

**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY**

ZADANIE 2 – KONVOLUČNÁ NEURÓNOVÁ SIEŤ

**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY**

ZADANIE 2 – KONVOLUČNÁ NEURÓNOVÁ SIETĚ

Študijný program:	Aplikovaná mechatronika a elektromobilita
Predmet:	Strojové videnie a výpočtová inteligencia
Prednášajúci:	doc. Ing. Oto Haffner, PhD. doc. Ing. Erik Kučera, PhD.
Cvičiaci:	Ing. Michal Hlavatý Ing. Martin Pajpach

Zadanie

Vytvorte a overte konvolučnú neurónovú sieť (CNN), ktorá bude schopná rozpoznávať najmenej 4 triedy. Po natrénovaní bude sieť schopná rozpoznávať všetky naučené triedy. Úlohy v skrátenej verzii:

1. Pripravte 1 vlastný dataset o plastovej fľaši. Počet obrázkov má byť medzi 50-100.
2. Natrénujte konvolučnú neurónovú sieť pre aspoň 4 triedy. Vypíšte indexy tried a predikcie pre testovacie obrázky.
3. Otestujte samostatne 1 obrázok z každej triedy.
4. Vypíšte názov predikovanej triedy.
5. Zobrazte testovací obrázok a do obrázku vypíšte predikciu.
6. Pre sieť vyneste skóre presnosti, konfúznú maticu (confusion matrix) a klasifikačný report. Stručne zhodnoťte jej výsledky.
7. Natrénuvanú konvolučnú neurónovú sieť uložte, načítajte a znovu skompilujte.
8. Vytvorte experimenty s 3 rôznymi verziami siete. Nastavte rôzne parametre ako počet filtrov, veľkosť filtrov, počet neurónov v Dense vrstve, batch_size pri tréovaní, počet epoch, atď..., prípadne pridajte ďalšie konvolučné alebo Dense vrstvy. Pre siete uložte model a natrénuvané váhy (pozor, treba vždy zmeniť názov) a výpis súhrnu siete (použitím `.summary()`).
9. Neurónové siete z experimentov vzájomne porovnajte (konfúzne matice, accuracy, čas tréovania).

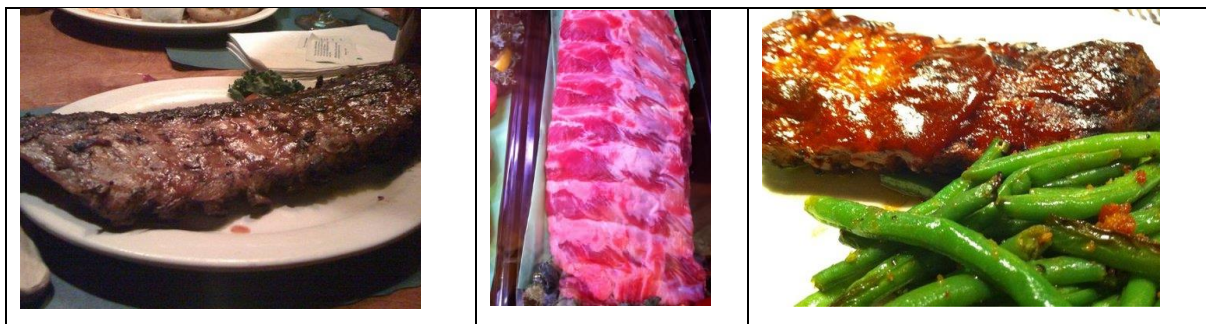
1. Dataset

Na riešenie zadanie som použil 6 rôznych kategórii s témou jedlo. Kategórie sú nasledovné:

- Jablkový koláč (apple pie)



