

Dokumentace Ročníkového projektu

Jáchym Mraček

Klasifikace muzejních umění pomocí konvoluční neuronové sítě

August 3, 2025

1 Popis tématu

Cílem projektu je klasifikovat 89 muzejních objektů pomocí konvoluční neuronové sítě.

2 Řešení Trénování modelu

2.1 Data

Trénovací data byla získaná pomocí zadaných videí, která byla nasamplována a následně filtrovaná pomocí kosínové podobnosti resnet50 embeddingů. Maximální tolerance největší kosínové podobnosti mezi přijatých embeddingů snímků je stanovena na 0.98, kde hyperparametrem této úlohy je počet kopií těchto přijatých snímků pro rozšíření datasetu.

Úloha obsahuje i přiložené obrázky, které nejsou obsaženy v daných videích a proto z nich získáme validační a testové data, tak že obrázky dané třídy rozdělíme na dvě skupiny, kde validační data obsahují první třetinu dat na danou třídu a testovací data obsahují dvě třetiny dat na danou třídu.

Všechny data jsou načítána pomocí Dataloaderu z knihovny Pytorch, kde parametry jsou zvoleny, tak aby se modely trénovaly co nejrychleji. Zvolené parametry jsou `batch_size = 8` a `num_workers = 8`.

2.2 Výběr modelu

Typ modelu byl vybrán na základě pěti epochového trénování modelů, kde práh podobnosti obrázku v datasetu byl stanoven na 98% s žádnou kopií obrázků a ztrátová funkce byla nastavena na CrossEntropyLoss a optimizer na `optim.SGD(self.model.parameters(), lr=0.001, momentum=0.9)`.

Průběh trénování modelů byl následující a byl vybrán MaxVit model, kvůli efektivnímu trénování na našich validačních a trénovacích datech.

model	best val. acc	epoch	train accuracy	validate accuracy	test accuracy
AlexNet	0.72	1	0.76	0.66	0.67
		2	0.86	0.67	0.66
		3	0.89	0.72	0.67
		4	0.89	0.65	0.63
		5	0.90	0.64	0.66

model	best val. acc	epoch	train accuracy	validate accuracy	test accuracy
ConvNeXt	0.82	1	0.79	0.62	0.61
		2	0.90	0.78	0.74
		3	0.92	0.82	0.75
		4	0.93	0.70	0.71
		5	0.95	0.78	0.73
DenseNet	0.82	1	0.84	0.68	0.67
		2	0.92	0.82	0.77
		3	0.94	0.81	0.76
		4	0.94	0.80	0.77
		5	0.92	0.82	0.77
EfficientNet	0.59	1	0.13	0.02	0.02
		2	0.49	0.18	0.17
		3	0.68	0.41	0.37
		4	0.77	0.50	0.47
		5	0.83	0.59	0.52
EfficientNetV2	0.87	1	0.81	0.74	0.71
		2	0.90	0.87	0.78
		3	0.93	0.82	0.81
		4	0.94	0.85	0.78
		5	0.95	0.85	0.82
GoogLeNet	0.82	1	0.60	0.40	0.39
		2	0.78	0.63	0.59
		3	0.85	0.76	0.69
		4	0.88	0.82	0.74
		5	0.90	0.82	0.75
MaxVit	0.88	1	0.68	0.48	0.49
		2	0.85	0.77	0.73
		3	0.89	0.82	0.79
		4	0.90	0.86	0.83
		5	0.92	0.88	0.82
MNASNet	0.71	1	0.00	0.00	0.00
		2	0.13	0.14	0.11
		3	0.45	0.38	0.32
		4	0.65	0.49	0.53
		5	0.81	0.71	0.65
MobileNet V2	0.78	1	0.60	0.38	0.38
		2	0.80	0.61	0.60
		3	0.87	0.74	0.70

model	best val. acc	epoch	train accuracy	validate accuracy	test accuracy
RegNet	0.88	4	0.89	0.78	0.73
		5	0.91	0.78	0.77
		1	0.80	0.67	0.64
		2	0.87	0.76	0.72
		3	0.91	0.77	0.71
		4	0.92	0.88	0.76
		5	0.94	0.86	0.77
		1	0.80	0.67	0.64
		2	0.87	0.76	0.72
		3	0.91	0.77	0.71
ResNet	0.76	4	0.92	0.88	0.76
		5	0.94	0.86	0.77
		1	0.70	0.54	0.54
		2	0.88	0.73	0.71
		3	0.91	0.76	0.72
ResNeXt	0.88	4	0.92	0.76	0.74
		5	0.94	0.74	0.73
		1	0.86	0.72	0.69
		2	0.91	0.82	0.77
		3	0.92	0.79	0.79
ShuffleNet V2	0.06	4	0.93	0.88	0.76
		5	0.94	0.82	0.79
		1	0.06	0.03	0.04
		2	0.06	0.04	0.04
		3	0.07	0.04	0.03
SqueezeNet	0.68	4	0.08	0.04	0.03
		5	0.11	0.06	0.04
		1	0.60	0.48	0.44
		2	0.76	0.62	0.56
		3	0.84	0.63	0.52
SwinTransformer	0.85	4	0.85	0.68	0.62
		5	0.87	0.63	0.61
		1	0.85	0.76	0.73
		2	0.90	0.81	0.78
		3	0.91	0.79	0.72
VGG	0.82	4	0.93	0.85	0.78
		5	0.94	0.82	0.73
		1	0.78	0.68	0.63

