

**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY**

Evidenčné číslo: FEI-5382-104537

**POROVNANIE METÓD UMELEJ INTELIGENCIE NA
ROZPOZNÁVANIE RUKOU PÍSANÝCH ČÍSLIC**

BAKALÁRSKA PRÁCA

2022

Filip Mikuš

**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY**

Evidenčné číslo: FEI-5382-104537

**POROVNANIE METÓD UMELEJ INTELIGENCIE NA
ROZPOZNÁVANIE RUKOU PÍSANÝCH ČÍSLIC**
BAKALÁRSKA PRÁCA

Študijný program: Aplikovaná informatika
Názov študijného odboru: 9.2.9 Aplikovaná informatika
Školiace pracovisko: Ústav informatiky a matematiky
Vedúci záverečnej práce: Ing. Eugen Antal, PhD.

Bratislava 2022

Filip Mikuš



ZADANIE BAKALÁRSKEJ PRÁCE

Študent: **Filip Mikuš**

ID študenta: 104537

Študijný program: aplikovaná informatika

Študijný odbor: informatika

Vedúci práce: Ing. Eugen Antal, PhD.

Vedúci pracoviska: Dr. rer. nat. Martin Drozda

Miesto vypracovania: Ústav informatiky a matematiky

Názov práce: **Porovnanie metód umelej inteligencie na rozpoznávanie rukou písaných čísl**

Jazyk, v ktorom sa práca vypracuje: slovenský jazyk

Špecifikácia zadania:

Historické šifry (dobové nomenklátry) používané od 16. storočia začali používať čísla na reprezentáciu šifrovaného textu postupne, v čoraz väčšom pomere. Od 18. storočia predstavujú čísla značnú časť šifrovaného textu. Táto práca sa zameriava na rozpoznávania (automatizovanú počítačovú transkripciu) čísel v šifrovaných dokumentoch. Cieľom práce je preskúmať a porovnať existujúce metódy a nástroje slúžiace na rozpoznávanie rukou písaných textov (HTR), konkrétnie čísl. Výsledkom má byť implementácia rôznych prístupov a aplikácia slúžiaca na vykonanie rozpoznávania čísl a na vyhodnotenie rôznych experimentov. Dosiahnuté výsledky je potrebné systematicky porovnať aj s existujúcimi nástrojmi, natrénovanými neurónovými sieťami a inými dostupnými riešeniami. Predpokladá sa využitie umelej inteligencie a metód strojového učenia. Výskumná úloha zapojená do grantu VEGA 2/0072/20 Moderné metódy spracovania šifrovaných archívnych dokumentov.

Úlohy:

1. Naštudujte ako funguje nomenklátor a ako vyzerajú nomenklátorové kľúče v historických dokumentoch.
2. Naštudujte metódy HTR a rozpoznávania rukou písaných čísl.
3. Analyzujte dostupné riešenia.
4. Navrhnite aplikáciu podľa zadania.
5. Naprogramujte a vyhodnoťte riešenie.

Zoznam odbornej literatúry:

1. Antal, E. – Grošek, O. *Moderná kryptoanalýza klasických šifier: dát. obhaj. 26.4.2017, č. ved. odb. 9-2-9*. Dizertačná práca. Bratislava : 2017. 133 s.
2. Antal, E. – Zajac, P. Analýza Raben Hauptovo zašifrovaného dopisu. *Crypto-World*, 15. s. 9–17.
3. Antal, E. – Zajac, P. – Mírka, J. Solving a mystery from the thirty years' war: Karel Raben Haupt ze Sučé's encrypted letter to landgravine Amalie Elisabeth. In Dahlke, C. *HistoCrypt 2021*. Linköping: University Electronic Press, 2021, s. 12–24. ISBN 978-91-7929-026-9.
4. Beáta Megyesi. Transcription of Historical Ciphers and Keys. In Proceedings of the 3rd International Conference on Historical Cryptology, HistoCrypt 2020, pages 106 – 115.
5. Elonka Dunin and Klaus Schmeh. Codebreaking: A Practical Guide. Robinson. Great Britain. 2020.

Termín odovzdania bakalárskej práce:

03. 06. 2022

Dátum schválenia zadania bakalárskej práce:

19. 05. 2022

Zadanie bakalárskej práce schválil:

Dr. rer. nat. Martin Drozda – garant študijného programu

SÚHRN

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Študijný program:

Aplikovaná informatika

Autor:

Filip Mikuš

Bakalárská práca:

Porovnanie metód umelej inteligencie na rozpoznanie rukou písaných číslíc

Vedúci záverečnej práce:

Ing. Eugen Antal, PhD.

Miesto a rok predloženia práce:

Bratislava 2022

Digitalizácia historických dokumentov je fenomén, ktorý umožňuje uchovávanie historických prameňov na digitálnych médiach. Nezanedbatelnú časť takýchto dokumentov tvoria aj šifrované dokumenty špecifickej štruktúry, nomenklátory. Šifrované dokumenty sú často podrobene komplexným kryptologickým analýzam, s čím je spojená snaha o ich automatizovanú transkripciu.

Cieľom našej práce bolo oboznámiť sa so štruktúrou a podobou nomenklátorov, naštudovať, analyzovať a porovnať techniky na predspracovanie obrazu a rozpoznávanie rukou písaného textu, a zhromaždiť, porovnať a analyzovať vyhovujúce dátové množiny. V praktickej časti sme mali za úlohu podľa získaných poznatkov implementovať jednoduchú aplikáciu na predspracovanie obrazu a vykonať porovnanie rôznych implementácií a konfigurácií modelov na rozpoznávanie rukou písaných číslíc.

Kľúčové slová: nomenklátor, digitalizácia historických dokumentov, predspracovanie obrazu, rozpoznávanie rukou písaných číslíc, strojové učenie, počítačové videnie

ABSTRACT

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF ELECTRICAL ENGINEERING AND INFORMATION TECHNOLOGY

Study Programme:	Applied Informatics
Author:	Filip Mikuš
Bachelor's thesis:	Comparison of artificial intelligence methods for handwritten number recognition
Supervisor:	Ing. Eugen Antal, PhD.
Place and year of submission:	Bratislava 2022

Digitization of historical documents is a phenomenon that allows the conservation of historical documents on digital media. Significant part of preserved documents also form encrypted documents with a specific structure - nomenclators. Encrypted historical documents are often transcribed to digital representation. This transcribed documents are then often part of complex cryptanalysis, too.

The aim of this work was to become familiar with structure and form of nomenclators, study, analyse and compare image preprocessing and handwritten text recognition techniques, and collect, analyse and compare suitable datasets. In the second part of our work, we implement a simple image preprocessing application and compare different implementations and configurations of handwritten number recognition systems.

Keywords: nomenclator, digitization of historical documents, image preprocessing, handwritten number recognition, machine learning, computer vision

Vyhľásenie

Ja, dolu podpísaný Filip Mikuš čestne vyhlasujem, že som bakalársku prácu Porovnanie metód umelej inteligencie na rozpoznávanie rukou písaných čísl na základe poznatkov získaných počas štúdia a informácií z dostupnej literatúry uvedenej v práci.

Uvedenú prácu som vypracoval pod vedením Ing. Eugena Antala, PhD.

Bratislava, dňa 3.6.2022

.....

podpis autora

Podčakovanie

Chcem sa podčakovať vedúcemu záverečnej práce, ktorým bol Ing. Eugen Antal, PhD., za odborné vedenie, rady a pripomienky, ktoré mi pomohli pri vypracovaní tejto bakalárskej práce.

Obsah

Úvod	1
1 Analýza a rozpoznávanie obrazových dokumentov	2
1.1 Historické dokumenty	2
1.2 Šifrované historické dokumenty	3
1.3 Nomenklátor	4
2 Systémy na rozpoznávanie rukou písaného textu a znakov	6
2.1 HCR - rozpoznávanie rukou písaných znakov	7
2.2 HTR - rozpoznávanie rukou písaného textu	7
3 Štruktúra HCR systému	9
3.1 Predspracovanie	9
3.1.1 Grayscale	10
3.1.2 Normalizácia	10
3.1.3 Binarizácia	11
3.1.4 Bitreverse	11
3.1.5 Skeletonizácia	12
3.1.6 Morfologické operácie	12
3.2 Lokalizácia	13
3.2.1 Sliding Window	14
3.2.2 R-CNN	14
3.2.3 Fast R-CNN	15
3.2.4 Faster R-CNN	17
3.2.5 YOLO	18
3.3 Klasifikácia	19
3.3.1 MLP - viacvrstvová perceptrónová neurónová siet	20
3.3.2 CNN - konvolučná neurónová siet	22
3.4 Segmentácia	25
3.4.1 FCN - plne konvolučná siet	26
4 Analýza riešenia	27
4.1 Voľba nástrojov	27

4.2	Dostupné detekčné modely	27
4.2.1	Detekčný model Mask R-CNN	27
4.2.2	COCO formát datasetu	27
4.2.3	Detekčný model YOLO v5	28
4.2.4	YOLO formát datasetu	28
4.3	Dostupné datasety rukou písaných číslí	29
4.3.1	ARDIS	30
4.3.2	Chars74K	32
4.3.3	DIDA	33
4.3.4	MNIST	34
4.3.5	EMNIST	35
4.3.6	SEMEION	37
4.3.7	USPS	38
5	Implementácia riešenia	40
5.1	Volba nástrojov	40
5.2	Aplikácia na predspracovanie obrazu	41
5.2.1	Metódy predspracovania	41
5.2.2	Grafické používateľské rozhranie	43
5.3	Detekčný model Mask R-CNN	45
5.3.1	ARDIS dataset	45
5.3.2	VEGA dataset	47
5.3.3	VEGA-Cropped dataset	49
5.3.4	Zhrnutie	51
5.4	Detekčný model YOLO v5	53
5.4.1	ARDIS dataset	53
5.4.2	VEGA dataset	55
5.4.3	VEGA-Cropped dataset	56
5.4.4	Zhrnutie	57
5.5	Porovnanie a zhrnutie - Mask R-CNN, YOLO v5	59
Záver		60
Zoznam použitej literatúry		61
Prílohy		I

A Štruktúra elektronickej prílohy s praktickým výstupom	II
B Štruktúra COCO JSON anotácií	IV
C Mask R-CNN - konfigurácia trénovania a inferencie na datasete ARDIS	V
D Mask R-CNN - konfigurácia trénovania a inferencie na datasete VEGA	VI
E Mask R-CNN - konfigurácia trénovania a inferencie na datasete VEGA-Cropped	VII
F Metriky na vyhodnotenie úspešnosti detekčných systémov	VIII
G Vyhodnotenie úspešnosti detekčných systémov	IX

Zoznam obrázkov a tabuliek

Obrázok 1	Nomenklátorový klúč. [3]	5
Obrázok 2	Schéma systému na rozpoznávanie rukou písaných znakov alebo textu. [7]	6
Obrázok 3	Schéma procesov systému na rozpoznávanie rukou písaných znakov alebo textu.	9
Obrázok 4	Schéma aplikácie grayscale filtra.	10
Obrázok 5	Schéma aplikácie normalizácie.	10
Obrázok 6	Schéma aplikácie binarizácie.	11
Obrázok 7	Schéma aplikácie bitreverse.	11
Obrázok 8	Schéma aplikácie skeletonizácie.	12
Obrázok 9	Schéma aplikácie diletácie.	12
Obrázok 10	Schéma aplikácie erózie.	13
Obrázok 11	Schéma procesu lokalizácie.	13
Obrázok 12	Schéma procesu lokalizácie: Sliding Window.	14
Obrázok 13	Schéma procesu lokalizácie: R-CNN.	15
Obrázok 14	Schéma procesu lokalizácie: Fast R-CNN. [18]	16
Obrázok 15	Schéma procesu lokalizácie: Faster R-CNN. [19]	17
Obrázok 16	Schéma RPN siete. [19]	18
Obrázok 17	Schéma procesu lokalizácie a klasifikácie: YOLO.	19
Obrázok 18	Schéma procesu klasifikácie.	19
Obrázok 19	Schéma procesu klasifikácie: MLP.	20
Obrázok 20	Schéma umelého neurónu.	21
Obrázok 21	Schéma procesu klasifikácie: CNN.	23
Obrázok 22	Schéma konvolúcie. [24]	23
Obrázok 23	Schéma poolingu. [25]	24
Obrázok 24	Schéma procesu segmentácie.	25
Obrázok 25	Schéma procesu segmentácie: FCN.	26
Obrázok 26	Ukážka vzoriek datasetu: ARDIS IV.	31
Obrázok 27	Graf početnosti vzoriek: ARDIS IV.	31
Obrázok 28	Ukážka vzoriek datasetu: Chars74K.	32
Obrázok 29	Graf početnosti vzoriek: Chars74K.	32
Obrázok 30	Ukážka vzoriek datasetu: DIDA.	33

Obrázok 31	Graf početnosti vzoriek: DIDA.	33
Obrázok 32	Ukážka vzoriek datasetu: MNIST.	34
Obrázok 33	Graf početnosti vzoriek: MNIST.	35
Obrázok 34	Ukážka vzoriek datasetu: EMNIST.	36
Obrázok 35	Graf početnosti vzoriek: EMNIST.	36
Obrázok 36	Ukážka vzoriek datasetu: SEMEION.	37
Obrázok 37	Graf početnosti vzoriek: SEMEION.	38
Obrázok 38	Ukážka vzoriek datasetu: USPS.	38
Obrázok 39	Graf početnosti vzoriek: USPS.	39
Obrázok 40	Ukážka aplikácie na predspracovanie.	44
Obrázok 41	Ukážka masiek dátovej vzorky: ARDIS.	45
Obrázok 42	Ukážka inferencie Mask R-CNN: ARDIS.	46
Obrázok 43	Ukážka masiek dátovej vzorky: VEGA.	48
Obrázok 44	Ukážka inferencie Mask R-CNN: VEGA.	48
Obrázok 45	Ukážka masiek dátovej vzorky: VEGA-Cropped.	50
Obrázok 46	Ukážka inferencie Mask R-CNN: VEGA-Cropped.	50
Obrázok 47	Ukážka inferencie YOLO v5: ARDIS.	54
Obrázok 48	Ukážka inferencie YOLO v5: VEGA.	55
Obrázok 49	Ukážka inferencie YOLO v5: VEGA-Cropped.	57
Tabuľka 2	Zoznam datasetov a ich charakteristik	30
Tabuľka 3	Úspešnosť detekcie a klasifikácie Mask R-CNN: ARDIS	47
Tabuľka 4	Úspešnosť detekcie a klasifikácie Mask R-CNN: VEGA.	49
Tabuľka 5	Úspešnosť detekcie a klasifikácie Mask R-CNN: VEGA-Cropped.	51
Tabuľka 6	Úspešnosť detekcie a klasifikácie YOLO v5: ARDIS	54
Tabuľka 7	Úspešnosť detekcie a klasifikácie YOLO v5: VEGA.	56
Tabuľka 8	Úspešnosť detekcie a klasifikácie YOLO v5: VEGA-cropped.	57

Zoznam skratiek

ADAM	Adaptive Moment Estimation
AI	Artificial Intelligence
ARDIS	Arkiv Digital Sweeden dataset
BB	Bounding Box
BP	Backpropagation
Chars74K	Dataset of Characters 74 000
CNN	Convolutional Neural Network
COCO	Common Objects in Context dataset
CV	Computer Vision
DIDA	Digits Dataset
DT	Decision Tree
EMNIST	Extended MNIST
Fast R-CNN	Fast Region based Convolutional Neural Network
Faster R-CNN	Faster Region based Convolutional Neural Network
FCN	Fully Convolutional Network
GD	Gradient Descent
HCR	Handwritten Character Recognition
HTR	Handwritten Text Recognition
KNN	K-Nearest Neighbors
Mask R-CNN	Mask Region based Convolutional Neural Network
ML	Machine Learning
MLP	Multilayer Perceptron
MNIST	Modified National Institute of Standards and Technology dataset
NBC	Naive Bayes Classifier
NN	Neural Network
OCR	Optical Character Recognition
OTR	Optical Text Recognition
R-CNN	Region based Convolutional Neural Network
ReLU	Rectified Linear Unit

ROI	Region Of Interest
RPN	Region Proposal Network
SEMEION	Semeion Research Center of Sciences of Communication dataset
SGD	Stochastic Gradient Descent
SSD	Single Shot Detector
SVM	Support Vector Machine
USPS	United States Postal Service dataset
YOLO	You Only Look Once

Úvod

Už začiatkom nového milénia s rozmachom informačných technológií začala snaha o digitalizáciu historických dobových dokumentov a prameňov. Dokumenty a pramene okrem informačného charakteru často plnili úlohu komunikačného média, čo viedlo k ochrane citlivých údajov a informácií pomocou aplikácie rôznych techník utajovania a šifrovania textu medzi ktoré patria aj nomenklátory. Z dôvodu snahy o analýzu takýchto dokumentov je potrebná ich transkripcia do digitálnej podoby. Automatizovaná transkripcia takýchto dokumentov by totiž mohla proces analýzy značne ulahčiť, zefektívniť a zrýchliť. V našej práci sme sa konkrétnie zamerali na detekciu a rozpoznávanie číslic, napäťko práve tie tvoria nezanedbateľnú časť dobových historických dokumentov.

