

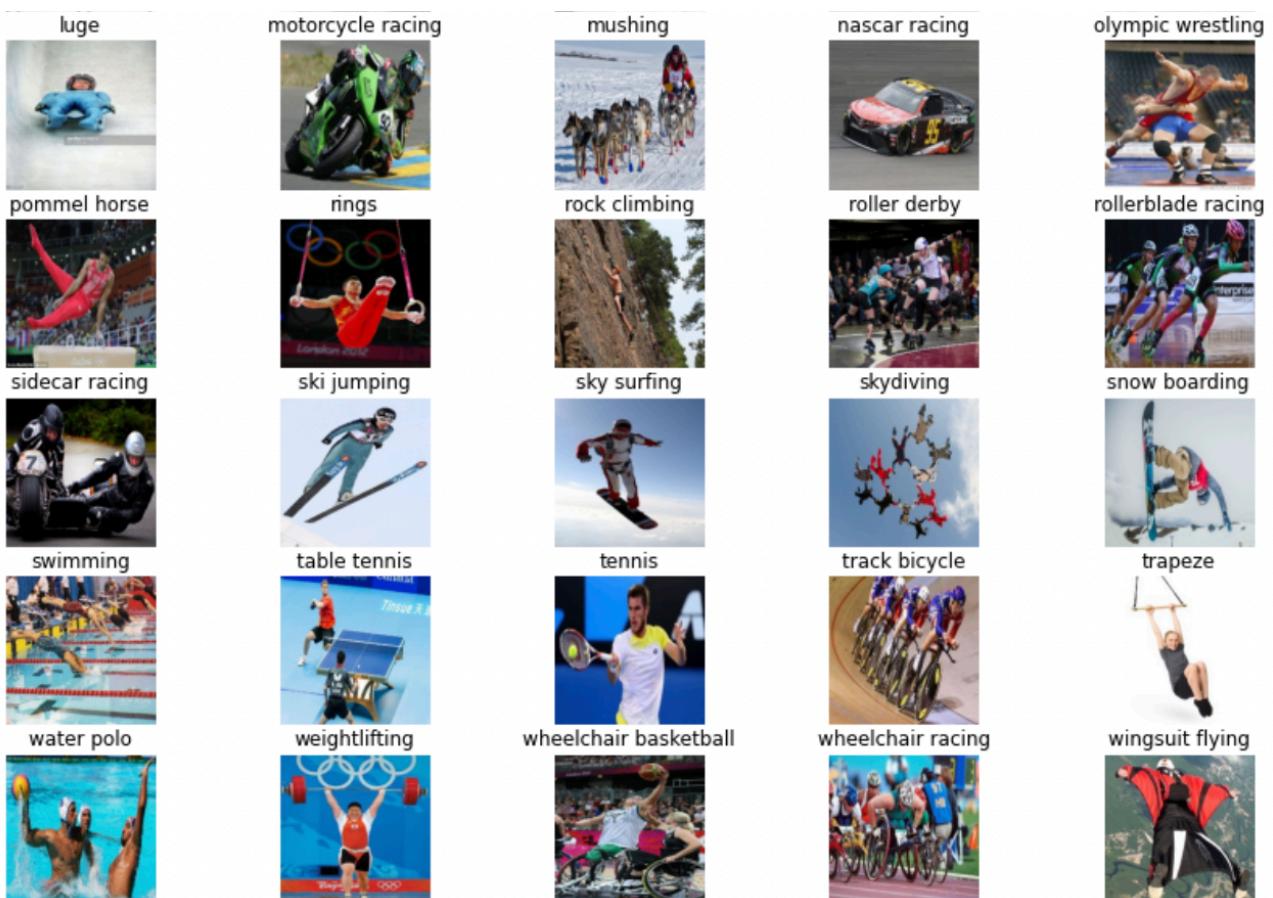
I-SUNS: Zadanie č.3
Konvolučné siete a prenos vedomostí

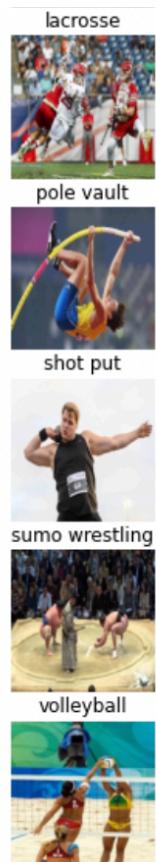
Matúš Vetrík

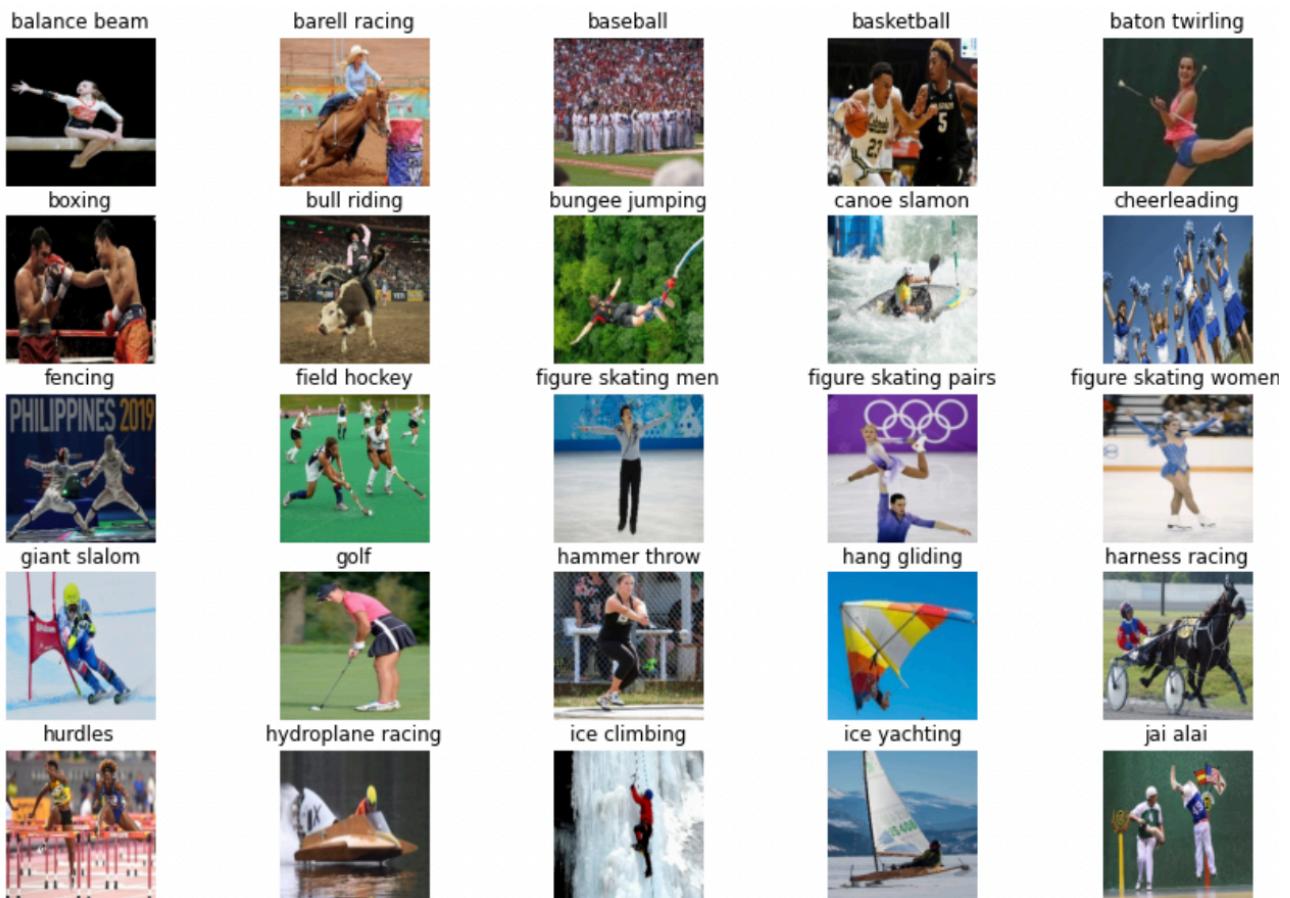
Úlohou práce je zanalyzovať skupinu fotografií zobrazujúce rôzne športy a ntrénovat' neurónovú siet', ktorá bude vedieť rozoznať šport na základe danej fotografie.

EDA

Náš dataset obashuje 100 kategórií športov. V trénovacej množine máme 13 667 obrázkov, v testovacej množine 601 obrázkov a vo validačnej množine taktiež 601 obrázkov. Ako prvé si zobrazíme všetký kategórie a pre každú kategóriu aj jedného reprezentanta.

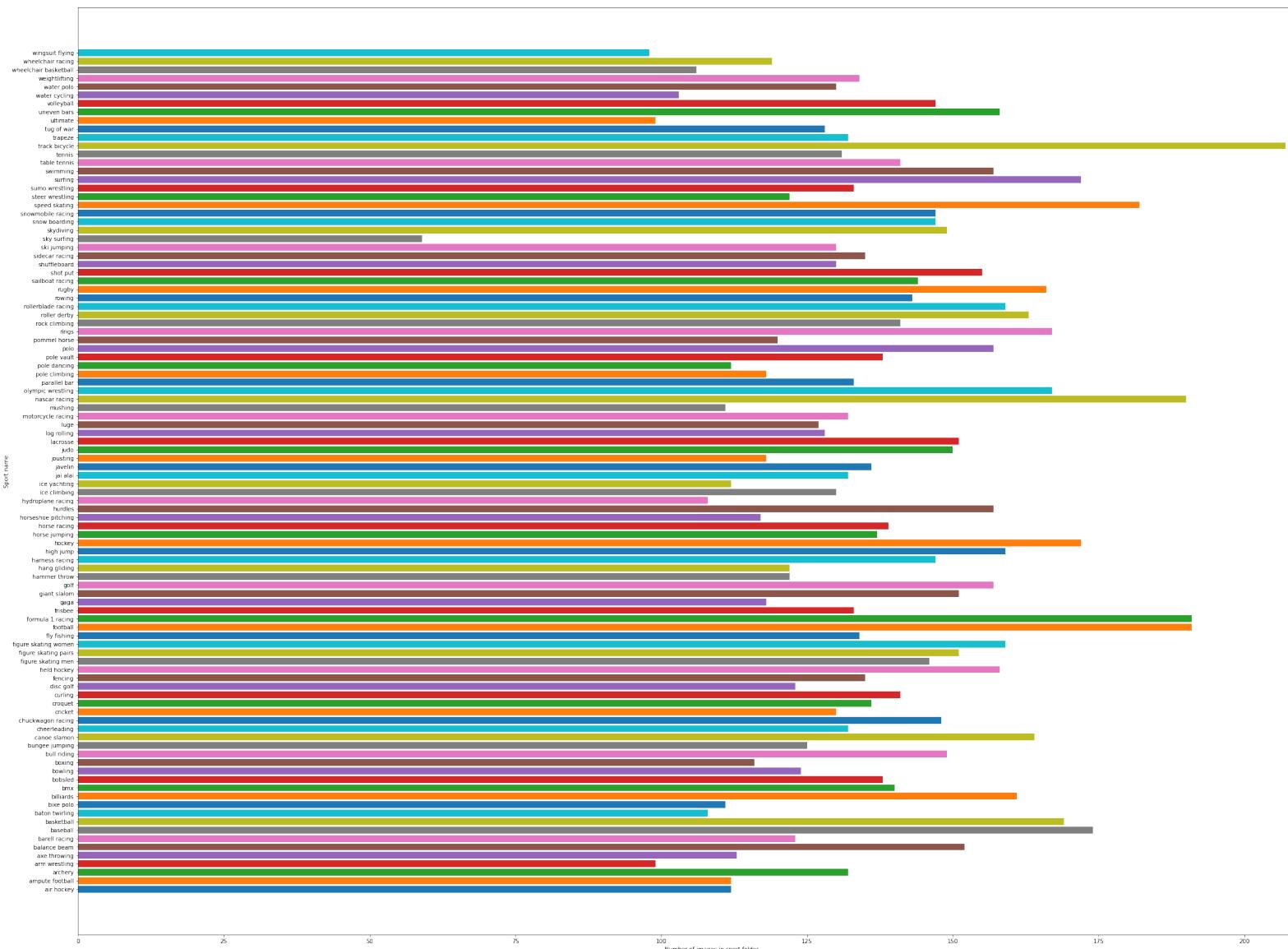






Ako môžeme vidieť nachádzajú sa tu rôznorodé športy. Od klasických športov ako hokej, futbal, tenis či basketbal tak aj extrémne športy ako jazda na býkovi, paragliding či sky surfing. Ďalej tu máme mnoho zimných športov ako lyžovanie, krasokorčuľovanie alebo boby ale ja letných športov ako rugby, plávanie alebo golf.