model	best val. acc	epoch	train accuracy	validate accuracy	test accuracy
		2	0.83	0.76	0.74
		3	0.89	0.80	0.72
		4	0.91	0.81	0.73
		5	0.92	0.82	0.75
VisionTransformer	0.76	1	0.86	0.60	0.63
		2	0.92	0.69	0.64
		3	0.93	0.72	0.72
		4	0.94	0.76	0.71
		5	0.96	0.71	0.67
Wide ResNet	0.82	1	0.79	0.59	0.55
		2	0.90	0.75	0.72
		3	0.92	0.77	0.75
		4	0.94	0.79	0.74
		5	0.95	0.82	0.78

2.3 Výběr Optimizeru

Výběr optimizeru proběhl obdobně, jako u výběru typu modelu, jenom s tím, že jsme trénovali MaxVit model a všechny parametry optimizerů byly vybrány defaultně podle dokumentace. Vybraný optimizer je SGD.

optimizer	best val. acc	epoch	train accuracy	validate accuracy	test accuracy
Adadelata	0.75	1	0.53	0.35	0.32
		2	0.67	0.46	0.44
		3	0.87	0.75	0.70
		4	0.77	0.65	0.71
		5	0.88	0.75	0.76
Adagrad	0.35	1	0.13	0.12	0.11
		2	0.22	0.12	0.09
		3	0.43	0.20	0.16
		4	0.64	0.26	0.26
		5	0.72	0.35	0.32
Adam	0.68	1	0.69	0.53	0.50
		2	0.78	0.57	0.63
		3	0.82	0.66	0.61
		4	0.85	0.68	0.64
		5	0.87	0.64	0.57
AdamW	0.72	1	0.69	0.62	0.49
		2	0.74	0.55	0.51

optimizer	best val. acc	epoch	train accuracy	validate accuracy	test accuracy
		3	0.83	0.68	0.62
		4	0.81	0.65	0.58
		5	0.88	0.72	0.67
Adamax	0.83	1	0.87	0.79	0.75
		2	0.90	0.77	0.77
		3	0.92	0.82	0.75
		4	0.92	0.83	0.79
		5	0.93	0.83	0.78
ASGD	0.86	1	0.69	0.51	0.51
		2	0.85	0.74	0.72
		3	0.90	0.82	0.78
		4	0.91	0.86	0.81
		5	0.93	0.85	0.82
NAdam	0.74	1	0.74	0.56	0.51
		2	0.76	0.56	0.51
		3	0.82	0.70	0.62
		4	0.86	0.71	0.63
		5	0.78	0.74	0.61
RAdam	0.79	1	0.83	0.79	0.71
		2	0.83	0.76	0.71
		3	0.83	0.74	0.68
		4	0.81	0.71	0.63
		5	0.78	0.64	0.61
RMSprop	0.01	1	0.03	0.01	0.01
		2	0.03	0.01	0.01
		3	0.01	0.01	0.01
		4	0.00	0.01	0.01
		5	0.03	0.01	0.01
Rprop	0.03	1	0.02	0.01	0.00
		2	0.04	0.03	0.05
		3	0.04	0.01	0.01
		4	0.04	0.01	0.03
		5	0.02	0.01	0.01
SGD	0.88	1	0.69	0.49	0.48
		2	0.83	0.73	0.70
		3	0.89	0.81	0.79
		4	0.91	0.85	0.83
		5	0.92	0.88	0.81

2.4 Parametry Optimizeru

MaxVit Model s SGD optimizerem byl testován na několik parametrů, kde nejdříve byla testována dvojice lr a momentum a následně doladěny ostatní parametry. Vybrané parametry jsou lr = 0.01, momentum = 0.6, weight_decay = 0, dampening = 0 a nesterov = False.

parameters	best val. acc	epoch	train accuracy	validate accuracy	test ac- curacy
lr = 0.01	0.88	1	0.83	0.76	0.74
momentum = 0.6		2	0.91	0.82	0.77
weight_decay = 0		3	0.92	0.88	0.84
dampening = 0		4	0.94	0.86	0.83
nesterov = False		5	0.96	0.88	0.83

2.5 Počet dat a vyváženost datasetu

Natrénoval jsem model na 5 epochách a zaznamenal epochu s nejlepší validační přesností. Zkoušel jsem variace vybalancovaného datasetu a počet kopií. Vybral jsem 2 kopie na vyvážený dataset, i když rozdíly mezi jednotlivými variacemi nejsou velké.

copies	best validation accuracy	balance dataset (yes/no)
1	0.89	no
2	0.89	no
3	0.89	no
1	0.89	yes
2	0.90	yes
3	0.90	yes
5	0.89	yes

2.6 Finálně natrénovaný model

Nejlépeší epocha je:

Train Accuracy	Validate Accuracy	Test Accuracy
0.99	0.91	0.86

Kde trénování modelu probíhalo následovně:

3 Použité technologie

Nejdůležitější knihovní technologie projektu jsou: Pytorch knihovny, Numpy, matplotlib, Sklearn, Threading, Pyside6, Concurrent.futures, argparse, shutil, collections, abc, os, cv2, PIL, tqdm, Enum

Trénování probíhalo na GPU cuda a celý projekt byl programován v jazyce Python ve vývojovém prostředí Visual Studio Code.

