

# Počítačové videnie - Úvod do deep learningu

Ing. Viktor Kocur  
victor.kocur@fmph.uniba.sk

DAI FMFI UK

5.12.2018

- 1 Inštalácia
- 2 Evaluácia
  - Evaluácia
- 3 Tensorboard
  - Tensorboard
- 4 Konvolučné neurónové siete
  - Konvolučné vrstvy
  - Poolingové vrstvy
- 5 Transfer learning
  - Rozdelenie dát

# Inštalácia

## Verzie

Na Windowsoch v škole je python 3.7, ale ten nieje supported tensorflowom. Preto budeme odteraz pracovať v Linuxe, ale ak máte vlastný počítač je to jedno. Kto má grafickú kartu od nvidia, môže si na stránkach tensorflowu nájsť inštalačné inštrukcie a inštalovať tensorflow s podporou gpu.

## pip3

```
pip3 install --user tensorflow
pip3 install --user tensorboard
pip3 install --user keras
```

## Keras - kód

Ak nemáte tak si stiahnite trénovací kód k minulému cviku.

# Evaluácia

## Evaluácia

Väčšinou chceme model overiť na celom datasete. Preto musíme testovaciu množinu dostať do rovnakej formy ako trénovaciu množinu.

## Kód

```
x_test = np.reshape(x_test, (10000, 784)).\n           astype(np.float32)/255\ny_test = keras.utils.to_categorical(y_test, n_cls)\nscore = model.evaluate(x_test, y_test)
```

# Tensorboard

## Tensorboard

Tensorboard je nástroj na sledovanie vývoja učenia, kontrolovanie grafu neurónovej siete a ďalšie užitočné veci.

## Kód - pred model.fit

```
tb_callback = keras.callbacks.TensorBoard(  
    log_dir='./logs')  
callbacks.append(tb_callback)
```

## Cez shell spustíme

```
tensorboard --logdir=logs
```

# Conv2D

## Maticové usporiadanie

Keďže prejdeme na konvolučné neurónové siete, tak chceme mať na vstupe obrázky v tvare 28x28.

## Úloha

Upravte trénovacie dáta tak, aby obrázky mali tvar (počet, 28, 28, 1). Pozrite si dokumentáciu `keras.layers.Conv2D` v `kerase`. Použite na začiatku modelu Conv2D vrstvy.

# Riešenie

## Riešenie

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu'))
model.add(Conv2D(32, (3, 3), activation='relu'))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(n_cls, activation='softmax'))
```

# Poolingové vrstvy

## Úloha

Pozrite si dokumentáciu a `keras.layers.MaxPooling2D` a pridajte ju do konvolučného modelu.

## Úloha

V dokumentácii sa pozrite aj na iné poolingové vrstvy.



# Riešenie

## Riešenie

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(n_cls, activation='softmax'))
```

# DropOut a BatchNorm

## Úloha

Pozrite si dokumentáciu a `keras.layers.Dropout` a pridajte ju na vhodné miesto do konvolučného modelu.

## Úloha

Pozrite si v dokumentácii `BatchNorm`. Síce na tak malú sieť to asi nieje dobrá vrstva skúste ju pridať do modelu.

# Riešenie

## Riešenie

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(n_cls, activation='softmax'))
```

# Transfer learning

## Nedostatok dát

Ak máme málo dát, tak je veľmi malá šanca, že sa nám podarí natrénovať hlbokú sieť z inicializácie. Našťastie môžeme použiť sieť natrénovanú na iných dátach!

## transfer\_train.py

V zipe s kódom k dnešnému cvičeniu je súbor `transfer_train.py`. Ten si teraz zanalyzujeme. Dataset k úlohe si taktiež stiahnete zo stránky. Originálne pochádza z <https://www.kaggle.com/huan9huan/walk-or-run>

## Úloha

Modifikujte kód tak, aby sa dali trénovať aj nejaké posledné konvolučné vrstvy, alebo aj celá sieť. Skúste upraviť augmentáciu. Dá sa dosiahnuť lepší výsledok?