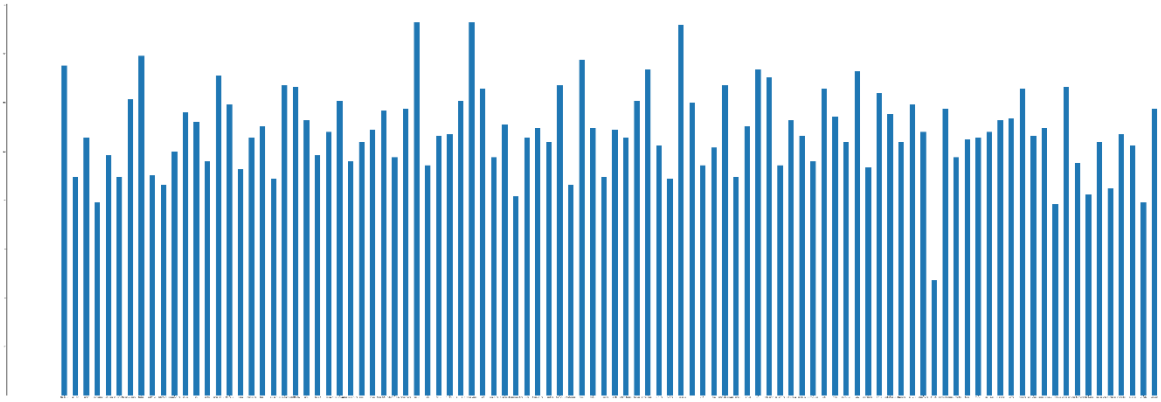
Zadanie 3

Marek Dráb 97757

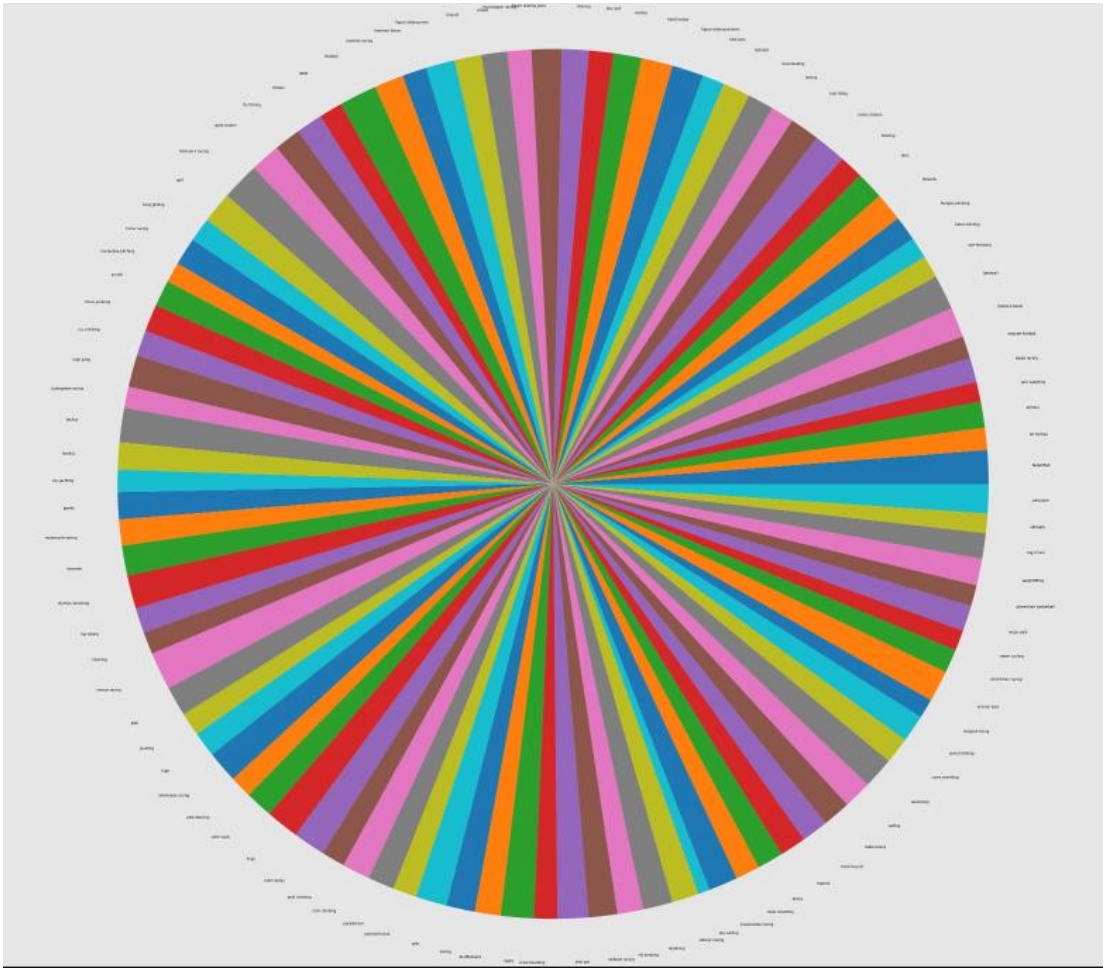
EDA

Početnosti v triedach



Obrázok 1 Graf početnosti v triedach

Na zobrazenom grafe je možné vidieť, že 3 triedy majú vyššiu početnosť než je priemer zatiaľ čo výrazne pod priemerom je len jedna trieda – sky surfing.