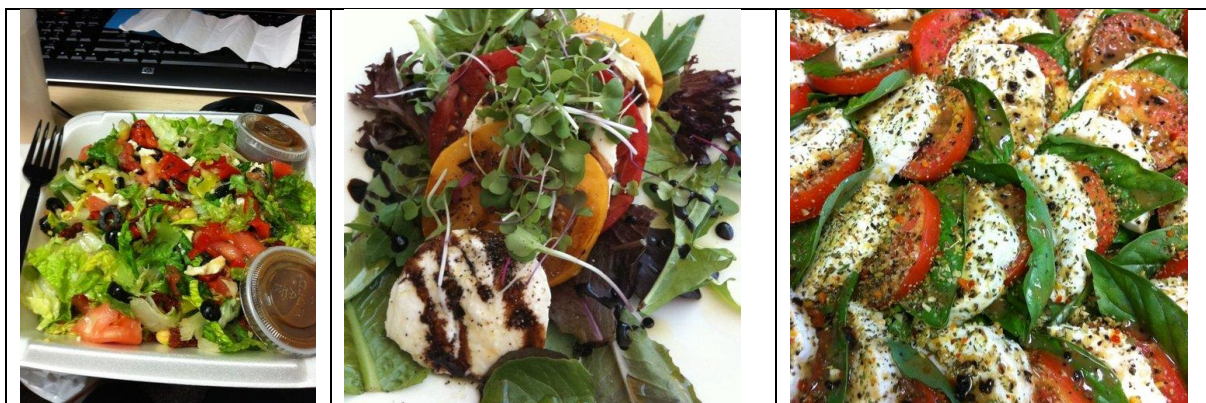
- Rebierka (baby back ribs)



- Cézar šalát



- Šalát Caprese (caprese salad)



- Muffiny (cup cakes)



- Donutky (donuts)



V každej jednej kategórie som mal k dispozícii 1 000 obrázkov. Rozdelil som to na 900 obrázkov na trénovacie dáta, a 100 obrázkov na testovanie. Dataset môžete stiahnuť z nasledujúceho linku: <https://www.kaggle.com/kmader/food41>

V zadaní som ešte mal za úlohu spraviť 50-100 fotiek o plastovej fľaše. Tento dataset môžete stiahnuť z nasledujúceho linku: <https://drive.google.com/drive/folders/17hfQRibdK-FNBrZXC6Akyrwdy6Gila69?usp=sharing>

2. Trénovanie CNN

Prvú vec čo som spravil, som načítal trénovacie dáta. Vy normalizoval som obrázky (vynásobil som ich s 1/255) a zároveň zmenšil som ich na rozlíšenie 64x64, aby som zrýchlil proces trénovania. V nasledujúcom kóde vidíme tento proces:

```
train_datagen = ImageDataGenerator(rescale = 1./255,
                                    shear_range = 0.0,
                                    zoom_range = 0.0,
                                    horizontal_flip = True)

training_set = train_datagen.flow_from_directory('data/train',
                                                target_size=(64, 64),
                                                batch_size=30,
                                                class_mode='categorical',
                                                shuffle=False)
```

Pri načítaní testovacích obrázkov, kód vyzerá identicky, len *batch_size* som zmenšil, keďže mam len 100 obrázkov na testovanie.

```
test_datagen = ImageDataGenerator(rescale = 1./255)
test_set = test_datagen.flow_from_directory('data/test',
                                            target_size = (64, 64),
                                            batch_size = 10,
                                            class_mode='categorical',
                                            shuffle=False)
```

Pri oboch načítaní dát, používam class metódu *categorical*, lebo ideme rozoznávať viaceré kategórie obrázkov. Po spustení kódu vidíme nasledujúci output najprv pre trénovacie dáta a potom pre testujúce:

```
Found 5400 images belonging to 6 classes.
```

```
Found 600 images belonging to 6 classes.
```

Po načítaní sme si vypísali indexy jednotlivých tried ktoré sme načítali. Pre trénovacie dáta to bolo nasledovné:

```
training_set.class_indices
[9] ✓ 0.4s
... {'apple_pie': 0,
     'baby_back_ribs': 1,
     'caesar_salad': 2,
     'caprese_salad': 3,
     'cup_cakes': 4,
     'donuts': 5}
```