V prvej časti našej práce sme zmapovali a preštudovali danú problematiku. Následne sme získané poznatky využili pri implementácii jednoduchej aplikácie na predspracovanie obrazových dát. Vzhľadom na fakt, že väčšina systémov na rozpoznávanie objektov funguje na princípe počítačového videnia a strojového učenia, bolo nutné zozbierať, porovnať a analyzovať volne dostupné datasety, ktoré by boli vhodné na trénovanie systémov na rozpoznávanie rukou písaných číslic. Po zhromaždení vhodných dátových množín sme vykonali porovnanie rôznych konfigurácií rozdielnych detekčných systémov, ktorých výsledky sme systematicky vyhodnotili a porovnali.

Vzhľadom na skutočnosť, že stále neexistuje univerzálny prístup detekcie textu pre rôzne scenáre využitia, je tento typ porovnania relevantnou a aktuálnou problematikou.

1 Analýza a rozpoznávanie obrazových dokumentov

Analýza a rozpoznávanie obrazových dokumentov predstavuje extrakciu a interpretáciu znalostí a informácií z obrazových súborov, ktoré zachytávajú fyzické dokumenty v digitálnej forme. Prvé systémy na optické čítanie znakov boli vyvinuté už v 70. rokoch 20. storočia, no progres vo vývoji a výskume v tejto oblasti nastal až s príchodom technológie skenera dokumentov v roku 1980. V tomto čase však bola väčšina systémov a scenárov využitia zameraná na digitalizáciu, uchovávanie a interpretáciu tlačených kancelárskych dokumentov a materiálov. [1]

V súčasnej dobe sa systémy na uchovávanie, zhotovenie alebo interpretáciu obrazových dokumentov stali súčasťou každodenného života v pracovnej, ale aj súkromnej sfére. S rýchlym vývojom a širokým využitím týchto systémov sa však objavili nové oblasti využitia, ktoré však so sebou prinášajú nové výzvy a problémy, ktoré je nutné pre optimálne fungovanie prekonat a vyriesiť. [1]

1.1 Historické dokumenty

Jednu z týchto oblastí tvorí analýza a rozpoznávanie historických obrazových dokumentov. Už začiatkom nového milénia vo veľkom začali snahy a pokusy o digitalizáciu historických dokumentov. Hlavnými dôvodmi tohto konania bolo [1]:

- zachovanie kultúrneho dedičstva,
- zastavenie dodatočnej degradácie dokumentov,
- zníženie a minimalizovanie nákladov na konzerváciu dokumentov,
- pridanie možnosti hľadania v obsahu dokumentov,
- sprístupnenie dokumentov pre pospolitú verejnosť.

V praxi sa však len veľmi zriedka stretнемe s digitalizovanými dokumentami s plnou transkripciou obsahu. Dôvodom je z majoritnej časti nutnosť manuálnej transkripcie obsahu digitalizovaných dokumentov, čo tento proces značne spomaľuje. [1]

So silným rozmachom informatizácie, rýchlym vývojom výpočtovej techniky a informatiky ako takej sa však už dlho očakávalo širšie využitie nových prostriedkov na tieto činnosti v súvislosti s historickými dokumentmi. Donedávna sa však plne automatizovaná transkripcia, vyhľadávanie a iné činnosti analýzy dokumentov vzdali ako náročné, miestami až krajne utopické.

Medzi hlavné problémy patrili:

- rôzne štruktúry dokumentov,
- rôzne štýly a techniky písania,
- rôznorodosť rukopisov,
- stav dokumentu,
- kvalita digitalizácie dokumentu.

Rôzne kombinácie týchto problémov teda predstavujú komplexný ľahko riešiteľný problém.

S rýchlym vývojom hardvéru a nových algoritmov sa však vyššie spomínané aktivity stali s veľmi veľkou pravdepodobnosťou uskutočniteľné a prekážky prekonateľné, niektoré už v súčasnosti, ostatné v blízkej budúcnosti. Obzvlášť významný smer v informatike, ktorý svojím rýchlym a dynamickým vývojom prispieva k riešeniu problémov a výziev v tejto oblasti je umelá inteligencia, konkrétnie techniky a metódy strojového učenia (ML), respektívne počítačového videnia (CV).

1.2 Šifrované historické dokumenty

V súvislosti s rozpoznávaním a hlavne následnou analýzou historických obrazových dokumentov je potrebné spomenúť, že v histórii ľudskej komunikácie bol už od nepamäti kladený dôraz na ochranu samotných správ, citlivých informácií a celej komunikácie. Správy boli chránené pomocou šifier rôzneho typu a prevedenia. Dokumenty mali teda odlišnú formu od bežných dokumentov, často avšak bola zachovaná štruktúra daná konkrétnou šifrou.

Pojmom šifrovanie nazývame metodiku na prevod informácie z čitateľnej podoby (otvorený text) na nečitateľnú podobu (zatvorený text). Dešifrovanie je potom metodika na opačný prevod, teda z nečitateľnej podoby (zatvorený text) na čitateľnú podobu (otvorený text). Ich skúmaním sa zaoberá vedný odbor kryptológia. Kryptológia sa delí na kryptografiu, ktorá sa zameriava na návrh systémov utajenia a na kryptoanalýzu, ktorá sa zameriava na narušenie systémov utajenia. Jedným z takýchto kryptosystémov sú nomenklátory. [2, 3]

Z tejto skutočnosti implicitne vyplýva, že aj v oblasti kryptológie je kladený dôraz na automatizované rozpoznávanie historických dokumentov za účelom následnej efektívnejšej analýzy, repsectívne kryptoanalýzy.

1.3 Nomenklátor

Nomenklátor predstavuje kryptosystém patriaci do skupín klasických a substitučných šifier. Tento druh šifry pravdepodobne vznikol rozšírením jednoduchej/homofónnej substitúcie o kódové knihy, ktorými boli v správach šifrované osobné údaje panovníkov a iných dobových predstaviteľov. Ich využitie v značnej miere datujeme približne od roku 1400 do roku 1850. [3, 4]

Kategória klasických šifier je charakteristická svojím jednoduchou uskutočniteľným šifrovaním a dešifrovaním pre použitie v rôznych podmienkach a situáciach. Šifrovanie má byť vykonateľné len pomocou pera, papiera alebo jednoduchých (elektro) mechanických pomôcok. [2]

Substitučné kryptosystémy fungujú na princípe nahradenia (substitúcie) jednotlivých znakov otvoreného textu pri dodržaní vopred stanovených pravidiel. Každý znak pôvodného textu je postupne jednotlivo prevedený z abecedy otvoreného textu na príslušný znak abecedy zatvoreného textu. V nomenklátoroch sú tieto pravidlá uvedené v takzvaných nomenklátorových klúčoch. [2, 3]

Z pohľadu kriptografie nomenklátori pozostávajú z viacerých jednoduchých substitučných šifier, ktoré sú v tomto kryptosystéme kombinované. Klasický nomenklátor obsahuje [3, 5, 6]:

- jednoduchú/homofónnu substitúciu,
- polygramovú substitúciu,
- kódovú knihu,
- klamače.

Nomenklátori môžu byť rôznych foriem a štruktúr. Najjednoduchšie formy pozostávajú len z jednoduchých substitúcií a mála kódových slov. Komplexnejšie už obsahujú komplikovanú homofónnu substitúciu, klamače a veľa kódových slov. Špecifickou charakteristikou tohto šifrovacieho systému je jeho pevná štruktúra, kde je klúč graficky oddelený od jeho ostatných častí. [3]

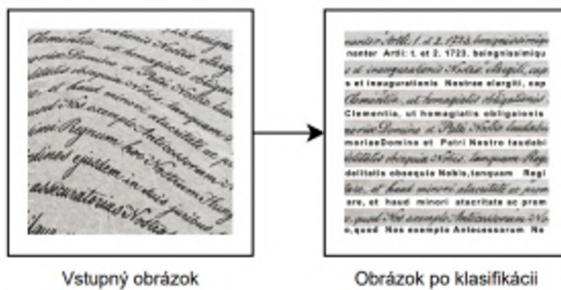
Vzhľadom na to, že nomenklátori môžu byť z veľkej časti tvorené číslami, sme sa v našej práci zamerali práve na rozpoznávanie rukou písaných čísl.

Pražské městské koncile v roce 1640. Knihy žádostí											
F	B	C	Z	E	F	G	H	I	K	L	M
1500	1600	1700	1800	1900	2000	2100	2200	2300	2400	2500	2600
62	63	72	77	82	87	92	97	102	107	112	117
63	65	73	78	83	88	93	98	103	108	113	118
64	67	74	79	84	89	94	99	104	109	114	119
65	70	75	80	86	90	95	100	105	110	115	120
66	71	76	91	94	96	102	106	111	116	121	126
<hr/>											
N	O	P	E	R	S	T	U	W	X	Y	Z
1400	1300	1200	1100	1000	900	800	700	600	500	400	300
57	52	47	42	37	32	27	22	17	12	7	11
58	53	48	43	38	33	28	23	18	13	8	2
59	54	49	44	39	34	29	24	19	14	9	3
60	55	50	45	40	35	30	25	20	15	10	5
61	56	51	46	41	36	31	26	21	16	6	1
<hr/>											
Matica											
a	b	c	d	e	f	g	h	i	k	l	m
aa	bb	cc	dd	ee	ff	gg	hh	ii	kk	ll	mm
uu	vv	ww	xx	yy	zz	pp	qq	rr	tt	uu	vv
xx	yy	zz	uu	vv	ww	pp	qq	rr	tt	uu	vv
<hr/>											
Pojíšek - - - - -	6000.										
Romig z Spaniag - - -	9000.										
Romig z Pomeránie - - -	10000.										
Romig z Uhřitv - - -	11000.										
Romig z Engelsem - - -	12000.										
Romig z Německem - - -	13000.										
<u>Don Cipk z Rodigo</u> - - -	14000.										
<u>František František z Jindřichy</u> - - -	15000.										
<u>Jozef Václav Oranž</u> - - -	16000.										
<u>Jan Matyáš Gerasol</u> - - -	17000.										
<u>Cardinal Karel z Lichtenštejna</u> - - -	18000.										
<u>Jan Václav z Lichtenštejna</u> - - -	19000.										
<u>Jan Matyáš z Lichtenštejna</u> - - -	20000.										
<u>Jan Matyáš z Lichtenštejna</u> - - -	21000.										
<u>František z Lichtenštejna</u> - - -	22000.										
<u>František z Lichtenštejna</u> - - -	23000.										
<u>František z Lichtenštejna</u> - - -	24000.										
<hr/>											
Hejtman - - - - -	25000.										
Lambrecht z Šternberka - - -	26000.										
Lambrecht z Šternberka - - -	27000.										
Čabuš z Čabuše - - - - -	28000.										
<u>František z Dubé</u> - - - - -	29000.										
<u>František z Dubé</u> - - - - -	30000.										
Hejtman - - - - -	31000.										
Lichtenštejn - - - - -	32000.										
Haibach - - - - -	33000.										
Lambrecht - - - - -	34000.										
<u>František z Dubé</u> - - - - -	35000.										
<u>František z Dubé</u> - - - - -	36000.										
<u>František z Dubé</u> - - - - -	37000.										
<u>František z Dubé</u> - - - - -	38000.										
Wittenberg - - - - -	39000.										
<u>František z Dubé</u> - - - - -	40000.										

Obr. 1: Nomenklátorový klúč. [3]

2 Systémy na rozpoznávanie rukou písaného textu a znakov

V praxi sa systémy na optické rozpoznávanie rukou písaných znakov alebo textu (OCR, OTR) používajú na automatizovanú transkripciu textu zhotoveného pomocou rukou písaných grafém do digitálnej reprezentácie. Vstupom tohto systému môže byť napríklad sken fyzického dokumentu, fotografia fyzického dokumentu, výstup z dotykovej obrazovky alebo iné média, ktoré zachytávajú alebo ktorých obsahom je rukou písaný text. Výstupom je digitálna reprezentácia vstupu. Forma výstupu môže byť rôzna a zavisí od účelu a zámeru použitia daného systému. Rozpoznávanie textu vo všeobecnosti je rozsiahly a robustný problém, nakoľko vstupy takého systému splňajú len veľmi vágne spoločné vlastnosti. Táto charakteristika problému úplne zamedzuje použitiu deterministického systému na jeho vyriešenie. Na riešenie problémov v tejto oblasti sa teda využívajú stochastické systémy, konkrétnie systémy a aplikácie umelej inteligencie (AI). Využívané sú metódy a algoritmy viacerých disciplín a smerov umelej inteligencie, prevažne sa ale jedná o počítačové videnie, respektívne strojové učenie. [1, 7, 8]



Obr. 2: Schéma systému na rozpoznávanie rukou písaných znakov alebo textu. [7]

Rozpoznávanie objektov je všeobecný opis rôznych úloh počítačového videnia, ktoré zahrňajú identifikáciu objektov na digitálnych fotografiách alebo obrázkoch. Spektrum zahrnuté v tomto pojme je velmi široké a zahŕňa napríklad techniky ako detekciu, segmentáciu, klasifikáciu a iné. [8]

Vo všeobecnosti sa systémy počítačového videnia snažia implementovať postupy, ktoré človek využíva pri vnímaní okolitého sveta zrakom. Tie mu následne umožňujú vnímať farby, tvary, svetlo a orientáciu. Z biologického hľadiska je cieľom počítačového videnia prísť s výpočtovými modelmi ľudského zrakového systému. Z hľadiska počítačovej implementácie sa počítačové videnie zameriava na vyhotovenie autonómnych systémov a algoritmov, ktoré by mohli vykonávať úlohy, ktoré vykonáva ľudský zrakový systém. Pri samotnej praktickej

realizácií sa využívajú rôzne techniky strojového učenia, napríklad stroje s podpornými vektormi (SVM), najčastejšie však neurónové siete (NN). [9]

Pri rozpoznávaní znakov a teda textu sa existujúce riešenia snažia analogicky nápodobniť proces čítania textu ľudskými bytostami, kde najprv ľudské oko lokalizuje jednotlivé znaky textu a následne ich ľudský mozog vdaka kognitívnym schopnostiam klasifikuje ako jednotlivé písmená, číslice alebo iné symboly.