Obrázok 2 Koláčový graf početnosti v triedach

Vybrané triedy

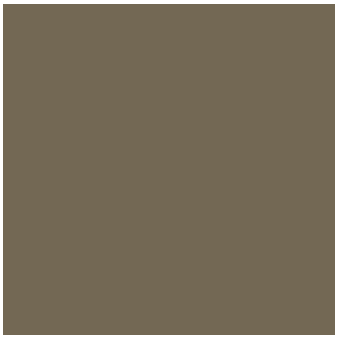
V rámci analýzy farebnosti pixelov boli vybrané

- priemerná farba získavaná pomocou funkcie average z numpy
- dominantné farby v danom obrázku pomocou zhlukovania pomocou KMeans

Basketbal

Obrázok 3 Basketbal

Priemerná farba



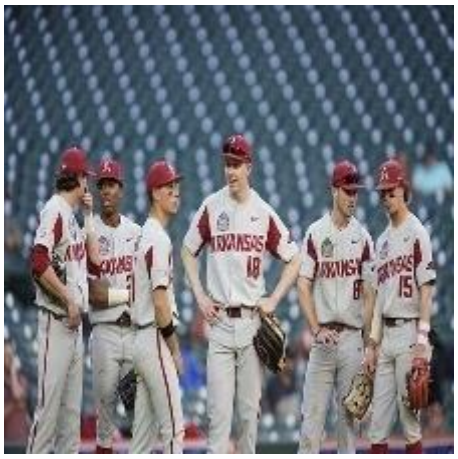
Obrázok 4 RGB(84, 104, 115)

Najviac vyskytované farby

Obrázok 5 Dominantné farby - basketbal

RGB(75, 87, 86)	7,5%
RGB(174, 181, 135)	9,31%
RGB(70, 45, 38)	11,21%
RGB(232, 238, 247)	21,12%
RGB(14, 14, 24)	50,86%

Basebal



Obrázok 6 Basebal

Priemerná farba

Obrázok 7 RGB(122, 114, 112)

Najviac vyskytované farby

Obrázok 8 Dominantné farby – basebal

RGB(53, 45, 50)	13,35%
RGB(203, 199, 199)	15,88%
RGB(112, 114, 125)	18,82%
RGB(159, 155, 157)	18,83%
RGB(66, 79, 93)	33,13%

Hod sekerou

Obrázok 9 Hod sekerou

Priemerná farba

Obrázok 10 RGB(111, 113, 114)

Najviac vyskytované farby

Obrázok 11 Dominantné farby - hod sekerou

RGB(163, 157, 155)	14,33%
RGB(28, 33, 29)	18,7%
RGB(114, 102, 89)	19,26%
RGB(227, 230, 232)	20,74%
RGB(61, 65, 67)	26,98%

BMX

Obrázok 12 BMX

Priemerná farba

Obrázok 13 RGB(165, 164, 154)

Najviac vyskytované farby

Obrázok 14 Dominantné farby – BMX

RGB(47, 44, 53)	13,45%
RGB(113, 116, 136)	14,38%
RGB(182, 167, 162)	15,15%
RGB(110, 145, 80)	16,09%
RGB(212, 227, 246)	40,94%

F1

Obrázok 15 F1

Priemerná farba

Obrázok 16 RGB(93, 125, 125)

Najviac vyskytované farby

Obrázok 17 Dominantné farby - F1

RGB(218, 226, 235)	7,83%
RGB(236, 159, 146)	11,53%
RGB(28, 30, 34)	11,59%
RGB(102, 108, 121)	17,35%
RGB(115, 129, 64)	51,7%

Hod oštepom

Obrázok 18 Hod oštepom

Priemerná farba

Obrázok 19 RGB(101, 101, 101)

Najviac vyskytované farby

Obrázok 20 Dominantné farby - hod oštepom

RGB(221, 221, 221)	10,45%
RGB(166, 166, 166)	14,72%
RGB(115, 115, 115)	19,7%
RGB(74, 74, 74)	26,81%
RGB(38, 38, 38)	28,31%

Trénovanie

Generátor dát

Po načítaní a vykreslení potrebných grafov boli dáta normalizované. Veľkosť bola zmenená na 32x32px pre jednoduchšie trénovanie. Taktiež bol vytvorený aj generátor pre trénovaciu, validačnú a testovaciu množinu. Validačné dáta používajú rovnaký generátor ale využívajú sa dáta určené na validovanie. Batch size bol nastavený na 512 pri všetkých 3 množinách. Pri trénovacej a validačnej bol použitý aj shuffle. V prvotnom návrhu boli skúšané aj možnosti ako horizontal a vertical flip, ale po problémoch s trénovaním siete bolo od nich upustené.

```
train_datagen = ImageDataGenerator(rescale = 1/255,
                                   #  rotation_range = 40,
                                   #  width_shift_range = 0.2,
                                   #  height_shift_range = 0.2,
                                   #  shear_range = 0.2,
                                   #  zoom_range = 0.2,
                                   #  horizontal_flip = True,
                                   #  vertical_flip =True
                                    )

test_datagen = ImageDataGenerator(rescale = 1/255)
```

```
train_gen = train_datagen.flow_from_dataframe(dataframe = train_df,
                                              x_col = 'Images',
                                              y_col ='Image_label',
                                              target_size = (32,32), batch_size = 512,
                                              class_mode = 'categorical',
                                              shuffle = True)
val_gen = train_datagen.flow_from_dataframe(valid_df,
                                            target_size=(32,32),
                                            x_col = 'Images',
                                            y_col ='Image_label',
                                            class_mode='categorical',
                                            batch_size=  512,
                                            shuffle=True)
test_gen = test_datagen.flow_from_dataframe(test_df,
                                            target_size = (32,32), x_col = 'Images', y_col ='Image_label',
                                             class_mode = 'categorical',
                                            batch_size = 512, shuffle = False)
```

Obrázok 21 ImageDataGenerator

Neurónová sieť

Sieť pozostáva z 3 vrstiev Conv2D, taktiež 3 vrstiev MaxPooling2D, ktoré boli použité vždy po Conv2D. Ďalej bol použitý Flatten a 2 vrstvy Dense. Ako optimalizátor bol použitý Adam s learning ratom 0,01. Použitý bol Early Stopping s limitom na 3 epochy. Trénovalo sa vždy na 16 epoch.