4 Operační systém

projekt je určen pro operační systém Windows nebo Linux.

5 Uživatelská a programátorská dokumentace

5.1 Itemy

Startovní tlačítko: Stisknutím startovacího tlačítka se uživatel dostane z úvodní obrazovky na url obrazovku.

End tlačítko: Stisknutím tohoto tlačítka se kompletní aplikace zavře.

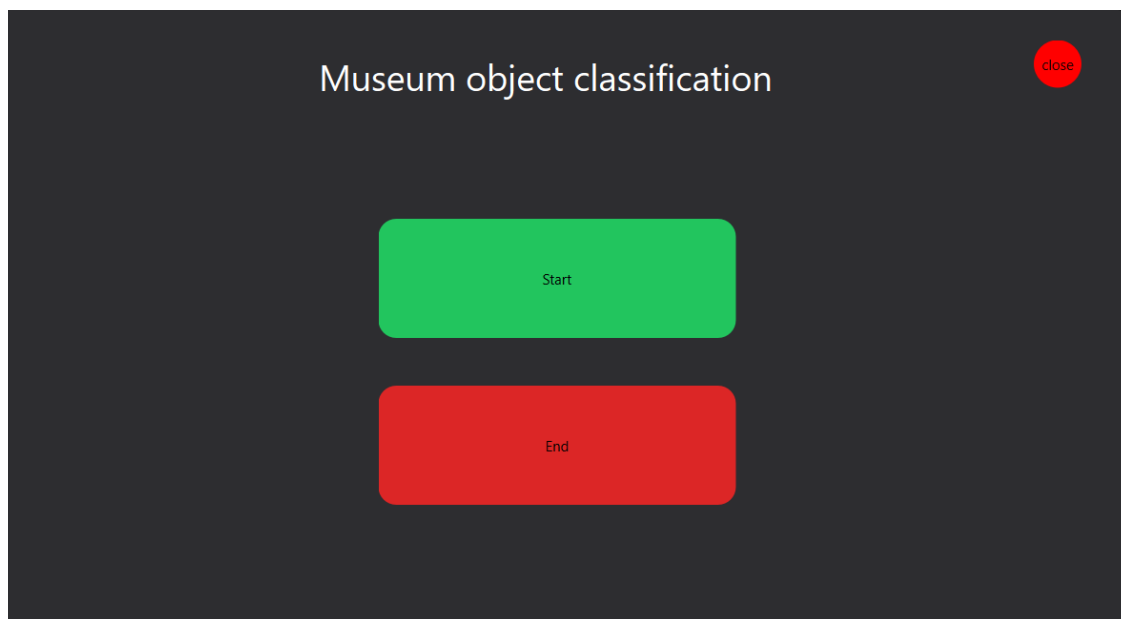
Close tlačítko: Stisknutím tlačítka se celá aplikace hned ukončí.

Text box: Uživatel zde může napsat cestu k obrázku.

Send tlačítko: Stisknutím tlačítka se odešle cesta k zadanému obrázku a začne jeho klasifikace.

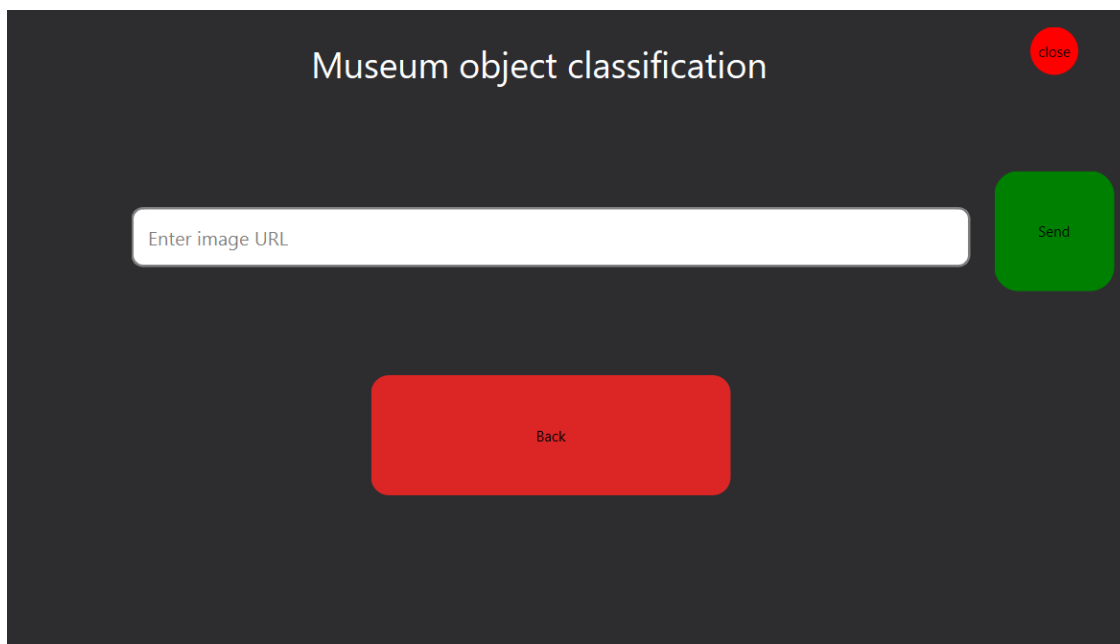
5.2 Obrazovky

5.2.1 Úvodní obrazovka



Obsahuje tlačítka Start, End a Close a také nadpis s textem Museum object classification.

