Horse or human

František Forgáč

Dataset

- Pozostáva z obrázkov (1283)
  - Trénovacie dáta (1027)
    - Obrázky koní (500)
    - Obrázky ľudí (527)
  - Testovacie dáta (256)
    - Obrázky koní (128)
    - Obrázky ľudí (128)
- Typ .png
- Rozmery 300x300

Testovací obrázok koňa

Testovací obrázok človeka

Trénovací obrázok koňa

Trénovací obrázok človeka

Príprava dát

- Matica
- Knižnica cv2 - openCV
  - Funkcia COLOR_BGR2GRAY

In [3]:
```
# nacitanie jedneho obrazku, aby som videl ako vyzera, ake data nacitavam, ako vyzera vysledny obrazok - teda aj matica, s ktorou pracujem
priklad = img.imread("horse-or-human/train/horses/horse01-6.png")
priklad = cv2.cvtColor(priklad, cv2.COLOR_BGR2GRAY) # vytvorenie matice 2D obrazku, pouzivam funkciu z kniznice cv2
print(priklad.dtype)
print(priklad.shape)
plt.imshow(priklad)
plt.axis('off')
plt.show()
```

```
float32
(300, 300)
```

Načítanie dát

- Umiestnené v root priečinku
  - trening_kone_cesta = ("horse-or-human/train/horses/")
- Načítavam všetky obrázky z danej kategórie do zoznamu
  - trening_kone = []
- Výpis využívam aj ako overenie, či som prešiel všetkými dátami
- Proces zopakujem aj pre dáta z ostatných kategórií

```
for png in os.listdir(trening_kone_cesta):
    imageread = img.imread(trening_kone_cesta+png)
    imageread = cv2.cvtColor(imageread, cv2.COLOR_BGR2GRAY)
    trening_kone.append(imageread)
    #print(imgread.shape) # (300, 300)

print(len(trening_kone) , "obrazkov koni v priecinku s trenovacimi datami pre kone")
```

```
500 obrazkov koni v priecinku s trenovacimi datami pre kone
```

Klasifikácia

- 0 → zodpovedá hodnote pre kone
- 1 → zodpovedá hodnote pre ľudí

In [11]:
```
# vytvorenie klasifikacie dat
# 0 -> kone
# 1 -> ludia
zero = np.zeros(len(trening_kone) + len(test_kone)) # vsetky obrazky koni
one = np.ones(len(trening_ludia) + len(test_ludia)) # vsetky obrazky ludi
print("Celkovy pocet obrazkov ludi :", one.size)
print("Celkovy pocet obrazkov koni :", zero.size)
```

```
Celkovy pocet obrazkov ludi : 655
Celkovy pocet obrazkov koni : 628
```

Rozdelenie dát

- sklearn knižnica
  - train_test_split
- Podmnožiny
  - trénovacie
  - Testovacie

In [13]:
```
# pouzitie kniznice sklearn
# rozdeli polia alebo matice do nahodnych trenovacich a testovaich podmnozin
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_data, y, test_size = 0.25, random_state = 42)

number_of_train = x_train.shape[0]
number_of_test  = x_test.shape[0]

print("pocet trenovacich dat :", number_of_train)
print("pocet testovacich dat :", number_of_test)
```

```
pocet trenovacich dat : 898
pocet testovacich dat : 385
```

Flatten

- Metóda, ktorou z pôvodného n-dimenzionálneho pola dostanem jedno-dimenzionálne pole

In [14]:
```
# konvertovanie povodneho pola
# funkciou reshape (zmena tvaru) dostanem z povodnych hodnot 300x300 -> 90000, ako keby som 2D obraz chcel zobrazit v 1D
x_train_flatten = x_train.reshape(number_of_train, x_train.shape[1] * x_train.shape[2])
x_test_flatten = x_test.reshape(number_of_test, x_test.shape[1] * x_test.shape[2])

print("X train Flatten : ",x_train_flatten.shape)
print("X test Flatten : ",x_test_flatten.shape)
x_train = x_train_flatten
x_test = x_test_flatten
```

```
X train Flatten :  (898, 90000)
X test Flatten :  (385, 90000)
```