Ďalej si ukážeme početnosti v jednotlivých triedach. Najvyššiu početnosť podľa nasledujúceho grafu má cyklistika. Taktiež vysokú početnosť má nascar, formula 1, football a baseball. Na druhej strane najnižšiu početnosť má šport sky surfing. Taktiež nízku početnosť môžeme vidieť v triedach športov arm wrestling, bike polo a ultimate.



Následne si zobrazíme dataset podľa farebnosti. Na nasledujúcej ukážke zobrazíme každú kategóriu a jej najbežnejšiu/najčastejšiu farbu.

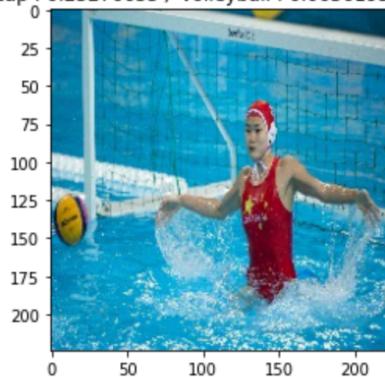




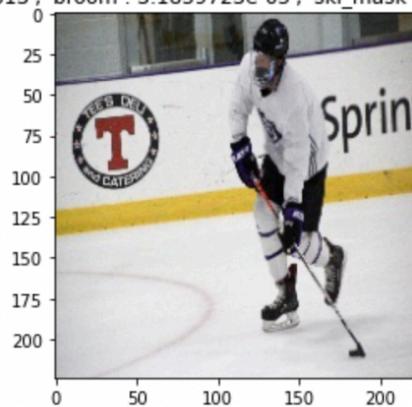
Ako môžeme vidieť mnohé športy by sa nedali identifikovať na základe farby ale niektoré aj áno. Napríklad na šport vodná cyklistika nám vyšla modrá farba, na pozemký hokej, ktorý sa hrá na trávniku vyšla zelená farba a paraglading bledo modrá farba.

Niekteré obrázky si analyzujeme aj na vopred vytvorennej neurónovej sieti imagenet. Zoberieme si model VGG16 a skúsime vyhodnotiť pár obrázkov. Ako očakávaný výstup budú objekty a ich percentuálna zhoda, ktoré model na obrázku našiel.

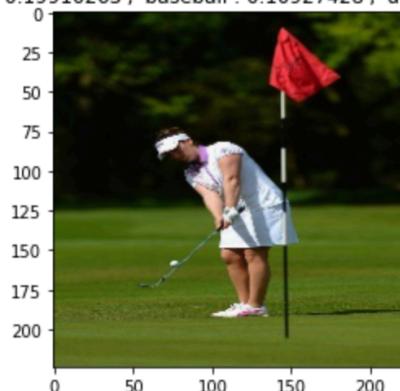
['swimming_trunks : 0.60930127', 'bathing_cap : 0.25276655', 'volleyball : 0.06361939', 'maillot : 0.022047017', 'bikini : 0.015200613']



['puck : 0.99977595', 'ski : 0.00018742813', 'broom : 3.1859723e-05', 'ski_mask : 8.3226814e-07', 'knee_pad : 8.287563e-07']



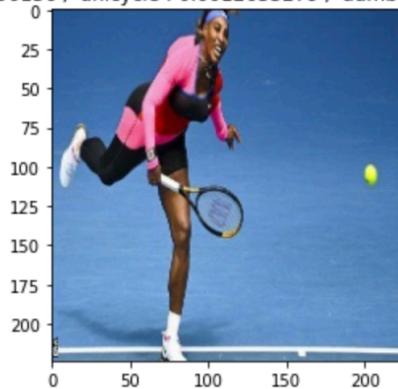
['croquet_ball : 0.36091805', 'ballplayer : 0.19910263', 'baseball : 0.10927428', 'umbrella : 0.09808033', 'golf_ball : 0.07314573']



['volleyball : 0.9618205', 'horizontal_bar : 0.016262012', 'basketball : 0.0086868955', 'balance_beam : 0.006060118', 'parallel_bars : 0.0018243139']



['racket : 0.99082196', 'tennis_ball : 0.00590136', 'unicycle : 0.0012633279', 'dumbbell : 0.0009105392', 'barbell : 0.00022802522']

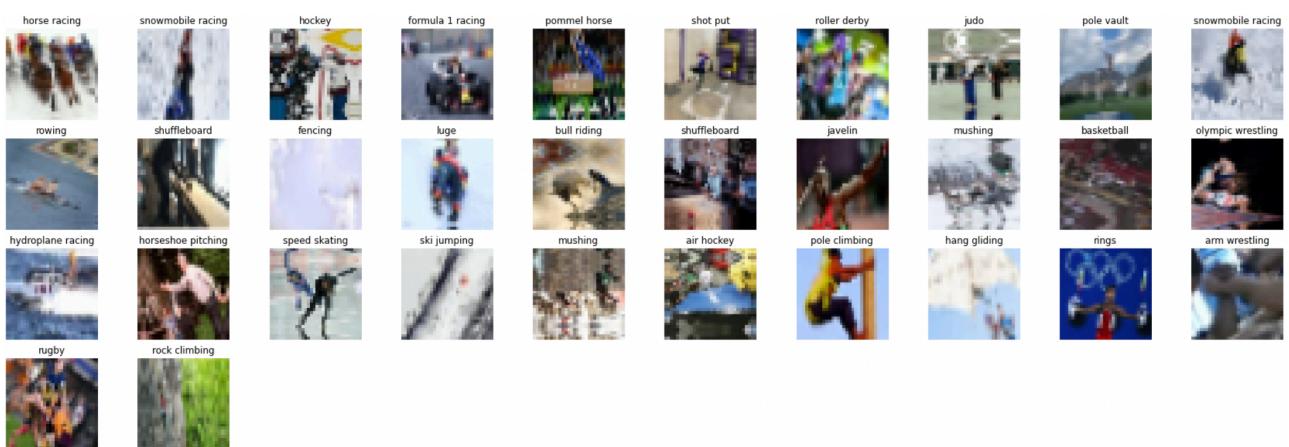


Pri prvom obrázku kde je vyobrazené vodné pólo model vyhodnotil na 60%, že na obrázku našiel plavky a na 25%, že našiel čiapku na plávanie. Pri druhom obrázku ľadového hokeja model našiel puk na 99%. Na treťom obrázku golfu si model pomýlil hru s kriketom a vyhodnotil, že objavil kriketovú palicu a basebal. Na štvrtom obrázku volejbalu model objavil volleyball a horizontálnu tyč. A nakoniec na poslednom obrázku tenisu model našiel raketu a tenisovú loptu. Model VGG16 väčšinu obrázkov vyhodnotil korektnie.

Spracovanie dát

Dáta nie sú v správnom formáte na trénovanie neurónovej siete, preto musíme podniknúť nasledujúce kroky. Ako prvé si musíme previesť fotografie do datasetu, s ktorým budeme vedieť ďalej narábať. Použili sme generátor na prevedenie fotografií na databázu pomocou knižnice `keras.utils.image_dataset_from_directory`. Táto funkcia prevedie obrázok do datasetu kde každý pixel je preformatovaný do `rgb` tvaru. Týmto spôsobom vieme porovnávať obrázky na základe podobnosti medzi hodnotami `rgb`. Obrázky sú taktiež v príliš veľkých rozmeroch a mohlo by nám to výrazne predĺžiť trénovanie. Preto zmenšíme obrázky na veľkosť 32×32 .

