18NES1 Neuronové sítě 1 - Konvoluční neuronové sítě

18NES1 - 17. -18. hodina, LS 2024/25

Zuzana Petříčková

14. a 17. dubna 2025

Co jsme probírali minule

Úvod do konvolučních neuronových sítí

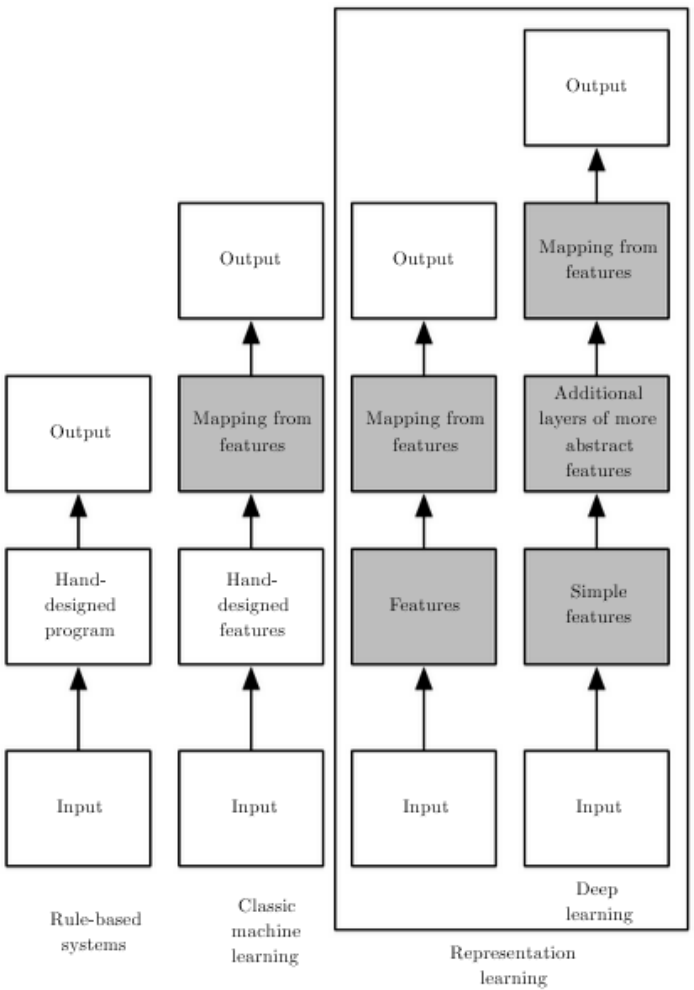
- Motivační příklad: rozpoznávání pěvců
- Operace konvoluce, její význam a parametry
- Konvoluční neuronová síť a její architektura (vrstvy, filtry, pooling)
- Klasická architektura konvoluční neuronové sítě + ukázka na datech MNIST

Tento týden

Konvoluční neuronové sítě - pokračování

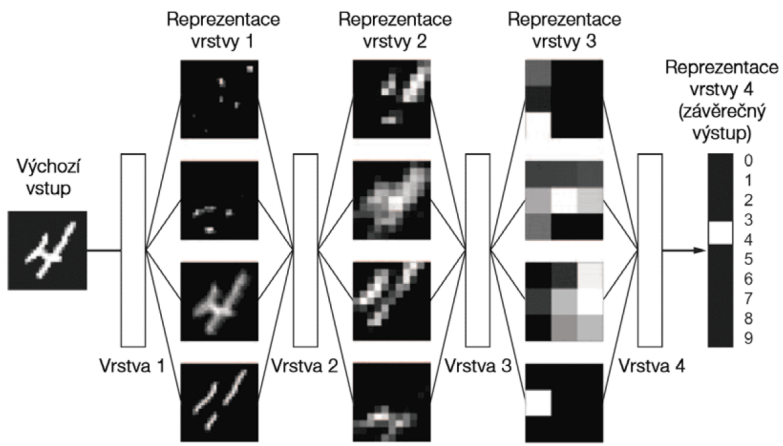
- Připomenutí základních pojmů
- Bipyramidální architektura
- Vizualizace fungování konvoluční neuronové sítě
- Příprava a augmentace obrazových dat a jejich efektivní zpracování pomocí data loaderů
- Učení modelu CNN od nuly
- Techniky pro zlepšení schopnosti CNN zobecňovat
- Přenesené učení (transfer learning) - úvod
- Praktické ukázky a experimenty

Připomenutí: Hluboké učení



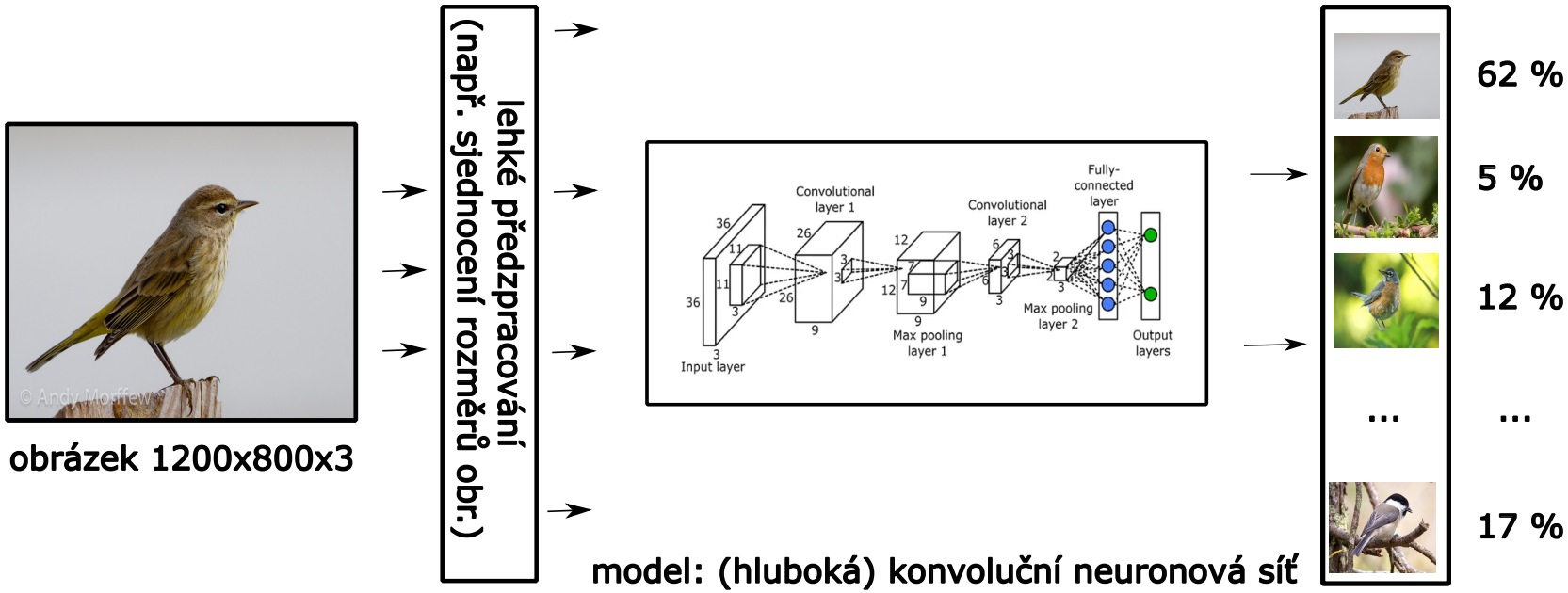
I. Goodfellow and Y. Bengio and Aaron Courville: Deep Learning, 2016, Figure 1.5

- Využívá umělé neuronové sítě s mnoha vrstvami (tzv. hluboké sítě)
- Modely se samy učí extrahovat příznaky z dat – méně ručního předzpracování
- Architektura často přizpůsobena konkrétnímu typu dat (obraz, text, zvuk, ...)



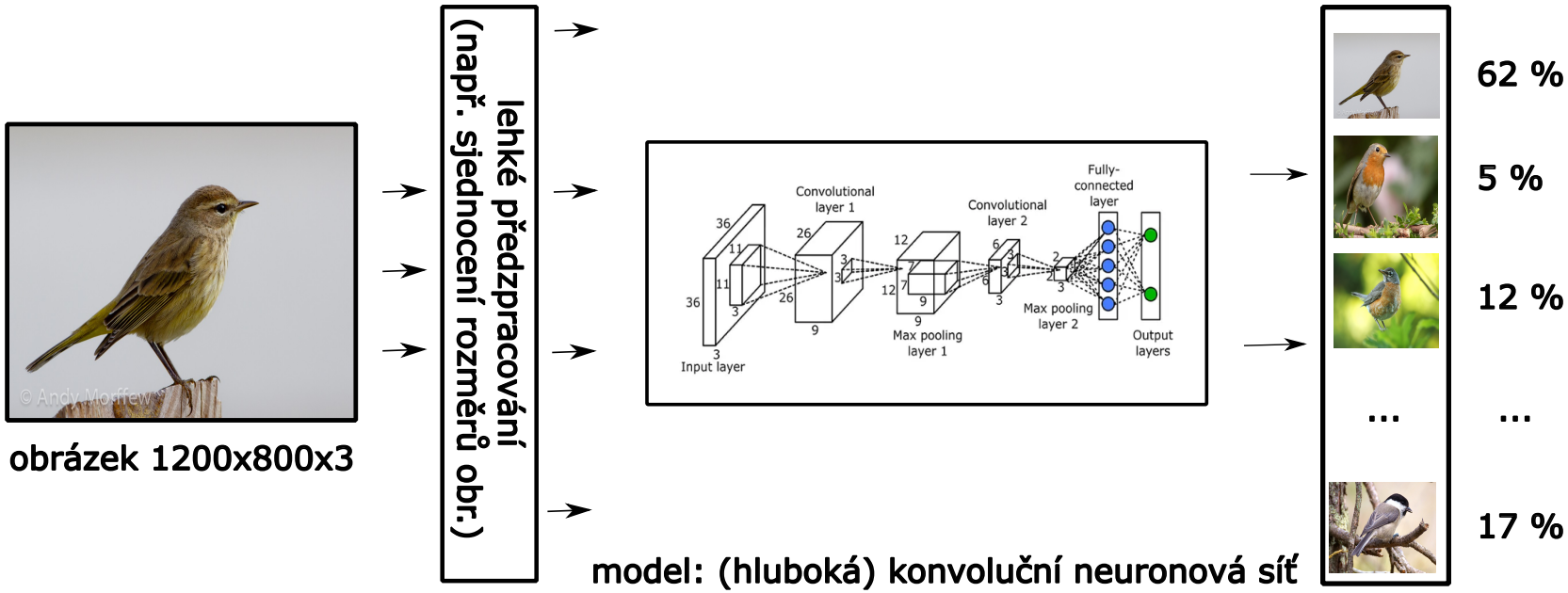
F. Chollet: Deep learning v jazyku Python, obr. 1.6

