PANEURÓPSKA VYSOKÁ ŠKOLA

Fakulta informatiky

Semestrálne zadanie 2

Opice – Deep Learning

PhDr. Ing. Mgr. Miroslav Reiter, DBA

Kontroling

prof. Ing. Štefan Kozák, PhD.

26.12.2022

Obsah

1. Produktové požiadavky (Znenie úlohy)

Vytvorte klasifikačnú neurónovú sieť, ktorá bude rozoznávať, aký objekt sa nachádza na obrázku. Spustite niekoľko rôznych tréningov a porovnajte ich. Aké máte najlepšie výsledky? Aké ste skúsili modely?

Zodpovedajte nasledovné otázky podľa príslušných častí:

- Definícia datasetu
    - Čo obsahuje dataset?
    - Aké kategórie predikujeme?
    - Koľko máme vstupov pre každú kategóriu?
    - Aké je rozdelenie datasetu na jednotlivé subsety?
- Experimenty
    - Aké nastavenia modelu sme využili?
    - Aké výsledky dosahuje dané nastavenie?
    - Ako vyzerajú jednotlivé krivky počas tréningu?
- Výber modelu
    - Prečo ste vybrali práve tento model?
    - Ako vyzerá jeho chybová funkcia a jeho metrika (accuracy/MCC)?
- Výsledky vybraného modelu
    - V akej epoche a s akou hodnotou sledovanej metriky sme získali najlepší model?
    - Odpredikujte testovací dataset a uložte výsledky do tabuľky.
    - **Bonus (3b)**: Vykreslite confusion matrix pre testovací dataset (využite tabuľku z predošlého bodu) a interpretujte

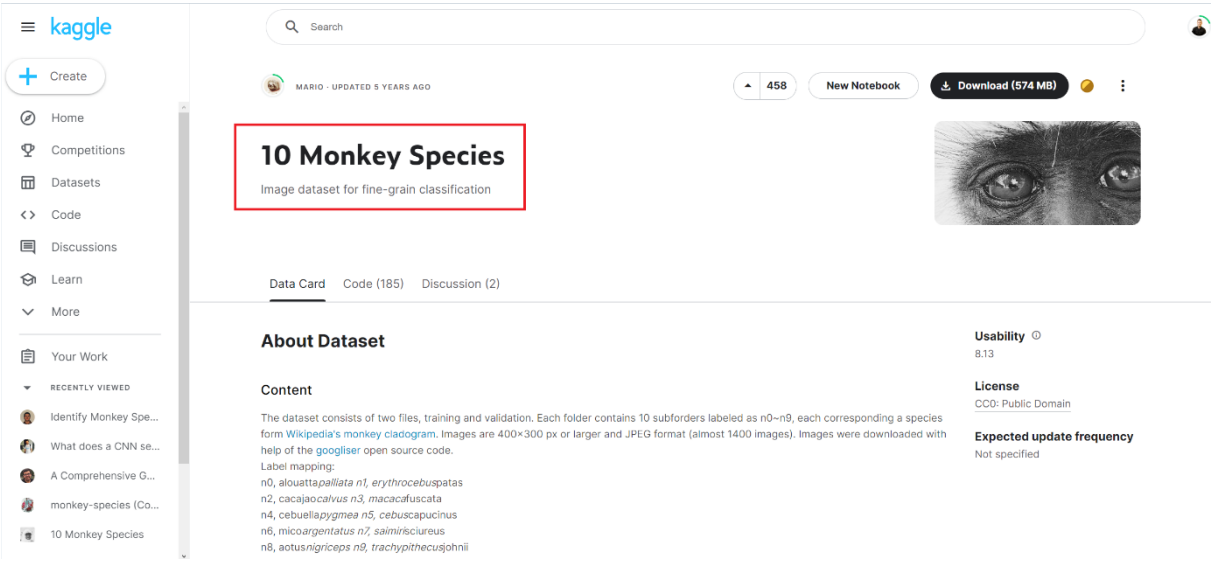
Odovzdajte zazipované: **zdrojový kód** (bez datasetu), **logy** z tréningov (bez natrénovaného modelu) a **dokumentáciu**.

Štruktúra priečinkov:

- logs
    - exp_001
    - exp_002
    - ...

2. Definícia datasetu

V rámci zadaní sme si vybrali rovno prvý dataset a to opice (10 monkey species). Uvedené zadanie sme si vybrali z osobných dôvodov, pretože manželka je v znamení opice a má rada opičky. Má ešte radšej psíky (máme doma 3 pomeranian špice), ale to sme už robili na cvičenia, takže bolo rozhodnuté. Tento dataset má podľa Kaggle použitelnosť (Usability) 8,13.



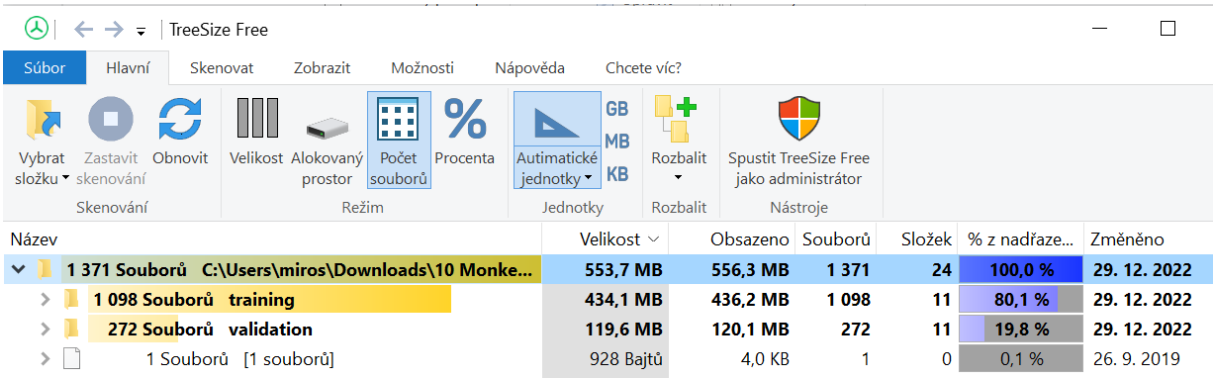
Obrázok 1 *Vybraný Dataset 10 Monkey Species (Opice) z Kaggle*

Kompletný zdrojový kód je k dispozícii v prílohe alebo na GitHube repozitári: https://github.com/miroslav-reiter/PEVS-PANI-Kontrolling

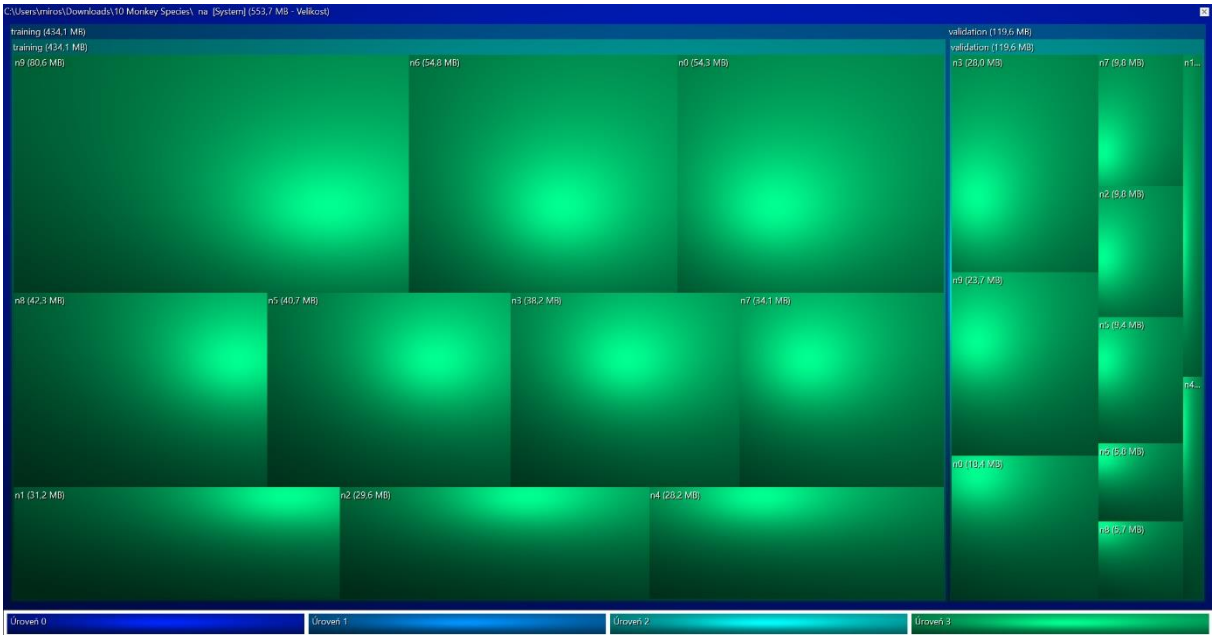
V tejto časti sme zodpovedali na nasledovné otázky (obsah niektorých odpovedí často zodpovedá viacerým otázkam):

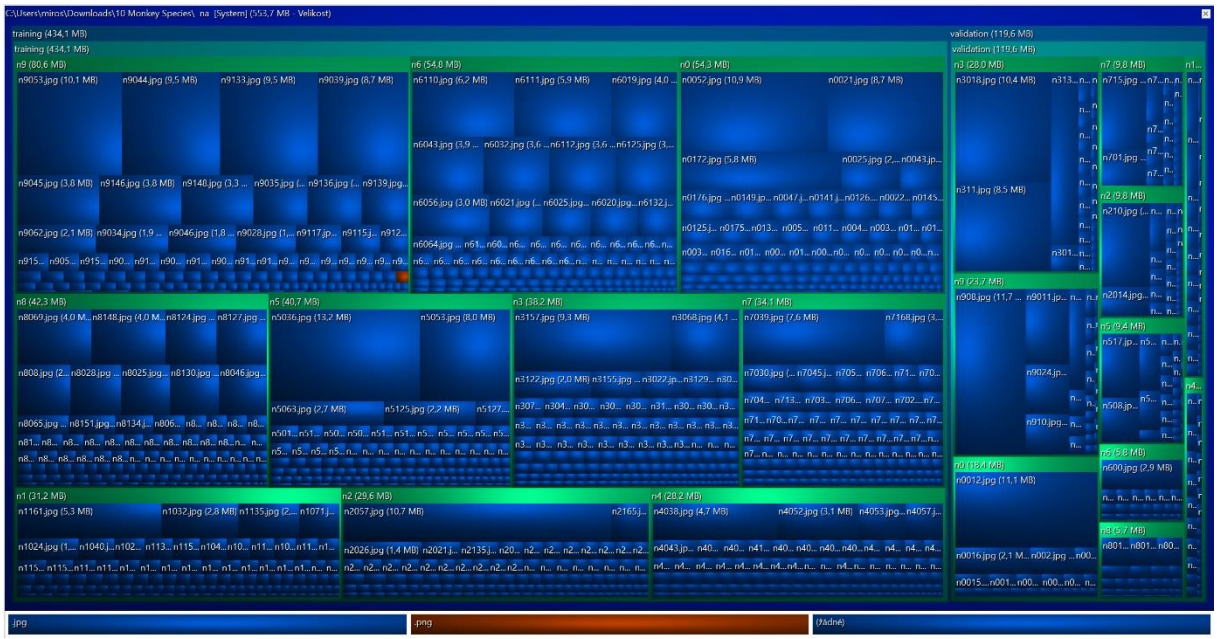
A. Čo obsahuje dataset?

Dataset o veľkosti 547 MB (v ZIP formáte s 98 % kompresným pomerom), po rozbalení 553 MB a 1 371 súborov, 24 priečinkov. Dataset pozostáva z 2 priečinkov, tréningového (training) a validačného (validation). Každý priečinok obsahuje 10 podpriečinkov označených ako n0 až n9, pričom každý zodpovedá druhom opíc. Obrázky majú veľkosť 400x300 px alebo viac a majú formát JPEG (takmer 1400 obrázkov). K dispozícii je aj textový súbor (monkey_labels), ktorý obsahuje menovky druhov/kategórií/subsety opíc, latinské názvy, počet trénovacích aj validačných obrázkov. Dataset je podľa popisu z Kaggle určený ako testovací prípad pre úlohy klasifikácie.