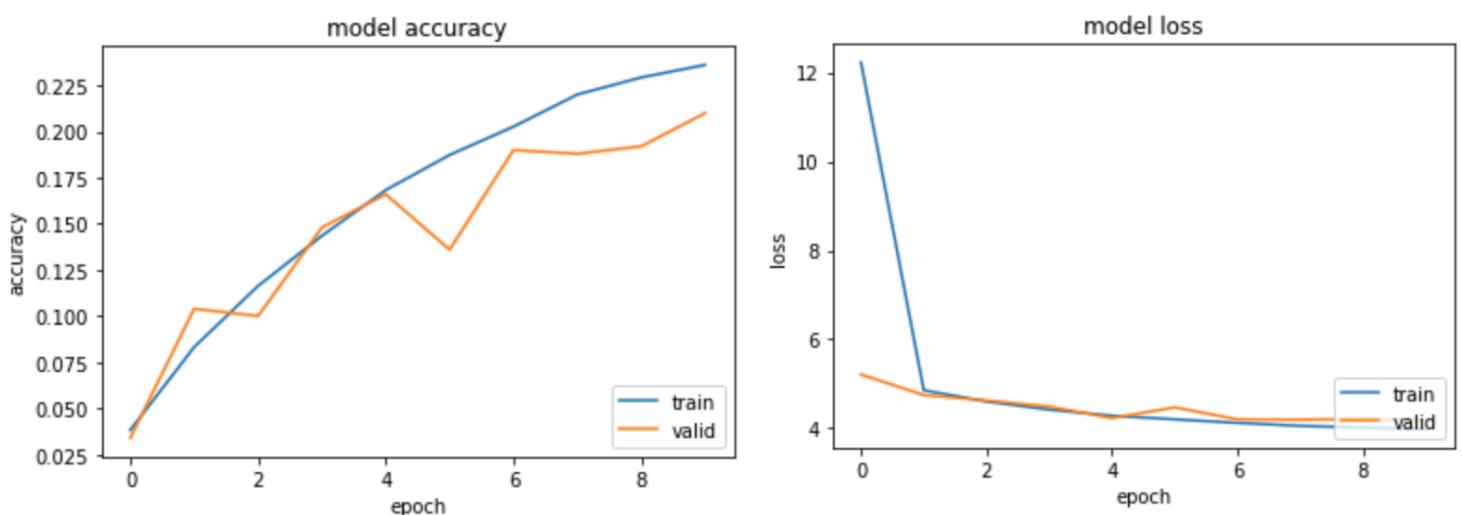
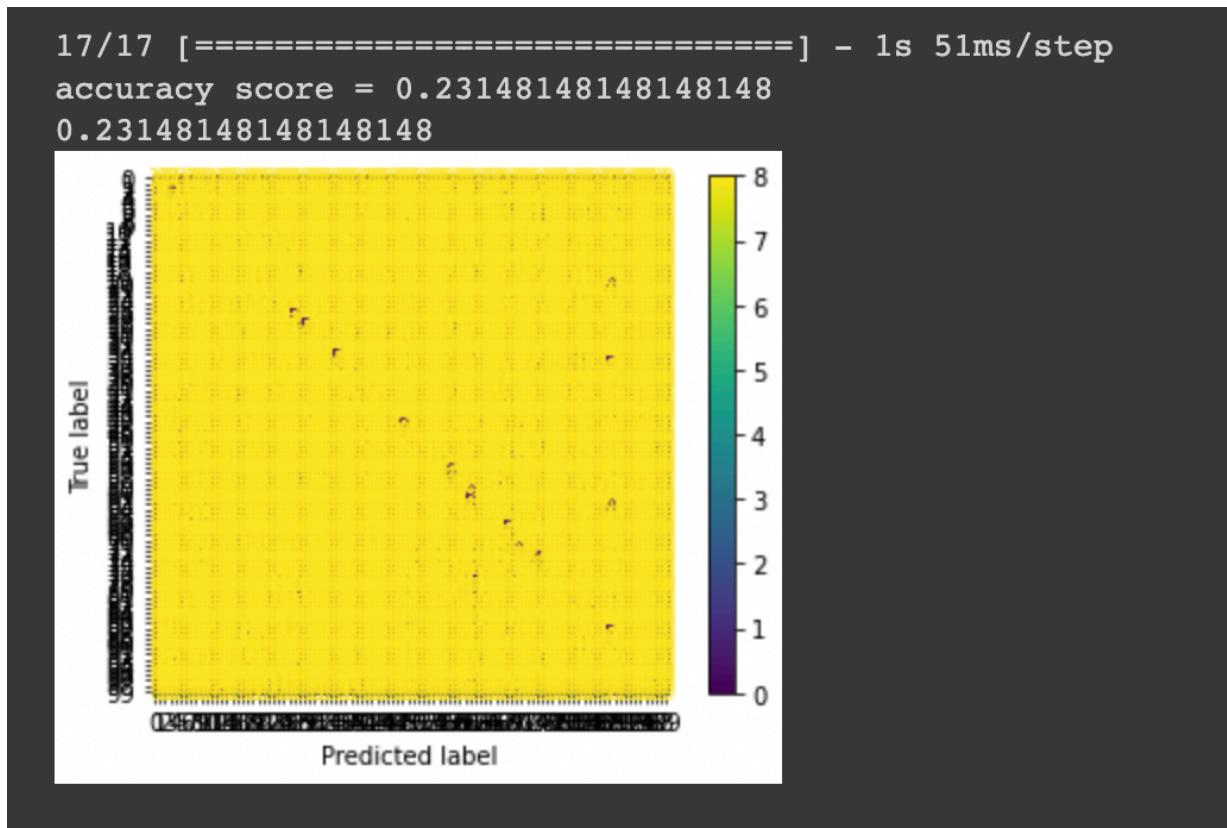
Ďalej musíme dáta správne augmentovať. Augmentácia je proces pri ktorom robíme dataset viac rôznorodejší, čiže obrázky z kategórií náhodne otáčame podľa osi x/y , približujeme a rotujeme. Použitie augmentácie môžeme vidieť na nasledovnej ukážke.



Ďalším krokom je normalizácia datasetu. V prvom kroku sme hodnoty, pri prevádzaní obrázkov do datasetu, preformátovali z pixelov do hodnot `rgb` v intervale 0-255. Tieto hodnoty by neurónová sieť sa pri trénovaní t'azko čítala, takže predelíme každú hodnotu hodnotou 255 aby sme dostali hodnoty v intervale od 0 po 1.

Trénovanie

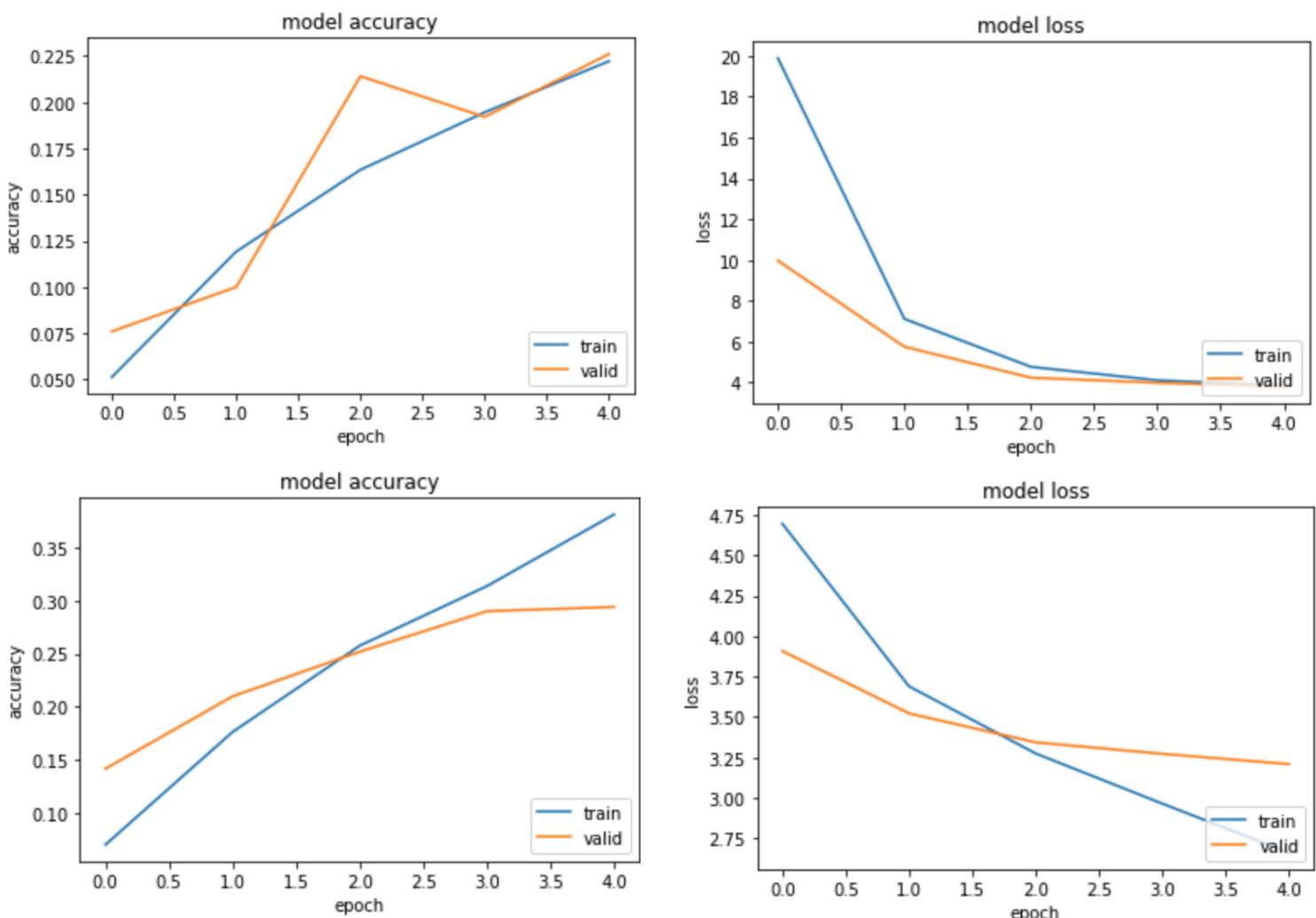
Vytvorili sme model s konvolučnými vrstvami na rozpoznávanie orázkov. Použili sme model Sequential a pridali sme doň 3 konvolučné vrstvy. Konvolučným vrstvám sme nastavili filter s rozmermy 3×3 . Prvej vrstve sme nastavili 32 filtrov a zvyšným dvom 64 filtrov. Výsledky sú nasledovné.