Indexy tried pre testovacie dáta:

```
test_set.class_indices
[10] ✓ 0.2s
... {'apple_pie': 0,
     'baby_back_ribs': 1,
     'caesar_salad': 2,
     'caprese_salad': 3,
     'cup_cakes': 4,
     'donuts': 5}
```

Ako vidíme, indexy sa zhodujú. Ďalším krokom bolo nastaviť štruktúru konvolučnej neurónovej siete. Kód je ten istý, ktorý sme použili na cvičení č. 10, s rozdielom, že som prepísal počet neurónov v Full Connection vrstve na 256, a zmenil som aktivačnú funkciu pre vystupujúcu vrstvu na *softmax*.

```
cnn = tf.keras.models.Sequential()

# Step 1 - Convolution
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3,
                                activation='relu', input_shape=[64, 64, 3]))

# Step 2 - Pooling
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))

# Adding a second convolutional layer
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))

# Step 3 - Flattening
cnn.add(tf.keras.layers.Flatten())

# Step 4 - Full Connection
cnn.add(tf.keras.layers.Dense(units=256, activation='relu'))

# Step 5 - Output Layer
cnn.add(tf.keras.layers.Dense(units=6, activation='softmax'))
```

Ďalej som skompiloval sieť, s nasledujúcim kódom:

```
cnn.compile(optimizer='adam', loss='categorical_crossentropy',
            metrics=['accuracy'])
```

Pri trénovaní sieti, som pridal callback na *EarlyStopping*. Dvodom bolo, že sa mi sieť pretrénovala pri 80-ich epokoch. EarlyStopping som nastavil s *patience* na 5, kde mi zastavil trénovanie pri epoku 21. Takto mám sieť minimálne pretrénovanú. Nižšie vidíme output z *.summary()* pre moju sieť.

```
.. Model: "sequential_1"

Layer (type)                 Output Shape              Param #
=====
conv2d_2 (Conv2D)            (None, 62, 62, 32)       896
max_pooling2d_2 (MaxPooling2 (None, 31, 31, 32)       0
conv2d_3 (Conv2D)            (None, 29, 29, 32)       9248
max_pooling2d_3 (MaxPooling2 (None, 14, 14, 32)       0
flatten_1 (Flatten)          (None, 6272)             0
dense_2 (Dense)              (None, 256)              1605888
dense_3 (Dense)              (None, 6)                1542
=====
Total params: 1,617,574
Trainable params: 1,617,574
Non-trainable params: 0
```

Výpisy z *.predict(test_set)*

```
predictions = cnn.predict(test_set)
predictions


✓ 1.9s

array([[ 2.3343037e-07,  9.2803203e-03,  2.0004412e-04,  7.9660976e-01,
         1.2872557e-01,  6.5184101e-02],
       [ 4.8861648e-03,  5.0537737e-06,  3.7269987e-04,  4.1929793e-06,
         9.9392468e-01,  8.0718729e-04],
       [ 3.9520450e-03,  6.7490618e-07,  1.9475289e-08,  2.3670367e-05,
         6.3416939e-03,  9.8968190e-01],
       ...,
       [ 8.8768208e-01,  1.1121281e-02,  4.5630150e-02,  5.5892635e-03,
         1.4521903e-03,  4.8524968e-02],
       [ 2.7904876e-02,  7.5369951e-04,  5.1929455e-08,  4.1109845e-03,
         9.3910551e-01,  2.8124930e-02],
       [ 2.8795827e-02,  1.7392727e-08,  2.5677180e-02,  6.1896110e-01,
         2.8478336e-01,  4.1782495e-02]], dtype=float32)
```



3. Testovanie obrázku z každej triedy

Pre tento krok som si vybral náhodou jeden obrázok z každej jednej triedy a spustil som kód na predikciu. Výsledky sú v tabuľkách nižšie, v ľavej bunke sa nachádza použitý obrázok na predikciu a v pravej bunke je výpis z predikcie.