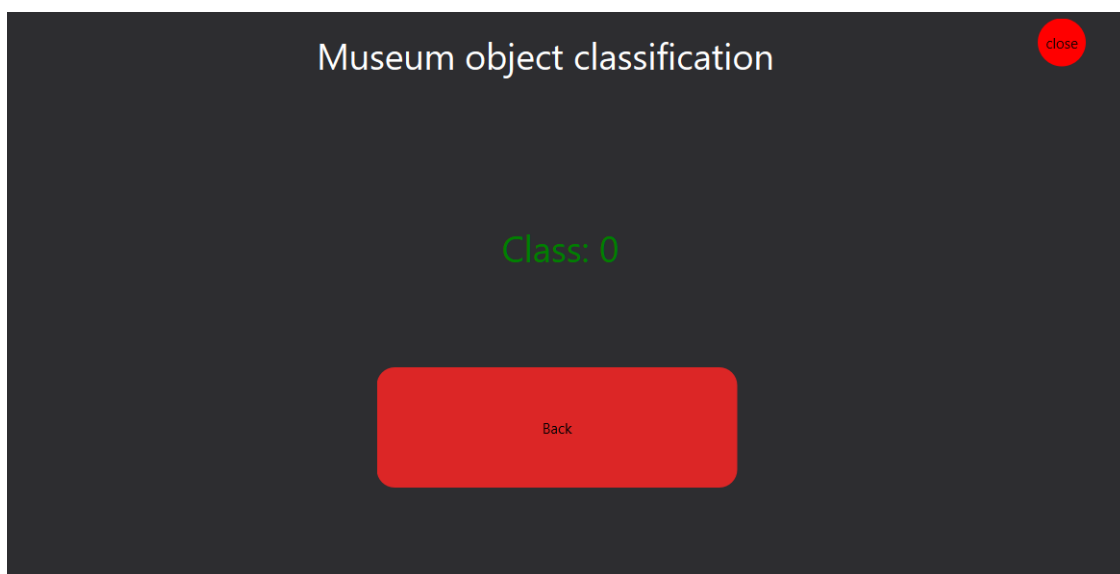
5.2.2 Url obrazovka



Obsahuje tlačítka: Back, Close, Send a text box, do kterého uživatel může napsat url obrázku, který má být klasifikován a také nadpis s textem Museum object classification.

Kod je rozdělen na 3 hlavních Python soubory se jmény build_dataset.py, build_classifier.py a user.py.

5.2.3 Classification page



Po stisknutí tlačítka send se uživateli zobrazí třída, do jaké obrázek byl klasifikován.

5.3 build_dataset.py

Soubor řešící zpracování všech typů dat a jejich ukládání.

5.3.1 Input

Uživatel může vybrat tři operace nad naším datasetem, které jsou zadrátované ve třídě class Regime(Enum), kde tři zmíněné operace jsou:

Frame: Uživatel napíše na příkazové řádce pomocí správného parametru build_dataset.py –regime Frame, čímž se začnou zpracovávat zadaná videa a jednotlivé snímky se začnou ukládat uživateli do složky s názvem train_video_images, která obsahuje další pod složky s názvem daného videa obsahující třídivé snímky z daného videa.

Split: Uživatel zvolí pomocí správného parametru na příkazové řádce build_dataset.py – Split, čímž se uživateli vytvoří dvě složky s názvy validate_frames a test_frames, ve kterých budou validační a testovací snímky v daných podsložkách.

Balance: Uživatel zvolí pomocí parametru build_dataset.py – Balance na příkazové řádce, čímž se vytvoří vyvážený dataset pomocí kopií daných snímků. Všechny snímky jsou ve složce s názvem Balance_dataset s příslušnými podsložkami.

5.3.2 Vyjimky

Další implementované třídy jsou vyjimky, které jsou řešeny pomocí abstraktní třídy Errors dědící od třídy Exceptions. První vyjimka je řešena pomocí třídy ExtraVideo řešící navíc přiložené video, které nemá testovací data.

Další třídou je PlayVideoError, které řeší nepřehrávatelné video a poslední třetí vyjimku řeší třída InputError, která zachycuje špatně zvolený vstupní –regime ARGUMENT. Všechny tyto neabstraktní třídy dědí od Errors.

5.3.3 Embeddingy

Resnet embeddingy získáváme pomocí třídy Embedder, která obsahuje metodu get_embedding() vracící příznakový vektor daného snímku.

5.3.4 Složky

Každá složka dědí od abstraktní třídy folders, která v konstruktoru vytvoří danou složku na uživatelský disk, pokud daná složka s danou cestou neexistuje. Zároveň tato abstraktní třída si ukládá cestu k dané složce do proměnné path.

