

SUNS - Zadanie 3

12. decembra 2022

1 Povinne úlohy

Použil som knižnicu **tensorflow keras** na načítávanie a spracovanie obrázkov pre toto zadanie. Vytvoril som trenovacie a validačné množiny a pre nich som aplikoval augmentáciu (obrázok 1). Testovaciu množinu som načítal bez augmentácií.

```
IMG_SIZE = (64, 64)
BATCH_SIZE = 200

image_generator = ImageDataGenerator(
    rescale = 1./255.,
    shear_range = 0.2,
    zoom_range = 0.2,
    horizontal_flip = True
)

train_generator = image_generator.flow_from_directory(
    directory = path_train,
    target_size = IMG_SIZE,
    batch_size = BATCH_SIZE,
)

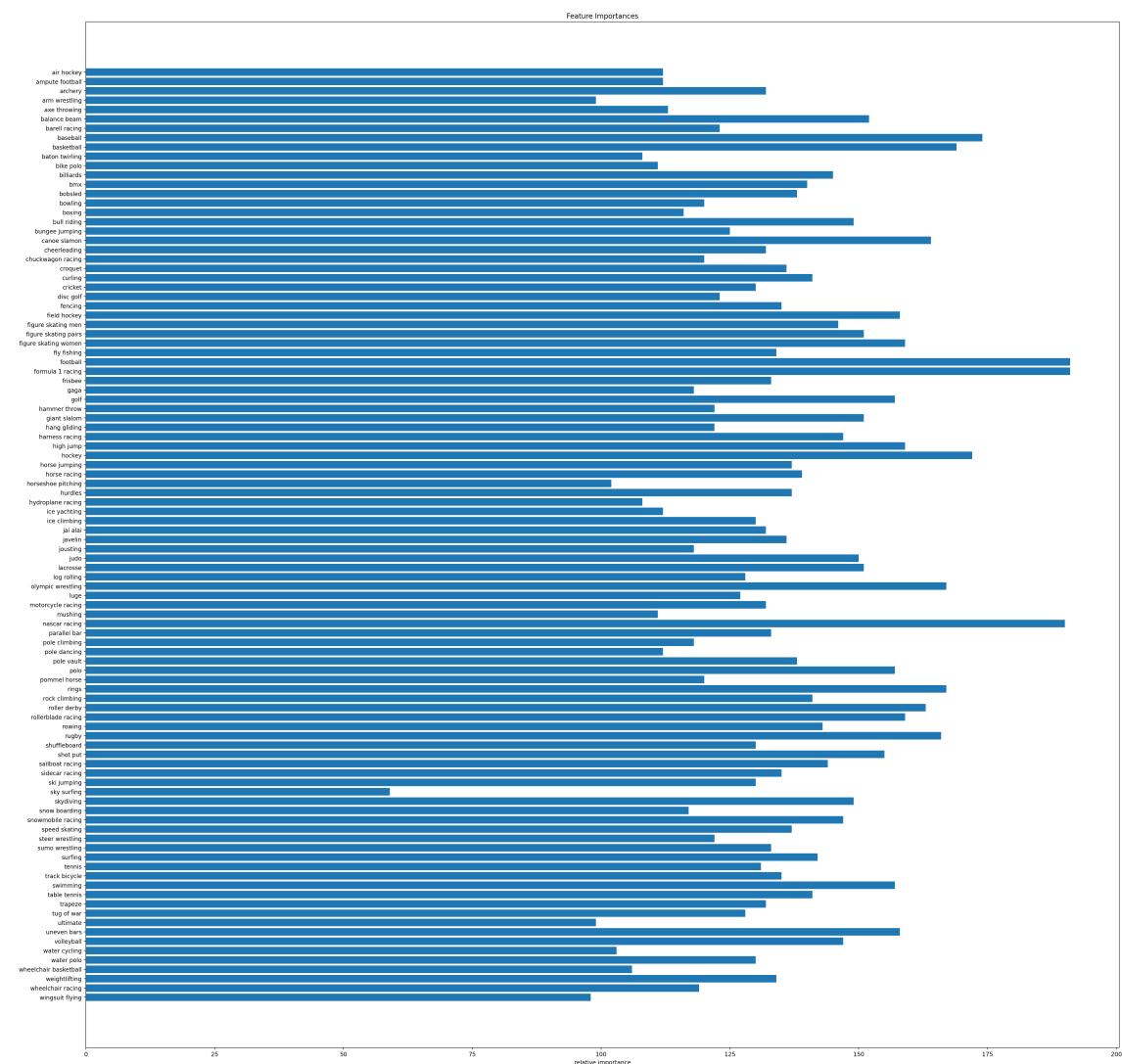
valid_generator = image_generator.flow_from_directory(
    directory = path_valid,
    target_size = IMG_SIZE,
    batch_size = BATCH_SIZE,
)

test_generator = tf.keras.preprocessing.image_dataset_from_directory(
    directory = path_test,
    image_size = IMG_SIZE,
    batch_size = BATCH_SIZE,
    shuffle = False,
)
```