Klasifikátor

- Knižnica Keras
- Vytvorenie vrstiev neurónovej siete
- funkcia

In [16]:
```
# vytvorenie funkcie pre klasifikaciu - neuronova siet
def build_classifier():
    classifier = Sequential()
    classifier.add(Dense(units = 50, kernel_initializer = 'uniform', activation = 'relu', input_dim = x_train.shape[1])) # 1. skryta vrstva
    classifier.add(Dense(units = 40, kernel_initializer = 'uniform', activation = 'relu')) # 2. skryta vrstva
    classifier.add(Dense(units = 30, kernel_initializer = 'uniform', activation = 'relu')) # 3. skryta vrstva
    classifier.add(Dense(units = 20, kernel_initializer = 'uniform', activation = 'relu')) # 4. skryta vrstva
    #classifier.add(Dense(units = 20, kernel_initializer = 'uniform', activation = 'relu')) # netreba
    #classifier.add(Dense(units = 10, kernel_initializer = 'uniform', activation = 'relu')) # netreba
    classifier.add(Dense(units = 1, kernel_initializer = 'uniform', activation = 'sigmoid')) # vystupna vrstva

    classifier.compile(optimizer='adam', loss = 'binary_crossentropy', metrics=['accuracy'])
    return classifier
```

Trénovanie

- Použitie funkcie build_classifier()
- EarlyStopping

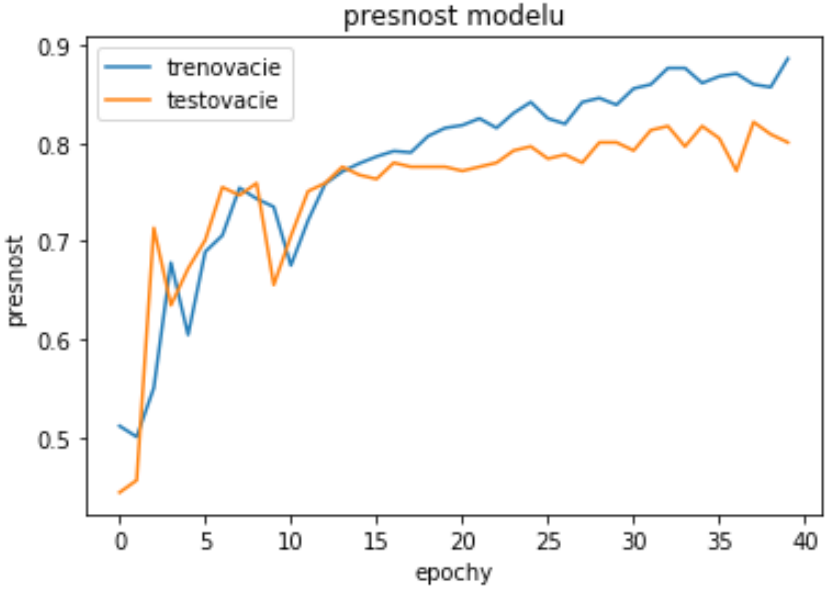
In [17]:
```
# zavolanie funkcie, ktora vracia klasifikaciu, ulozim ju do premennej cf, na ktorej potom pustim fukniu "fit"
cf = build_classifier()
# 'poistka' proti pretrenovaniu siete
earlyStop = EarlyStopping(monitor='val_loss', min_delta=0.001, patience = 10)
X = x_train
Y = y_train
# verbose=2 -> step-by-step vypis kazdeho epochu
# epoch -> pocet, kolkokrat chcem prejst vsetkymi trenovacimi datami odpredu a odzadu
# batch_size -> pocet vzoriek pouzitych pri jednom prechode
history = cf.fit(X, Y, validation_split=0.25, epochs=100, batch_size=300, verbose=2, callbacks = [earlyStop])
```

