18NES1 Neuronové sítě 1 - Konvoluční neuronové sítě

18NES1 - 16. hodina, LS 2024/25

Zuzana Petříčková

10. dubna 2025

Co jsme probírali minule

Neuronové sítě a zobecňování

- Techniky pro porovnání modelů (obecně i v případě, že máme málo trénovacích dat)
- Techniky pro prevenci přeučení
- Ukázky a příklady

Zbývá dokončit (minule jsme nestihli): příklady images_simple_mlp_autoencoder.ipynb

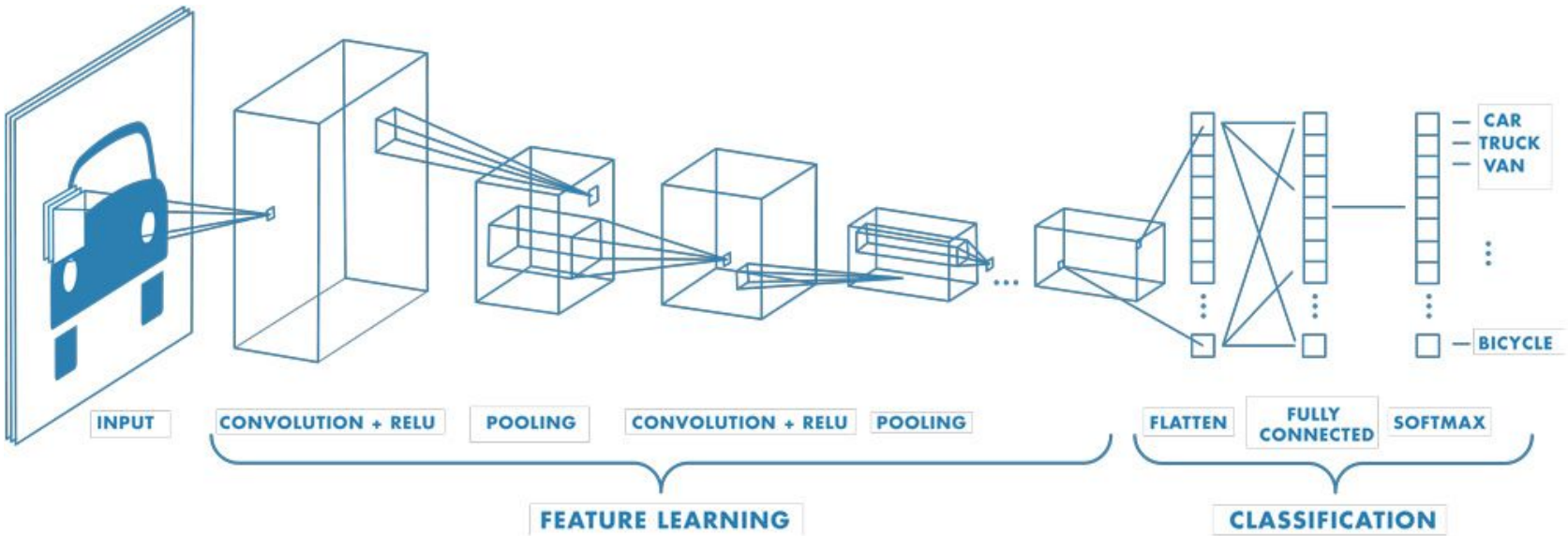
- Jednoduchý autoencoder (architektura, kdy je vstup a požadovaný výstup stejný)
- Ilustrace toho, jak může učení ovlivnit regularizace (zde - rozšíření trénovací množiny o další vzory)
- Zkuste experimentovat s trénovací množinou i s počtem neuronů a podívejte se na výsledky

Dobrovolný úkol

images_simple_mlp_autoencoder.ipynb

- Použijte vlastní obrázky nebo fotografie a upravte notebook tak, aby výrazně pozměnil jejich barevnost (ale jinak než v původním notebooku)
- Zvolte si vlastní trénovací obrázek a vlastní rozšíření trénovací množiny
- Je možné, že bude třeba nastavit počet neuronů ve skryté vrstvě na míru danému obrázku.
- Pošlete mi pozměněný notebook, původní a pozměněné obrázky.

Dnešní hodina: Úvod do konvolučních neuronových sítí

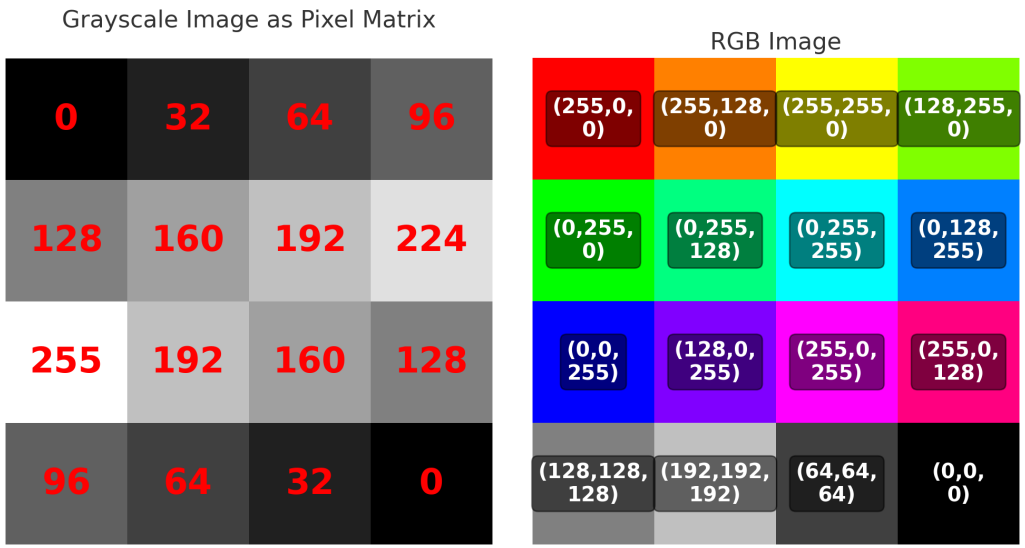


Zdroj :

https://matlabacademy.mathworks.com/details/deep-learning-onramp/deeplearning

Připomenutí: digitální reprezentace obrazu

- digitální obraz = matice (tenzor) pixelů
- každý pixel („picture element") popisuje barvu na konkrétní pozici obrazu



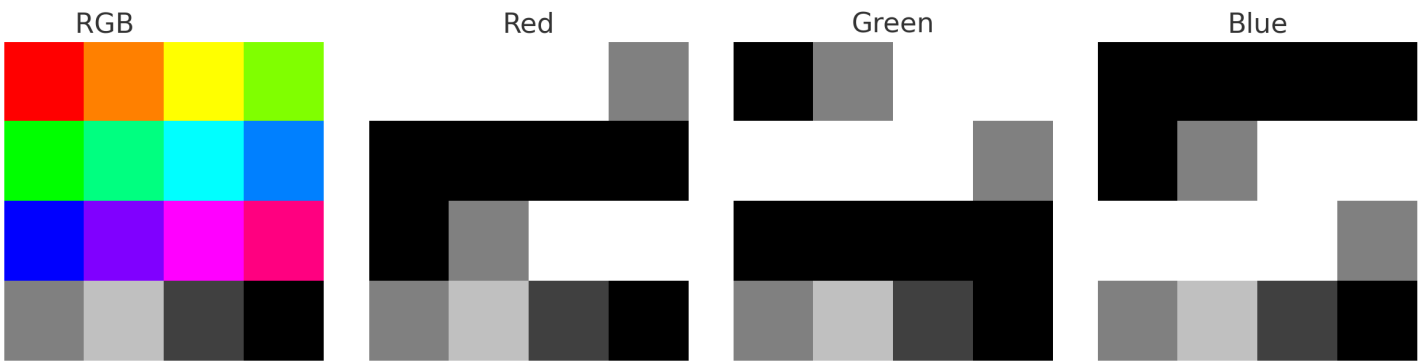
Obraz v odstínech šedi

- každý pixel je číslo udávající jas (např. 0 = černá, 255 = bílá)
- pro účely stroj. učení se hodnoty normalizují do intervalu $[0, 1]$

Připomenutí: digitální reprezentace obrazu

Barevný obraz (RGB)

- každý pixel má 3 složky: R (red), G (green), B (blue)
- obraz je reprezentován jako 3D tenzor tvaru (výška x šířka x 3)
- třem barevným složkám obrazu se říká kanály (channels)