V súčasnosti existuje niekoľko variant a typov systémov na rozpoznávanie textu a znakov, ako napríklad rozpoznávanie rukou písaných znakov (HCR), rozpoznávanie rukou písaného textu (HTR) a iné.

2.1 HCR - rozpoznávanie rukou písaných znakov

Metódy na rozpoznávanie rukou písaných znakov sú odvodenou podskupinou OCR systémov, ktoré vykonávajú optickú transkripciu vstupných dát reprezentujúcich médium obsahujúce rukou písaný text na výstupné dátá reprezentujúce znaky v digitálnej podobe, ktoré sa systému podarilo rozpoznať na vstupe. HCR patrí medzi detekčné algoritmy, ktoré lokalizujú objekty na vstupe pomocou ohraničujúceho rámčeka (BB) a následne tieto objekty klasifikuje do skupín, respektívne tried.

Objektmi na jednej inštancii výstupu tohto systému sú teda jednotlivé a samostatné písmena alebo znaky, čo v realite znamená, že systém nehľadá kontext medzi týmito znakmi a chápe ich ako samostatné objekty. Existuje niekoľko typov implementácií tohto prístupu, napríklad [1, 7, 8]:

- sliding window,
- region based convolutional neural networks (R-CNN, Fast R-CNN, Faster R-CNN, Mask R-CNN),
- you only look once (YOLO).

Nakoľko sa v našej práci venujeme rozpoznávaniu číslíc, budeme sa venovať aplikácií a porovnaniu metód rozpoznávanie rukou písaných znakov, teda HCR.

2.2 HTR - rozpoznávanie rukou písaného textu

Metódy na rozpoznávanie rukou písaného textu sú odvodenou podskupinou OCR a HCR systémov, ktoré vykonávajú optickú transkripciu vstupných dát reprezentujúcich médium obsahujúce rukou písaný text na výstupné dátá reprezentujúce slová, vety alebo riadky v digitálnej podobe, ktoré sa systému podarilo rozpoznať na vstupe. Objektmi na

jednej inštancii výstupu tohto systému sú slová, riadky alebo vety, závisí od konkrétnej implementácie a použitej metódy. Problematika a samotné systémy tohto typu sú komplexnejšie ako HCR, nakoľko je ich štruktúra komplikovanejšia a obsahuje niekoľko krokov a logických častí, ktoré pri HCR nie sú potrebné. Medzi metódy HTR patria napríklad [7, 10]:

- convolutional neural network (CNN),
- semi incremental method,
- incremental method,
- line and word segmentation method,
- part-based method,
- slope and slant correction method,
- ensemble method,
- zoning method.

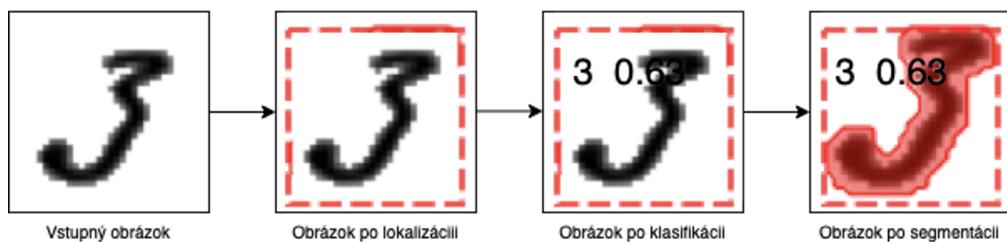
Nakoľko sa HTR venuje problematike rozpoznávania textu ako takého, v našej práci sa jej už ďalej nebudeme venovať.

3 Štruktúra HCR systému

Napriek širokému spektru rozličných typov prístupov, riešení a progresu v tejto oblasti, stále nejestvuje všeobecná schéma systému, ktorý by bol plne univerzálnym a autonómnym riešením. V druhej väčšine prípadov je pre lepšie fungovanie konkrétnej aplikácie nutné dané riešenie prispôsobiť charakteru a vlastnostiam konkrétneho problému, napríklad štruktúre dokumentu alebo jazyku. Aj keď je problematika HCR jednoduchšia ako HTR, stále sa nejedná o triviálny problém, jej riešenie je komplikované a vyžaduje značné úsilie pri jej analýze, návrhu riešenia a samotnej implementácii.

Napriek tomu je zrejmé, že štruktúra systému musí obsahovať niektoré procesy, ktoré sú invariantné. Tieto procesy tvorí [7]:

- predspracovanie,
- lokalizácia,
- klasifikácia,
- segmentácia.



Obr. 3: Schéma procesov systému na rozpoznávanie rukou písaných znakov alebo textu.

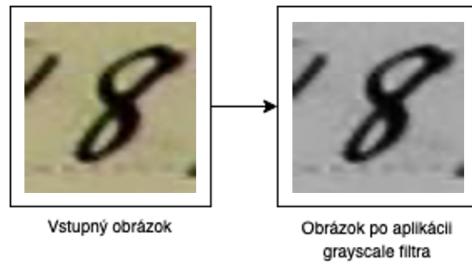
3.1 Predspracovanie

Predspracovanie alebo obrazový preprocessing je postup, ktorý predchádza lokalizácii, klasifikácii a segmentácii. Pri predspracovaní sa na vstupné dátá, teda obrázok, aplikujú rôzne techniky, ktoré majú za úlohu upraviť vstup takým spôsobom, aby následná segmentácia a klasifikácia dosahovala optimálnejšie výsledky.

Vďaka predspracovaniu je možné vstupné dátu zredukovať, normalizovať alebo upraviť ich vlastnosti bez straty alebo len s minimálnou stratou relevantnej obrazovej informácie. Techniky predspracovania obrazu sú napríklad grayscale, normalizácia, binarizácia, bit-reverse, rotácia, narovnanie sklonu riadkov, skeletonizácia, morfologické operácie alebo detekcia hrán. [7]

3.1.1 Grayscale

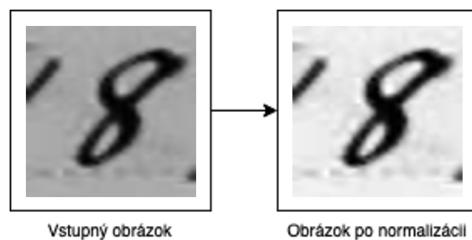
Grayscale je formát obrazových dát, kde je každý pixel reprezentovaný len jednou hodnotou, ktorá predstavuje intenzitu svetla konkrétneho pixelu, kde najnižšia hodnota spektra intenzity má čiernu farbu a najvyššia bielu farbu. Z tohto dôvodu je táto reprezentácia označovaná ako čierno-biela alebo jedno-kanálová. Obrazové dátá vo formáte grayscale sú tvorené odtieňmi šedej farby. [11]



Obr. 4: Schéma aplikácie grayscale filtra.

3.1.2 Normalizácia

Normalizácia je proces meniaci spektrum intenzity svetla pixelov v špecifikovanom rozsahu. Používa sa napríklad na zvýšenie rozsahu pri zachovaní charakteristických vlastností pôvodných obrazových dát. Rozlišujeme lineárnu a nelineárnu normalizáciu. Pri normalizácii je možné využiť filtre ako Gaussov, mediánový a iné. [7, 12]



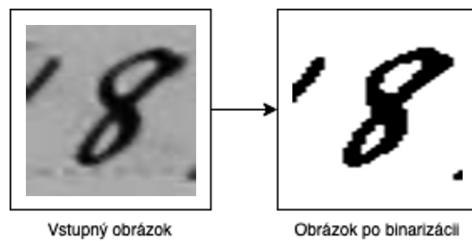
Obr. 5: Schéma aplikácie normalizácie.

3.1.3 Binarizácia

Binarizácia je proces, pri ktorom sa vykonáva transformácia obrázku z formátu grayscale s plným spektrom intenzity svetla na obrázok s množinou hodnôt intenzity svetla 0 (čierna) a 255 (biela). Táto transformácia závisí od threshold prahovej hodnoty, ktorá určuje, či sa konkrétny pixel sfarbí do bielej alebo čiernej farby. Binarizácia sa delí na statickú a dynamickú. [7]

Pri statickej binarizácii je prahová hodnota threshold pevne zvolená, najčastejšie sa používa hodnota 127, ako polovica z možného intervalu intenzity svetla 0 až 255. Vo väčšine prípadov je však statická binarizácia menej kvalitná. [7]

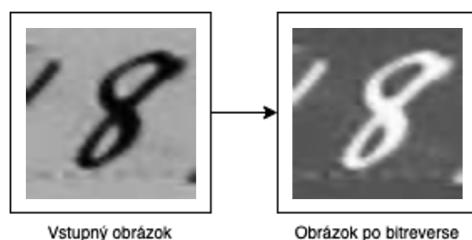
Pri dynamickej binarizácii je prahová hodnota threshold dopočítavaná automaticky. Existuje niekoľko postupov na automatický výpočet threshold hodnoty, napríklad na základe lokálnych hodnôt susedstva pixelov alebo podľa kumulatívnych momentov z histogramu pixelov (Otsu). [7]



Obr. 6: Schéma aplikácie binarizácie.

3.1.4 Bitreverse

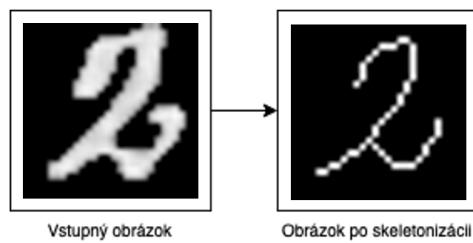
Bitreverse je proces, pri ktorom sa invertujú hodnoty intenzity svetla pixelov. Používa sa na prevrátenie farieb v rámci obrázka.



Obr. 7: Schéma aplikácie bitreverse.

3.1.5 Skeletonizácia

Skeletonizácia je proces, pri ktorom sa vykonáva extrakcia skeletu (kostry) objektu na obrázku. Pri skeletonizácii sa vo viacerých iteráciach kontrolujú susedné pixely konkrétneho pixelu v matici o rozmeroch 3x3 a pri splnení určitých kritérii sa intenzita svetla susedného pixelu nastaví na hodnotu 0. Iterácie sa pixel po pixele opakujú dovtedy, kým hrúbka skeletu (kostry) nie je práve jeden pixel. [7]

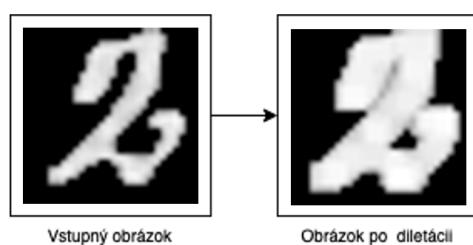


Obr. 8: Schéma aplikácie skeletonizácie.

3.1.6 Morfologické operácie

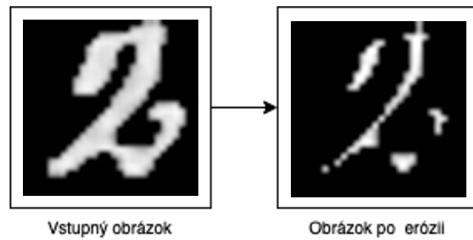
Morfologické operácie sú postupy, ktoré menia hrúbku a šírku kontúr objektu na obrázku. Proces sa podobne ako pri skeletonizácii vykonáva postupným prechádzaním všetkých pixelov maticou a následnou zmenou hodnoty pixelu podľa prítomnosti susedného pixelu s hodnotou 0 alebo 255, podľa typu operácie. Rozmer matice pri týchto operáciach nie je pevne definovaný. Rozlišujeme dva typy morfologických operácií - diletáciu a eróziu.

Pod pojmom diletácia rozumieme rozširovanie bielych pixelov, teda s hodnotou 255. Princíp spočíva v tom, že v prípade ak je matica veľkosti napríklad 3x3, výstupný pixel bude biely, teda intenzita bude mať hodnotu 255, ak sa v jeho susedstve vo vnútri matice 3x3 nachádza biely pixel. [7]



Obr. 9: Schéma aplikácie diletácie.

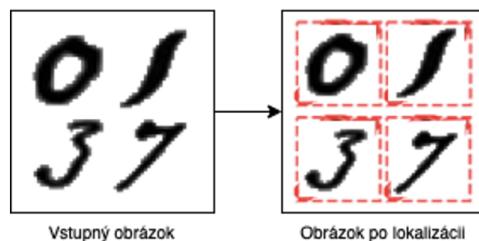
Pod pojmom erózia teda analogicky rozumieme rozširovanie čiernych pixelov, teda s hodnotou 0. Princíp spočíva v tom, že v prípade, ak je matica veľkosti napríklad 3×3 , výstupný pixel bude čierny, teda intenzita bude mať hodnotu 0, ak sa v jeho susedstve vo vnútri matice 3×3 nachádza čierny pixel. [7]



Obr. 10: Schéma aplikácie erózie.

3.2 Lokalizácia

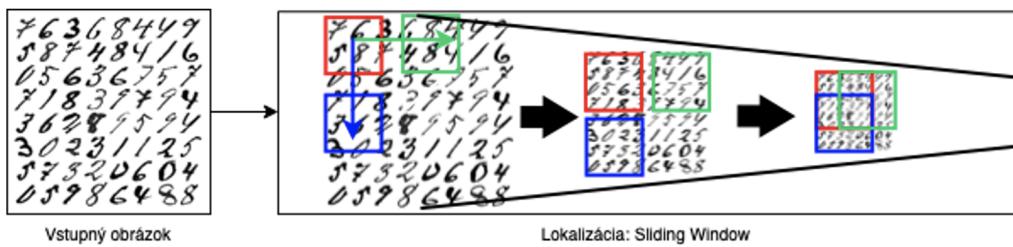
Lokalizácia je postup, ktorý predchádza klasifikácii a segmentácii. Pri lokalizácii sa na vstupných obrazových dátach, teda obrázku, vyhľadávajú jeho časti, regióny záujmu (ROI), ktoré obsahujú potencionálne inštancie objektov. Vstupným elementom pri lokalizácii je obrázok. Na výstupe sa potom nachádza množina hodnôt reprezentujúcich x-ovú a y-ovú súradnicu počiatku, výšku a šírku ohraničujúceho okna danej lokalizovanej inštancie objektu na vstupnom obrázku. Medzi metódy uskutočňujúce lokalizáciu patria napríklad sliding window, single shot detectors (SSD), region based convolutional neural networks (R-CNN), fast region based convolutional neural networks (Fast R-CNN), faster region based convolutional neural networks (Faster R-CNN) alebo you only look once (YOLO). [8, 13]



Obr. 11: Schéma procesu lokalizácie.

3.2.1 Sliding Window

Sliding windows, alebo vo voľnom preklade metóda kĺzajúceho okna, je metóda, ktorá vyberá oblasti záujmu pre detekčný systém pomocou štvorcového výseku (okna), ktorý je posúvaný v horizontálnom, respektívne vertikálnom smere. Oblasti záujmu sú následne klasifikované pomocou klasifikátora. Nevýhodou tohto prístupu je vysoká chybovosť a časová náročnosť pri samotnej detekcii. Metóda je definovaná hodnotami posunu výseku a jeho rozmermi.



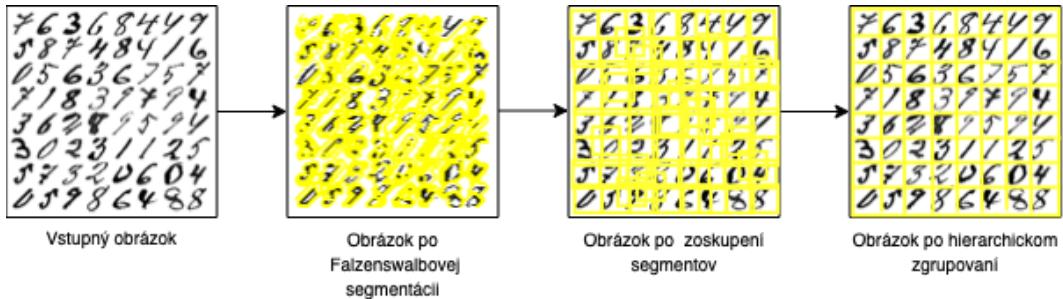
Obr. 12: Schéma procesu lokalizácie: Sliding Window.

