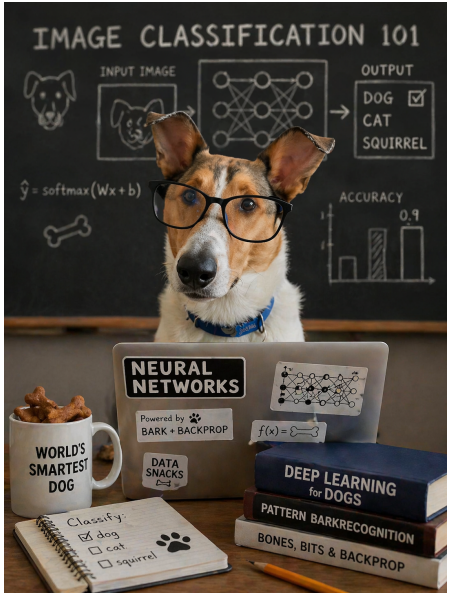
Klasifikace – Neuronové sítě

Pokročilá analýza obrazu

Mgr. Markéta Trnečková, Ph.D.

Klasifikace

- Využití neuronových sítí při klasifikaci obrazů
- Vícevrstvé, plně propojené neuronové sítě – vstup vektory příznaků
- Konvoluční neuronové sítě – vstup obrazy
- Nepůjdeme do hloubky v teorii neuronových sítí – to je cílem jiných předmětů



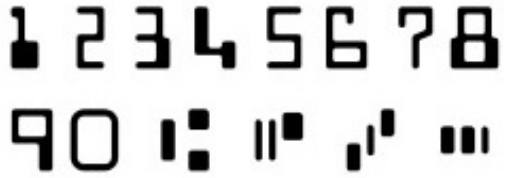
Neuronové sítě – Pozadí

- Podstatou je využití velkého množství elementárních nelineárních výpočetních prvků (**umělé neurony**)
- Uspořádány do sítí s propojeními podobnými neuronům ve vizuální kůře savců
- Používáme označení **neuronové sítě**
- Slouží k adaptivnímu učení parametrů rozhodovacích funkcí pomocí trénovacích vzorů

- Klasické přístupy:
  - transformace dat pomocí člověkem navržených metod
  - např. extrakce příznaků
- Neuronové sítě:
  - automatické učení reprezentací ze surových dat
  - vrstvy postupně zpřesňují reprezentaci

Neuronové sítě – Poznámky

- Vyžadují specifikaci parametrů:
  - počet vrstev
  - počet neuronů
  - koeficienty
- Návrh a trénování není exaktní věda
- Vyžaduje znalosti a experimentování
- Některé úlohy vhodnější pro tradiční metody
- Příklad:
  - font E-13B – vhodný klasifikátor s minimální vzdáleností
  - neuronové sítě vhodné pro obecnější úlohy (např. rukopis)



Aplikace neuronových sítí

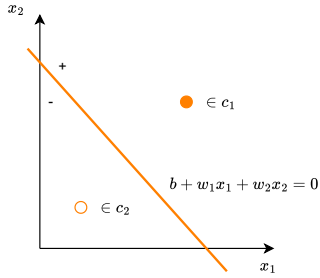
- Rozpoznávání řeči
- Filtry spamu
- Čtení PSČ
- Autonomní vozidla
- Robotika
- Objevování léčiv
- Predikce mutací DNA
- Porozumění přirozenému jazyku

Neuronové sítě ve zpracování obrazu

- Pomalejší nástup v klasifikaci obrazů
- **Konvoluční neuronové sítě (CNN)**
- LeCun et al. [1989] – čtení PSČ
- ImageNet 2012 – zásadní rozšíření
- Dnes preferovaný přístup

Perceptron

- **Perceptron** je základní model neuronu
- Učí se lineární rozhodovací hranici mezi třídami
- Použitelný pro **lineárně separovatelné** třídy
- Vstupem je vektor příznaků
- Výstupem je přiřazení do jedné ze dvou tříd
- Rozhodovací hranice v 2D je přímka
- Obecná rovnice: $y = ax + b$ $(y - ax - b = 0)$
- Body na jedné straně patří do jedné třídy, na druhé do druhé
- Rozšíření do vyšších dimenzí $\rightarrow$ rozhodovací hranice je **nadrovina**