Konvoluční neuronová síť



- Specializovaný typ sítě pro zpracování obrazových dat
- Efektivní extrakce příznaků pomocí konvolučních vrstev (filtrů)
- Bere v potaz prostorové uspořádání pixelů a lokální souvislosti
- Díky sdílení vah má méně parametrů než plně propojené vrstvy

Výhody konvolučních sítí



- Zachovávají prostorovou informaci a vnímají lokální vztahy
- Výrazně méně parametrů díky sdílení vah
- Lepší škálovatelnost pro větší obrazy
- Robustnost vůči posunu a změně měřítka objektů

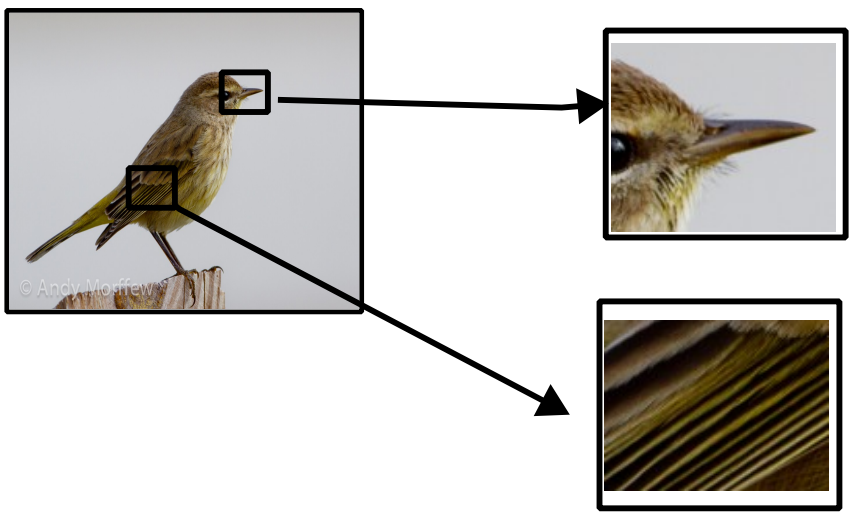
Konvoluční neuronová síť

- neuronová síť obsahující konvoluční vrstvy

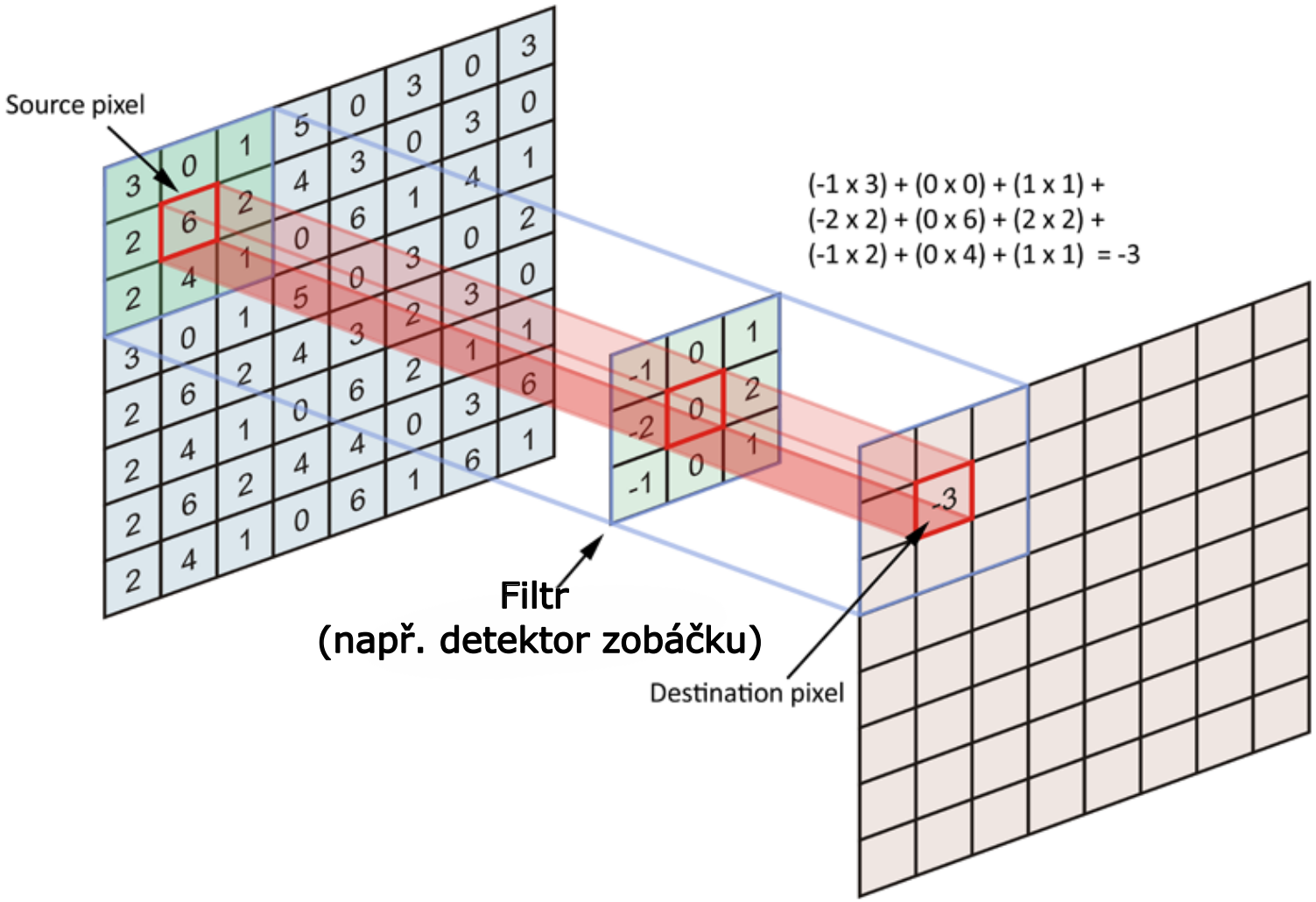
Konvoluční vrstva

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

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)

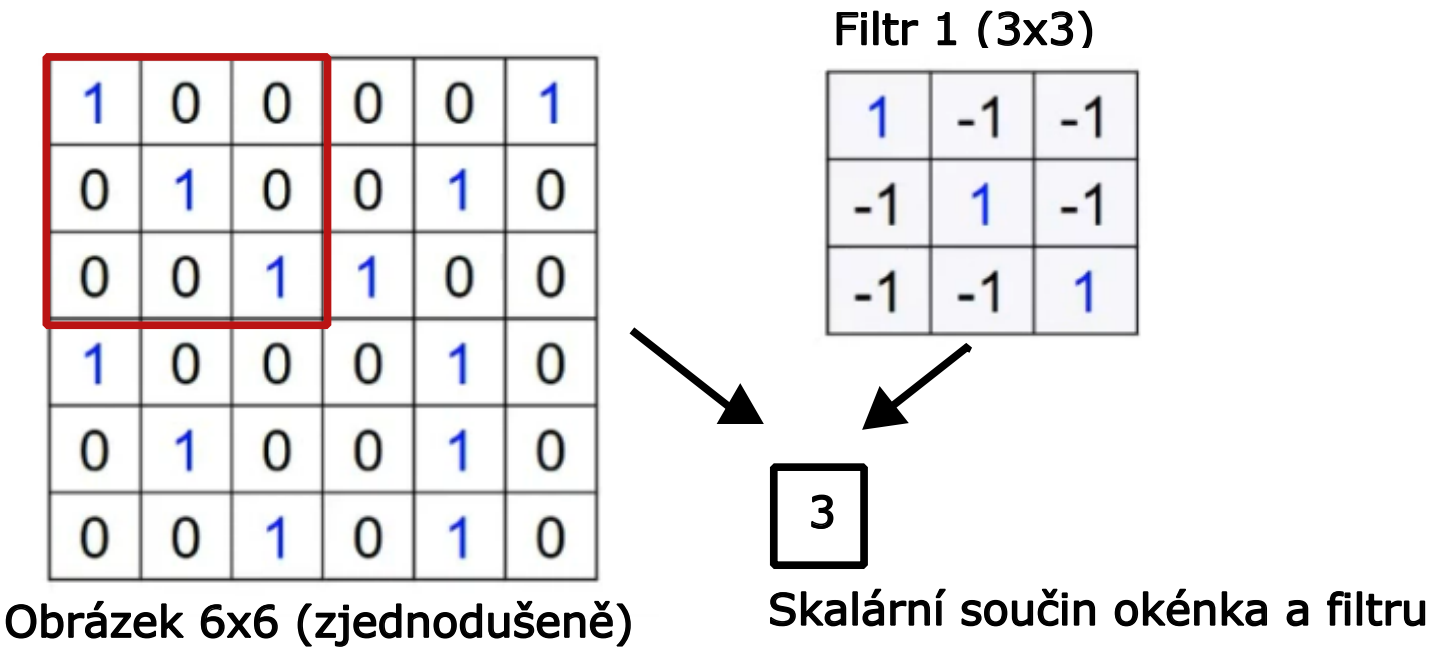
. . .

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

zdroj příkladu: Petr Doležel: Konvoluční neuronová síť,

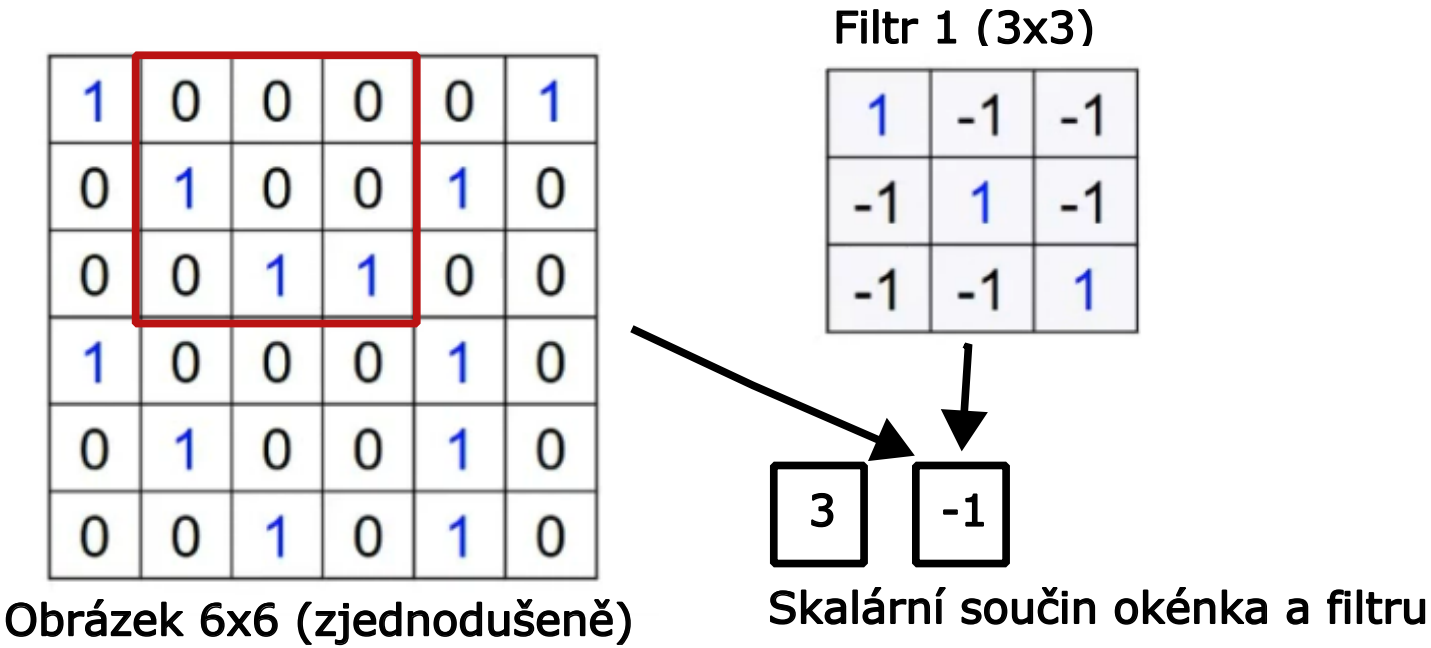
https://www.youtube.com/watch?v=-2vEi-Aa0FA