Metóda pozostáva z niekoľkých iterácií. Jedna iterácia pozostáva z prejdenia celého obrázku v horizontálnom aj vertikálnom smere, pričom sa vykonáva na jednom konkrétnom rozmere vstupného obrázku. Tento postup zmeny veľkosti pôvodného obrázku podľa určitej mierky sa nazýva obrazová pyramída. [14]

3.2.2 R-CNN

R-CNN, konvolučná neurónová sieť založená na regiónoch, je metóda, ktorá na lokalizáciu objektov využíva prístup spadajúci do skupiny selektívneho vyhľadávania. Patrí do skupiny dvojúrovňových detektorov. Táto lokalizačná metóda má fixne určený počet vygenerovaných oblastí záujmu, a to 2000, čo znížuje časovú náročnosť pri samotnej detekcii. Oblasti záujmu sú následne klasifikované pomocou klasifikátora, pri R-CNN ide o konvolučnú neurónovú sieť. Štruktúra metódy pozostáva z nasledovných krokov [15, 16]:

- Felzenswalbova segmentácia,
- zoskupenie segmentov do regiónov záujmu,
- hierarchické zgrupenie regiónov záujmu.



Obr. 13: Schéma procesu lokalizácie: R-CNN.

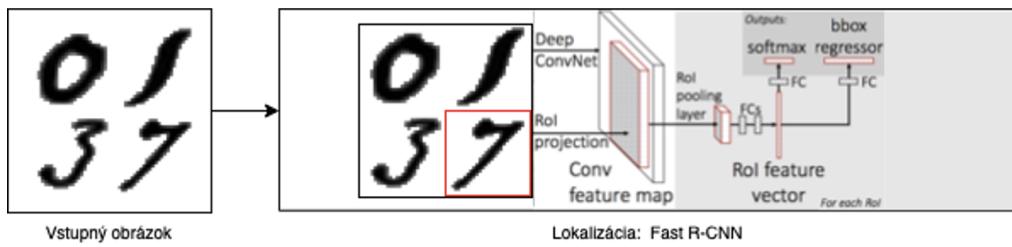
Felzenswalbova segmentácia je nízkoúrovňový segmentačný algoritmus so širokým využitím. Využíva grafový prístup, kde jednotlivé pixely reprezentujú vrcholy grafu, hrany spájajú susedné pixely a ich váha je určitým meradlom odlišnosti medzi dvomi pixelmi spojených touto hranou, napríklad rozdiel v intenzite, farbe, umiestnení alebo inom lokálnom atribúte. Výsledné segmenty dostaneme po prejdení všetkých pixelov (vrcholov), na ktoré sú aplikované rôzne grafové algoritmy a operácie. Tieto segmenty sú následne zoskupené do počiatočných regiónov záujmu. [17]

Na zoskupenie počiatočných regiónov záujmu do finálnej vzorky, na ktorej následne prebieha klasifikácia sa využíva hierarchické zgrupovanie, ktoré pracuje na princípe chameťivého algoritmu. Iteratívne sa vypočítajú podobnosti medzi susednými regiónmi, následne sú dva najpodobnejšie zoskupené do jedného. Po tomto kroku sú opäťovne vypočítané podobnosti medzi novovzniknutými susednými regiónmi. Proces sa opakuje, až kým všetky susedné regióny s podobnosťami nie sú zoskupené do jedného regiónu záujmu. [16]

3.2.3 Fast R-CNN

Fast R-CNN alebo “rýchla” konvolučná neurónová sieť založená na regiónoch je metóda, ktorá na lokalizáciu objektov využíva prístup spadajúci do skupiny selektívneho vyhľadávania. Jej návrh voľne vychádza z R-CNN. Taktiež patrí do skupiny dvojúrovňových detektorov. Lokalizáciu a klasifikáciu v tomto prípade však vykonáva jedna konvolučná neurónová sieť zložená z vrstiev rôzneho typu špecifických pre každý z týchto procesov. Výhodou tohto prístupu je nižšia časová náročnosť ako pri R-CNN. Samotnú lokalizáciu regiónov záujmu zabezpečujú [18]:

- konvolučné vrstvy,
- max-pooling vrstvy,
- ROI-pooling vrstva.



Obr. 14: Schéma procesu lokalizácie: Fast R-CNN. [18]

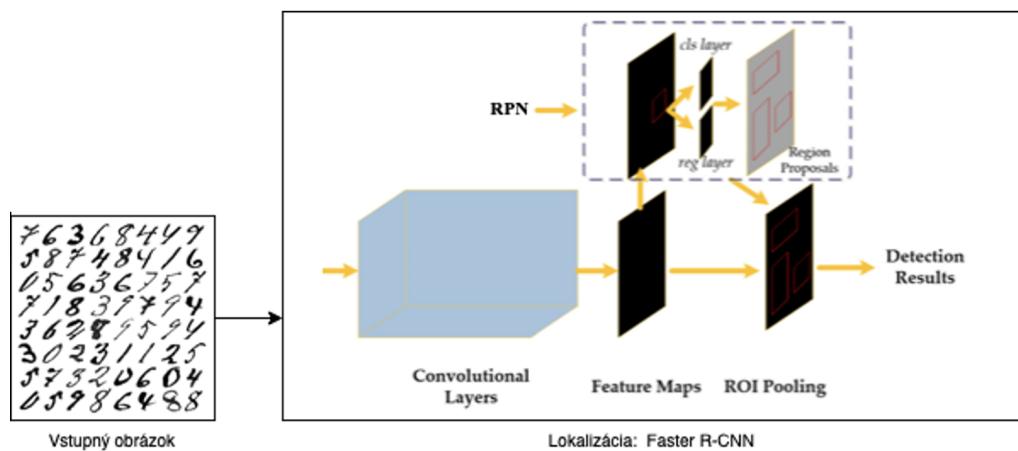
Konvolučná vrstva je typ vrstvy v neurónových sieťach, ktorý na vstup aplikuje lineárnu operáciu konvolúcii. Pri konvolúcii sa na vstupné dátu, v našom prípade obrázok, respektívne vektor, matica alebo tenzor opakovane aplikuje posuvný filter, ktorý môže mať preddefinovanú rôznu veľkosť a hodnoty. To viedie k transformácii vstupu na výstup, nazývaný mapa, respektívne vektor, matica alebo tenzor príznakov. Mapa príznakov následne obsahuje špecifické charakteristiky vstupu určené konkrétnym filtrom. [8]

Max-pooling vrstva je typ vrstvy v neurónových sieťach, ktorá na vstup aplikuje operáciu pooling. Max-pooling vrstva vo väčšine prípadov nasleduje za konvolučnou vrstvou, teda na vstup je privedený výstup konvolučnej vrstvy, mapa príznakov. Pri pooling-u dochádza k redukovaniu rozmerov vstupu pomocou posuvnej matice, respektívne jadra pevných rozmerov, kde sa na výstup premieta maximálna hodnota, ktorá sa nachádza v matici jadra. Rozmery matice jadra musia byť menšie ako rozmery vstupu. Novovzniknutá redukovaná mapa príznakov následne zachováva charakteristiky vstupnej mapy príznakov pri menších rozmeroch. [8]

ROI-pooling vrstva je typ vrstvy v Fast R-CNN neurónových sieťach, ktorý extrahuje finálne regióny záujmu z vstupnej mapy príznakov. Na túto extrakciu sa taktiež používa operácia max-poolingu, kde je vstupná mapa príznakov konkrétneho regiónu záujmu rozdelená do niekoľko pod-máp príznakov, na ktoré je následne operácia aplikovaná. Počet pod-máp je priamo závislý od hodnoty hyperparametrov, nakoľko šírka pôvodnej mapy príznakov je vydelená hyperparametrom šírky výstupného regiónu záujmu. Analogicky je teda výška pôvodnej mapy príznakov vydelená hyper-parametrom výšky výstupného regiónu záujmu. Výstupné regióny záujmu sú reprezentované ako vektory, ktoré sú následne vstupom do plne-prepojenej vrstvy, ktorá pri Fast R-CNN vykonáva klasifikáciu. [18]

3.2.4 Faster R-CNN

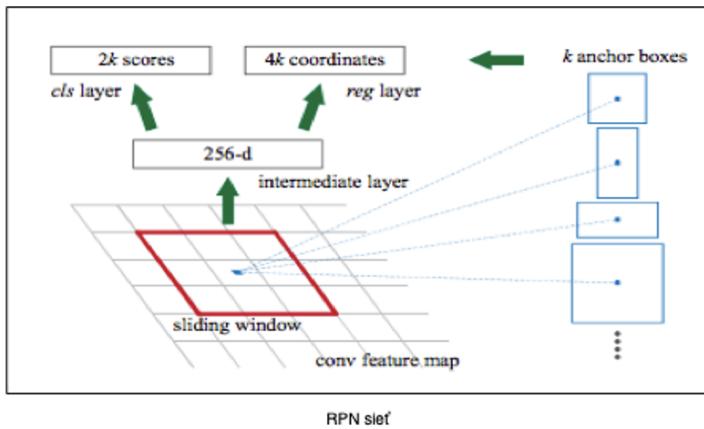
Faster R-CNN, „rýchlejšia“ konvolučná neurónová sieť založená na regiónoch je metóda, ktorá už na samotnú lokalizáciu objektov využíva strojové učenie. Patrí takisto do skupiny dvojúrovňových detektorov. Na lokalizáciu a klasifikáciu sa využívajú dve samostatné neurónové siete. Na lokalizáciu objektov a teda návrh regiónov záujmu je určená sieť pre návrh regiónov (RPN), ktorej predchádza veľké množstvo konvolučných vrstiev. Klasifikáciu následne vykonáva štandardná konvolučná neurónová sieť. Výhodou tohto prístupu je veľmi nízka časová náročnosť. [19]



Obr. 15: Schéma procesu lokalizácie: Faster R-CNN. [19]

Region Proposal Network (RPN), respektívne siet pre návrh regiónov je neurónová sieť na lokalizáciu objektov na vstupných obrazových dátach. Vstupom do tejto siete je výrez mapy príznakov z predošej konvolučnej vrstvy vygenerovaný malým kízajúcim oknom, pričom generovaný výrez môže mať rôznu mierku a pomer strán. Tento typ siete obsahuje [19]:

- box-classification vrstvu,
- box-regression vrstvu.



Obr. 16: Schéma RPN siete. [19]

Box-classification vrstva určuje pravdepodobnosť, že na navrhovanom regióne záujmu sa nachádza objekt. Box-regression vrstva regresuje výsledné súradnice navrhovaného regiónu. [20, 19]

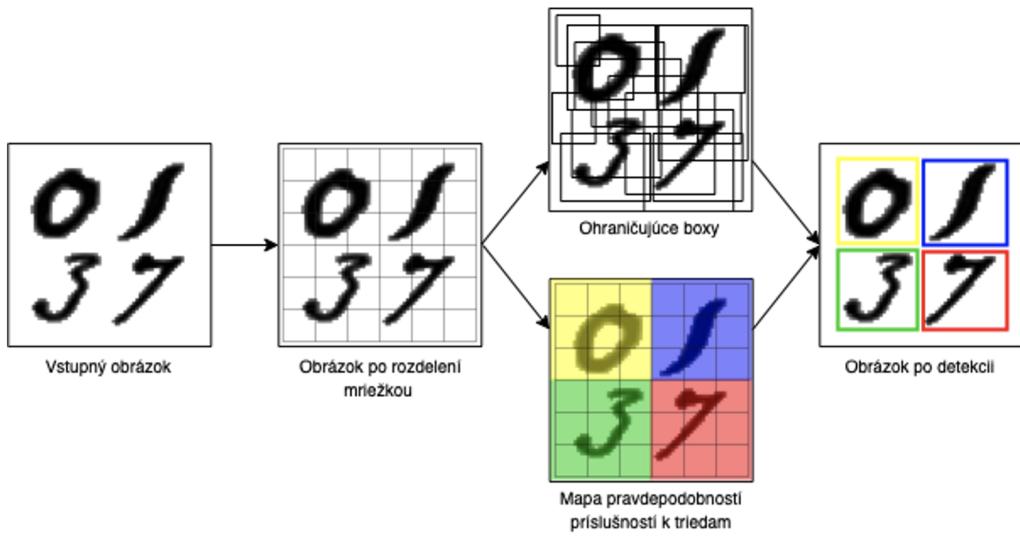
3.2.5 YOLO

You Only Look Once (YOLO) je metóda, ktorá lokalizáciu aj následnú klasifikáciu objektov využíva jednu neurónovú sieť. Patrí do skupiny jednoúrovňových detektorov, čo znamená, že lokalizácia a klasifikácia neprebieha v dvoch samostatných sériových procesoch, ale len v jednom. Sieť teda vykonáva predpovede ohraničujúcich boxov a príslušnosti k triedam súbežne. Výhodou tohto prístupu je veľmi vysoká rýchlosť a nízky výpočtový čas. Jeho nevýhodou je naopak nižšia presnosť pri samotnej detekcii oproti dvojúrovňovým detektorom. YOLO je veľmi často využívané a považované za etalón pri detekcii vo videách alebo na záberoch v reálnom čase. [21]

Počiatočný krok pri YOLO spočíva v rozdelení vstupného obrázku, respektívne snímky do N podoblastí pomocou mriežky, kde každá podoblasť ma zhodný rozmer SxS. Následne nad každou podoblasťou prebehne klasifikácia. Tento krok však prináša vysoký počet duplicitných buniek s jedným objektom a buniek bez prítomnosti objektu. [21]

YOLO následne vyberie bunky pre každý klasifikovaný objekt triedy s najvyšším pravdepodobnostným skoré. Následne dochádza k iteratívному potláčaniu buniek, ktoré majú nižšie skóre pri rovnakej klasifikovanej triede a zároveň majú najvyšší pomer presahu, čo znamená, že viac rámčekov ohraničuje jednu inštanciu objektu. Tento proces končí pri zhodení finálnych ohraničujúcich boxov. [21]

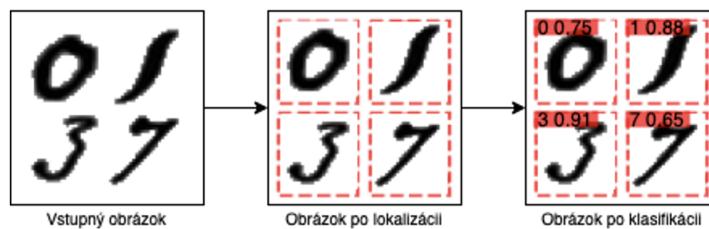
Nevýhodou tohto prístupu je neschopnosť detegovať veľmi malé alebo veľké objekty, ktoré počiatočné rozdelenie do mriežky nedokáže oddeliť od ostatných. [21]



Obr. 17: Schéma procesu lokalizácie a klasifikácie: YOLO.

3.3 Klasifikácia

Klasifikácia alebo rozpoznávanie vzorov je postup, ktorý pri sa HTR systémoch vykonáva bezprostredne po, respektívne súbežne s lokalizáciou. Počas klasifikácie dochádza k zatriedeniu lokalizovaného objektu ku konkrétnnej triede a vyčísleniu pravdepodobnosti príslušnosti k triede. Vstupným elementom pri obrazovej klasifikácii je obrázok s lokalizovaným objektom, na výstupe sa následne nachádza dátová reprezentácia príslušnosti daného objektu k triede a hodnota reprezentujúca pravdepodobnosť tejto príslušnosti.