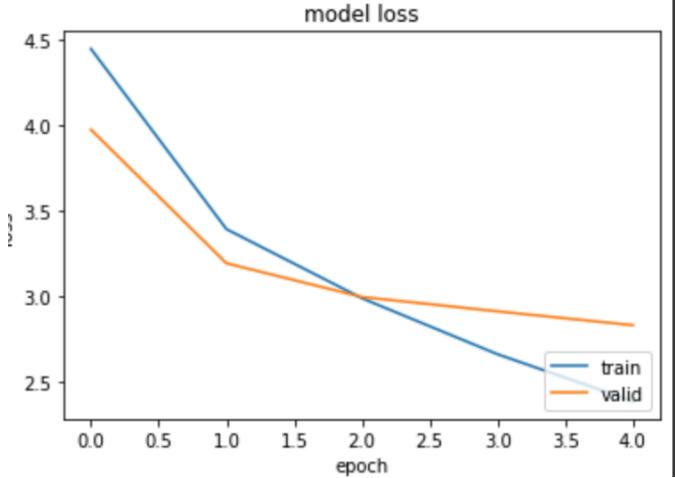
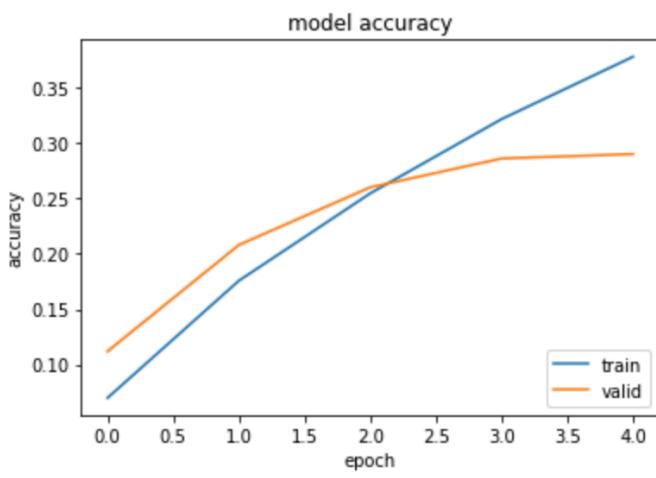
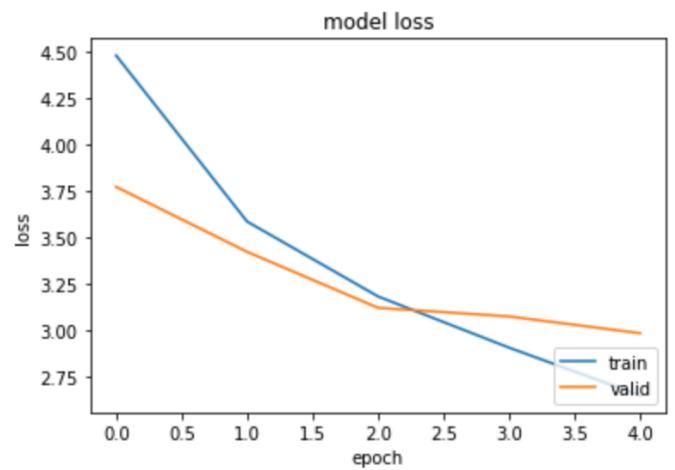
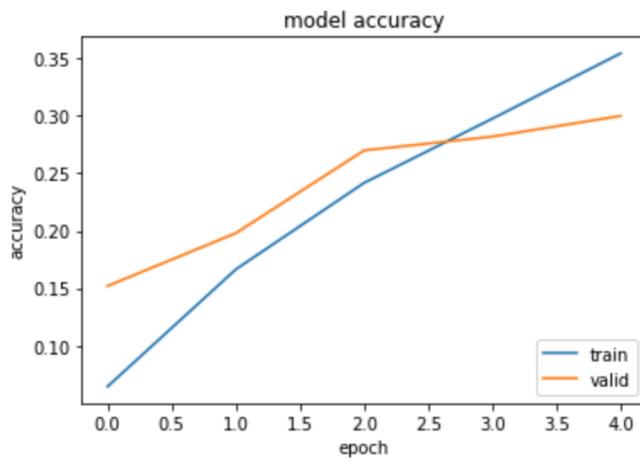
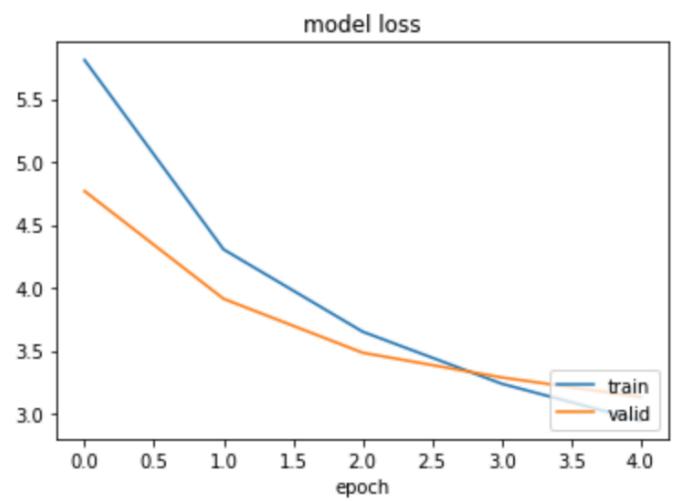
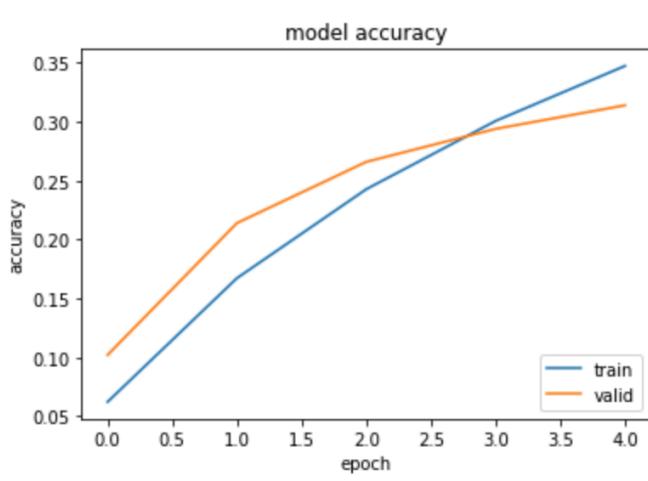
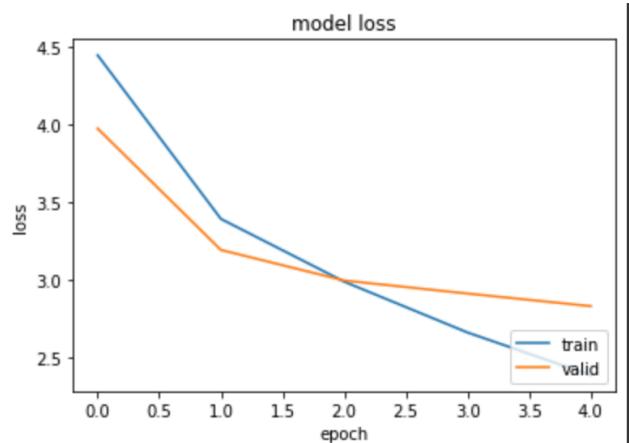
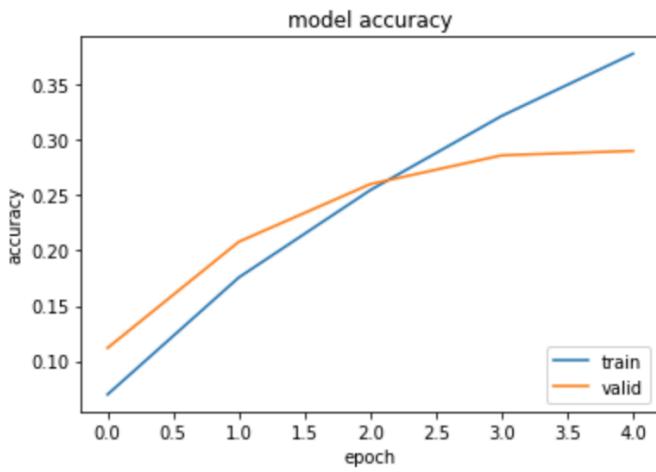


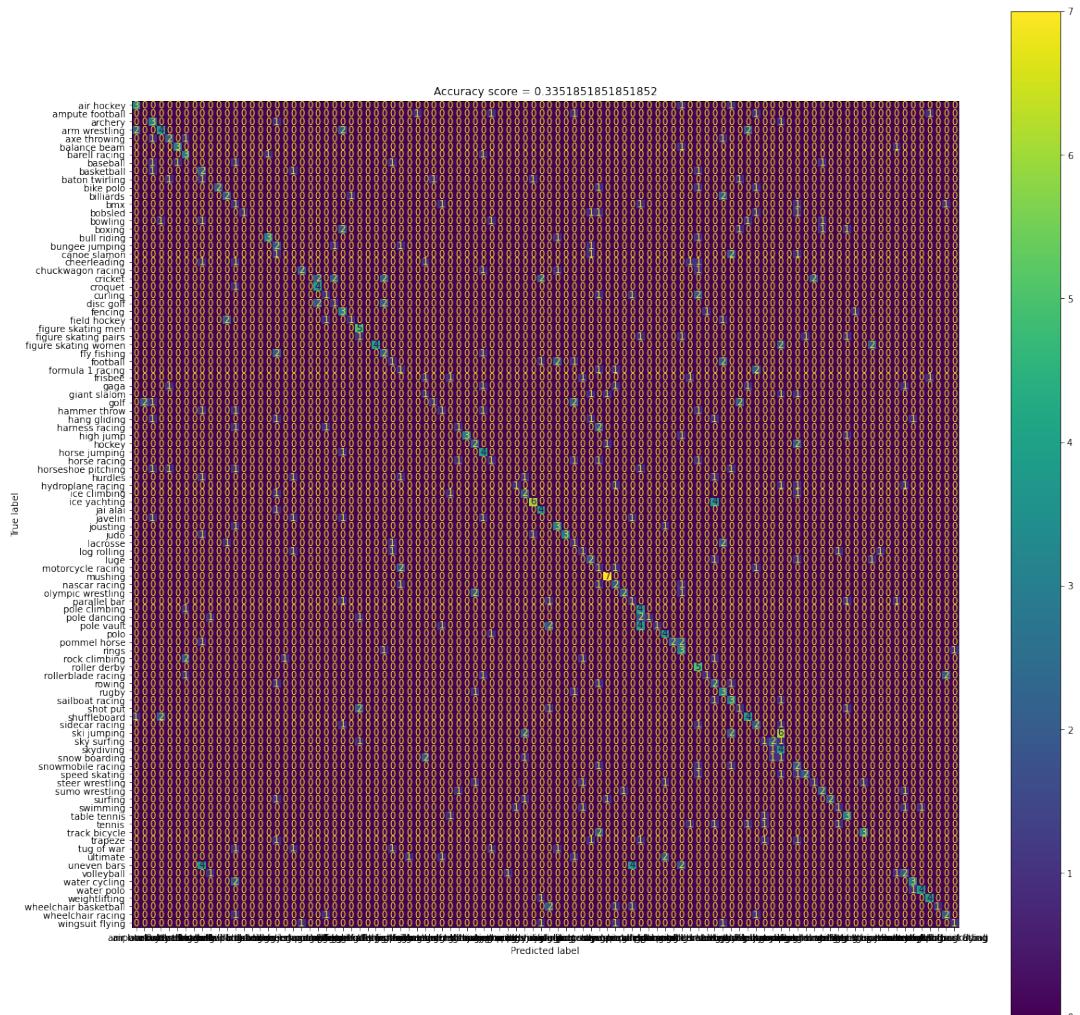
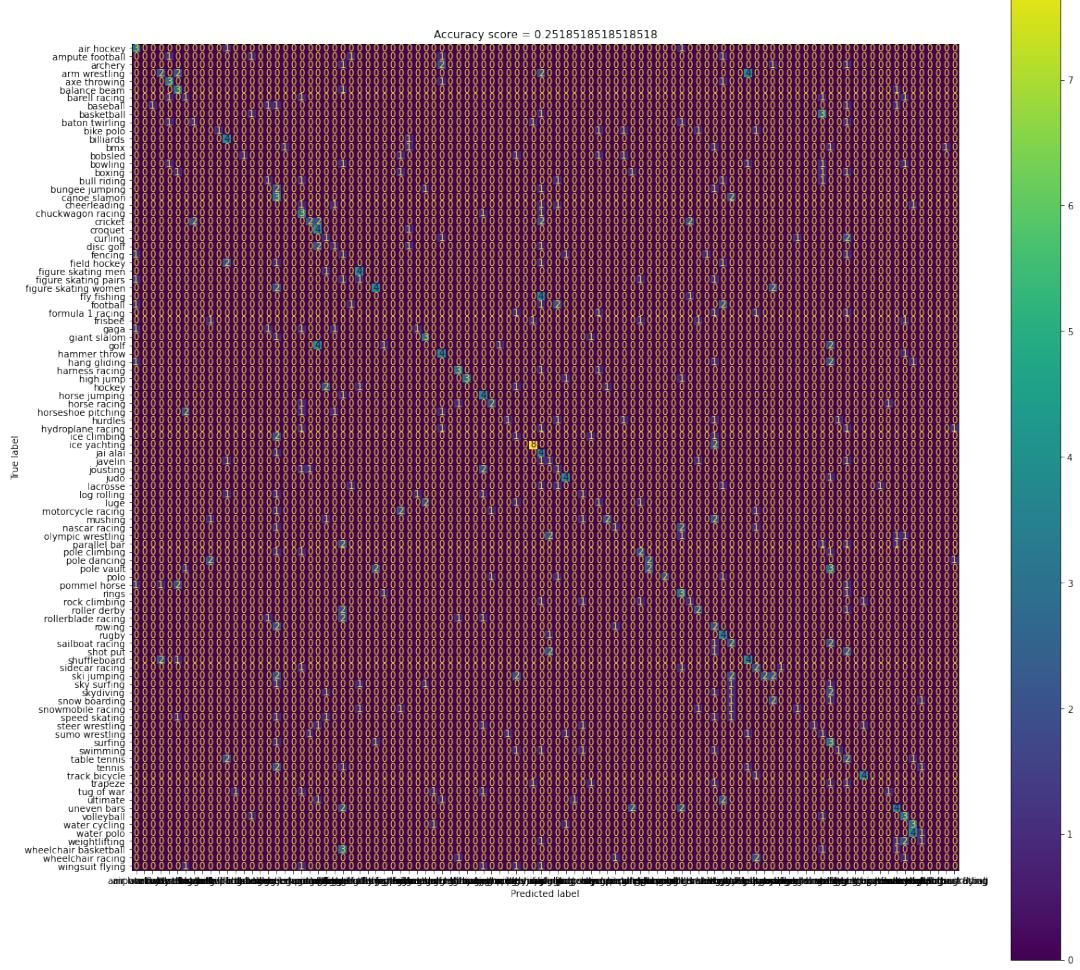
Ako môžeme vidieť, výsledky nie sú vôbec priaznivé. Taktiež na koleračnej matici nie je vidno ani hlavná diagonála. Pre lepšiu viditeľnosť koleračnú maticu zväčšíme v ďalších pokusoch.