Vnořené složky dědí od abstraktní třídy NestedFolder(Folders).

5.3.5 Zpracování dat, které nejsou ve videích

Hlavní třídou rozdělující data na validační a testovací je NotVideoFrames, která dědí od Folders. Rozdělující metoda je split(), která získává informace pro rozdělení z metody get_info_for_splitting().

Potom následuje abstraktní třída s názvem PartNotVideoFrames(Folder,ABC), která je děděna třídami ValidateFrames(PartNotVideoFrames) a TestFrames(PartNotVideoFrames) sloužící k

vytvoření složky s testovacími a validačními snímky. Příslušná abstraktní třída obsahuje metodu `add()`, které slouží k přidání snímku do správné složky.

5.3.6 Balance dataset

Pro vyvážení datasetu používáme třídu `BalanceTrainingFrames(Folder)`, která obsahuje metodu `balance()` balancující dataset pomocí metod `get_count_img_classes()` vracející slovník s klíčem jména třídy a hodnotou je počet snímků v dané třídě a metody `get_max_imgs_in_class()` vracící maximální počet snímků v dané třídě a metody `duplicate()` tvořící duplikáty daných snímků.

Závěrečnou třídu je `BalanceClassFrames(NestedFolder)` s metodou `add()`, která reprezentuje vnořenou složku s kopiemi snímků. Tyto vnořené třídy jsou umístěné v balancované složce.

5.3.7 Vytvoření datasetu ze snímků z videa

`TrainFrames(Folder)` je třída, která slouží ke zpracování zadaných videí obsahující metodu `process_all_videos()`, která využívá vlákna zpracovávající videa pomocí metody `process_video()`, která zavolá metodu pro framování videa a snímky uloží.

5.3.8 Zpracování vstupu

Obsahuje abstraktní třídu `InputProcessor(ABC)`, která je dědena i v `build_classifier.py`, jelikož také zpracovává vstupní uživatelské argumenty, a třídu `RegimeProcessor(InputProcessor)`, která zpracuje argumenty v `build_dataset.py`.

5.4 user.py

5.4.1 Window

Soubor sloužící k vytvoření hlavního uživatelského okna, ve kterém uživatel může klikat na tlačítka, pohybovat s oknem nebo zadat url obrázku, který má být klasifikován.

Hlavní třídou je `class MainWindow(QMainWindow)`, která má v konstruktoru příkazy sestavující úvodní okno a metody `mouseMoveEvent()`, `mousePressEvent()` sloužící k pohybu okna, pokud uživatel chce okno přesunout.

Dále obsahuje metodu `delete_items()`, která odebere veškeré itemy (tlačítka, labely...) z okna a je volána při přechodu na jinou stránku, jelikož každá stránka obsahuje jiné itemy. Metoda `go_url_page()` slouží k přepnutí na obrazovku obsahující text box, do které uživatel může zadat cestu k obrázku a `go_start_page()` sloužící k přepnutí na úvodní obrazovku a metoda `go_classification_page(self)` slouží k přepnutí na obrazovku obsahující výsledek klasifikace.

Metody `choose_img()` slouží ke zpracování zadané cesty obrázku, pokud dané url je chybné, pak se v url boxu zobrazí chybová zpráva, pokud je správné, pak uložíme obrázek do proměnné.

5.4.2 Properties

Obsahuje několik tříd tlačítek, labelů...které tvoří dané obrazovky.

Tlačítka jsou implementované pomocí tříd `StartButton(Button)`, `EndButton(Button)`, `BackButton(Button)`, `ClosedButton(Button)`, `SendButton(Button)`.

Nadpis je implementovaná pomocí třídy `Title(QLabel)` a url box pomocí třídy `UrlBox(QLineEdit)` obsahující metody `load_img()` která načte do programu obrázek a metoda `write_wrong_url_message()` napíše do url_boxu chybovou zprávu v případě chybného zadaného url.

5.5 build_classifier.py

Soubor sloužící k sestavení a trénování konvoluční neuronové sítě.

5.5.1 Vstup

Uživatel si může zvolit několik možností trénování našeho modelu, které buď slouží k trénování kompletního modelu nebo prohledávání nějakých parametrů a vybírání optimální hodnoty daného parametru. Vstupní parametry jsou:

type_model: Uživatel zadá na příkazovém řádku `build_classifier.py -train_mode type_model`, čímž se model na 5 epochách natrénuje a vypíše průběh trénování jednotlivých typů modelu.

type_optimizer: Uživatel zadá na příkazovém řádku `build_classifier.py -train_mode type_optimizer`, čímž se model na 5 epochách natrénuje a vypíše průběh trénování jednotlivých typů optimizéru.

lr_momentum: Uživatel zadá na příkazovém řádku `build_classifier.py -train_mode lr_momentum`, čímž se model na 5 epochách natrénuje a vypíše průběh trénování dvojice lr,momentum hodnot.

weight_decay: Uživatel zadá na příkazovém řádku `build_classifier.py -train_mode weight_decay`, čímž se model na 5 epochách natrénuje a vypíše průběh trénování jednotlivých hodnot parametru.

nesterov: Uživatel zadá na příkazovém řádku `build_classifier.py -train_mode nesterov`, čímž se model na 5 epochách natrénuje a vypíše průběh trénování jednotlivých hodnot parametru.

full_fit: Uživatel zadá na příkazovém řádku `build_classifier.py -train_mode full_fit`, čímž se model na 20 epochách kompletně natrénuje.