Operace konvoluce



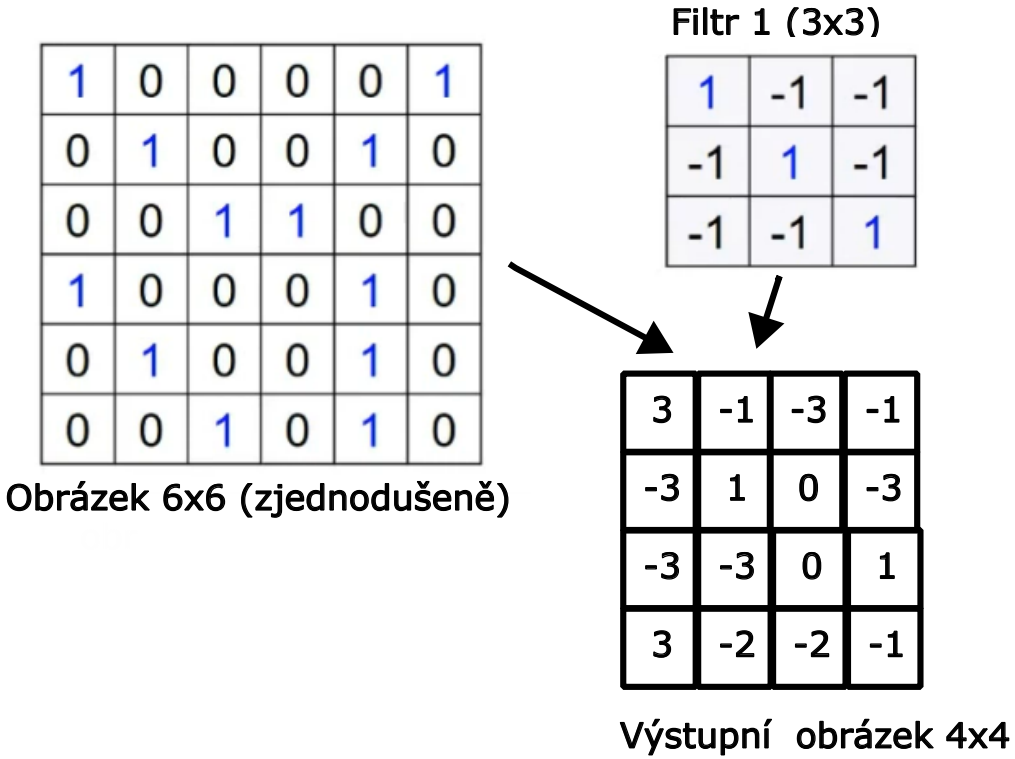
- spočteme skalární součin:
  $y = \sum_{i=1}^{9} w_i x_i + b$ (pro linearizované matice)

Operace konvoluce



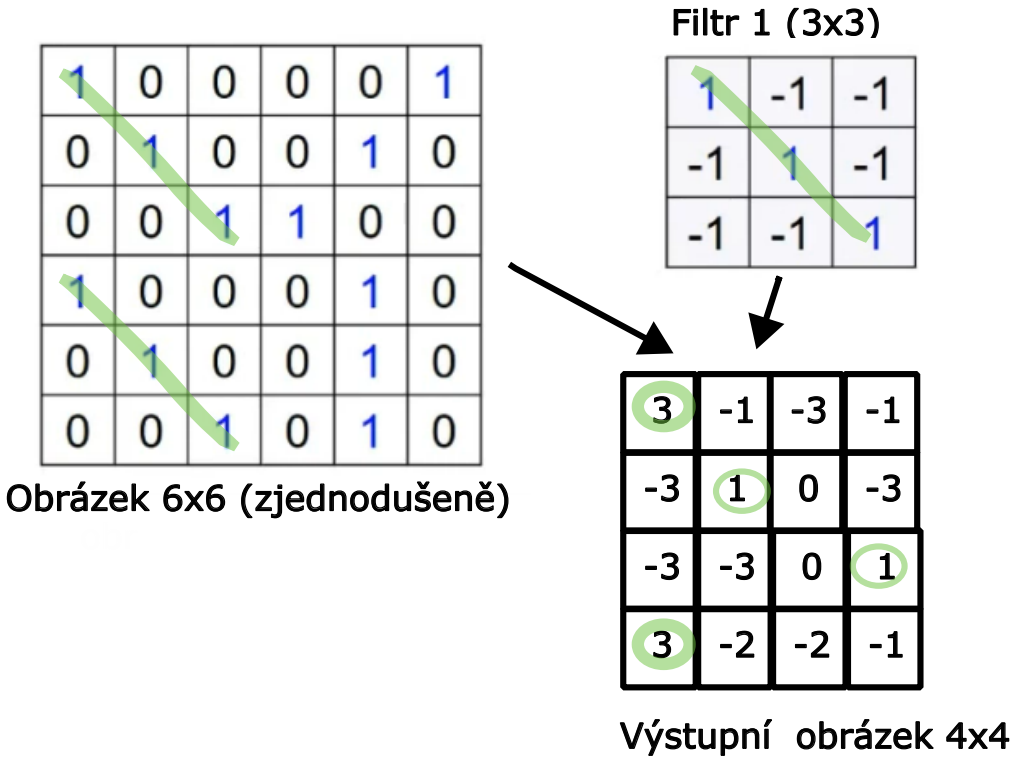
- okénko posunu a opět spočteme skalární součin

Operace konvoluce



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

Operace konvoluce

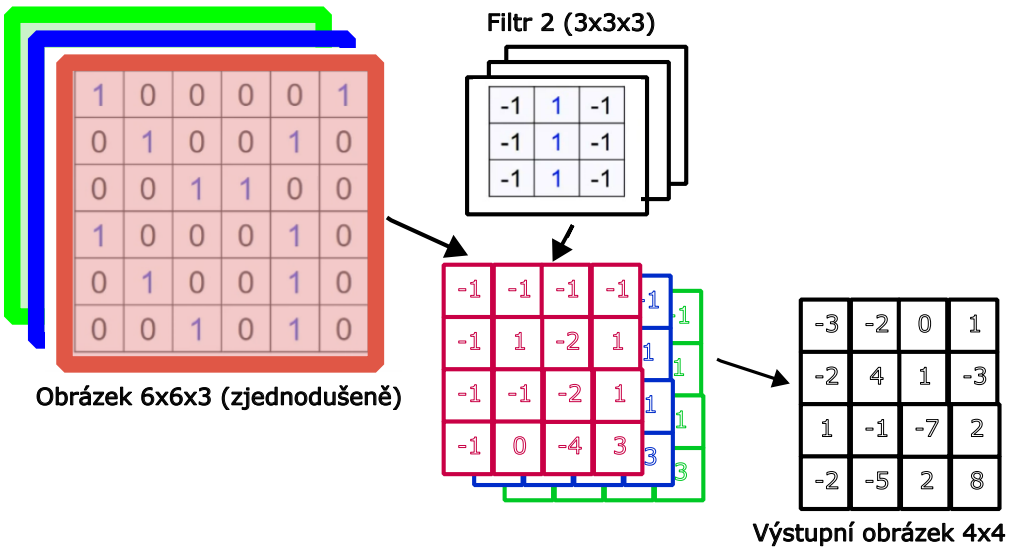


Matice příznaků (feature map)

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

Operace konvoluce

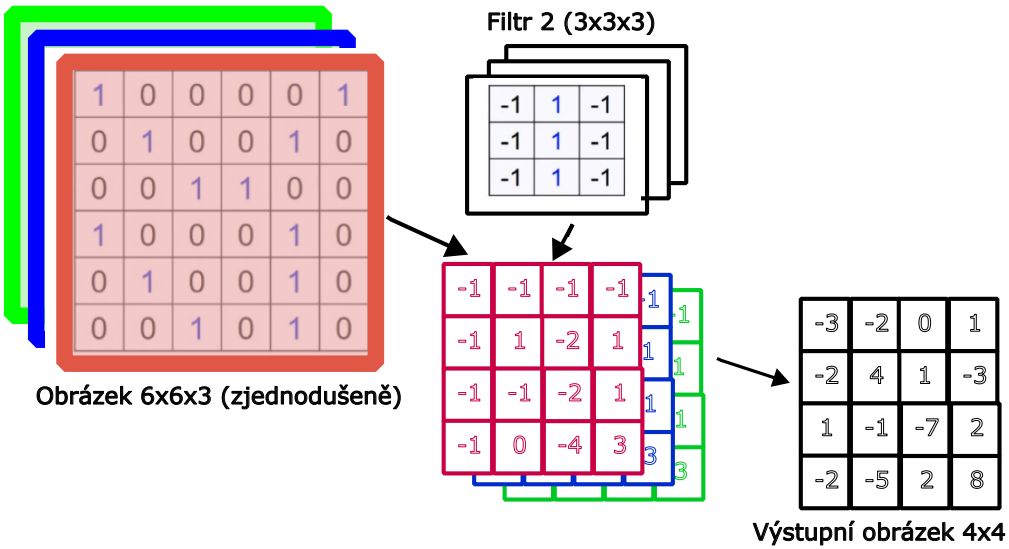
Barevný obraz: 3 kanály (channels) R, G, B



