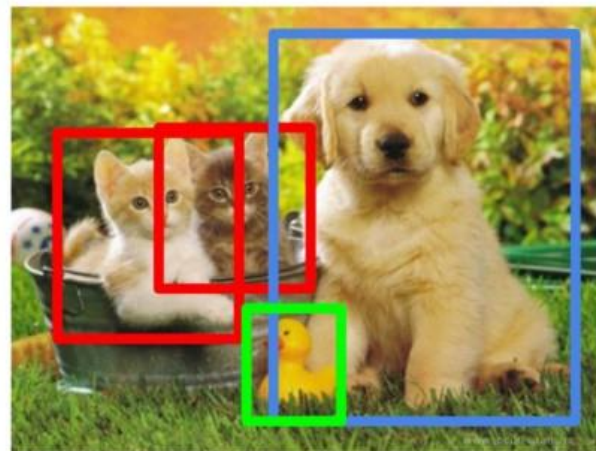
CAT

**Classification
+ Localization**



CAT

Object Detection



CAT, DOG, DUCK

**Instance
Segmentation**



CAT, DOG, DUCK

Single object

Multiple objects

Přístupy k úlohám detekce a rozpoznávání

- Tradiční způsoby
 - Barevné rozpoznávání
 - Tvarové rozpoznávání
 - Šablonové rozpoznávání
 - Výpočty příznaků
 - Klasifikace
- Metody založené na hlubokém učení (deep learning)
- Kombinace předchozích způsobů

Historie rozpoznávání snímků a detekce objektů

- 2001

- První real-time algoritmus pro detekci obličejů od Paul Viola a Michael Jones
- Známý jako Haar Cascades k nalezení v OpenCV

- 2005

- První použitelný algoritmus pro detekci osob od Navneet Dalal and Bill Triggs.
- Známý jako deskriptor Histograms of Oriented Gradients (HOG), k nalezení v OpenCV

- 2012

- Deep learningová síť od autorů Alex Krizhevsky, Ilya Sutskever, a Geoffrey Hinton šokuje svět výhrou v soutěži ImageNet dramatickým zvýšením přesnosti rozpoznávání

- 2015

- Deep learning je mainstream, algoritmy překonaly přesnost rozpoznávání lidí
- Přesnost klasifikace snímků převyšuje 95 %

Historie rozpoznávání snímků a detekce objektů

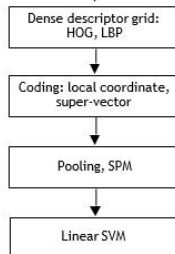
Large Scale Visual Recognition Challenge 2017 (ILSVRC2017)

Soutěž, který vyhodnocuje algoritmy pro lokalizaci/detekci objektů ve snímcích/video.

IMAGENET Large Scale Visual Recognition Challenge

Year 2010

NEC-UIUC

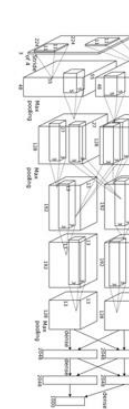


[Lin CVPR 2011]

Lion image by Swissfrog is
licensed under CC BY 3.0

Year 2012

SuperVision

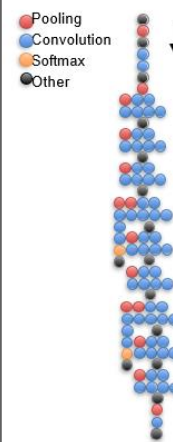


[Krizhevsky NIPS 2012]

Figure copyright Alex Krizhevsky, Ilya
Sutskever, and Geoffrey Hinton, 2012.
Reproduced with permission.

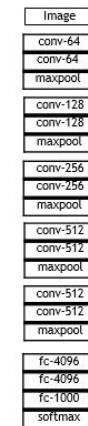
Year 2014

GoogLeNet



[Szegedy arxiv 2014]

VGG



[Simonyan arxiv 2014]

Year 2015

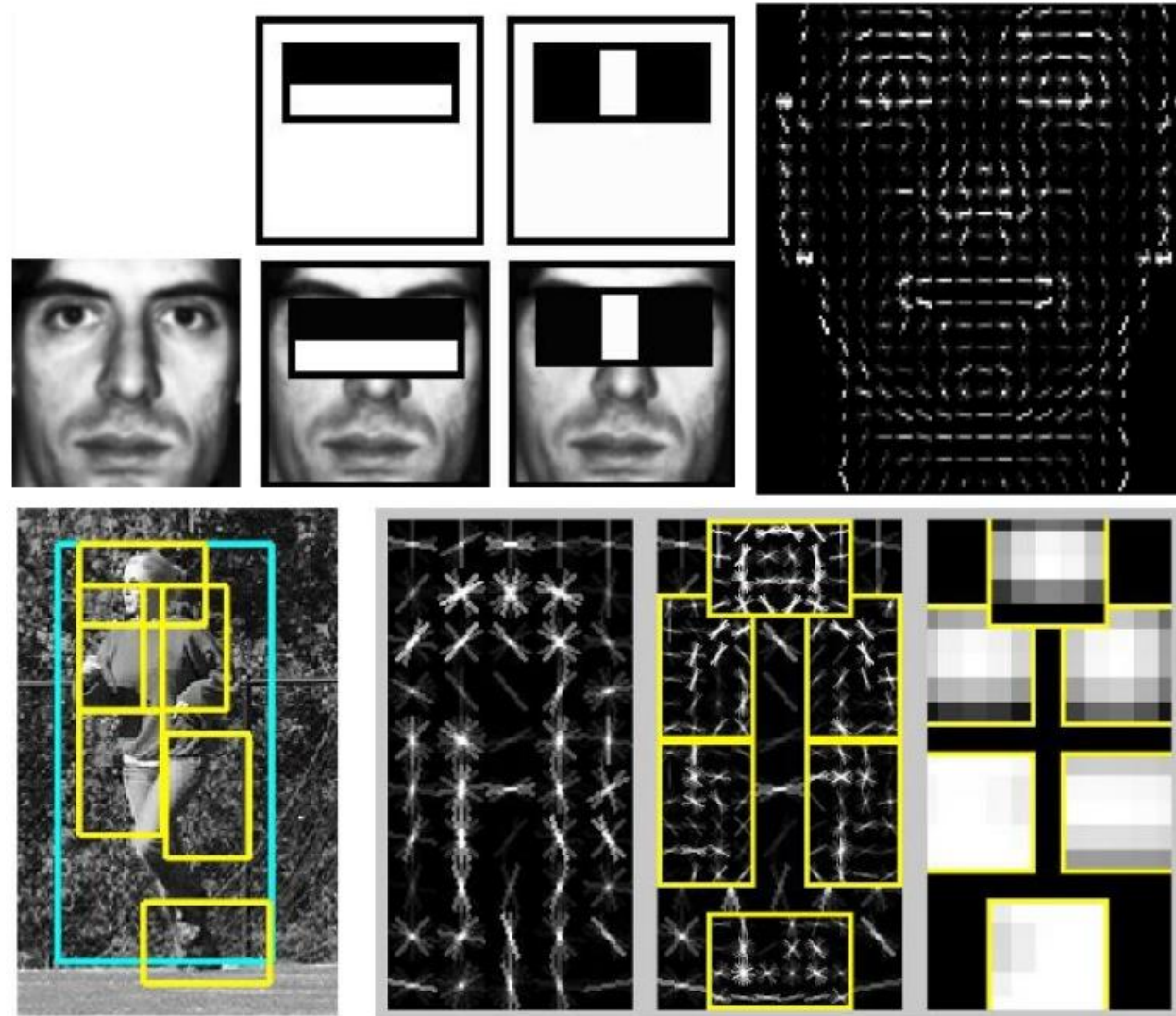
MSRA



[He ICCV 2015]