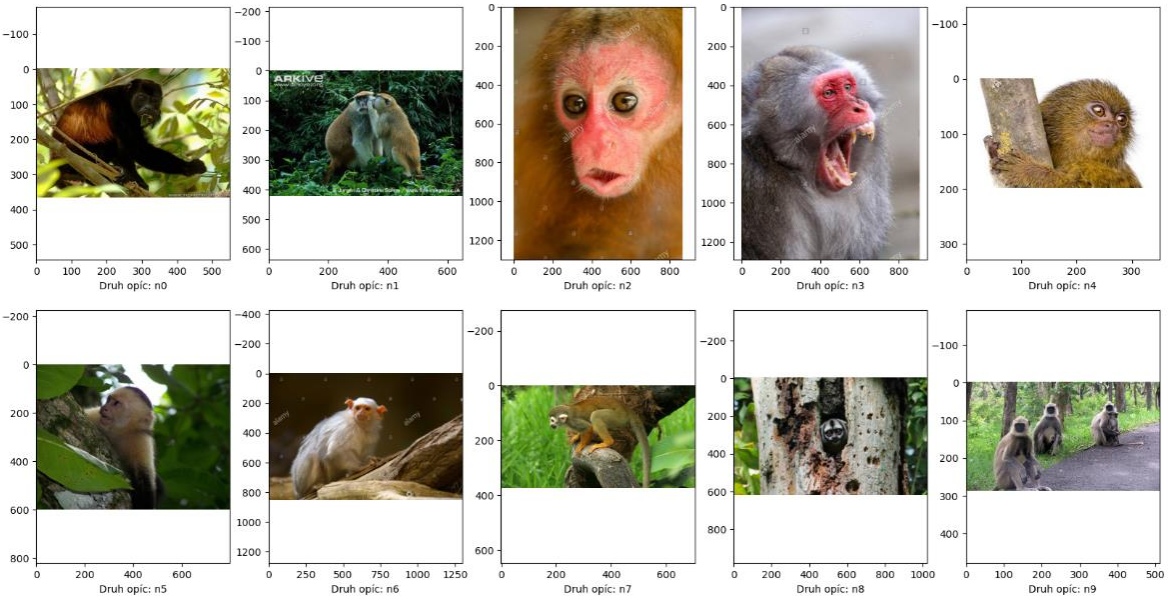
Obrázok 2 Obsah Datasetu





```
Rozmer obrazka (data/train/n0\n0018.jpg) je: (367, 550, 3) px
```


```
Dáta/obrázky z každého druhu opice
```



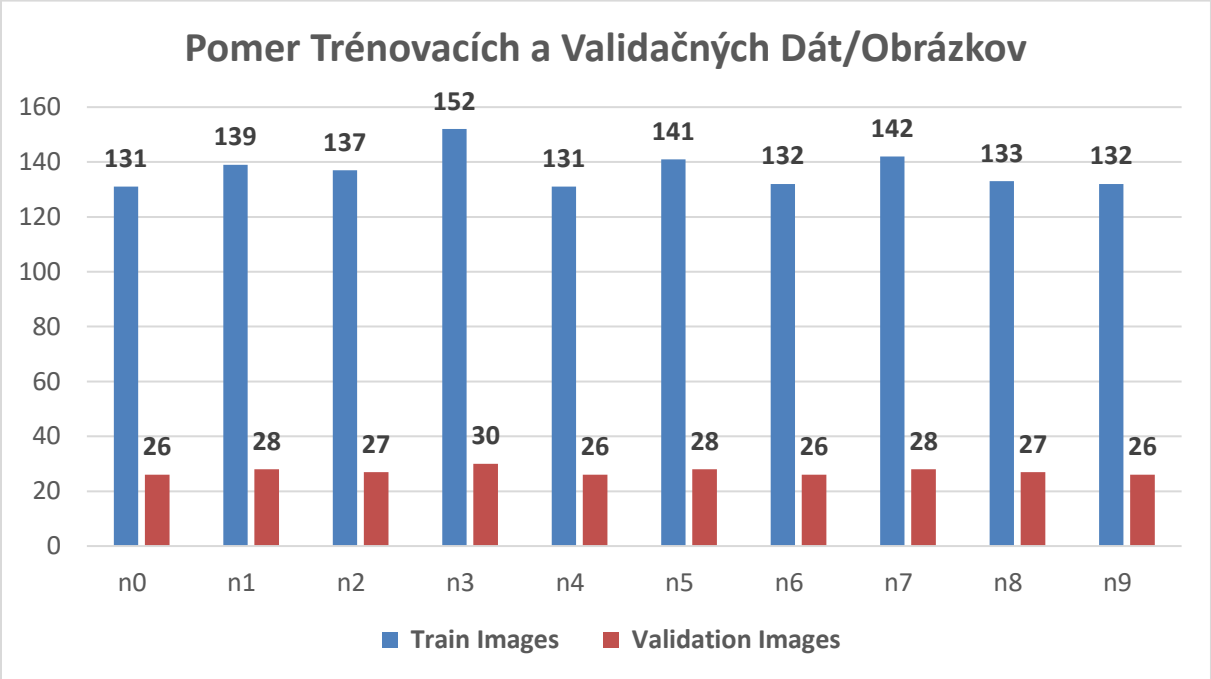
B. Aké kategórie predikujeme?

Každý priečinok obsahuje 10 podpriečinkov označených ako n0 až n9, pričom každý zodpovedá druhom opíc. Obrázky majú veľkosť 400x300 px alebo viac a majú formát JPEG (1369 obrázkov) a 1 obrázok má formát PNG. K dispozícii je aj textový súbor (monkey_labels), ktorý obsahuje menovky druhov/kategórií/subsety opíc, latinské názvy, počet trénovacích aj validačných obrázkov.

Label	Latin Name	Common Name	SK
n0	alouatta_palliata	mantled_howler	Vrešťan Pláštikový
n1	erythrocebus_patas	patas_monkey	Mačiak Husársky
n2	cacajao_calvus	bald_uakari	Uakari Šarlátovolíci
n3	macaca_fuscata	japanese_macaque	Makak Japonský
n4	cebuella_pygmea	pygmy_marmoset	Kosmáč Trpasličí
n5	cebus_capucinus	white_headed_capuchin	Panamský Kapucín Bielolíci
n6	mico_argentatus	silvery_marmoset	Kosmáč Striebristý
n7	saimiri_sciureus	common_squirrel_monkey	Saimiri Vevericovitý
n8	aotus_nigriceps	black_headed_night_monkey	Nočná Opica Čiernohlavá
n9	trachypithecus_johnii	nilgiri_langur	Hulman Nilgirský
Počet Kategórií	**10**		

C. Koľko máme vstupov pre každú kategóriu?

Label	Latin Name	Common Name	Train Images	Validation Images
n0	alouatta_palliata	mantled_howler	131	26
n1	erythrocebus_patas	patas_monkey	139	28
n2	cacajao_calvus	bald_uakari	137	27
n3	macaca_fuscata	japanese_macaque	152	30
n4	cebuella_pygmea	pygmy_marmoset	131	26
n5	cebus_capucinus	white_headed_capuchin	141	28
n6	mico_argentatus	silvery_marmoset	132	26
n7	saimiri_sciureus	common_squirrel_monkey	142	28
n8	aotus_nigriceps	black_headed_night_monkey	133	27
n9	trachypithecus_johnii	nilgiri_langur	132	26
Počet Kategórií	**10**	**Súčet**	**1370**	**272**
		Priemer	**137**	**27,2**
		Minimum	**131**	**26**
		Maximum	**152**	**30**



Deskriptívna štatistika nad datasetom Opice

8 rows | 8 rows × 2 columns

	Train_Images	Validation_Images
count	10.000000	10.000000
mean	137.000000	27.200000
std	6.733003	1.316561
min	131.000000	26.000000
25%	132.000000	26.000000
50%	135.000000	27.000000
75%	140.500000	28.000000
max	152.000000	30.000000

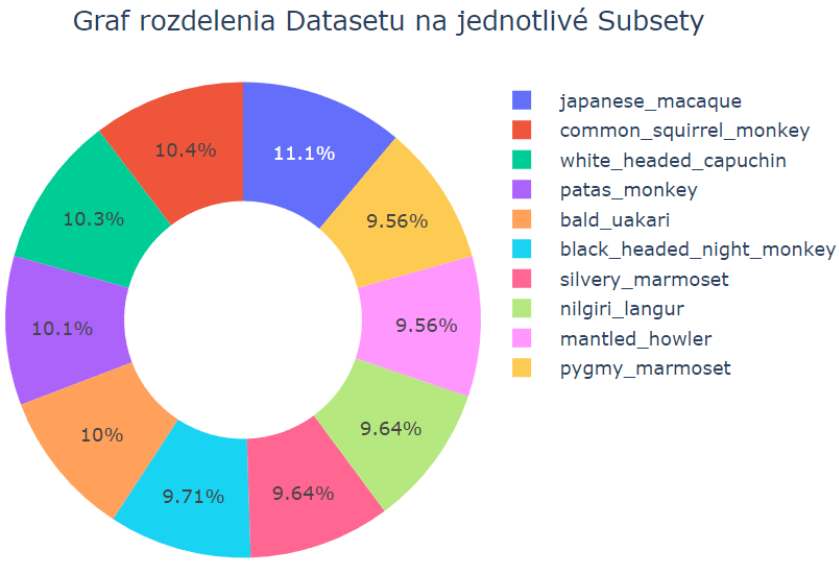
11 rows | 11 rows × 5 columns

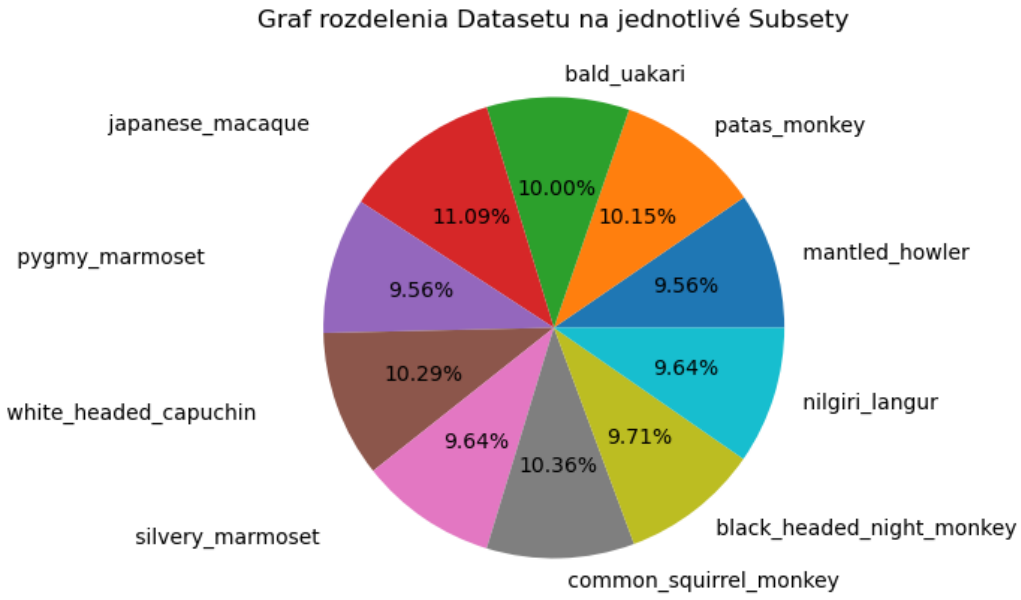
	Label	Latin_Name	Common_Name	Train_Images	Validation_Images
count	10	10	10	10.000000	10.000000
unique	10	10	10	NaN	NaN
top	n0	alouatta_palliata\t	mantled_howler	NaN	NaN
freq	1	1	1	NaN	NaN
mean	NaN	NaN	NaN	137.000000	27.200000
std	NaN	NaN	NaN	6.733003	1.316561
min	NaN	NaN	NaN	131.000000	26.000000
25%	NaN	NaN	NaN	132.000000	26.000000
50%	NaN	NaN	NaN	135.000000	27.000000
75%	NaN	NaN	NaN	140.500000	28.000000
max	NaN	NaN	NaN	152.000000	30.000000

D. Aké je rozdelenie datasetu na jednotlivé subsety?

Na vizualizáciou rozdelenia sme použili knižnicu matplolib aj plotly express. Keďže máme 10 kategórií, tak vidíme pomerne rovnomerne rozdelené jednotlivé subsety od 9,56 % (mantled_howler - Vrešťany Pláštikovité) až po 11,1 % (japanese_macaque - japonské makaky). Priemerná hodnota rozdelenia je teda 10 %.