Perceptron – matematický model

- Vektor vstupu: $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$
- Váhový vektor: $\mathbf{w} = [w_1, w_2, \ldots, w_n]^T$
- Skalární součin: $\mathbf{w}^T\mathbf{x}$
- Rozhodovací funkce: $\mathbf{w}^T\mathbf{x} + b = 0$
- Výstup:
  - > 0 jedna třída
  - < 0 druhá třída

Učení perceptronu

- Cílem je najít váhy $\mathbf{w}$ a bias b
- Iterativní proces
- Pro každý vzor:
  - správná klasifikace $\Rightarrow$ žádná změna
  - chybná klasifikace $\Rightarrow$ úprava vah
- Úprava závisí na třídě vzoru
- Pro vzor $\mathbf{x}(k)$:
  - pokud $\mathbf{x}(k) \in c_1$ a klasifikace je špatně:
    $\mathbf{w}(k+1) = \mathbf{w}(k) + \alpha\mathbf{x}(k)$
  - pokud $\mathbf{x}(k) \in c_2$ a klasifikace je špatně:
    $\mathbf{w}(k+1) = \mathbf{w}(k) - \alpha\mathbf{x}(k)$
- α je konstanta učení

Vlastnosti perceptronu

- Konverguje pro lineárně separovatelné třídy
- Najde řešení v konečném počtu kroků
- Neřeší nelineárně separovatelné problémy
- Výsledkem je lineární rozhodovací hranice

- Nelze řešit složitější problémy
- Příklad:
  - **XOR problém**
- Neexistuje lineární hranice oddělující třídy
- Vyžaduje vícevrstvé modely

Perceptron – příklad klasifikace

- Úloha: klasifikace siluet psích hlav
  - třídy: **kolie** (trida 1) a **buldok** (trida 2)
  - příznaky: kruhovitost, solidita

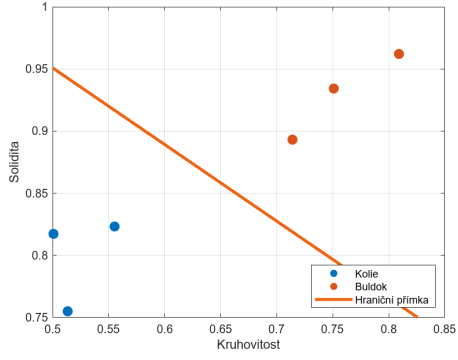
- Každý objekt reprezentován jako vektor $\mathbf{x} = [x_1, x_2]$
- Naučené váhy perceptronu:

$$\mathbf{w} = \begin{bmatrix} 0 \\ -1.4667 \\ -1.4071 \end{bmatrix}$$

- Lineární rozhodovací funkce
- **Rozhodovací pravidlo:** $\mathbf{w}^T\mathbf{x} > 0 \Rightarrow$ Kolie, $< 0 \Rightarrow$ Buldok
- **Po dosazení:** $-1.4667\, x_1 - 1.4071\, x_2 > 0$

Perceptron – rozhodovací hranice

- Rozhodovací hranice je **přímka**
- Váhy určují její orientaci (záporný sklon)
- Vykreslení je v původním prostoru příznaků
- Model byl učen na **standardizovaných datech**
- Proto přímka:
  - neprochází počátkem
  - ale správně odděluje třídy
- Kolie a buldoci jsou lineárně separovatelné třídy
- Učení konvergovalo po 2 epochách

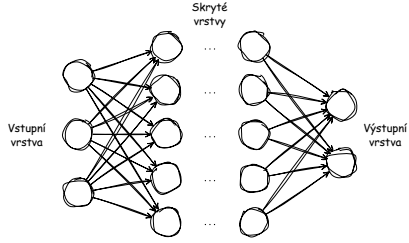


Perceptron – rozhodovací hranice

- Perceptron funguje jen pro **lineárně separabilní** data
- V praxi: data jsou často složitější
- Potřebujeme:
  - nelineární rozhodovací hranice
  - učení minimalizací chyby
- To vede k:
  - vícevrstvým neuronovým sítím