Zdroj:
https://www.tensorflow.org/tutorials/images/classification?fbclid=IwAR0MekBg88c56N9FUx0bHlgcFO2MPVtYxQ6c4xtObuVIdns06hErfiLGTHA

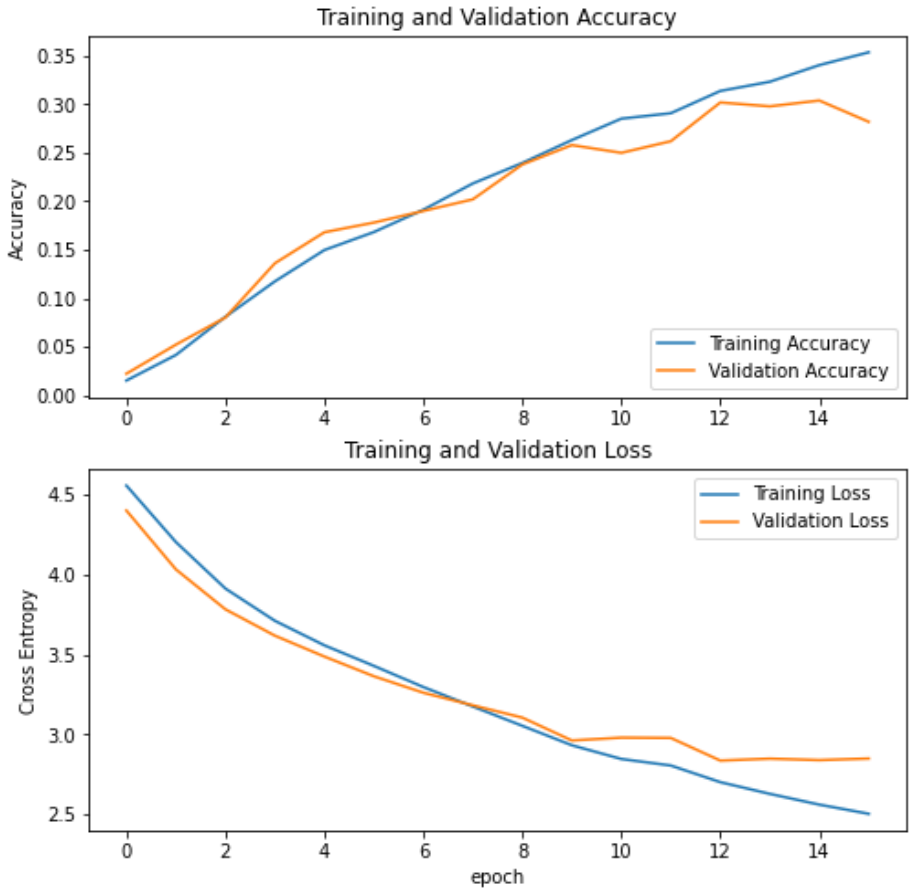
Priebeh a výsledky trénovania

Trénovanie modelu zo zdroja

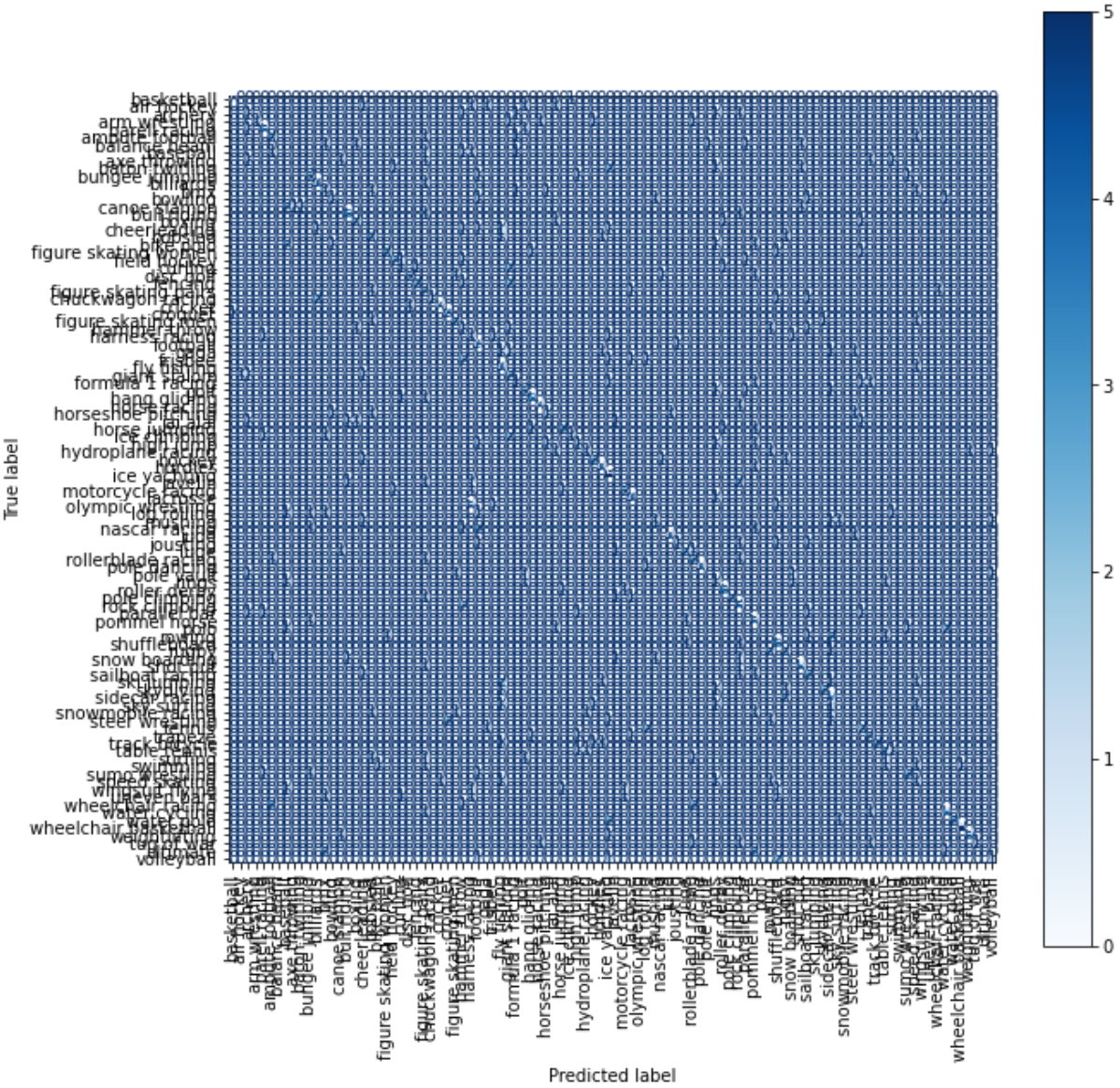
Na grafoch priebehu je možné vidieť plynulé zlepšovanie siete. Na konfúznej matici na testovacej množine máme viditeľnú priamku. Úspešnosť na testovacej množine bola približne 30%.

```
model = Sequential()
model.add(Conv2D(16,3, padding='same', input_shape=(32,32,3), activation='relu'))

model.add(MaxPooling2D())

model.add(Conv2D(32,3, padding='same', activation='relu'))

model.add(MaxPooling2D())

model.add(Conv2D(64,3, padding='same', activation='relu'))

model.add(MaxPooling2D())

#model.add(GlobalAveragePooling2D())
# model.add(Dense(128))
model.add(Flatten())

model.add(Dense(128, activation='relu'))
#model.add(Dropout(0.3))

model.add(Dense(100,activation='softmax'))

optimizer = keras.optimizers.Adam(learning_rate=0.01)

model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics = ['accuracy'], run_eagerly=True)
print(model.summary())
```