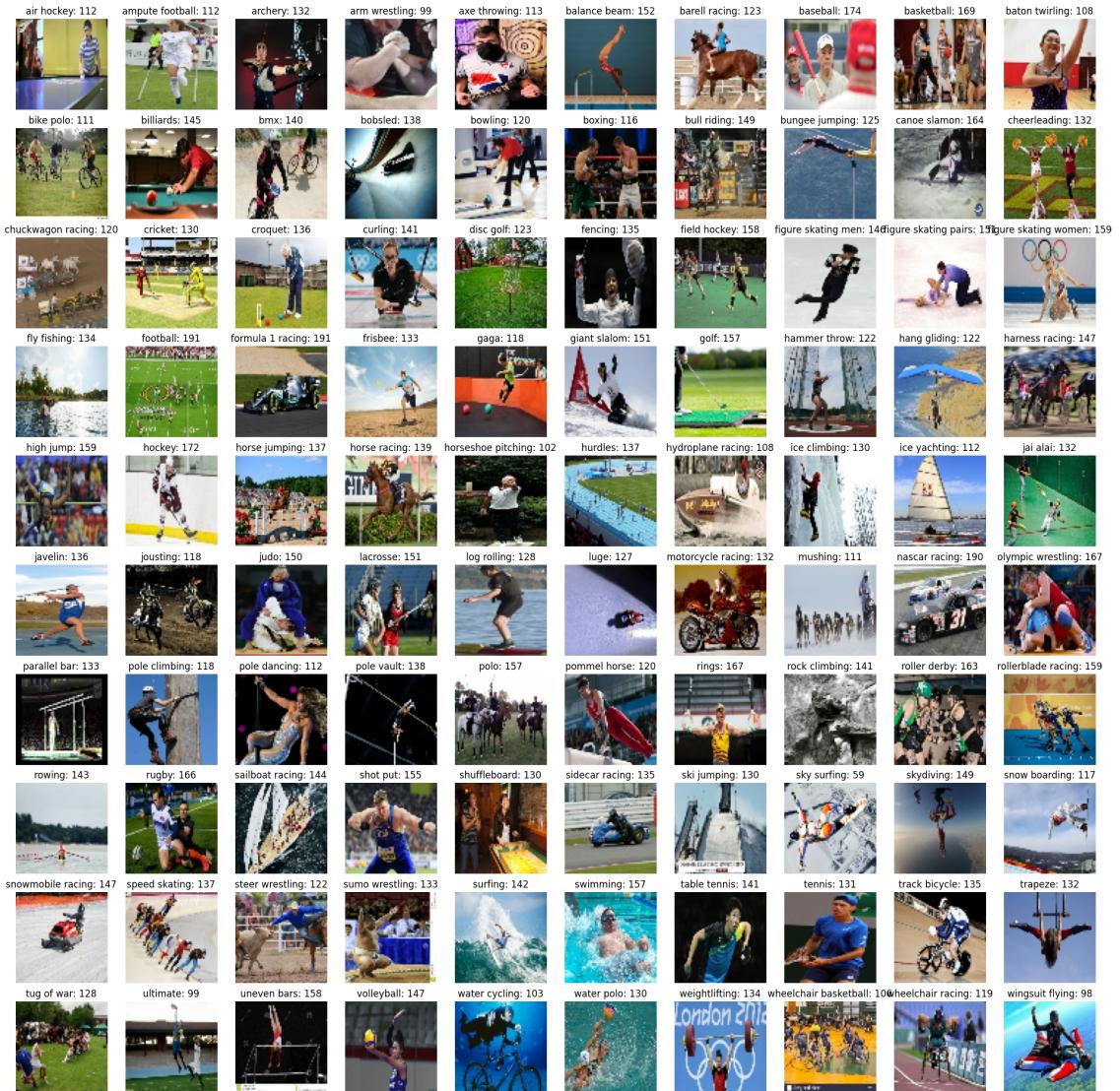
Obrázok 1: Načítávanie obrázkov

1.1 EDA

Prvý bod som splnil tak, že som vyplabil dva grafy. Jedna zobrazuje početnosť obrázkov z každej triede na bar plote, a na druhom je napísaná číselna hodnota kolko je obrázkov v každej triede, spolu s jedným reprezentativným obrázkom tej triedy (obrázky 2 a 3).