Vícevrstvá neuronová síť

- Skládá se z:
  - vstupní vrstvy
  - jedné nebo více skrytých vrstev
  - výstupní vrstvy
- Každá vrstva obsahuje neurony, které jsou plně propojené s další vrstvou



Role skrytých vrstev

Skryté vrstvy

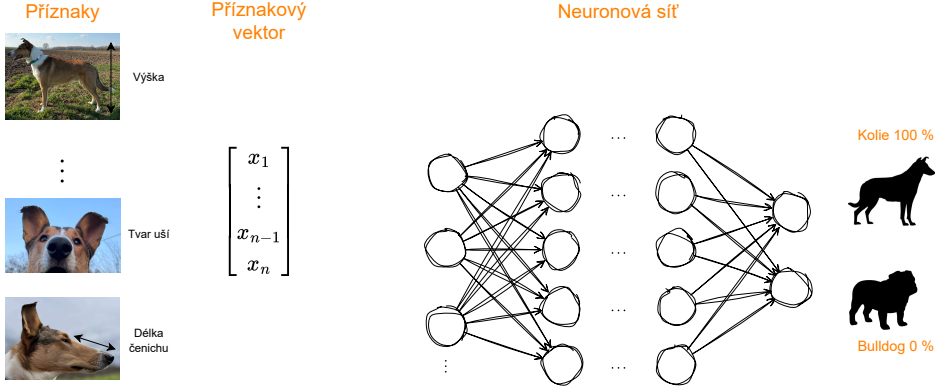
- Transformují data
- Postupná transformace:
  - vstup $\rightarrow$ vhodnější reprezentace
- **Feature transformation**

Výstupní vrstva

- Určuje výslednou klasifikaci
- Typ aktivace závisí na úloze
  - binární klasifikace
  - více tříd

Souvislost s obrazovou analýzou

- **Vstup**: příznaky z obrazu
- **Síť**: transformace příznaků
- **Výstup**: klasifikace objektu



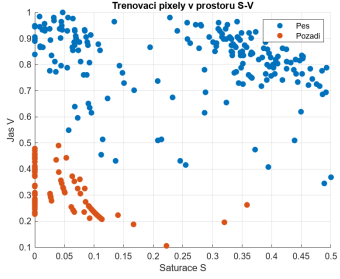
Příklad

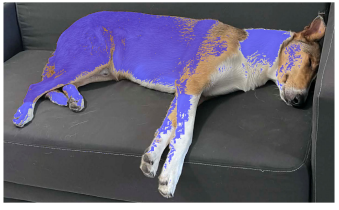
- **Vstup**: Barevný obraz
- **Síť**: klasifikátor pes vs. pozadí
- **Výstup**: segmentovaný pes

- Pro každý pixel vytváříme vektor příznaků:
  - RGB: R, G, B
  - HSV: H, S, V
  - prostorové souřadnice: x, y
- Vektor příznaků $\mathbf{x} = [R, G, B, H, S, V, x, y]$

Příklad

- Ručně vybrané pixely psa a pozadí (cca 50 bodů od každého)
- Každý pixel jeden trénovací vzor
- Lineární klasifikátor
- Výsledkem je binární maska určující třídu výsledku (1 pes, 0 pozadí)

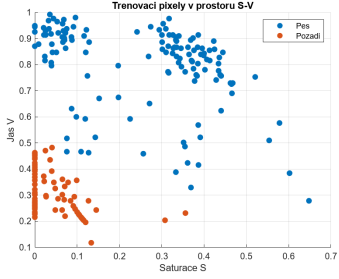


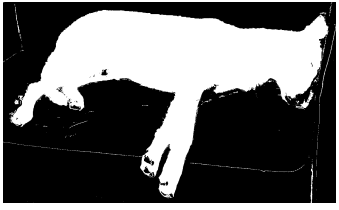


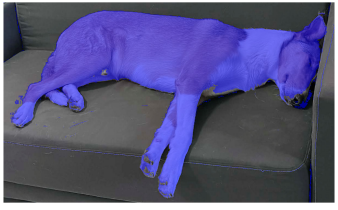
Epoch 1000, Performance 0.608

Příklad

- Ručně vybrané pixely psa a pozadí (cca 50 bodů od každého)
- Každý pixel jeden trénovací vzor
- 1 skrytá vrstva s 15 neurony
- Výsledkem je binární maska určující třídu výsledku (1 pes, 0 pozadí)