Obrázok 22 Model 1



Obrázok 23 Priebeh trénovania model 1



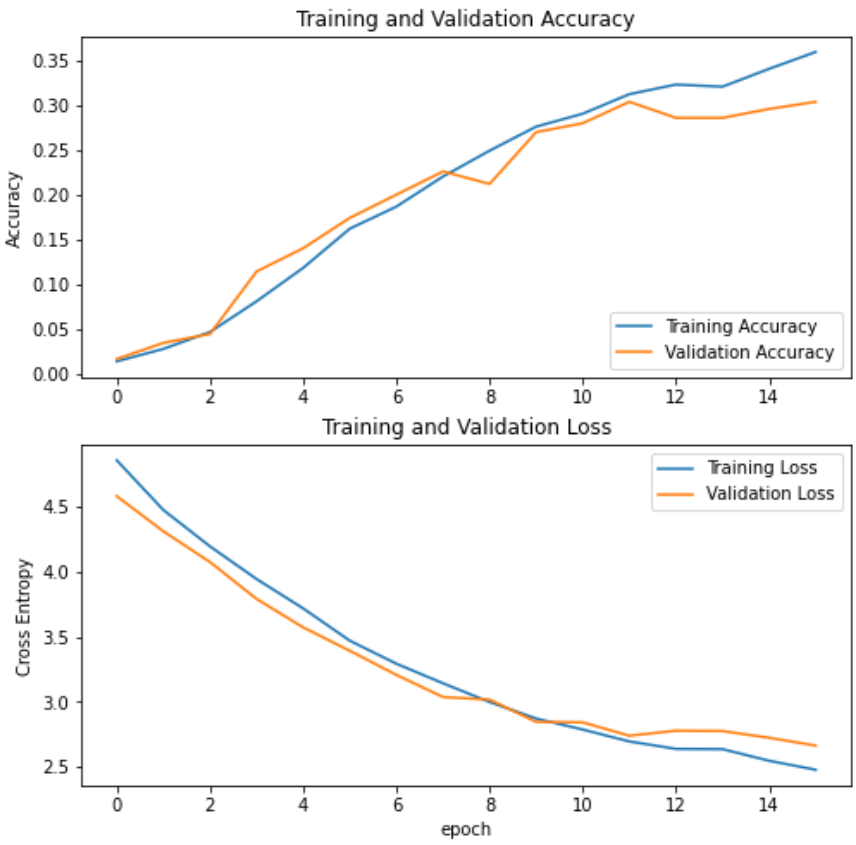
Obrázok 24 Konfúzna matica model 1

Zvýšenie počtu neurónov v Conv2D vrstvách

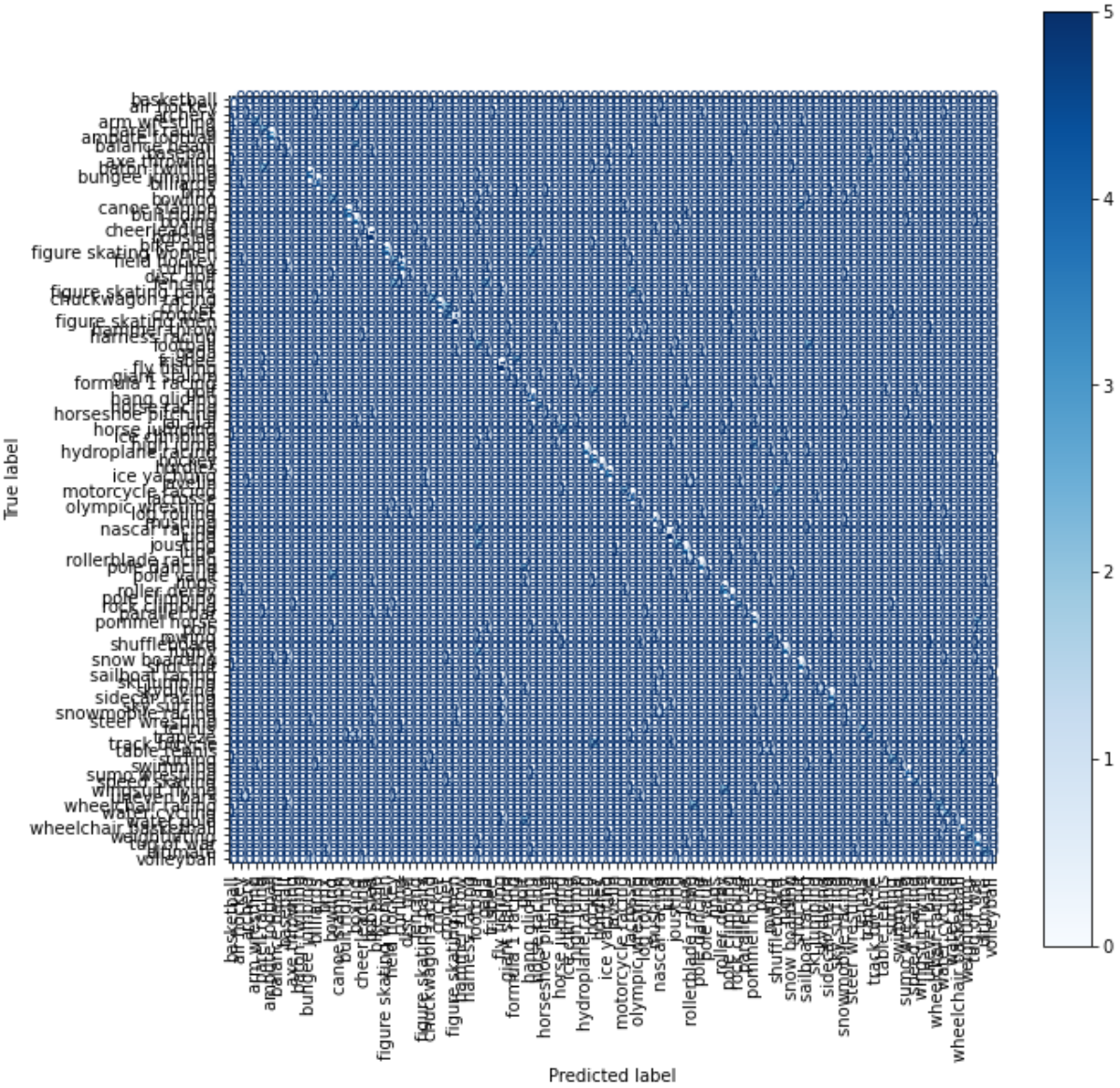
Po zvýšení počtu neurónov v Conv2D vrstvách môžeme pozorovať mierne zlepšenie, avšak nie je až také výrazné. Podobne aj na testovacej množine bola úspešnosť približne 30%.

```python
model = Sequential()
model.add(Conv2D(32,3, padding='same', input_shape=(32,32,3), activation='relu'))

model.add(MaxPooling2D())

model.add(Conv2D(64,3, padding='same', activation='relu'))

model.add(MaxPooling2D())

model.add(Conv2D(128,3, padding='same', activation='relu'))

model.add(MaxPooling2D())

#model.add(GlobalAveragePooling2D())
# model.add(Dense(128))
model.add(Flatten())

model.add(Dense(128, activation='relu'))
#model.add(Dropout(0.3))

model.add(Dense(100,activation='softmax'))

optimizer = keras.optimizers.Adam(learning_rate=0.01)

model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics = ['accuracy'], run_eagerly=True)
print(model.summary())
```