Obr. 18: Schéma procesu klasifikácie.

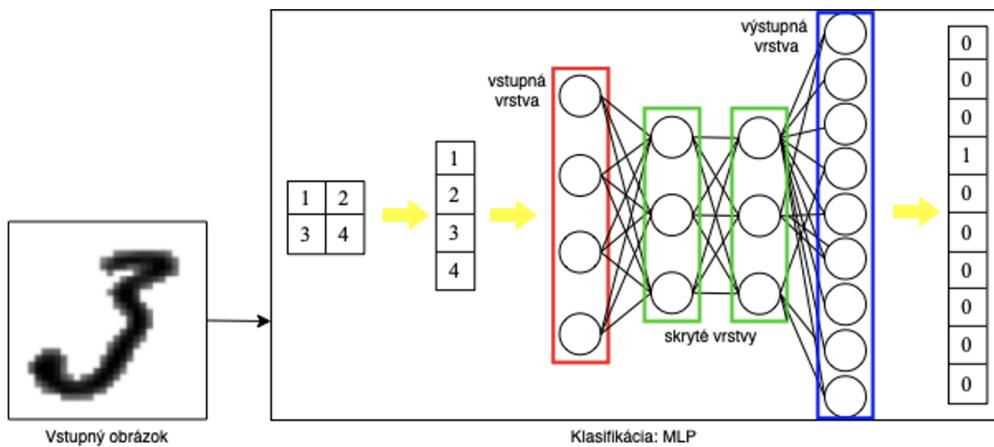
Klasifikácia obrazových dát so sebou prináša niekoľko špecifík a problémov. Medzi elementárne z nich patrí invariancia na pozícii, rotáciu a veľkosť objektu. Daný klasifikátor musí byť teda schopný rozpoznať objekt bez ohľadu na to, kde presne sa v obrazu nachádza, aký je veľký a ako je rotovaný. [22]

Existuje veľké množstvo rôznych typov klasifikátorov, napríklad rozhodovacie stromy (DT), Bayesov naivný klasifikátor (NBC), k najbližších susedov (KNN), stroje s podpornými vektormi (SVM), viacvrstvová perceptrónová neurónová sieť (MLP) alebo konvolučná neurónová sieť (CNN).

Nakoľko sa v našej práci venujeme obrazovým dátam, podrobnejšie si priblížime najrelevantnejšie z nich, konkrétnie viacvrstvové perceptrónové neurónové siete a konvolučné neurónové siete.

3.3.1 MLP - viacvrstvová perceptrónová neurónová sieť

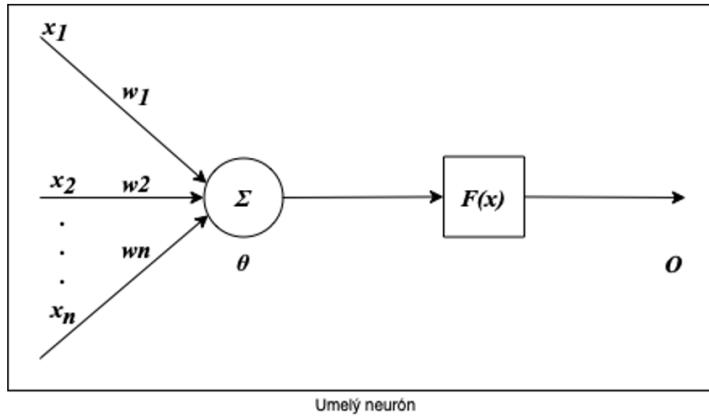
Viacvrstvová perceptrónová neurónová sieť (MLP) je model skladajúci sa z neurónov, ktoré sú usporiadané vo vrstvách, kde je jeden neurón s druhým prepojený tak, že výstup neurónu vo vrstve je privedený na vstup neurónu v ďalšej vrstve a je mu pridelená váha. Vrstvy sú postupne aktivované jedna za druhou, čo v praxi znamená, že najprv sa aktivujú všetky neuróny v prvej vrstve a ďalej sa pokračuje nasledujúcou vrstvou. Vrstva používa teda už vyčíslené výstupy predošej vrstvy. Sieť s takto prepojenými vrstvami sa nazýva dopredná sieť. Viacvrstvová perceptrónová neurónová sieť sa skladá zo vstupnej vrstvy, výstupnej vrstvy a skrytých vrstiev. [23]



Obr. 19: Schéma procesu klasifikácie: MLP.

Pri klasifikácii je na vstupe obrázok, ktorý je prevedený do maticovej reprezentácie, ktorá je následne spoštená na vektor, ktorý je privedený do vstupnej vrstvy siete. Je dôležité poznamenať, že vstupná vrstva musí obsahovať rovnaký počet neurónov ako je počet zložiek vstupného vektora. Následne prebehne prešírenie informácie sietou, pričom na výstup dostávame vektor hodnôt pravdepodobnosti príslušnosti k triedam. Výstupná vrstva musí teda taktiež obsahovať rovnaký počet neurónov, ako je počet klasifikovaných tried. Klasifikácia sa pri neurónových sietach tiež označuje ako rozpoznávanie vzorov.

Základnou stavebnou jednotkou neurónových sieti je neurón. Neurón je matematická štruktúra slúžiaca ako model biologického neurónu. Rovnako ako v prípade biologického neurónu aj matematický neurón má teda určitý počet vstupov, výstup, prahový potenciál a mechanizmus, ktorým sa aktivuje, teda svoju aktivačnú funkciu. Fakt, že jedno synaptické spojenie medzi neurónmi môže byť silnejšie než iné je modelované váhou každého vstupu. [23]



Obr. 20: Schéma umelého neurónu.

Neurón je možné charakterizovať pomocou [23]:

- $X = \{x_1, x_2, \dots, x_n\}$ – množina vstupov neurónu,
- $W = \{w_1, w_2, \dots, w_n\}$ – množina váh vstupov neurónu,
- θ – prahový potenciál,
- $F(x)$ – aktivačná funkcia neurónu,
- $O = F(x)$ – výstup neurónu.

Na fungovanie samotnej siete je potrebné prebudenie alebo aktivácia konkrétneho neurónu, ktorý na základe vstupných informácií odošle informáciu na výstup, respektívne vstup ďalšieho neurónu. Mechanizmus aktivácie neurónu popisuje aktivačná funkcia, ktorá vyjadruje akým spôsobom sa vnútorný potenciál neurónu transformuje na výstup. Existuje niekoľko variant používaných aktivačných funkcií ako znamienková funkcia, funkcia hyperbolického tangensu, rektifikovaná lineárna funkcia (ReLU) alebo sigmoidálna funkcia. Vo viacvrstvových perceptrónových neurónových sietach je najpoužívanejšia sigmoidálna funkcia. [23]

Správne fungovanie siete pri plnení konkrétnej úlohy je podmienené vhodnému nastaveniu parametrov siete. Správne nastavenie siete sa v praxi dosahuje jej trénovaním. Pri neurónových sietach sa používa technika iteratívneho trénovania. Pri iteratívnom trénovaní sa počiatočné hodnoty parametrov siete vo viacerých epochách postupne reestimujú podľa určeného pravidla. Neurónové siete pracujú s nelinárnymi aktivačnými funkciami, z tohto dôvodu sú volené iteratívne postupy, ktoré začínajú s určitou, vo väčšine prípadov náhodne vygenerovanou, počiatočnou konfiguráciou parametrov siete, ktorú je snaha v jednotlivých epochách po malých krokoch optimalizovať. [22]

Na výčislenie úspešnosti siete riešiť daný problém sa využíva chybová funkcia, ktorej hodnotu sa počas procesu trénovania snažíme minimalizovať. Na výpočet tejto hodnoty sa používa metóda klesajúceho gradientu (GD), ktorá v kombinácii s delta pravidlom výčíslí chybovosť vrstvy siete, následne sa pomocou princípu spätného šírenia chyby (BP) prešíri všetkými vrstvami siete, čo má za následok konečnú chybovosť celej siete. [22]

Nevýhodou tejto architektúry siete pri klasifikácii obrazových dát je jej nízka úspešnosť, čo viedlo k vytvoreniu konvolučných neurónových sietí.

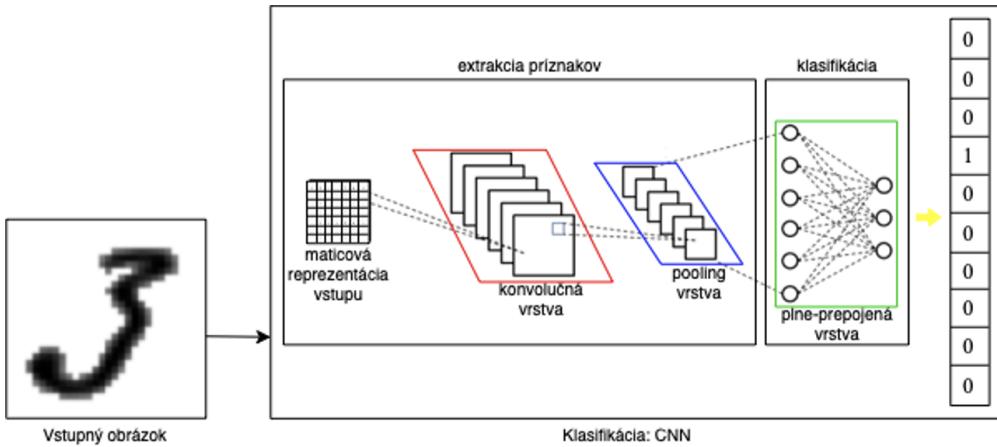
3.3.2 CNN - konvolučná neurónová sieť

Pod pojmom konvolučná neurónová sieť (CNN) rozumieme špecifický typ neurónovej siete, ktorý bol primárne vyvinutý na rozpoznávanie obrazových dát. Nakolko pri obrazových dátach je základnou požiadavkou invariancia na pozíciu, rotáciu a veľkosť objektu, bolo nutné pôvodnú MLP sieť rozšíriť a upraviť. Konvolučné siete so sebou prinášajú architektúru, ktorá obsahuje niekoľko špecifík, vďaka ktorým je možne týmto požiadavkám vyuholiť. Konvolučné neurónové siete pridávajú ku klasickej viacvrstvovej perceptrónovej sieti niekoľko ďalších vrstiev vykonávajúcich rôzne operácie nad vstupnými obrazovými dátami. [22]

Konvolučná neurónová sieť je tvorená:

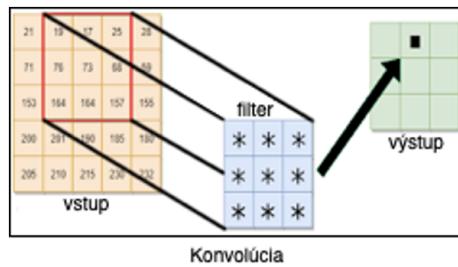
- konvolučnými vrstvami,
- pooling vrstvami,
- plne-prepojenou vrstvou.

Pri klasifikácii je na vstupe obrázok, ktorý je prevedený do maticovej reprezentácie pevných rozmerov pri obrázku typu grayscale, respektívne do tenzorovej reprezentácie pevných rozmerov pri obrázku typu RGB. Následne je vykonaná extrakcia príznakov zo vstupu, výsledná informácia je potom sietou preširená až na výstup, kde dostávame vektor hodnôt pravdepodobnosti príslušnosti k triedam. Výstupná vrstva musí rovnako ako pri MLP sieti obsahovať rovnaký počet neurónov ako je počet klasifikovaných tried. [22]



Obr. 21: Schéma procesu klasifikácie: CNN.

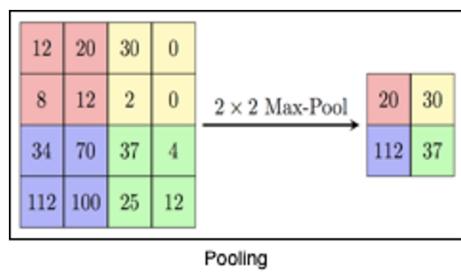
Konvolučná vrstva je typ vrstvy, ktorý na vstupnú matice, vektor alebo tenzor aplikuje lineárnu operáciu konvolúcie. Pri konvolúcii sa postupne prechádzajú body obrazu a na okolie každého z nich sa aplikuje konvolučné jadro. Konvolučné jadro je číselná matica alebo tenzor. Pri jeho aplikovaní sa po prvkoch vynásobia jednotlivé body z okolia a výsledok sa sčíta dokopy. Súčet sa použije ako nová hodnota pre daný bod. Konvolučné jadro teda možno chápať ako maticu alebo tenzor s preddefinovanými hodnotami, s ktorým klžeme po celej vstupnej matici alebo tenzore. Posun jadra je definovaný hodnotou konvolučného kroku. [22]



Obr. 22: Schéma konvolúcie. [24]

Existuje veľké množstvo typov konvolučných jadier, ktoré dokážu na vstupe zvýrazniť určité charakteristické znaky, nazývané príznaky. Konvolučné jadrá, tiež filtre alebo detektory príznakov, majú za úlohu extrahovať z vstupu také príznaky, pomocou ktorých je možné rozpoznať objekt, vykonať segmentáciu a podobne. V konvolučnej sieti je filter realizovaný umelým neurónom. Na vstup neurónu je privodený určitý výsek zo vstupu a konvolučné jadro je definované váhami neurónu. Aplikácia niekoľkých filtrov v každej konvolučnej vrstve umožňuje detegovať príznaky viacerých typov, kde je každý filter reprezentovaný unikátnym neurónom. Výstupom každého z týchto neurónov po aplikácii na celý vstup je mapa príznakov. Ak napríklad jeden neurón deteguje na vstupe horizontálne a druhý vertikálne čiary, výstupom vrstvy budú dve mapy príznakov so zodpovedajúcim obsahom. [22]

Pooling alebo združovacia vrstva spravidla nasleduje po konvolučnej vrstve. Jej úlohou je redukovať rozmiery vstupnej matice alebo tenzoru na výstupe. Pooling vrstvy zároveň znižujú citlivosť siete na presné umiestnenie príznaku v pôvodnom obrazze, čo tiež napomáha realizovať invariantnosť siete na pozícii. Pri poolingu každý neurón v združovacej vrstve priemeruje svoje vstupné hodnoty. Často sa používajú aj typ poolingu, ktorý namiesto priemeru počíta zo združovaných hodnôt maximum. Podľa tohto kritéria sa rozlišuje medzi priemerovacími združovacími vrstvami a maximálnymi združovacími vrstvami. [22]



Obr. 23: Schéma poolingu. [25]

Po extrakcii príznakov konvolučnými a združovacími vrstvami nasleduje samotná klasifikácia pomocou plne-prepojenej vrstvy. Plne-prepojená vrstva v sebe obsahuje klasifikátor v podobe viacvrstvovej perceptrónovej siete, na ktorej vstupnú vrstvu je privodená vyextrahovaná mapa príznakov spoštená na vektor. Klasifikácia následne prebieha rovnako ako pri pôvodnej architektúre MLP siete. Napriek zachovaniu architektúry však plne-prepojená vrstva v konvolučných sietach prináša niekoľko nových riešení a vylepšení. Aktivačná funkcia vo väčšine prípadov tvorí rektifikovaná lineárna funkcia (ReLU), výpočet hodnoty chybovej funkcie sa realizuje pomocou metódy stochastického klesajúceho gradientu (SGD) alebo metódy odhadu adaptívneho momentu (ADAM). Ďalším špecifickom je trénovanie.