Epoch 1, Performance 1.45

Konvoluční neuronové sítě

- Dosud:
  - vstup = **vektor příznaků**
  - příznaky navrženy člověkem
- Alternativa:
  - vstup = přímo obraz
  - síť učí příznaky sama
- Problém vektorizace obrazu:
  - ztráta prostorových vztahů
  - ignorování struktur (hrany, rohy, textury . . .)
- obojí ale lze částečně řešit volbou příznaků

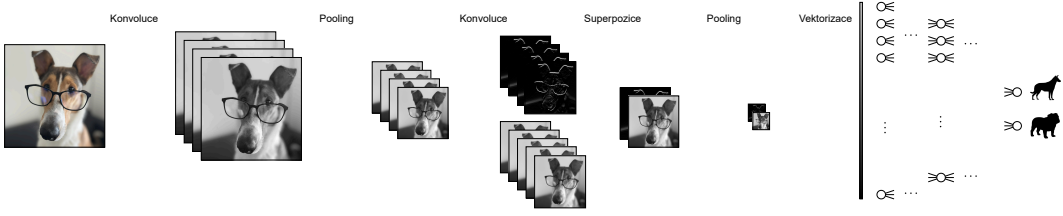
- **Konvoluční neuronové sítě** (CNN) – speciální typ neuronových sítí
- Navrženy pro zpracování obrazu

Konvoluční neuronové sítě

- **Architektura LeNet**
  - jednoduchá a přehledná
  - vhodná pro vysvětlení principů CNN
- Klíčový rozdíl oproti předchozím sítím:
  - CNN: vstup = **2D pole (obraz)**
  - plně propojené sítě: vstup = vektor
- Výpočty jsou podobné:
  - součet součinů
  - přičtení biasu
  - aplikace aktivační funkce
  - výstup $\rightarrow$ další vrstva

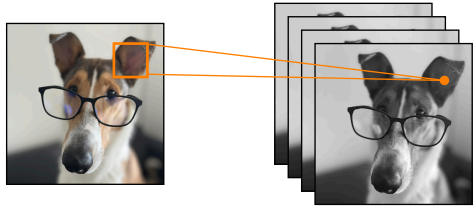
Konvoluční neuronové sítě

- **Rozdíly**:
  - vstup
  - učí se příznaky přímo ze syrových dat
  - v plně propojených NS je výstup každého neuronu v jedné vrstvě přiveden na vstup každého neuronu následující vrstvy
  - zde na každý vstup vrstvy je přiváděna jediná hodnota, která vznikla konvolucí nad prostorovým okolím ve výstupu předchozí vrstvy
  - dvourozměrná pole jsou mezi vrstvami subsamplována (pooling)



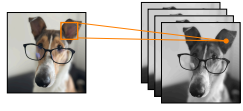
Konvoluční neuronové sítě

- Základem je konvoluce (tu známe), kterou provádíme nad celým vstupem
- Výstupem je opět pole velikosti vstupu
- Výsledkem v každé pozici (x, y) je skalární hodnota
- K této hodnotě přičteme bias a tato hodnota projde aktivační funkcí
- V terminologii CNN místo definice okolí konvoluce nazýváme **receptivní pole** (receptive fields)



Konvoluční neuronové sítě

- Nejprve provádíme konvoluci (posun receptivního pole) – součet součinu vah a hodnot definovaných tímto polem
- Sadu vah nazýváme **jádro**
- Počet kroků o které se receptivní pole posouvá, se nazývá **krok (stride)** (dosud jsme používali stride $= 1$)
- Použití vyššího stride vede k redukci dat
  změna stride z 1 na 2 sníží rozlišení obrazu na polovinu v každém směru
- Ke každé hodnotě konvoluce přičteme bias a výsledek projde aktivační funkcí
- Tato hodnota je následně předána na odpovídající pozici (x, y) vstupu následující vrstvy
- Po zopakování tohoto procesu pro všechny pozice vznikne dvourozměrná množina hodnot – **mapa příznaků** (feature map)
- Stejný bias se používá pro všechny pozice
- Na obrázku jsou 4 mapy příznaků (každá má jinou sadu vah a jiný bias)