- 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 (channels)**

Operace konvoluce

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

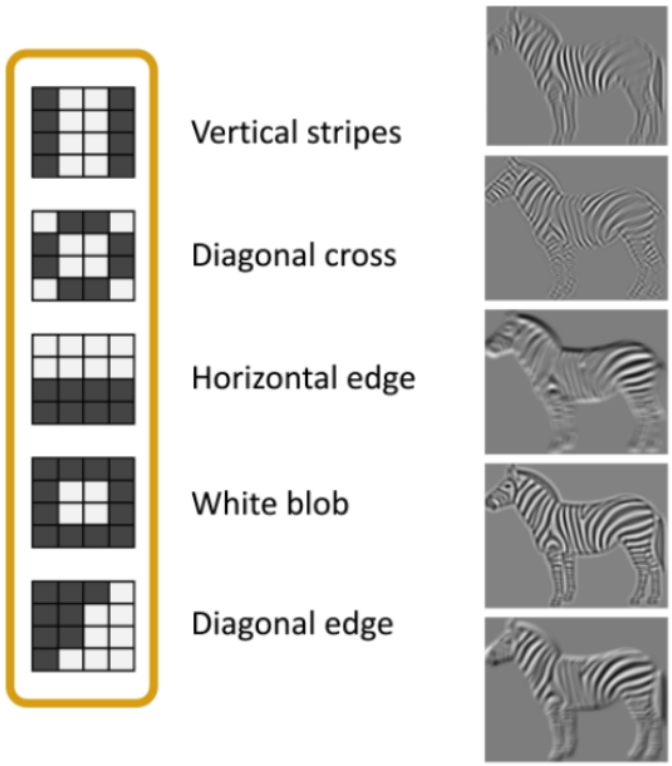


Reprezentace konvoluční vrstvy

- 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



- ukázky filtrů a výsledných matic příznaků

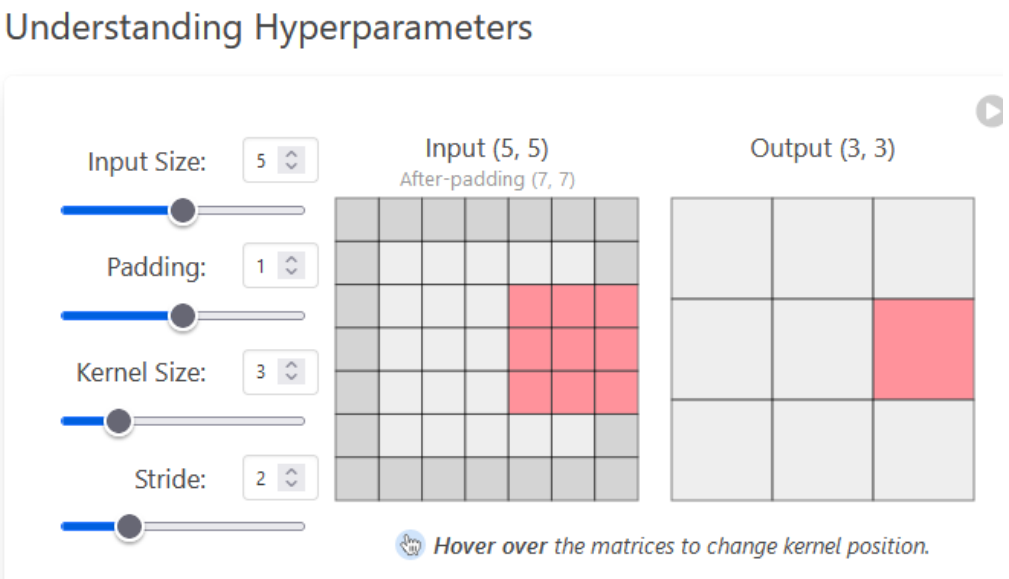
Zdroj příkladu:

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

Operace konvoluce

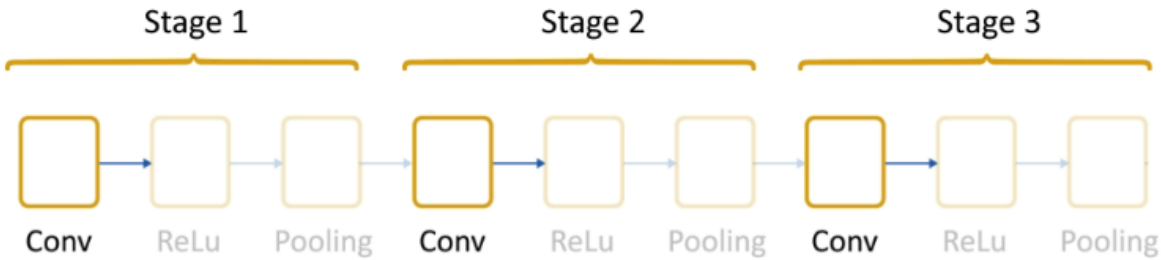
Parametry konvoluční operace

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



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

Klasická architektura konvoluční neuronové sítě

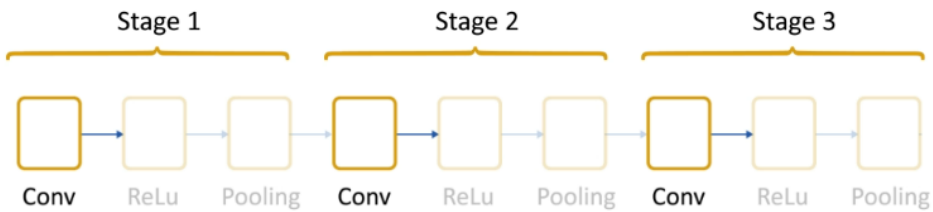


Princip: konvoluční vrstvy (resp. konvoluční bloky) vrstvíme na sebe

- 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



Typická struktura konvolučního bloku:

- Konvoluční vrstva
- Aplikace nelineární přenosové funkce (např. ReLU)
- Pooling vrstva

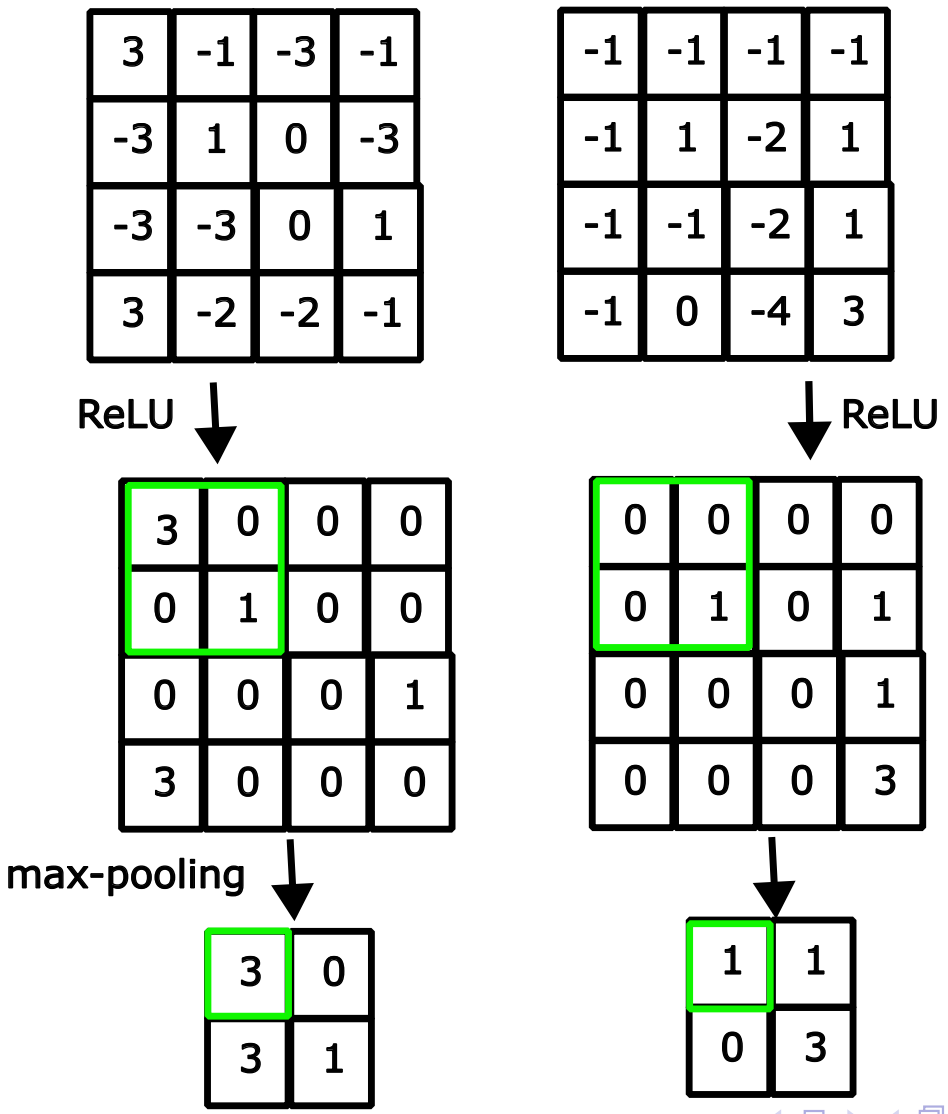
Pooling (subsampling) vrstva

- 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

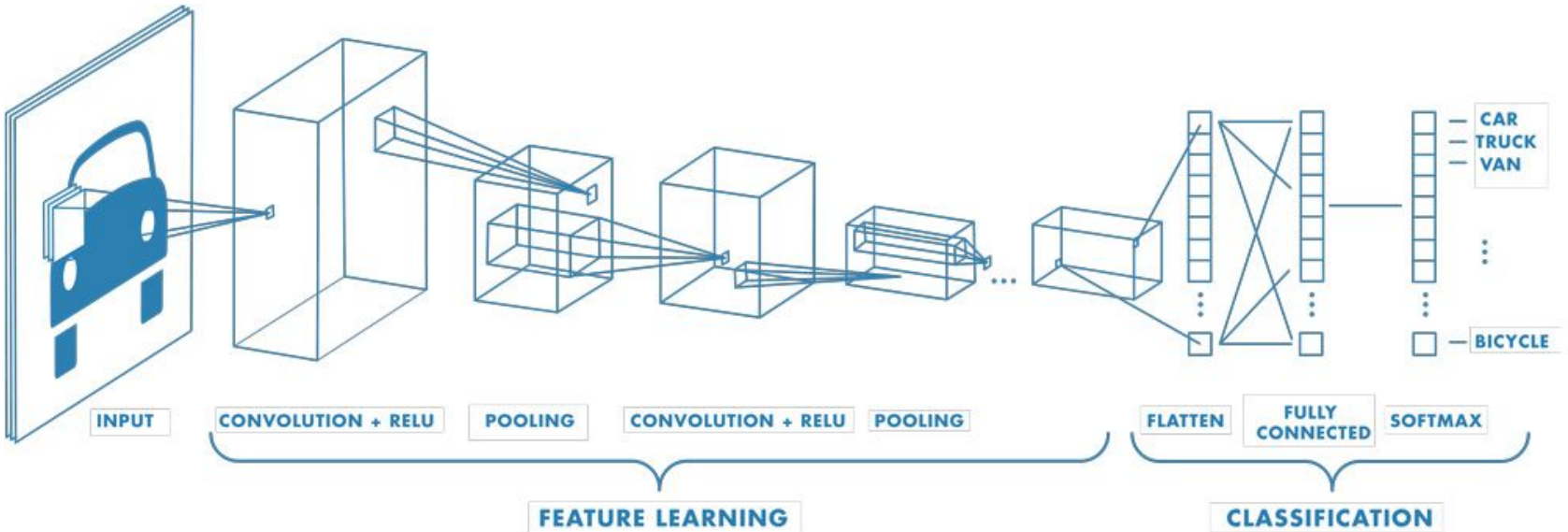
Střídání konvoluční a pooling vrstvy: bipyramidální efekt