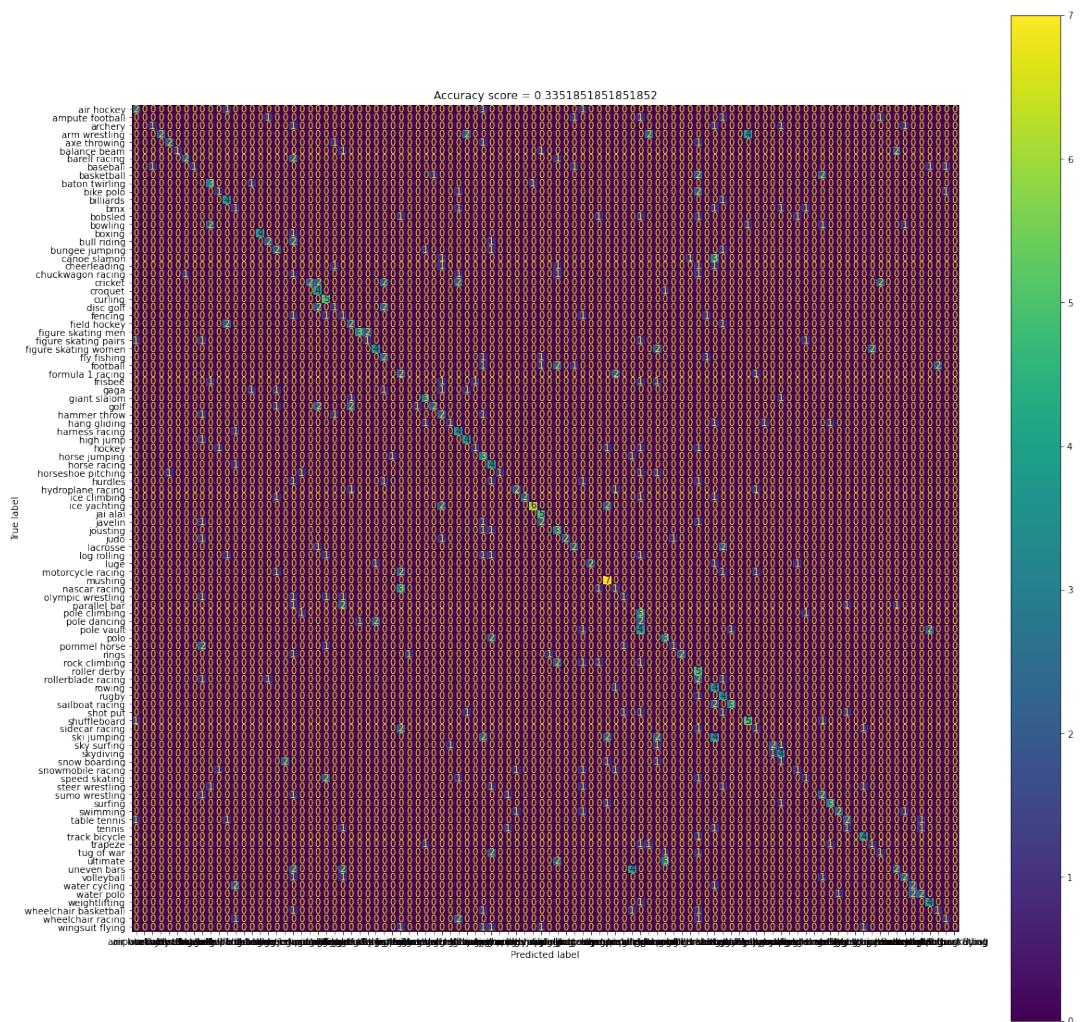
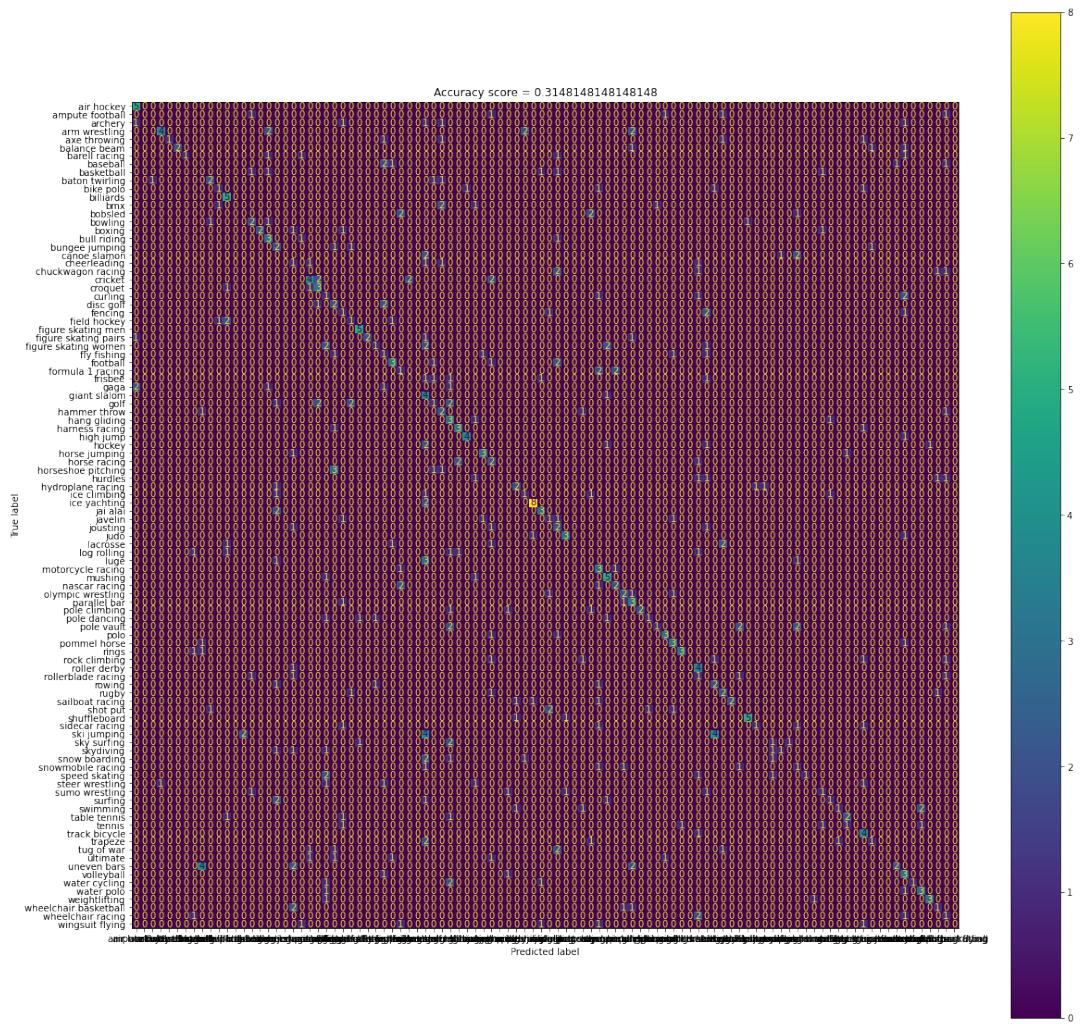
Pre zlepšenie presnosti pridáme v ďalších pokusoch aj regularizátory. Skúsime si vytvoriť vlastný grid search s dvoma regularizátormi $L1$ a $L2$, a s rôznymi hodnotami regularizátorov. Kvôli zdlžavému vypočítavaniu bude testovať s 5 epochami.

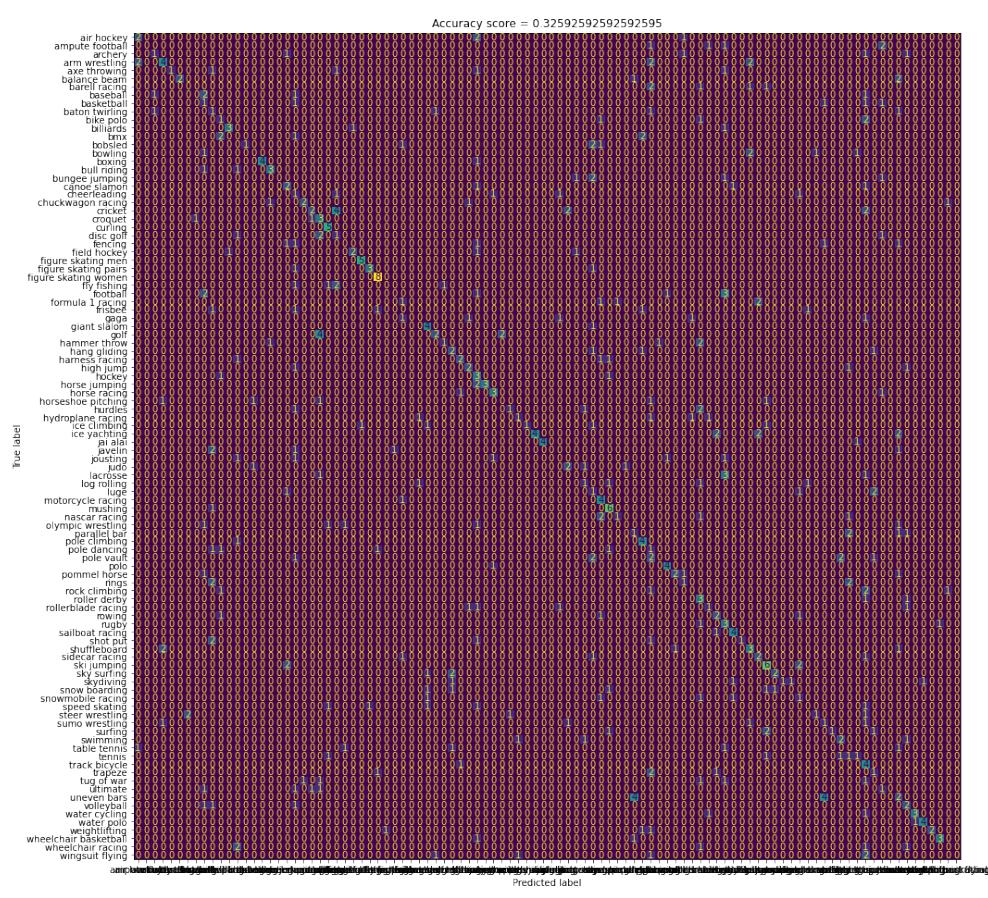
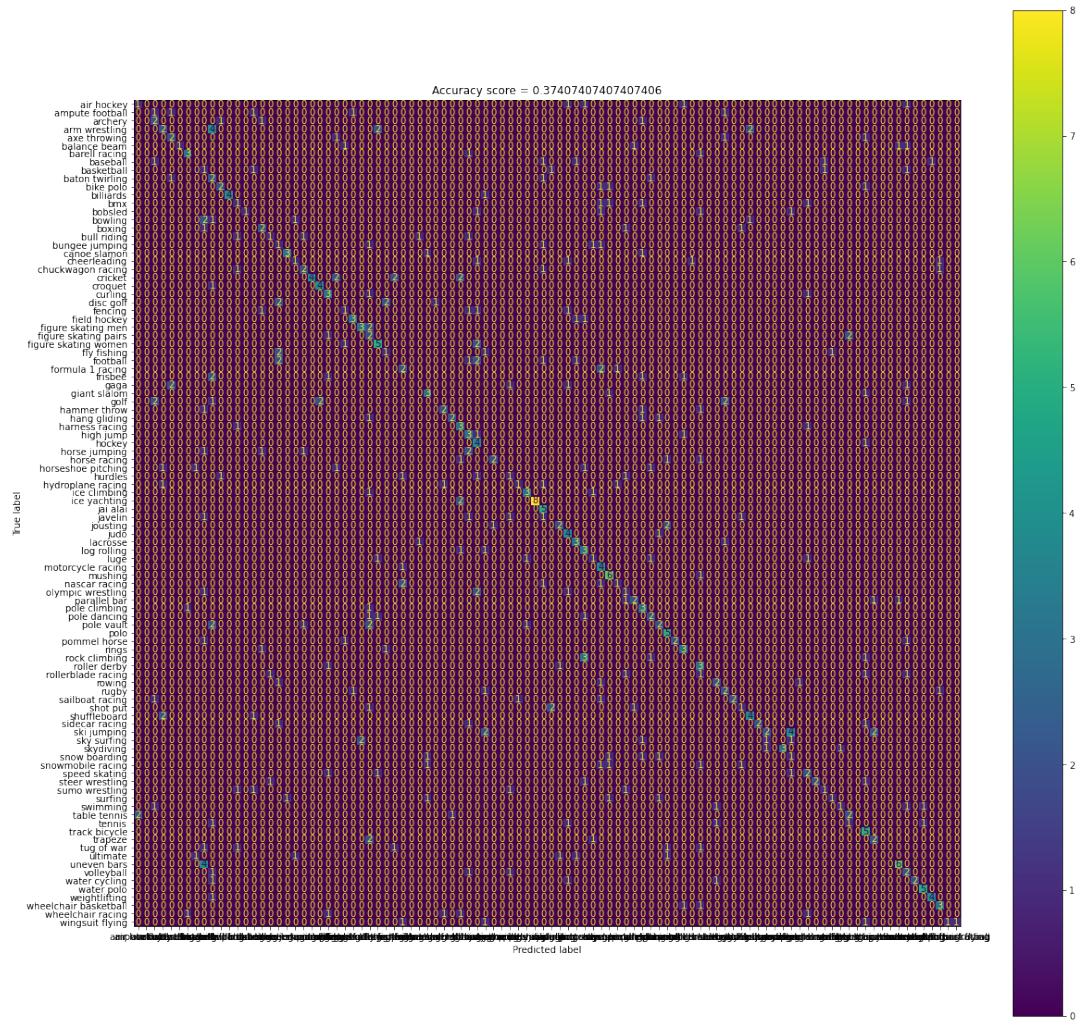
	L1 - 0.01	L1 - 0.001	L1 - 0.0001	L2 - 0.01	L2 - 0.001	L2 - 0.0001
Hodnota	0.01	0.001	0.0001	0.01	0.001	0.0001
Epochy	5	5	5	5	5	5
Presnosť	0.2518	0.3351	0.3148	0.3351	0.3740	0.3259