Konvoluční neuronové sítě

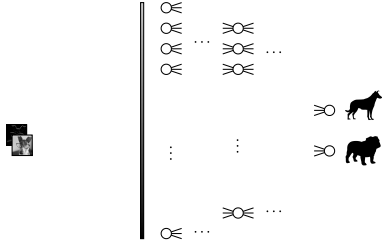
- Po konvoluci a aktivaci následuje **pooling** (subsampling)
- Snížení prostorového rozlišení
  - přispívá k invarianci vůči posunu
  - snižuje objem dat
- Pooling
  - rozdělení mapy příznaků na malé oblasti (typicky 2×2) – **poolingová okolí**
  - nahrazení těchto okolí jednou hodnotou
    - průměrování (average pooling)
    - maximum (max-pooling)
    - L2 pooling (odmocnina ze součtu čtverců)
- Každé mapě příznaků odpovídá **poolová mapa**

Konvoluční neuronové sítě

- Poolové mapy jsou pak vstupem do další vrstvy (nyní máme více vstupů)
- Výpočet jedné výstupní mapy:
  - pro každou vstupní mapu:
    - konvoluce s vlastním jádrem
    - přičtení biasu
    - aplikace aktivační funkce
  - vznikne jedna mezivýsledná mapa pro každý vstup
- Kombinace výsledků:
  - pro každou pozici (x, y):
    - máme více hodnot (z různých vstupů)
    - tyto hodnoty se sčítají
- Počet jader – (počet vstupů) $\times$ (počet výstupních map)

Konvoluční neuronové sítě

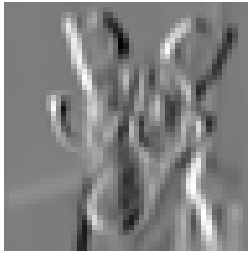
- Výsledné poolované mapy jsou převedeny na vektor (vektorizace)
- Vektor je vstupem do plně propojené neuronové sítě
- Počet výstupů sítě odpovídá počtu tříd
- Rozhodnutí je dáno maximální hodnotou výstupu



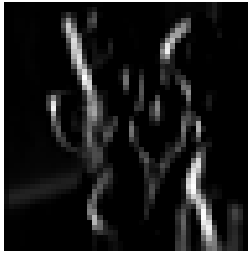
Ukázka komponent CNN

- Vstupní obraz postupně zpracujeme pomocí:
  - **konvoluce**
  - **aktivační funkce ReLU**
  - **max poolingu**

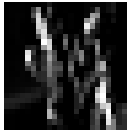
vstup



konvoluce

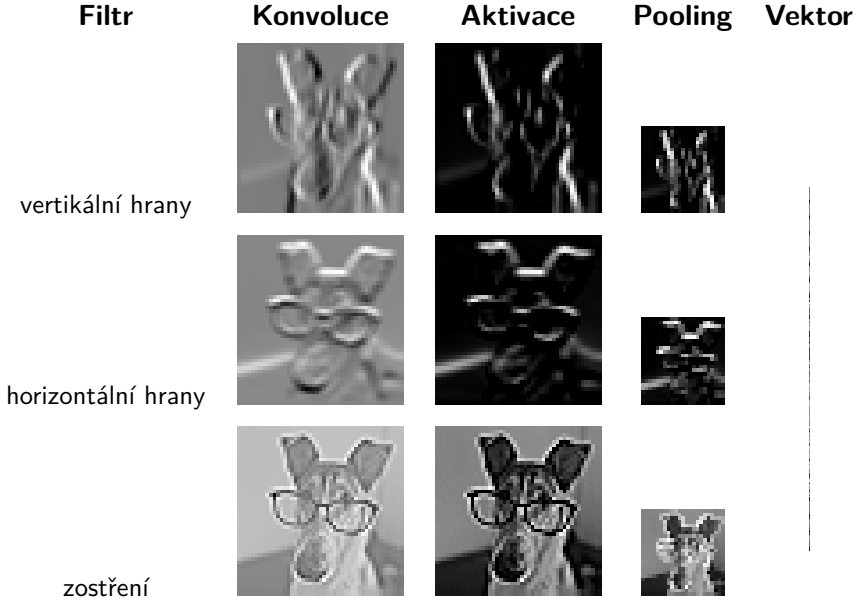


aktivace (ReLU)