Všechny tyto argumenty zaznamenává třída `TrainModes(Enum)`.

5.5.2 Příprava datasetu

Jeden z parametrů je počet kopií snímků v trénovacím datasetu, které vytváříme pomocí třídy `class Dataset(torch.utils.data.Dataset)` tvořící nový dataset s tímto počtem kopií.

Trénovací data obsahují transformace, které modifikují obrázek pro zobecnění trénovacích dat.

5.5.3 Trénování, validace a testování

Vytvořená třída `EpochsEvaluator` zaznamenává přesnosti a ztráty po každé epoše jednotlivých trénovacích, testovacích a validačních dat, kde přesnost se počítá pomocí matice záměn.

Třída obsahuje metody `add_loss()` a `add_accuaries()` přidávající ztrátu, přesnost do `self.losses` a `self.accuaries`.

5.5.4 Načítání dat

Data nahráváme do programu pomocí tříd, které dědí od `Data(ABC)` obsahující `dataloader`. Trénovací data nahrává třída `Train(Data)`, validační `Validate(NotAugmentedData)` a testovací `Test(NotAugmentedData)`, kde validační a testovací třída dědí od třídy `NotAugmentedData(Data)`, která načítá data rovnou z `ImageFolderu`, kterou nevyužíváme u trénovacích dat kvůli vytváření nového datasetu s kopiemi.

5.5.5 Řešení Hyperparametrů

Pro výběr správných parametrů trénování modelu používáme abstraktní třídu `Parameter(ABC)`, která je dědena třídami `TypeModel(Parameter)` hledající nejvhodnější konvoluční model pro naši úlohu.

`TypeOptimizer(Parameter)` hledá nejvhodnější optimizer na již vybraném modelu.

`lrMomentum(Parameter)` hledá dvojici parametrů `lr, momentum` na vybraném modelu a optimizeru,

`WeightDecay(Parameter)` hledá nejlepší řešení parametru `weight_decay` pro vybraný model, optimizer, `lr` a momentum.

`Nesterov(Parameter)`, která hledá nejlepší parametr pro již nalezených parametrů. Každá z těchto tříd obsahuje `searching()` override metodu, která prohledává daný parametry, tedy `searching()` je abstraktní metoda třídy `Parameter(ABC)`.

`Copies(Parameter)`, hledá optimální počet kopií obrázků.

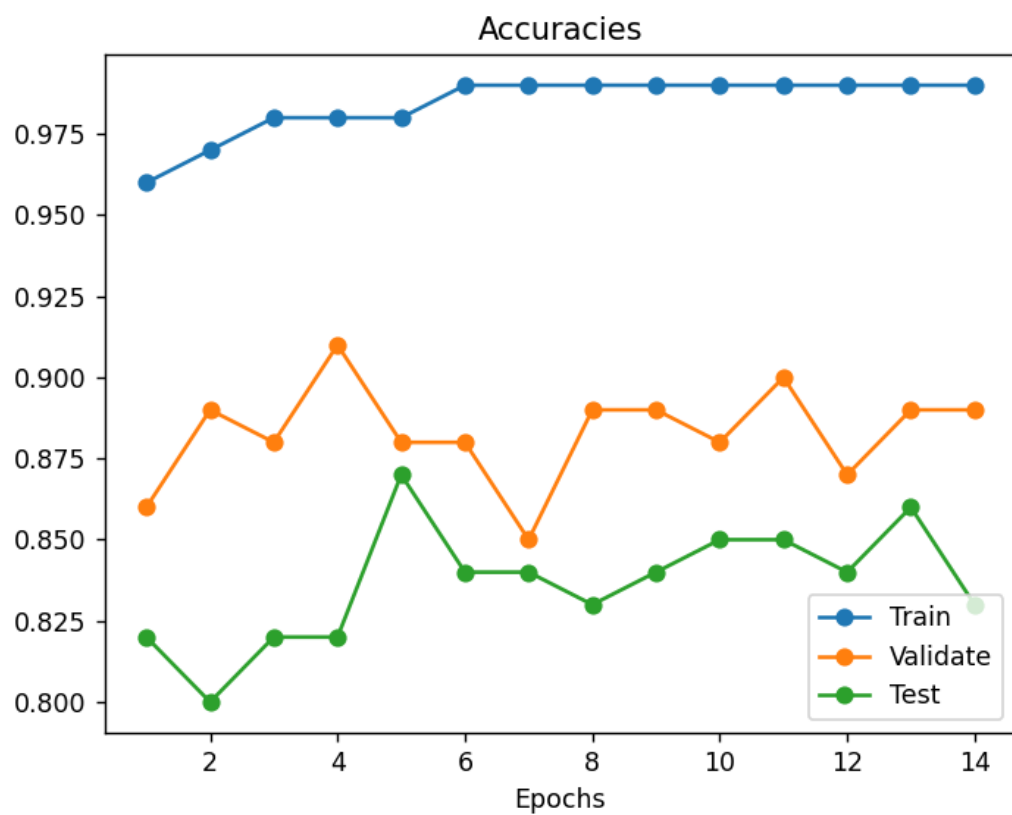
Třída `Parameter(ABC)` obsahuje abstraktní metodu `searching()`, která je v každé zmíněné třídě přepsána.