Label	Latin Name	Common Name	Train Images	Validation Images	% Train Images	% Validation Images
n0	alouatta_palliata	mantled_howler	131	26	9,56%	9,56%
n1	erythrocebus_patas	patas_monkey	139	28	10,15%	10,29%
n2	cacajao_calvus	bald_uakari	137	27	10,00%	9,93%
n3	macaca_fuscata	japanese_macaque	152	30	11,09%	11,03%
n4	cebuella_pygmea	pygmy_marmoset	131	26	9,56%	9,56%
n5	cebus_capucinus	white_headed_capuchin	141	28	10,29%	10,29%
n6	mico_argentatus	silvery_marmoset	132	26	9,64%	9,56%
n7	saimiri_sciureus	common_squirrel_monkey	142	28	10,36%	10,29%
n8	aotus_nigriceps	black_headed_night_monkey	133	27	9,71%	9,93%
n9	trachypithecus_johnii	nilgiri_langur	132	26	9,64%	9,56%
Počet Kategórií	**10**	**Súčet, Priemer**	**1370**	**272**	**10,00%**	**10,00%**



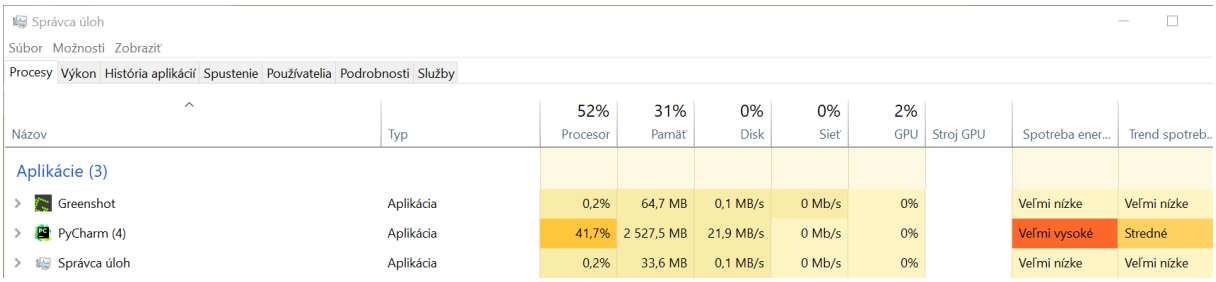


Pre klasifikáciu obrázkov sme využili modul ImageDataGeneratora z TensorFlow. Zdrojový kód je priamo dostupný na stránke s oficiálnou dokumentáciou (https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator) a súčasne aj na Kaggle. ImageDataGeneratora generuje dávky obrazových dát s rozšírením dát v reálnom čase. Skript odhalil, že:

Found 1098 images belonging to 10 classes. (generátor našiel 1098 obrázkov, ktoré prislúchajú 10 triedam)
Found 272 images belonging to 10 classes. generátor našiel 272 obrázkov, ktoré prislúchajú 10 triedam)

V oficiálnej dokumentácii je síce označený ako deprecated t.j. zastaralé riešenie, ale väčšina autorov aj na cvičení využívalo práve ImageDataGenerator. V mojom Jupyter notebooku som do ImageDataGeneratora zvolil ako velkosť dávky (batch_size) t.j. počet obrázkov, ktoré dostane neuronová sieť naraz na 32 pri experimente. Zvolil som teda dvojnásobnú hodnotu oproti cvičeniam ("batch_size": 16). V skripte z cvičenia som ponechal pri prvých 3 experimentoch predvolenú hodnotu batch_size na 16.

Pri spustení skriptov na notebooku s 63,8 GB RAM a Intel Core i9-10980HK (16 CPUs) bola costs (réžia) 2 527,5 MB RAM a 52 % zaťaženie CPU.



Čo ak:

- dataset má veľké množstvo kategórií, použite max 10.

  ✔ **Dataset má presne 10 kategórií (n0 až n9)**

- dataset, ktorý ste si vybrali neobsahuje testovaciu zložku, použite validačný set.

  ✔ **Tento dataset neobsahuje testovaciu zložku. Vytvorili sme ju z validačného setu.**

- dataset nemá validačnú zložku, použite testovaciu.

  ✘ **Dataset má validačnú zložku.**

- dataset nie je rozdelený, rozdeľte si ho sami.

  ✘ **Dataset je rozdelený, nebolo ho potrebné rozdeľovať.**

Dataset keďže je vyvážený budeme používať accuracy ako metriku, ktorú si budeme sledovať a vykresľovať. Nepoužívame teda MCC metriku.

3. Experimenty

Načítali sme dataset rôznych druhov opíc (obrázky JPG a PNG). Neurónová sieť zoberie obrázky a naučila sa, o aký príslušný typ opice ide. Neurónová sieť pozostávala z rôznych konvolučných blokov a pozostáva aj z obrázkových generátorov (ImageDataGeneratora). Natrénovali sme takúto neurónovú sieť potom sme s tou neurónovou sieť robili predikciu (úspešnú) nad testovacím datasetom (validačného setu).

Vo vzorom súbore z cvičení sme už mali vytvorený automatizovaný test pomocou for cyklov a automatickej tvorby priečinkov pre logy, ktorý vytvára nasledovnú štruktúru priečinkov:

- logs
    - exp_000
    - exp_001
    - exp_002
    - ...

- **Aké nastavenia modelu sme využili?**

Na vstupe (vstupnej vrstve) máme RGB obrázky o veľkosti 128 x 128 px, 3 kanály. Veľkosť trénovacích vzoriek je 1098 a počet validačných vzoriek je 277. Rozdelenie je teda 80 % trénovacie vzorky a 20 % validačné t.j. paretovo pravidlo 80/20. Epochy sme si nastavili na počet 50.

Celkový počet vzoriek:	**1370**	**100%**
Počet trénovacích vzoriek:	1098	80%
Počet validačných vzoriek:	**272**	**20%**

Pre prvé 3 experimenty sme ponechali nastavenia, ktoré sme si demonštrovali na cvičenia t.j. spoločné vlastnosti pre všetky 3 experimenty sú:

- Počet konvolučných blokov na 2,
- Využívame optimizer Adam,
- Veľkosť dávky (batch_size) je 16
- dropout má hodnotu 0.2 (20 % prepojení nám náhodne vypadne, dropoutom sa vynulujú tie prepojenie, nevymažú sa vyslovene).

Experimenty sa líšia v:

- Rôzne veľkosti filtrov (filter_size) 3,3 a 5,5
- Rôzne is_max_pooling exp_000 použije maxpooling exp_001 použije averagepooling
- Exp002 má is_max_pooling false oproti Exp000

Experiment 000

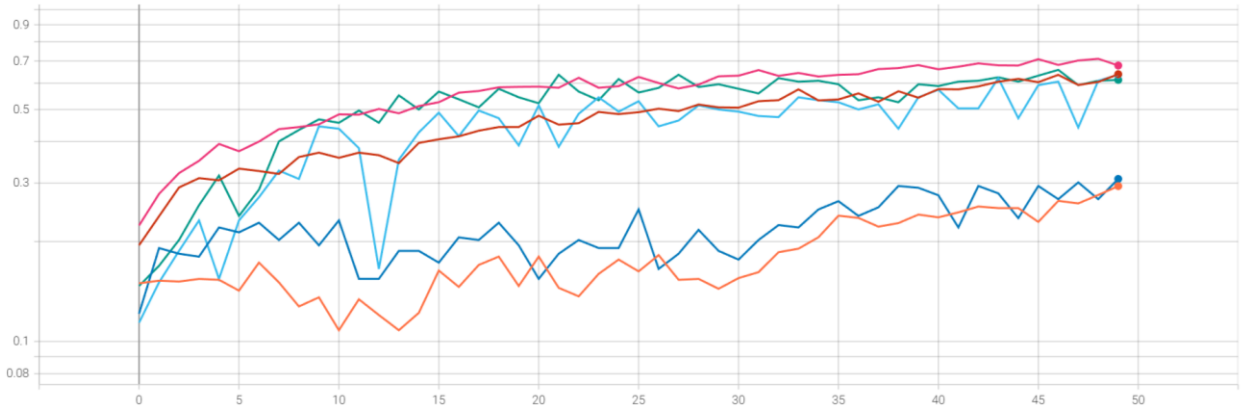
```
{
    # Nazov experimentu
    "name": "exp_000",
    # Počet konvolučných blokov
    "conv_num": 2,
    # V rámci konvolučných vrstviev, používame rôzne typy filtrov
    "filter_num": 32,
    # Rôzne velkosti filtrov
    "filter_size": (3, 3),
    # Veľkosť max_poolingu, aby sme redukovali výsledné matematické operácie, ktoré prebehli cez konvolúcie
    # a potom výsledné matice, akým spôsobom sa zgrupovali a zmenšovali, aby sa neuronová sieť rýchlejšie natrénovala
    # a lepšie rozpoznávala jednotlivé oblasti
    "max_pooling": (2, 2),
    # Regularizačná technika, pri prepojených/plne prepojených vrstvách bude náhodne dávať váhu
    # jednotlivých prepojení na 0, čiže ako keby ich vypol
    # Regularizačný prvok, vďaka týmto výpadkom je neuronová sieť vyslovene nútená hladať prepojenia vo vzoroch
    # Môže tam byť nejaký šum
    # 0.2 znamená, že 20 % prepojení nám náhodne vypadne, dropoutom sa vynulujú tie prepojenie, nevymažú sa vyslovene
    "dropout": 0.2,
    # Optimalizátor, najzákladnejší
    "optimizer": "adam",
    # Veľkosť dávky, v ktorom neuronová sieť bude jednotlivé informácie vyhodnocovať
    # na koľko obrázkov sa naraz pozrie než sa aktualizuje
    "batch_size": 16,
    # Ak je True využite sa funkcia maxpooling, ak je False využije sa averagepooling
    "is_max_pooling": True,
}
```

Experiment 001 a 002

```
# Rozdiel medzi exp_000, exp_001 a exp_002:
# - Rôzne velkosti filtrov (filter_size) 3,3 a 5,5
# - Rôzne is_max_pooling exp_000 použije maxpooling exp_001 použije averagepooling
{
    "name": "exp_001",
    "conv_num": 2,
    "filter_num": 32,
    "filter_size": (5, 5),
    "max_pooling": (2, 2),
    "dropout": 0.2,
    "optimizer": "adam",
    "batch_size": 16,
    "is_max_pooling": False,
},
{
    "name": "exp_002",
    "conv_num": 2,
    "filter_num": 32,
    "filter_size": (3, 3),
    "max_pooling": (2, 2),
    "dropout": 0.2,
    "optimizer": "adam",
    "batch_size": 16,
    "is_max_pooling": False,
}
```

- Aké výsledky dosahuje dané nastavenie? a Ako vyzerajú jednotlivé krivky počas tréningu?