<http://image-net.org/challenges/LSVRC/2017/>

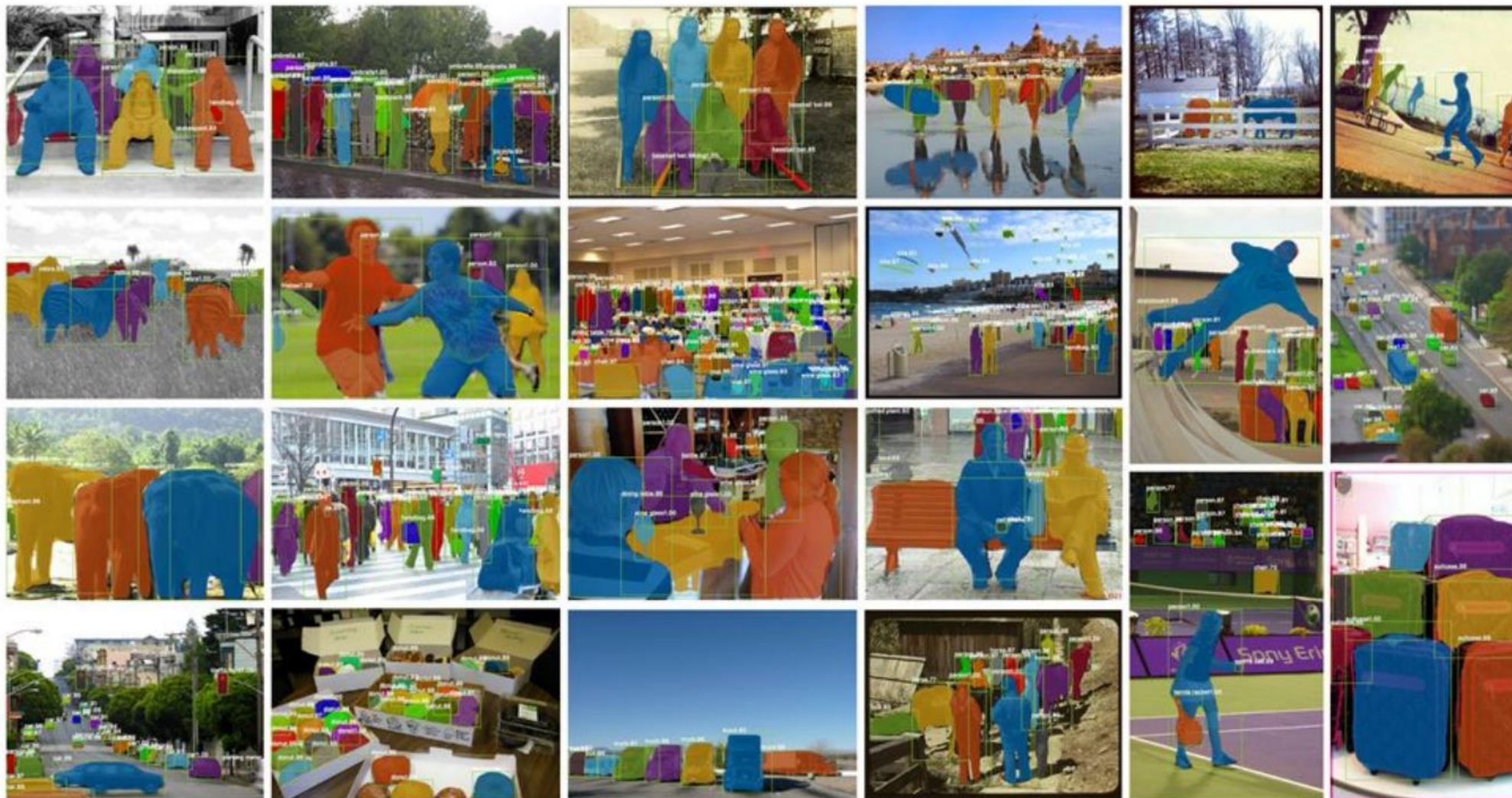
Detekce objektů 2001 – 2007



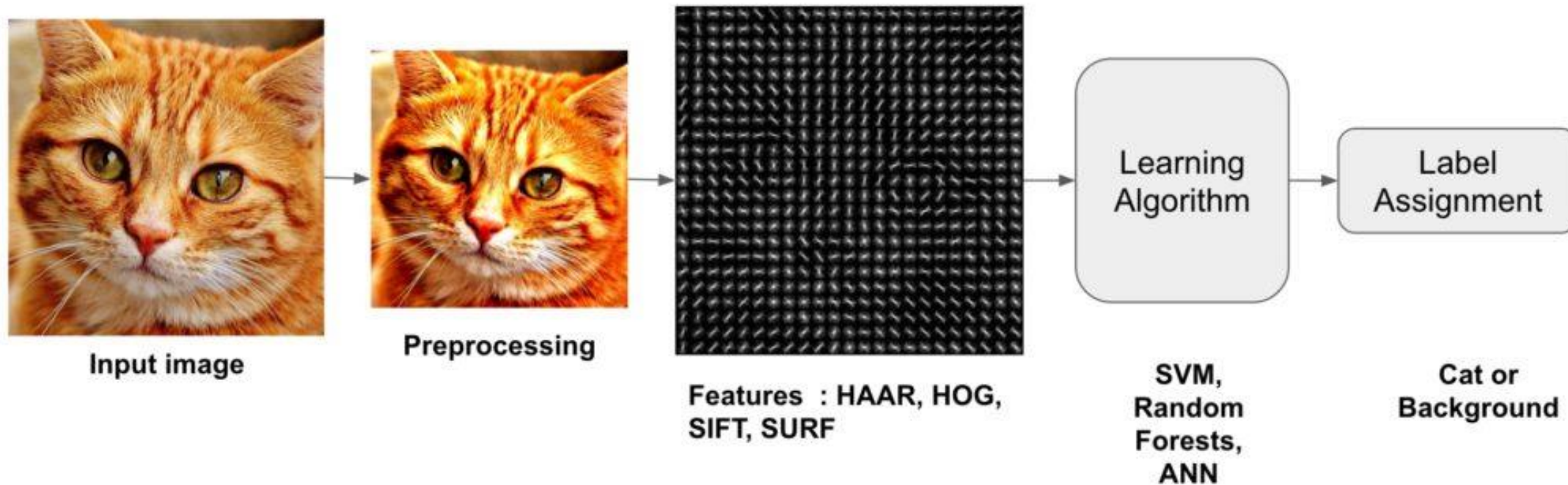
Detekce objektů 2007 – 2012



Detekce objektů nyní



Tradiční techniky počítačového vidění - pipeline



Většina tradičních způsobů respektuje tuto pipeline, deep learningové algoritmy zpravidla přeskakují část s extrakcí příznaků

Rozpoznávání obrazu (klasifikace obrazu)

- Vstupem je obraz
- Výstupem je třída, popisující co daný obraz obsahuje
 - Tedy např.: kočka, pes, auto, člověk
- Algoritmus rozpoznávání je potřeba nastavit (nejčastěji natrénovat na vzorových datech), tím zajistíme rozdílné klasifikace mezi různými třídami
- Pokud tedy chceme klasifikovat kočky a psy v obraze pomocí klasifikátoru, který se učí na vzorových datech, musíme mít stovky (až tisíce) trénovacích vzorků
- Pro zjednodušení dále uvažujme pouze binární klasifikátory (klasifikují do dvou tříd), kam patří i např. detekce lidí a obličejů

Oblíbený challenge – dog vs mop

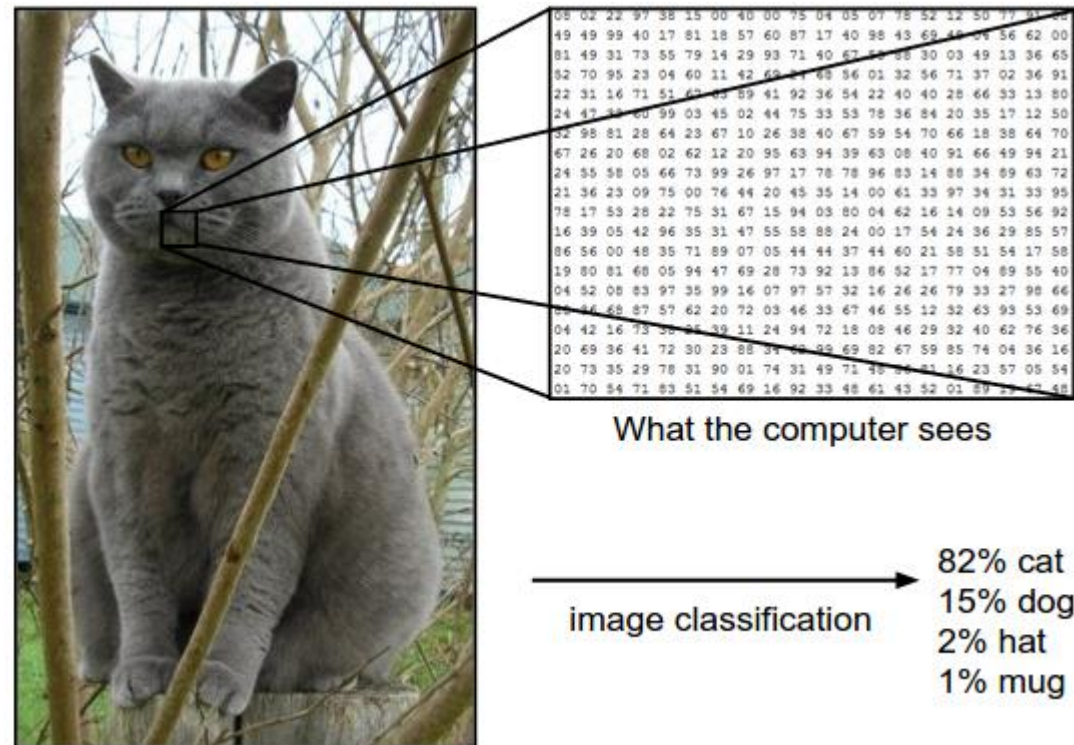


Pipeline rozpoznávání obrazu – předzpracování

- Ve většině případu se vstupní obraz předzpracovává, časté operace:
 - Úprava jasu a kontrastu
 - Gamma korekce
 - Ekvalizace histogramu
 - Odečtení průměru obrazu a vydělení směrodatnou odchylkou (normalizace)
 - Převod do různých barevných prostorů (RGB na HSV)
 - Perspektivní transformace
 - Ořez a škálování do předem daných rozměrů (kvůli extrakci příznaků)
- Problém je v tom, že nikdo ve skutečnosti neví, jaké operace předzpracování je potřeba provést a v jakém pořadí.
- Vše vychází z experimentů, kde ověřujeme to, který druh předzpracování poskytuje přesnější výsledky.

Pipeline rozpoznávání obrazu – extrakce příznaků

- Vstupní předzpracovaný obraz má v sobě příliš mnoho informací, které nejsou pro klasifikaci nutné
- Obraz s šířkou 248, výškou 400, v RGB prostoru obsahuje 297 600 čísel



Pipeline rozpoznávání obrazu – extrakce příznaků

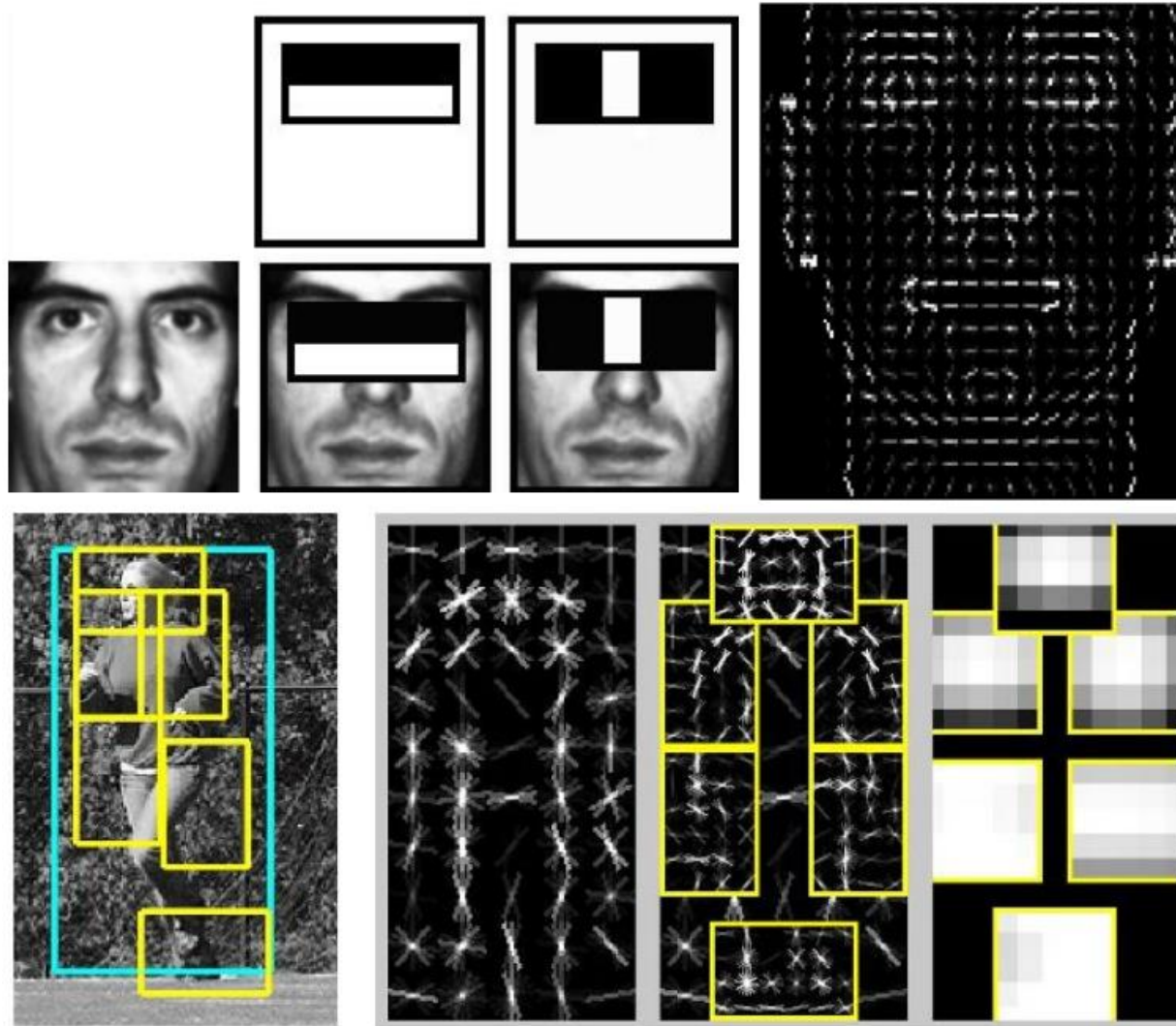
- Proto je prvním krokem v klasifikaci obrázků zjednodušení obrazu extrahováním důležitých informací (příznaků) obsažených v obraze a vynecháním ostatních informací
- Pokud budeme chtít hledat knoflíky na košili v obraze, zjistíme, že máme v okolí knoflíků významně rozdílné hodnoty RGB pixelů
- Nicméně, pomocí spuštění hranového detektoru můžeme obrázek zjednodušit a stále dokážeme snadno rozpoznat kruhový tvar knoflíků. Tedy zachováváme klíčové informace a zahazujeme nepotřebné (RGB hodnoty)



Pipeline rozpoznávání obrazu – extrakce příznaků

- V tradičních přístupech je navrhování extrahovaných příznaků rozhodující pro přesnost algoritmu
- Existují samozřejmě robustnější způsoby k extrakci příznaků, než je samotná hranová detekce, mezi nejznámější patří:
 - [Haar Cascades](#)
 - [Histogram of Oriented Gradients](#)
 - [Harris corners](#)
 - [SIFT](#)
 - [SURF](#)
 - [Local Binary Patterns \(LBP\)](#)
 - Histogramy