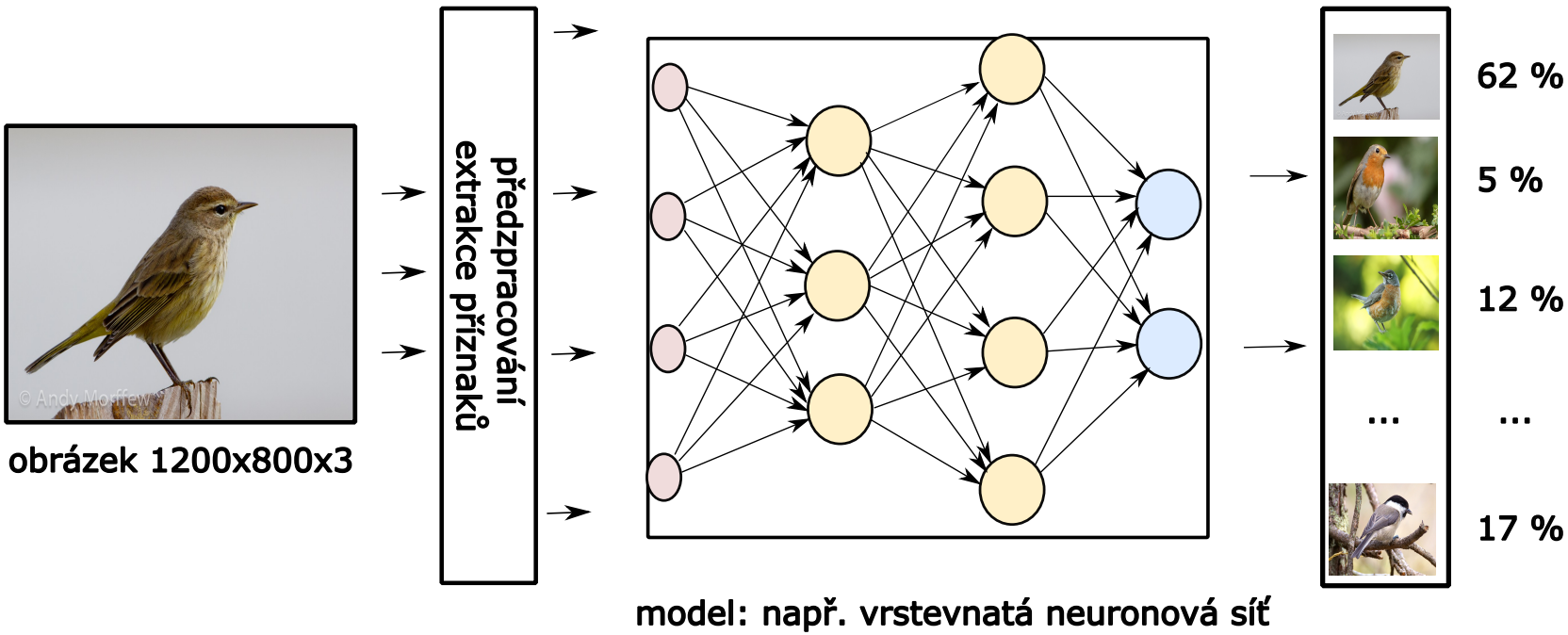


Ukázka: convolution_introduction.ipynb

Motivační příklad: klasifikace obrázků
Rozpoznávání pěvců

Motivační příklad: Rozpoznávání pěvců

Klasický přístup ve strojovém učení

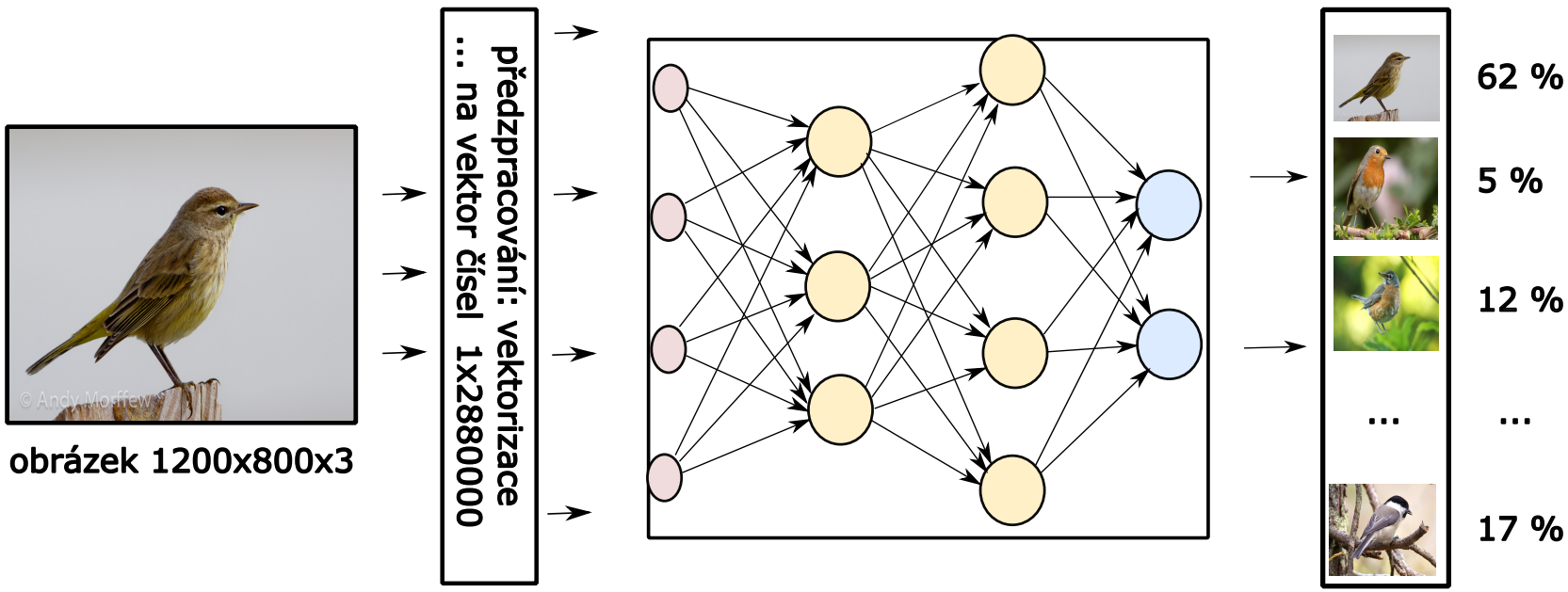


Extrakce příznaků

- detekce hran, LBP histogramy,...
- ztráta informace, poměrně náročné na návrh

Motivační příklad: Rozpoznávání pěvců

Co takhle učit neuronovou síť přímo?

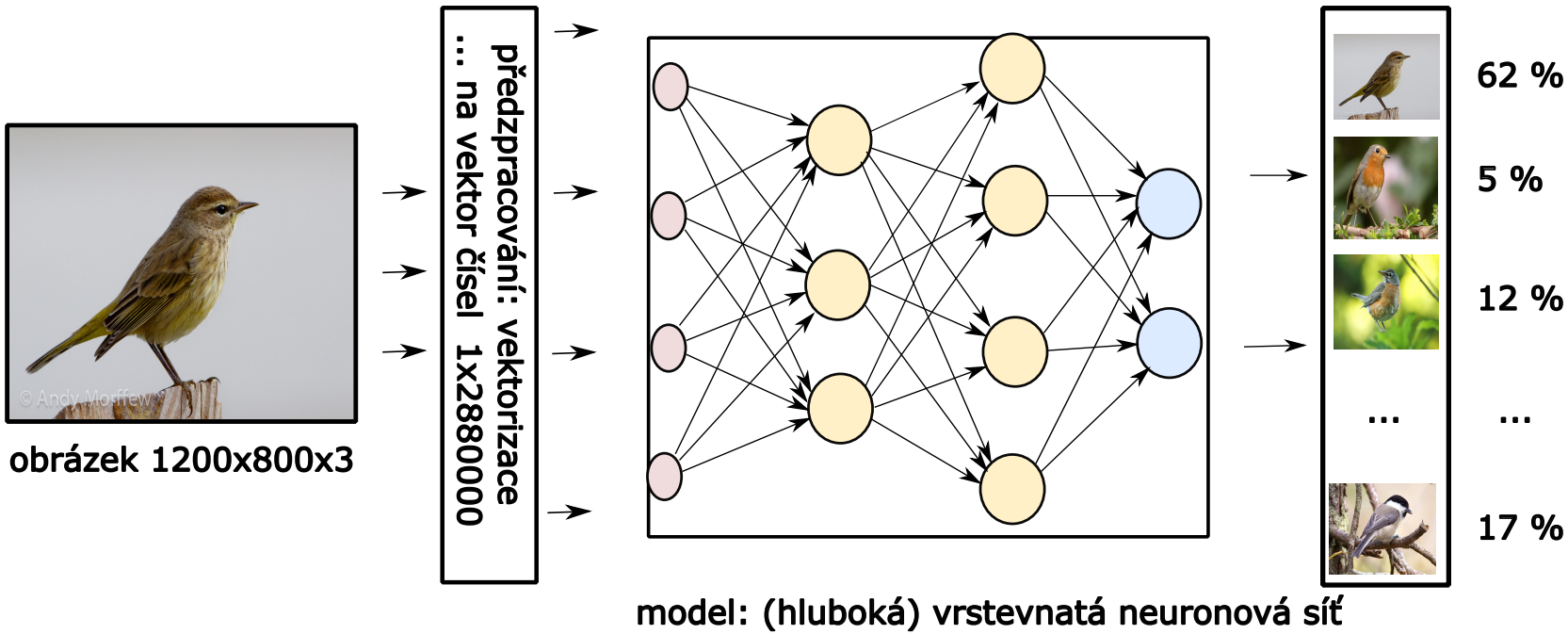


Princip hlubokého učení

- extrakci příznaků necháme na modelu

Motivační příklad: Rozpoznávání pěvců

Co takhle učit neuronovou síť přímo?

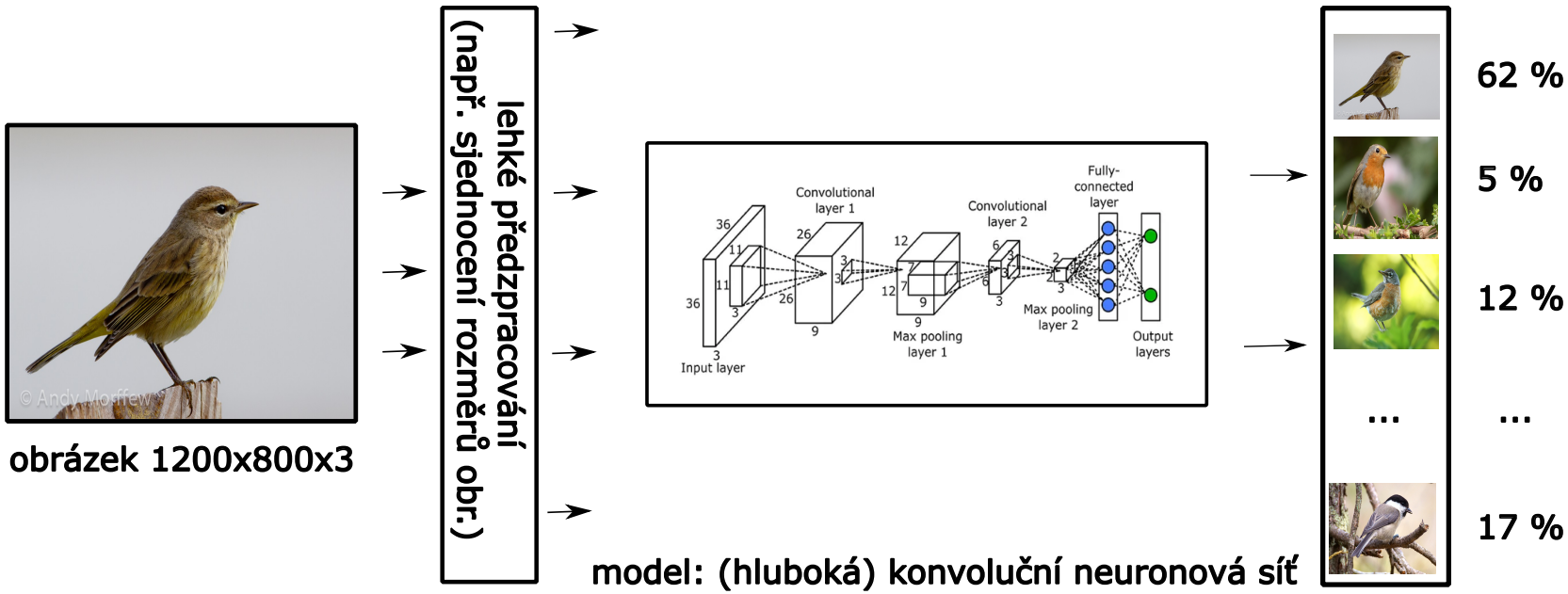


Nevýhody klasického přístupu (vrstevnatá neuronová sít, tj. pouze plně propojené vrstvy):

- velký počet příznaků
- ztráta informace o vzájemné poloze pixelů
- obtížné učení

Motivační příklad: Rozpoznávání pěvců

Jak by to šlo udělat lépe?

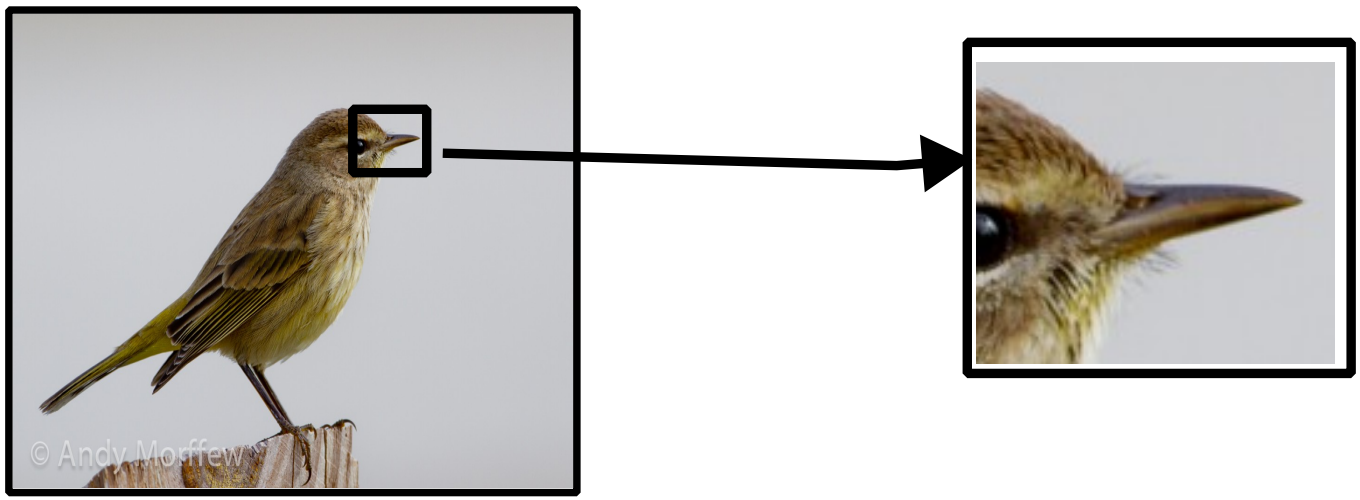


Konvoluční neuronová síť:

- neuronová síť s konvolučními vrstvami
- bere v potaz prostorové rozložení pixelů
- méně parametrů a snadnější učení než v případě plně propojených vrstev

Motivační příklad: Rozpoznávání pěvců

V datech jsou vzory: např. zobáček



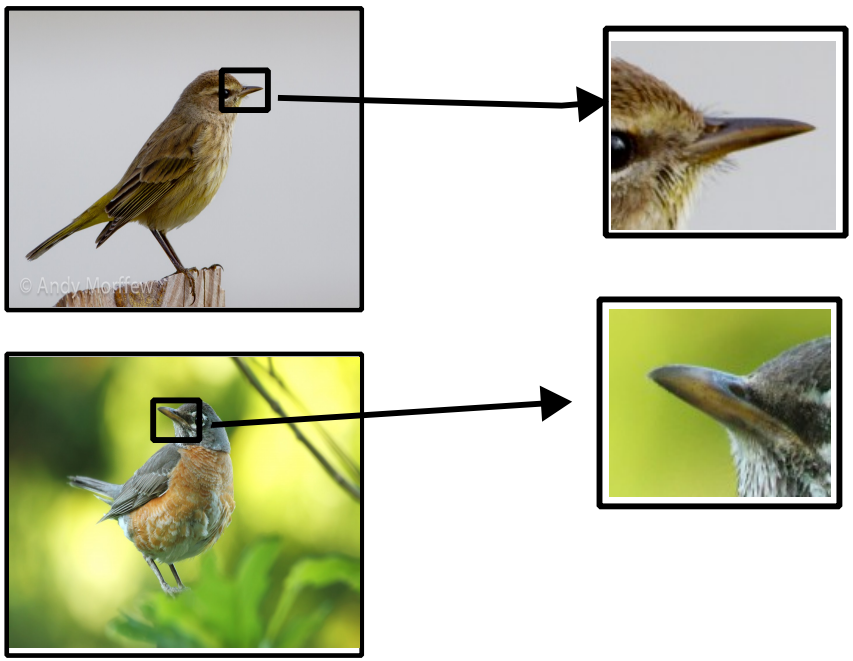
Co vytvořit detektor zobáčku:

- jednoduchý model (např. jednovrstvá neuronová síť), co najde na obrázku zobáček

Ale: zobáček může být na různých místech v obrázku

Motivační příklad: Rozpoznávání pěvců

V datech jsou vzory: např. zobáček

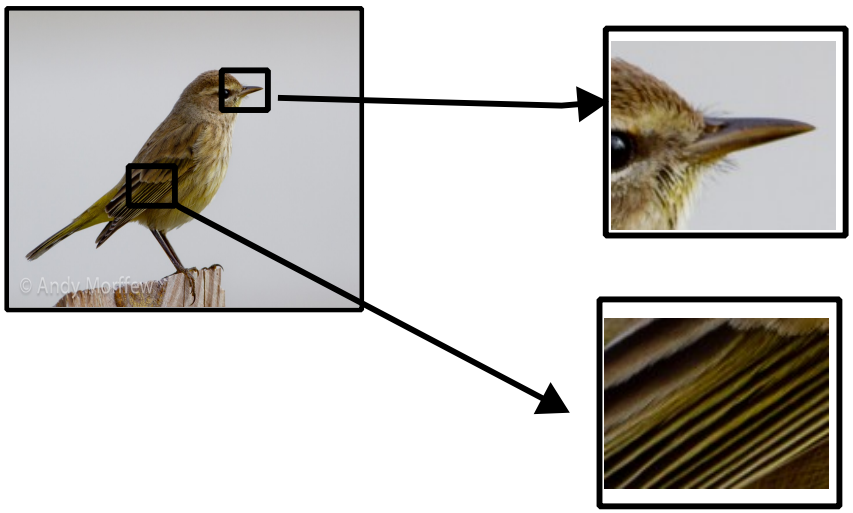


zobáček může být na různých místech v obrázku

- detektor by měl najít zobáček v libovolném obrázku a na libovolném místě v obrázku
  $\rightarrow$ detektor se bude pohybovat zpracovávaným obrázkem

Motivační příklad: Rozpoznávání pěvců

V datech jsou různé vzory:



Myšlenka

- vytvořím množinu detektorů pro různé příznaky (vzory)
- detektory by měly najít příznak v libovolném obrázku a na libovolném místě v obrázku → detektory se budou pohybovat zpracovávaným obrázkem
- detektory budou tvořit počáteční vrstvy konvoluční neuronové sítě

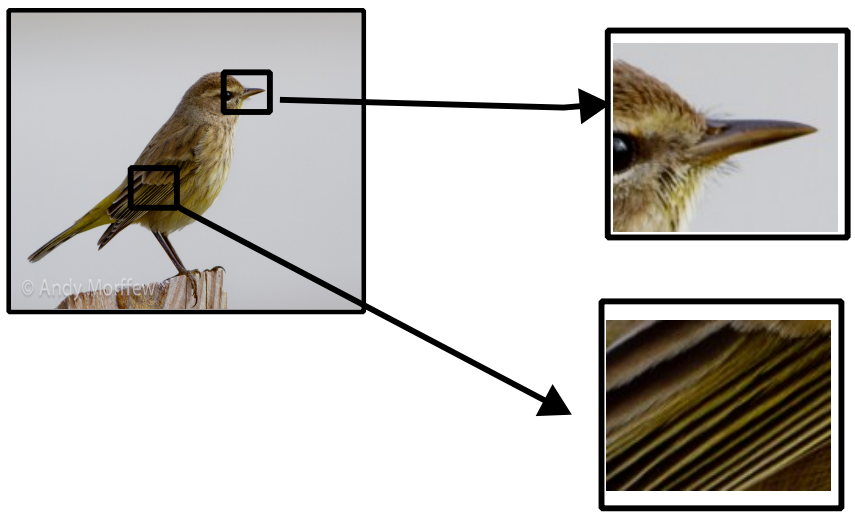
Konvoluční neuronová síť

- neuronová síť obsahující konvoluční vrstvy

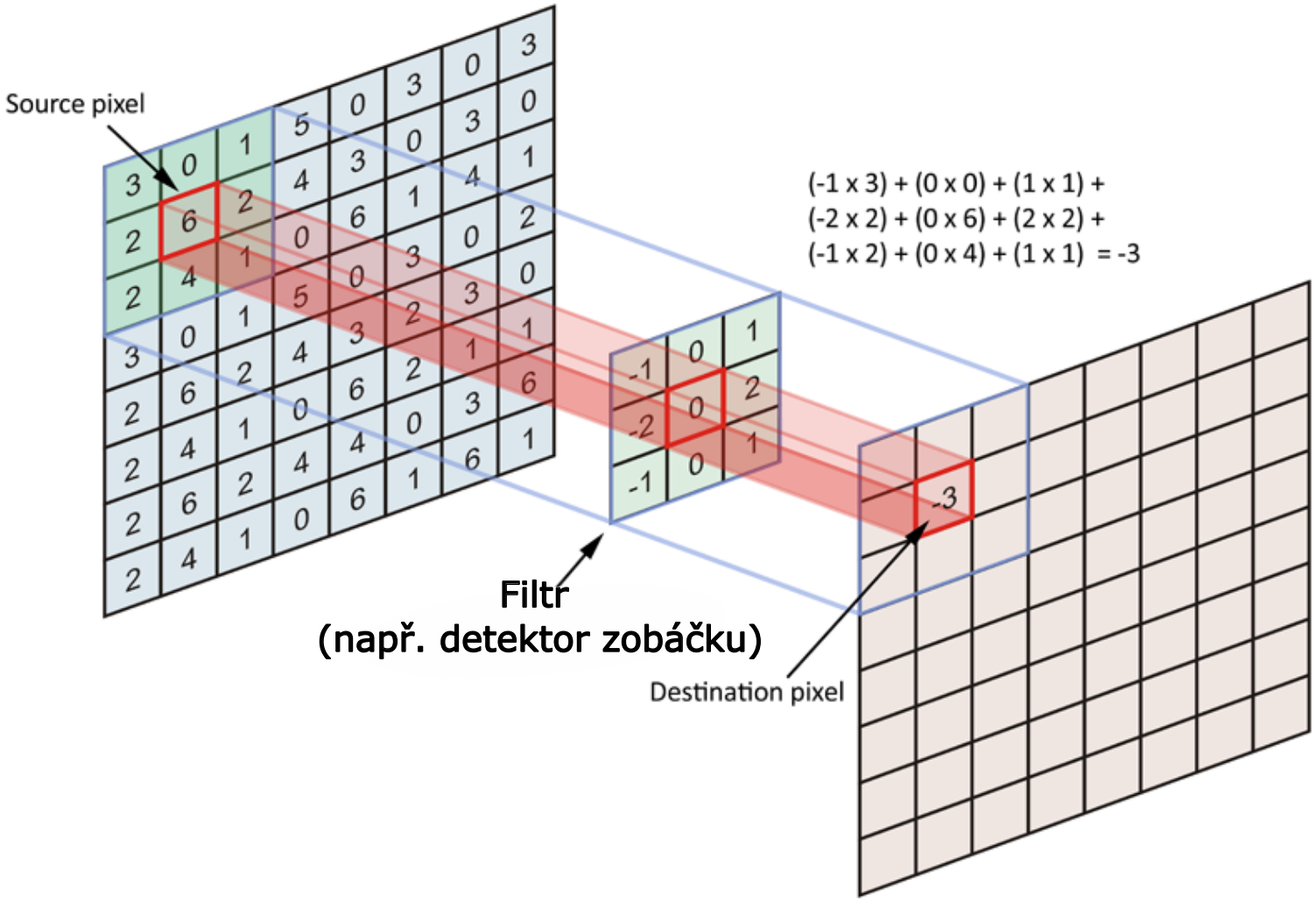
Konvoluční vrstva

- tvořená skupinou filtrů (jader, detektorů)
- filtry provádí operaci konvoluce nad vstupním obrázkem
- do další vrstvy postupuje výsledek konvoluce (matice příznaků)

Filtr = detektor nějakého vzoru (příznaku) v obrázku



Operace konvoluce



https://i.stack.imgur.com/YDusp.png

Operace konvoluce

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

Obrázek 6x6 (zjednodušeně)

Konvoluční vrstva

1	-1	-1
-1	1	-1
-1	-1	1

Filtr 1 (3x3)