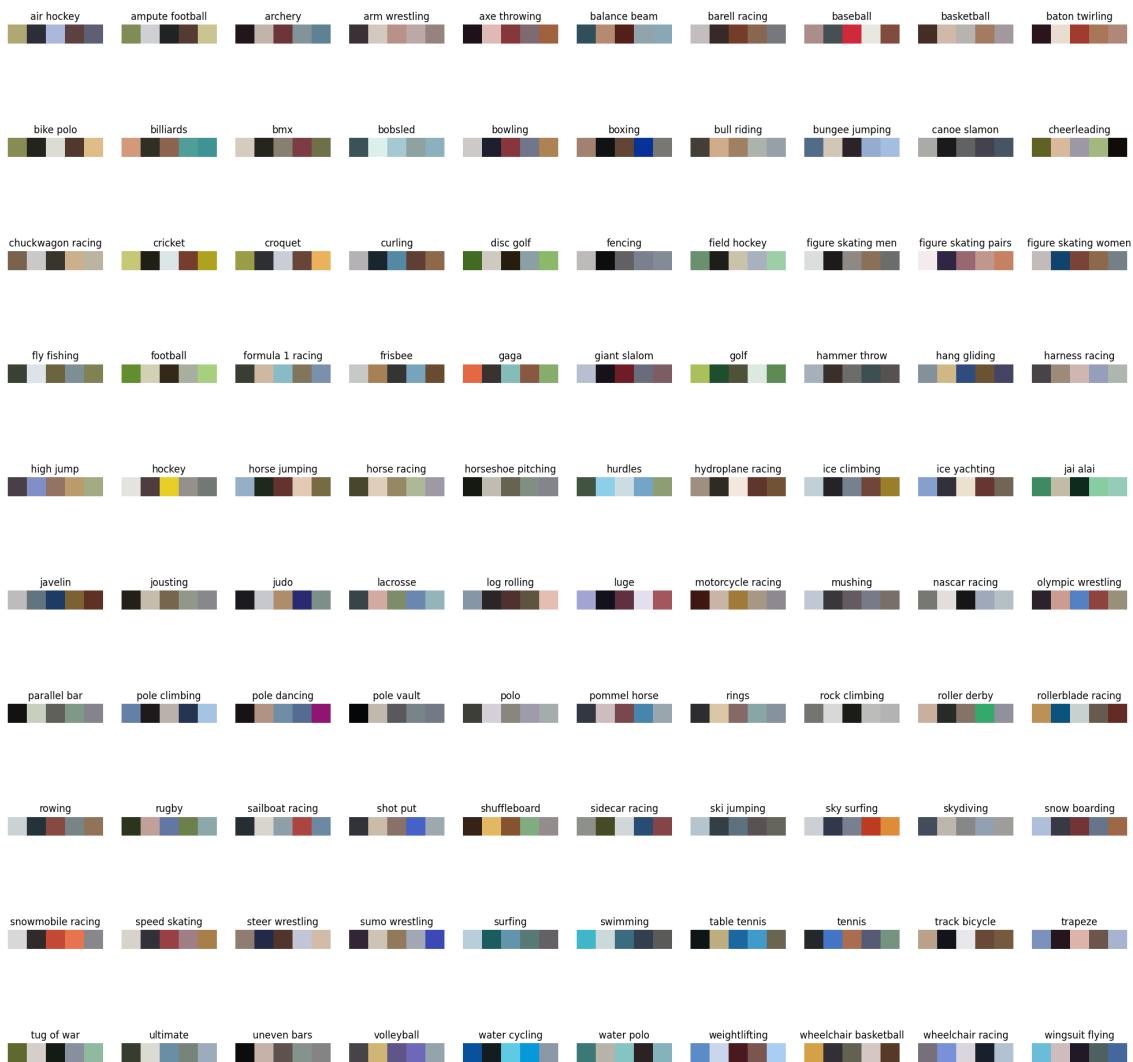


Obrázok 2: Bar plot početnosti obrázkov každej triedy



Obrázok 3: Obrázky reprezentujúc svoje triedy spolu s početnosťou obrázkov každej triedy