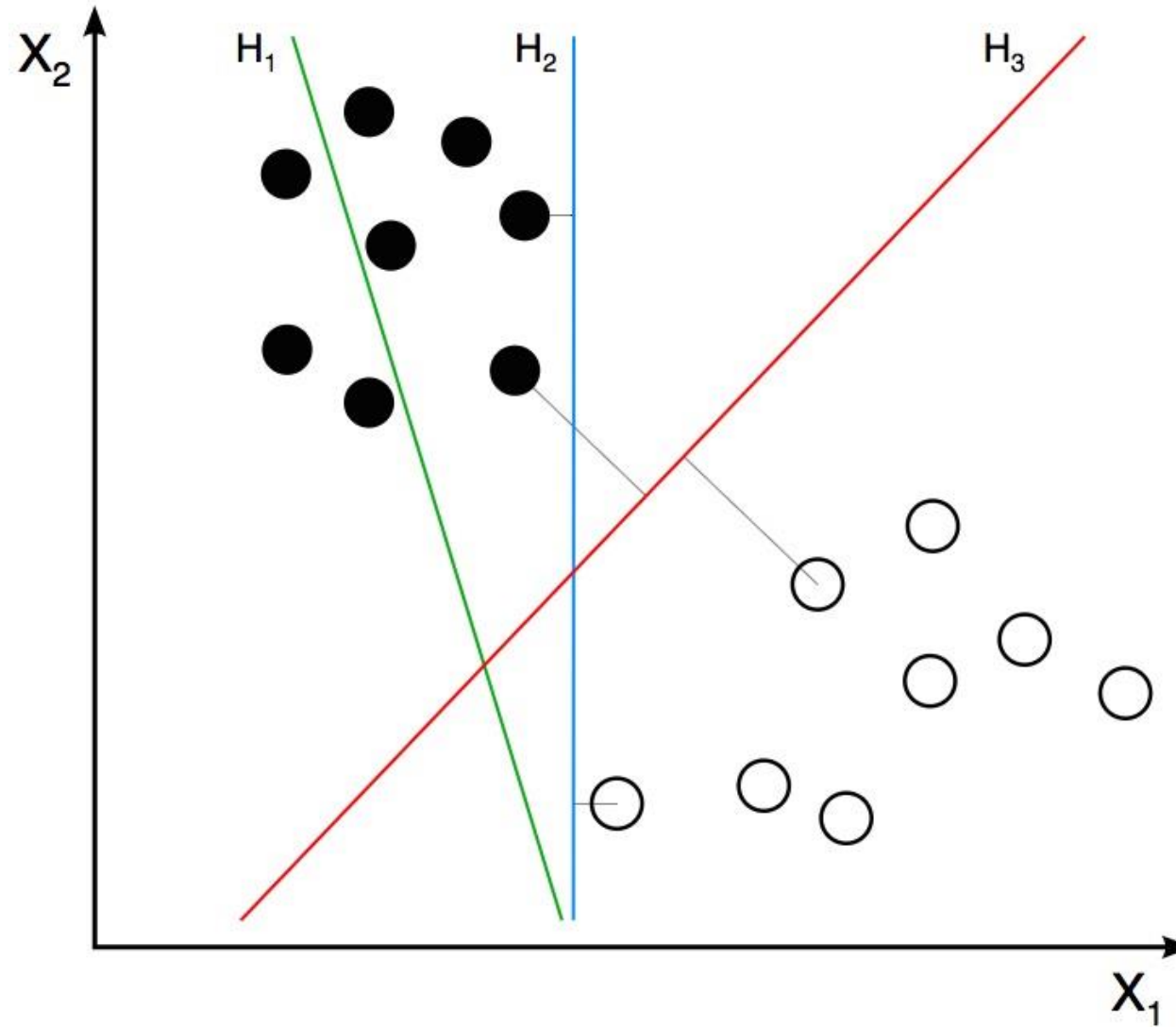
Pipeline rozpoznávání obrazu – extrakce příznaků



Pipeline rozpoznávání obrazu – klasifikátor

- Již víme, jak převést obraz na vektor důležitých příznaků
- Nyní potřebujeme vzít tento vektor, dát ho na vstup klasifikátoru a rozhodnout o jakou třídu objektu se jedná – kočka nebo pes
- Předtím než je klasifikátor postavený na učení z dat použitelný, musíme jej natrénovat na tisících příkladů
- Jak takový klasifikátor funguje, si popíšeme na jednom z donedávna nejpoužívanějších binárních klasifikátorů Support Vector Machines (SVM)
 - SVM se snaží nalézt „ideální“ rovinu, pomocí které rozdělí prostor koček a psů

Pipeline rozpoznávání obrazu – klasifikátor SVM

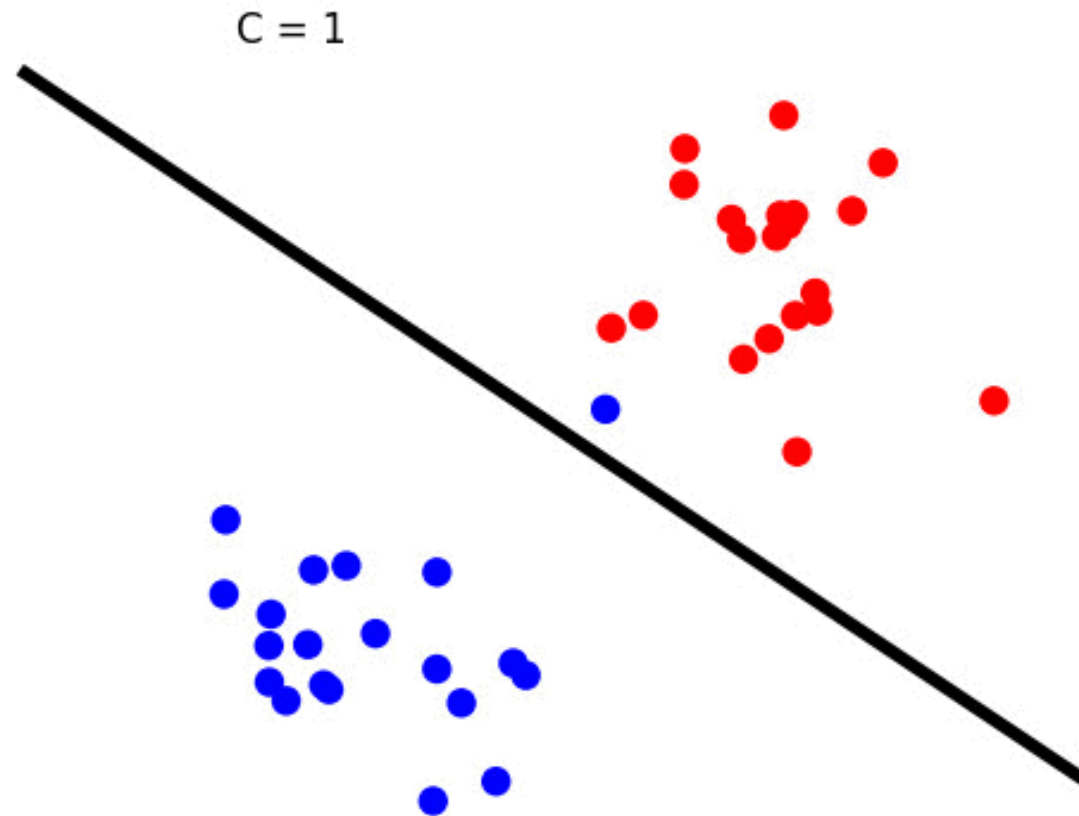


Pipeline rozpoznávání obrazu – klasifikátor

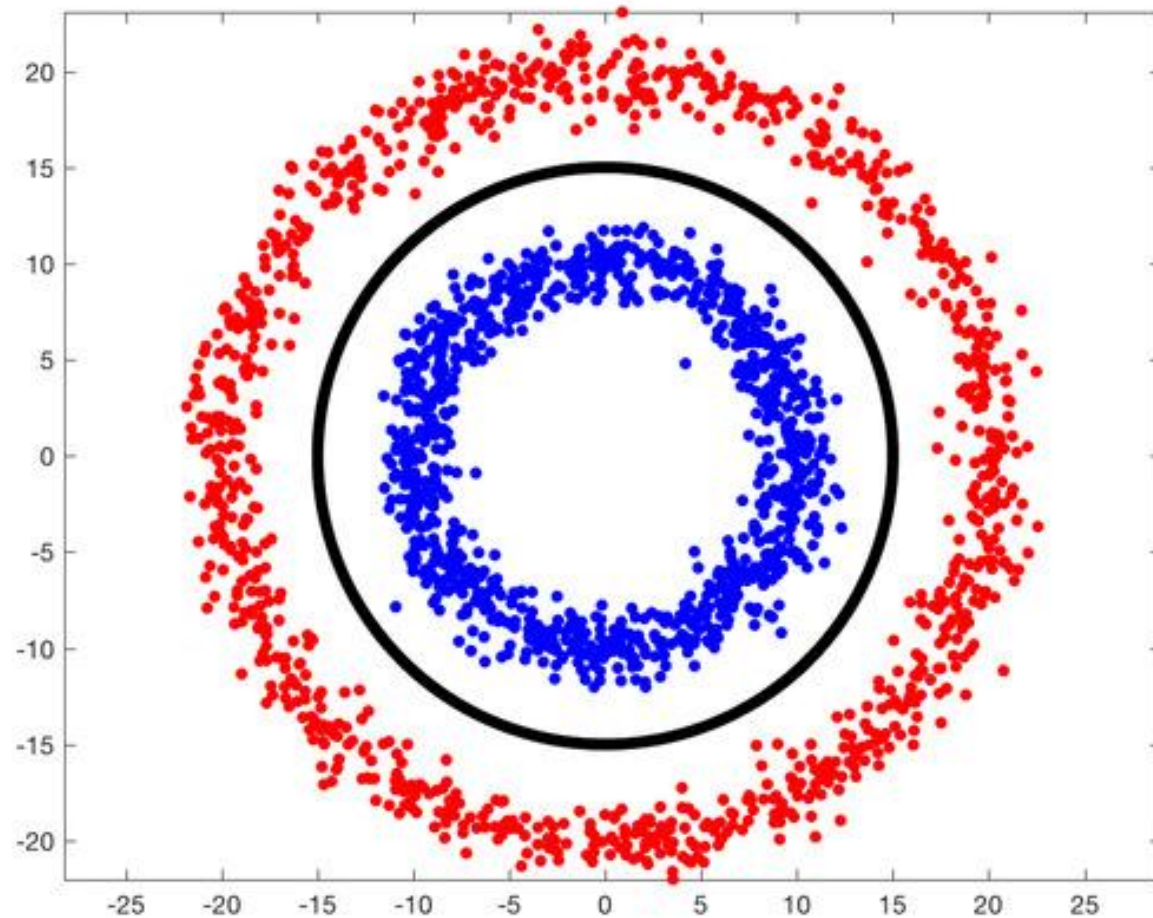
- V předchozím zjednodušeném příkladu máme pouze body ve 2D reprezentující dvě třídy - kočka nebo pes
- Černé tečky patří do třídy kočka, bílé tečky do třídy pes
- Během trénování iterativně hledáme nejlepší přímku rozdělující tyto dvě třídy
- K určení toho, jak si náš klasifikátor vede se využívá chybová/ztrátová funkce (loss function)
- Obecně se nepohybujeme pouze v 2D prostoru, ale v tisíce dimenzionálním – ten se ale hůř vizualizuje

Co když třídy nejdou jednoznačně oddělit?