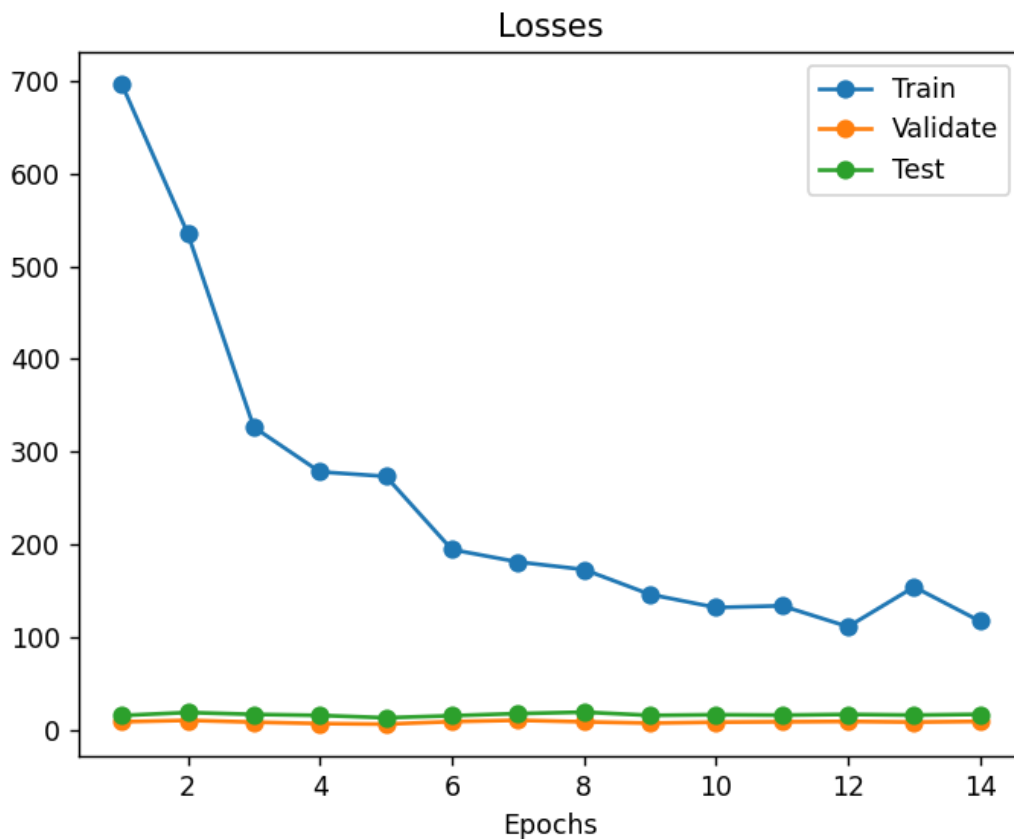
5.5.6 Vizualizace trénování modelu

Třída `class Visualisator` obsahuje metodu `plot()`, který vykresluje do grafu průběh trénování modelu na trénovacích, validačních a testovacích datech.

5.5.7 Fitting model

Třída `Model(nn.Module)` obsahuje v konstruktoru uložený typ modelu, vybraný optimizer a počet tříd do kterých budeme vstupy klasifikovat. Trénování modelu je uzpůsobené architektuře modelu a povoleným vrstvám pro klasifikaci. Třída obsahuje klasické metody `train_epoch()` a `eval_epoch()` pro trénování a vyhodnocování modelu. Metoda `fit()` slouží k trénování modelu na všech fázích -trénování, validaci a testování.





Kde byl uložený model vykazující nejlepší přesnost na validační sadě a příslušná epocha je:

Train Accuracy	Validate Accuracy	Test Accuracy
0.98	0.91	0.82

5.5.8 Zpracování vstupu

Vstupní argumenty jsou zpracovány pomocí třídy `InputProcessorTraining(InputProcessor)`, která obsahuje override metodu `process()` zpracovávající vstupní uživatelský argument z příkazové řádky.

Metoda `build()` slouží k zahájení programu sloužící k čtení vstupu a trénování modelu.

6 Zdroje

https://docs.pytorch.org/tutorials/beginner/data_loading_tutorial.html

<https://www.geeksforgeeks.org/machine-learning/confusion-matrix-machine-learning/>

<https://www.codegenes.net/blog/pytorch-detach-cpu-numpy/>

<https://numpy.org/doc/stable/reference/generated/numpy.trace.html>

<https://www.geeksforgeeks.org/python/enum-intenum-inpython/#:~:text=With%20the%20help%20of%20enum.I>