pooling

Ukázka komponent CNN – různé filtry



Učení CNN

Krok 1

- Vstup: trénovací obrazy
  $a^{(0)}$ = vstupní obraz
- Inicializace:
  - váhy $w^{(l)}$ – malé náhodné hodnoty
  - biasy $b^{(l)}$

Krok 2 – dopředný průchod

- Pro každou vrstvu l a pozici (x, y):
  $z_{x,y}^{(l)} = w^{(l)} * a_{x,y}^{(l-1)} + b^{(l)}$
  $a_{x,y}^{(l)} = h(z_{x,y}^{(l)})$
- Postup:
  - konvoluce
  - bias
  - aktivace ($h(\cdot)$ = aktivační funkce (např. ReLU))
- Výstup: feature mapy

Učení CNN

Krok 3: zpětné šíření

- Výpočet chyby: od výstupu zpět do sítě
- Pro vrstvu l:
  $\delta^{(l)} = \left(\delta^{(l+1)} * \text{rot180}(w^{(l+1)})\right) \cdot h'(z^{(l)})$
- $\delta^{(l)}$ – chyba ve vrstvě l, určující, jak upravit váhy
- Význam:
  - rotace jádra
  - konvoluce chyby
  - derivace aktivace

Krok 4: aktualizace parametrů

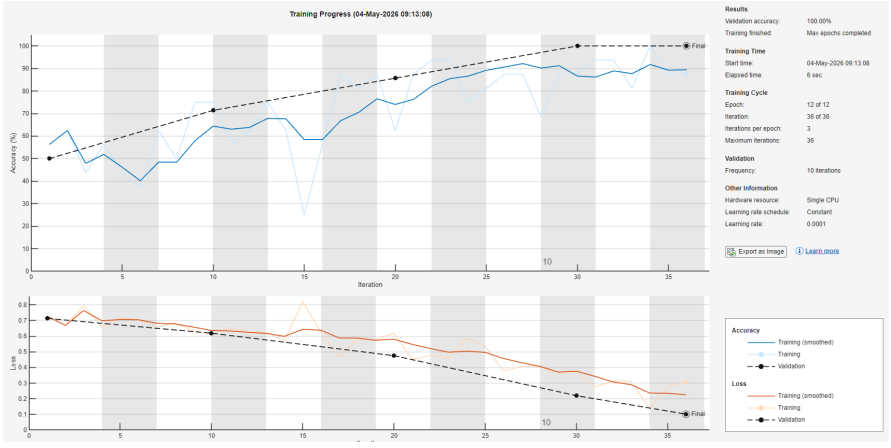
- Aktualizace vah:
  $w^{(l)} \leftarrow w^{(l)} - \alpha\, \delta^{(l)} * \text{rot180}(a^{(l-1)})$
- Aktualizace biasu:
  $b^{(l)} \leftarrow b^{(l)} - \alpha \sum_{x,y} \delta^{(l)}_{x,y}$
- $\alpha =$ krok učení

Poznámka: pooling a vektorizace

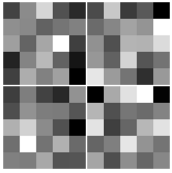
- Pooling:
  - při backprop:
  - nutné **upsampling**
- Vektorizace:
  - 2D mapy $\rightarrow$ vektor
  - vstup do plně propojené sítě
- Jeden průchod všemi daty – **epocha**