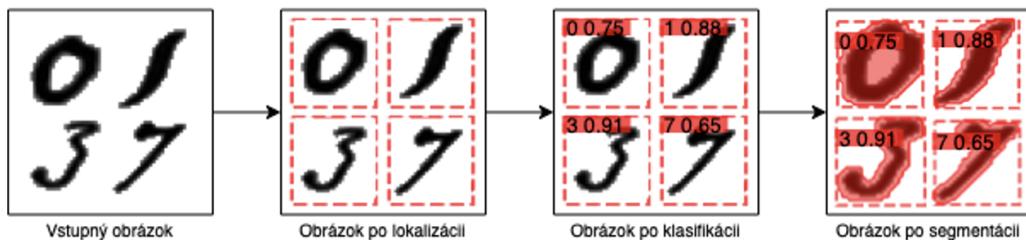
Vzhľadom na veľkosť a robustnosť dátových množín a samotných konvolučných neurónových sietí sa aplikuje transfer trénovanie. Tento proces umožňuje už komplexné predtrénované siete dotrénovať aj s pomerne malým množstvom dát tak, aby plnila inú úlohu, než na akú bola pôvodne určená. Zmeny parametrov siete pri jednotlivých epochách a iteráciach sú v tomto prípade pri porovnaní s klasickým trénovaním neporovnatelne menšie, aby nedošlo k poškodeniu parametrov pôvodnej siete a tým aj strate jej znalostí. Ďalším vylepšením pri trénovaní konvolučných neurónových sietí je metóda dropout. [22]

Dropout pri trénovaní deaktivuje niektoré neuróny v skrytých vrstvách, čím zafixuje ich váhy pri danej epoce. Tento prístup zabraňuje fixácii siete na trénovacích dátach, čím zlepšuje jej schopnosť zovšeobecňovať a tým znížuje potencionálne riziko pre-trénovania siete. [22]

Aj napriek výborným výsledkom tejto architektúry pri práci s obrazovými dátami sú badateľné jej nedostatky. Medzi ne patrí potrebná komplexnosť a robustnosť trénovacích dátových množín alebo vysoká hárverová a časová náročnosť trénovania.

3.4 Segmentácia

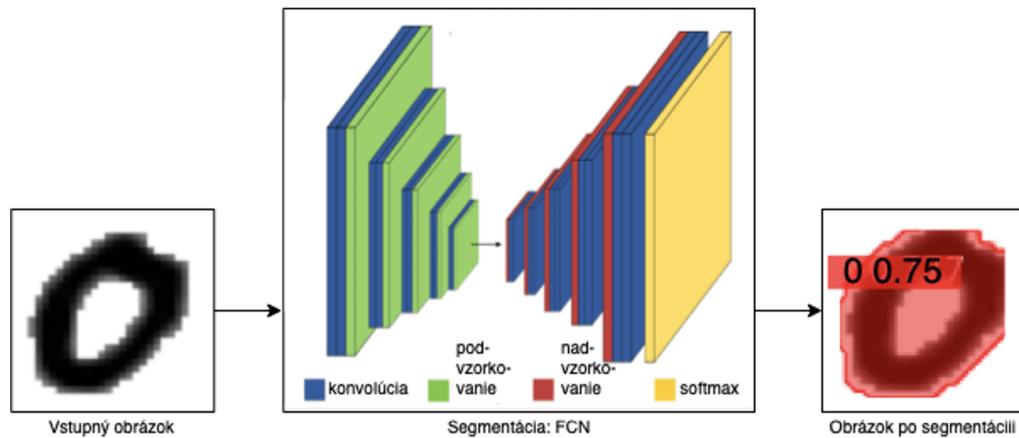
Segmentácia je proces, ktorý je v niektorých implemetáciach HCR systémov rozšírením klasifikácie. Pri segmentácii dochádza k určeniu hraníc detekovaného objektu. Vstupom segmentácie je dátová reprezentácia vstupnej obrazovej vzorky s lokalizovaným a klasifikovaným objektom, výstup potom tvorí množina ohraničujúcich pixelov segmentovaného objektu. Pri segmentácii teda dochádza k zoskupovaniu pixelov s podobnými atribútmi, ktoré reprezentujú daný objekt. Výsledkom segmentácie je teda dodatková informácia o tvare detekovaného objektu, nazývaná maska. [26]



Obr. 24: Schéma procesu segmentácie.

3.4.1 FCN - plne konvolučná sieť

Na segmentáciu sa vo väčšine prípadov využíva plne konvolučná sieť (FCN). Plne konvolučná sieť obsahuje len konvolučné vrstvy, medzi ktorými sa vykonáva podvzorkovanie a nadvzorkovanie. [26]



Obr. 25: Schéma procesu segmentácie: FCN.

Podvzorkovaním a následnou konvolúciou vo viacerých vrstvách získavame sémantické a kontextové informácie. Pomocou nadvzorkovania a konvolúcie potom opäťovne získavame priestorové informácie. Na úplné obnovenie priestorových informácií stratených pri podvzorkovaní sa používa metóda vynechania spojení, ktorá obchádza aspoň jednu vrstvu siete. Zlúčenie máp príznakov z rôznych úrovní rozlíšenia pomáha skombinovať kontextové informácie s priestorovými informáciami. Výsledkom tohto procesu je segmentovaný objekt. [27]

4 Analýza riešenia

Na základe získaných teoretických poznatkov problematiku HCR systémov podrobíme analýze. V nej sa zameriame na analýzu dostupných detekčných systémov, príslušných formátov datasetov týchto systémov a dostupných datasetov obsahujúcich rukou písané číslice.

4.1 Volba nástrojov

Na analýzu použijeme interaktívne webové vývojové prostredie pre Python, **Jupyter-Lab**, na manipuláciu s datasetmi knižnice **Pandas** a **NumPy**, respektívne **Matplotlib** na vizualizáciu. Samotný programovací jazyk **Python** bude vo verzii 3.8..

4.2 Dostupné detekčné modely

V súčasnosti existuje veľké množstvo voľne dostupných riešení detekčných systémov. V našej práci sa konkretne zameriame na dvojúrovňový detekčný model Mask R-CNN a na jednoúrovňový detekčný model YOLO v5.

4.2.1 Detekčný model Mask R-CNN

Mask R-CNN je implementácia dvojúrovňového detektora vychádzajúca z Faster R-CNN modelu, ktorý využíva siet pre návrh regiónov (RPN) na lokalizáciu a konvolučnú neurónovú sieť (CNN) na klasifikáciu objektu. Mask R-CNN navyše pridáva plne konvolučnú sieť (FCN) na segmentáciu objektu. [28]

4.2.2 COCO formát datasetu

Na úspešne trénovanie a fungovanie modelu je nutné pochopiť a oboznámiť sa so štruktúrou datasetu, ktorú siet Mask R-CNN využíva. Mask R-CNN pracuje s datasetmi vo formáte COCO. Formát COCO je formát, ktorý bol definovaný rovnomenným referenčným datasetom určeným na trénovanie a validáciu detekčných a segmentačných algoritmov počítačového videnia. Dataset obsahuje 328 000 obrázkov, 80 kategórii objektov a 1 500 000 ich inštancií. Formát datasetu sa stal štandardom pre väčšinu algoritmov počítačového videnia a datasetov určených na ich trénovanie. [29]

Dataset vo formáte COCO pozostáva z podpriečinka obsahujúceho samotné dátá, teda obrázky a zo súboru s označeniami dát a objektov vo formáte .json. Súbor s označeniami má nasledovnú štruktúru, pričom podrobnú formu JSON objektu v súbore uvádzame v prílohe B. Pre jednoduchosť uvádzame len povinné atribúty (objekty):

- **categories:** zoznam tried a ich metadát, do ktorých sú objekty klasifikované,
 - **id:** obsahuje jedinečný číselný identifikátor triedy,
 - **name:** obsahuje názov triedy,
- **images:** zoznam obrázkov datasetu a ich metadát,
 - **id:** obsahuje jedinečný číselný identifikátor obrázku,
 - **file_name:** obsahuje názov obrázku, respektívne cestu k obrázku,
 - **height:** obsahuje výšku obrázka,
 - **width:** obsahuje šírku obrázka,
- **annotations:** zoznam anotácií objektov,
 - **image_id:** obsahuje číselný identifikátor obrázku, na ktorý sa viaže anotácia,
 - **category_id:** obsahuje číselný identifikátor triedy, na ktorý sa viaže anotácia,
 - **bbox:** obsahuje zoznam x-ovej a y-ovej súradnice začiatku ohraničujúceho boxu, a jeho šírku a výšku ([x-začiatok, y-začiatok, šírka, výška]),
 - **segmentation:** obsahuje zoznam zoznamov pixelov tvoriacich polygón/y ([x-ová súradnica, y-ová súradnica, ...], ...]), ktorý/é tvoria masku segmentovaného objektu.

4.2.3 Detekčný model YOLO v5

YOLO v5 je implementácia jednoúrovňového detektora vychádzajúca z pôvodného modelu YOLO. YOLO v5 teda na detekciu a následnú klasifikáciu využíva jednu konvolučnú neurónovú siet, čo prispieva k nízkej časovej náročnosti trénovania a aj rozpoznávania. [30]

4.2.4 YOLO formát datasetu

Napriek obdobnému zameraniu modelov Mask R-CNN a YOLO v5 sa forma dát s ktorými modely pracujú líši. Model YOLO v5 pracuje s datasetmi vo formáte YOLO Darknet TXT. Tento formát vychádza zo štandardu definovaného pôvodnou verziou modelu YOLO. Formát je využívaný všetkými verziami modelu YOLO. [31]

Dataset formátu YOLO pozostáva z .yaml konfiguračného súboru a podriečinku s dátami a .txt anotáciami. Konfiguračný súbor obsahuje pole názovov tried klasifikovaných objektov, cesty k dátovým vzorkám a ich .txt anotáciám. Každej dátovej vzorke zodpovedá jeden .txt anotačný súbor s rovnakým menom. Jeho štruktúra je nasledovná:

- **class:** reprezentuje číselný index viazaný na konkrétnu triedu daného detekovaného objektu,
- **x-center:** reprezentuje x-ovú súradnicu bodu, od ktorého je vertikálne a horizontálne pripočítaná výška a šírka k hranám ohraničujúceho boxu,
- **y-center:** reprezentuje y-ovú súradnicu bodu, od ktorého je vertikálne a horizontálne pripočítaná výška a šírka k hranám ohraničujúceho boxu,
- **width:** reprezentuje šírku, ktorá je horizontálne pripočítaná alebo odpočítaná k stredovému bodu na dosiahnutie ľavej a pravej hrany ohraničujúceho boxu,
- **height:** reprezentuje výšku, ktorá je vertikálne pripočítaná alebo odpočítaná k stredovému bodu na dosiahnutie ľavej a pravej hrany ohraničujúceho boxu.

Forma samotného anotačného .txt súboru:

```
#class x-center y-center width height
0 0.48 0.63 0.69 0.71
```

4.3 Dostupné datasety rukou písaných číslí

Vzhľadom na skutočnosť, že takmer všetky v súčasnosti využívané systémy a architektúry HCR fungujú na princípe strojového učenia je potrebné tieto systémy pre dosiahnutie správneho fungovania trénovať na optimálnych dátových množinách. Druhým krokom našej praktickej časti bude teda zhromaždenie, analýza a porovnanie voľne dostupných datasetov obsahujúcich vzorky rukou písaných číslí.

Na realizáciu analýzy datasetov použijeme webové vývojové prostredie pre Python, JupyterLab, ktoré umožňuje kompliaciu kódu v bunkách. Bunky poskytujú náhľad samotnej implementácie ako aj vizualizáciu výstupu danej bunky. Na uvedenie textových komentárov použijeme značkovací jazyk Markdown. S datasetmi budeme manipulovať pomocou knižnice Pandas, ktorá umožňuje import datasetu z rôznych formátov do vlastného dátového typu DataFrame, čím uľahčujú prácu so samotnými dátami. Ďalšou knižnicou, ktorú pri manipulácii s datasetmi využijeme bude NumPy, ktorá poskytuje rozhranie pre prácu s vektormi, maticami a tenzormi. Na vizualizáciu vzoriek a grafov použijeme grafickú knižnicu Matplotlib.

Počas našej práce sa nám podarilo zhromaždiť niekoľko datasetov v rôznych formátoch a z rôznych zdrojov, pričom ich názvy a základné charakteristiky sú uvedené v nasledujúcej tabuľke. Podrobnejšia analýza datasetov je obsiahnutá v elektronickej prílohe v .ipynb súboroch, ktoréj štruktúru uvádzame v prílohe A.

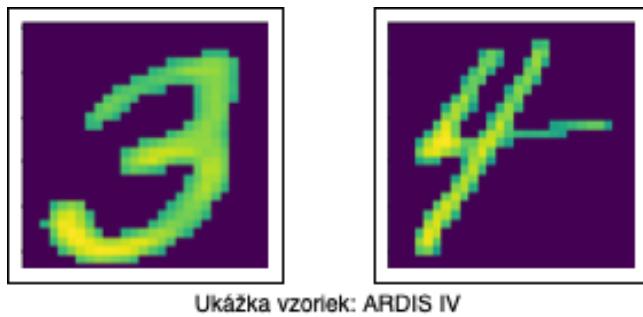
	Trénovacie vzorky	Testovacie vzorky	Rozmery vzorky	Farebné spektrum	Formát
ARDIS IV	6600	1000	28x28	Grayscale	.csv
Chars74K		550	1200x900	RGB	.png, .m
DIDA		252860	rôzne	RGB	.jpg
MNIST	60000	10000	28x28	Grayscale	.csv
EMNIST	240000	40000	28x28	Grayscale	.csv
SEMEION		1593	16x16	Grayscale	.csv
USPS	7291	2007	16x16	Grayscale	.h5

Tabuľka 2: Zoznam datasetov a ich charakteristik

4.3.1 ARDIS

ARDIS je referenčný dataset obsahujúci historické ručne písané čísllice. Vznikol ako reakcia na nízku úspešnosť klasifikácie systémov trénovaných na súčasných a testovaných na historických dátach. Dataset bol vytvorený v roku 2019 zo švédskej historických cirkevných záznamov vytvorených v rokoch 1895 až 1970. Dataset je tvorený 4 sadami dát: ARDIS I, ARDIS II, ARDIS III, ARDIS IV. [32, 33]

Sada ARDIS I bola vytvorená manuálnym obdĺžnikovým výberom veľkosti 175x95 pixelov zachytávajúcim roky z približne 10 000 cirkevných dokumentov v pravom, respektívne v ľavom hornom rohu. Sada obsahuje 10 000 vzoriek. ARDIS II bola vytvorená manuálnym obdĺžnikovým výberom variabilnej veľkosti číslic z približne 15 000 cirkevných dokumentov, pri zachovaní reálnej pôvodnej veľkosti vzorky v závislosti od veľkosti samotného obdĺžnikového výberu. Sada obsahuje 7 600 vzoriek. Sada ARDIS III bola odvodenaná od ARDIS II, kde sú pôvodné vzorky manuálne odšumené od pozostatkov iných číslíc, riadkov a iných nečistôt. Sada obsahuje rovnako 7 600 vzoriek. Sada ARDIS IV bola odvodenaná práve od sady ARDIS III, kde sú vzorky transformované a normalizované do formátu, ktorý korešponduje s referenčným datasetom MNIST, teda na veľkosť 28x28 pixelov a farebný formát grayscale. Vzorky sú rozdelené do trénovacej a testovacej podmnožiny v pomere 6 600 k 1 000. Sada je vo formáte .csv. Kvôli týmto charakteristikám sme sa v našej analýze a práci zamerali práve na túto sadu. [32, 33]

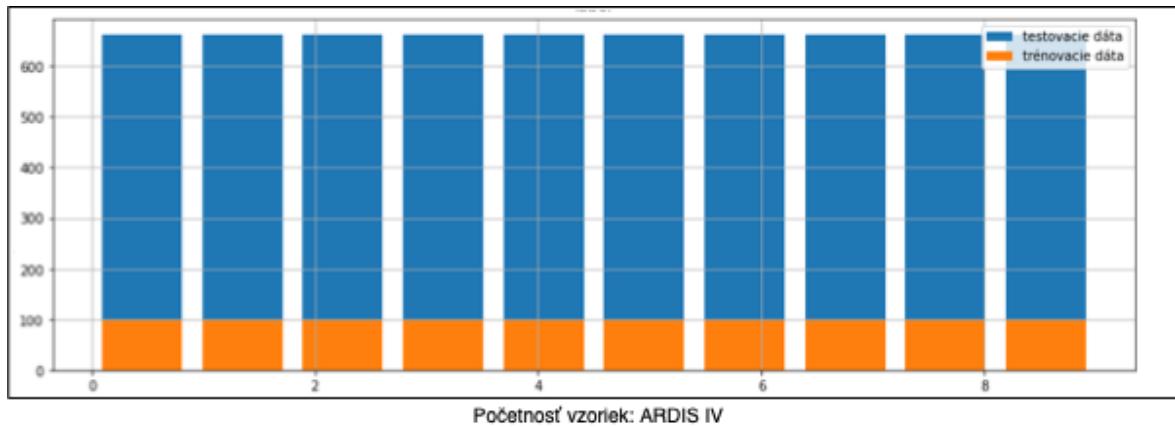


Ukážka vzoriek: ARDIS IV

Obr. 26: Ukážka vzoriek datasetu: ARDIS IV.