<https://pypi.org/project/overrides/>
<https://stackoverflow.com/questions/75440/how-do-i-get-the-string-with-name-of-a-classom>
<https://www.geeksforgeeks.org/python/line-chart-in-matplotlib-python/>
<https://www.geeksforgeeks.org/python/how-to-draw-a-circle-using-matplotlib-in-python>
<https://www.geeksforgeeks.org/python/matplotlib-pyplot-xticks-in-python>
https://matplotlib.org/stable/gallery/color/named_colors.html
https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.plot.html
https://docs.pytorch.org/tutorials/beginner/transfer_learning_tutorial.html
<https://docs.pytorch.org/tutorials/beginner/introyt/trainingyt.html>
https://docs.ray.io/en/latest/train/examples/pytorch/pytorch_resnet_finetune.html
<https://docs.pytorch.org/docs/stable/generated/torch.cat.html>
<https://www.datacamp.com/tutorial/tqdm-python>
<https://tqdm.github.io/docs/tqdm/>
<https://docs.pytorch.org/vision/stable/models.html>
<https://docs.pytorch.org/docs/stable/optim.html>
https://docs.pytorch.org/tutorials/beginner/transfer_learning_tutorial.html
<https://docs.pytorch.org/docs/stable/optim.html>
<https://stackoverflow.com/questions/6784084/how-to-pass-arguments-to-functions-by-the-click-of-button-in-pyqt>
<https://stackoverflow.com/questions/75465473/how-to-make-round-edges-for-the-main-window-in-pyqt?utm>
<https://forum.qt.io/topic/113585/change-qpushbutton-opacity-on-hover-and-pressed-states/>
<https://doc.qt.io/qt-6/stylesheet-syntax.html#selector-types>
<https://doc.qt.io/qtforpython-6/PySide6/QtWidgets/QWidget.html#PySide6.QtWidgets.QWidget.setStyleSheet>
<https://doc.qt.io/qtforpython-6/PySide6/QtWidgets/QLineEdit.html#PySide6.QtWidgets.PySide6.QtWidgets.QLineEdit.setStyleSheet>
<https://www.youtube.com/watch?v=PGBlE4B0UyQ>
https://groups.google.com/g/python_inside_maya/c/Z-Jh_uzPGe4
https://docs.pytorch.org/tutorials/beginner/data_loading_tutorial.html
<https://www.geeksforgeeks.org/machine-learning/confusion-matrix-machine-learning/>
<https://www.codegenes.net/blog/pytorch-detach-cpu-numpy/>
<https://numpy.org/doc/stable/reference/generated/numpy.trace.html>
<https://www.geeksforgeeks.org/python/enum-intenum-in-python/#:~:text=With%20the%20help%20of%20enum.>
<https://pypi.org/project/overrides/>

<https://stackoverflow.com/questions/75440/how-do-i-get-the-string-with-name-of-a-classom>
<https://www.geeksforgeeks.org/python/line-chart-in-matplotlib-python/>
<https://www.geeksforgeeks.org/python/how-to-draw-a-circle-using-matplotlib-in-python>
<https://www.geeksforgeeks.org/python/matplotlib-pyplot-xticks-in-python>
https://matplotlib.org/stable/gallery/color/named_colors.html
https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.plot.html
https://docs.pytorch.org/tutorials/beginner/transfer_learning_tutorial.html
<https://docs.pytorch.org/tutorials/beginner/introyt/trainingyt.html>
https://docs.ray.io/en/latest/train/examples/pytorch/pytorch_resnet_finetune.html
<https://docs.pytorch.org/docs/stable/generated/torch.cat.html>
<https://www.datacamp.com/tutorial/tqdm-python>
<https://tqdm.github.io/docs/tqdm/>
<https://docs.pytorch.org/vision/stable/models.html>
<https://docs.pytorch.org/docs/stable/optim.html>
https://docs.pytorch.org/tutorials/beginner/transfer_learning_tutorial.html
<https://docs.pytorch.org/docs/stable/optim.html>
https://docs.pytorch.org/tutorials/beginner/data_loading_tutorial.html
<https://www.geeksforgeeks.org/machine-learning/confusion-matrix-machine-learning/>
<https://www.codegenes.net/blog/pytorch-detach-cpu-numpy/>
<https://numpy.org/doc/stable/reference/generated/numpy.trace.html>
<https://www.geeksforgeeks.org/python/enum-intenum-in-python/#:~:text=With%20the%20help%20of%20enum.>
<https://pypi.org/project/overrides/>
<https://stackoverflow.com/questions/75440/how-do-i-get-the-string-with-name-of-a-classom>
<https://www.geeksforgeeks.org/python/line-chart-in-matplotlib-python/>
<https://www.geeksforgeeks.org/python/how-to-draw-a-circle-using-matplotlib-in-python>
<https://www.geeksforgeeks.org/python/matplotlib-pyplot-xticks-in-python>
https://matplotlib.org/stable/gallery/color/named_colors.html
https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.plot.html
https://docs.pytorch.org/tutorials/beginner/transfer_learning_tutorial.html
<https://docs.pytorch.org/tutorials/beginner/introyt/trainingyt.html>
https://docs.ray.io/en/latest/train/examples/pytorch/pytorch_resnet_finetune.html
<https://docs.pytorch.org/docs/stable/generated/torch.cat.html>

<https://www.datacamp.com/tutorial/tqdm-python>

<https://tqdm.github.io/docs/tqdm/>

<https://docs.pytorch.org/vision/stable/models.html>

<https://docs.pytorch.org/docs/stable/optim.html>

https://docs.pytorch.org/tutorials/beginner/transfer_learning_tutorial.html

<https://docs.pytorch.org/docs/stable/optim.html>