Deep Learning pro počítačové vidění

Pokyny k online schůzce

1. Používejte nejlépe desktopovou aplikaci Teams.

2. Pokud nekladete dotaz nebo se neúčastníte diskuze, mějte prosím vypnutý mikrofon.

3. Pokud jen trochu můžete, mějte puštěnou kameru. Vás výraz tváře pomůže vyučujícímu :)

4. Jak můžete položit dotaz:
   a. Zapněte si mikrofon a rovnou se zeptejte.
      nebo:
   b. Napište dotaz do chatu. nebo:
   c. Použijte tlačítko zvednout ruku.
      (Po vyvolání ruku sundejte)

Agenda

- Konvoluční neuronové sítě
- Datasety MNIST
- Dataset Dogs vs. Cats
- Použití předtrénovaného modelu

Konvoluční neuronové sítě

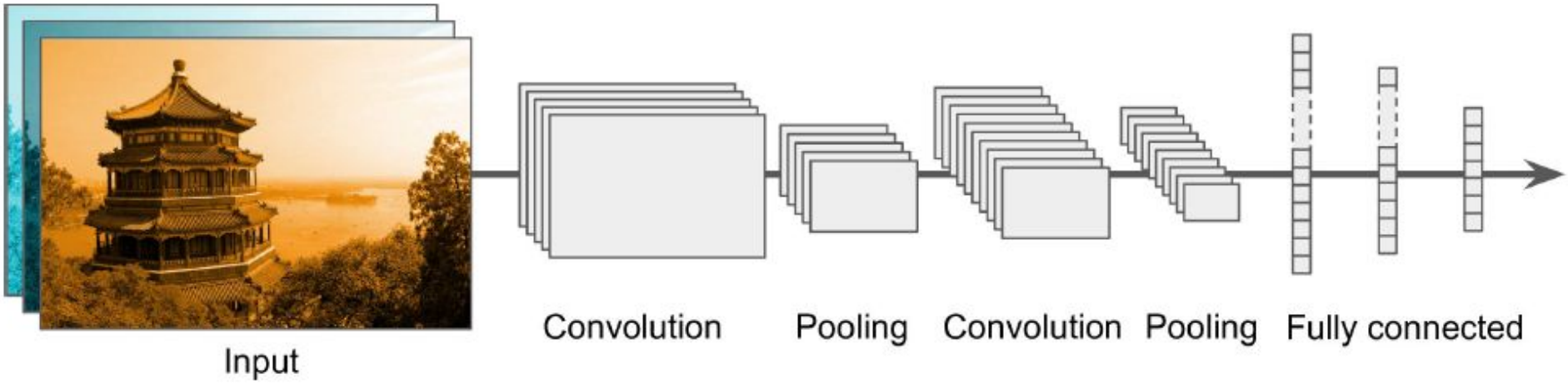
Úvod do konvolučních neuronových sítí

- CNN = convolutional neural network
- vycházejí z principu, na kterém funguje zraková kůra v mozku
- objevily se již v 80. letech, ale výrazného zlepšení se dosáhlo až v posledních 10 letech
  - výpočetní výkon
  - množství trénovacích dat
  - nové postupy pro trénování hlubokých neuronových sítí (problémy s propagací gradientu)
- oblasti použití CNN
  - rozpoznávání obrazu a vyhledávání obrázků podle obsahu
  - automatická klasifikace obsahu videa
  - autonomní vozidla
  - rozpoznávání hlasu
  - zpracování přirozeného jazyka

Princip konvolučních neuronových sítí

- neuronové sítě, se kterými jsme doteď pracovali, používaly tzv. hustě propojené vrstvy (dense layer)
  - všechny neurony v jedné vrstvě jsou propojené se všemi neurony v nižší vrstvě
- konvoluční neuronové sítě zavádějí dva nové typy vrstev:
  - konvoluční vrstva Conv2D
  - sdružovací vrstva MaxPooling2D
- celá neuronová síť je pak tvořena z několika vrstev Conv2D a MaxPooling2D a nad nimi je pak jedna či více klasických hustě propojených dense vrstev
- dense vrstvy jsou sestavené z (dlouhé) řady neuronů, proto i vstupy do dense vrstev jsou v 1D (např. MNIST 2D 28x28 => 1D 784)
  - konvoluční vrstvy oproti tomu pracují s 2D, díky čemuž je možné zachovat prostorové uspořádání