ARDIS IV pozostáva z podmnožín obsahujúcich dát a z podmnožín obsahujúcich označenie príslušných dát. Podmnožinu dát tvorí 784 stĺpcov, pričom každý obsahuje informáciu v diskrétnej kvantitatívnej premennej o jednotlivých pixeloch (0-1) danej číslice pri rozmere mriežky 28x28 pixelov. Podmnožina označení pozostáva z 10 stĺpcov, pričom na indexe číslice, ktorú vzorka reprezentuje sa nachádza hodnota 1, ostatné hodnoty sú nastavené na 0. Štruktúra sady je pri jeho trénovacej aj testovacej časti zhodná. Trénovacia časť pozostáva z 6 600 riadkov, testovacia z 1 000 riadkov.

Pri datasetoch určených na použitie pri strojovom učení je dôležitá ich vhodná robustnosť a škálovanie, ktoré dokážu pri vhodnej optimalizácii daného systému maximalizovať úspešnosť klasifikácie a minimalizovať trénovací čas. Nakolko ARDIS IV vychádza z referenčného datasetu MNIST, jeho trénovacia aj testovacia časť obsahuje absolútne rovnomerný počet vzoriek.



Obr. 27: Graf početnosti vzoriek: ARDIS IV.

4.3.2 Chars74K

Chars74K je dataset obsahujúci číslice a písmená. Vznikol ako reakcia na potrebu klasifikovania číslíc, písmen a symbolov pomocou interiérových a exteriérových kamier. Dataset je celkovo tvorený 6 sadami s číslicami a písmenami z anglického a kanadského jazyka napísaných rukou alebo zaznamenanými pomocou rôznych interiérových a exteriérových kamier. Menovite ide o: English Img, English Hnd, English Fnt, Kannada Img, Kannada Hnd a Kannada Fnt. [34]

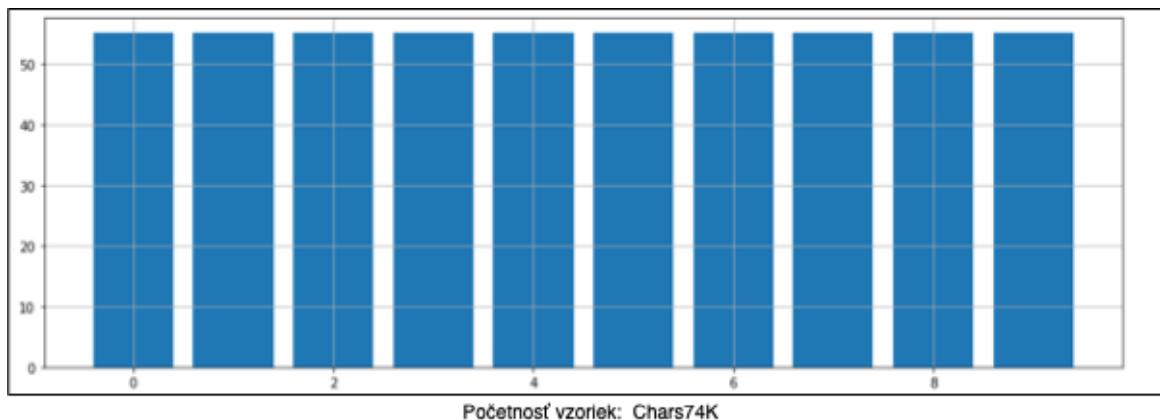


Ukážka vzoriek: Chars74K

Obr. 28: Ukážka vzoriek datasetu: Chars74K.

V tejto konkrétnej analýze sme sa zamerali na sadu Chars74K English Hnd. Sada vznikla v roku 2009 zaznamenaním 3 430 číslíc a písmen od 55 respondentov pomocou grafického tabletu. Z tohto počtu celkovo 550 vzoriek tvoria arabské číslice. Sada je tvorená 10 podpriečinkami pre každú triedu, ktoré obsahujú .png súbory reprezentujúce vzorky číslíc s veľkosťou 1200x900 vo farebnom formáte RGB. Napriek formátu RGB pixely nadobúdajú len hodnoty 0 a 1. [34]

Chars74K English Hnd je tvorený jednou všeobecnou podmnožinou s absolútne rovnomerným rozdelením počtu vzoriek.



Obr. 29: Graf početnosti vzoriek: Chars74K.

4.3.3 DIDA

DIDA je najrobustnejší dataset obsahujúci ručne písané číslice. Je využívaný pri trénovaní obrazspracujúcich systémov, pričom jadrom tohto využitia je strojové učenie. Dataset je tvorený 252 860 vzorkami rukou písaných číslic. [35, 36]

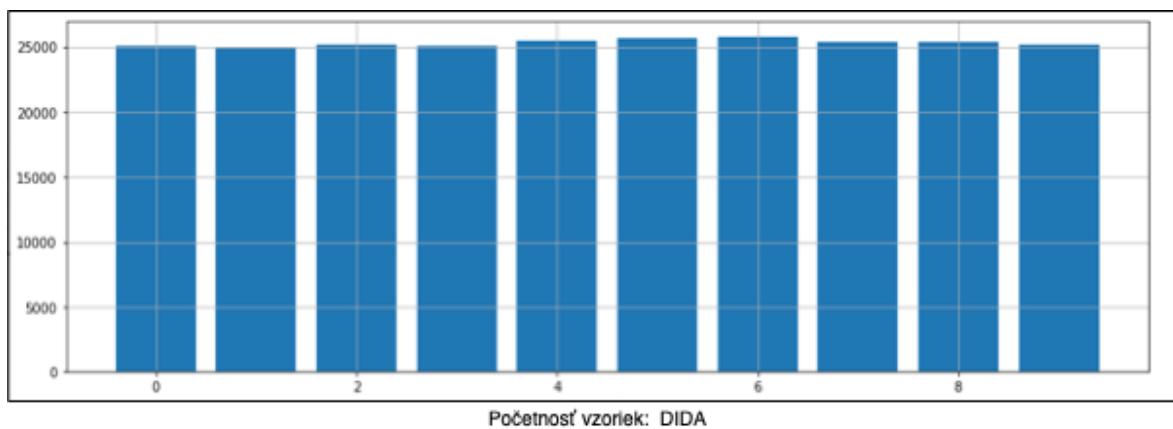


Ukážka vzoriek: DIDA

Obr. 30: Ukážka vzoriek datasetu: DIDA.

Dataset DIDA bol vytvorený v roku 2020 extrahovaním vzoriek zo 75 000 švédskych historických dokumentov vytvorených v rokoch 1800 až 1940. Dáta sú rozdelené do 10 podpriečinkov pre každú triedu. Vzorky číslic datasetu majú rôzne veľkosti a naturálny farebný formát RGB, uložené sú v súboroch typu .jpg. Vzorky sú v úplne "surovej" forme, neprešli normalizáciou, transformáciou ani úpravou. Rámce s číslicami obsahujú časti iných číslic a šum, pričom samotné čísla sú rôznej hrúbky a podklad rôznej farby a štruktúry. [35, 36]

Dataset DIDA vo svojej jedinej všeobecnej podmnožine obsahuje relatívne rovnomerný počet vzoriek.

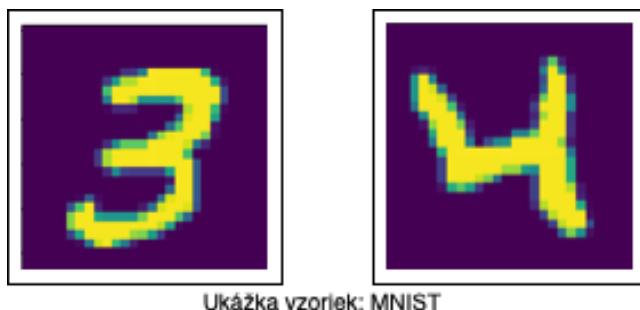


Obr. 31: Graf početnosti vzoriek: DIDA.

4.3.4 MNIST

MNIST je referenčný dataset obsahujúci ručne písané čísllice. Patrí medzi najpopulárnejšie datasety využívané pri strojovom učení. Dataset bol vytvorený v roku 1998. MNIST pozostáva z trénovacej a testovacej podmnožiny dát. Trénovacia podmnožina obsahuje 60 000 vzoriek rukou písaných číslic. Testovacia podmnožina naproti tomu obsahuje 10 000 vzoriek. Vzorky číslic datasetu majú veľkosť 28x28 pixelov a farebný formát grayscale, teda každý pixel vzorky obsahuje hodnotu reprezentujúcu intenzitu svetla. [37]

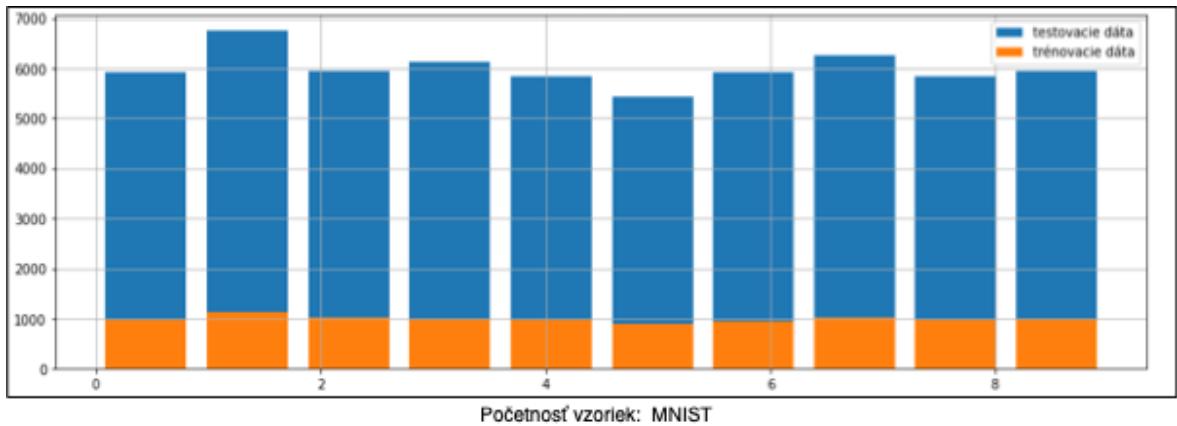
Dataset MNIST bol vytvorený v roku 1998 z NIST Special Database 19, konkrétnie datasetov NIST SD-1 a NIST SD-3 Národného inštitútu pre štandardy a technológie (NIST) Spojených štátov amerických. Sady NIST SD-1 a NIST SD-3 boli vytvorené v roku 1995. NIST SD-1 bola pôvodne koncipovaná ako testovacia a NIST SD-3 ako trénovacia sada. Toto rozdelenie sa ale ukázalo ako nevhodné. NIST SD-3 disponoval čistejšími a kvalitnejšími vzorkami číslic, nakoľko sada bola vytvorená zozbieraním dát medzi zamestnancami americkej federálnej agentúry Census Bureau a sada NIST SD-1 medzi študentmi amerických stredných škôl. Súčasný MNIST obsahuje po 30 000 vzoriek z NIST SD-3 a NIST SD-1 v trénovacej podmnožine, respektívne 5 000 vzoriek z NIST SD-3 a NIST SD-1 v testovacej podmnožine. Množiny vzoriek daných sád sú nesúrodé. Pôvodné vzorky číslic z NIST podmnožín boli normalizované, aby sa zmestili do rámcu o veľkosti 20x20 pixelov pri zachovaní ich pomeru strán. Vzorky číslic datasetu MNIST boli vycentrované do rámcu o veľkosti 28x28 vypočítaním tažiska pixelov a transformáciou obrázku. Podmnožiny sú vo formáte .csv. [37]



Obr. 32: Ukážka vzoriek datasetu: MNIST.

Dataset pozostáva zo 785 stĺpcov. Prvý s označením label obsahuje informáciu v nominálnej kategorickej premennej, o ktorú číslicu sa jedná (0-9), zvyšných 784 obsahuje informáciu v diskrétnnej kvantitatívnej premennej o jednotlivých pixeloch (0-255) danej číslice pri rozmere mriežky 28x28. Štruktúra datasetu je pri jeho trénovacej aj testovacej podmnožine zhodná. Trénovacia časť pozostáva zo 60 000 riadkov, testovacia z 10 000 riadkov.

Trénovacia aj testovacia podmnožina obsahuje relatívne rovnomerný počet vzoriek.

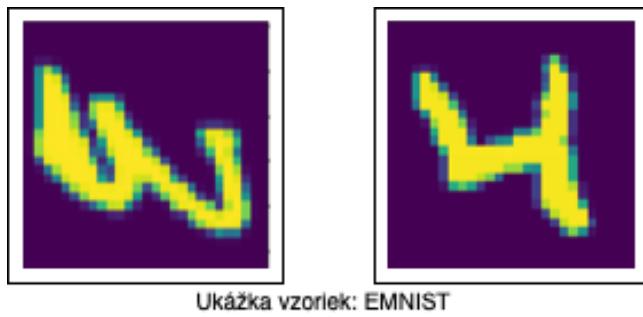


Obr. 33: Graf početnosti vzoriek: MNIST.