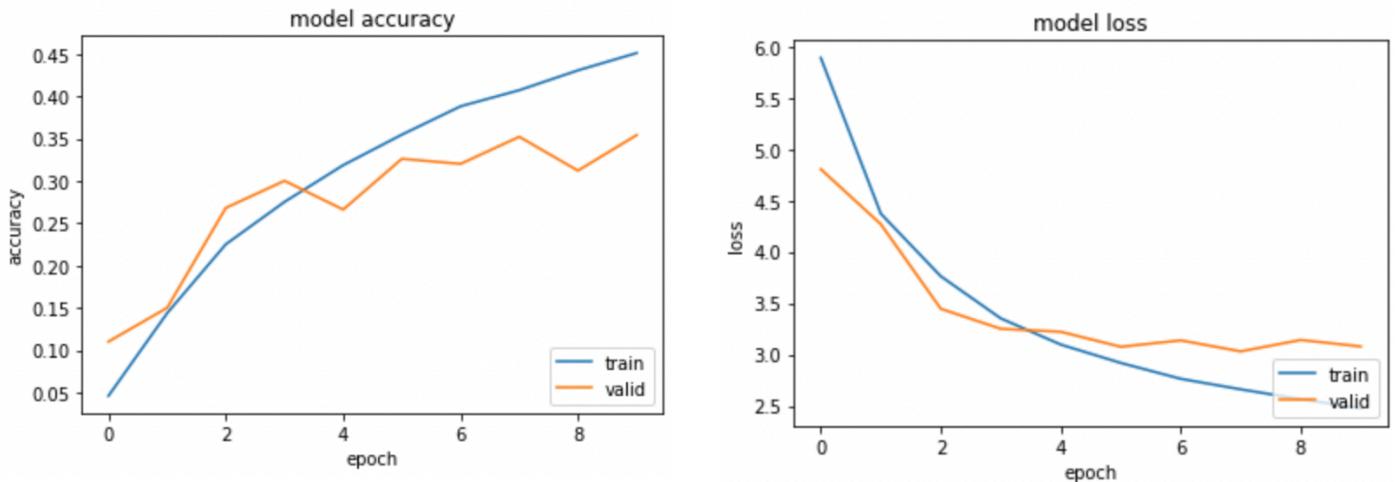






Ako môžeme z tabuľky vidieť s najväčšou presnosťou bol natrénovaný model s regularizátorom $L2$ s hodnotou 0.001. Na konfúznej matici taktiež môžeme vidieť najviditeľnejšiu diagonálu z pomedzi všetkých konfúznych matíc.

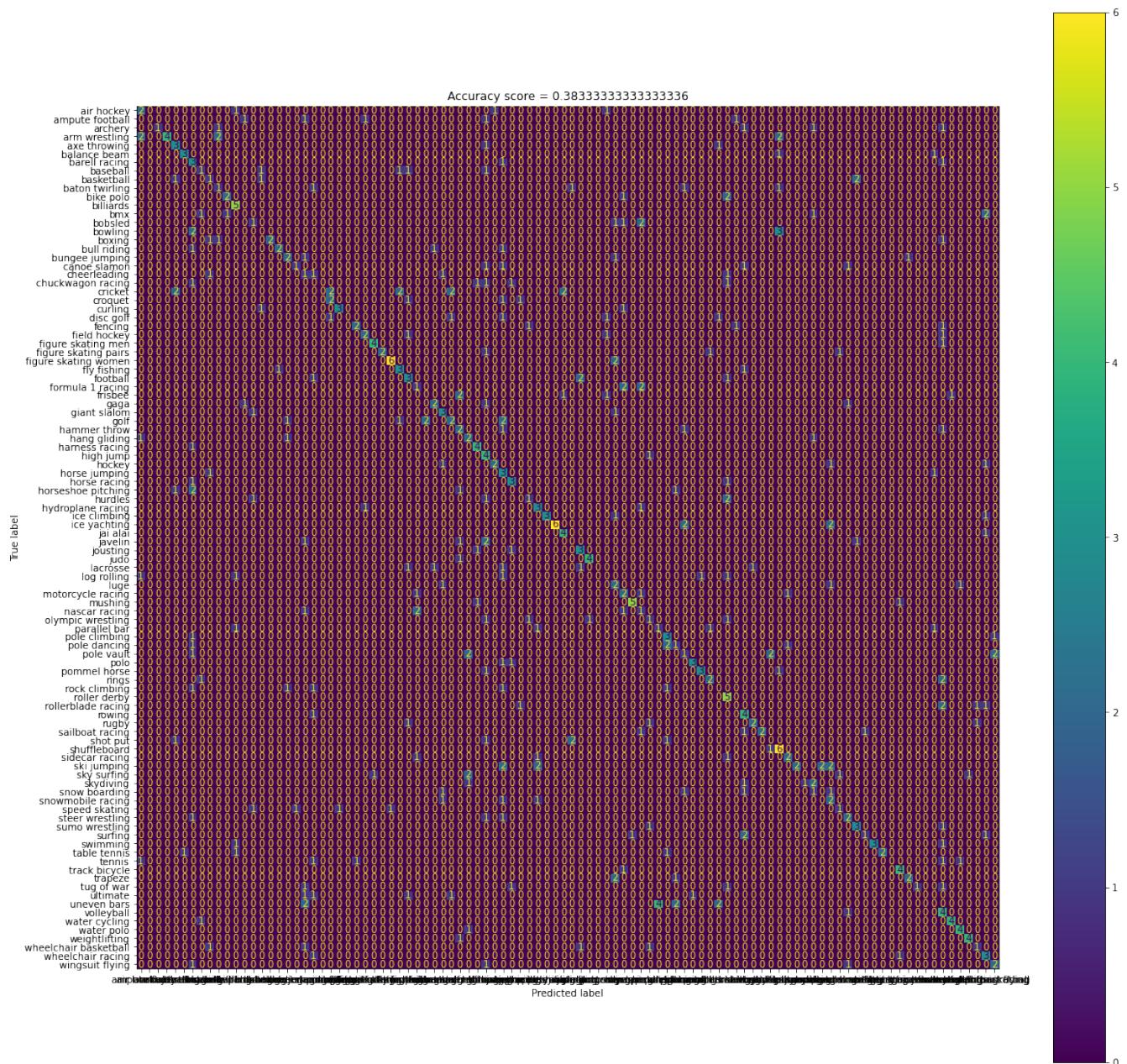
Použijeme tieto parametre do našej sieti a zvýšime počet konvolučných sietí o jeden s filtrom 64. Taktiež zvýšime epochy z 5 na 10. Výsledky sú nasledovné.



```

Epoch 2: val_accuracy improved from 0.11000 to 0.15000, saving model to checkpoints/normal_model.ckpt
433/433 [=====] - 22s 51ms/step - loss: 4.3713 - accuracy: 0.1437 - val_loss: 4.2689 - val_accu
Epoch 3/10
431/433 [=====>.] - ETA: 0s - loss: 3.7619 - accuracy: 0.2249
Epoch 3: val_accuracy improved from 0.15000 to 0.26800, saving model to checkpoints/normal_model.ckpt
433/433 [=====] - 22s 50ms/step - loss: 3.7616 - accuracy: 0.2251 - val_loss: 3.4452 - val_accu
Epoch 4/10
425/433 [=====>.] - ETA: 0s - loss: 3.3525 - accuracy: 0.2747
Epoch 4: val_accuracy improved from 0.26800 to 0.30000, saving model to checkpoints/normal_model.ckpt
433/433 [=====] - 22s 50ms/step - loss: 3.3515 - accuracy: 0.2751 - val_loss: 3.2502 - val_accu
Epoch 5/10
430/433 [=====>.] - ETA: 0s - loss: 3.0966 - accuracy: 0.3189
Epoch 5: val_accuracy did not improve from 0.30000
433/433 [=====] - 30s 67ms/step - loss: 3.0966 - accuracy: 0.3185 - val_loss: 3.2213 - val_accu
Epoch 6/10
428/433 [=====>.] - ETA: 0s - loss: 2.9178 - accuracy: 0.3548
Epoch 6: val_accuracy improved from 0.30000 to 0.32600, saving model to checkpoints/normal_model.ckpt
433/433 [=====] - 30s 69ms/step - loss: 2.9180 - accuracy: 0.3545 - val_loss: 3.0751 - val_accu
Epoch 7/10
431/433 [=====>.] - ETA: 0s - loss: 2.7629 - accuracy: 0.3882
Epoch 7: val_accuracy did not improve from 0.32600
433/433 [=====] - 23s 52ms/step - loss: 2.7632 - accuracy: 0.3880 - val_loss: 3.1374 - val_accu
Epoch 8/10
431/433 [=====>.] - ETA: 0s - loss: 2.6603 - accuracy: 0.4071
Epoch 8: val_accuracy improved from 0.32600 to 0.35200, saving model to checkpoints/normal_model.ckpt
433/433 [=====] - 22s 49ms/step - loss: 2.6608 - accuracy: 0.4070 - val_loss: 3.0305 - val_accu
Epoch 9/10
425/433 [=====>.] - ETA: 0s - loss: 2.5661 - accuracy: 0.4299
Epoch 9: val_accuracy did not improve from 0.35200
433/433 [=====] - 23s 52ms/step - loss: 2.5648 - accuracy: 0.4305 - val_loss: 3.1414 - val_accu
Epoch 10/10
430/433 [=====>.] - ETA: 0s - loss: 2.4754 - accuracy: 0.4513
Epoch 10: val_accuracy improved from 0.35200 to 0.35400, saving model to checkpoints/normal_model.ckpt
433/433 [=====] - 24s 54ms/step - loss: 2.4769 - accuracy: 0.4509 - val_loss: 3.0782 - val_accu

```



Na ukážkach môžeme vidieť priebeh trénovanie v grafoch ale aj výpisoch po epochách. Taktiež môžeme vidieť konfúznu maticu so zretelne viditeľnou diagonálou. Výsledok sa nám zlepšili v priemere o 1%. Výsledná presnosť je 38.333%. Týmto považujeme trénovanie neurónovej sieti za ukončené.