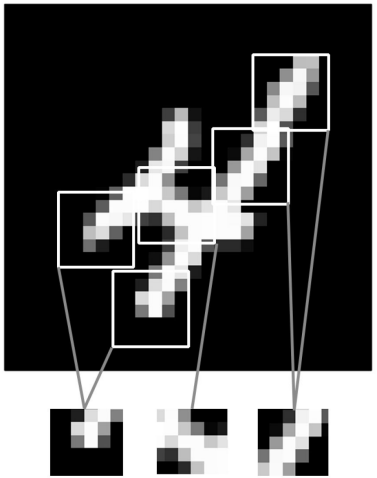
Typická architektura CNN



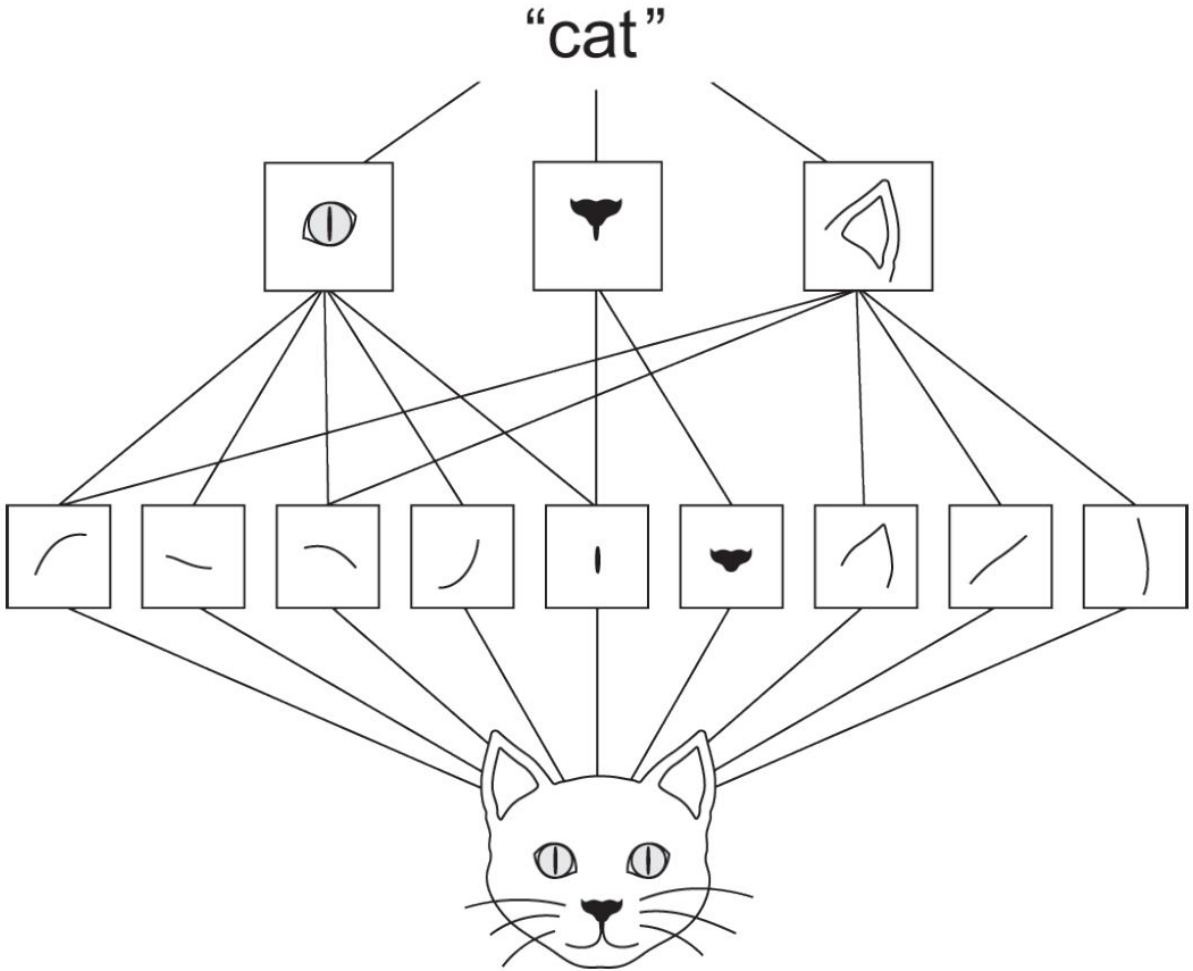
Zdroj: (Géron 2019)

Konvoluční vrstva Conv2D

- dense vrstva se učí globální vzory ve vstupních příznacích
  - v případě MNIST číslice jsou to vzory zahrnující všechny vstupní pixely najednou
- konvoluční vrstva se učí lokální vzory
  - typicky “kouká” na malá 2D okna ve vstupních příznacích o velikosti 3x3 nebo 5x5
  - lokálním vzorem může být např. diagonální hrana, svislá čára a podobně
- klíčové vlastnosti konvoluční vrstvy:
  - naučené vzory jsou invariantní z hlediska umístění
    - vzor naučený v pravém dolním rohu obrazu pak dokáže konvoluční síť rozpoznat i kdekoliv jinde
    - oproti tomu dense vrstva by se musela vzor naučit znovu, pokud by se objevil na jiném místě
    - skutečný vizuální svět je z hlediska umístění invariantní => výrazně větší efektivita CNN než DNN
      - stačí mnohem méně trénovacích příkladů a méně parametrů
  - učí se prostorové hierarchie vzorů
    - první vrstva se naučí malé lokální vzory jako jsou hrany
    - druhá konvoluční vrstva se naučí větší vzory tvořené kombinací vzorů z první vrstvy, např. čtverce, trojúhelníky a podobně
    - takto se konvoluční síť dokáže efektivně naučit stále komplexnější a abstraktnější vizuální pojmy



Zdroj: (Chollet 2021)