- postupně se zmenšují obrázky a zvětšuje se počet feature maps

Pooling (subsampling) vrstva



Typická 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:

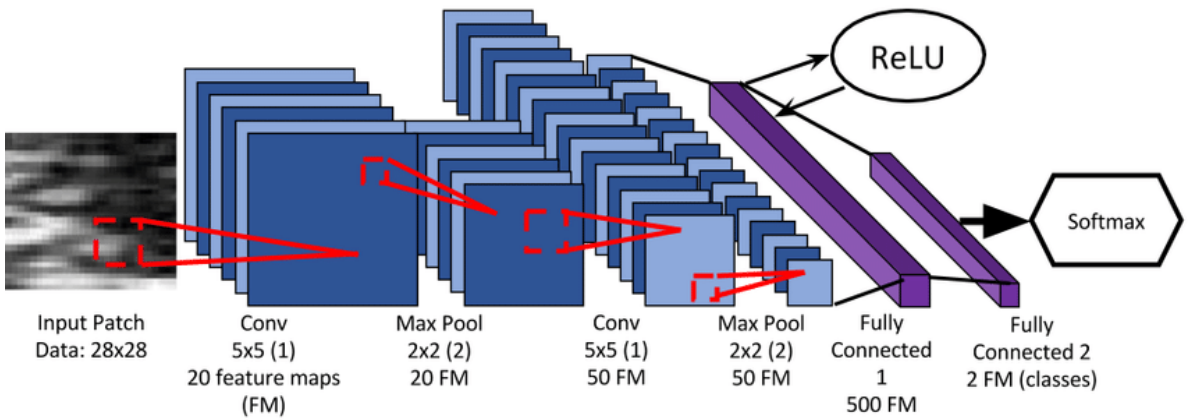
https://matlabacademy.mathworks.com/details/deep-learning-onramp/deeplearning

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.

Bipyramidální architektura

Typická ukázka bipyramidální architktury:

- Počet filtrů se ve vrstvách postupně zdvojnásobuje (např. 32, 64, 128, . . .)
- Nejčastěji používaná velikost filtru: 3×3
- Max-pooling 2×2 často kombinován se zdvojnásobením počtu kanálů
- Pokud použijeme více konvolučních vrstev, nepotřebujeme větší počet plně propojených vrstev
- (Volitelně) Následuje jedna nebo několik plně propojených vrstev pro klasifikaci

visualize_cnn_mnist.ipynb

- Praktický příklad: datová sada MNIST (číslice)

Reprezentace vstupu a parametrů konvoluční vrstvy

Vstup do vrstvy = 4D tenzor tvaru:
`(počet vzorků v batchi, výška, šířka, počet kanálů)`
`= (batch_size, height, width, channels)`

- Např. pro 8 RGB obrázků velikosti 32×32 px:
  $\Rightarrow$ tvar vstupního tenzoru: `(8, 32, 32, 3)`
- V hlubších vrstvách vstup tvoří matice příznaků z předchozí vrstvy $\Rightarrow$ tvar tenzoru např. : `(8, 32, 32, 64)`

Váhový tenzor konvoluční vrstvy (filtry) = 4D tenzor tvaru:
`(výška filtru, šířka filtru, počet vstupních kanálů, počet filtrů)`
`= (filter_height, filter_width, in_channels, out_channels)`
visualize_cnn_mnist.ipynb

Reprezentace vstupu a parametrů konvoluční vrstvy

Váhový tenzor konvoluční vrstvy (filtry) = 4D tenzor tvaru:
`(výška filtru, šířka filtru, počet vstupních kanálů, počet filtrů)`
`= (filter_height, filter_width, in_channels, out_channels)`

- Např. 64 filtrů 3×3 pro RGB vstup: `(3, 3, 3, 64)`
- Každý filtr „skenuje“ vstupní obraz (nebo matice příznaků) a vytváří 1 výstupní kanál
- Výsledkem je 4D výstupní tenzor: `(batch_size, new_height, new_width, out_channels)`

visualize_cnn_mnist.ipynb

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 architekturu osvědčenou pro daný typ problému
  - bipyramidální architektura
  - některá z modernějších architektur (např. https://keras.io/api/applications/)

Ukázky: Vizualizace

visualize_cnn_mnist.ipynb

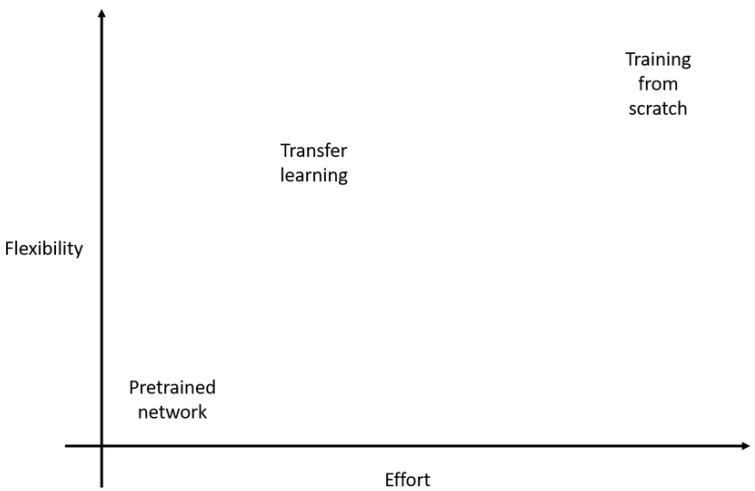
- Praktický příklad nad MNIST rozšířený o vizualizaci filtrů na jednotlivých vrstvách, o vizualizaci map příznaků a o vizualizaci důležitosti jednotlivých pixelů pro rozhodování

Zajímavé odkazy

- vizualizace konvoluční neuronové sítě https://poloclub.github.io/cnnexplainer/
- Mathworks - face
- vizualizace operace konvoluce https://generic-github-user.github.io/Image-Convolution-Playground/src/

Možnosti vytvoření a učení konvolučních neuronových sítí

- učení od začátku (training from scratch)
- použití již naučeného modelu (pretrained network)
- přenesené učení (transfer learning)
- doučení již naučeného modelu (finetuning)



Zdroj :

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

Ukázka modelu učeného od začátku na malých datech: klasifikace květin

- Zdroj dat: **Oxford 102 Flower Dataset** paperswithcode.com/dataset/oxford-102-flower
- Obsahuje **8189 obrázků** ve **102 třídách** květin (každá třída cca 40–258 obrázků)
- Velikost datasetu: cca **330 MB** (JPEG)
- Vhodné pro testování trénování modelu CNN od nuly i přeneseného učení

Úloha:

- pro začátek vybereme jen podmnožinu dat: 3 zhruba nejpočetnější třídy
- data rozdělíme na trénovací, validační a testovací množinu
- na datech naučíme základní -bipyramidální - CNN model

CNN_from_scratch_flowers_3.ipynb

Ukázka modelu učeného od začátku na malých datech: klasifikace květin

CNN_from_scratch_flowers_3.ipynb Na příkladu si ukážeme různé techniky

- zpracování obrazových dat
- efektivní načítání dat pomocí data loaderů
- funkcionální API v Kerasu pro vytváření modelů
- vizualizace v případě barevných (RGB) obrázků
- augmentace dat a regularizační techniky (později)

Ukázka modelu učeného od začátku na malých datech: klasifikace květin

Pozorování

- Oproti MLP se CNN učí poměrně pomalu.
- Oproti modelu nad MNIST vytváří model nad Flowers variabilnější filtry, nejen pro přechodové hrany, ale i pro složitější vzory
- Zajímavé je pozorovat na Saliency mapě, podle kterých částí obrazů se model rozhodoval.
- Model se mírně přeučil.

Rozšíření : Klasifikace do všech 102 tříd
CNN_from_scratch_flowers_102.ipynb

- **Pozorování:** Zde naučený model zobecňuje velmi špatně.
  $\rightarrow$ co použít vhodnou regularizační techniku?

Konvoluční neuronové sítě a zobecňování

- Konvoluční neuronové sítě mívají problémy s přeučením
- Problémem je, pokud máme k dispozici málo dat (stovky, malé tisíce vzorů)
- Navíc: učení může být velmi pomalé
- Jak zobecňování zlepšit?
  - **standardní regularizace**
  - **data augmentation**: rozšíření dat
  - **transfer learning**

Konvoluční neuronové sítě a zobecňování

Nejčastěji používané techniky regularizace

- **Early stopping**
- **L1/L2 regularizace** - u ReLU jednotek v konvolučních a plně propojených vrstvách
- **Dropout** - přidání speciální vrstvy za každou plně propojenou vrstvu
- **Normalizace** dat, vah, výstupů vrstev, oblíbená technika je Batch Normalization
- **Label smoothing** (zašumění labelů)
- **Ensembling**

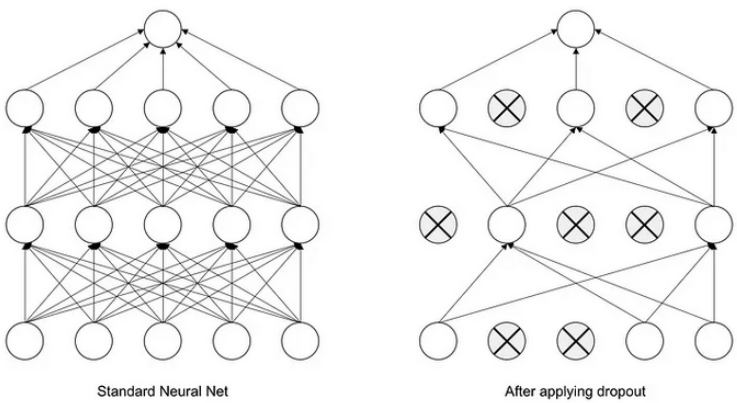
Typické pro konvoluční sítě':