Apple pie

	<pre>array([[9.7801709e-01, 1.3846256e-04, 3.5462126e-03, 6.1594699e-05, 2.1281395e-04, 1.8023886e-02]], dtype=float32)</pre>
---	---

Baby back ribs

	<pre>array([[1.0021721e-09, 1.0000000e+00, 1.5550188e-20, 3.2922628e-13, 1.9017503e-11, 4.9779936e-10]], dtype=float32)</pre>
--	---

Caesar salad

	<pre>array([[4.2113247e-03, 3.0555257e-05, 6.4808488e-01, 3.2099888e-01, 2.3323912e-03, 2.4341935e-02]], dtype=float32)</pre>
---	---

Caprese salad



```
array([[3.5948336e-02,  
1.0294523e-05, 1.1134449e-01,  
6.2112594e-01, 2.7832431e-03,  
2.2878766e-01]], dtype=float32)
```

Cup cakes



```
array([[9.37740907e-08,  
5.95498361e-16, 1.53276503e-09,  
1.37295375e-08, 9.99999642e-01,  
2.76628583e-07]],  
dtype=float32)
```

Donuts



```
array([[2.6492989e-02,  
3.4645951e-04, 1.3184798e-01,  
1.2669930e-02, 5.2162565e-02,  
7.7648008e-01]], dtype=float32)
```

4. Výpis názvu predikovanej triedy

Na výpis som použil slovník kde som ako kľúč napísal číslo ktoré korešponduje indexu triedy. Ako *value* som napísal meno triedy. Na obrázku nižšie vidíme, že je vypísaný výsledok, „Na obrázku je donut“ keďže som naposledy testoval predikciu pre donutku.

```
▷ ~
slovník = {0: "apple pie", 1: "baby back ribs", 2: "caesar salad", 3: "caprese
salad", 4: "cup cake", 5: "donut"}
predikcia = np.argmax(result, axis=1).astype(int)

print('Na obrázku je', slovník[predikcia[0]])

[30] ✓ 0.3s Pyth
... Na obrázku je donut
```

5. Obrázok s výpisom predikcie v obrázku

Pre túto úlohu som použil testovací obrázok apple_pie.jpg. Použitý kód na výpis textu predikcie do obrázku:

```
import cv2
im = cv2.imread(img_name, 1)
font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(im, slovník[predikcia[0]], (0, 200),
            font, 2, (0, 255, 0), 2, cv2.LINE_AA)
cv2.imwrite('result-text.jpg', im)
```

Výsledný obrázok:



6. Skóre presnosti, konfúzna matica, klasifikačný report

Presnosť siete ktorú som vytvoril:

```
#vypis presnosti neuronovej siete
accuracy_score(test_set.classes, predictions)
```

✓ 0.5s

0.5216666666666666

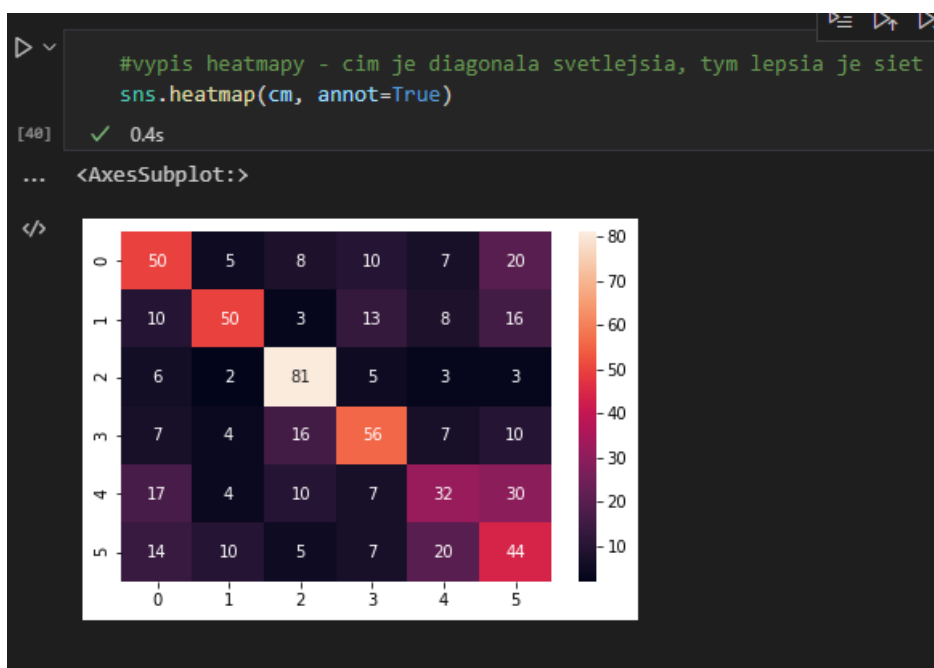
Konfúzna matica:

```
#vypis confusion matrixu
cm = confusion_matrix(test_set.classes, predictions)
cm
```