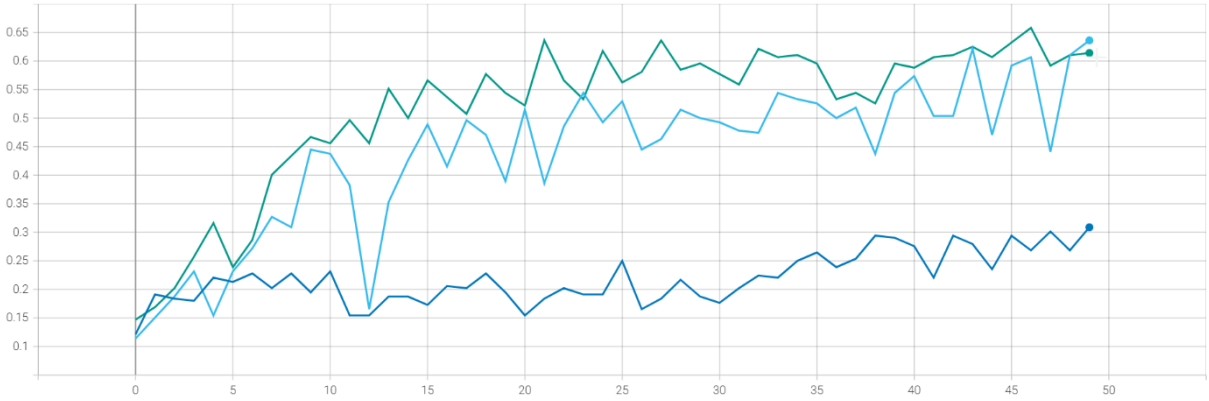
  V TensorBoarde sme zaznamenávali postupný vývoj a trénovanie algoritmov. Celkový trend accuracy sa nám zvyšoval, čo je dobré.



Obrázok 3 *Graf s krivkami epoch_accuracy pre experimenty 000, 001 a 002*

Name	Smoothed	Value	Step	Time	Relative
exp_000\train	0.2939	0.2939	49	Fri Dec 30, 14:37:59	42m 49s
exp_000\validation	0.3088	0.3088	49	Fri Dec 30, 14:37:59	42m 49s
exp_001\train	0.6386	0.6386	49	Fri Dec 30, 15:15:31	36m 38s
exp_001\validation	0.636	0.636	49	Fri Dec 30, 15:15:31	36m 38s
exp_002\train	0.6784	0.6784	49	Fri Dec 30, 15:49:45	33m 30s
exp_002\validation	0.614	0.614	49	Fri Dec 30, 15:49:45	33m 30s

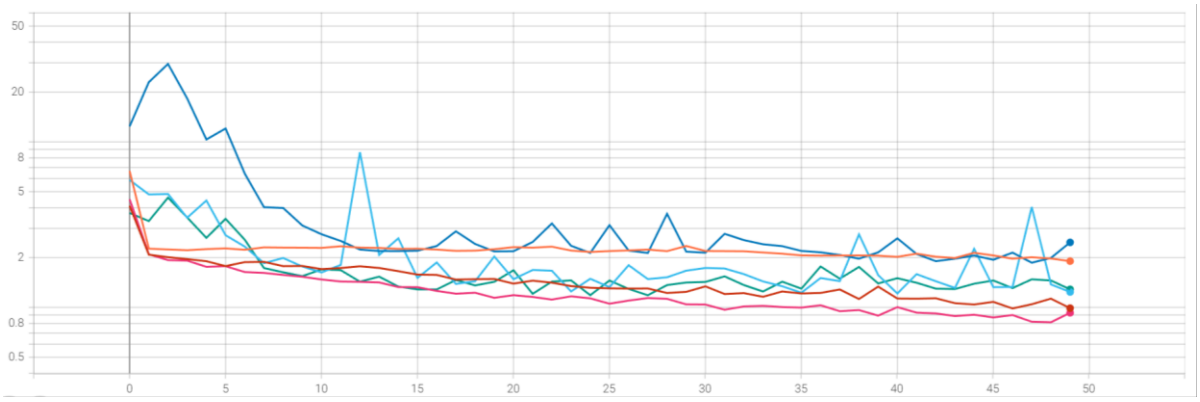
Vyfiltrovali sme si krivky s validáciou a vybrali ako **najlepší experiment 002**, pretože má najhladšiu, čo najrýchlejšie stúpajúcu a čo najvyššie stúpajúcu krivku (zelená) aj so smoothingom, aj bez neho.



Obrázok 4 *Graf s krivkami epoch_accuracy pre experimenty filter validation*

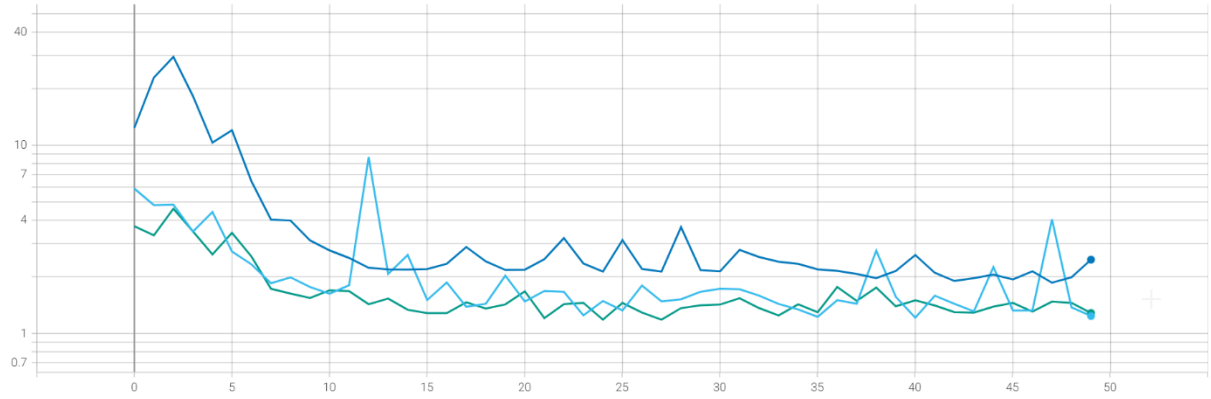
Name	Smoothed	Value	Step	Time	Relative
exp_000\validation	0.3088	0.3088	49	Fri Dec 30, 14:37:59	42m 49s
exp_001\validation	0.636	0.636	49	Fri Dec 30, 15:15:31	36m 38s
exp_002\validation	0.614	0.614	49	Fri Dec 30, 15:49:45	33m 30s

To isté vidíme aj na samotnej loss funkcií, tá postupne klesá a to je presne čo chceme.



Obrázok 5 *Graf s krivkami epoch_lost pre experimenty 000, 001 a 002*

Name	Smoothed	Value	Step	Time	Relative
exp_000\train	6.674	6.674	0	Fri Dec 30, 13:55:10	0s
exp_000\validation	12.39	12.39	0	Fri Dec 30, 13:55:10	0s
exp_001\train	4.108	4.108	0	Fri Dec 30, 14:38:53	0s
exp_001\validation	5.887	5.887	0	Fri Dec 30, 14:38:53	0s
exp_002\train	4.5	4.5	0	Fri Dec 30, 15:16:14	0s
exp_002\validation	3.718	3.718	0	Fri Dec 30, 15:16:14	0s



Vyfiltrovali sme si krivky s validáciou a vybrali ako **najlepší experiment 001**, pretože má, čo najlepšie klesajúcu a čo najnižšie položenu krivku (zelená) aj so smoothingom, aj bez neho.

Obrázok 6 *Graf s krivkami epoch_lost pre experimenty filter validation*

Name	Smoothed	Value	Step	Time	Relative
exp_000\validation	2.469	2.469	49	Fri Dec 30, 14:37:59	42m 49s
exp_001\validation	1.238	1.238	49	Fri Dec 30, 15:15:31	36m 38s
exp_002\validation	1.285	1.285	49	Fri Dec 30, 15:49:45	33m 30s

2. várka experimentov

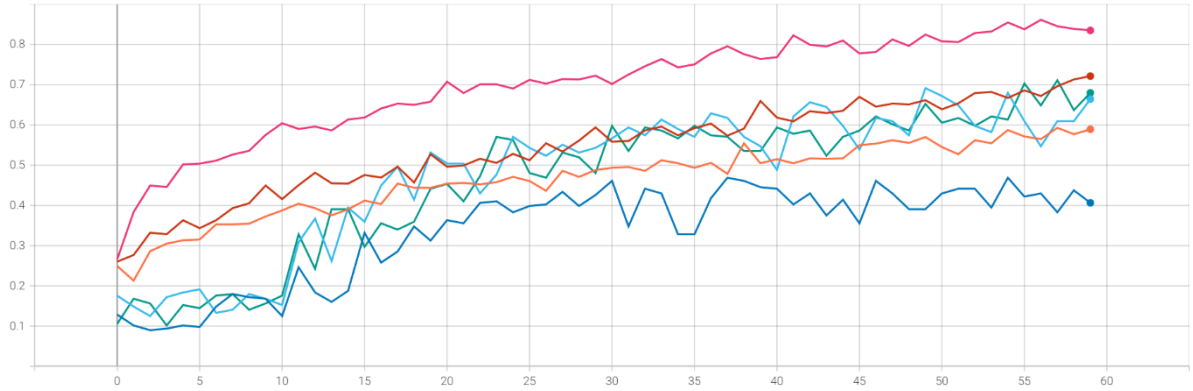
Na základe odporúčaní profesora Kozáka a z cvičení sme v ďalších experimentoch zvýšili počet epoch z 50 na 60. Predpoklad je, že by sme mali dosiahnuť lepšie výsledky vyššie hodnoty accurace a nižšie hodnoty loss. Samozrejme treba dať pozor, aby sme model zbytočne nepretrénovali. Ďalej sme sa rozhodli pre zväčšenie batch_size na 32 obrázkov, vďaka tomu sme nemali priebehy jednotlivých tréningov príliš rozkmitané, ako bolo aj prezentované na cvičení. Rovnako sme zväčšili veľkosť dropout a v ďalších experimentoch testovali dropout 0,2; 0,3 a 0,5. Podľa odporúčaní je ideálny dropout 0,5 - 0,8. Predpoklad ohľadom lepších výsledkov v 2. várke experimentov sa nám potvrdil.

```
{
    "name": "exp_000",
    "conv_num": 2,
    "filter_num": 32,
    "filter_size": (3, 3),
    "max_pooling": (2, 2),
    "dropout": 0.5,
    "optimizer": "adam",
    "batch_size": 32,
    "is_max_pooling": False,
},
{
    "name": "exp_001",
    "conv_num": 2,
    "filter_num": 32,
    "filter_size": (5, 5),
    "max_pooling": (2, 2),
    "dropout": 0.3,
    "optimizer": "adam",
    "batch_size": 32,
    "is_max_pooling": False,
},
{
    "name": "exp_002",
    "conv_num": 2,
    "filter_num": 32,
    "filter_size": (3, 3),
    "max_pooling": (2, 2),
    "dropout": 0.2,
    "optimizer": "adam",
    "batch_size": 32,
    "is_max_pooling": False,
}
```

***Obrázok 7 2. várka experimentov**, batch_size=32, epochy=60, rôzne dropouty*

2. várka experimentov

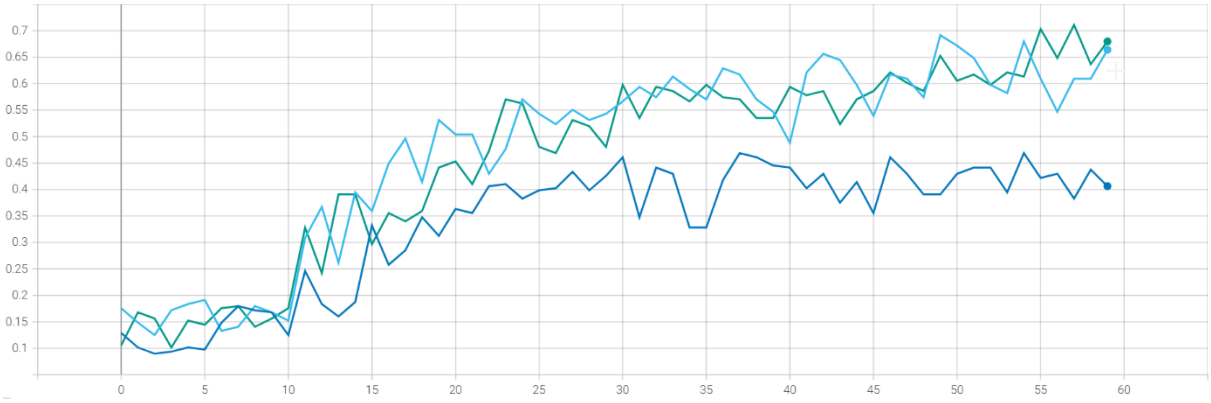
Opäť sa nám celkový trend accuracy zvyšoval, čo je dobré.