- **Augmentace dat**
- **Transfer learning (přenesené učení)**

Konvoluční neuronová síť a regularizace

Dropout (Srivastava et al., 2014)

- vysoce účinná metoda
- spočívá v náhodném vypínání (deaktivování) některých skrytých neuronů během učení
- při testování a používání modelu jsou všechny neurony aktivované
- implementované přidáním speciální **dropout** vrstvy za každou plně propojenou vrstvu

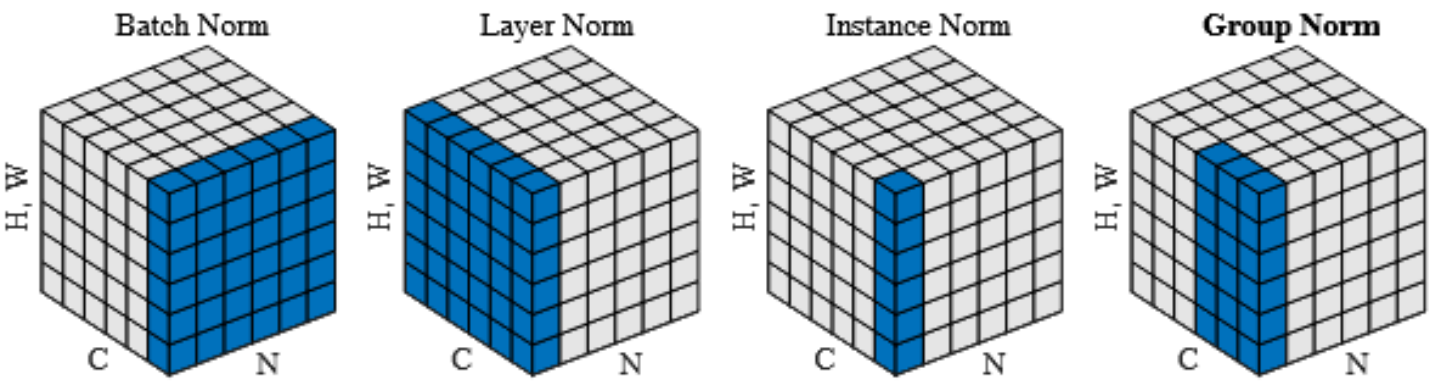


Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting", JMLR 2014

Konvoluční neuronová síť a regularizace

Normalizace výstupů jednotlivých vrstev

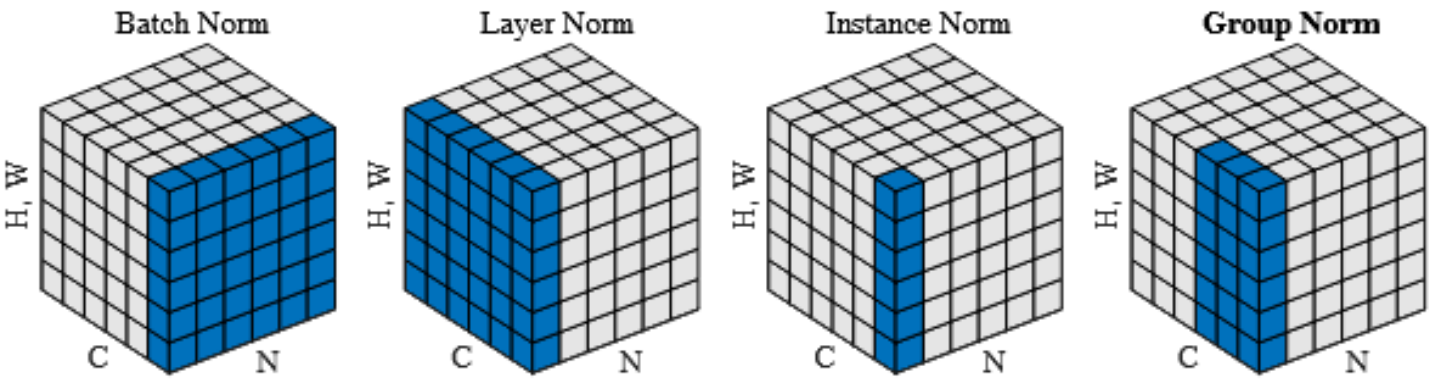
- snaha zafixovat střední hodnoty a rozptyly výstupů každé vrstvy
- snaha o řešení problému mizejících gradientů
- Vstup konvluční vrstvy je 4-rozměrný tenzor tvaru N $\times$ H $\times$ W $\times$ C
- N ... batch (počet vzorů), C ... channels (kanály), H, W ... rozměry matice příznaků
- různé varianty:



Konvoluční neuronová síť a regularizace

Normalizace vstupů jednotlivých vrstev

- implementováno pomocí přidání další vrstvy (např. za konvoluční vrstvu)
- rychlejší učení, menší citlivost na inicializaci vah
- robustnost k šumu v datech (nahradí Dropout)



Wu, Y., et al. "Group Normalization", https://arxiv.org/pdf/1803.08494

CNN a regularizace

Normalizace: Kdy selhává

- Přes své teoretické výhody může normalizace v praxi **zhoršit učení**:
  - **Malé velikosti batchů a malé datasety** vedou k nestabilnímu nebo zkreslenému odhadu střední hodnoty a rozptylu.
  - Může **zpomalit konvergenci** nebo **způsobit uvíznutí modelu v lokálním minimu**.
  - **Interferuje s dropoutem, reziduálními hranami nebo některými přenosovými funkcemi**.
- **Praktický tip:**
  - U malých nebo jednoduchých datasetů mohou lépe fungovat jednodušší modely bez normalizace.
  - U hlubokých modelů má normalizace jednoznačněji pozitivní dopad

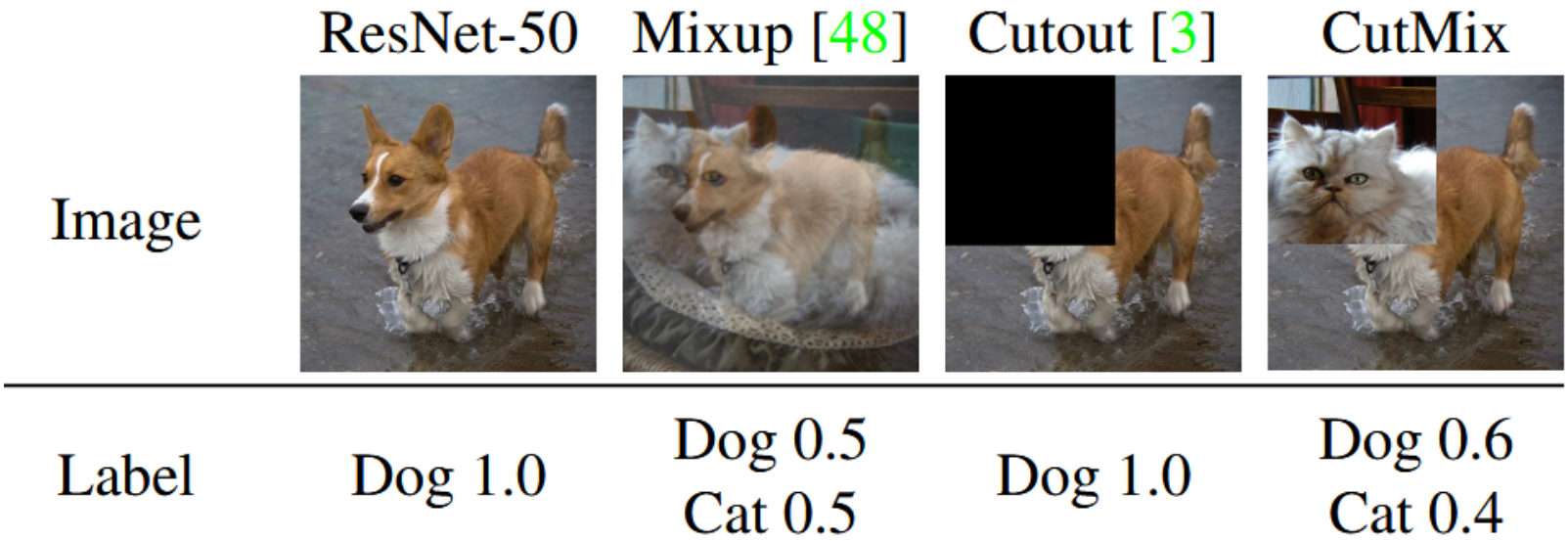
Augmentace dat

- různé (náhodné) transformace obrazu (rotace, posun, zrcadlení, zešikmení, měna rozlišení, změna jasu a kontrastu, oříznutí, přidání šumu, blur, kombinace)
- v Kerasu implementujeme pomocí speciální vrstvy

Zdroj :

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

Augmentace dat - oblíbené varianty



Zdroj : Yun et all., CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features, https://arxiv.org/pdf/1905.04899

Augmentace dat

Výhody:

- Uměle zvětšuje velikost a rozmanitost trénovací množiny
- Pomáhá předcházet overfittingu tím, že model vystavuje širší škále variací vstupu
- Zlepšuje generalizaci na neviděná data a zvyšuje odolnost vůči deformacím

Implementace v Kerasu:

- Snadno použitelná pomocí vrstev jako `RandomFlip`, `RandomRotation`, `RandomZoom` atd.
- Augmentace probíhá **za běhu během trénování**, což šetří paměť

Praktické ukázky regularizace

regularization_cnn_mnist.ipynb

- pokračování příkladu MNIST

CNN_from_scratch_flowers_3.ipynb,
CNN_from_scratch_flowers_102.ipynb

- pokračování příkladu Flowers 102

Praktické ukázky regularizace

Klasifikace Flowers do tří tříd - Pozorování

- Danou ,,menší" úlohu (klasifikace do třech tříd) se model i bez regularizace naučil poměrně dobře, i když se mírně přeučil.
- Regularizace (augmentace dat, dropout a early stopping) pomohla k lepšímu naučení
- Batch normalizace v tomto případě (poměrně mělký model) nevede k lepším výsledkům

Klasifikace Flowers do všech 102 tříd - Pozorování:

- Tentokrát model bez regularizace zobecňuje velmi špatně
- Regularizace pomůže, ale jen trochu

Další krok

- Co takhle použít již naučený model, popř. přenesené učení (transfer learning)?

Použití předučeného modelu

Co takhle použít již nějaký existující model naučený na velké sadě dat?