Pipeline rozpoznávání obrazu – klasifikátor



Pipeline rozpoznávání obrazu – klasifikátor



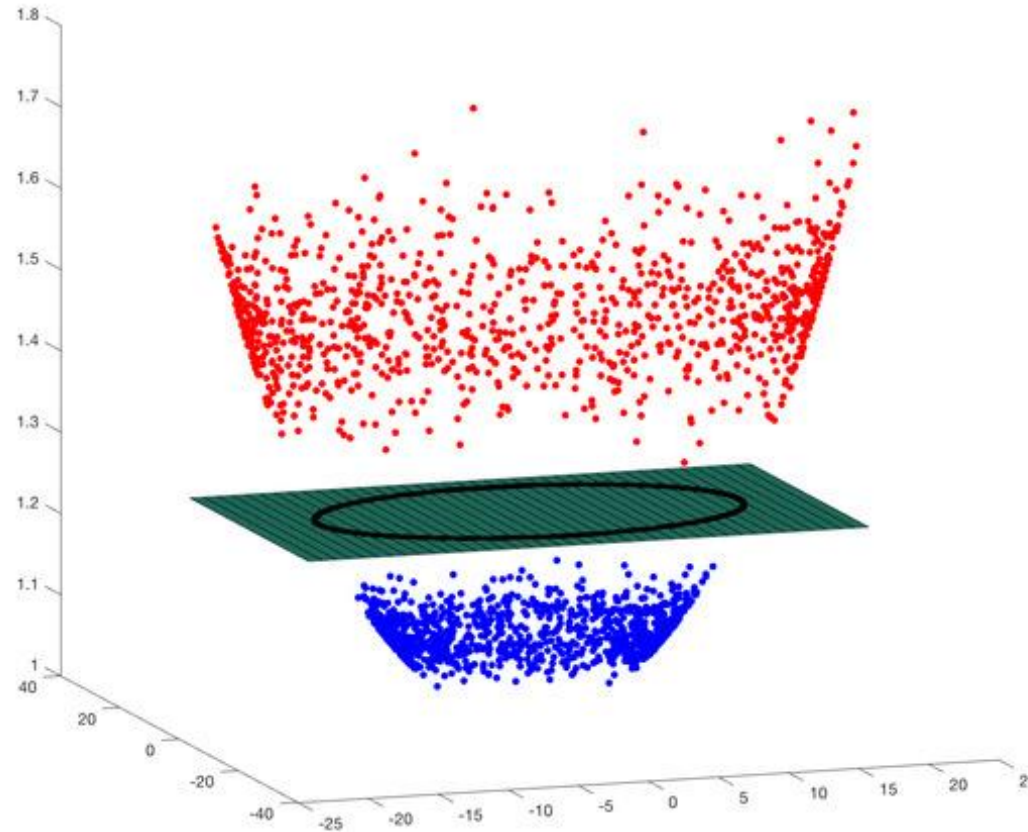
Pipeline rozpoznávání obrazu – klasifikátor

Přidání další dimenze –
tzv. rozšíření báze

U SVM sítě lze použít
např. jádro (kernel)

$$z = e^{-\gamma(x^2+y^2)}$$

Po transformaci 2D->3D
lze obě množiny snadno
lineárně oddělit.

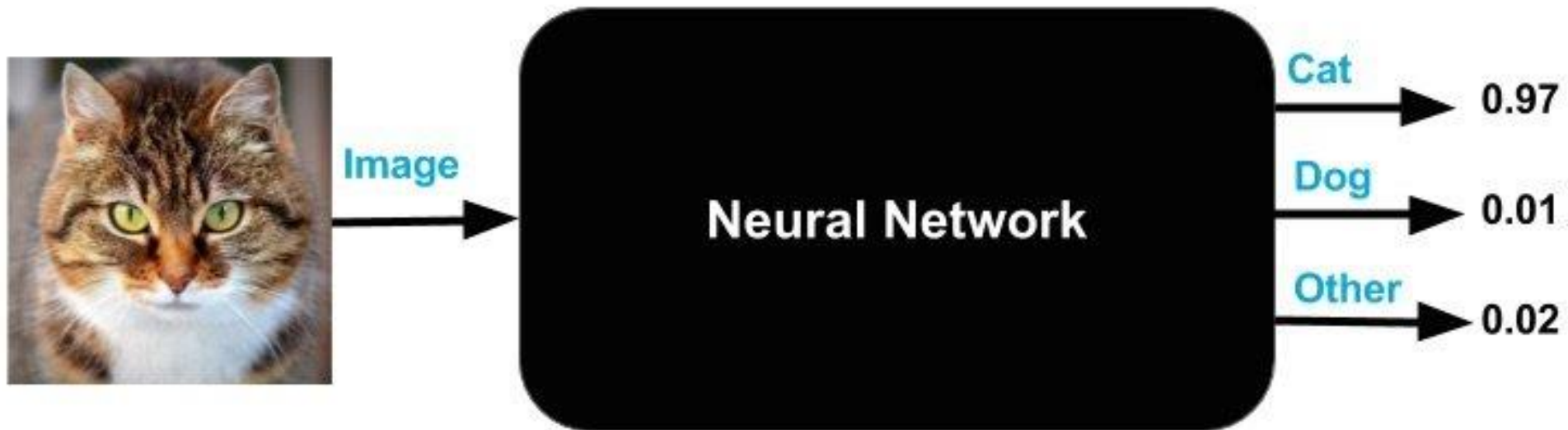


Problémy tradičních technik klasifikace

- Rotace
- Barevnost
- Osvětlení
- Škálování
- Doménový posun v datech (concept drift, domain shift)
- **Obecně nedostatečná generalizace**

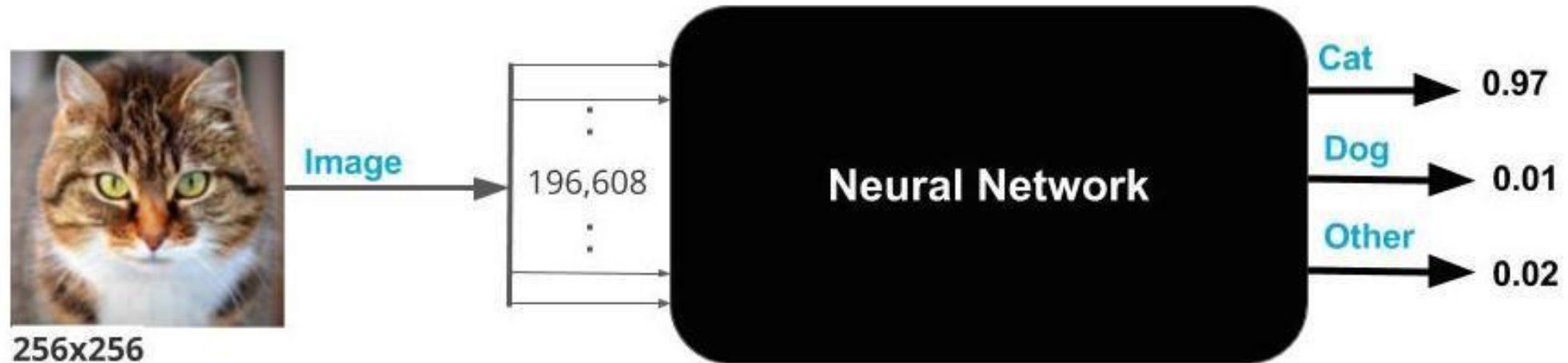
Deep learning, neuronové sítě - pipeline

- Minimální úsilí na předzpracování snímku a využívání black-boxu



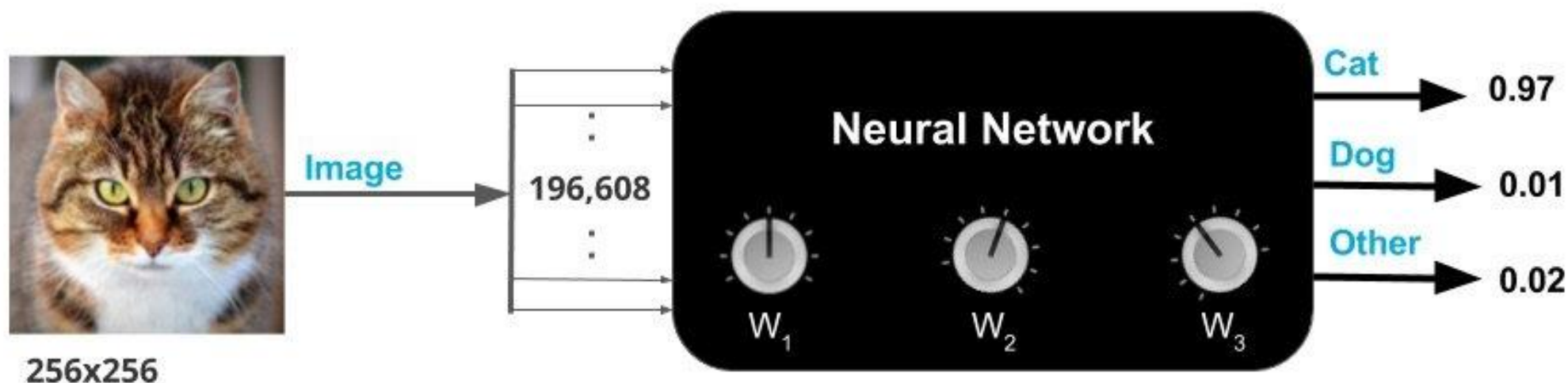
Deep learning, neuronové sítě - pipeline

- Vstupní snímek však musíme převést alespoň do vhodné reprezentace – vektoru fixní délky

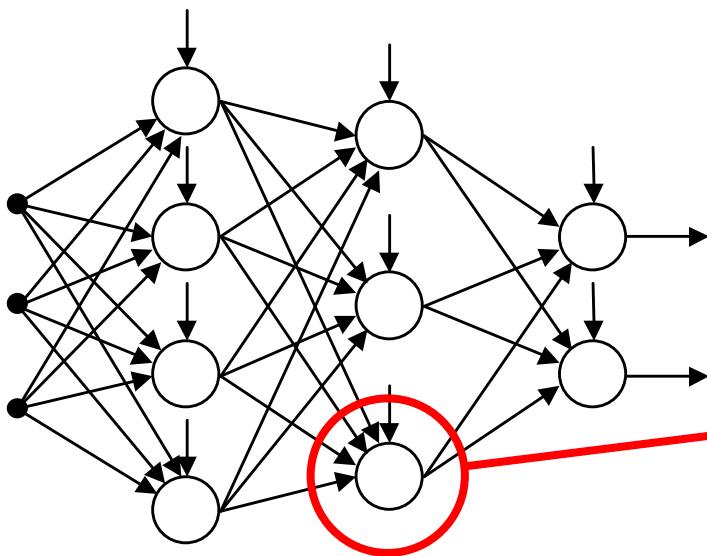


Deep learning, neuronové sítě - pipeline

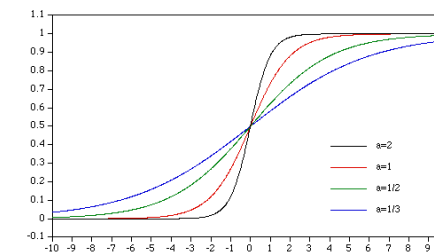
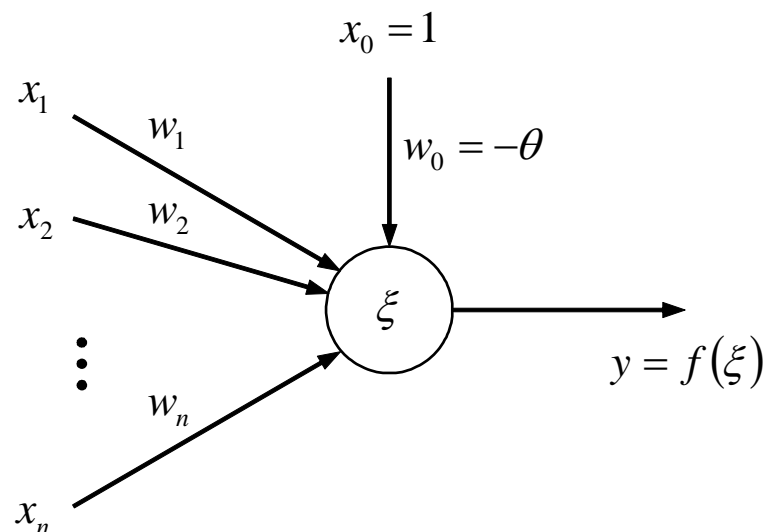
- Neuronové sítě k predikci využívají miliony vnitřních parametrů (váhy neuronů), které je k dosažení smysluplných výsledků, nutno správně natrénovat



Neuronová síť



Multilayer Perceptron (MLP)