[37] ✓ 0.4s

```
... array([[50,  5,  8, 10,  7, 20],
          [10, 50,  3, 13,  8, 16],
          [ 6,  2, 81,  5,  3,  3],
          [ 7,  4, 16, 56,  7, 10],
          [17,  4, 10,  7, 32, 30],
          [14, 10,  5,  7, 20, 44]], dtype=int64)
```

Heatmapa:



Pri konfúznej matici a heatmapy treba vedieť, že čím viac máme presnejšiu sieť, tým viac čísiel budeme mať na diagonále. V našom prípade vidíme na obrázkoch, že máme čísla trochu rozhádzané do rohov matice, ale väčšina sa nachádza na diagonále.

Klasifikačný report:

```
print(classification_report(test_set.classes, predictions))
```

[41] ✓ 0.7s

...	precision	recall	f1-score	support
0	0.48	0.50	0.49	100
1	0.67	0.50	0.57	100
2	0.66	0.81	0.73	100
3	0.57	0.56	0.57	100
4	0.42	0.32	0.36	100
5	0.36	0.44	0.39	100
accuracy			0.52	600
macro avg	0.53	0.52	0.52	600
weighted avg	0.53	0.52	0.52	600

7. Uloženie, načítanie a skompilovanie siete

Neurónovú sieť som uložil do JSON súboru pod menom *cnn.json* následne som ju zase načítal spolu s váhami a vypísal *.summary()* z načítanej siete.

```
#ulozenie ns
model_json = cnn.to_json()
with open('cnn.json', 'w') as json_file:
    json_file.write(model_json)

#ulozenie vahy
network_saved = save_model(cnn, 'weights.hdf5')
```