Předučené modely:

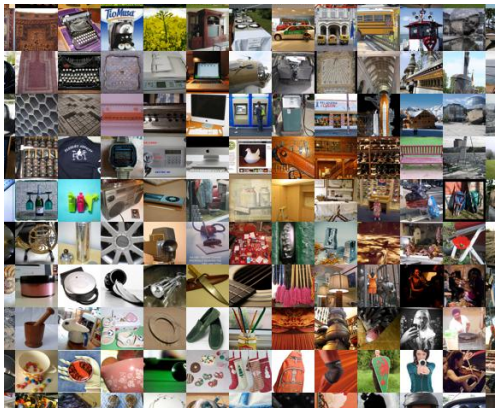
- https://keras.io/api/applications/
- oblíbené architektury konvolučních neuronových sítí
  - VGG16
  - MobileNet
  - ResNet
  - ...
- váhy těchto modelů jsou buď náhodně inicializované, nebo jsou sítě předučené, typicky na datové sadě **ImageNet**

pretrained_model.ipynb

- praktická ukázka využití předučeného modelu

Datová sada ImageNet

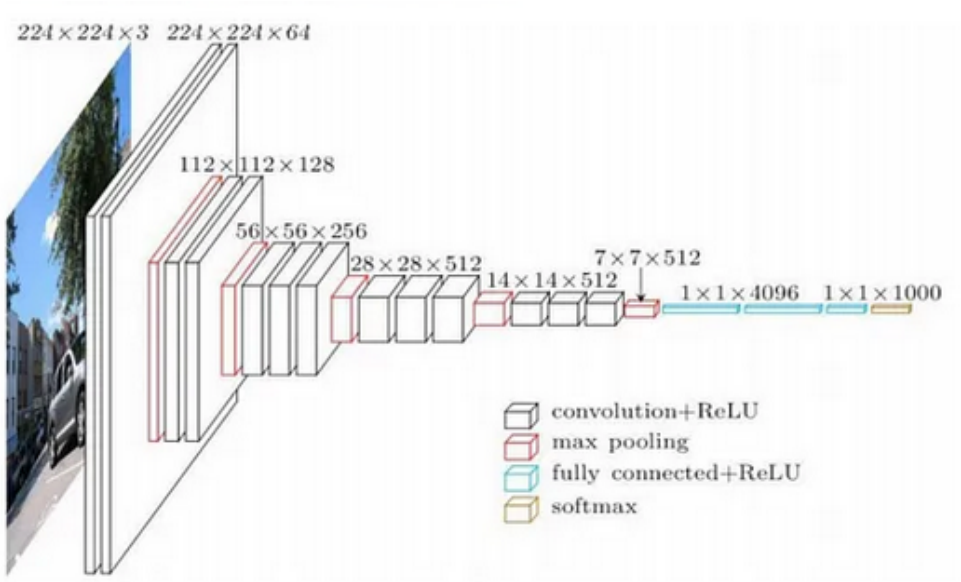
- 16 milionů barevných obrázků z 20 tisíc kategorií
- Vznikla v rámci soutěže ImageNet Large-Scale Visual Recognition Challenge (ILSVRC, 2010-2017)
- Tato soutěž odstartovala boom konvolučních neuronových sítí v rozpoznávání obrazu
- ImageNet se stal standardní referenční sadou pro porovnávání modelů (nahradil MNIST)



Zdroj: https://cs.stanford.edu/people/karpathy/cnnembed/cnn_embed_full_1k.jpg

Příklad předučeného modelu: VGGNet

- Karen Simonyan a Andrew Zisserman, 2014, rodina modelů (např. VGG16, VGG19)
- Klasická bipyramidální architektura, poměrně mělký model (16, resp. 19 vrstev)



Zdroj: https://medium.com/nerd-for-tech/vgg-16-easiest-explanation-12453b599526

Použití předučeného modelu

- Obrazová data je třeba převést na požadované rozměry a typicky do RGB

- Požadovaná velikost obrázků může být u každého modelu jiná (ale zpravidla bývá poměrně malá, kolem 200×200)
- Obrázky se obvykle přeškálovávají a popř. ořezávají

Od předučeného modelu k přenesenému učení (transfer learning)

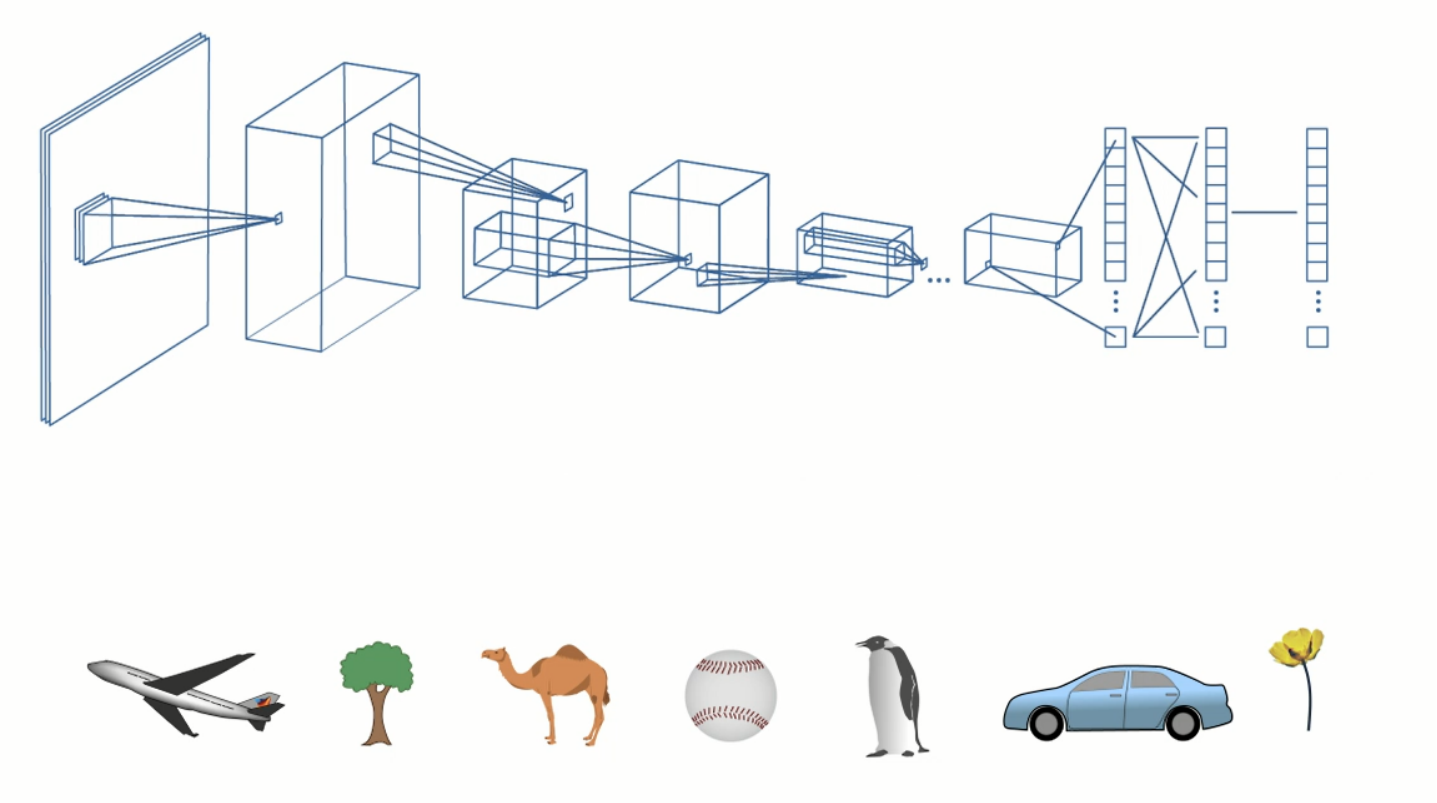
Použití předučeného modelu je skvělý začátek - ale obvykle to takhle jednoduše nepůjde

- ImageNet sice obsahuje 20 000 tříd, ale v případě našich dat i tak klasifikace nemusí být příliš přesná.
- viz. náš příklad: **pretrained_model.ipynb**

Ale jak přesně na to?

Od předučeného modelu k přenesenému učení

- Náš naučený model klasifikuje do 20000 tříd:

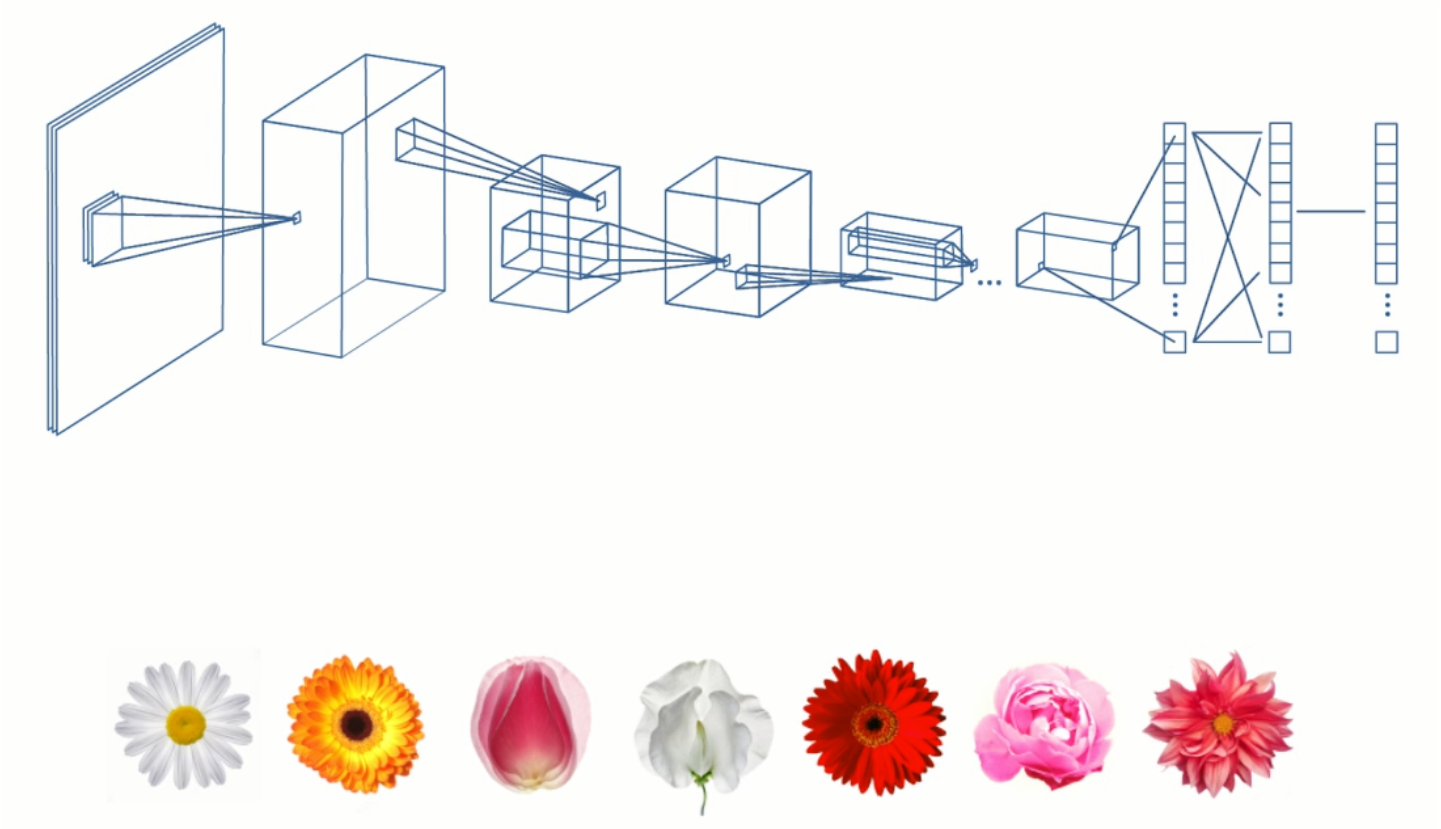


Zdroj :