Transfer learning

Transfer learning slúži na využitie už vopred natrénovaného modelu na pre naše potreby. Týmto spôsobom môžeme daný model upraviť podľa našich preferencií a trénovať na tom. Vybrali sme si predtrénovaný model ResNet50 a predikneme si na ňom nase dátu. Vyhodené príznaky potrebujeme zredukovať na X dimensií pre rýchlejšie dodatočné trénovanie. Použili sme PCA s 3 komponentami na úpravu dimensií. Na príznakoch teraz naténujeme klasifikátor z minulého zadania. Vybrali sme si SVR model, pretože pri ňom sme sledovali najlepšie výsledky. Vyskúšame rôzne parametre pomocou grid searchu.

```
svc_para = {
    'kernel': ['rbf'],
    'C': [0.1, 0.01],
    'gamma': [0.001, 0.01]
}
```

Cross validation nastavíme na 3 a otestujeme grid search. Výsledky sú nasledovné.

```
Fitting 3 folds for each of 4 candidates, totalling 12 fits
[CV 1/3] END C=0.001, gamma=0.1, kernel=rbf;, score=(train=0.015, test=0.015) total time= 19.4s
[CV 2/3] END C=0.001, gamma=0.1, kernel=rbf;, score=(train=0.015, test=0.015) total time= 19.6s
[CV 3/3] END C=0.001, gamma=0.1, kernel=rbf;, score=(train=0.015, test=0.015) total time= 19.8s
[CV 1/3] END C=0.001, gamma=0.001, kernel=rbf;, score=(train=0.015, test=0.015) total time= 18.0s
[CV 2/3] END C=0.001, gamma=0.001, kernel=rbf;, score=(train=0.015, test=0.015) total time= 18.3s
[CV 3/3] END C=0.001, gamma=0.001, kernel=rbf;, score=(train=0.015, test=0.015) total time= 18.0s
[CV 1/3] END C=0.01, gamma=0.1, kernel=rbf;, score=(train=0.015, test=0.015) total time= 20.5s
[CV 2/3] END C=0.01, gamma=0.1, kernel=rbf;, score=(train=0.015, test=0.015) total time= 21.8s
[CV 3/3] END C=0.01, gamma=0.1, kernel=rbf;, score=(train=0.015, test=0.015) total time= 23.0s
[CV 1/3] END C=0.01, gamma=0.001, kernel=rbf;, score=(train=0.015, test=0.015) total time= 18.6s
[CV 2/3] END C=0.01, gamma=0.001, kernel=rbf;, score=(train=0.015, test=0.015) total time= 18.6s
[CV 3/3] END C=0.01, gamma=0.001, kernel=rbf;, score=(train=0.015, test=0.015) total time= 18.5s
Best params: {'C': 0.001, 'gamma': 0.1, 'kernel': 'rbf'}
Best scores: 0.014965298016485148
```

Najlepšie skóre sme dosiahli 1.49% čo nie je vôbec úspešne. Skúsimo ďalšiu úpravu parametrov.

```

svc_para = [
    'kernel': ['rbf'],
    'C': [10, 30, 50],
    'gamma': [0.001, 0.01]
]

```

S týmito parametrami sú výsledky nasledovné.

```

Fitting 3 folds for each of 6 candidates, totalling 18 fits
[CV 1/3] END C=10, gamma=0.001, kernel=rbf;, score=(train=0.097, test=0.055) total time= 18.4s
[CV 2/3] END C=10, gamma=0.001, kernel=rbf;, score=(train=0.097, test=0.052) total time= 21.1s
[CV 3/3] END C=10, gamma=0.001, kernel=rbf;, score=(train=0.097, test=0.053) total time= 21.5s
[CV 1/3] END C=10, gamma=0.01, kernel=rbf;, score=(train=0.389, test=0.034) total time= 24.3s
[CV 2/3] END C=10, gamma=0.01, kernel=rbf;, score=(train=0.400, test=0.034) total time= 20.8s
[CV 3/3] END C=10, gamma=0.01, kernel=rbf;, score=(train=0.400, test=0.037) total time= 22.3s
[CV 1/3] END C=30, gamma=0.001, kernel=rbf;, score=(train=0.112, test=0.053) total time= 23.5s
[CV 2/3] END C=30, gamma=0.001, kernel=rbf;, score=(train=0.114, test=0.052) total time= 22.5s
[CV 3/3] END C=30, gamma=0.001, kernel=rbf;, score=(train=0.112, test=0.050) total time= 29.5s
[CV 1/3] END C=30, gamma=0.01, kernel=rbf;, score=(train=0.590, test=0.029) total time= 28.1s
[CV 2/3] END C=30, gamma=0.01, kernel=rbf;, score=(train=0.602, test=0.033) total time= 22.5s
[CV 3/3] END C=30, gamma=0.01, kernel=rbf;, score=(train=0.599, test=0.031) total time= 27.2s
[CV 1/3] END C=50, gamma=0.001, kernel=rbf;, score=(train=0.121, test=0.051) total time= 29.6s
[CV 2/3] END C=50, gamma=0.001, kernel=rbf;, score=(train=0.125, test=0.049) total time= 23.9s
[CV 3/3] END C=50, gamma=0.001, kernel=rbf;, score=(train=0.122, test=0.047) total time= 24.0s
[CV 1/3] END C=50, gamma=0.01, kernel=rbf;, score=(train=0.694, test=0.028) total time= 35.7s
[CV 2/3] END C=50, gamma=0.01, kernel=rbf;, score=(train=0.701, test=0.032) total time= 40.5s
[CV 3/3] END C=50, gamma=0.01, kernel=rbf;, score=(train=0.698, test=0.028) total time= 24.7s
Best params: {'C': 10, 'gamma': 0.001, 'kernel': 'rbf'}
Best scores: 0.05357141345015292

```

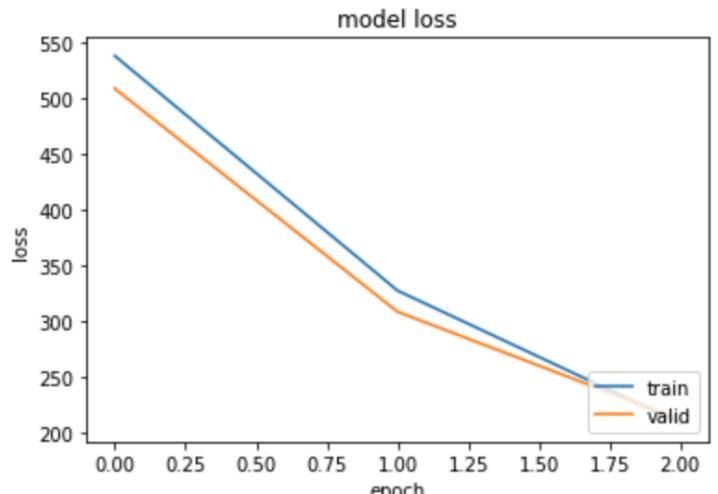
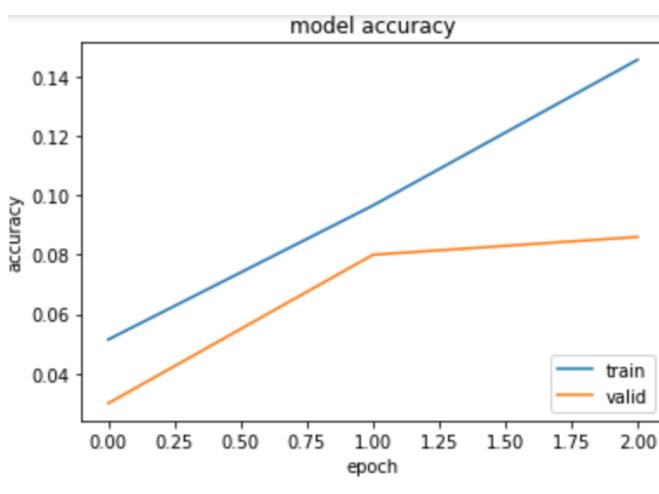
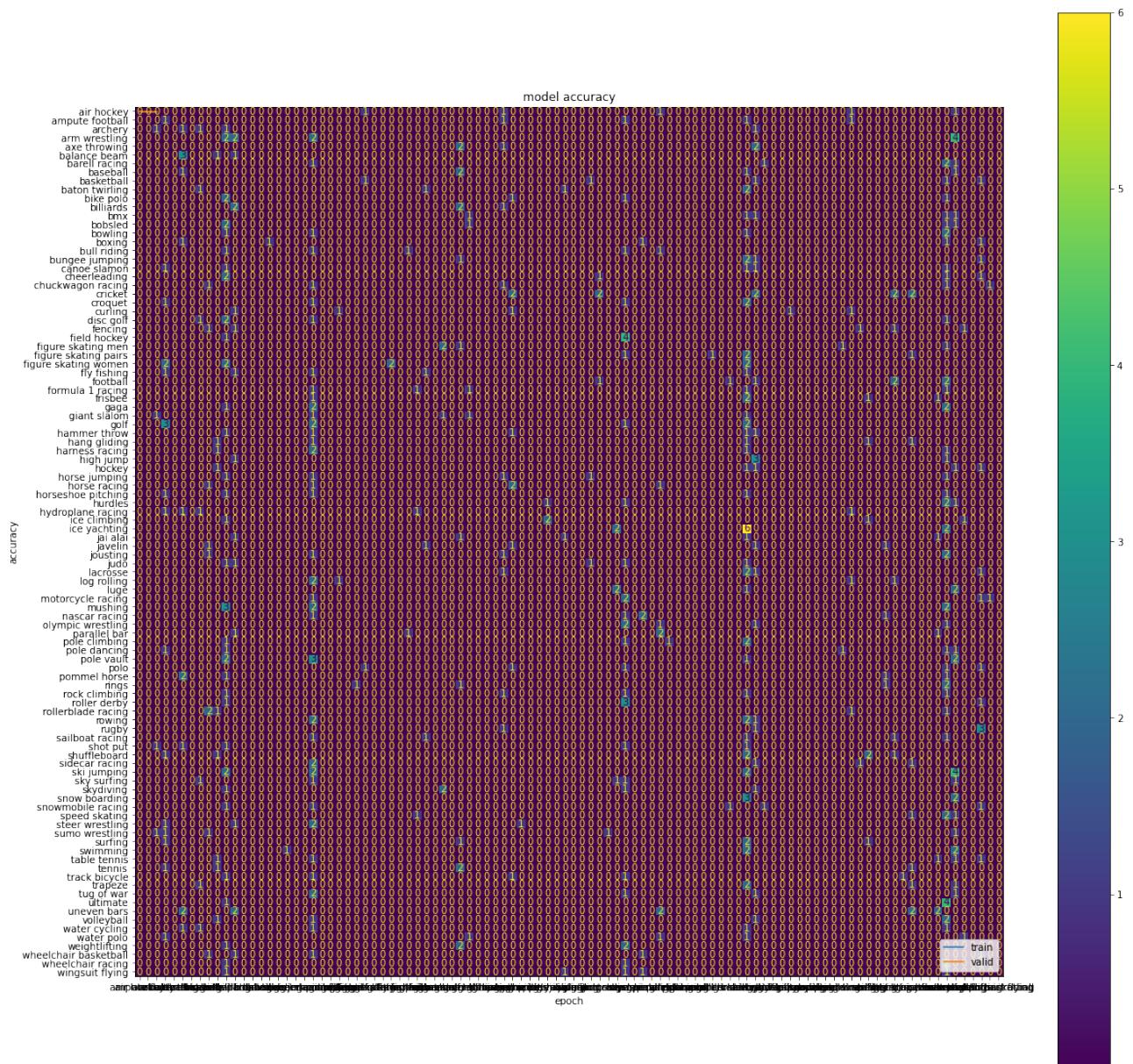
Skóre sa nám výrazne zlepšilo ale stále je veľmi nízke. Touto metódou sa ku lepším výsledkom pravdepodobne nedostaneme tak súšime použiť inú metódu.