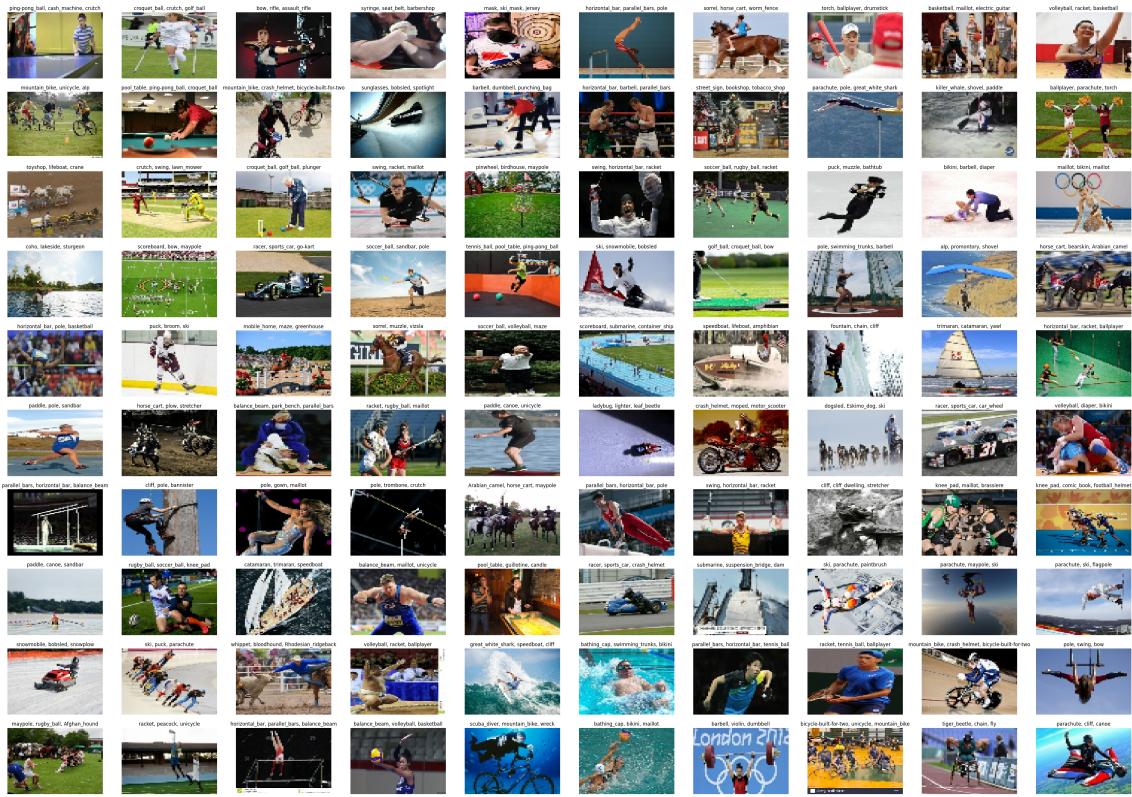
Ďalej som analyzoval triedy podľa farebnosti pixelov. Extraktoval som 5 farbov, ktoré sa najviac vyskytovali v obrázkoch, ktoré reprezentujú svoje triedy:



Obrázok 4: 5 najčastejších farbov každej triedy

Zaujímavé je pozrieť ako zimné športy majú najdominantnejšiu farbu bielu. Taktiež športy, ktoré sa vyskytujú vo vode a na nebe majú najdominantnejšiu farbu modrú atď.

Ako finálny krok v EDA-e som použil už natrénovanú siet **ResNet50**, ktorý skúsil prepovedať čo sa nachádza na obrázku reprezentanta každej klase. Top 3 výsledky sme vyplotili:



Obrázok 5: Predpovede ResNet50 siete

1.2 CNN

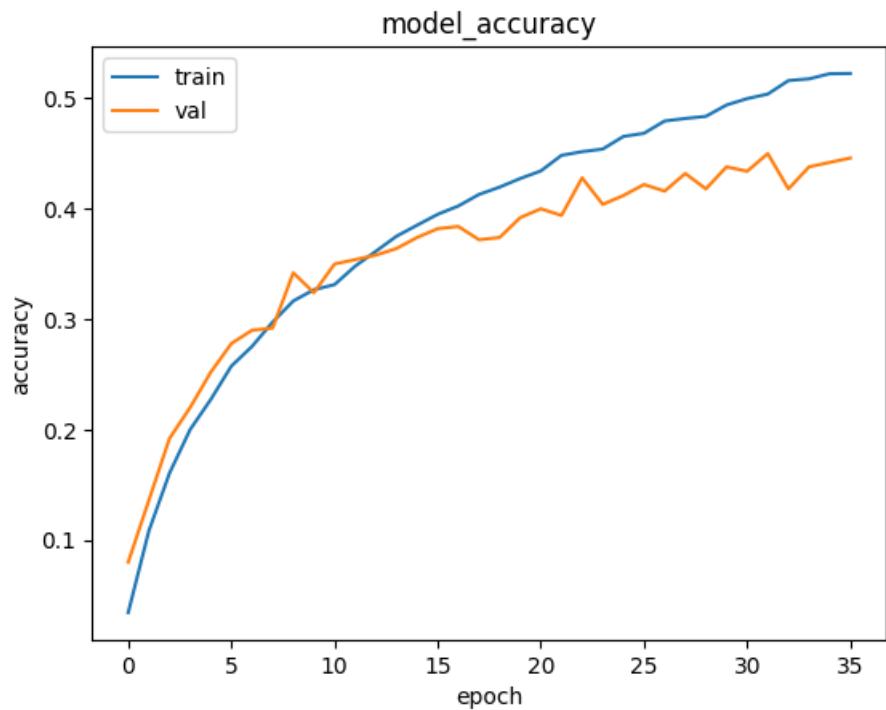
Ďalej som trenoval konvolučnú neurónovú siet. Použil som Sequential model z tensorflow.keras a nastavil som následovné parametre (obrázok 6).

```
model = Sequential([
    Conv2D(filters = 32, kernel_size = (3, 3), activation = 'relu', padding = 'same', input_shape = (64, 64, 3)),
    MaxPool2D(pool_size = (2, 2), strides = 2),
    Conv2D(filters = 64, kernel_size = (3, 3), activation = 'relu', padding = 'same'),
    MaxPool2D(pool_size = (2, 2), strides = 2),
    Flatten(),
    Dense(units = 100, activation = 'softmax'),
])
model.summary()
```

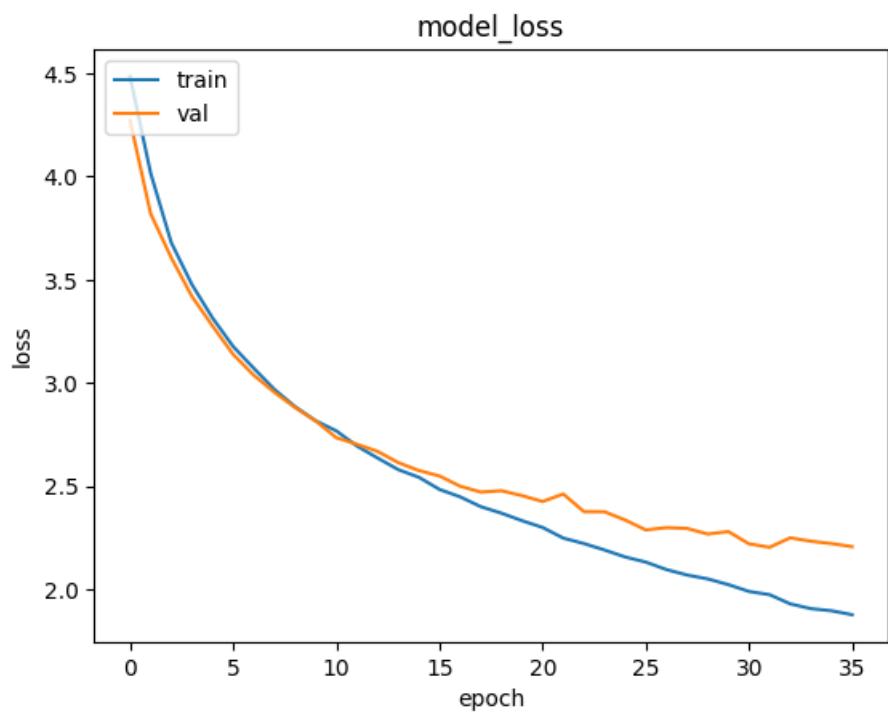
Obrázok 6: Nastavenie sekvenčného modelu

Model som trénoval v 50 epóch, ale som používal EarlyStopping s trpezlivosťou 4. Po 35 epóche sa zastavila trénovať siet (tým pádom sme sa vyhnuli pretrénovaniu). Úspešnosť

bola **46%** na trénovacej množine (čo je vynikajúce). Model sme potom sejvovali a sme vyplotili históriu trenovania sieti na trénovacej a validačnej množine (obrázky 7 a 8).