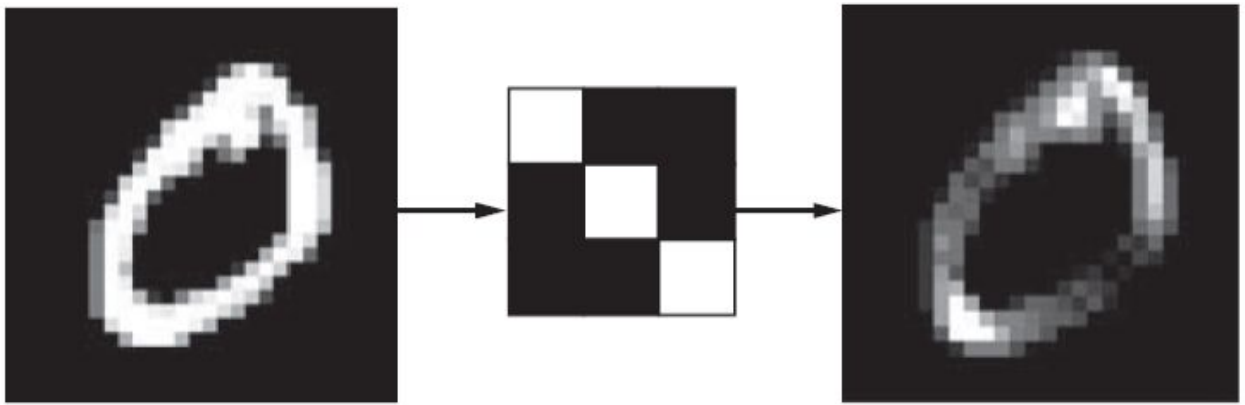
Zdroj: (Chollet 2021)

Konvoluční operace

- konvoluce pracují s 3D tenzory, které se nazývají mapy příznaků (feature map)
  - dvě prostorové osy (šířka, výška)
  - osa hloubky (depth), nazývaná také jako osa kanálů (channels)
  - příklad RGB obrázku: osa hloubky má rozměr 3 (tři barevné kanály), černobílý obrázek 1
- konvoluční operace postupně extrahuje z mapy příznaků malé oblasti (patches) a aplikuje na ně jednu a tu samou transformaci (filtr)
  - vznikne mapa výstupních příznaků
- mapa výstupních příznaků je stále 3D tenzor
  - hloubka už však neodpovídá počtu vstupních kanálů, ale může být libovolná (hyperparametr)
    - jednotlivé kanály v této ose hloubky představují filtry
      - filtr na nízké úrovni může rozpoznávat např. diagonální hrany
      - na vysoké úrovni zase filtr může kódovat např. přítomnost tváře ve vstupních datech

Konvoluční operace

- budeme si ukazovat Conv2D na datasetu MNIST
  - první konvoluční vrstva má na vstupu mapu příznaků velikosti (28, 28, 1) a na výstupu mapu příznaků o velikosti (26, 26, 32)
    - aplikuje tedy 32 různých filtrů (ty se sama naučí) na svůj vstup
    - každý kanál v této ose hloubky tedy obsahuje mřížku 26x26 hodnot, tzv. mapa odpovědí (response map) na konkrétní filtr ve vstupních datech



Zdroj: (Chollet 2021)