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

Od předučeného modelu k přenesenému učení

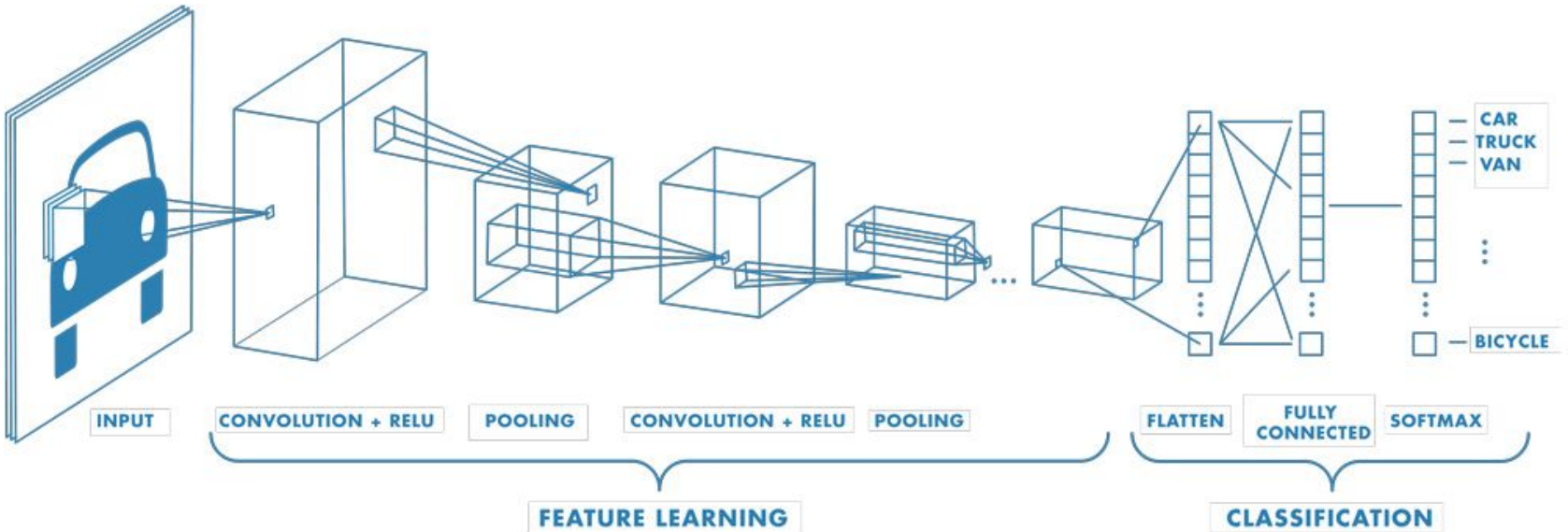
- je třeba klasifikovat do trochu jiných tříd:



Zdroj :

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

Připomenutí: Architektura konvoluční neuronové sítě



Části konvoluční neuronové sítě

- Konvoluční báze - pro extrakci hierarchických příznaků
- Flattening vrstva - převede data na vektor čísel
- Vrstevnatá neuronová síť (plně propojené vrstvy) pro klasifikaci ... **klasifikační hlava**

Zdroj :

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

Od předučeného modelu k přenesenému učení (transfer learning)

Jak přesně na to? ... první nápad:

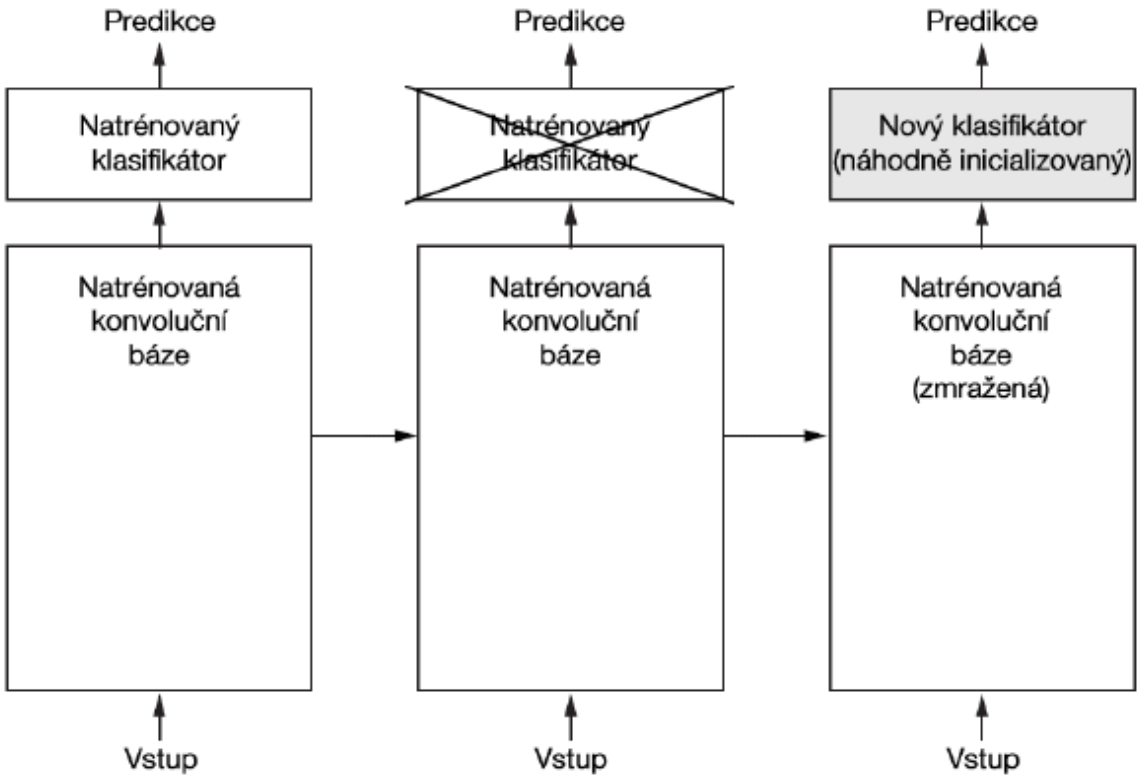
- Vezmeme síť natrénovanou na ImageNetu
- Odstraníme její klasifikační hlavu
- Tu použijeme na extrakci příznaků z dat ... vytvoříme trénovací množinu
- Vytvoříme novou vrstevnatou neuronovou síť a naučíme ji na extrahovaných příznacích

Tento přístup je velmi efektivní, ale často nepraktický

- Statický přístup, nelze jednoduše použít knihovní metody pro augmentaci dat

Přenesené učení (transfer learning)

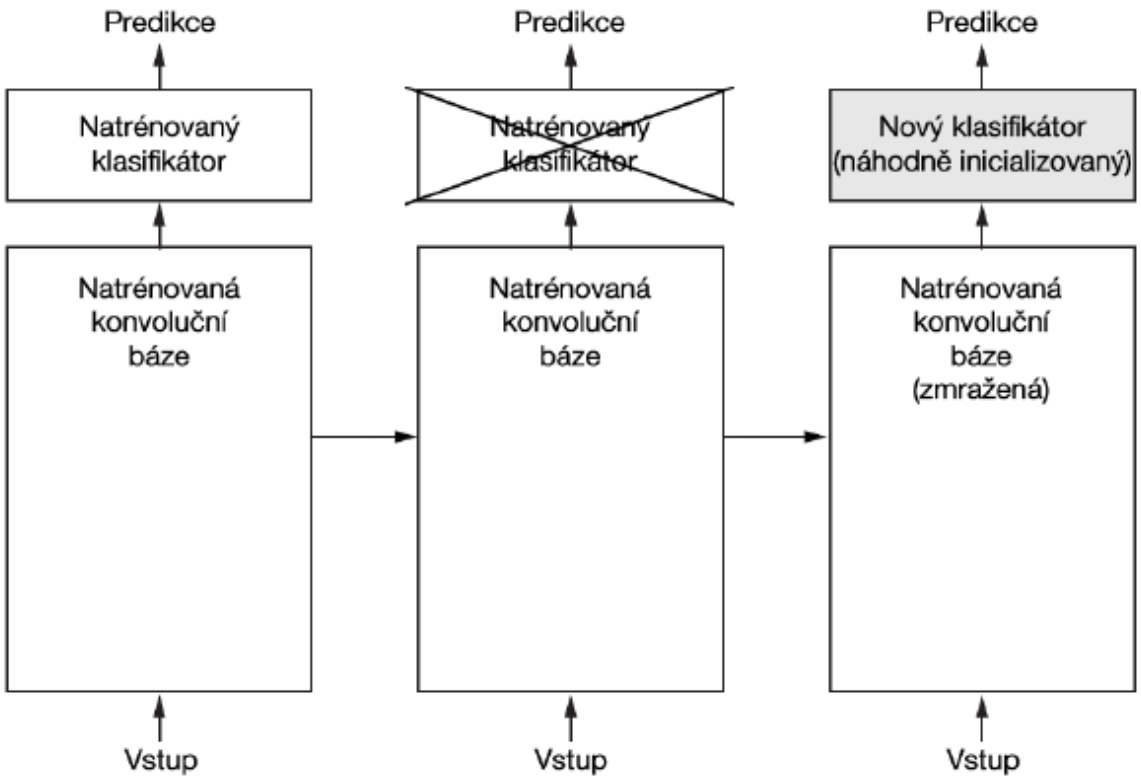
- Vezmeme síť natrénovanou na ImageNetu
- Klasifikační hlavu neuronové sítě (nebo jen její vrchní část) nahradíme novou (náhodně inicializovanou)



F. Chollet: Deep learning v jazyku Python, obr. 8.2

Přenesené učení (transfer learning)

- Novou klasifikační hlavu doučíme na nových datech (váhy předchozích vrstev zafixujeme, neboli ,,zmrazíme")



F. Chollet: Deep learning v jazyku Python, obr. 8.2