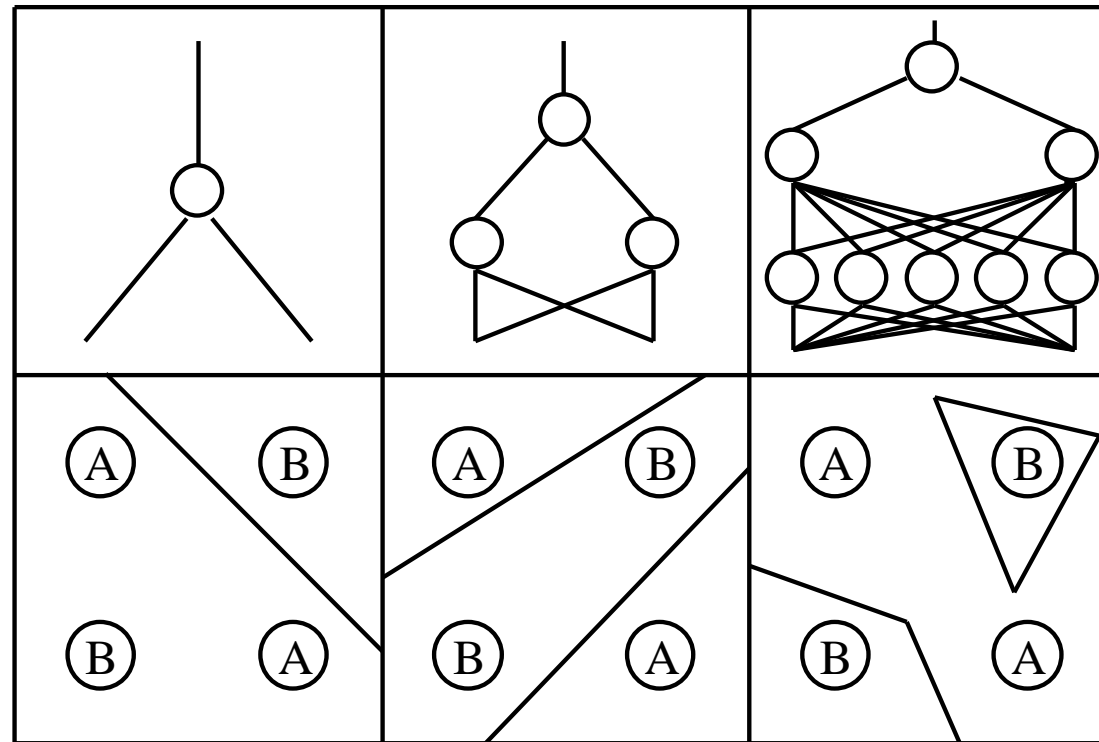
$$\xi = \sum_{i=1}^n w_i x_i - \theta = \sum_{i=0}^n w_i x_i$$

$$f(\xi) = \frac{1}{1 + e^{-\lambda \xi}}$$

1. výpočet (**post-synaptického**) **potenciálu**
2. výpočet hodnoty výstupu pomocí **aktivační funkce** (nejčastěji tzv. logistické sigmoidální funkce)

Neuronová síť

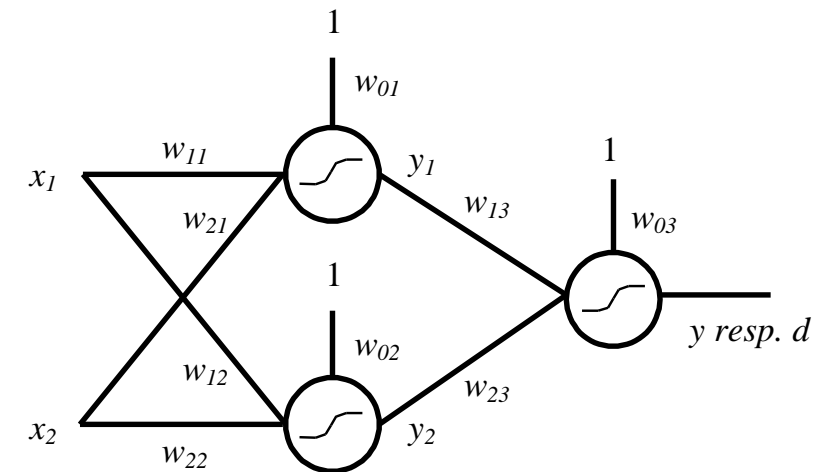
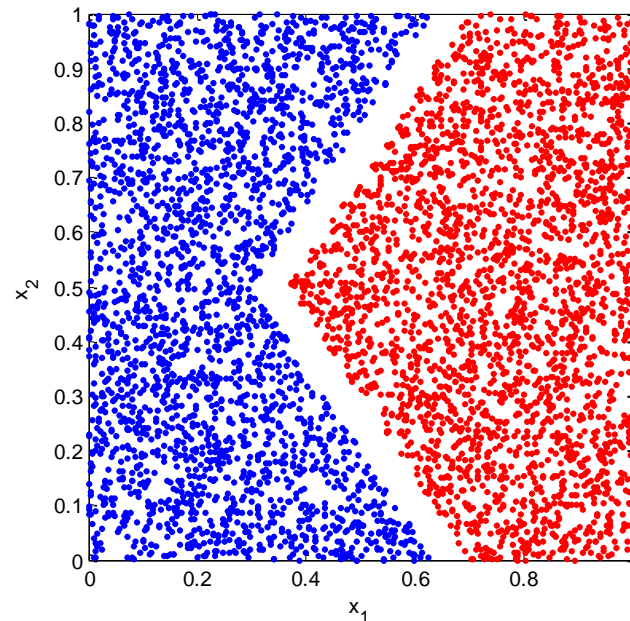
- Struktura sítě determinuje možnosti neuronové sítě



Neuronová síť

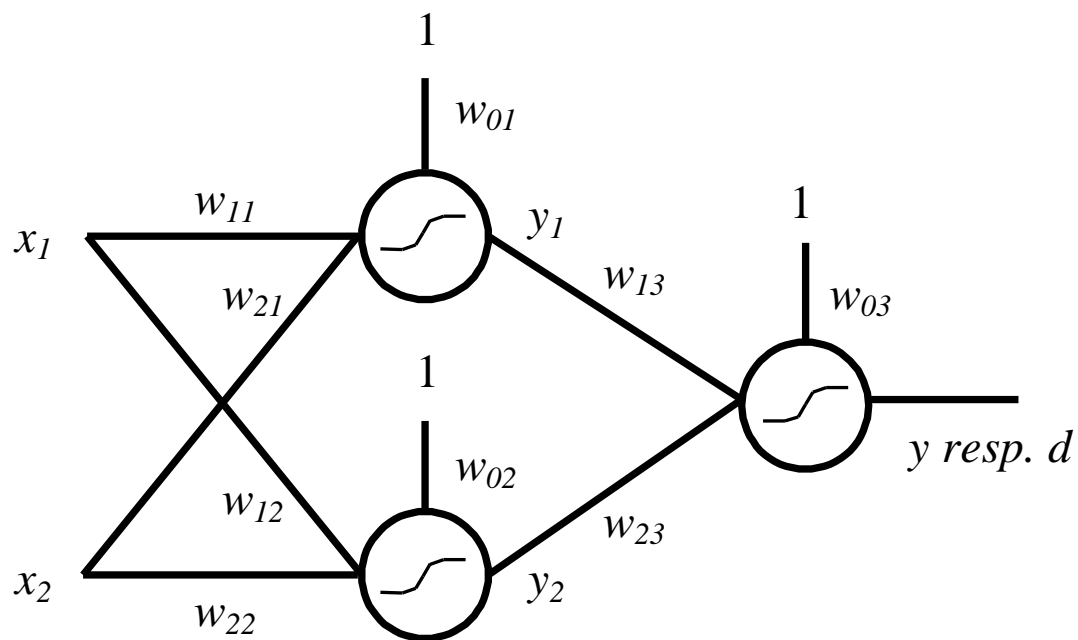
- Příklad trénování vícevrstvé perceptronové sítě

x1	x2	třída
0.8680	0.2640	1
0.1790	0.4230	0
0.6940	0.1630	1
0.5380	0.6450	1
0.6230	0.0030	0
0.7870	0.2680	1
0.4610	0.3860	1
0.6030	0.2790	1
0.1700	0.8050	0
...



Neuronová síť

- Učení neuronové sítě



Dopředná fáze

$$\xi_1 = w_{01} \cdot 1 + w_{11} \cdot x_1 + w_{21} \cdot x_2$$

$$y_1 = f(\xi_1)$$

(postsynaptický potenciál prvního neuronu)
(dílčí výstup z prvního neuronu)

$$\xi_2 = w_{02} \cdot 1 + w_{12} \cdot x_1 + w_{22} \cdot x_2$$

$$y_2 = f(\xi_2)$$

(postsynaptický potenciál druhého neuronu)
(dílčí výstup z druhého neuronu)

$$\xi_3 = w_{03} \cdot 1 + w_{13} \cdot y_1 + w_{23} \cdot y_2$$

$$y = f(\xi_3)$$

(postsynaptický potenciál třetího neuronu)
(výstup z třetího neuronu, tedy celkový výstup sítě)

Zpětná fáze (zpětné šíření chyby – **backpropagation**)

$$\delta = y(1 - y)(d - y)$$

$$w_{03} = w_{03} + \eta \cdot \delta \cdot 1,$$

$$w_{13} = w_{13} + \eta \cdot \delta \cdot y_1,$$

$$w_{23} = w_{23} + \eta \cdot \delta \cdot y_2,$$

$$\delta_1 = y_1(1 - y_1) \cdot \delta \cdot w_{13}$$

$$\delta_2 = y_2(1 - y_2) \cdot \delta \cdot w_{23}$$

Neuronová síť

- Geometrická interpretace vah

$$\xi_1 = w_{01} \cdot 1 + w_{11} \cdot x_1 + w_{21} \cdot x_2 = 0,$$

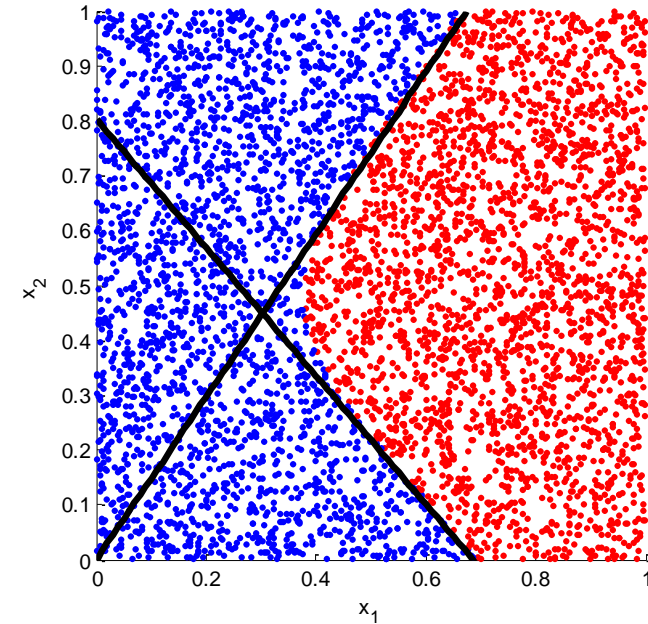
$$\xi_2 = w_{02} \cdot 1 + w_{12} \cdot x_1 + w_{22} \cdot x_2 = 0.$$