[2] ✓ 0.2s

Načítanie NS:

```
#nacitanie ns
with open('cnn.json', 'r') as json_file:
    json_saved_model = json_file.read()
json_saved_model

[48] ✓ 0.2s Python

... '{"class_name": "Sequential", "config": {"name": "sequential_1", "layers": [{"class_name": "InputLayer", "config": {"batch_input_shape": [null, 64, 64, 3], "dtype": "float32", "sparse": false, "ragged": false, "name": "conv2d_2_input"}}, {"class_name": "Conv2D", "config": {"name": "conv2d_2", "trainable": true, "batch_input_shape": [null, 64, 64, 3], "dtype": "float32", "filters": 32, "kernel_size": [3, 3], "strides": [1, 1], "padding": "valid", "data_format": "channels_last", "dilation_rate": [1, 1], "groups": 1, "activation": "relu", "use_bias": true, "kernel_initializer": {"class_name": "GlorotUniform", "config": {"seed": null}}, "bias_initializer": {"class_name": "Zeros", "config": {}}, "kernel_regularizer": null, "bias_regularizer": null, "activity_regularizer": null, "kernel_constraint": null, "bias_constraint": null}}, {"class_name": "MaxPooling2D", "config": {"name": "max_pooling2d_2", "trainable": true, "dtype": "float32", "pool_size": [2, 2], "padding": "valid", "strides": [2, 2], "data_format": "channels_last"}}, {"class_name": "Conv2D", "config": {"name": "conv2d_3", "trainable": true, "dtype": "float32", "filters": 32, "kernel_size": [3, 3], "strides": [1, 1], "padding": "valid", "data_format": "channels_last", "dilation_rate": [1, 1], "groups": 1, "activation": "relu", "use_bias": true, "kernel_initializer": {"class_name": "GlorotUniform", "config": {"seed": null}}, "bias_initializer": {"class_name": "Zeros", "config": {}}, "kernel_regularizer": null, "bias_regularizer": null, "activity_regularizer": null, "kernel_constraint": null, "bias_constraint": null}}, {"class_name": "MaxPooling2D", "config": {"name": "max_pooling2d_3", "trainable": true, "dtype": "float32", "pool_size": [2, 2], "padding": "valid", "strides": [2, 2], "data_format": "channels_last"}}, {"class_name": "Flatten", "config": {"name": "flatten_1", "trainable": true, "dtype": "float32", "data_format": "channels_last"}}, {"class_name": "Dense", "config": {"name": "dense_2", "trainable": true, "dtype": "float32", "units": 256, "activation": "relu", "use_bias": true, "kernel_initializer": {"class_name": "GlorotUniform", "config": {"seed": null}}, "bias_initializer": {"class_name": "Zeros", "config": {}}, "kernel_regularizer": null, "bias_regularizer": null, "activity_regularizer": null, "kernel_constraint": null, "bias_constraint": null}}, {"class_name": "Dense", "config": {"name": "dense_3", "trainable": true, "dtype": "float32", "units": 6, "activation": "softmax", "use_bias": true, "kernel_initializer": {"class_name": "GlorotUniform", "config": {"seed": null}}, "bias_initializer": {"class_name": "Zeros", "config": {}}, "kernel_regularizer": null, "bias_regularizer": null, "activity_regularizer": null, "kernel_constraint": null, "bias_constraint": null}}]}, "keras_version": "2.4.0", "backend": "tensorflow"}
```


Výpis `.summary()` pre načítanú sieť:

```
network_loaded = tf.keras.models.model_from_json(json_saved_model)
network_loaded.load_weights('weights.hdf5')
network_loaded.compile(loss='categorical_crossentropy',
                        optimizer='Adam', metrics=['accuracy'])
network_loaded.summary()
```

✓ 0.8s

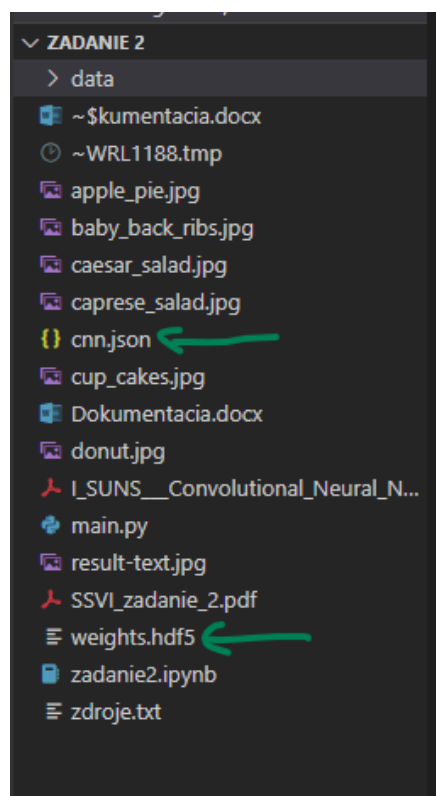
Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d_2 (MaxPooling2D)	(None, 31, 31, 32)	0
conv2d_3 (Conv2D)	(None, 29, 29, 32)	9248
max_pooling2d_3 (MaxPooling2D)	(None, 14, 14, 32)	0
flatten_1 (Flatten)	(None, 6272)	0
dense_2 (Dense)	(None, 256)	1605888
dense_3 (Dense)	(None, 6)	1542

=====
Total params: 1,617,574
Trainable params: 1,617,574
Non-trainable params: 0
=====

Keď si porovnáme `.summary()` načítanej neurónovej siete a `.summary()` vytvorenej neurónovej siete, zbadáme že sa zhodujú. To znamená že pri načítaní sa žiadne údaje nemenili a máme tu istú neurónovú sieť.