Obrázok 25 Model 2



Obrázok 26 Priebeh trénovania model 2



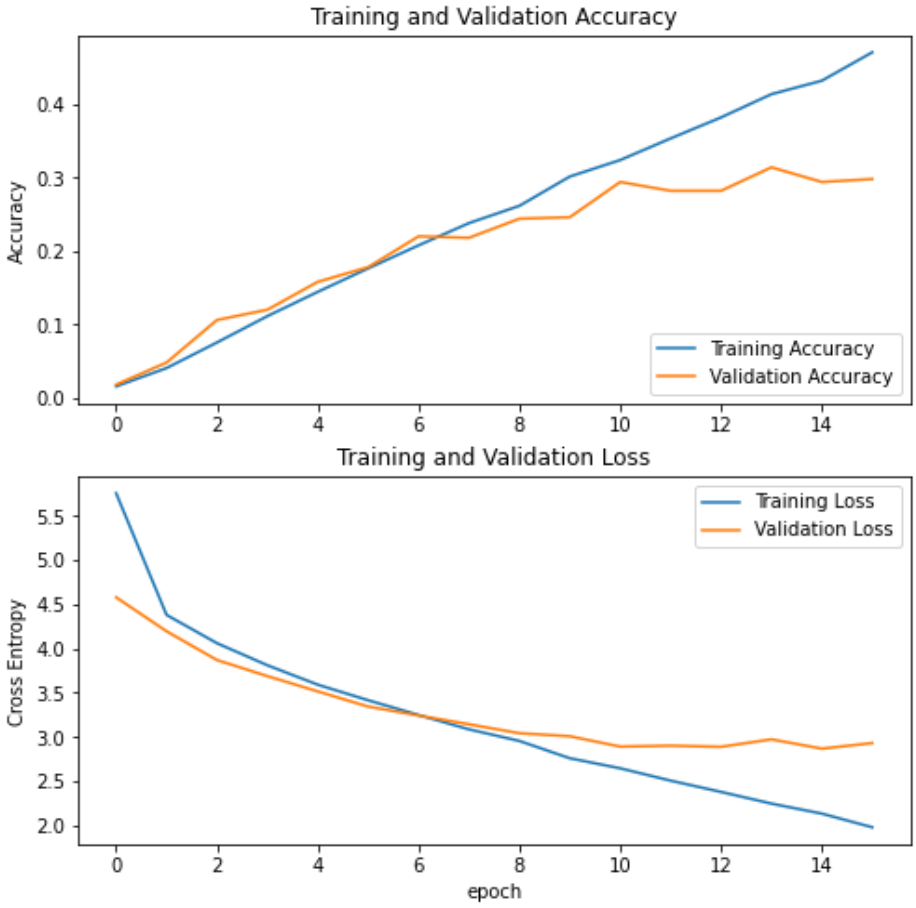
Obrázok 27 Konfúzna matica model 2

Zvýšenie počtu neurónov vo všetkých vrstvách

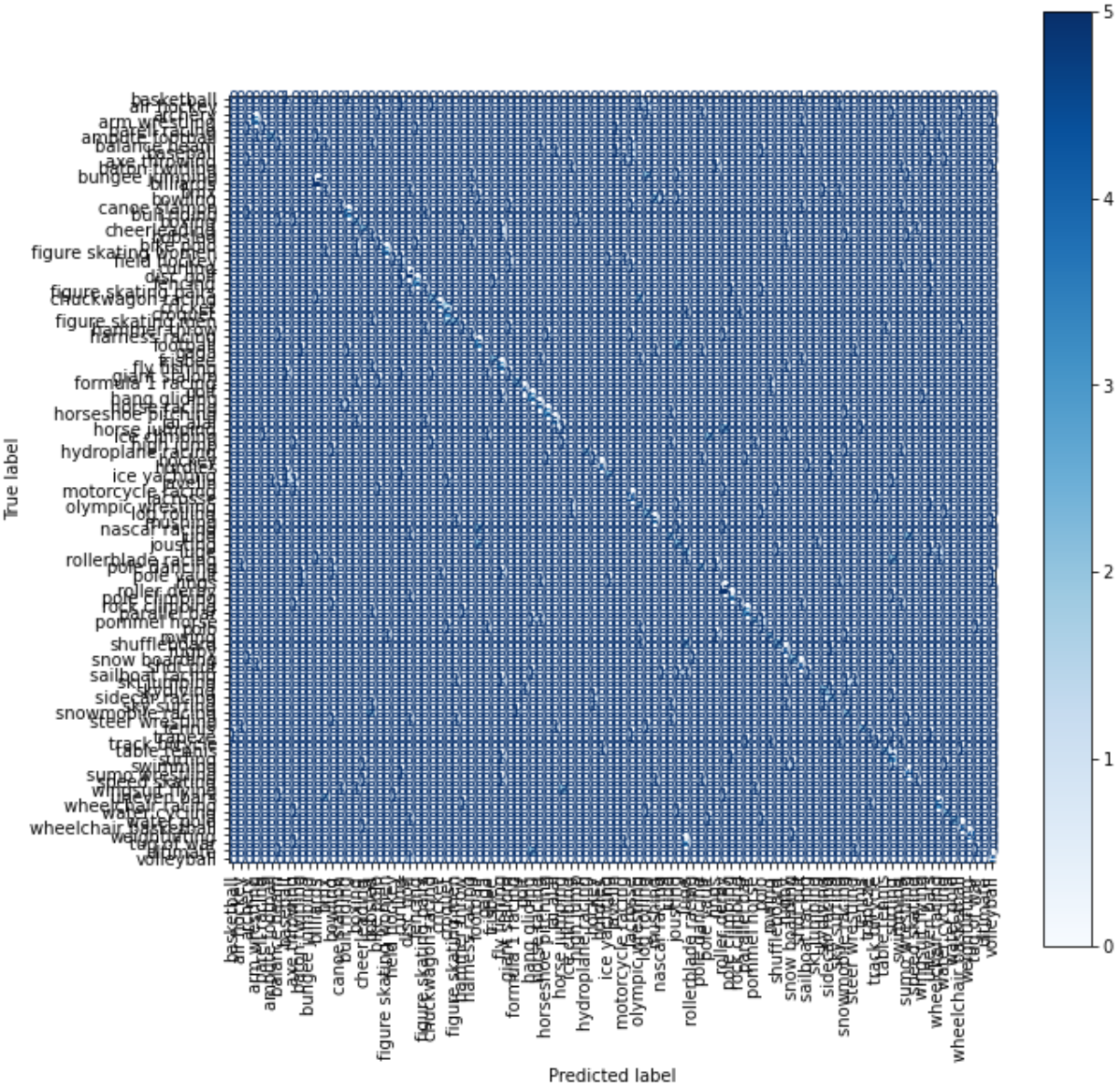
Oproti modelu 2 boli zvýšené počty neurónov dvojnásobne. Úspešnosť je výrazne vyššia na trénovacej množine. Na testovacej a validačnej sa stále drží na úrovni zhruba 30%. Na grafoch priebehu môžeme pozorovať pretrénovanie.

```
model = Sequential()
model.add(Conv2D(64,3, padding='same', input_shape=(32,32,3), activation='relu'))

model.add(MaxPooling2D())

model.add(Conv2D(128,3, padding='same', activation='relu'))

model.add(MaxPooling2D())

model.add(Conv2D(256,3, padding='same', activation='relu'))

model.add(MaxPooling2D())

#model.add(GlobalAveragePooling2D())
# model.add(Dense(128))
model.add(Flatten())

model.add(Dense(256, activation='relu'))
#model.add(Dropout(0.3))

model.add(Dense(100,activation='softmax'))

optimizer = keras.optimizers.Adam(learning_rate=0.01)

model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics = ['accuracy'], run_eagerly=True)
print(model.summary())
```