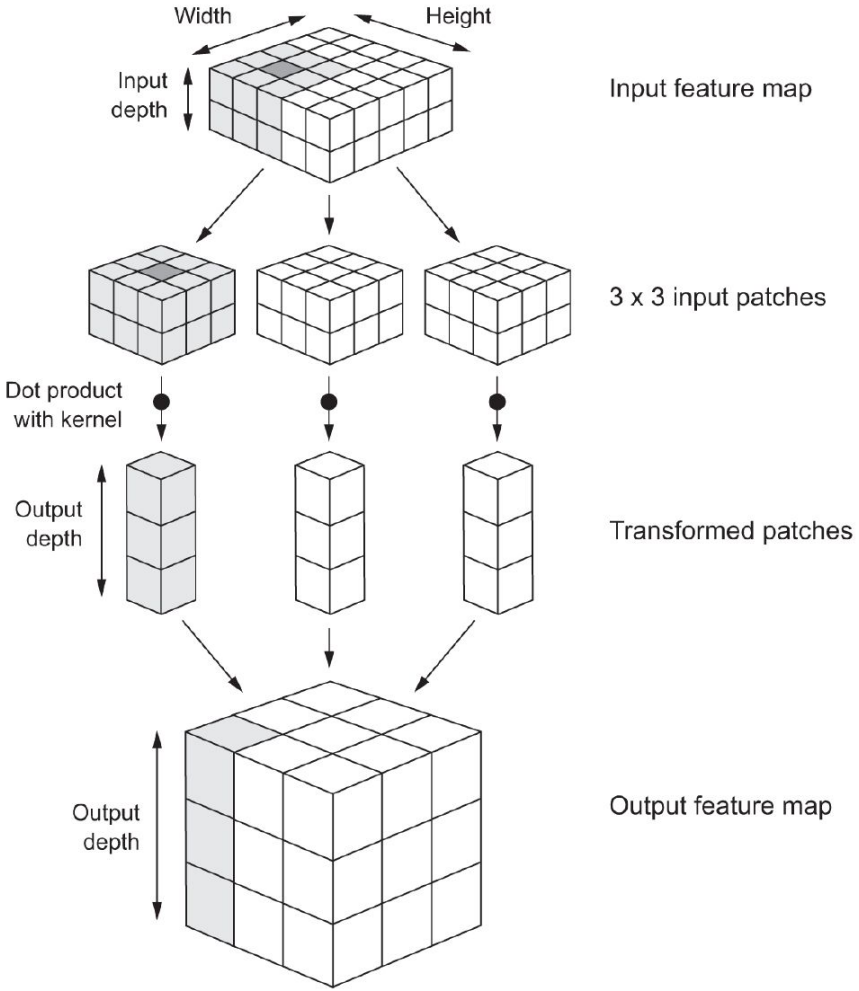
Parametry konvoluční operace

- velikost oblastí extrahovaných ze vstupů
  - typicky 3x3 (nejčastější volba) nebo 5x5
- hloubka mapy výstupních příznaků
  - tj. počet různých filtrů, které konvoluce počítá
  - typicky se začíná s menším množstvím filtrů a postupně se v dalších vrstvách počet zvyšuje

- `Conv2D(output_depth, (window_height, window_width))`

Postup konvoluční operace

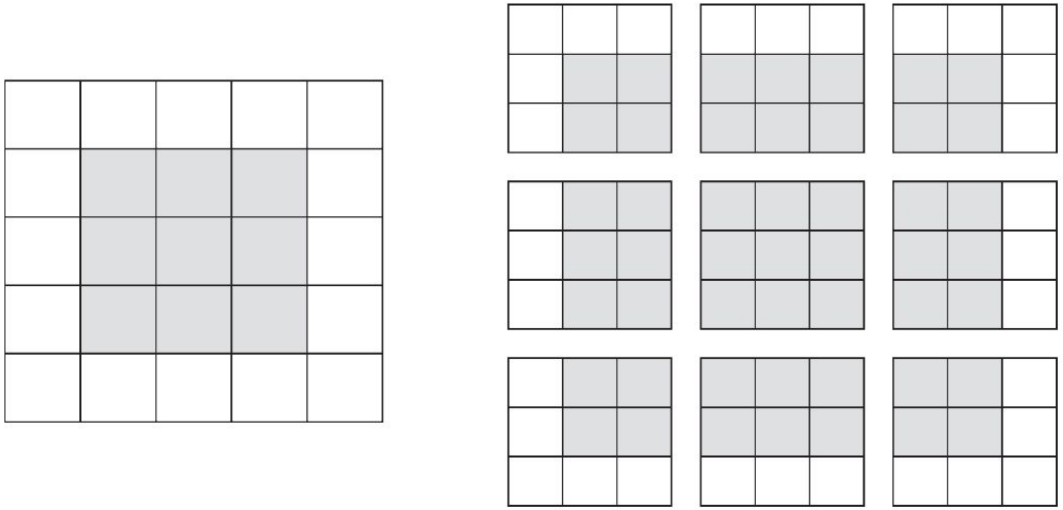
- konvoluce posouvá okna zvolené velikosti (3x3) přes mapu vstupních příznaků
  - zastaví na každém možném místě
  - extrahuje 3D oblast okolních příznaků (tvar (window_height, windows_width, input_depth))
  - každá tato 3D oblast je transformována na 1D vektor tvaru (output_depth)
    - transformace je skrz součin s váhovou maticí, která se nazývá konvoluční jádro (convolution kernel); konvoluční jádro představuje konkrétní filtry
  - prostorové umístění na mapě vstupních příznaků odpovídá umístění na mapě výstupních příznaků
- výška a šířka výstupu se může lišit od výšky a šířky vstupu
  - okrajové efekty
  - použití prodloužených kroků (strides)



Zdroj: (Chollet 2021)

Okrajové efekty a “vycpávky”

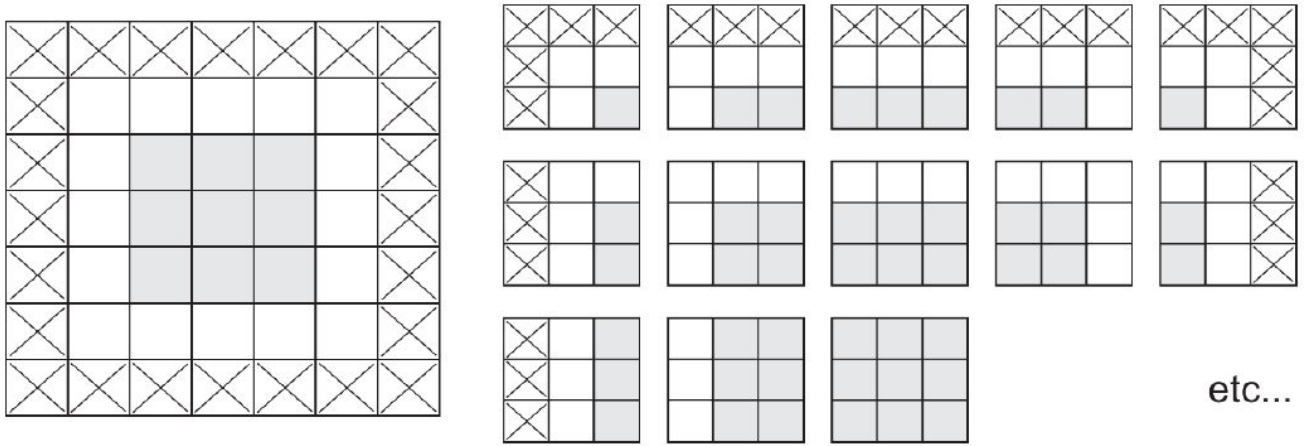
- máme-li mapu příznaků o velikosti 5x5, pak můžeme okna velikosti 3x3 umístit jen třemi možnými způsoby horizontálně i lineárně, vznikne tedy menší výstupní mapa o velikosti 3x3 místo 5x5