Obrázok 7: Trénovanie CNN



Obrázok 8: Loss CNN

Vhodne som otestoval funkčnosť modelu na testovacej množine a získal som rovnaký accuracy ako na validačnej množine **46%** (obrázok 9).

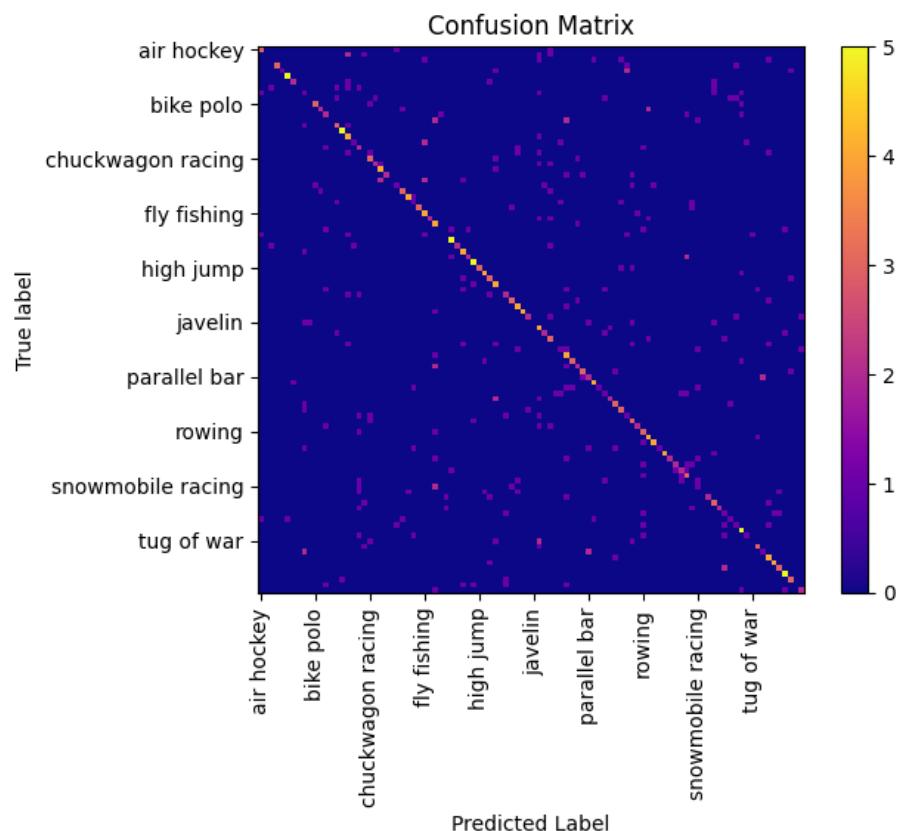
```

loss_v, accuracy_v = model.evaluate(valid_generator, verbose = 1)
loss, accuracy = model.evaluate(test_generator, verbose = 1)
print("Validation: accuracy = %f ; loss_v = %f" % (accuracy_v, loss_v))
print("Test: accuracy = %f ; loss = %f" % (accuracy, loss))
✓ 7.9s
3/3 [=====] - 3s 894ms/step - loss: 2.1986 - accuracy: 0.4560
3/3 [=====] - 1s 239ms/step - loss: 2.1541 - accuracy: 0.4560
Validation: accuracy = 0.456000 ; loss_v = 2.198602
Test: accuracy = 0.456000 ; loss = 2.154142