Obrázok 28 Model 3



Obrázok 29 Priebeh trénovania model 3



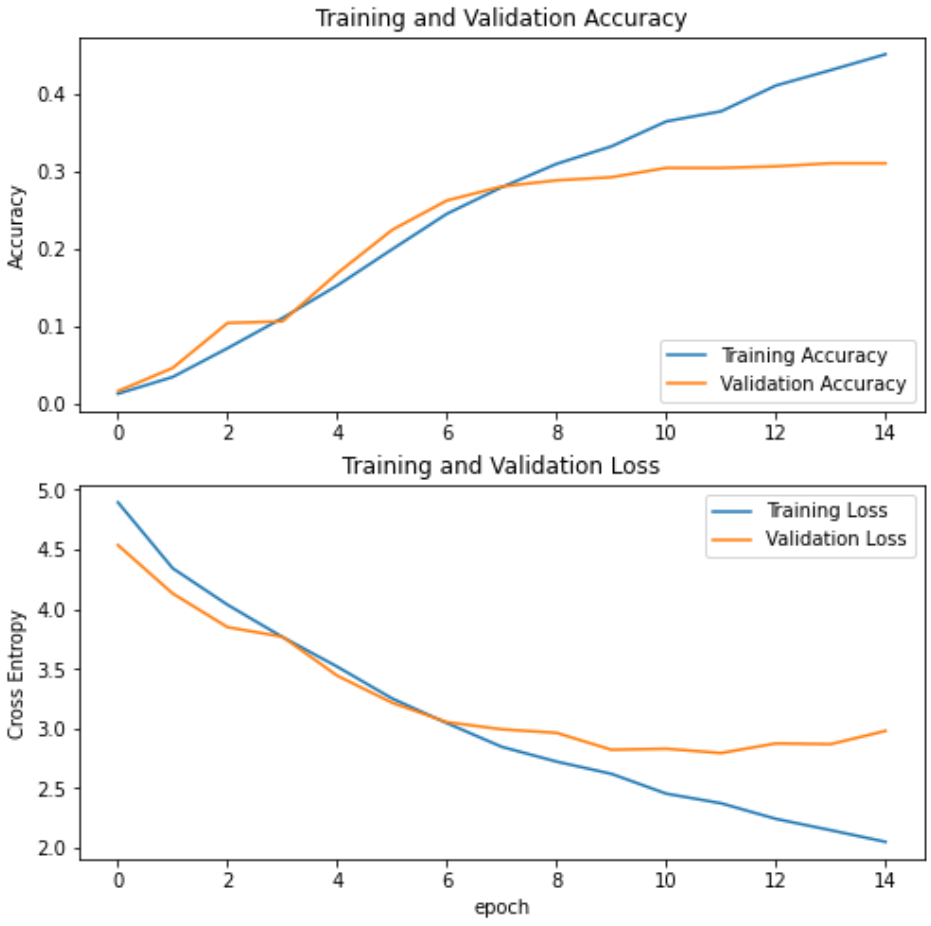
Obrázok 30 Konfúzna matica model 3

Iné nastavenia

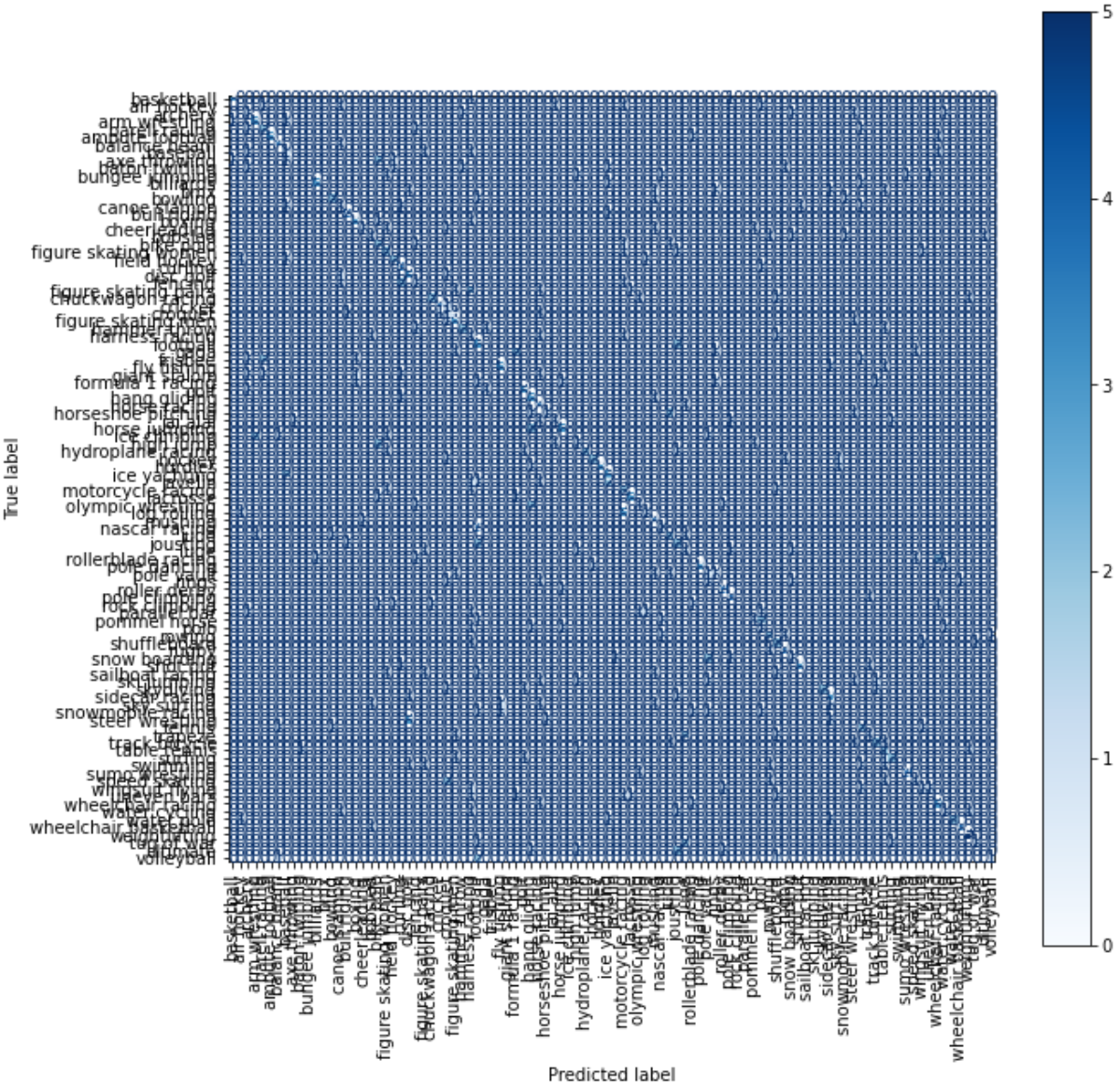
Pri modeli 4 je taktiež možné sledovať pretrénovanie.

```
model = Sequential()
model.add(Conv2D(64,3, padding='same', input_shape=(32,32,3), activation='relu'))

model.add(MaxPooling2D())

model.add(Conv2D(64,3, padding='same', activation='relu'))

model.add(MaxPooling2D())

model.add(Conv2D(128,3, padding='same', activation='relu'))

model.add(MaxPooling2D())

#model.add(GlobalAveragePooling2D())
#·model.add(Dense(128))
model.add(Flatten())

model.add(Dense(256, activation='relu'))
#model.add(Dropout(0.3))

model.add(Dense(100,activation='softmax'))

optimizer = keras.optimizers.Adam(learning_rate=0.01)

model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics = ['accuracy'], run_eagerly=True)
print(model.summary())
```