-1	1	-1
-1	1	-1
-1	1	-1

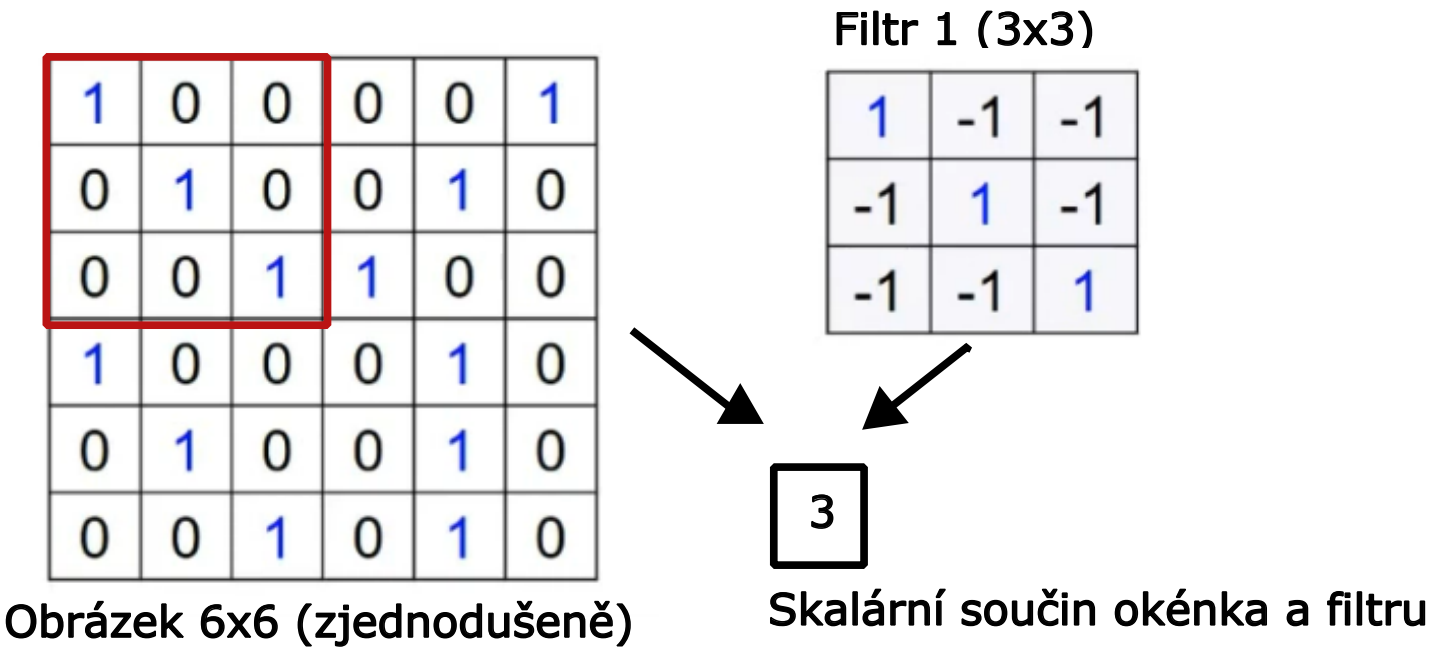
Filtr 2 (3x3)

. . .

- každá konvoluční vrstva obsahuje několik filtrů
- každý filtr detekuje vzor (příznak) velikosti 3x3 pixely (např. diagonální hrana, svislá hrana,...)

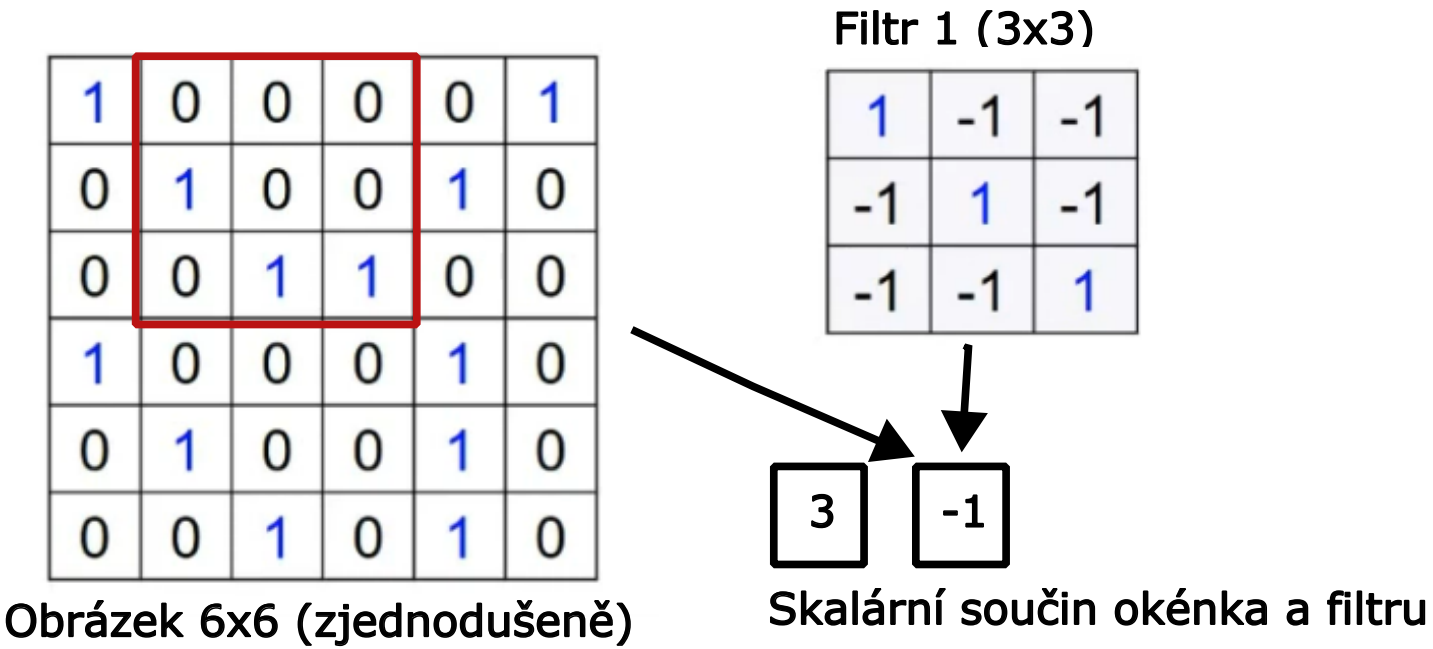
zdroj příkladu: Petr Doležel: Konvoluční neuronová síť, https://www.youtube.com/watch?v=-2vEi-Aa0FA

Operace konvoluce



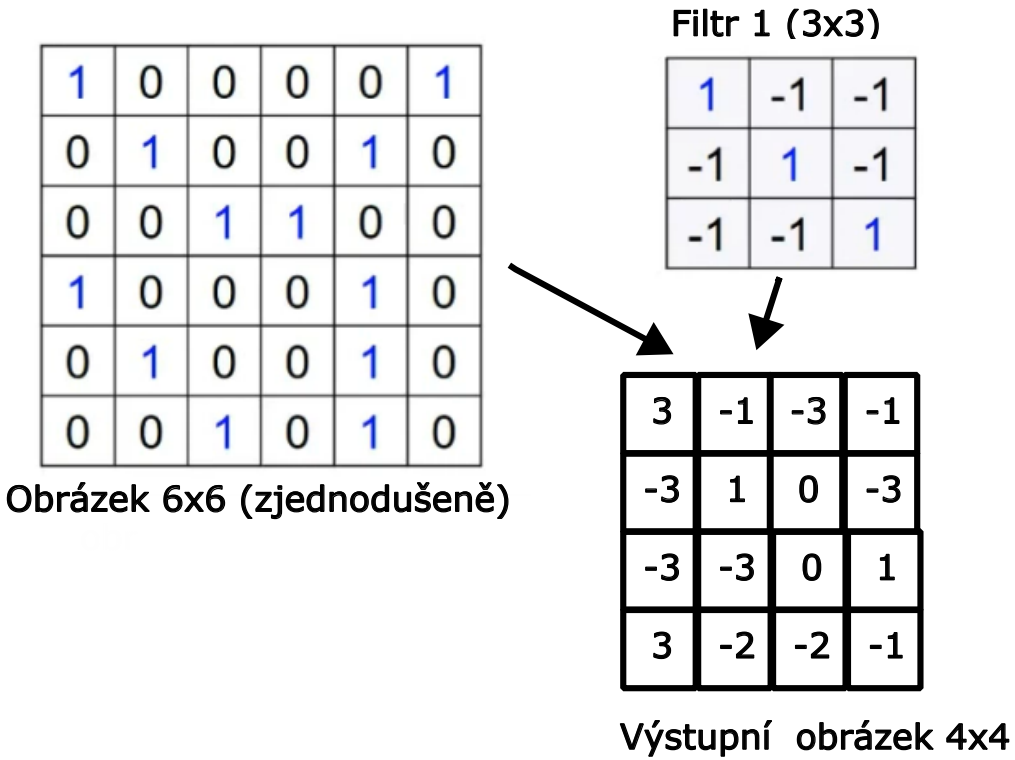
- spočteme skalární součin:
  $y = \sum_{i=1}^{9} w_i x_i + b$ (pro linearizované matice)

Operace konvoluce



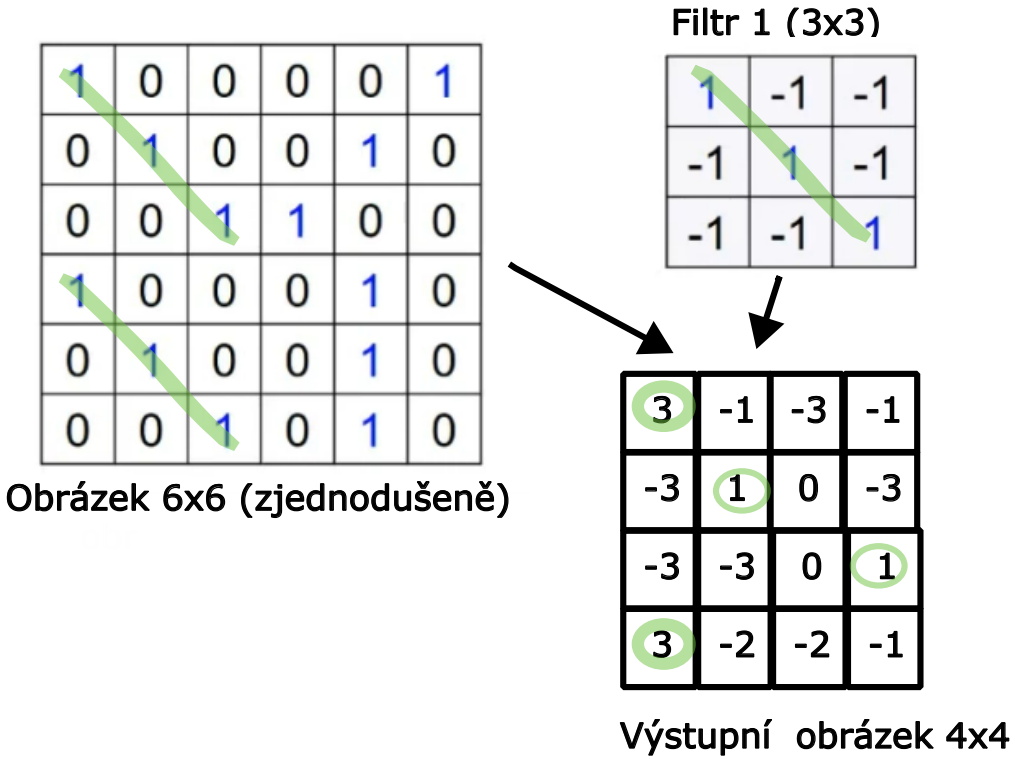
- okénko posunu a opět spočteme skalární součin

Operace konvoluce



- postupným posouváním okénka aplikujeme filtr na celý obrázek
- získáme nový obrázek ... **matice příznaků (feature map)**