$$x_2 = -\frac{w_{11}}{w_{21}} \cdot x_1 - \frac{w_{01}}{w_{21}},$$

$$x_2 = -\frac{w_{12}}{w_{22}} \cdot x_1 - \frac{w_{02}}{w_{22}},$$

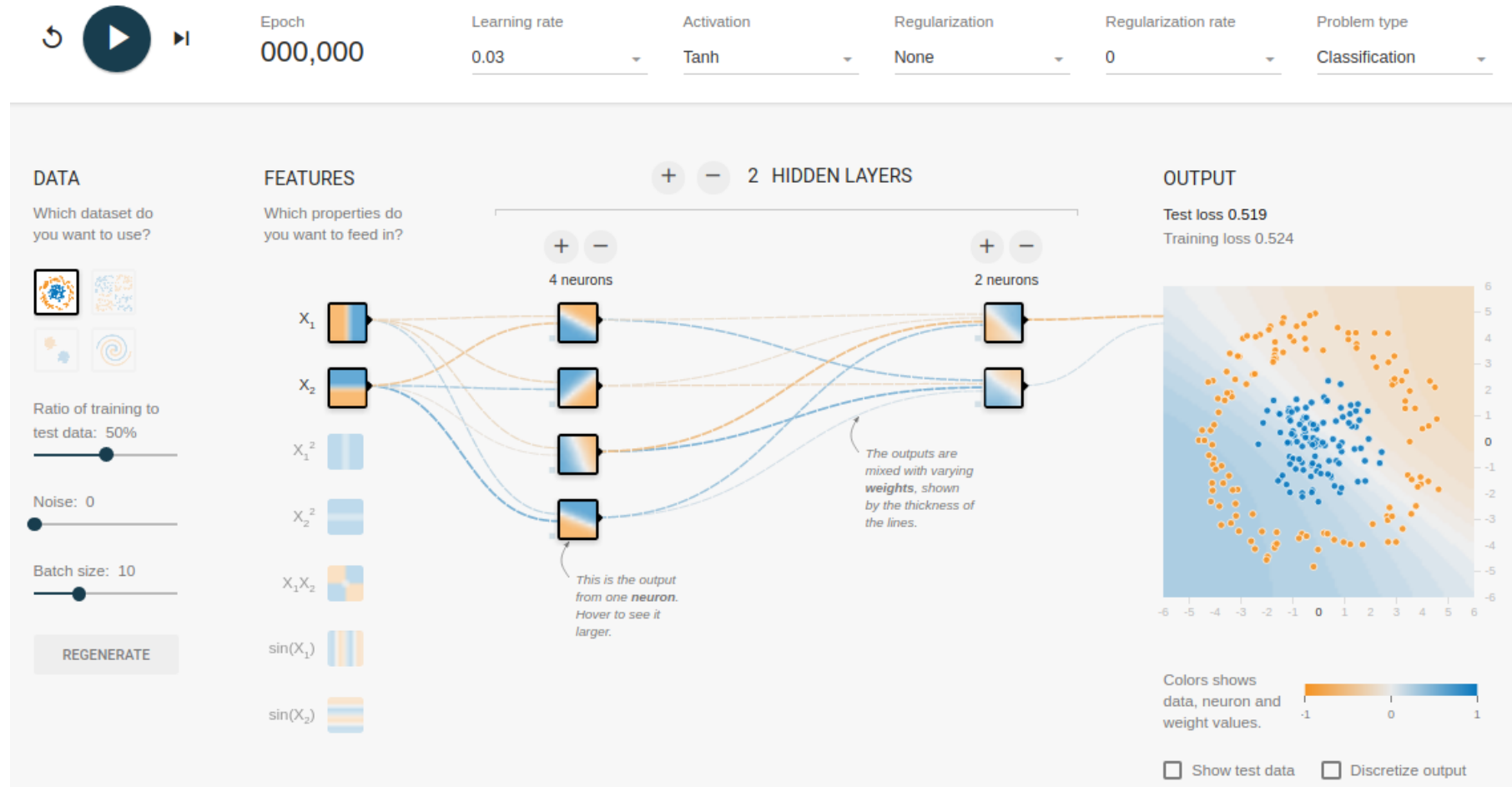
$$k_1 = -\frac{w_{11}}{w_{21}}, \quad k_2 = -\frac{w_{12}}{w_{22}}$$

$$q_1 = -\frac{w_{01}}{w_{21}}, \quad q_2 = -\frac{w_{02}}{w_{22}}$$



Neuronová síť

- Tensorflow



Neuronová síť

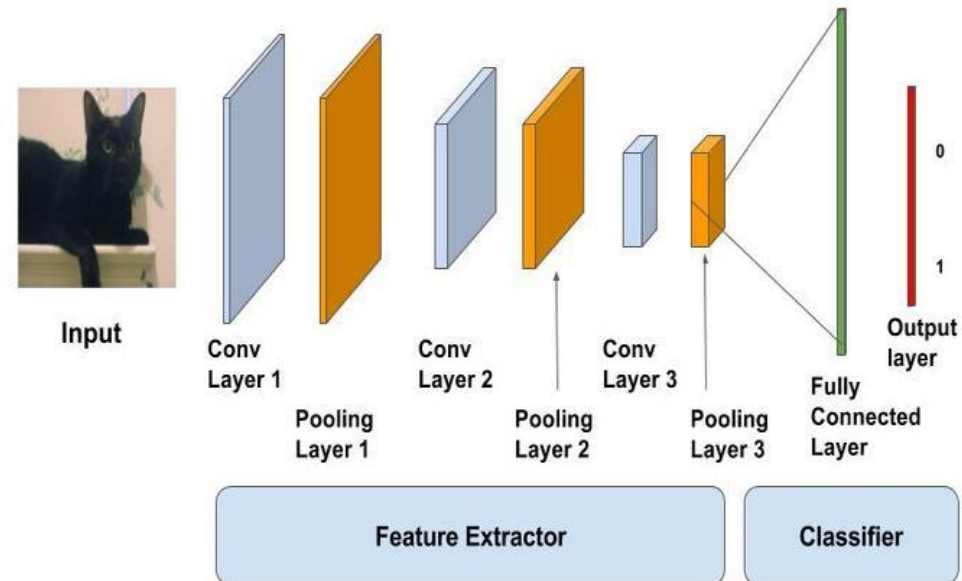
- Pokud aktivační funkce neuronu bude identita, tj. $y=f(x)=x$, pak neuron se chová jako obyčejná lineární kombinace vstupů vážených vahami.
- Transformace hodnot jedné vrstvy NS do druhé vrstvy NS pak lze maticově zapsat:

$$\begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} w_{11} & \cdots & w_{1n} \\ \vdots & \ddots & \vdots \\ w_{m1} & \cdots & w_{mn} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \quad \text{resp.} \quad \vec{y} = W \cdot \vec{x}$$

- Jde tedy o **lineární transformaci**. Pro specifické hodnoty matice vah W můžeme dostat známé lineární transformace, např. PCA, ICA, lineární filtry (např. pro detekci hran, rozmazání obrazu apod.)
- Konvoluce je také lin. transformací \rightarrow konvoluční neuronové sítě

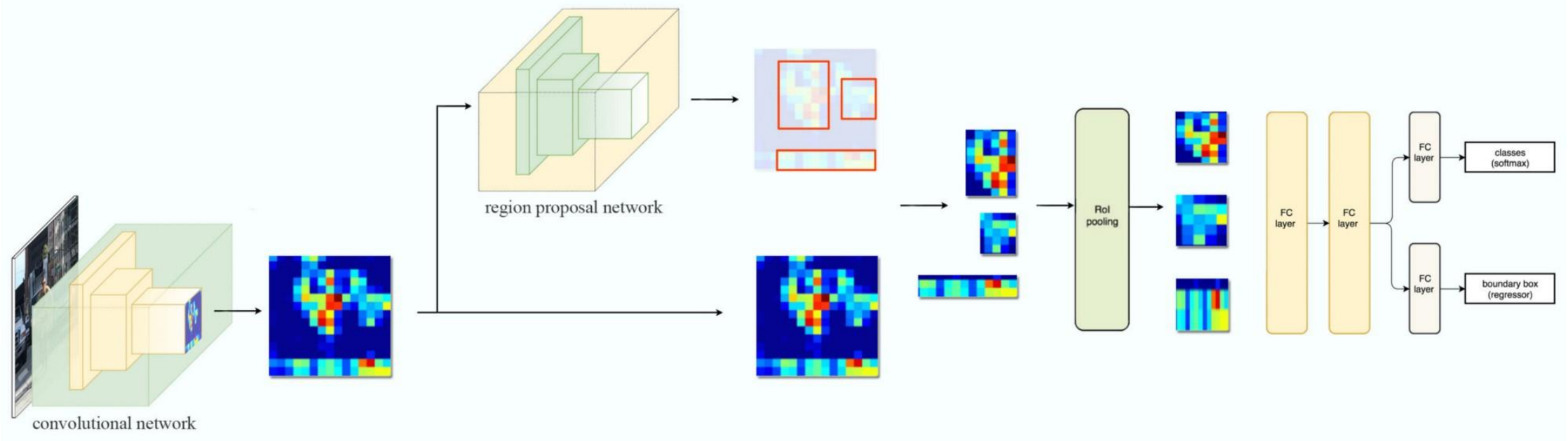
Deep learning, neuronové sítě – CNN

- Na obrázky se však nejčastěji používá Convolution Neural Network místo samotné Feed-forward NN
 - CNN se skládá z konvolučních a pooling prstev, pomocí kterých se snažíme získat vhodné atributy. Ty však již nelze rozumně interpretovat
 - Následně tyto atributy použijeme na vstup Feed-forward NN pro klasifikaci



Deep learning, neuronové sítě

- Existují samozřejmě mnohem složitější architektury, které se nestarají pouze o klasifikaci snímků, ale i o detekci několika objektů, případně získání jejich segmentační masky



Deep learning, neuronové sítě – detekce objektů



CAT? NO
DOG? NO

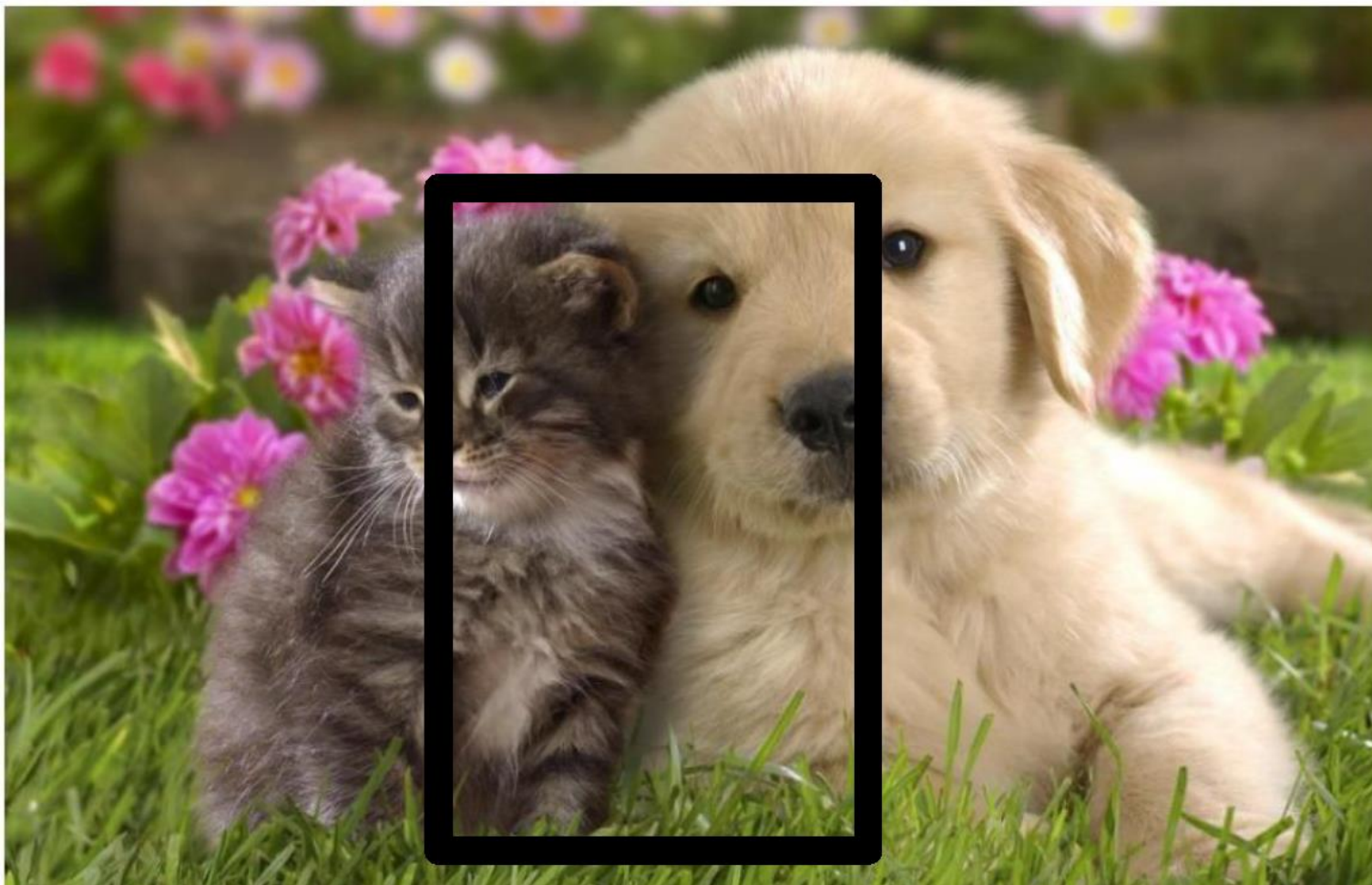
Deep learning, neuronové sítě – detekce objektů