Ukázka učení CNN: kolie vs. buldok

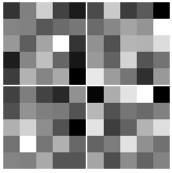
- Úloha:
  - klasifikace obrazů: **kolie** vs. **buldok**
- Architektura:
  - 2 konvoluční vrstvy (4 a 8 filtrů)
  - pooling + plně propojená vrstva



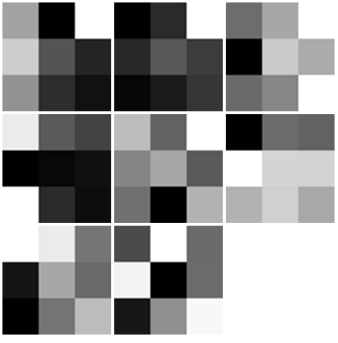
Ukázka učení CNN: kolie vs. buldok



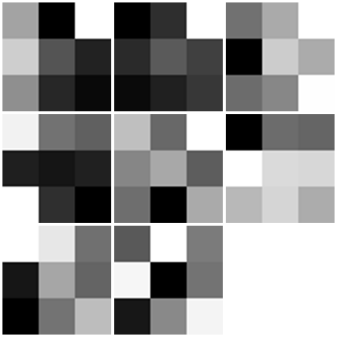
před učením



po učení



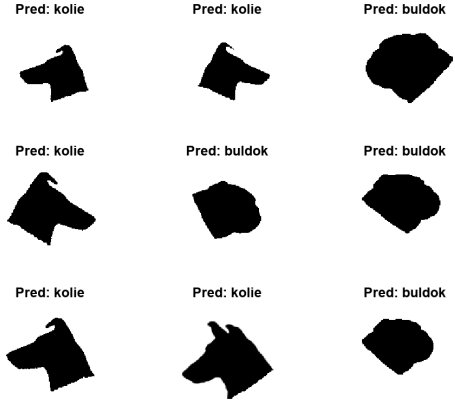
před učením



po učení

Ukázka učení CNN: kolie vs. buldok

- **Výsledek**:
  - přesnost roste během učení
  - validační přesnost až 100 %
  - rychlá konvergence (cca 6 s)
- **Interpretace**:
  - síť se učí rozlišovat příznaky
  - konvoluční masky se adaptují na data
  - výsledek závisí na velikosti datasetu



Transfer learning

- **Myšlenka:**
  - využití již natrénované sítě
  - místo trénování od začátku
- **Postup**:
  - vezmeme předtrénovanou CNN (např. ImageNet)
  - ponecháme konvoluční vrstvy
  - nahradíme klasifikační část
- **Proč to funguje**:
  - nižší vrstvy:
    - obecné příznaky (hrany, textury)
  - vyšší vrstvy:
    - specifické pro danou úlohu
- **Výhody**:
  - méně dat
  - rychlejší učení
  - lepší generalizace

Detekce objektů

- **Úloha:**
  - najít objekty v obraze
  - určit jejich **polohu** a **třídu**
- Výstup:
  - bounding box
  - klasifikace objektu
- Rozdíl oproti klasifikaci:
  - klasifikace: 1 label pro celý obraz
  - detekce: více objektů + jejich pozice
- Příklady:
  - YOLO
  - Faster R-CNN

YOLO (You Only Look Once)

- **Základní myšlenka:**
  - detekce v **jednom průchodu sítí**
- Postup:
  - obraz rozdělen na **mřížku**
  - pro každou buňku:
    - predikce polohy (box)
    - pravděpodobnost objektu
    - třída
- Vlastnosti:
  - rychlá detekce (real-time)
  - vše řešeno jednou CNN
- Nevýhoda:
  - horší přesnost pro malé objekty

Segmentace obrazů

- **Úkol:** přiřadit třídu každému pixelu
- Typy:
  - **semantic segmentation**
    - každý pixel má label
  - **instance segmentation**
    - rozlišení jednotlivých objektů
- Výstup:
  - maska stejné velikosti jako obraz

Generování obrazů

- Přístupy:
  - **GAN (Generative Adversarial Networks)**
  - **diffusion models**
- Princip:
  - model se učí rozdělení dat
  - generuje nové vzory podobné trénovacím
- Využití:
  - syntéza obrazů
  - augmentace dat
  - generativní AI