Druhou metódou je dotrénovanie posledných vrstiev ImageNet siete. Vybrali sme si model InceptionV3. Na modeli musíme nastaviť aby sa pôvodné vrstvy už netrénovali. Tieto vrstvy doplníme našimimi vrstvami, ktoré sme už raz vytvárali v našej neurónovej siete. Použijeme 2 Dense vrstvy s filtrami 512 a 100. Na kompliaciu modelu použijeme optimalizátor Adam s hodnotou 0.001. Model dáme natrénovať na 3 epochy s EarlyStoppingom. Výsledky sú nasledovné.

```

Epoch 1/3
433/433 [=====] - 531s 1s/step - loss: 537.7802 - accuracy: 0.0515 - val_loss: 508.5220 - val_accuracy: 0.0300
Epoch 2/3
433/433 [=====] - 481s 1s/step - loss: 327.7117 - accuracy: 0.0966 - val_loss: 308.7995 - val_accuracy: 0.0800
Epoch 3/3
433/433 [=====] - 491s 1s/step - loss: 209.1081 - accuracy: 0.1457 - val_loss: 212.3066 - val_accuracy: 0.0860

```

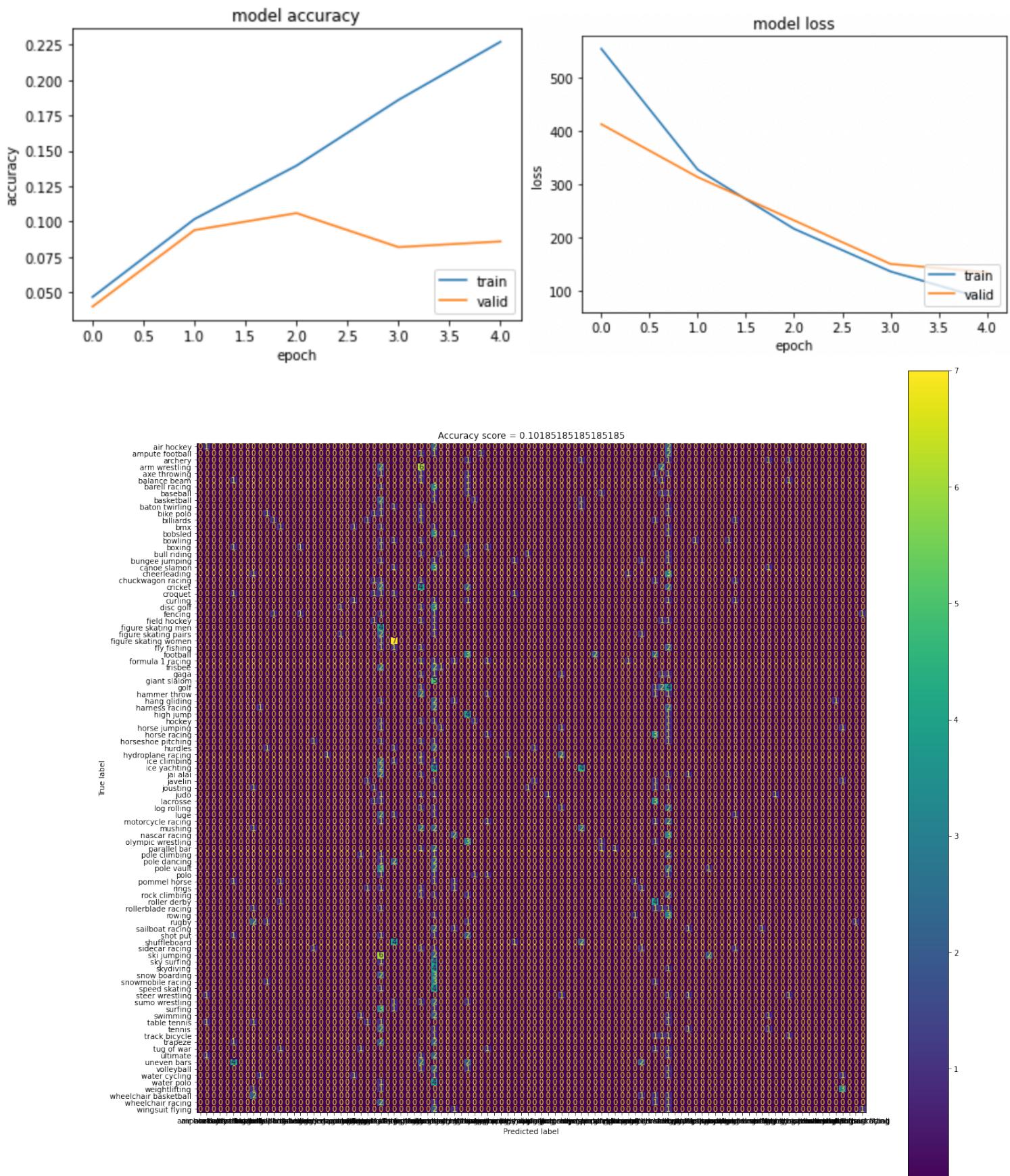


Presnosť siete je 8%. Skúsime ďalšie experimenty pre lepšie výsledky.
Zvýšime počet epoch.

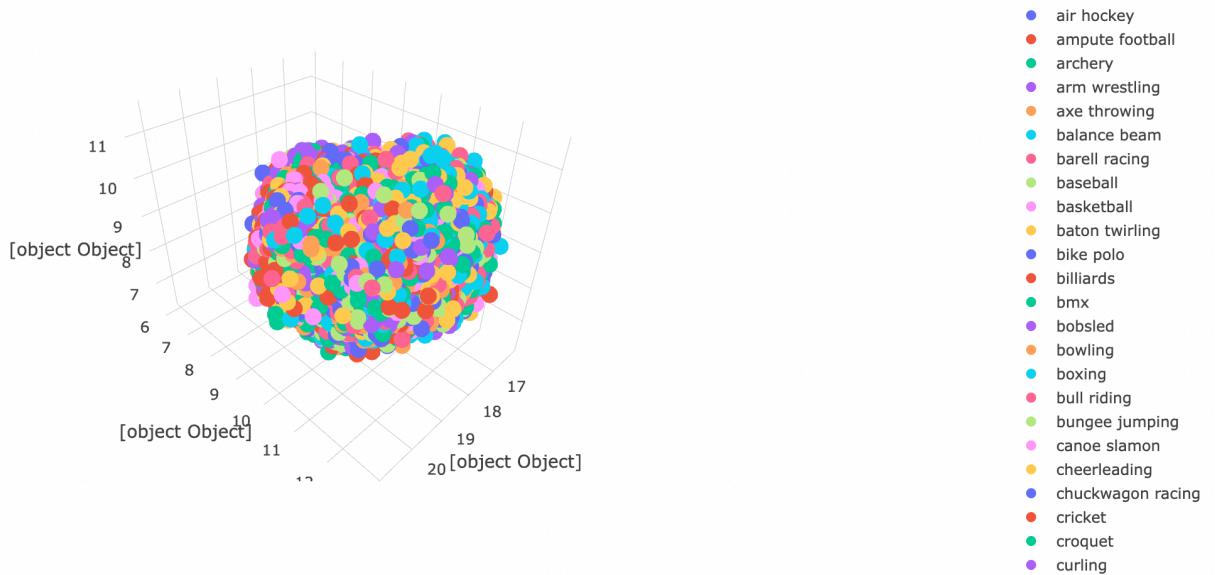
```

Epoch 1/5
433/433 [=====] - 490s/step - loss: 553.9572 - accuracy: 0.0468 - val_loss: 412.6864 - val_accuracy: 0.0400
Epoch 2/5
433/433 [=====] - 469s/step - loss: 327.6653 - accuracy: 0.1018 - val_loss: 313.5787 - val_accuracy: 0.0940
Epoch 3/5
433/433 [=====] - 479s/step - loss: 216.7502 - accuracy: 0.1395 - val_loss: 232.5375 - val_accuracy: 0.1060
Epoch 4/5
433/433 [=====] - 503s/step - loss: 136.9488 - accuracy: 0.1859 - val_loss: 150.9838 - val_accuracy: 0.0820
Epoch 5/5
433/433 [=====] - 470s/step - loss: 84.3935 - accuracy: 0.2269 - val_loss: 135.4590 - val_accuracy: 0.0860

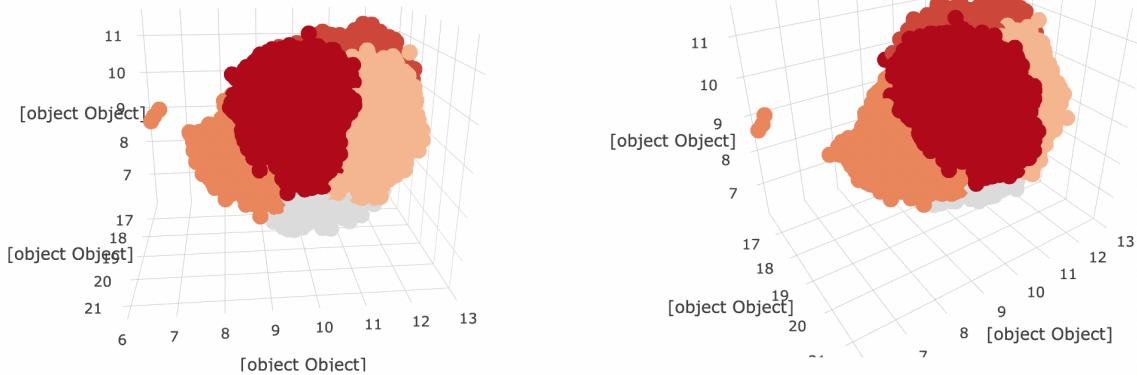
```



Ako posledné si predstavíme zhľukovanie dát do 3D grafu.



Na ukážke môžeme vidieť všetký kategórie z trénovacej množiny. Každú kategóriu reprezentuje farba. Skúsime si množinu kategorizovať pomocou clusteringu.



Ako môžeme vidieť množiny sa pogrupovali do 6 farieb. Každá farba reprezentuje jednu skupinu kategórií podľa ich vlastností.