Obrázok 8 *Graf s krivkami epoch_accuracy pre experimenty 000, 001 a 002* ***2. várka***

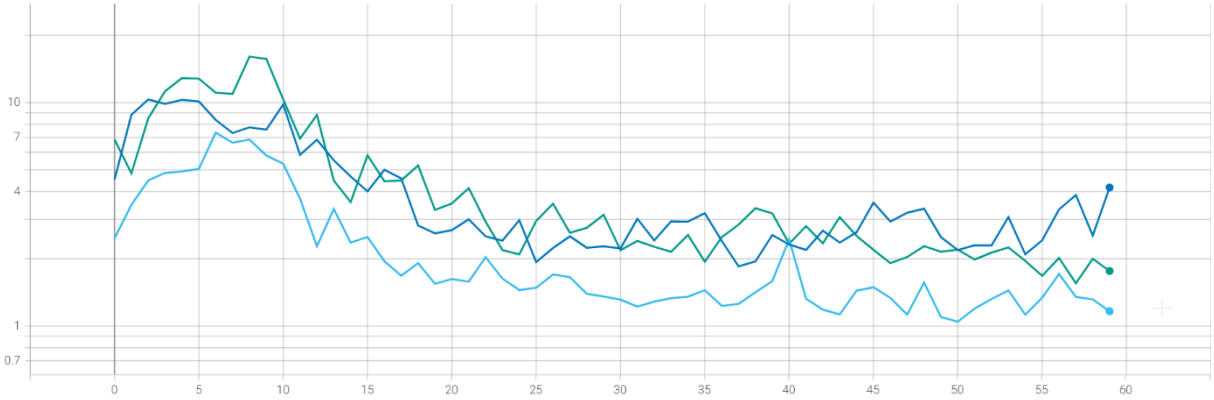
Name	Smoothed	Value	Step	Time	Relative
exp_000\train	0.5891	0.5891	59	Fri Dec 30, 18:40:29	37m 38s
exp_000\validation	0.4063	0.4063	59	Fri Dec 30, 18:40:29	37m 38s
exp_001\train	0.7214	0.7214	59	Fri Dec 30, 19:23:23	41m 58s
exp_001\validation	0.6641	0.6641	59	Fri Dec 30, 19:23:23	41m 58s
exp_002\train	0.8349	0.8349	59	Fri Dec 30, 20:03:59	39m 56s
exp_002\validation	0.6797	0.6797	59	Fri Dec 30, 20:03:59	39m 56s

Vyfiltrovali sme si krivky s validáciou a tentokrát vybrali ako **najlepší experiment 001**, pretože aj keď má o niečo nižšiu hodnoty ako experiment 002 je tento rozdiel zanedbateľný, ale hlavne má lepšiu loss funkciu. Pri smootingu má experiment 001 má najhladšiu, čo najrýchlejšie stúpajúcu a čo najvyššie stúpajúcu krivku (modrá).



Obrázok 9 *Graf s krivkami epoch_accuracy pre experimenty filter validation* ***2 várka***

Name	Smoothed	Value	Step	Time	Relative
exp_000\validation	0.4063	0.4063	59	Fri Dec 30, 18:40:29	37m 38s
exp_001\validation	0.6641	0.6641	59	Fri Dec 30, 19:23:23	41m 58s
exp_002\validation	0.6797	0.6797	59	Fri Dec 30, 20:03:59	39m 56s

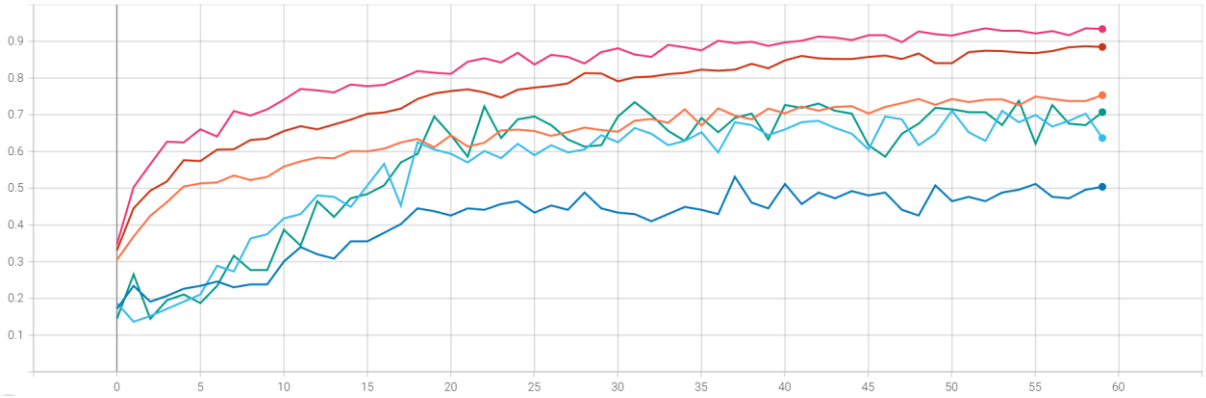


Obrázok 10 *Graf s krivkami epoch_lost pre experimenty 000, 001 a 002* ***2. várka***

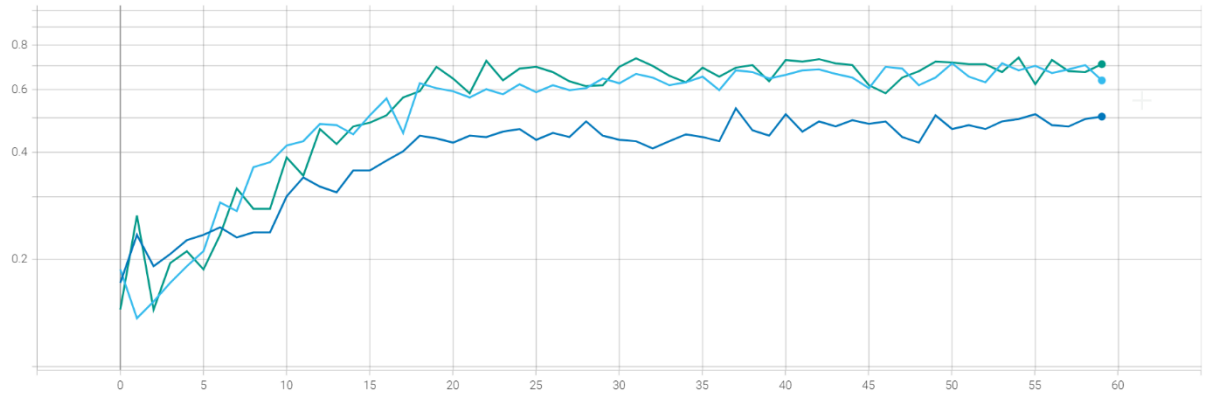
Name	Smoothed	Value	Step	Time	Relative
exp_000\validation	4.171	4.171	59	Fri Dec 30, 18:40:29	37m 38s
exp_001\validation	1.164	1.164	59	Fri Dec 30, 19:23:23	41m 58s
exp_002\validation	1.763	1.763	59	Fri Dec 30, 20:03:59	39m 56s

3. várka experimentov

V ďalší experimentoch sme sa rozhodli pre optimizer SGD (Gradient descent with momentum). Vybrali sme ho na základe odporúčaní z Kaggle, kde sa týmito optimizerom dosiahlo až accuracy 97 % a pri použití knižníc Pytorch až 99 % accuracy. 3. várka má rovnaké parametre (60 epoch, dropout 0.2-0.5, batch_size 32, is_max_pooling: False a filter_size (3, 3)) ako 2. várka s jediným rozdielom a to, že ako optimizer bol vybraný už spomínaný SGD.

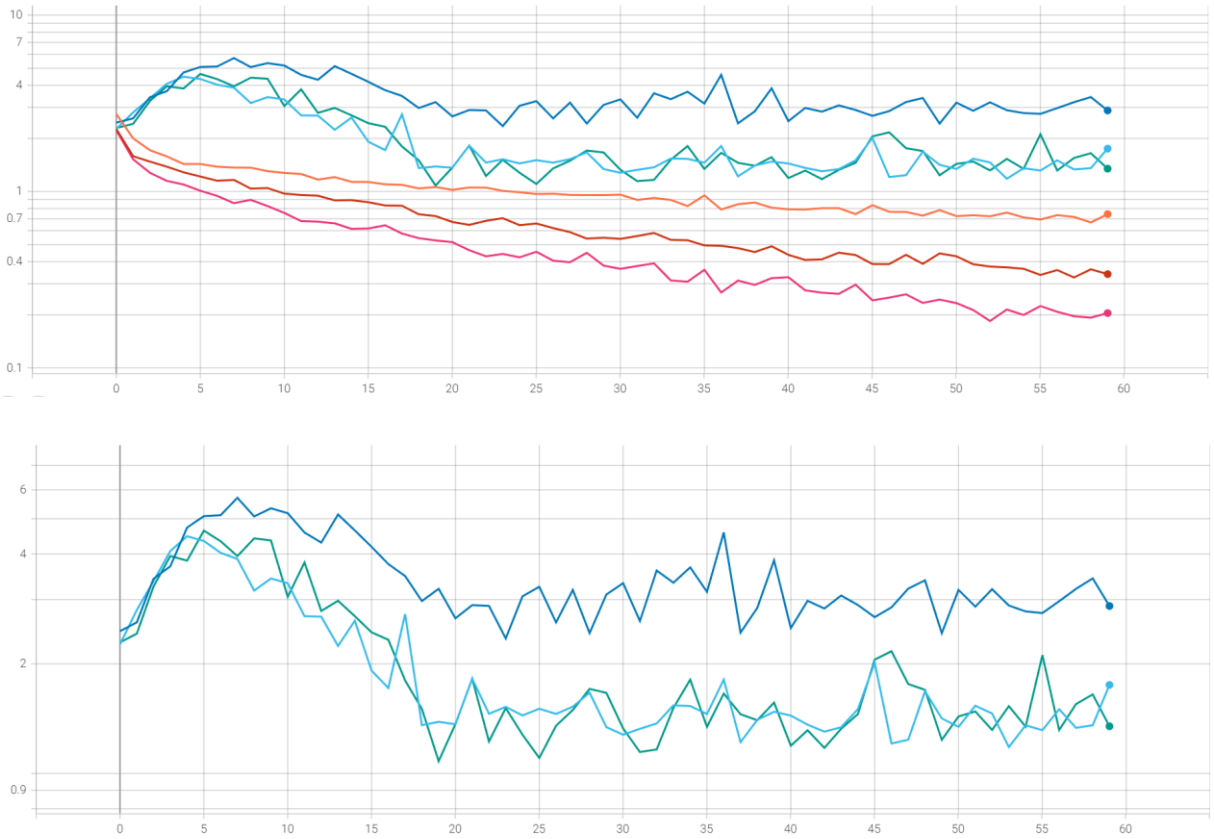


Obrázok 11 *Graf s krivkami epoch_accuracy pre experimenty 000, 001 a 002* ***3. várka, SGD***



Obrázok 12 *Graf s krivkami epoch_accuracy pre experimenty filter validation* ***3 várka SGD***

Name	Smoothed	Value	Step	Time	Relative
exp_000\validation	0.4087	0.5039	59	Fri Dec 30, 20:59:54	38m 54s
exp_001\validation	0.5553	0.6367	59	Fri Dec 30, 21:41:43	41m 7s
exp_002\validation	0.5736	0.707	59	Fri Dec 30, 22:20:35	38m 10s



Obrázok 13 *Graf s krivkami epoch_lost pre experimenty 000, 001 a 002* ***3. várka SGD***

	Name	Smoothed	Value	Step	Time	Relative
	exp_000\validation	3.419	2.881	59	Fri Dec 30, 20:59:54	38m 54s
	exp_001\validation	1.934	1.75	59	Fri Dec 30, 21:41:43	41m 7s
	exp_002\validation	2.01	1.347	59	Fri Dec 30, 22:20:35	38m 10s

Accuracy je lepšia pri exp 002 Loss je lepší pre exp 001. Kontrola cez smoothing. Vybrali by sme **experiment 002 (zelená krivka)**

4. várka - vlastný skript a so sekvenčnou štruktúrou modelu

V poslednom experimente som vytvoril jupyter notebook, kde otestoval vlastný skript a skripty z Kaggle, pretože som si chcel vykresliť cez matplotlib jednotlivé grafy pre accuracy a loss. Na cvičení ste hovorili, že nemáte rada len printscreeny. Zároveň má hnevalo dlhé trénovanie na 50 až 60 epoch čo je aproximácia 50 až 60 minút. Hlavne keď som zistil, že vzorové modely na Kaggle majú bežne 20-30 epoch. Samozrejme na cvičení aj logika hovorí, že pri vyššom počte epoch by sme mali dosiahnuť lepšie výsledky, ale súčasne musíme dávať pozor na pretrénovania. Spomínali ste aj, že máme experimentovať.