Zdroj: (Chollet 2021)

Padding

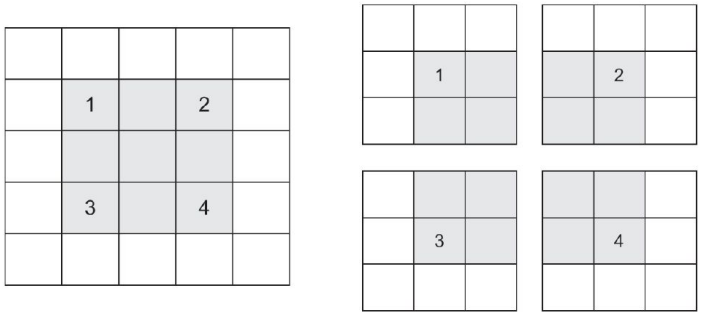
- vycpávky (padding) jde použít, pokud chceme získat výstupní mapu stejných rozměrů, jako je vstupní mapa příznaků
  - musí se tedy přidat příslušný počet sloupců a řádků na okraj vstupní mapy, aby šlo okno umístit
  - defaultně se používá varianta bez vycpávek



Zdroj: (Chollet 2021)

Konvoluční kroky (strides)

- velikost výstupní mapy může ovlivnit také velikost kroku (stride)
- zatím jsme pracovali s variantou, že středové dlaždice konvolučních oken k sobě přiléhají
  - vzdálenost mezi dvěma po sobě jdoucími okny je však hyperparametr označovaný jako krok (stride; výchozí hodnota 1)
- pokud použijeme krok 2, zmenší se velikost výstupní mapy na polovinu
  - obvykle se ale krokované konvoluce v klasifikačních úlohách nepoužívají



Zdroj: (Chollet 2021)

Sdružovací vrstva MaxPooling2D

- vrstva MaxPooling2D se používá k rapidnímu snížení velikosti mapy příznaků
  - ve výchozím nastavení na polovinu, tedy např. u MNIST je výstup Conv2D 26x26 a MaxPooling2D ho následně sníží na 13x13
- max pooling také používá extrakci pomocí oken z mapy vstupních příznaků
  - výstupem je však maximální hodnota každého kanálu (filtru) z okna
  - typicky je okno velikosti 2x2 a krok (stride) 2
    - oproti tomu konvoluční vrstva využívá 3x3 a krok 1
- proč se MaxPooling2D používá:
  - bez něj se model nenaučí prostorovou hierarchii příznaků na dostatečně velké ploše
    - např. model se 3 konvolučními vrstvami bez max poolingu by v poslední 3. vrstvě měl informace, které pocházejí z oken velikosti pouze 7x7 ve vstupní vrstvě (obrázku)
      - např. MNIST - rozpoznat číslici jen pomocí okna velikosti 7x7 by dost dobře nešlo
  - mapa příznaků by byla příliš velká s velkým množstvím parametrů => intenzivní přeučování
- alternativně se používal average pooling, ale v praxi lépe funguje max pooling
  - prostorová přítomnost určitého vzoru se lépe zjistí skrz jeho maximální přítomnost než průměrnou přítomnost

Ukázka na datasetu MNIST

- viz jupyter

Ukázka na datasetu Fashion MNIST

- viz jupyter

Paměťová a výpočetní náročnost

- příklad: konvoluční vrstva 5x5, 200 filtrů, velikost 150x100, vstupy RGB
  - počet parametrů: (5x5x3+1)*200 = 15200
    - hustě propojená vrstva stejné velikosti by měla 675 M parametrů
  - počet operací na jeden průchod: (5x5x3)x200x(150x100) = 225 M operací s float32
  - potřebná pamět: 200x(150x100)x32 bitů = 12 MB RAM na jednu instanci a jednu vrstvu
    - 100 instancí v rámci jedné dávky = 1.2 GB jen na jednu vrstvu
    - při trénování se musí držet v paměti výstupy všech vrstev
      - při predikcích už pak stačí jen výstup dvou po sobě následujících vrstev
- pokud trénování padá kvůli nedostatku RAM:
  - snížit velikost mini-dávky (batch_size)
  - snížit počet vrstev
  - snížit dimenzionalitu (použítím kroku nebo menší velikosti)
  - použít 16bitové parametry místo 32bitových