Obrázok o uloženej neurónovej sieti a váh:



8. Experimenty

Pri experimentoch som zmenil počet filtrov, jadrov a neurónov vo full control vrstve.

Experiment 1

Zmenené nastavenia:

- Prvá konvolučná vrstva: filtre $32 \rightarrow 26$ a $\text{kernel_size } 3 \rightarrow 4$
- Full conn. Vrstva: units $256 \rightarrow 126$

Výpis `.summary()`

```
> cnn.summary()
117] ✓ 0.3s
... Model: "sequential_7"
```

Layer (type)	Output Shape	Param #
conv2d_14 (Conv2D)	(None, 61, 61, 26)	1274
max_pooling2d_14 (MaxPooling)	(None, 30, 30, 26)	0
conv2d_15 (Conv2D)	(None, 28, 28, 32)	7520
max_pooling2d_15 (MaxPooling)	(None, 14, 14, 32)	0
flatten_7 (Flatten)	(None, 6272)	0
dense_15 (Dense)	(None, 126)	790398
dense_16 (Dense)	(None, 6)	762

```
=====  
Total params: 799,954  
Trainable params: 799,954  
Non-trainable params: 0  
=====
```

Experiment 2

Zmenené nastavenia:

- Prvá konvolučná vrstva: filtre $32 \rightarrow 14$, kernel_size $3 \rightarrow 6$
- Druhá konvolučná vrstva: kernel_size $3 \rightarrow 4$
- Full connection dense vrstva: units $256 \rightarrow 147$

Výpis .summary()

```
Model: "sequential_8"
```

Layer (type)	Output Shape	Param #
conv2d_16 (Conv2D)	(None, 59, 59, 14)	1526
max_pooling2d_16 (MaxPooling)	(None, 29, 29, 14)	0
conv2d_17 (Conv2D)	(None, 26, 26, 32)	7200
max_pooling2d_17 (MaxPooling)	(None, 13, 13, 32)	0
flatten_8 (Flatten)	(None, 5408)	0
dense_17 (Dense)	(None, 147)	795123
dense_18 (Dense)	(None, 6)	888

```
Total params: 804,737  
Trainable params: 804,737  
Non-trainable params: 0
```

Experiment 3

Zmenené nastavenia:

- Prva konvolucna vrstva: filtre $32 \rightarrow 29$, kernel_size $3 \rightarrow 2$
- Druha konvolucna vrstva: kernel_size $3 \rightarrow 4$
- Full connection vrstva dense: units $256 \rightarrow 126$
- Pridana extra dense vrstva: units 256

```
> cnn.summary()  
[85] ✓ 0.4s  
... Model: "sequential_4"  
  
Layer (type)                Output Shape                Param #  
=====
```

conv2d_8 (Conv2D)	(None, 63, 63, 29)	377
-------------------	--------------------	-----

```
-----  
max_pooling2d_8 (MaxPooling2 (None, 31, 31, 29)    0  
-----  
conv2d_9 (Conv2D)            (None, 28, 28, 32)         14880  
-----  
max_pooling2d_9 (MaxPooling2 (None, 14, 14, 32)    0  
-----  
flatten_4 (Flatten)          (None, 6272)               0  
-----  
dense_8 (Dense)              (None, 126)               790398  
-----  
dense_9 (Dense)              (None, 256)               32512  
-----  
dense_10 (Dense)             (None, 6)                 1542  
=====
```

```
Total params: 839,709  
Trainable params: 839,709  
Non-trainable params: 0  
-----
```