CAT? YES

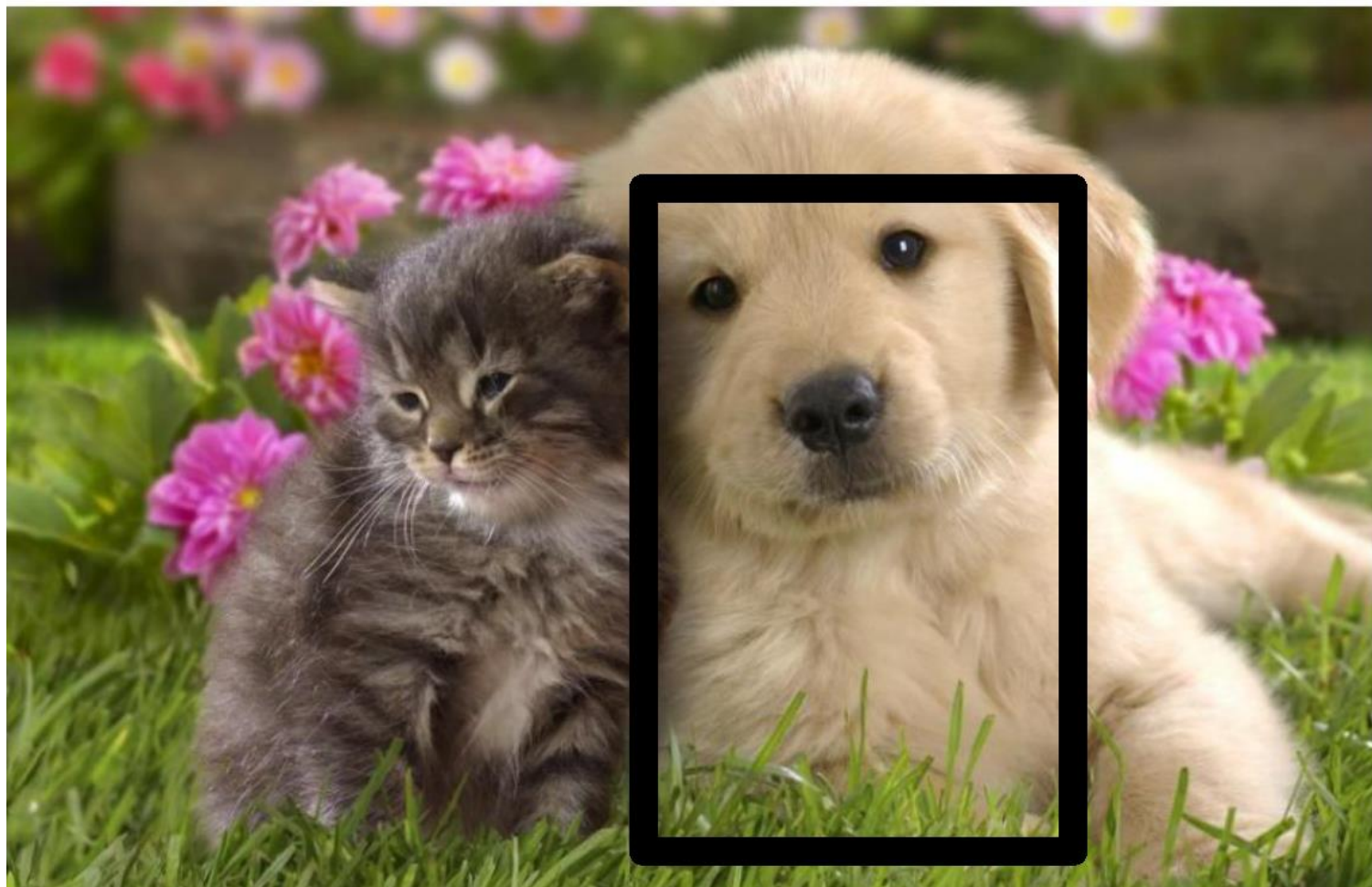
DOG? NO

Deep learning, neuronové sítě – detekce objektů



CAT? NO
DOG? NO

Deep learning, neuronové sítě – detekce objektů



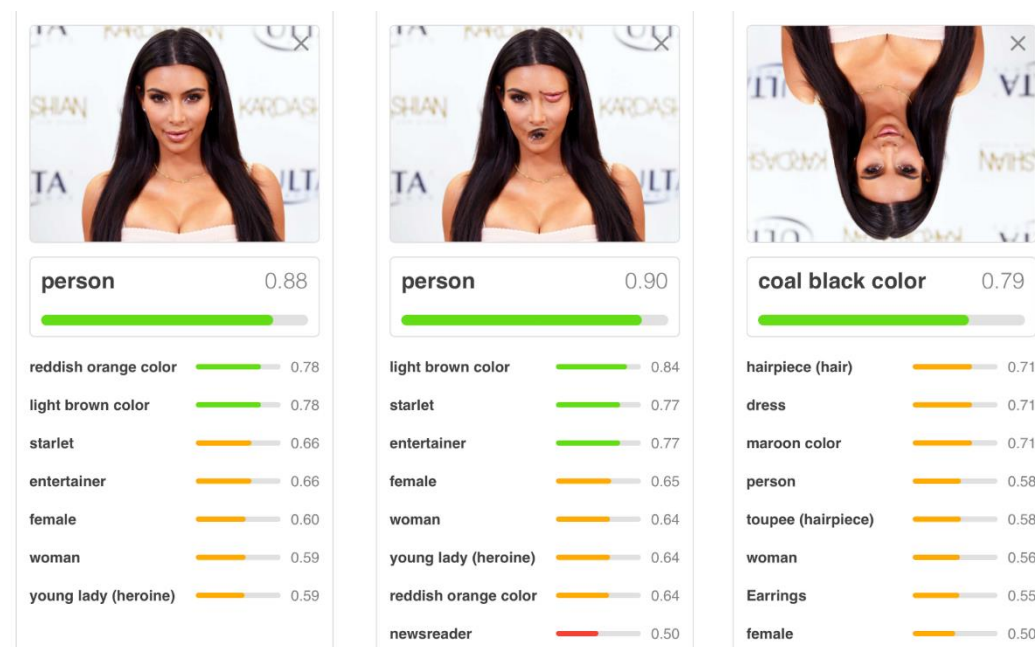
CAT? NO
DOG? YES

Problémy neuronových sítí – počítání, lokalizace, rotace

- **Klasické (konvoluční) neuronové sítě** nejsou konstruovány tak, aby zohledňovaly prostorové vztahy mezi daty popisující objekty.
- Proto je snadno „zmátnou“ obrázky, které jsou pouze pootočené, obsahují stejné ale prostorově přeházené objekty apod.
- Funkci NS pro rozpoznání obličeje bychom mohli symbolicky zapsat jako pravidlo:

```
if (2 eyes && 1 nose && 1 mouth) {  
    It's a face!  
}
```

- To ale bude vykazovat chyby, viz



Problémy neuronových sítí – počítání, lokalizace, rotace

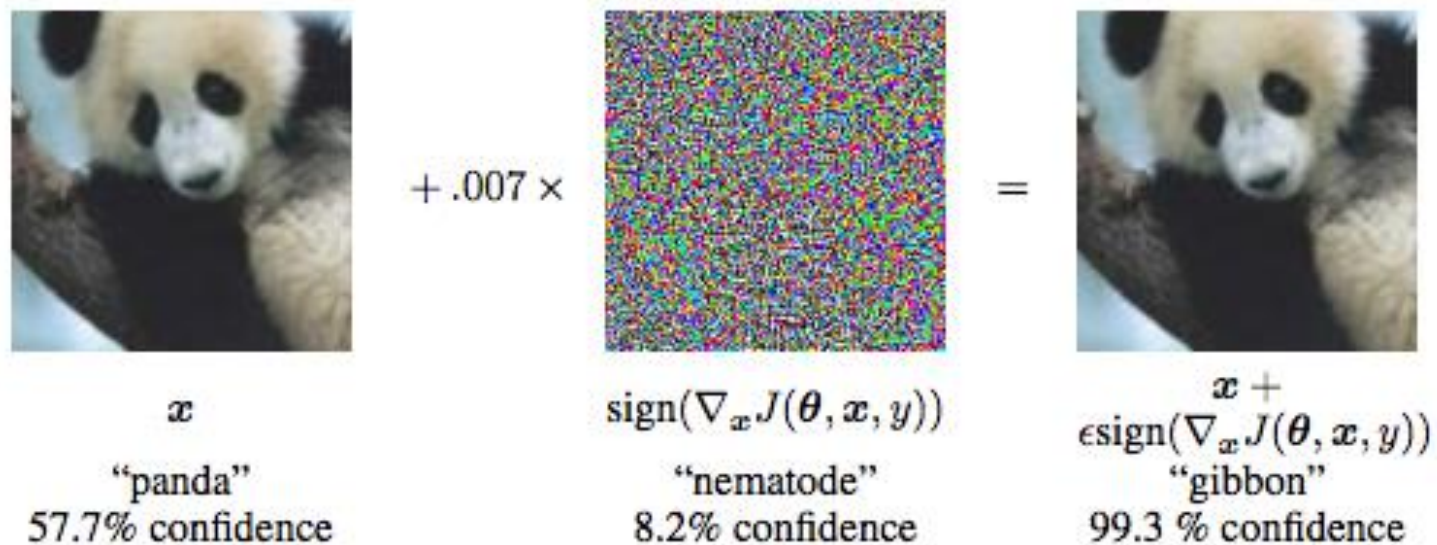
- Možným řešením je např. zvýšit počet trénovacích vzorů. Např. předložit sadu různě potočených obrázků, abychom byli schopni rozpoznávat potočené obrázky.
- Takové řešení je možné, ale představuje pouhou hrubou sílu.
- Jiným možným řešením je použít tzv. **Capsule Networks** (CapsNet), které zohledňují prostorovou informaci obsaženou v datech (obrázku)
- Jejich funkci bychom mohli symbolicky zapsat jako pravidlo:

```
if (2 adjacent eyes && nose under eyes && mouth under nose) {  
    It's a face!  
}
```

- Takovéto pravidlo je už obtížnější zmást.
- <https://hackernoon.com/capsule-networks-are-shaking-up-ai-heres-how-to-use-them-c233a0971952>
- <https://viking-sudo-rm.github.io/nlp/2018/11/29/Capsule-Networks-for-NLP/>

Problémy neuronových sítí – skryté vzory

- Skryté vzory mohou „rozhodit“ neuronovou síť



- **Vysvětlení:** I malý aditivní šum je v NS násoben vahami, které mohou příspěvek šumu zvednout natolik, že se stane významným.
- Pokud šum uměle odvodíme od hodnot vah, můžeme tento vliv velmi výrazně podpořit, viz <https://arxiv.org/pdf/1412.6572.pdf>

Datasets

2007

Pascal VOC

- 20 tříd
- 11K trénovacích obrázků
- 27K trénovacích objektů

Používán jako standard, nyní již pouze k rychlému otestování nového algoritmu

2013

ImageNet ILSVRC

- 200 tříd
- 476K trénovacích obrázků
- 534K trénovacích objektů

Pascal VOC na steroidech

2015

MS COCO

- 80 Classes
 - 200K trénovacích obrázků
 - 1.5M trénovacích objektů
- Více kategorií v jednotlivých obrazech. Zaměřený spíše na malé objekty

Hodnocení kvality klasifikace – matice záměn

- **True Positives (TP):** data jsou predikována do třídy 1 a správně patří do třídy 1
- **True Negatives (TN):** data jsou predikována do třídy 0 a správně patří do třídy 0
- **False Positives (FP):** data jsou predikována do třídy 1, ale správně patří do třídy 0
- **False Negatives (FN):** data jsou predikována do třídy 0, ale správně patří do třídy 1

		Actual	
		Positives(1)	Negatives(0)
Predicted	Positives(1)	TP	FP
	Negatives(0)	FN	TN

Hodnocení kvality klasifikace

- *Citlivost* (Sensitivity) = $TP / (TP + FN)$,
 - *Specifita* (Specificity) = $TN / (FP + TN)$,
 - *Pozitivní hodnotu predikce* (Positive predictive value) = $TP / (TP + FP)$,
 - *Negativní hodnotu predikce* (Negative predictive value) = $TN / (FN + TN)$,
 - *Efektivita* (Efficiency) = $(TP + TN) / (TP + TN + FP + FN)$
-
- *TP* – true positive, *TN* – true negative,
 - *FP* – false positive, *FN* – false negative