Štruktúra modelu vyzerá, tak že miesto for cyklov som využil sekvenčný model - Sequential(), ktorý je vhodný pre jednoduchý stack vrstiev, kde každá vrstva má presne jeden vstupný tenzor a jeden výstupný tenzor.

Z predchádzajúcich experimentov sme využili skúsenosti a nastavili parametre na:

- filters=32,
- kernel_size=(3,3),
- input_shape=(128,128,3),
- activation='relu'
- MaxPool2D(pool_size=(2,2)

Zároveň sme využili EarlyStopping pre zastavenie tréningu, keď sa monitorovaná metrika v našom prípade val_loss prestane zlepšovať. Takto zabezpečíme, že nemusíme čakať 50 až 60 epoch. Pre EarlyStopping sme nastavili parameter patience na 5. To reprezentuje počet epoch bez zlepšenia, po ktorých sa tréning zastaví.

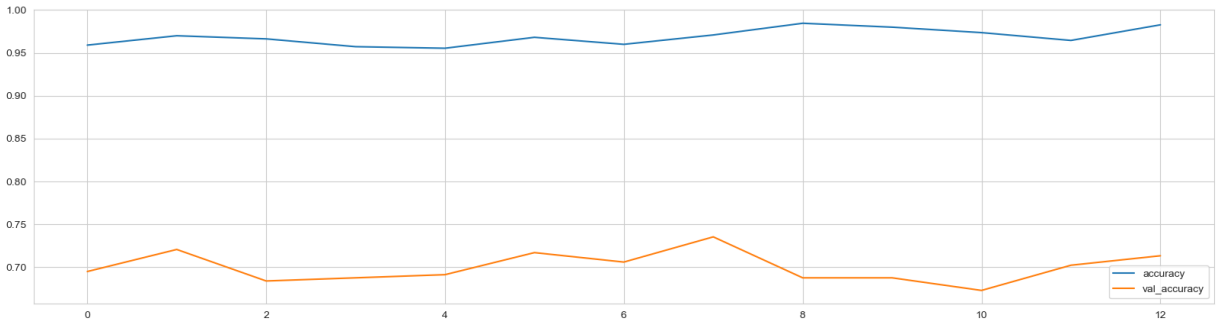
5 rows 5 rows × 4 columns

	loss	accuracy	val_loss	val_accuracy
0	0.159411	0.959016	1.639838	0.694853
1	0.122700	0.969945	1.333482	0.720588
2	0.095077	0.966302	1.363589	0.683824
3	0.135082	0.957195	1.323993	0.687500
4	0.169252	0.955373	1.375443	0.691176

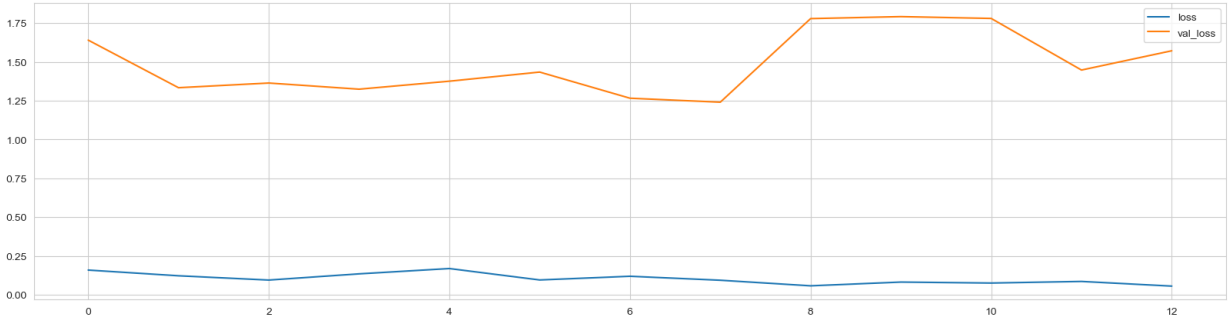
Tabuľka 1 *Dataframe s accuracy a loss pre vykreslenie grafov*

V tabuľke 1 a na grafoch vidíme, že sme dosiahli vysokú hodnotu accuracy a to až 95 % a vyššie a nízku loss hodnotu 0,15 % a nižšie.

Grafy sú dostupné spolu s dataframe v jupyter notebooku dnn_opice.ipynb.



Graf 1 *Hodnoty Accuracy*



Graf 2 *Hodnoty loss*

4. Výber modelu

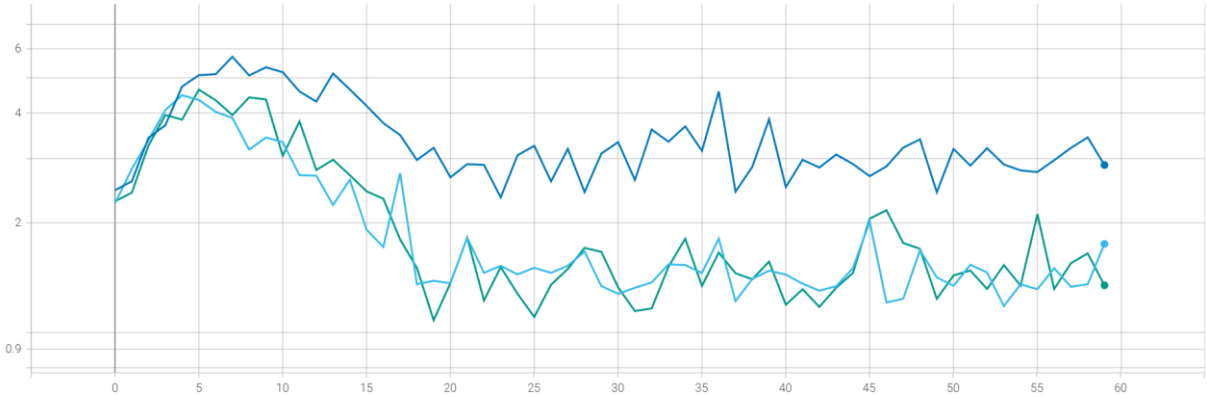
Celkovo sme testovali 3. várky (4. várka bola vlastný skript, do výberu pre nedostatok času sme ju už nezaradili) po 3 experimentoch s rôznymi parametrami (batch_size, dropout, epochy, filter_size, is_max_pooling) a optimizermi (Adam a SGD) a vytvorili aj samostatný test s experimentom so sekvenčným modelom. Najlepšie experimenty mali is_max_pooling: False t.j. využije sa averagepooling a rovnako väčší batch_size: 32.

- Prečo ste vybrali práve tento model?

Ak by sme mali vybrať model len z prvých dvoch várok vyberieme 2. várku, pretože mali is_max_pooling: False t.j. využije sa averagepooling a rovnako väčší batch_size: 32 a počet epoch 60.

Ak by sme mali vybrať model len z troch várok vyberieme tretiu várku a **experiment 002 (zelená krivka)** , kde sme použili SGD optimizer, ktorý mal lepšie hodnoty pri accuracy.

- Ako vyzerá jeho chybová funkcia a jeho metrika (accuracy/~~MCC~~)?



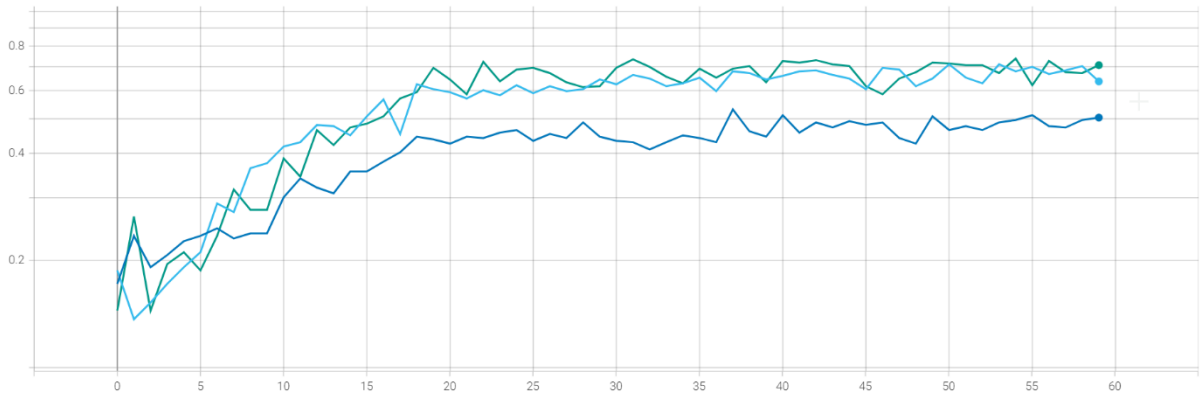
Name	Smoothed	Value	Step	Time	Relative
exp_000\validation	3.419	2.881	59	Fri Dec 30, 20:59:54	38m 54s
exp_001\validation	1.934	1.75	59	Fri Dec 30, 21:41:43	41m 7s
exp_002\validation	2.01	1.347	59	Fri Dec 30, 22:20:35	38m 10s

Výsledky vybraného modelu

- V akej epoche a s akou hodnotou sledovanej metriky sme získali najlepší model?

Najlepší model bol teda v 3. várke **experiment 002 (zelená krivka)** , kde sme použili SGD optimizer, ktorý mal lepšie hodnoty pri accuracy.

Použili sme 60 epoch, dropout 0.2, batch_size 32, is_max_pooling: False a filter_size (3, 3).