Úlohy s malými datasety

- za malý dataset se v případě počítačového vidění považují stovky až tisíce obrázků
  - teoreticky pro malý a dobře regularizovaný model mohou stačit řádově stovky instancí
- lze využít několik strategií, jak tuto úlohu řešit
  - natrénovat model od začátku na dostupném datasetu
  - vzít předtrénovaný model, použít ho pro extrakci příznaků a nad tím pak natrénovat klasifikátor
  - vzít předtrénovaný model a doladit ho pro potřeby naší úlohy
- modely hlubokého učení jsou totiž značně znovupoužitelné
  - řada výkonných modelů počítačového vidění je nyní veřejně dostupná
    - bývají natrénovány obvykle na tzv. Image-Net datasetu
      - cca 14 M obrázků řazených do 1000 tříd

Dataset Dogs vs. Cats

- https://www.kaggle.com/c/dogs-vs-cats/
  - stáhnout soubor train.zip (cca 500 MB)
- obrázky psů a koček
  - použijeme 4 tisíce fotek (2 tisíce psů, 2 tisíce koček)
  - originální dataset obsahuje 25 tisíc fotek, nicméně v praxi se obvykle setkáváme s malým množstvím dat
- budeme trénovat klasifikátor, který dokáže rozlišit kočku a psa
- viz jupyter

Dropout

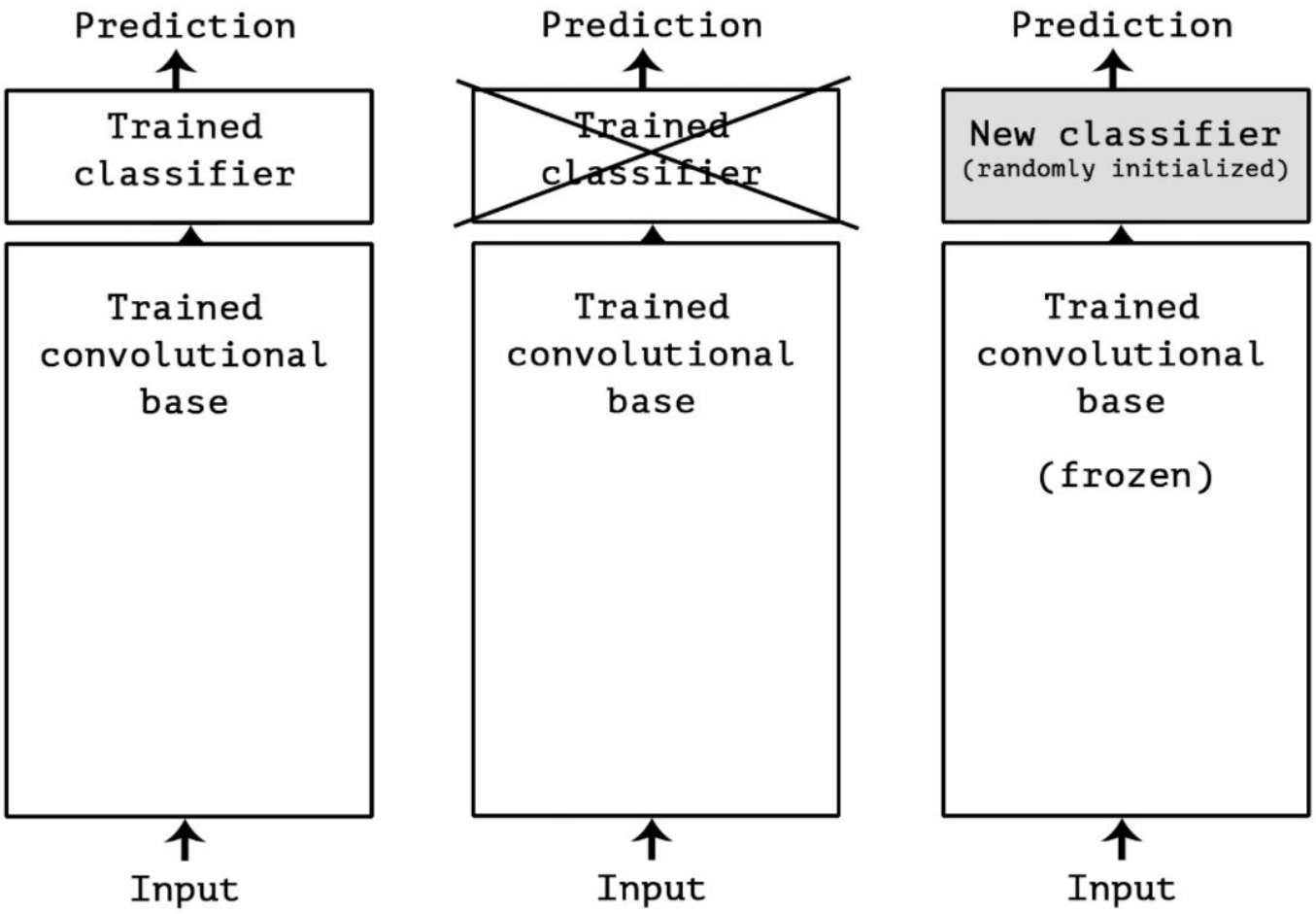
- jedna z nejpopulárnějších regularizačních technik pro neuronové sítě
- princip:
  - během trénování má každý neuron ve vrstvě, po které následuje dropout, určitou pravděpodobnost, že bude “vyhozen”
    - tzn. tento neuron jako by v síti vůbec nebyl
  - obvykle se pro CNN používá hodnota 40 - 50 %
  - po trénování, tedy při predikcích, se dropout už nepoužívá
- myšlenka, na které je dropout postaven:
  - aby se neurony začaly přeučovat, musí spolu spolupracovat
  - pokud se bude stávat, že některé neurony najednou zmizí, musí se ostatní neurony naučit fungovat více obecně a nebýt závislé na konkrétních vstupech