9. Vyhodnotenie experimentov

Experiment 1

Sieť sa trénovala 6 minút 7 sekúnd. Trénovanie sa zastavilo pri 18. epochu.

```
[14] ✓ 6m 7.7s  
.. Output exceeds the size limit. Open the full output data in a text editor  
Epoch 1/80  
180/180 [=====] - 20s 112ms/step - loss: 1.8063 -
```

```
> ✓ 0.5s  
#vypis presnosti neuronovej siete  
accuracy_score(test_set.classes, predictions)  
[126] ✓ 0.5s  
... 0.555
```

```
> ✓ 0.8s  
#vypis confusion matrixu  
cm = confusion_matrix(test_set.classes, predictions)  
cm  
[7] ✓ 0.8s  
array([[60,  9,  4,  9,  5, 13],  
       [ 8, 55,  3, 12,  5, 17],  
       [15,  1, 69, 13,  0,  2],  
       [ 6,  6, 12, 69,  4,  3],  
       [15,  7,  7,  9, 28, 34],  
       [20,  9,  3,  7,  9, 52]], dtype=int64)
```


Experiment 2

Sieť sa trénovala 9 minút a 20 sekúnd. Trénovanie sa zastavilo 27. epochu.

```
133] ✓ 8m 22.6s

... Output exceeds the size limit. Open the full output data in a text editor
Epoch 1/80
180/180 [=====] - 20s 110ms/step - loss: 1.8122 - a
Epoch 2/80
```

```
✓
#vypis confusion matrixu
cm = confusion_matrix(test_set.classes, predictions)
cm
] ✓ 0.4s

array([[42, 15,  8,  6,  9, 20],
       [ 2, 70,  3,  7,  5, 13],
       [ 8,  3, 70,  7,  4,  8],
       [ 3, 17, 14, 52,  4, 10],
       [12,  9,  9,  9, 30, 31],
       [13, 21,  9,  8,  7, 42]], dtype=int64)
```

```
#vypis presnosti neuronovej siete
accuracy_score(test_set.classes, predictions)
[143] ✓ 0.3s

... 0.51
```

Experiment 3

Siet sa trenovala 7 minut 54 sekund. Trenovanie sa zastavilo pri 22. epochu.

```
early_stop = EarlyStopping(monitor='val_accuracy', mode='max', patience=5)
cnn.fit(x=training_set, validation_data=test_set, epochs=80, callbacks=[early_stop])
```

✓ 7m 54.6s

Output exceeds the size limit. Open the full output data in a text editor

Epoch 1/80
180/180 [=====] - 21s 118ms/step - loss: 1.8043 - acc: 0.5366666666666666
Epoch 2/80

```
#vypis confusion matrixu
cm = confusion_matrix(test_set.classes, predictions)
cm
```

✓ 0.4s

```
array([[38,  7, 14,  5, 15, 21],
       [ 2, 66,  0,  5,  4, 23],
       [ 7,  2, 72,  8,  3,  8],
       [ 2,  8,  9, 54, 11, 16],
       [10,  4,  8, 10, 39, 29],
       [10, 13,  7,  6, 11, 53]], dtype=int64)
```

```
#vypis presnosti neuronovej siete
accuracy_score(test_set.classes, predictions)
```

✓ 0.3s

```
0.5366666666666666
```

Porovnanie výsledkov experimentov

Experiment 1 získalo najlepšie výsledky s presnosťou 0.555 a s najlepšiu konfúznou maticou. Ostatne experimenty boli trochu horšie. Zdá sa že zlata cesta je niekde medzi zníženými filtrami a zväčšenie počtu jadrov, pričom možno ešte jedna dense vrstva pred vystupujúcou vrstvou môže pomôcť. Najhoršie výsledky mal Experiment 2 kde som dramaticky znížil filtre a zväčšil počet jadier a znížil počet neurónov v dense full connection vrstve.