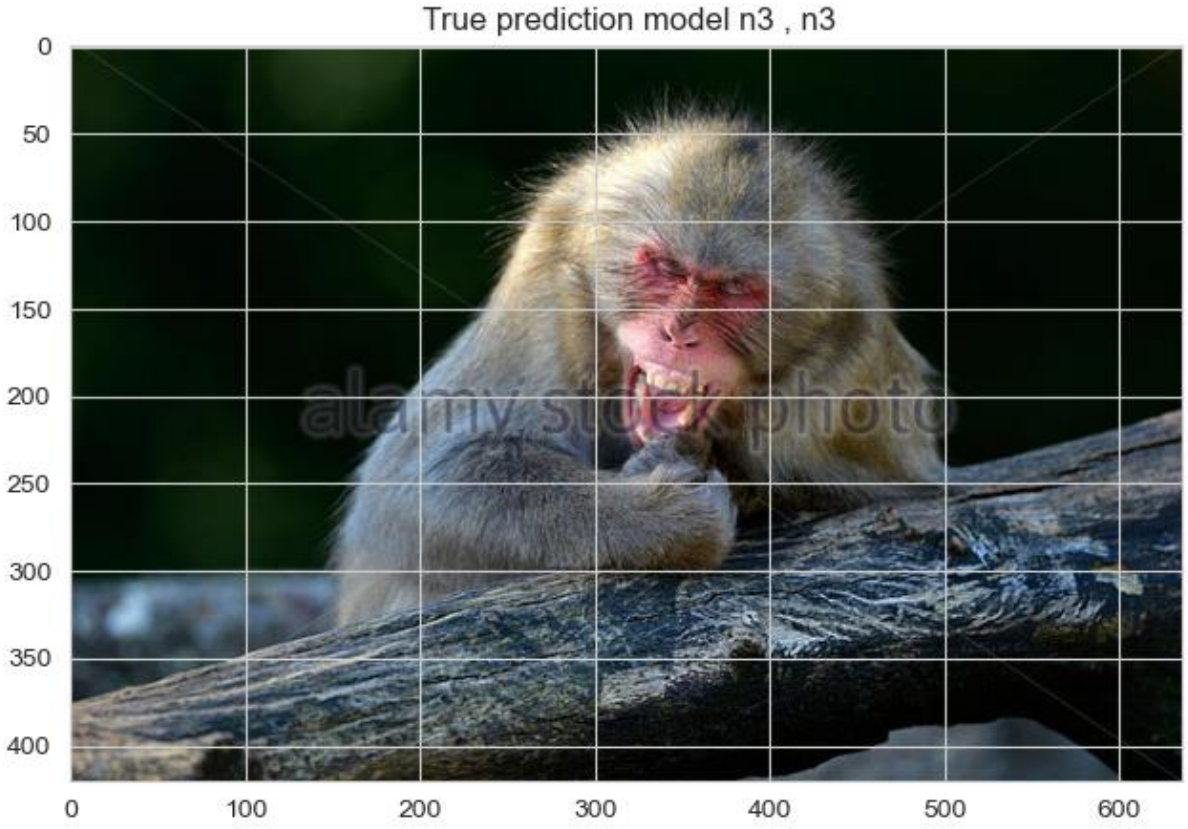


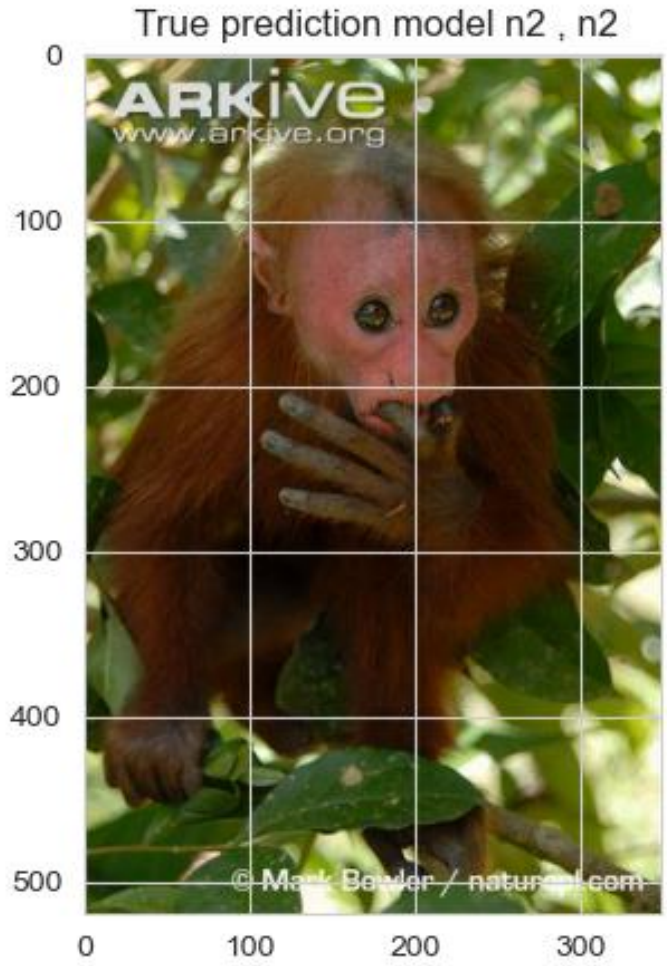
Obrázok 14 *Graf s krivkami epoch_accuracy pre experimenty filter validation* ***3 várka SGD***

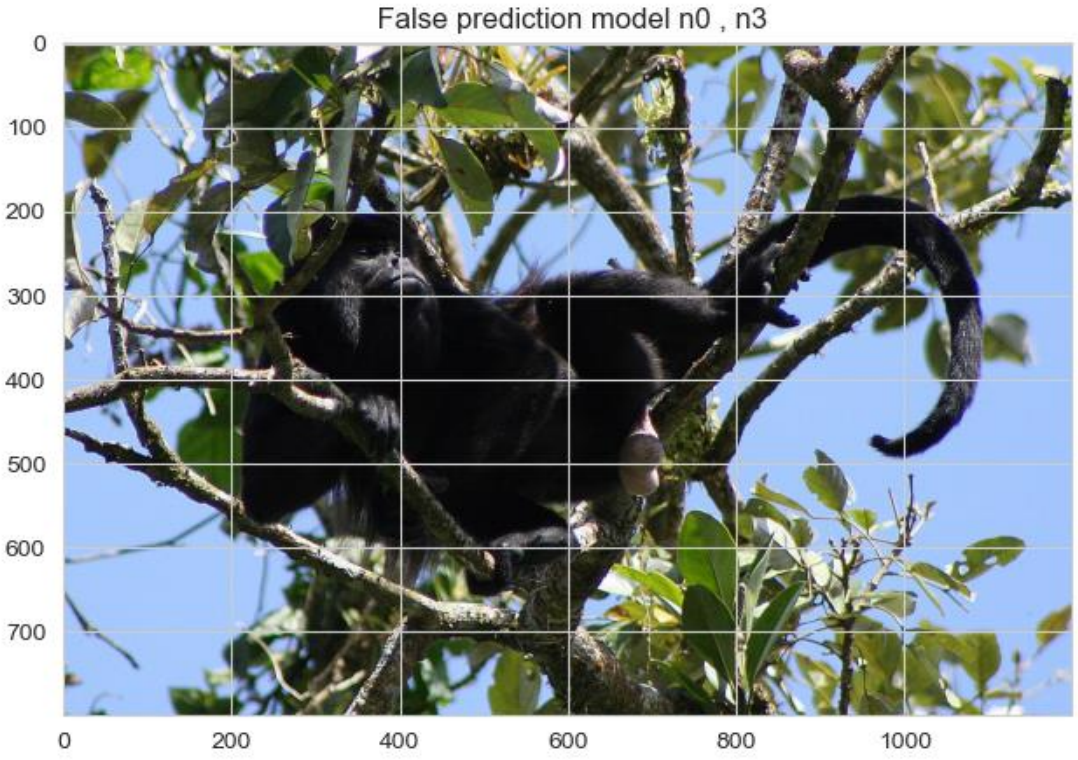
- Odpredikujte testovací dataset

Predikcia modelu

Všetky predikcie sú správne aj po manuálnej kontrole 1 .obrázok je skutočne Makak Japonský a 2. obrázok je skutočne Uakari Šarlátovolíci. 3. obrázok nie je Makak Japonský, ale Vrešťan Pláštikový, takže predikcia správne reagovala s výstupom False.




True prediction model n2 , n2
0
100
200
300
400
500
0
100
200
300
ARKive
www.arkive.org


False prediction model n0 , n3
0
100
200
300
400
500
600
700
0
200
400
600
800
1000

Záver

Všetky materiály, logy z tréningov, zdrojové kódy + jupyter notebook sú k dispozícií v prílohe a v rozšírenej podobe na mojom repozitári na GitHube: https://github.com/miroslav-reiter/PEVS-PANI-Kontrolling

Na tomto repozitári som zhrnul aj niektoré chyby, s ktorými som sa počas zadania stretol a ich riešenia, ktoré môžu pomôcť študentom, ktorí prídu po nás.

Zdrojové kódy, ktoré nám boli poskytnuté ako vzor na cvičeniach som kompletne okomentoval na základe informácií z cvičení + z dokumentácie Keras a modifikoval pre účely klasifikácie datasetu druhov opíc.

Môžeme zhodnotiť, že praktické cvičenia s demo ukážkami boli skvelé. Zadanie mi osobne robilo problémy pre časové okolnosti ako externého študenta. Plno experimentov mi len tak vyfailovalo pre nesprávne nastavenia hodnôt, chyby a iné okolnosti SW, aktualizácie, cloud, virtualizačné nástroje. Okrem modifikovaných súborov z cvičení som vytvoril jupyter notebook z analýzou tohto datasetu, ktorý slúži na rýchle overenie vybraných predpokladov.

Natrénovali sme takúto neurónovú sieť potom sme s tou neurónovou sieť robili predikciu nad testovacím datasetom (validačného setu). Na predikcie som nevyužil súbor z cvičení predictor.py, ale už spomínaný vlastný skript v jupyter notebooku. Predikcie sa nám potvrdili, aj pri manuálnej kontrole. To znamená, že uvedená neurónová sieť bola dobre natrénovaná. Naučili sme rozpoznávať neurónovú sieť, aký je to druh opice.

Prílohy

Zoznam použitých balíčkov/modulov

absl-py @ file:///C:/b/abs_5babsu7y5x/croot/absl-py_1666362945682/work
aiohttp @ file:///C:/b/abs_c4zmy2l696/croot/aiohttp_1670009573673/work
aiosignal @ file:///tmp/build/80754af9/aiosignal_1637843061372/work
anyio @ file:///C:/ci/anyio_1644463701441/work/dist
argon2-cffi @ file:///opt/conda/conda-bld/argon2-cffi_1645000214183/work
argon2-cffi-bindings @ file:///C:/ci/argon2-cffi-bindings_1644569878360/work
asttokens @ file:///opt/conda/conda-bld/asttokens_1646925590279/work
astunparse==1.6.3
async-timeout @ file:///C:/b/abs_43ozhz2a8g/croots/recipe/async-timeout_1664876362767/work
attrs @ file:///C:/b/abs_09s3y775ra/croot/attrs_1668696195628/work
Babel @ file:///C:/b/abs_a2shv_3tqi/croot/babel_1671782804377/work
backcall @ file:///home/ktietz/src/ci/backcall_1611930011877/work
beautifulsoup4 @ file:///C:/ci/beautifulsoup4_1650274792587/work
bleach @ file:///opt/conda/conda-bld/bleach_1641577558959/work
blinker==1.4
Bottleneck @ file:///C:/Windows/Temp/abs_3198ca53-903d-42fd-87b4-03e6d03a8381yfwsuve8/croots/recipe/bottleneck_1657175565403/work
brotlipy==0.7.0
cachetools @ file:///tmp/build/80754af9/cachetools_1619597386817/work
certifi @ file:///C:/b/abs_85o_6fm0se/croot/certifi_1671487778835/work/certifi
cffi @ file:///C:/b/abs_49n3v2hyhr/croot/cffi_1670423218144/work
charset-normalizer @ file:///tmp/build/80754af9/charset-normalizer_1630003229654/work
click @ file:///C:/ci/click_1646056799533/work
colorama @ file:///C:/Windows/TEMP/abs_9439aeb1-0254-449a-96f7-33ab5eb17fc8apleb4yn/croots/recipe/colorama_1657009099097/work
contourpy @ file:///C:/b/abs_d5rpy288vc/croots/recipe/contourpy_1663827418189/work
cryptography @ file:///C:/b/abs_36x9ifdcl4/croot/cryptography_1665612655344/work
cycler @ file:///tmp/build/80754af9/cycler_1637851556182/work
debugpy @ file:///C:/ci/debugpy_1637073815078/work
decorator @ file:///opt/conda/conda-bld/decorator_1643638310831/work
defusedxml @ file:///tmp/build/80754af9/defusedxml_1615228127516/work
entrypoints @ file:///C:/ci/entrypoints_1649926621247/work
executing @ file:///opt/conda/conda-bld/executing_1646925071911/work
fastjsonschema @ file:///C:/Users/BUILDE~1/AppData/Local/Temp/abs_ebruxzvd08/croots/recipe/python-fastjsonschema_1661376484940/work
flatbuffers @ file:///home/ktietz/cip/python-flatbuffers_1634039120618/work
flit_core @ file:///opt/conda/conda-bld/flit-core_1644941570762/work/source/flit_core
fonttools==4.25.0
frozenlist @ file:///C:/b/abs_2bb5uzghsi/croot/frozenlist_1670004511812/work
gast @ file:///Users/ktietz/demo/mc3/conda-bld/gast_1628588903283/work
google-auth @ file:///opt/conda/conda-bld/google-auth_1646735974934/work
google-auth-oauthlib @ file:///tmp/build/80754af9/google-auth-oauthlib_1617120569401/work