Použití předtrénovaných modelů pro vlastní modely

- tzv. transfer learning
- běžná a vysoce účinná metoda, jak dosáhnout dobrých výsledků i s malým datasetem
- předtrénovaný model byl typicky trénován na velkém datasetu jako Image Net
  - lze předpokládat, že se naučil prostorovou hierarchii příznaků tak dobře, že může být použit jako obecný model vizuálního světa
  - díky tomu mapy příznaků, které model produkuje, mohou být užitečné i pro jiné úlohy počítačového vidění a to dokonce i ty, které používají úplně jiné třídy, než na kterých byl model trénován
    - např. Image Net obsahuje zejména fotografie každodenních objektů a zvířat, ale můžeme ho použít jako základ pro model rozpoznávající konkrétní typy a modely nábytku
  - tato přenositelnost je unikátní pro modely hlubokého učení na rozdíl např. od mělkých modelů

Předtrénované modely

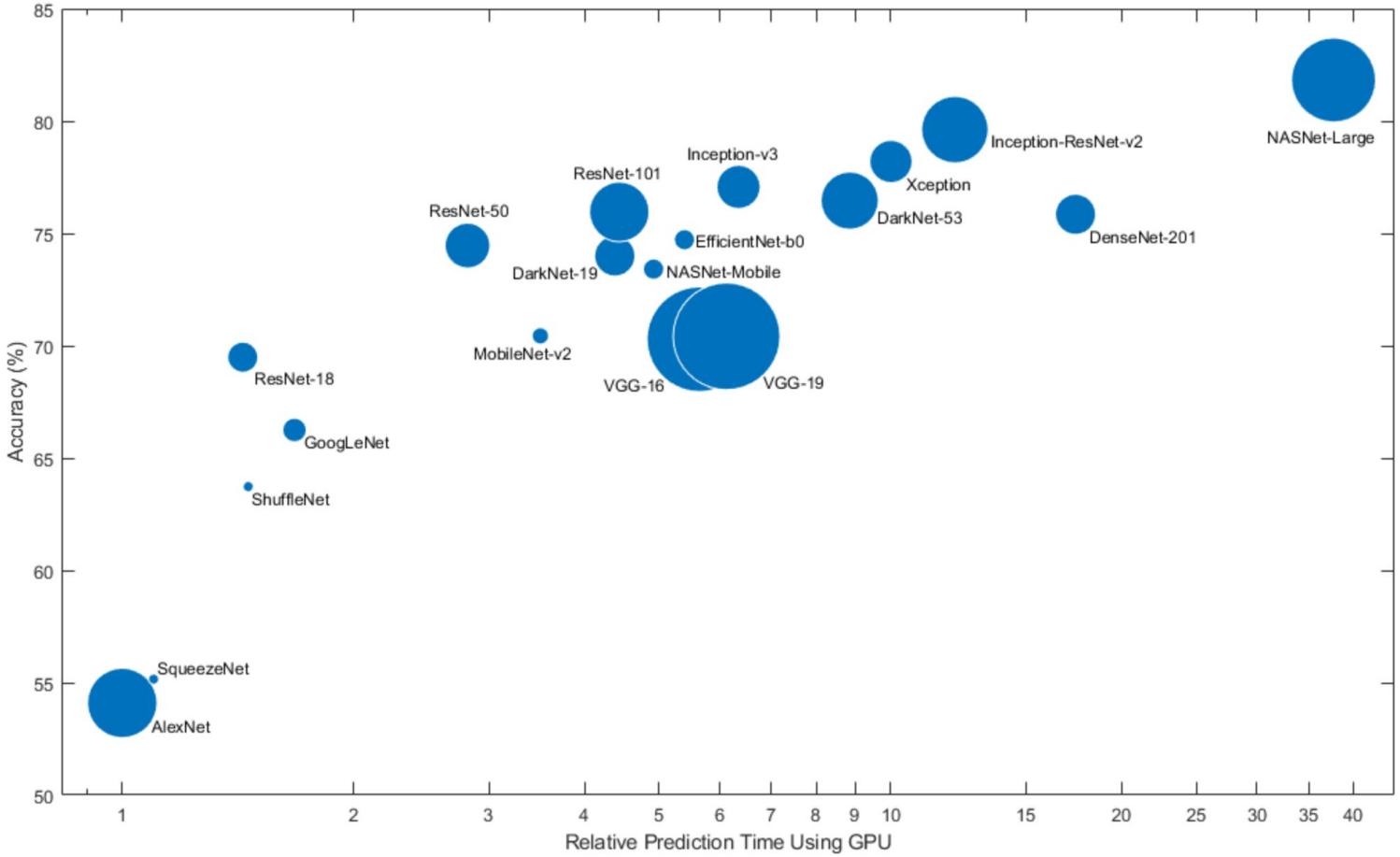
- dva způsoby použití předtrénovaného modelu:
  - extrakce příznaků
    - použije se tzv. konvoluční báze z předtrénovaného modelu (tj. vše kromě hustě propojených vrstev na vrcholu sítě)
    - pomocí konvoluční báze se získá mapa příznaků pro vstupní data, toto pak slouží jako vstup pro nově trénovaný klasifikátor tvořený jen z hustě propojených vrstev
      - hustě propojené vrstvy z původního modelu nejde použít, protože jsou specificky naučené pro úlohu, na kterou byly trénovány
  - jemné doladění modelu
    - použije se konvoluční báze a nahradí se hustě propojené vrstvy na vrcholu modelu
    - trénování se pak provádí tak, že v první fázi jsou “zmražené” váhy v konvoluční bázi a trénují se pouze hustě propojené vrstvy nahoře
      - ve druhé fázi se pak rozmrazí i konvoluční báze a trénuje se celý model