Metriky klasifikace – ostatní

- [Odkaz](#)

		True condition			
Total population		Condition positive	Condition negative	$Prevalence = \frac{\sum \text{Condition positive}}{\sum \text{Total population}}$	$Accuracy (ACC) = \frac{\sum \text{True positive} + \sum \text{True negative}}{\sum \text{Total population}}$
Predicted condition	Predicted condition positive	True positive , Power	False positive , Type I error	Positive predictive value (PPV), Precision = $\frac{\sum \text{True positive}}{\sum \text{Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\sum \text{False positive}}{\sum \text{Predicted condition positive}}$
	Predicted condition negative	False negative , Type II error	True negative	False omission rate (FOR) = $\frac{\sum \text{False negative}}{\sum \text{Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\sum \text{True negative}}{\sum \text{Predicted condition negative}}$
		True positive rate (TPR), Recall, Sensitivity, probability of detection $= \frac{\sum \text{True positive}}{\sum \text{Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\sum \text{False positive}}{\sum \text{Condition negative}}$	Positive likelihood ratio (LR+) = $\frac{TPR}{FPR}$	Diagnostic odds ratio (DOR) $= \frac{LR+}{LR-}$
		False negative rate (FNR), Miss rate $= \frac{\sum \text{False negative}}{\sum \text{Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) $= \frac{\sum \text{True negative}}{\sum \text{Condition negative}}$	Negative likelihood ratio (LR-) = $\frac{FNR}{TNR}$	
				$F_1 \text{ score} = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}}$	

sensitivity, recall, hit rate, or true positive rate (TPR)

$$\text{TPR} = \frac{\text{TP}}{P} = \frac{\text{TP}}{\text{TP} + \text{FN}} = 1 - \text{FNR}$$

specificity, selectivity or true negative rate (TNR)

$$\text{TNR} = \frac{\text{TN}}{N} = \frac{\text{TN}}{\text{TN} + \text{FP}} = 1 - \text{FPR}$$

precision or positive predictive value (PPV)

$$\text{PPV} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

negative predictive value (NPV)

$$\text{NPV} = \frac{\text{TN}}{\text{TN} + \text{FN}}$$

miss rate or false negative rate (FNR)

$$\text{FNR} = \frac{\text{FN}}{P} = \frac{\text{FN}}{\text{FN} + \text{TP}} = 1 - \text{TPR}$$

fall-out or false positive rate (FPR)

$$\text{FPR} = \frac{\text{FP}}{N} = \frac{\text{FP}}{\text{FP} + \text{TN}} = 1 - \text{TNR}$$

false discovery rate (FDR)

$$\text{FDR} = \frac{\text{FP}}{\text{FP} + \text{TP}} = 1 - \text{PPV}$$

false omission rate (FOR)

$$\text{FOR} = \frac{\text{FN}}{\text{FN} + \text{TN}} = 1 - \text{NPV}$$

accuracy (ACC)

$$\text{ACC} = \frac{\text{TP} + \text{TN}}{P + N} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

F1 score

is the harmonic mean of precision and sensitivity

$$F_1 = 2 \cdot \frac{\text{PPV} \cdot \text{TPR}}{\text{PPV} + \text{TPR}} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}$$

Matthews correlation coefficient (MCC)

$$\text{MCC} = \frac{\text{TP} \times \text{TN} - \text{FP} \times \text{FN}}{\sqrt{(\text{TP} + \text{FP})(\text{TP} + \text{FN})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}}$$

Informedness or Bookmaker Informedness (BM)

$$\text{BM} = \text{TPR} + \text{TNR} - 1$$

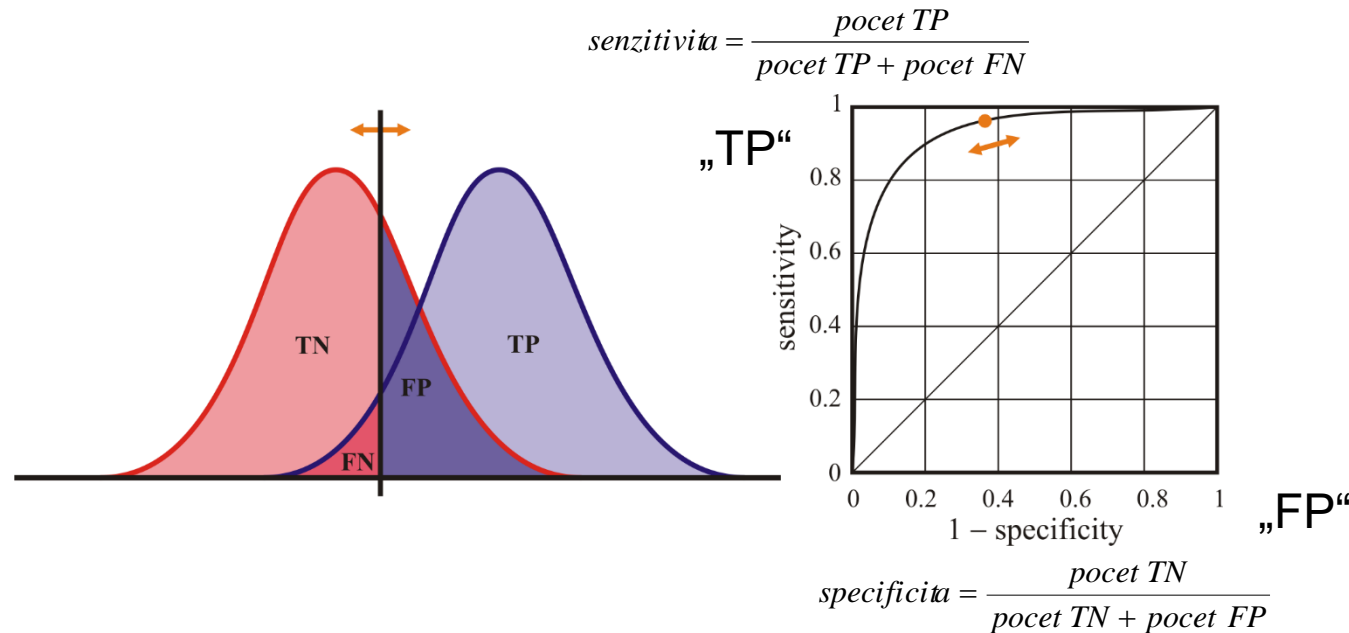
Markedness (MK)

$$\text{MK} = \text{PPV} + \text{NPV} - 1$$

Hodnocení kvality klasifikace

- **ROC křivka**


- Receiver Operating Characteristics
- z průběhu resp. plochy pod vynesenu křivkou můžeme usuzovat na kvalitu klasifikace resp. klasifikátoru

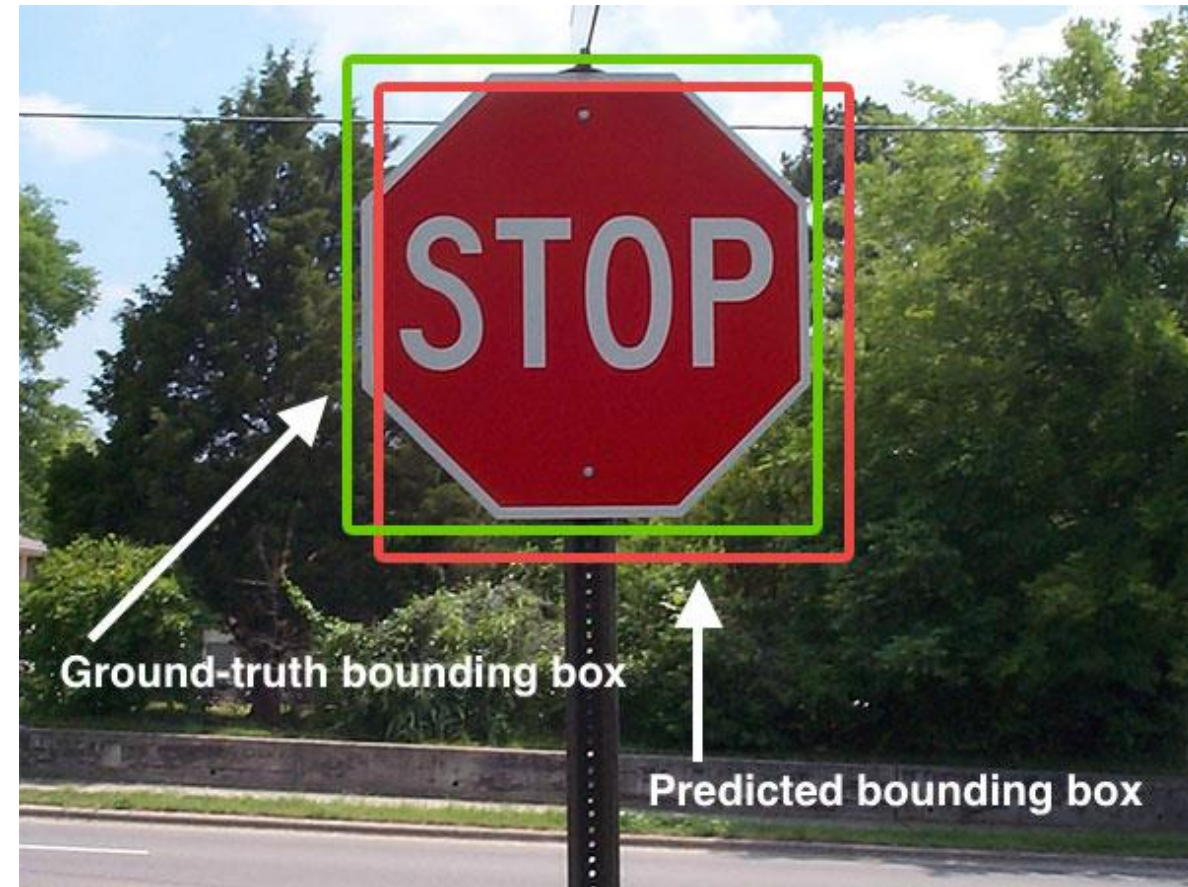


- <http://www.navan.name/roc/>

Metrika lokalizace – Intersection over Union (IoU)

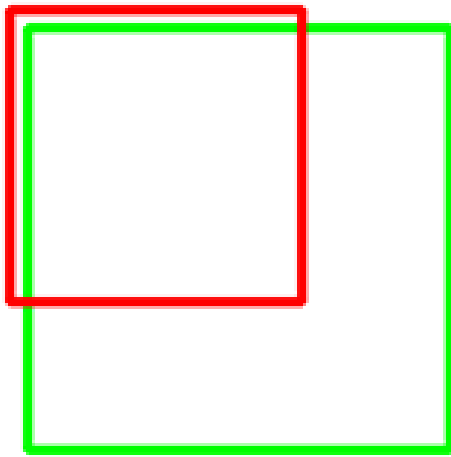
- Vzdálenostní metrika popisující přesnost skutečné lokalizace proti predikci

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$




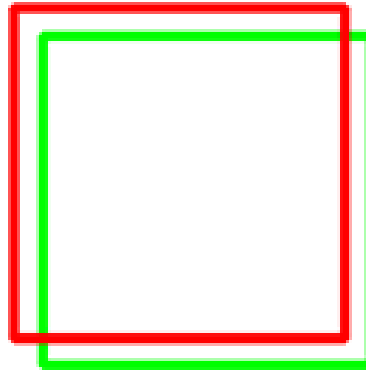
Metrika lokalizace – Intersection over Union (IoU)

IoU: 0.4034



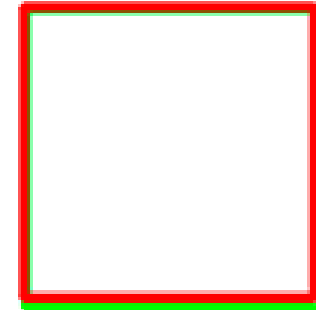
Poor

IoU: 0.7330



Good

IoU: 0.9264



Excellent

Zdroje

- <https://luozm.github.io/cv-tasks>
- <https://www.slideshare.net/Brodmann17/introduction-to-object-detection>
- <https://www.learnopencv.com/image-recognition-and-object-detection-part1/>
- <https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/>
- <https://medium.com/greyatom/performance-metrics-for-classification-problems-in-machine-learning-part-i-b085d432082b>