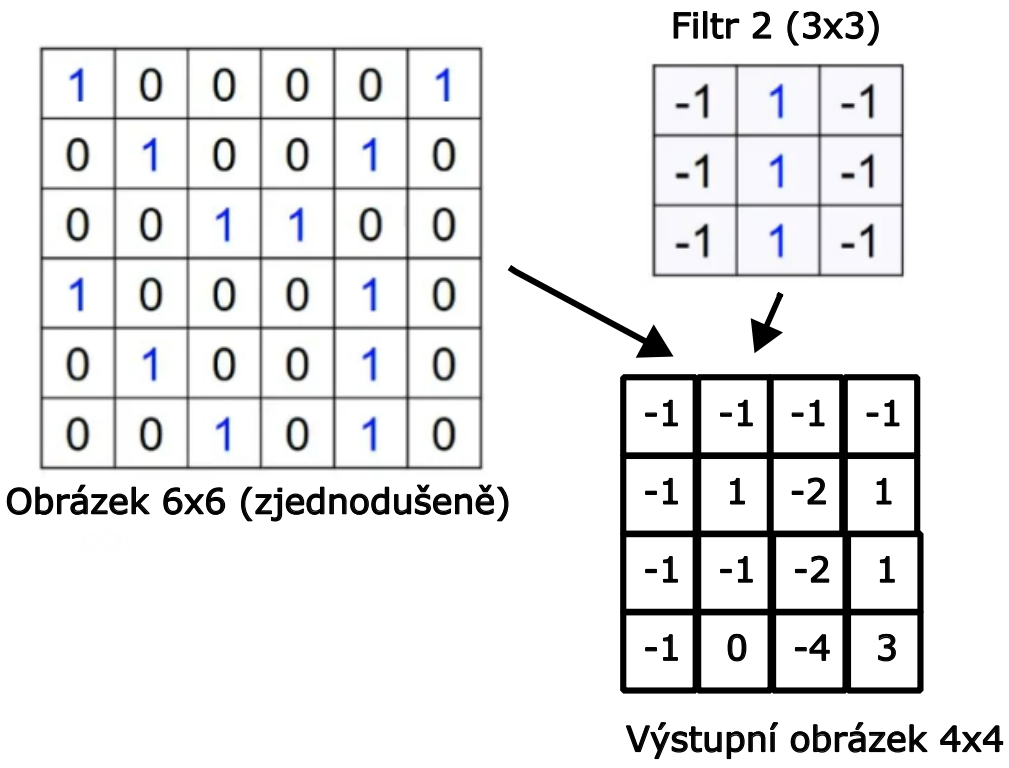
Operace konvoluce



Matice příznaků (feature map)

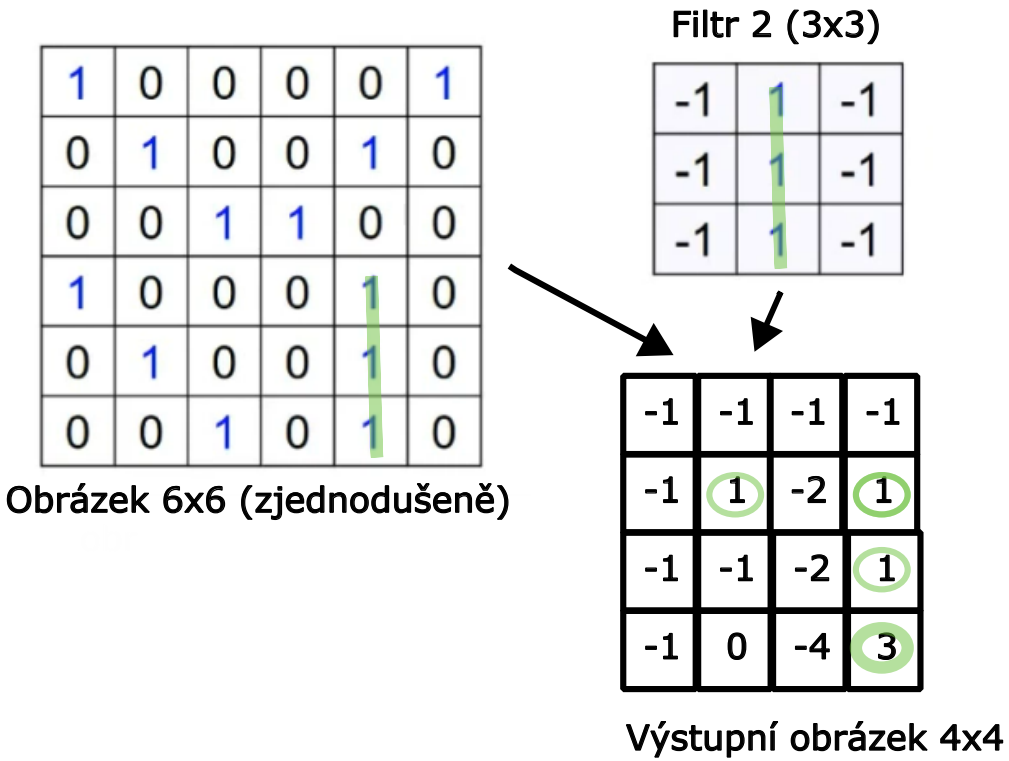
- říká, kde (a v jaké míře) se v původním obrázku vyskytuje vzor reprezentovaný filtrem
- zde: hranový filtr pro diagonální hranu

Operace konvoluce



- podobně mohu aplikovat druhý filtr

Operace konvoluce

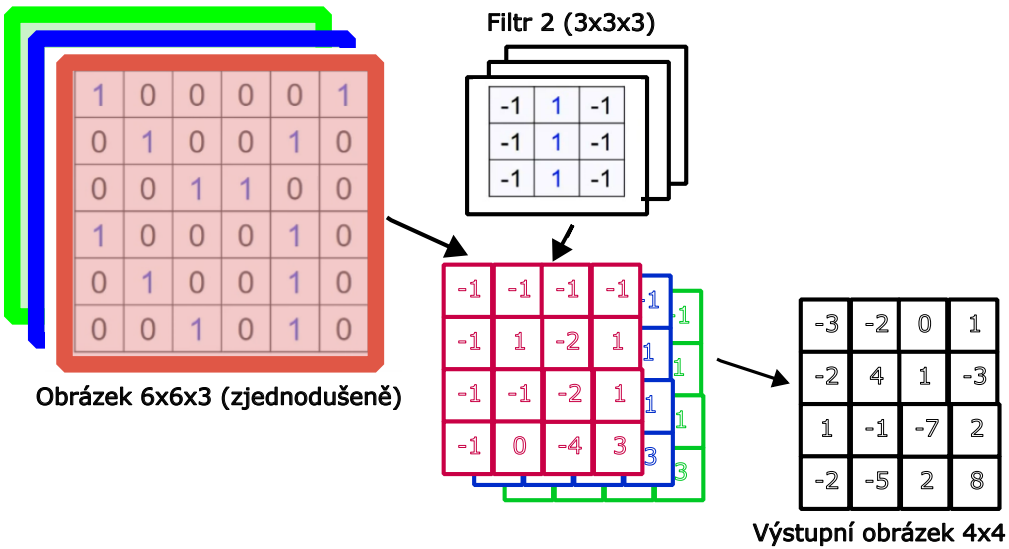


Druhá matice příznaků (feature map)

- říká, kde (a v jaké míře) se v původním obrázku vyskytuje vzor reprezentovaný filtrem
- zde: hranový filtr pro svislou hranu

Operace konvoluce

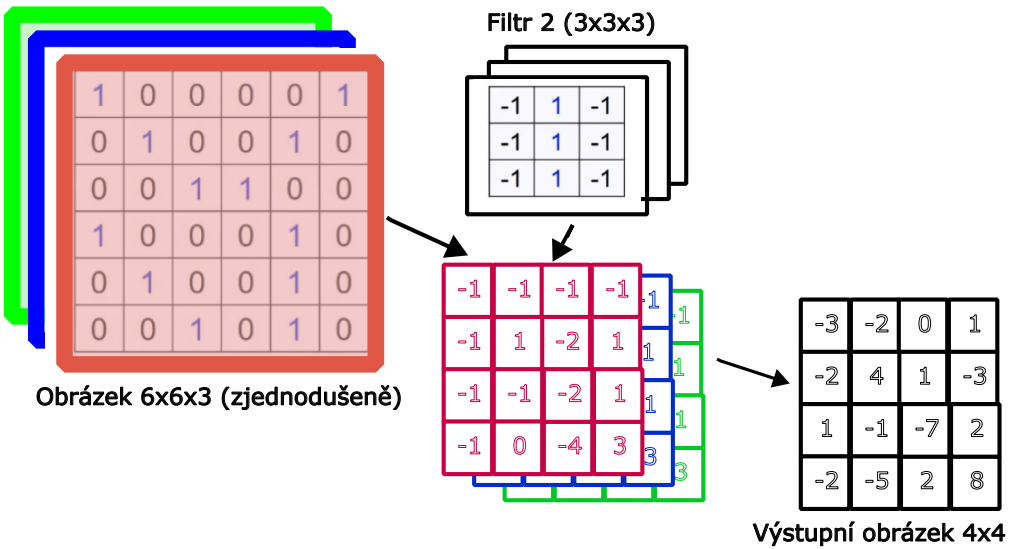
Barevný obraz: 3 kanály (channels) R, G, B



- Každý filtr má váhy pro **všechny vstupní kanály (R, G, B)**
- Výpočet: provede se konvoluce přes každý kanál zvlášť a výsledky se **sečtou**
- Každý filtr vytvoří jednu výstupní mapu příznaků (feature map)
- Počet filtrů určuje **počet výstupních kanálů**

Operace konvoluce

Barevný obraz: 3 kanály (channels) R, G, B



Reprezentace konvoluční vrstvy

- 4-dimenzionální tensor tvaru $u \times u \times c \times f$
  - $u \times u$ – prostorová velikost jednoho filtru (pro jeden kanál)
  - c – počet vstupních kanálů (např. 3 pro RGB)
  - f – počet filtrů, tedy počet výstupních kanálů

Operace konvoluce

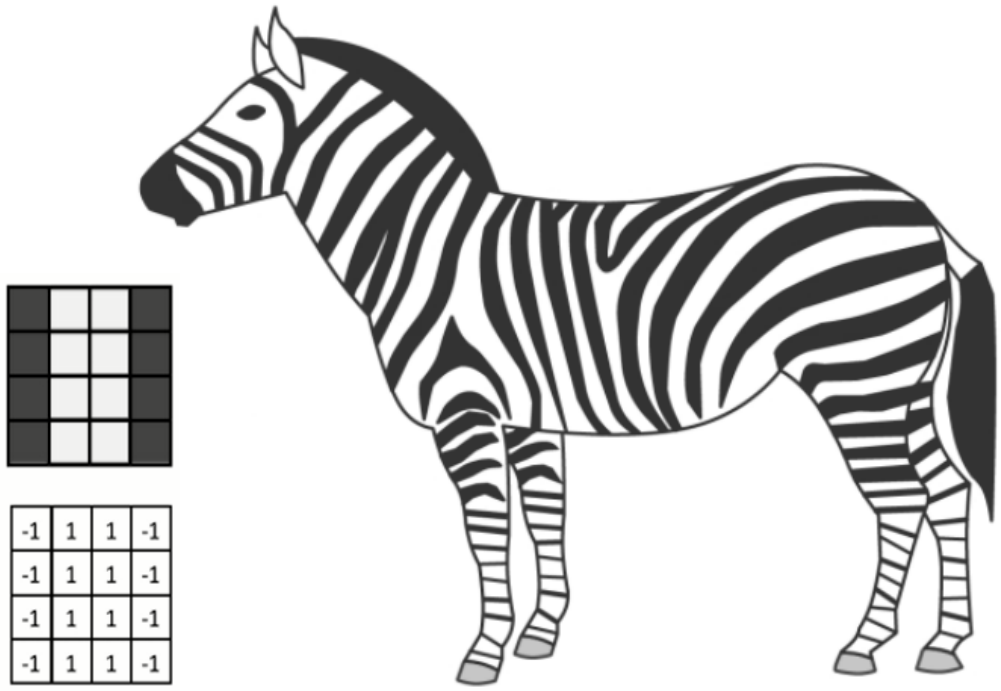
Příklad: zebra

Zdroj příkladu:

https://matlabacademy.mathworks.com/details/deep-learning-onramp/deeplearning

Operace konvoluce

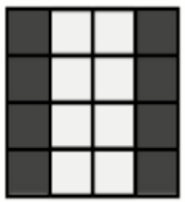
Příklad: zebra



- aplikujeme filtr pro detekci svislého proužku

Operace konvoluce

Příklad: zebra

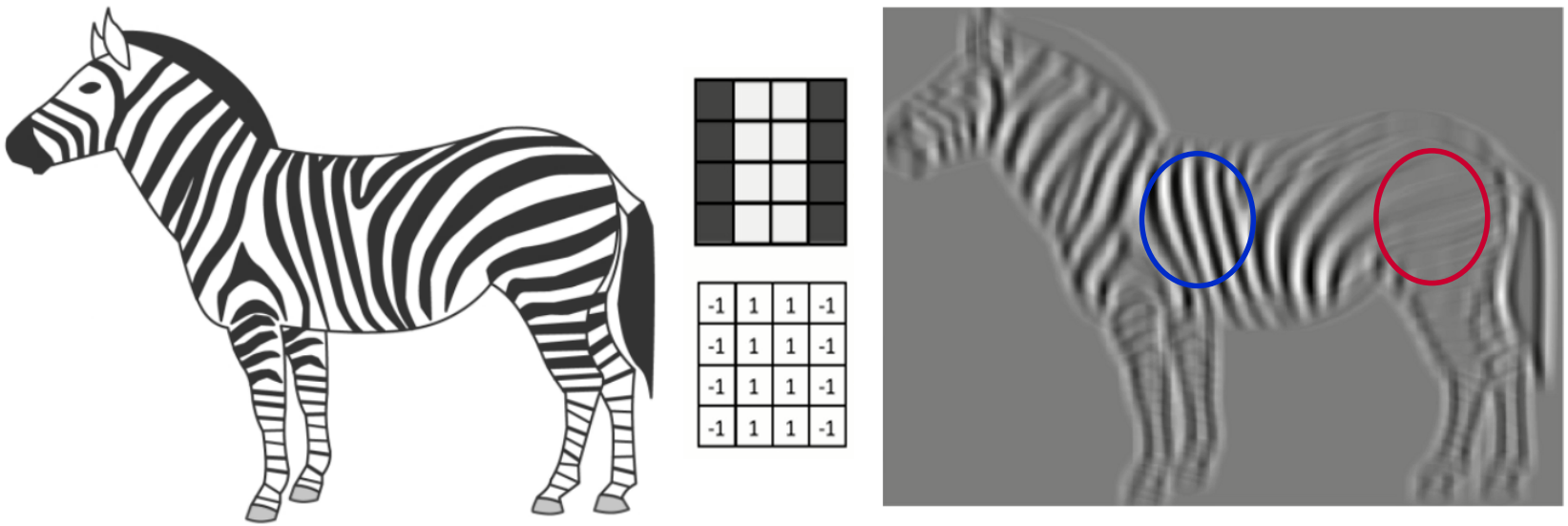


-1	1	1	-1
-1	1	1	-1
-1	1	1	-1
-1	1	1	-1

- takto vypadá výsledek aplikace filtru

Operace konvoluce

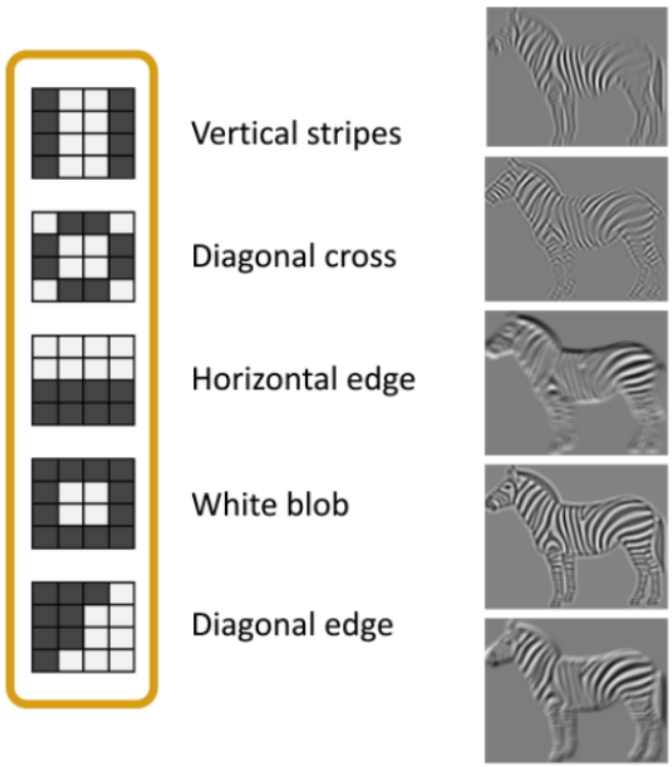
Příklad: zebra



- snadno rozeznáme oblasti, kde je vzor výrazně zastoupen a oblasti, kde není

Operace konvoluce

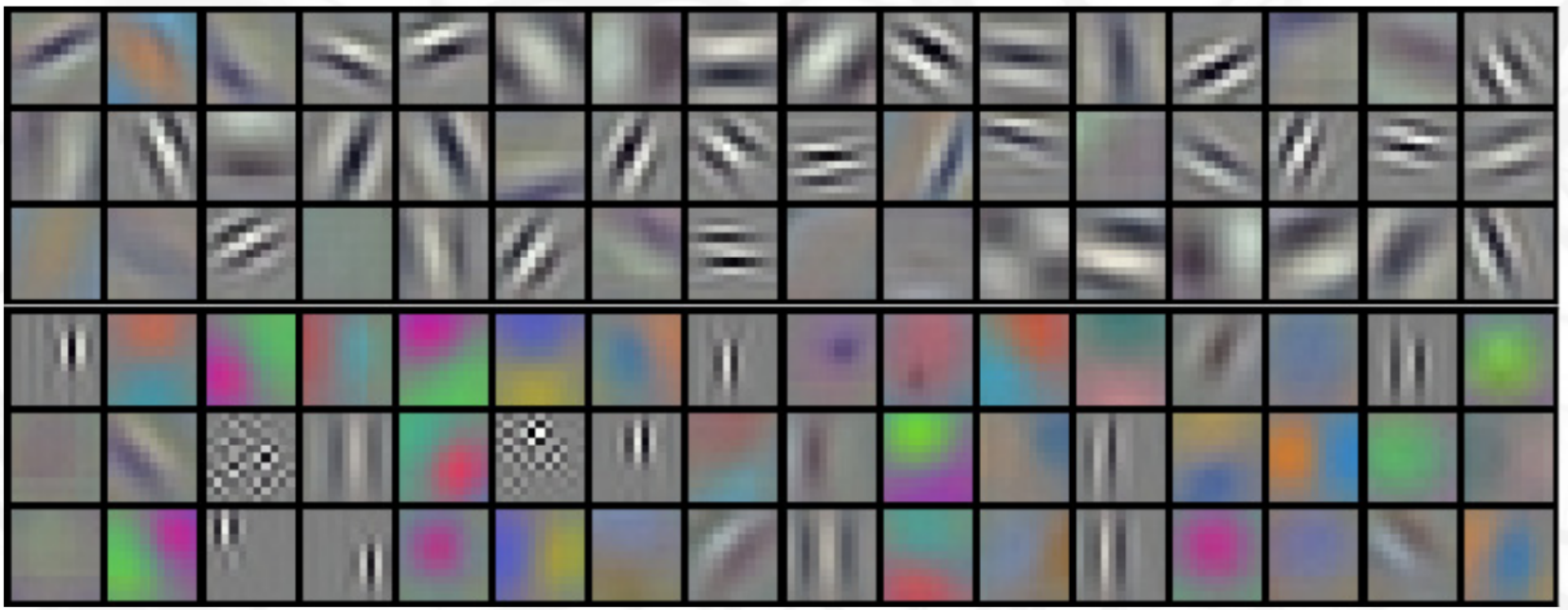
Příklad: zebra



- ukázky dalších filtrů a výsledných matic příznaků

Operace konvoluce

- **Ukázka:** 96 filtrů 11x11x3 v první konvoluční vrstvě u AlexNet



Zdroj obrázku: Alex Krizhevsky et al.: "ImageNet Classification with Deep Convolutional Neural Networks", 2012, , Figure 3

Konvoluční operace mimo neuronové sítě

Co se stane, když na stejný obrázek použijeme různé filtry?

- Každý filtr zvýrazňuje jiné vlastnosti obrazu:
  - hranové filtry (např. svislé, vodorovné, diagonální hrany)
  - rozmazání (blur) – potlačuje detaily a šum
  - zostření – zvýrazní detaily a hrany
  - texturové filtry – zvýrazní opakující se vzory

0	0	0
0	1	0
0	0	0

Identitní filtr

1/8	1/8	1/8
1/8	1/8	1/8
1/8	1/8	1/8

Rozmazávací filtr

Konvoluční operace mimo neuronové sítě

Co se stane, když na stejný obrázek použijeme různé filtry?

- **Stejný vstupní obrázek** vytvoří různé mapy příznaků podle použitého filtru
- V hlubokém učení: filtry se neučí ručně, ale automaticky z dat – síť si hledá ty nejvíce užitečné vzory

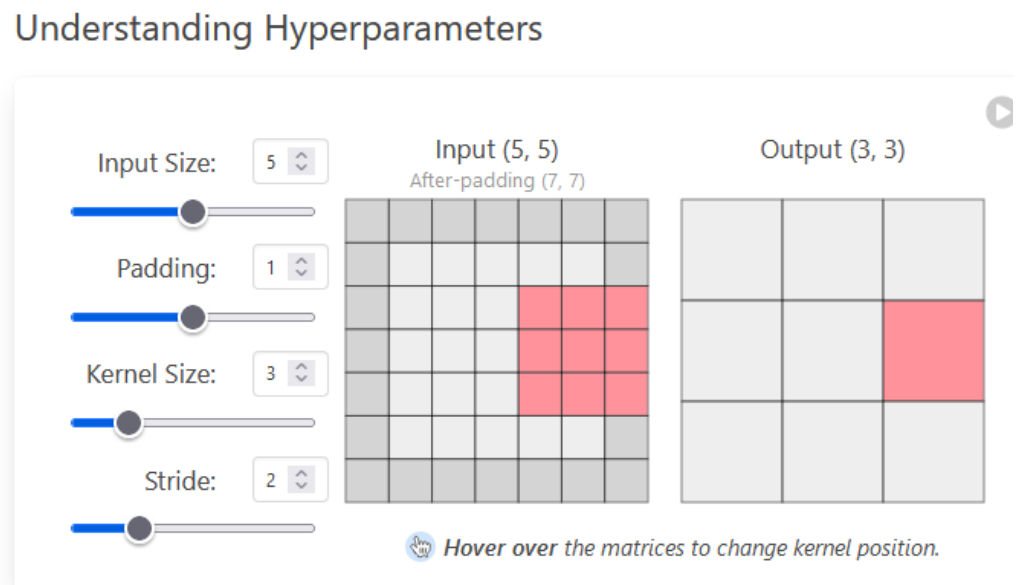
Tip: interaktivní formulář: https://generic-github-user.github.io/Image-Convolution-Playground/src/

Ukázkový notebook: convolution_introduction.ipynb

Operace konvoluce

Parametry konvoluční operace

- rozměry vstupního obrázku
- padding - okraje obrázku
- rozměry filtru
- stride = krok, pomocí kterého procházíme obrázek



pěkná vizualizace: https://poloclub.github.io/cnn-explainer/

Parametry konvoluce: Padding a Stride

Padding (doplnění okrajů)

- Přidává kolem obrázku "rámeček"z nul (nebo jiných hodnot)
- Běžné možnosti:
  - **Valid** – bez doplnění (výstup je o trošku menší)
  - **Same** – doplnění okrajů tak, aby výstup měl stejnou velikost jako vstup
- Pomáhá zachovat prostorové rozměry a zlepšuje detekci u okrajů