Zdroj: (Chollet 2021)

Předtrénované modely CNN

- LeNet-5
- AlexNet
- GoogLeNet
- VGGNet
  - tento použijeme
  - sice není nejnovější a nejvýkonnější, ale je velmi podobný tomu, co jsme si už ukazovali
- ResNet
- Xception
- SENet
- https://keras.io/api/applications/

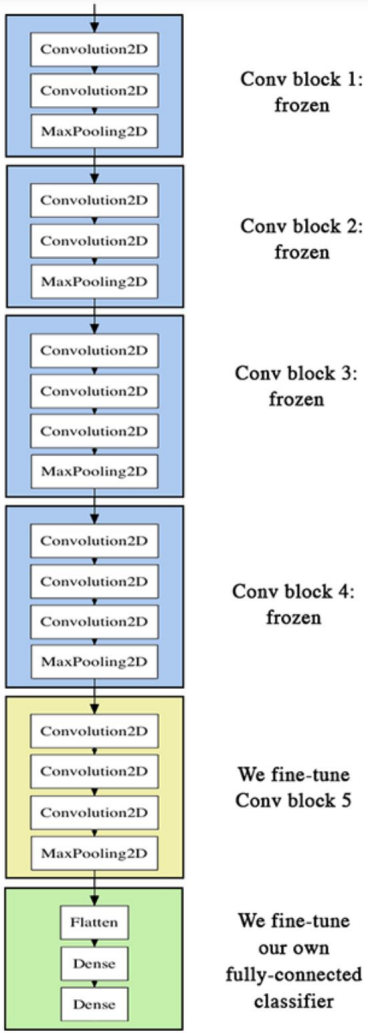

Accuracy (%)
Relative Prediction Time Using GPU
85
80
75
70
65
60
55
50
1
2
3
4
5
6
7
8
9
10
15
20
25
30
35
40
NASNet-Large
Inception-ResNet-v2
Xception
Inception-v3
ResNet-101
ResNet-50
DarkNet-53
DenseNet-201
EfficientNet-b0
DarkNet-19
NASNet-Mobile
MobileNet-v2
VGG-16
VGG-19
ResNet-18
GoogLeNet
ShuffleNet
SqueezeNet
AlexNet

Praktická ukázka předtrénovaného modelu

viz jupyter

Jemné doladění modelu

- vezmeme natrénovanou konvoluční bázi a nad ní dáme dense klasifikátor
- zmrazíme konvoluční bázi a natrénujeme dense vrstvy
- rozmrazíme několik horních konvolučních vrstev a trénujeme znovu
- viz jupyter



Zdroj: (Chollet 2021)

Použití předtrénovaných modelů pro klasifikaci

- ukázka klasifikace obrázků pomocí natrénovaných modelů
- viz jupyter dogs_cats_predict
- viz jupyter resnet50

Úkoly

- zkuste na datasetu MNIST dosáhnout co nejlepšího výsledku
  - lze se dostat někam k 99,5 - 99,7 % správnosti na testovacím setu
  - zkuste využít data augmentantion, dropout, learning rate scheduling, ...
- zkuste na datasetu Fashion MNIST dosáhnout co nejlepšího výsledku
- zprovozněte si doma nebo v cloudu (Google Colab, Amazon Cloud) prostředí tak, aby bylo možné spustit ukázky z dnešní hodiny
  - jak nahrát dataset na Google Colab: https://towardsdatascience.com/google-colab-import-and-export-datasets-eccf801e2971
  - dokážete modely doladit na ještě lepší výsledky?
- máte-li k dispozici dostatečný výpočetní výkon, zkuste natrénovat model na celém Dogs vs. Cats datasetu.
  - na jakou nejlepší hodnotu správnosti se dokážete dostat? Můžete zkusit využít i jiné předtrénované modely (např. Xception)
- vyzkoušejte si klasifikaci různých obrázků, které najdete, na různých předtrénovaných modelech (keras.applications)

Zdroje

- Coelho, L. P.; Richert, W. (2013) Building machine learning systems with Python. Birmingham: Packt Publishing. ISBN 978-1-78216-140-0.

- Géron, A. (2019) Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition. O'Reilly Media, Inc. ISBN 9781492032649.

- Chollet, F. (2021) Deep Learning with Python, Second Edition. Manning Publications. ISBN 9781617296864.

- Segaran, T. (2007) Programming collective intelligence: building smart web 2.0 applications. Beijing: O'Reilly Media. ISBN 0-596-52932-5.