google-pasta @ file:///Users/ktietz/demo/mc3/conda-bld/google-pasta_1630577991354/work
grpcio @ file:///C:/ci/grpcio_1637590968244/work
h5py @ file:///C:/ci/h5py_1659071640187/work
idna @ file:///C:/b/abs_bdhbebrioa/croot/idna_1666125572046/work
importlib-metadata @ file:///C:/ci/importlib-metadata_1648544472910/work
importlib-resources @ file:///tmp/build/80754af9/importlib_resources_1625135880749/work
ipykernel @ file:///C:/b/abs_21ykzkm7y_/croots/recipe/ipykernel_1662361803478/work
ipython @ file:///C:/b/abs_536xsn7rj0/croot/ipython_1670919360443/work
ipython-genutils @ file:///tmp/build/80754af9/ipython_genutils_1606773439826/work
ipywidgets @ file:///tmp/build/80754af9/ipywidgets_1634143127070/work
jedi @ file:///C:/ci/jedi_1644315425835/work
Jinja2 @ file:///C:/b/abs_7cdis66kl9/croot/jinja2_1666908141852/work
joblib @ file:///C:/b/abs_e60_bwl1v6/croot/joblib_1666298845728/work
json5 @ file:///tmp/build/80754af9/json5_1624432770122/work
jsonschema @ file:///C:/b/abs_59eyhnbyej/croots/recipe/jsonschema_1663375476535/work
jupyter @
file:///C:/Windows/TEMP/abs_56xfdi__li/croots/recipe/jupyter_1659349053177/work
jupyter-console @ file:///C:/b/abs_e8gx0csskd/croot/jupyter_console_1671541918848/work
jupyter-server @ file:///C:/b/abs_1cfi3__jl8/croot/jupyter_server_1671707636383/work
jupyter_client @ file:///C:/b/abs_b9pns5mx5p/croot/jupyter_client_1671703062216/work
jupyter_core @ file:///C:/b/abs_84df679bho/croot/jupyter_core_1672332237650/work
jupyterlab @ file:///C:/b/abs_36s1nw_wru/croot/jupyterlab_1672132698305/work
jupyterlab-pygments @
file:///tmp/build/80754af9/jupyterlab_pygments_1601490720602/work
jupyterlab-widgets @ file:///tmp/build/80754af9/jupyterlab_widgets_1609884341231/work
jupyterlab_server @
file:///C:/b/abs_5ayr1vni0w/croot/jupyterlab_server_1672127362727/work
keras @ file:///C:/Users/builder/adipietro/mc3/tf210/conda-
bld/keras_1669760570649/work/keras-2.10.0-py2.py3-none-any.whl
Keras-Preprocessing @ file:///tmp/build/80754af9/keras-preprocessing_1612283640596/work
kiwisolver @ file:///C:/ci/kiwisolver_1653292408254/work
lxml @ file:///C:/ci/lxml_1657527495424/work
Markdown @ file:///C:/b/abs_98lv_ucina/croot/markdown_1671541919225/work
MarkupSafe @ file:///C:/ci/markupsafe_1654489871526/work
matplotlib @ file:///C:/b/abs_03z_dj2gty/croot/matplotlib-suite_1670466145509/work
matplotlib-inline @ file:///C:/ci/matplotlib-inline_1661934035815/work
mistune==0.8.4
mkl-fft==1.3.1
mkl-random @ file:///C:/ci/mkl_random_1626186184278/work
mkl-service==2.4.0
multidict @ file:///C:/b/abs_6cx_8w3cv2/croot/multidict_1665674238352/work
munkres==1.1.4
nbclassic @ file:///C:/b/abs_26e3fkk516/croot/nbclassic_1668174974037/work
nbclient @ file:///C:/ci/nbclient_1650290386732/work
nbconvert @ file:///C:/b/abs_4av3q4okro/croot/nbconvert_1668450658054/work
nbformat @ file:///C:/b/abs_85_3g7dkt4/croot/nbformat_1670352343720/work
nest-asyncio @ file:///C:/ci/nest-asyncio_1649829929372/work
notebook @ file:///C:/b/abs_ca13hqvuzw/croot/notebook_1668179888546/work
notebook_shim @ file:///C:/b/abs_ebfcztg6x/croot/notebook-shim_1668160590914/work
numexpr @ file:///C:/b/abs_a7kbak88hk/croot/numexpr_1668713882979/work

numpy @ file:///C:/b/abs_5ct9ex77k9/croot/numpy_and_numpy_base_1668593740598/work
oauthlib @ file:///C:/b/abs_2eoymqc2ow/croot/oauthlib_1665490906043/work
opt-einsum @ file:///tmp/build/80754af9/opt_einsum_1621500238896/work
packaging @ file:///C:/b/abs_cfsup8ur87/croot/packaging_1671697442297/work
pandas @ file:///C:/b/abs_c6fuezktfm/croot/pandas_1670425103552/work
pandocfilters @ file:///opt/conda/conda-bld/pandocfilters_1643405455980/work
parso @ file:///opt/conda/conda-bld/parso_1641458642106/work
pickleshare @ file:///tmp/build/80754af9/pickleshare_1606932040724/work
Pillow==9.3.0
pkgutil_resolve_name @ file:///C:/Users/BUILDE~1/AppData/Local/Temp/abs_81wm45v3kb/croots/recipe/pkgutil-resolve-name_1661463352381/work
platformdirs @ file:///C:/b/abs_73cc5cz_1u/croots/recipe/platformdirs_1662711386458/work
plotly @ file:///C:/ci/plotly_1658160690859/work
ply==3.11
prometheus-client @ file:///C:/Windows/TEMP/abs_ab9nx8qb08/croots/recipe/prometheus_client_1659455104602/work
prompt-toolkit @ file:///tmp/build/80754af9/prompt-toolkit_1633440160888/work
protobuf==3.20.1
psutil @ file:///C:/Windows/Temp/abs_b2c2fd7f-9fd5-4756-95ea-8aed74d0039flsd9qufz/croots/recipe/psutil_1656431277748/work
pure-eval @ file:///opt/conda/conda-bld/pure_eval_1646925070566/work
pyasn1 @ file:///Users/ktietz/demo/mc3/conda-bld/pyasn1_1629708007385/work
pyasn1-modules==0.2.8
pycparser @ file:///tmp/build/80754af9/pycparser_1636541352034/work
Pygments @ file:///opt/conda/conda-bld/pygments_1644249106324/work
PyJWT @ file:///C:/ci/pyjwt_1657529430378/work
pyOpenSSL @ file:///opt/conda/conda-bld/pyopenssl_1643788558760/work
pyparsing @ file:///C:/Users/BUILDE~1/AppData/Local/Temp/abs_7f_7lba6rl/croots/recipe/pyparsing_1661452540662/work
PyQt5==5.15.7
PyQt5-sip @ file:///C:/Windows/Temp/abs_d7gmd2jg8i/croots/recipe/pyqt-split_1659273064801/work/pyqt_sip
pyrsistent @ file:///C:/ci/pyrsistent_1636111468851/work
PySocks @ file:///C:/ci/pysocks_1605287845585/work
python-dateutil @ file:///tmp/build/80754af9/python-dateutil_1626374649649/work
pytz @ file:///C:/b/abs_22fofvpn1x/croot/pytz_1671698059864/work
pywin32==305.1
pywinpty @ file:///C:/ci_310/pywinpty_1644230983541/work/target/wheels/pywinpty-2.0.2-cp38-none-win_amd64.whl
pyzmq @ file:///C:/ci/pyzmq_1657616005830/work
qtconsole @ file:///C:/ci/qtconsole_1662018992304/work
QtPy @ file:///C:/ci/qtpy_1662015036641/work
requests @ file:///C:/ci/requests_1657717096906/work
requests-oauthlib==1.3.0
rsa @ file:///tmp/build/80754af9/rsa_1614366226499/work
scikit-learn @ file:///C:/b/abs_eezai1guz5/croot/scikit-learn_1667587552067/work
scipy==1.9.3

seaborn @ file:///C:/b/abs_64ypwhwrqe/croot/seaborn_1669625732106/work
Send2Trash @ file:///tmp/build/80754af9/send2trash_1632406701022/work
sip @ file:///C:/Windows/Temp/abs_b8fxd17m2u/croots/recipe/sip_1659012372737/work
six @ file:///tmp/build/80754af9/six_1644875935023/work
sniffio @ file:///C:/ci/sniffio_1614030707456/work
soupsieve @ file:///C:/b/abs_fasraqxhlv/croot/soupsieve_1666296394662/work
stack-data @ file:///opt/conda/conda-bld/stack_data_1646927590127/work
tenacity @ file:///C:/Windows/TEMP/abs_980d07a6-8e21-4174-9c17-7296219678ads7dhdov_/croots/recipe/tenacity_1657899108023/work
tensorboard @ file:///C:/Users/builder/adipietro/mc3/tf210/conda-bld/tensorboard_1669760968711/work/tensorboard-2.10.0-py3-none-any.whl
tensorboard-data-server @ file:///C:/b/abs_2fhvpo862s/croot/tensorboard-data-server_1670853600144/work/tensorboard_data_server-0.6.1-py3-none-any.whl
tensorboard-plugin-wit @ file:///C:/tf/b/tensorboard-plugin-wit_1660162132996/work/tensorboard_plugin_wit-1.8.1-py3-none-any.whl
tensorflow==2.10.0
tensorflow-estimator @ file:///C:/Users/builder/adipietro/mc3/tf210/conda-bld/tensorflow-estimator_1669761460695/work/tensorflow_estimator-2.10.0-py2.py3-none-any.whl
termcolor @ file:///C:/b/abs_16qe7jhw7n/croot/termcolor_1668084642458/work
terminado @ file:///C:/b/abs_25nakickad/croot/terminado_1671751845491/work
threadpoolctl @ file:///Users/ktietz/demo/mc3/conda-bld/threadpoolctl_1629802263681/work
tinycss2 @ file:///C:/b/abs_52w5vfuaax/croot/tinycss2_1668168823131/work
toml @ file:///tmp/build/80754af9/toml_1616166611790/work
tomli @ file:///C:/Windows/TEMP/abs_ac109f85-a7b3-4b4d-bcfd-52622eceddf0hy332ojo/croots/recipe/tomli_1657175513137/work
tornado @ file:///C:/ci/tornado_1662476991259/work
tqdm @ file:///C:/b/abs_0axbz66qik/croots/recipe/tqdm_1664392691071/work
traitlets @ file:///C:/b/abs_e5m_xjjl94/croot/traitlets_1671143896266/work
typing_extensions @ file:///C:/b/abs_89eui86zuq/croot/typing_extensions_1669923792806/work
urllib3 @ file:///C:/b/abs_21qa0j12xt/croot/urllib3_1670527000672/work
wcwidth @ file:///Users/ktietz/demo/mc3/conda-bld/wcwidth_1629357192024/work
webencodings==0.5.1
websocket-client @ file:///C:/ci/websocket-client_1614804473297/work
Werkzeug @ file:///C:/b/abs_17q5kgb8bo/croot/werkzeug_1671216014857/work
widgetsnbextension @ file:///C:/ci/widgetsnbextension_1645009558218/work
win-inet-pton @ file:///C:/ci/win_inet_pton_1605306167264/work
wincertstore==0.2
wrapt @ file:///C:/Windows/Temp/abs_7c3dd407-1390-477a-b542-fd15df6a24085_diwiza/croots/recipe/wrapt_1657814452175/work
yarl @ file:///C:/Users/BUILDE~1/AppData/Local/Temp/abs_e5nlunspj6/croots/recipe/yarl_1661437086516/work
zipp @ file:///C:/ci/zipp_1652274073489/work