Stride (krok filtru)

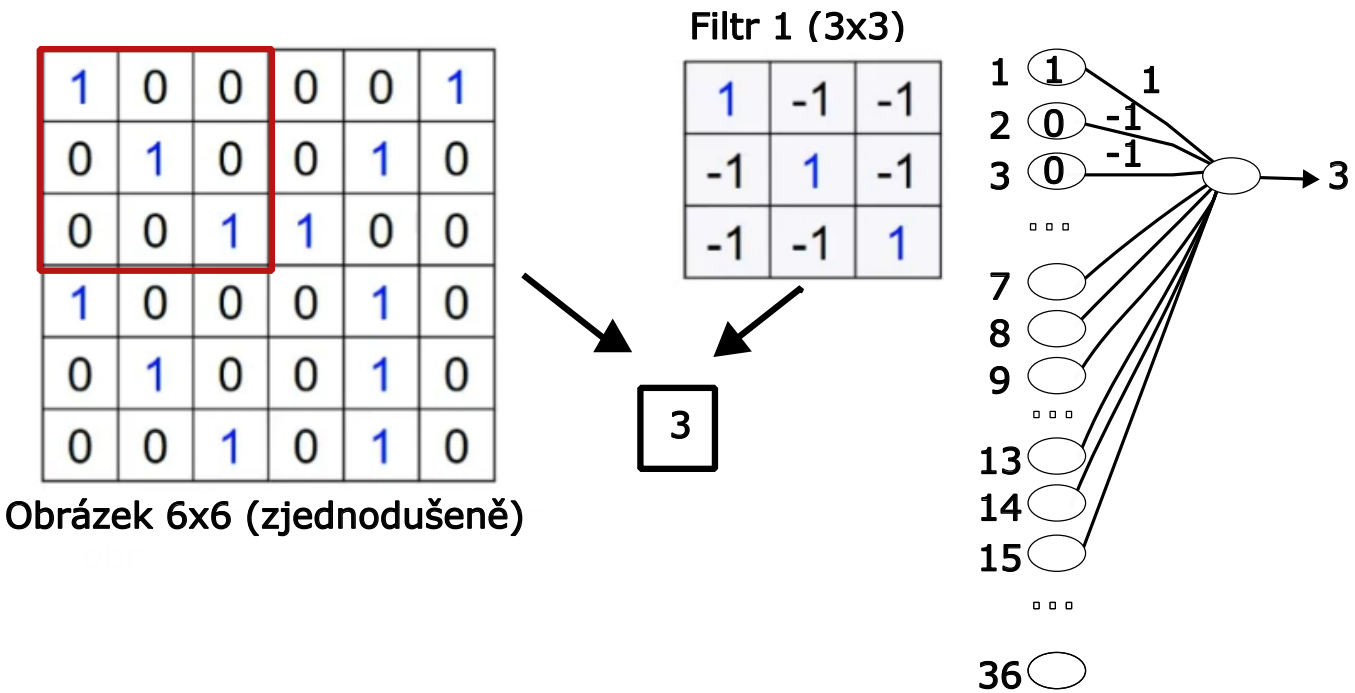
- Určuje, o kolik se filtr posune při každém kroku
- **Stride = 1**: běžné nastavení, husté pokrytí
- **Stride ¿ 1**: zmenšuje výstup, provádí downsampling

Tip: Vyzkoušejte si různé hodnoty v CNN Explaineru

Konvoluční vrstva vs. plně propojená vrstva

- na konvoluční vrstvu se můžeme dívat jako na klasickou vrstvu neuronové sítě (bez přenosové funkce)
- neurony ale nejsou plně propojené

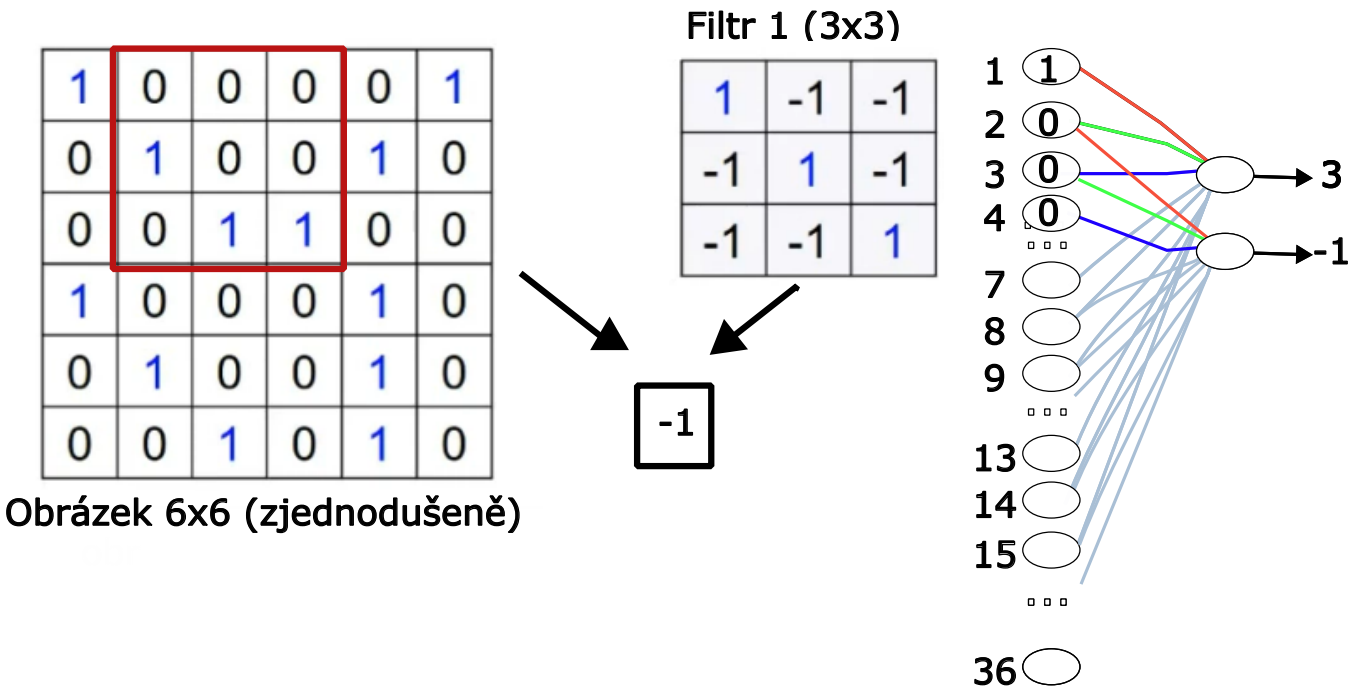
→ mnohem méně parametrů



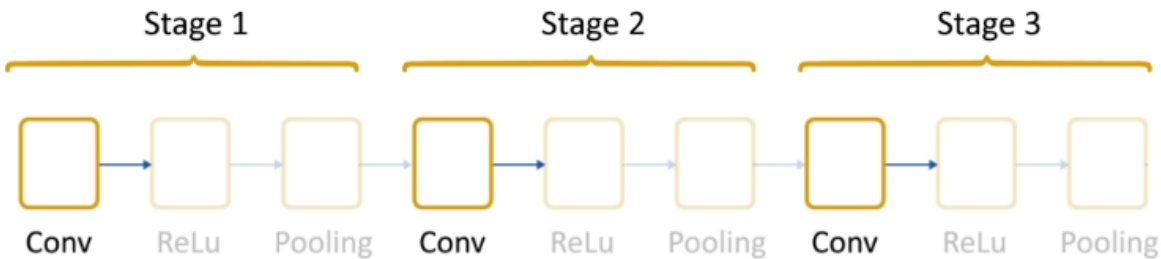
Konvoluční vrstva vs. plně propojená vrstva

- na konvoluční vrstvu se můžeme dívat jako na klasickou vrstvu neuronové sítě
- „fiktivní" neurony ve skryté vrstvě sdílejí stejné váhy

→ ještě méně parametrů, efektivnější učení



Klasická architektura konvoluční neuronové sítě

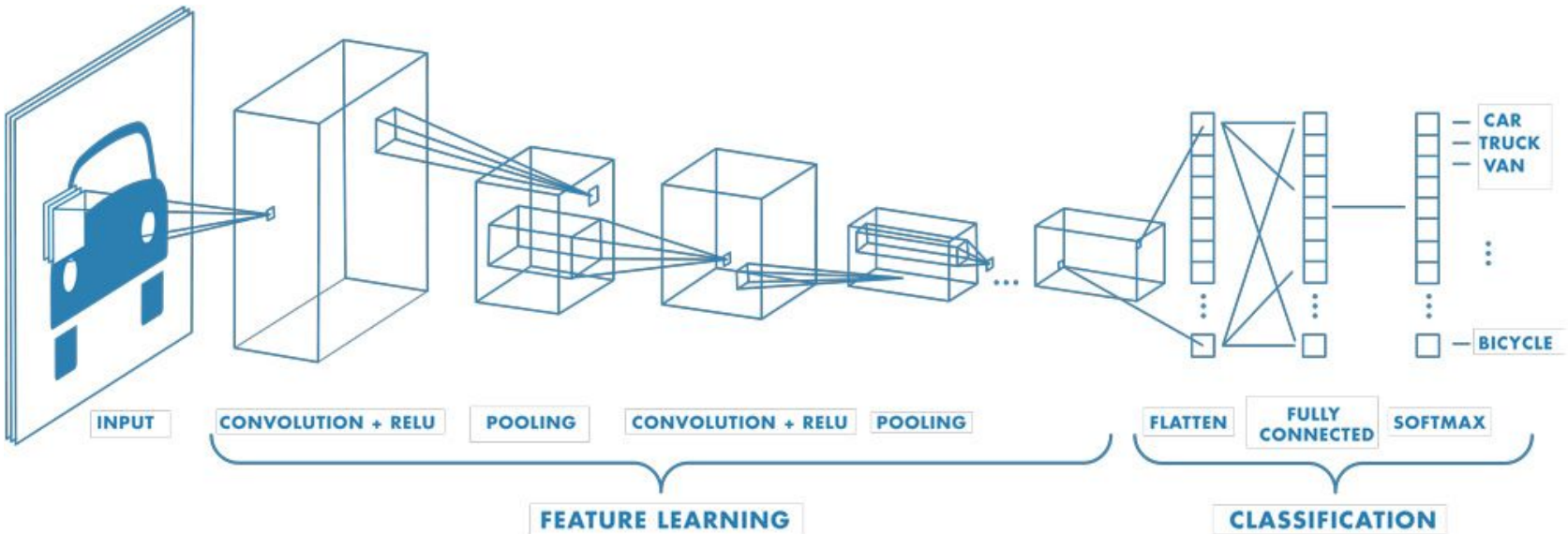


Princip: konvoluční vrstvy (resp. konvoluční bloky) vrstvíme na sebe

- první konvoluční vrstva detekuje jednoduché příznaky, např. hrany, bloby
- každá další konvoluční vrstva extrahuje příznaky (vzory) vyšší úrovně:

Hierarchická struktura příznaků: hranice $\rightarrow$ tvary $\rightarrow$ části objektů $\rightarrow$ celé objekty

Klasická architektura konvoluční neuronové sítě



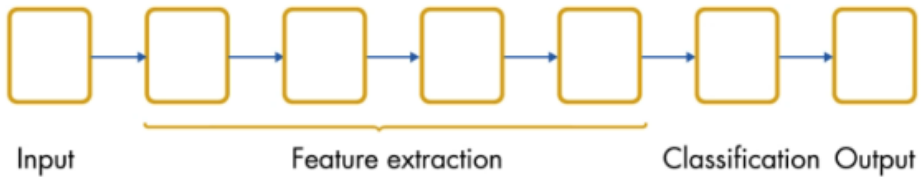
Části konvoluční neuronové sítě

- Konvoluční bloky pro extrakci příznaků
- Flattening vrstva - převede data na vektor čísel
- Vrstevnatá neuronová síť (plně propojené vrstvy) pro klasifikaci