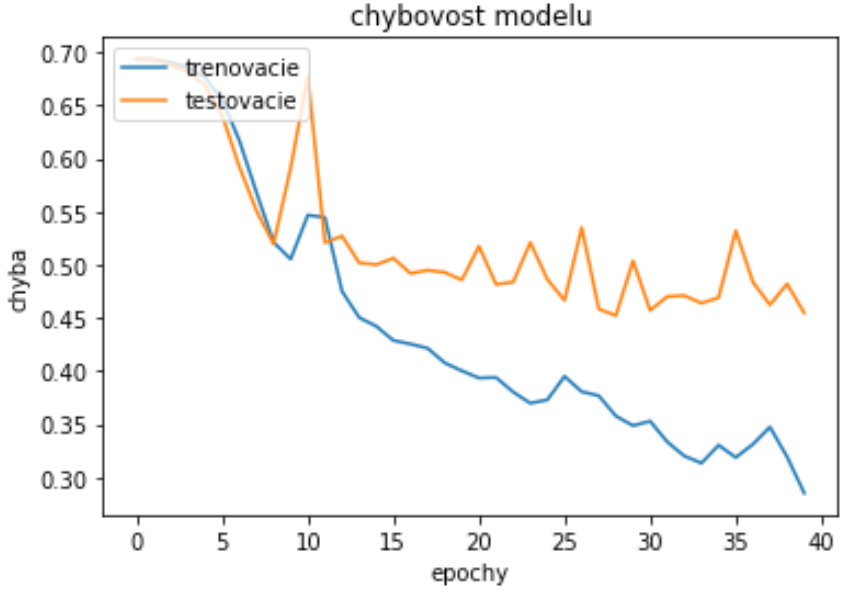
Trénovanie - epochy

- Počet iterácií po datasete

```
Train on 721 samples, validate on 241 samples
Epoch 1/100
40s - loss: 0.6931 - acc: 0.5118 - val_loss: 0.6935 - val_acc: 0.4440
Epoch 2/100
3s - loss: 0.6929 - acc: 0.5007 - val_loss: 0.6931 - val_acc: 0.4564
Epoch 3/100
4s - loss: 0.6904 - acc: 0.5506 - val_loss: 0.6879 - val_acc: 0.7137
Epoch 4/100
3s - loss: 0.6858 - acc: 0.6782 - val_loss: 0.6821 - val_acc: 0.6349
Epoch 5/100
3s - loss: 0.6762 - acc: 0.6047 - val_loss: 0.6685 - val_acc: 0.6722
Epoch 35/100
3s - loss: 0.3304 - acc: 0.8613 - val_loss: 0.4690 - val_acc: 0.8174
Epoch 36/100
4s - loss: 0.3187 - acc: 0.8682 - val_loss: 0.5321 - val_acc: 0.8050
Epoch 37/100
3s - loss: 0.3311 - acc: 0.8710 - val_loss: 0.4837 - val_acc: 0.7718
Epoch 38/100
3s - loss: 0.3473 - acc: 0.8599 - val_loss: 0.4623 - val_acc: 0.8216
Epoch 39/100
3s - loss: 0.3194 - acc: 0.8571 - val_loss: 0.4820 - val_acc: 0.8091
Epoch 40/100
4s - loss: 0.2853 - acc: 0.8863 - val_loss: 0.4546 - val_acc: 0.8008
```

Trénovanie - vizualizácia





Zdroje

- https://keras.io/api/models/sequential/#sequential-class
- https://keras.io/api/callbacks/early_stopping/
- https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
- https://stackoverflow.com/questions/50284898/keras-earlystopping-which-min-delta-and-patience-to-use
- https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/
- https://keras.io/api/layers/core_layers/dense/
- https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_gui/py_image_display/py_image_display.html
- https://www.geeksforgeeks.org/python-os-listdir-method/
- https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_colorspaces/py_colorspaces.html