Obrázok 31 Model 4



Obrázok 32 Priebeh trénovania model 4

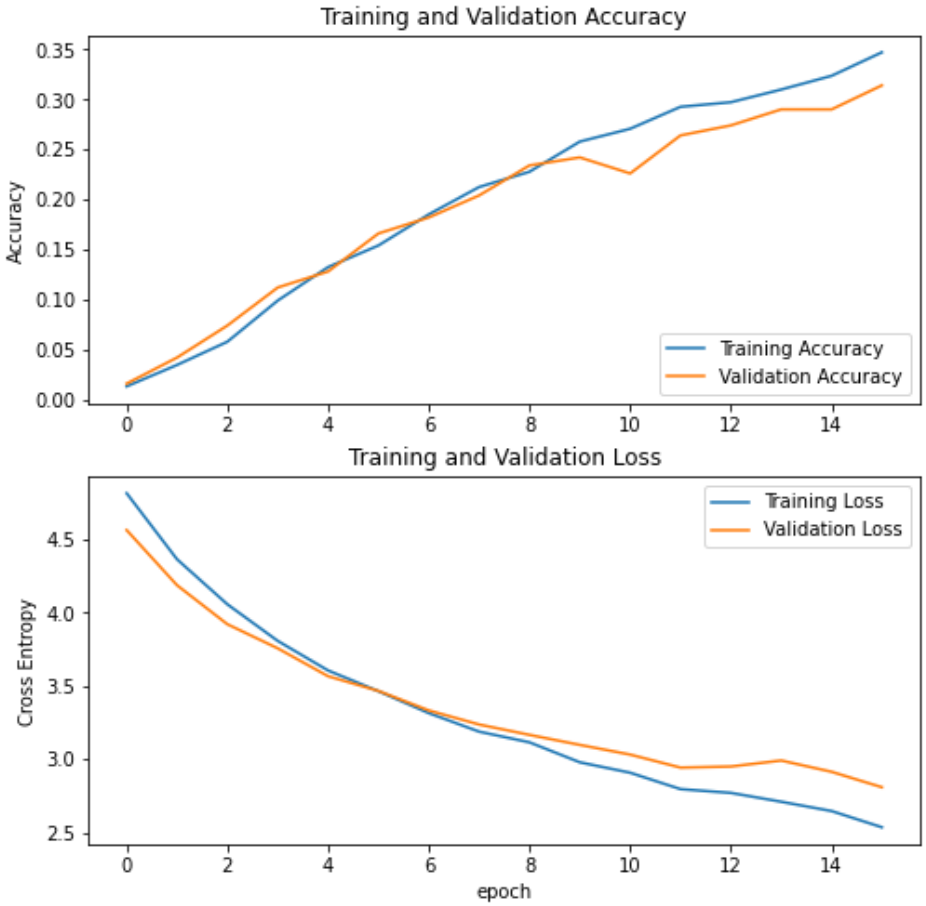


Obrázok 33 Konfúzna matica model 4

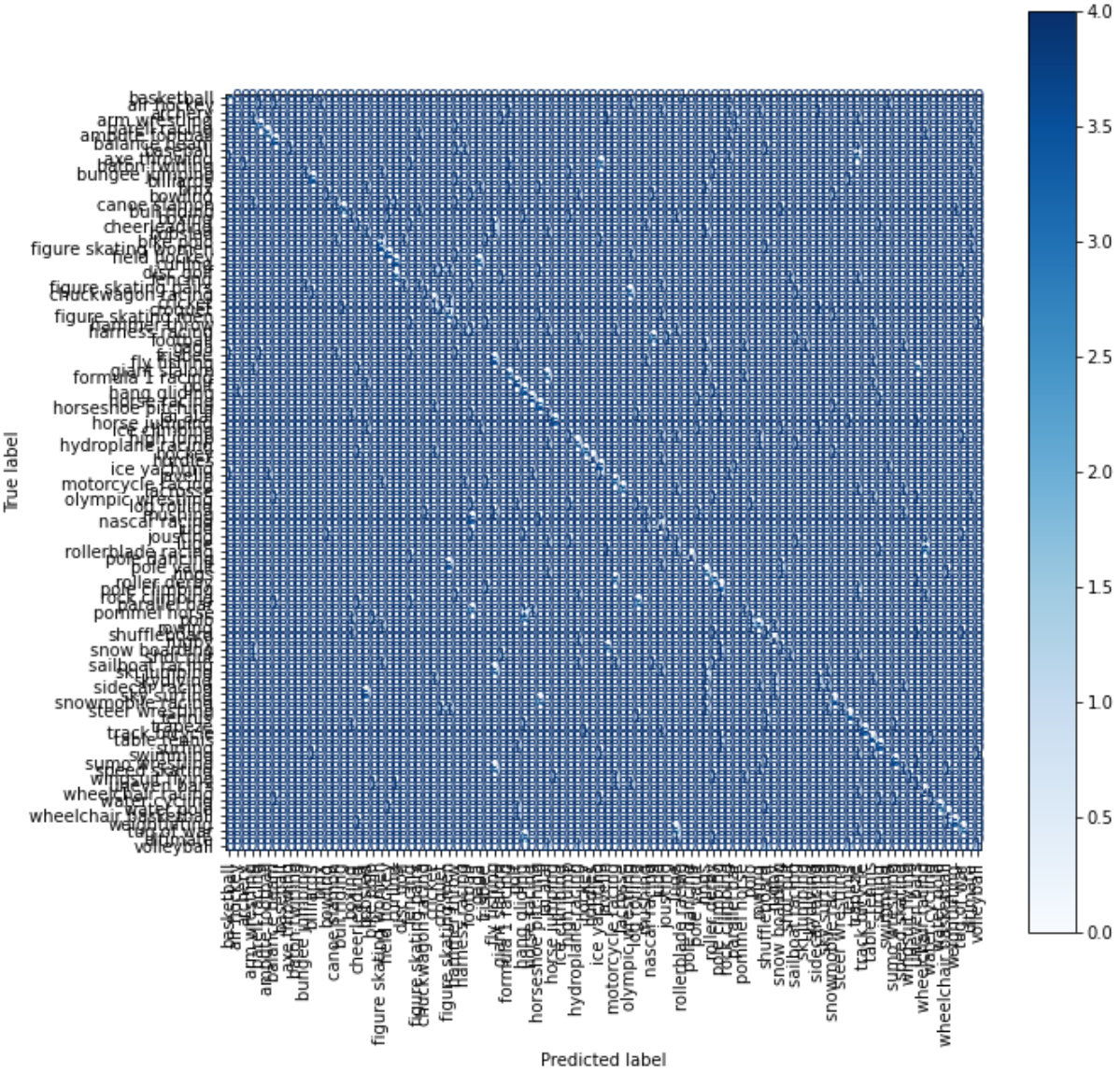
Pri modeli 5 je viditeľné zlepšenie oproti pretrénovaným modelom a grafy priebehu približne kopírujú grafy modelu 1. Rovnako aj úspešnosti, ktoré sú rovnako zhruba 30%.

```
model = Sequential()
model.add(Conv2D(64,3, padding='same', input_shape=(32,32,3), activation='relu'))

model.add(MaxPooling2D())

model.add(Conv2D(128,3, padding='same', activation='relu'))

model.add(MaxPooling2D())

model.add(Conv2D(128,3, padding='same', activation='relu'))

model.add(MaxPooling2D())


model.add(Flatten())

model.add(Dense(128, activation='relu'))
#model.add(Dropout(0.2))

model.add(Dense(100,activation='softmax'))

optimizer = keras.optimizers.Adam(learning_rate=0.01)

model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics = ['accuracy'], run_eagerly=True)
print(model.summary())
```

Obrázok 34 Model 5



Obrázok 35 Konfúzna matica model 5



Obrázok 36 Konfúzna matica model 5