```

Obrázok 9: Výsledky trénovania

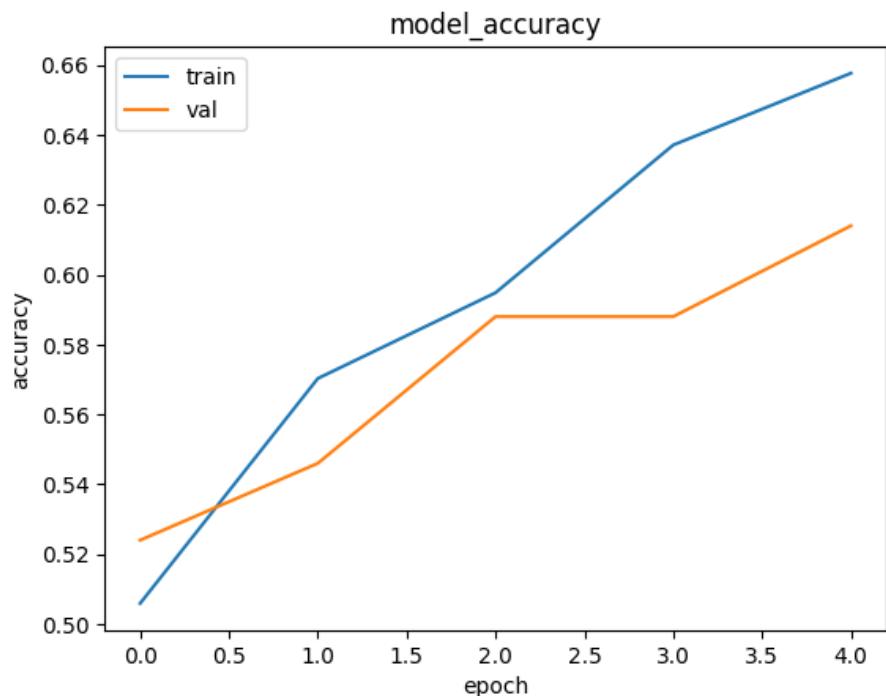
A na konci som vyplotil konfúznu maticu. Na obrázku 10, je možné vidieť diagonálu, čo znamená, že nám je siet úspešne natrénovaná.



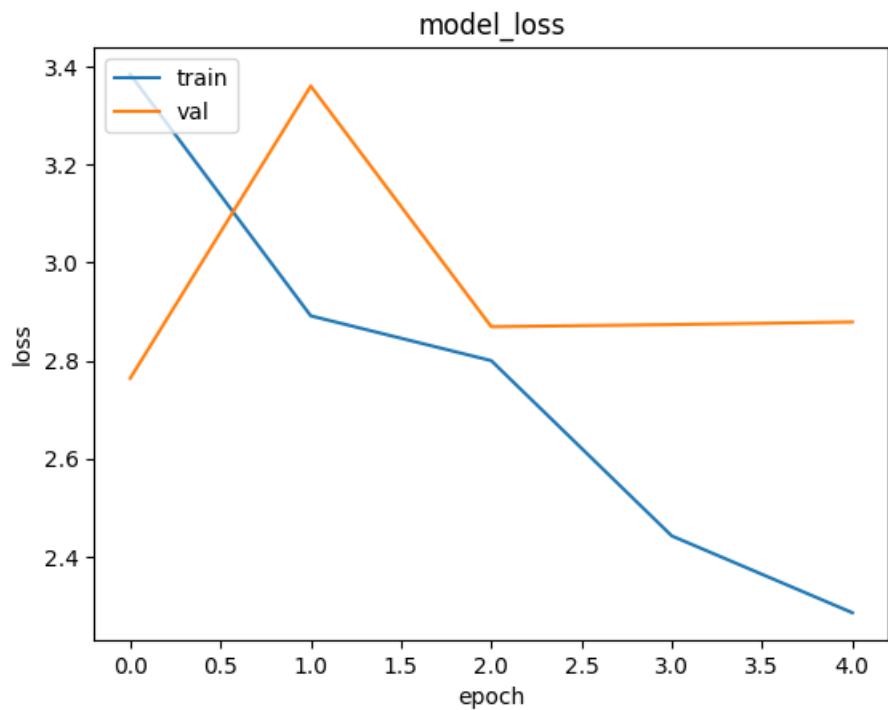
Obrázok 10: Confusion Matrix

1.3 Transfer learning

Rozhodol som sa istť cestou fine-tuning. Použil som už natrénovaný modél z imageNetu pre našu databázu. V mojom prípade, zvolil som si **MobileNet**. Vytvoril som nový modél, ktorý zdedil skoro všetky outputy MobileNet modélu, okrem posledných dvoch, kde som tie posledné dve pridal v kóde. Potom som zamrzol všetky layers okrem posledných 23. Tento modél som potom dal trénovať. Výsledky trénovalia sú následovné:



Obrázok 11: Trénovanie ImageNet



Obrázok 12: Loss ImageNet

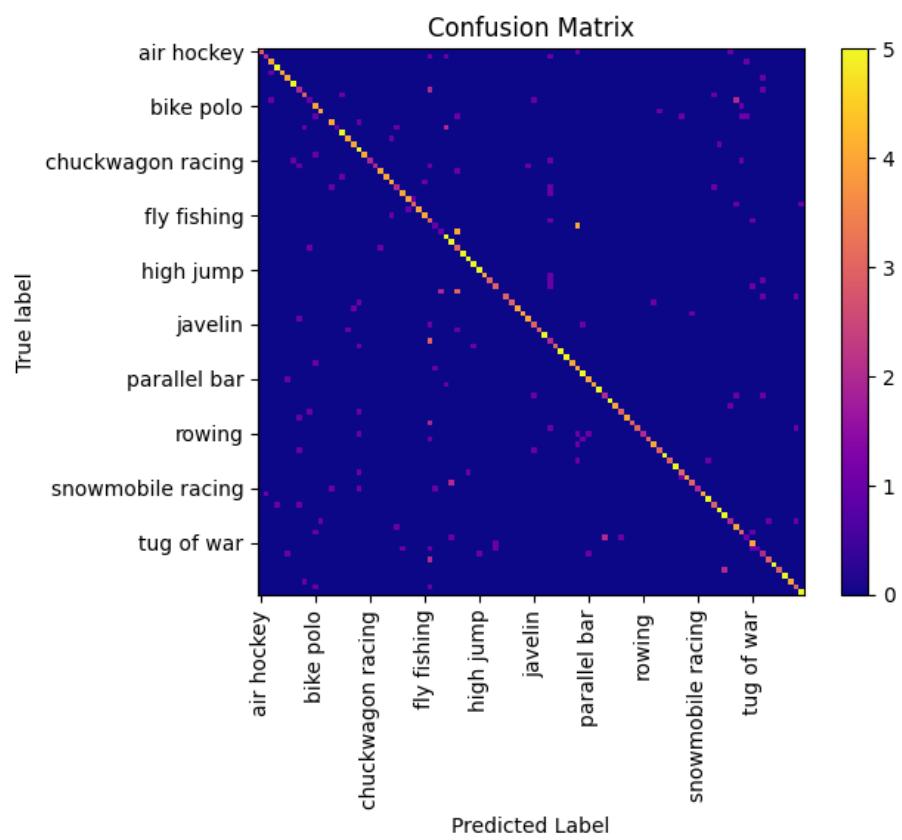
Výsledky sú lepšie ako minulé. Validačné dátá dostáli úspešnosť **63%** a testovacie dátá ač **70%** (obrázok 13).

```
# test_generator = test_generator.map(one_hot_label, num_parallel_calls = autotune)
# test_generator = test_generator.cache().prefetch(buffer_size = autotune)

predictions = model.predict(x = test_generator, verbose = 0)
loss_v, accuracy_v = model.evaluate(valid_generator, verbose = 1)
loss, accuracy = model.evaluate(test_generator, verbose = 1)
print("Validation: accuracy = %f ; loss_v = %f" % (accuracy_v, loss_v))
print("Test: accuracy = %f ; loss = %f" % (accuracy, loss))
✓ 34.8s
16/16 [=====] - 5s 309ms/step - loss: 2.6371 - accuracy: 0.6340
500/500 [=====] - 15s 29ms/step - loss: 2.0788 - accuracy: 0.7020
Validation: accuracy = 0.634000 ; loss_v = 2.637061
Test: accuracy = 0.702000 ; loss = 2.078831
```

Obrázok 13: ImageNet výsledky

Aj konfúzna matica tohto nového modélu krajšie vyzerá:



Obrázok 14: Confusion Matrix

2 Nepovinné úlohy

2.1 Vlastná databáza

Vytvoril som nový priečinok vo priečinku z3-data pod menom own-data. Tu som pridal 22 vlastných obrázkov, ktoré som stiahol z internetu. Testovacia úspešnosť bola **27%**, čo je v pohode.

```
ownTest = './data-z3/own-tests/'

test_generator_2 = tf.keras.preprocessing.image_dataset_from_directory(
    directory = ownTest,
    image_size = IMG_SIZE,
    batch_size = BATCH_SIZE,
    shuffle = False
)
```

Obrázok 15: Načítanie vlastnej databázy

```
predictions = np.array([])
labels = np.array([])
for x, y in test_generator_2:
    predictions = np.concatenate([predictions, np.argmax(model.predict(x), axis = -1)])
    labels = np.concatenate([labels, np.argmax(y.numpy(), axis=-1)])]

loss, accuracy = model.evaluate(test_generator_2, verbose = 1)

cm = tf.math.confusion_matrix(labels=labels, predictions=predictions).numpy()
✓ 1.1s

Found 22 files belonging to 100 classes.
1/1 [=====] - 0s 206ms/step
1/1 [=====] - 0s 178ms/step - loss: 4.3363 - accuracy: 0.2727
```

Obrázok 16: Výsledky testovania