Zdroj obrázku:

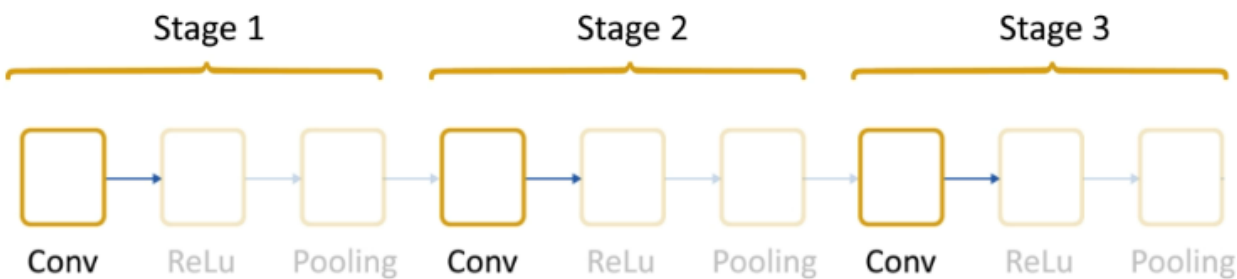
https://matlabacademy.mathworks.com/details/deep-learning-onramp/deeplearning

Klasická architektura konvoluční neuronové sítě



Typická struktura konvolučního bloku:

- konvoluční vrstva
- aplikace nelineární přenosové funkce (např. ReLU)
- pooling vrstva



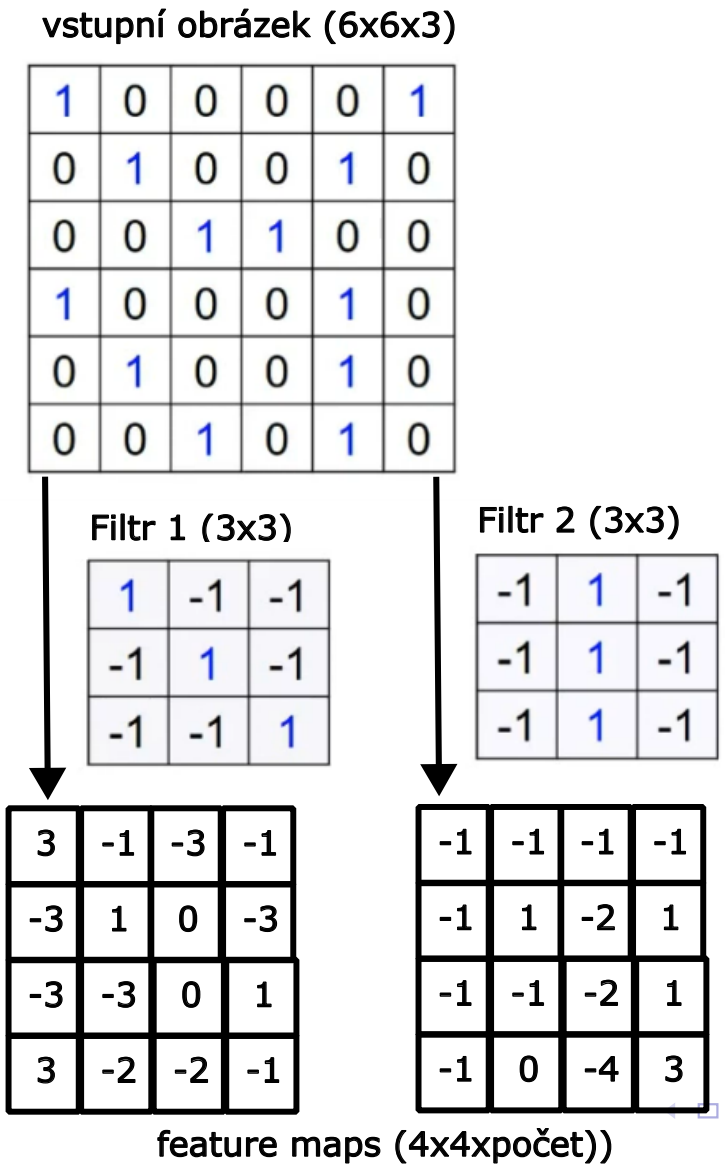
Pooling (subsampling) vrstva

- Slouží k zefektivnění výpočtu, zmenšuje rozlišení obrázku, aniž by se (příliš) zmenšila přenášená informace
- opět posouváme okénko, tentokrát např. velikosti 2x2, bez překryvu (stride $= 2$)
- operace MAX (max-pooling) nebo AVERAGE (average-pooling), žádné váhy

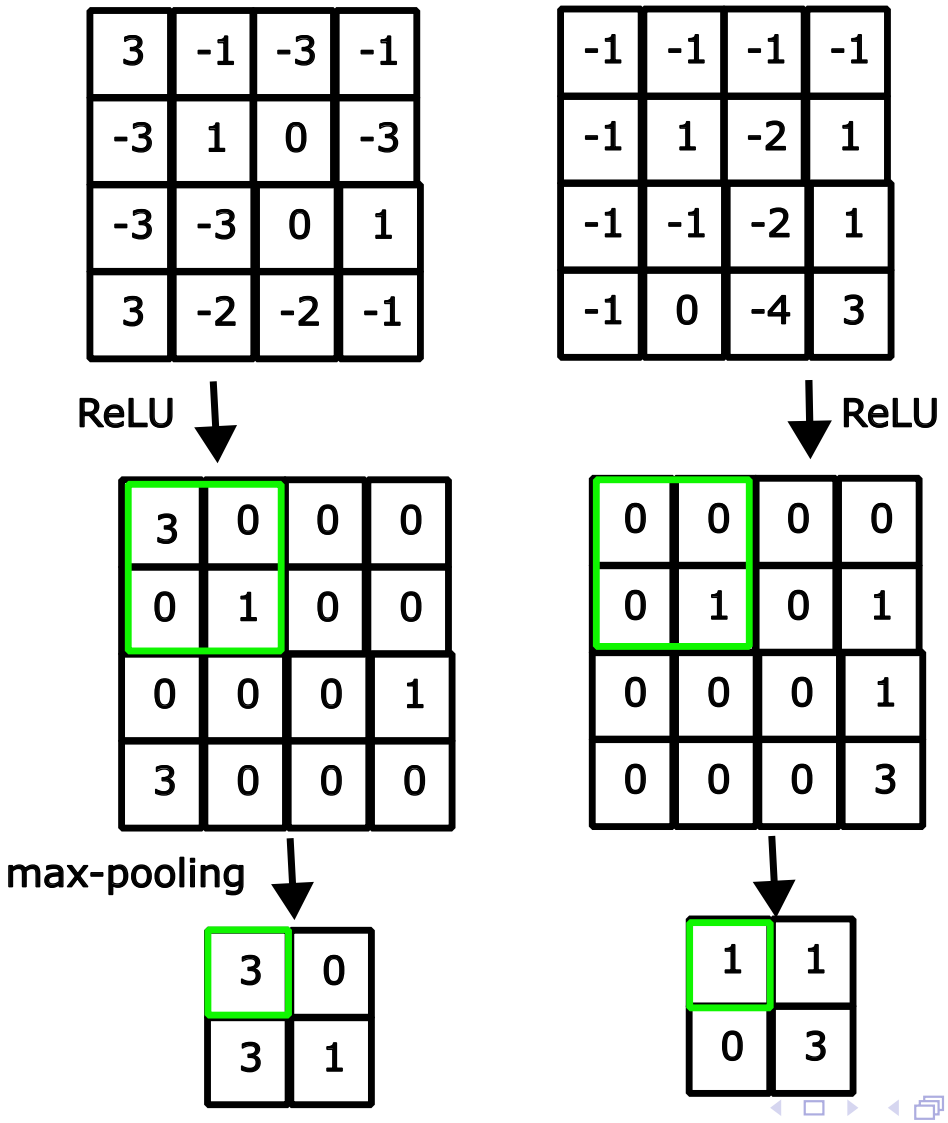
Význam pooling vrstvy

- Zahušťuje obrázek při zachování informace (dat) - kde a jak silně se v obrázku vyskytuje příznak (vzor)
- zmenšuje data (2x2 $\rightarrow$ na čtvrtinu)

Konvoluční blok - ukázka: konvoluční vrstva



Konvoluční blok - ukázka: pooling vrstva



Konvoluční blok

Proč na sebe neskládat čistě konvoluční vrstvy (bez poolingu)?

- Pokud je vrstvíme za sebe, s každou další vrstvou roste počet parametrů
- Přitom se nemění velikost obrázku (až na okraje, v závislosti na padding)
  $\rightarrow$ s každou další vrstvou roste velikost dat

Pooling (subsampling) vrstva

- Zahušťuje obrázek při zachování informace (dat) - kde a jak silně se v obrázku vyskytuje příznak (vzor)
- Zmenšuje data (2x2 $\rightarrow$ na čtvrtinu)

Střídání konvoluční a pooling vrstvy: bipyramidální efekt

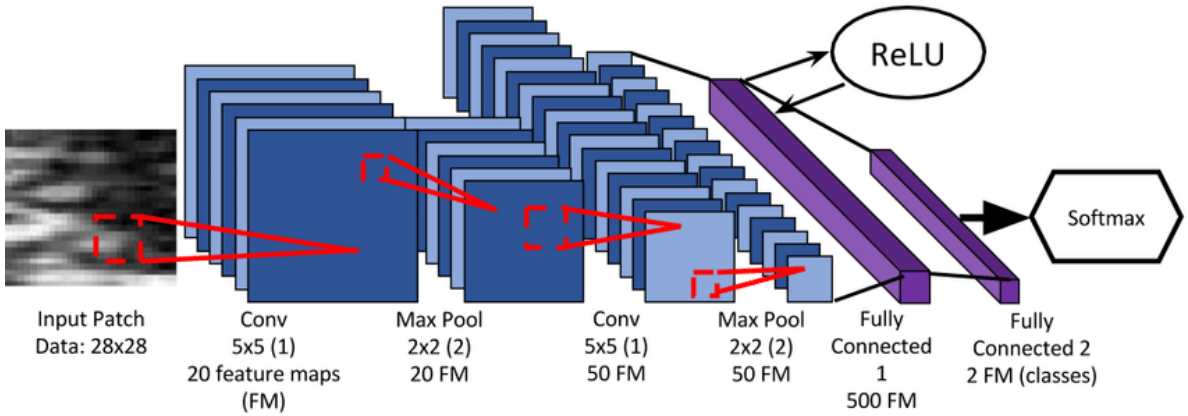
- postupně se zmenšují obrázky a zvětšuje se počet feature maps

Bipyramidální architektura

- nejstarší typy architektur: široké, mělké, často hlubší plně propojená část, odpovídají více základnímu schématu

LeNet 5

- jedna z původních architektur (Yann LeCun, 1998), poměrně jednoduchá, učená na MNIST



Zdroj obrázku: M. H. Yap et al., "Automated Breast Ultrasound Lesions Detection Using Convolutional Neural Networks," in IEEE Journal of Biomedical and Health Informatics, vol. 22, 2018.

Učení konvoluční neuronové sítě

- nějaká varianta algoritmu zpětného šíření (např. SGD)
- mini-batch učení, model potřebuje k naučení větší množství dat
- velké množství parametrů

Jak zvolit vhodnou architekturu v praxi?

- neoptimalizujeme počet vrstev a neuronů
- vybereme z dostupné literatury topologii osvědčenou pro daný typ problému

Ukázky

visualize_cnn_mnist.ipynb

- Praktický příklad: datová sada MNIST (číslice)