4.3.5 EMNIST

EMNIST je referenčný dataset obsahujúci ručne písané číslice a písmená. Vznikol ako rozšírenie datasetu MNIST. Dataset je celkovo tvorený 6 sadami obsahujúcimi rukou písané číslice, písmena a ich kombinácie. Ide o: EMNIST Balanced Dataset, EMNIST By_Merge Dataset, EMNIST By_Class Dataset, EMNIST Letters Dataset, EMNIST MNIST Dataset, EMNIST Digits Dataset. [38]

Dataset bol vytvorený v roku 2017 ako rozšírenie referenčného datasetu MNIST z pôvodného datasetu NIST Special Database 19, ktorý bol vytvorený medzi zamestnancami americkej federálnej agentúry Census Bureau a študentmi amerických stredných škôl. Bol vytvorený zmiešaním dát z NIST SD-1 a NIST SD-3 a následne ich rozšírením pomocou deformácie, rotácie a prevrátenie v trénovacej a testovacej sade. Na tieto vzorky veľkosti 128x128 sa následne aplikovali metódy na zjemnenie hrán a boli z nich vybraté a vycentrované samotné symboly pomocou obdĺžnikového výberu do finálnych vzoriek vo veľkosti 28x28. [38]

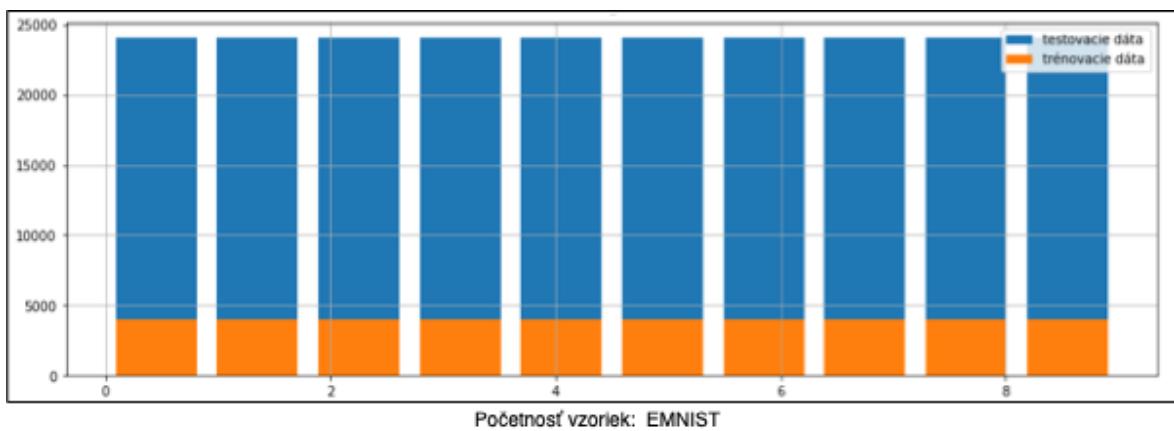


Obr. 34: Ukážka vzoriek datasetu: EMNIST.

V tejto konkrétnej analýze sme sa zamerali na sadu EMNIST Digits Dataset. Sada je tvorená trénovacou a testovacou podmnožinou dát. Trénovacia podmnožina obsahuje 240 000 vzoriek rukou písaných čísl, testovacia podmnožina naproti tomu obsahuje 40 000 vzoriek rukou písaných čísl. Vzorky čísl datasetu majú veľkosť 28x28 pixelov a farebný formát grayscale, teda každý pixel vzorky obsahuje hodnotu reprezentujúcu intenzitu svetla. Podmnožiny sú vo formáte .csv.

Dataset pozostáva zo 785 stĺpcov, prvý obsahuje informáciu v nominálnej kategorickej premennej, o ktorú číslicu sa jedná (0-9), zvyšných 784 obsahuje informáciu v diskrétnej kvantitatívnej premennej o jednotlivých pixeloch (0-255) danej číslice pri rozmeri mriežky 28x28. Štruktúra datasetu je pri jeho trénovacej aj testovacej časti zhodná. Trénovacia časť pozostáva z 240 000 riadkov, testovacia zo 40 000 riadkov.

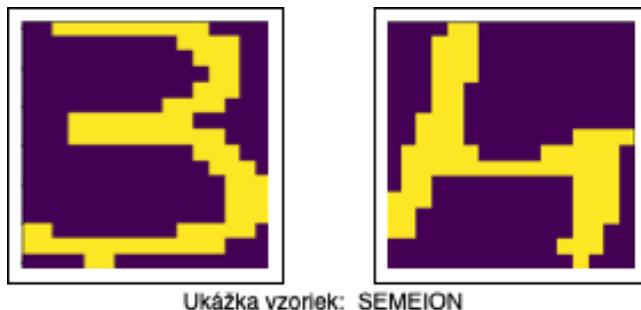
Nakoľko EMNIST vychádza z referenčného datasetu MNIST, trénovacia aj testovacia podmnožina obsahuje absolútne rovnomerný počet vzoriek.



Obr. 35: Graf početnosti vzoriek: EMNIST.

4.3.6 SEMEION

SEMEION je dataset obsahujúci ručne písané čísllice. Dataset je tvorený 1 593 vzorkami rukou písaných čísliec. Vzorky čísliec datasetu majú veľkosť 16x16 pixelov a farebný formát grayscale, teda každý pixel vzorky obsahuje hodnotu reprezentujúcu minimálnu alebo maximálnu intenzitu svetla. [38, 39]



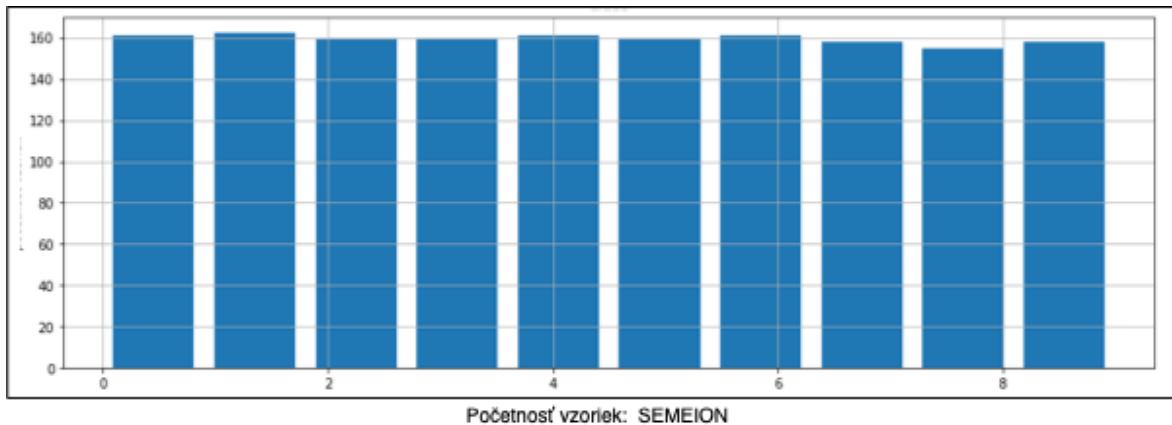
Obr. 36: Ukážka vzoriek datasetu: SEMEION.

Dataset SEMEION bol vytvorený v roku 1994 z formulárov približne 80 respondentov, ktorí mali napísať čísllice od 0 po 9. V prvom prípade s dôrazom na prevedenie a v druhom s dôrazom na rýchlosť. Tieto jednotlivé vzorky boli natiahnuté do štvorcového rámcu o veľkosti 16x16 vo farebnom formáte grayscale, kde bola následne každá hodnota reprezentujúca pixel upravená na hodnotu 0 alebo 1 pomocou prahovania. [38, 39]

Dataset pozostáva zo 257 stĺpcov, prvých 256 obsahuje informáciu v diskrétnej kvantitatívnej premennej o jednotlivých pixeloch (0 alebo 1) danej číslice pri rozmere mriežky 16x16. Posledný stĺpec s označením Class obsahuje informáciu v nominálnej kategorickej premennej, o ktorú čísllicu sa jedná (0-9). Počet riadkov je 1 593.

Pri vizualizácii vzoriek dát sme dospeli k záveru, že označenie ich príslušnosti k daným čísliciam nie je logicky správne. Preto sme dataset modifikovali, aby ich označenie priamo korešpondovalo s tým, akú čísllicu vzorka reprezentuje.

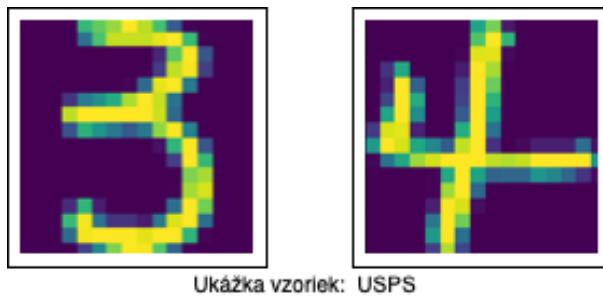
Dataset Semeion vo svojej jedinej sade obsahuje relatívne rovnomerný počet vzoriek.



Obr. 37: Graf početnosti vzoriek: SEMEION.

4.3.7 USPS

USPS je dataset obsahujúci ručne písané čísllice. Dataset je tvorený trénovacou a testovaciou sadou dát. Trénovacia sada obsahuje 7 291 vzoriek rukou písaných čísllic. Testovacia sada naproti tomu obsahuje 2 007 vzoriek rukou písaných čísllic. Vzorky čísllic datasetu majú veľkosť 16x16 pixelov a farebný formát grayscale, teda každý pixel vzorky obsahuje hodnotu reprezentujúcu intenzitu svetla. Dataset je uchovaný vo formáte .h5. [40]

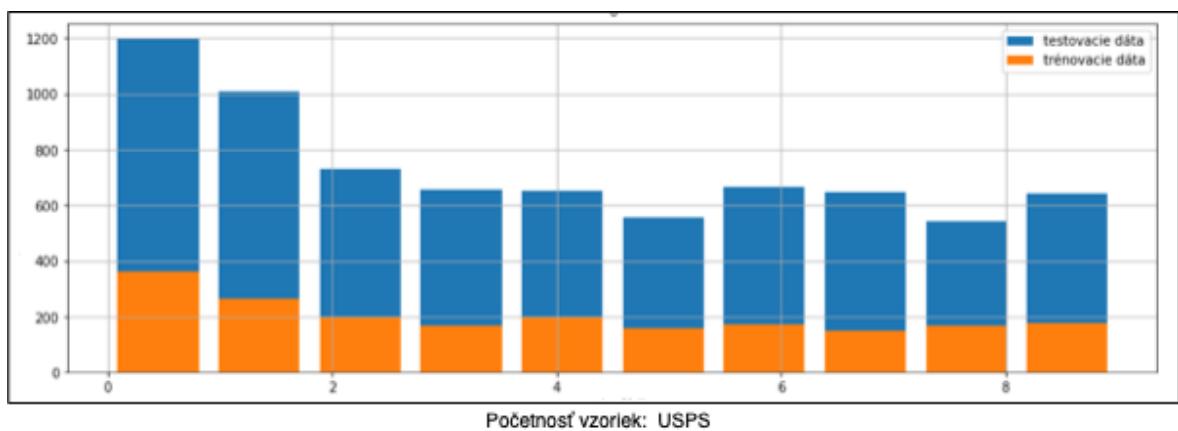


Obr. 38: Ukážka vzoriek datasetu: USPS.

Dataset USPS bol vytvorený zo záznamov americkej Poštovej služby, konkrétnie z databázy 5 000 názvov miest, 5 000 názvov štátov a 10 000 poštových smerových čísel. [40]

Dataset pozostáva z podmnožín obsahujúcich dát a označenie príslušných dát. Podmnožina dát pozostáva z 256 stĺpcov, pričom každý obsahuje informáciu v diskrétnej kvantitatívnej premennej o jednotlivých pixeloch (0-1) danej číslice pri rozmere mriežky 16x16. Podmnožina označení pozostáva z 1 stĺpca, ktorý obsahuje informáciu v nominálnej kategorickej premennej, o ktorú číslicu sa jedná (0-9). Štruktúra datasetu je pri jeho trénovacej aj testovacej časti zhodná. Trénovacia časť pozostáva z 7 291 riadkov, testovacia z 2 007 riadkov.

Početnosť vzoriek v triedach v datasete USPS je rôzna, pričom nie je zachovaný ani pomer číslic v rovnakých triedach v trénovacej a testovacej sade. Dataset teda nie je vhodne vybalansovaný a je otázne, aký vplyv by táto vlastnosť mala na výsledný detekčný systém.



Obr. 39: Graf početnosti vzoriek: USPS.

5 Implementácia riešenia

Nadobudnuté znalosti o HCR systémoch a jeho súčastiach sme aplikovali v implementačnej časti našej práce, ktorej hlavným cieľom bolo porovnanie rôznych metód, modelov a techník rozpoznávania historických rukou písaných číslic. Ako už bolo povedané problematika HCR systémov je veľmi široká, preto sme postup našej práce rozdelili do niekolkých podúloh.

HCR systémy sa skladajú z 3 hlavných častí, a to predspracovania, lokalizácie a klasifikácie. Niektoré systémy tiež pridávajú možnosť segmentácie objektu. Na základe tejto štruktúry bolo prvou úlohou vytvorenie jednoduchej aplikácie na predspracovanie obrázkov. Druhým krokom následne bola implementácia a porovnanie samotných detekčných systémov.

5.1 Volba nástrojov

Na implementáciu riešenia v našej práci sme využili interpretovaný skriptovací jazyk **Python**, ktorý ponúka širokú škálu funkcionálít a knižníc pri dátovej analýze a strojovom učení. V súčasnosti je dostupné veľké množstvo verzii a distribúcii tohto programovacieho jazyka, v našej práci sme pracovali s distribúciou **Anaconda**, ktorá okrem svojho cielenia na dátovú analýzu a strojové učenie prináša prehľadnú správu balíkov a knižníc, programovací jazyk Python bude vo verzii 3.8..

Pri vývoji aplikácie na predspracovanie sme použili knižnicu **OpenCV** na implementáciu metód úpravy obrazových dát a knižnicu **Tkinter** na realizáciu grafického používateľského rozhrania.

Pri porovnaní detekčných systémov sme použili existujúce aplikačné rámce využívajúce knižnice **TensorFlow** alebo **PyTorch**, ktoré implementujú rôzne metódy strojového učenia, prevažne neurónové siete. Na vyhodnotenie sme využili interaktívne webové vývojové prostredie pre Python, **JupyterLab**.

Je dôležité poznamenať, že zmienené knižnice a aplikačné rámce na pozadí pracujú s ďalšími knižnicami, pri ktorých však v rámci účelov našej práce nie je nutné zachádzať do podrobností.

5.2 Aplikácia na predspracovanie obrazu

Prvým krokom našej práce bola implementácia jednoduchej aplikácie na predspracovanie obrázkov. Aplikácia umožňuje vykonávanie základných metód a procesov využívaných pri predspracovaní obrazu v užívateľsky prívetivom grafickom rozhraní. Aplikácia je obsiahnutá v elektronickej prílohe, ktorej štruktúra je opísaná v prílohe A.

Na implementáciu metód predspracovania obrazu sme použili knižnicu OpenCV. Grafické rozhranie celej oknovej aplikácie bolo realizované pomocou knižnice Tkinter.

Pri vývoji celej našej aplikácie sme čerpali z existujúceho riešenia, ktoré bolo výstupom tímového projektu s názvom Porovnanie metód a nástrojov na rozpoznávanie rukopisov: Shougao realizovaného na Fakulte elektrotechniky a informatiky Slovenskej technickej univerzity v Bratislave. [7]

Z pôvodnej aplikácie sme odstránili nadbytočnú funkcionality, ktorá nekorešpondovala s naším zámerom využitia, upravili sme dizajn a rozloženie ovládacích prvkov, a architektúru samotnej implementácie.

Napriek prítomnosti procesu predspracovania v HCR systémoch je nutné podotknúť, že aplikácia vo finálnom procese rozpoznávania nebola využitá.

5.2.1 Metódy predspracovania

Naša aplikácia ponúka možnosť realizácie niektorých základných metód predspracovania obrazu využívaných pri HCR systémoch, ktoré boli realizované pomocou knižnice OpenCV. Konkrétnie ide o:

- konverziu do grayscale,
- odšumenie,
- binarizáciu (prahovanie/adaptívne prahovanie),
- detekciu hrán Canny,
- morfologické operácie (diletácia/erózia).

Medzi základnú metódu pri predspracovaní patrí **prevod obrázku do grayscale**, teda prevod z trojfarebného spektra do spektra odtieňov šedej. Grayscale formát je v aplikáciach počítačového videnia často využívaný pre jeho nízke pamäťové nároky a zároveň schopnosť zachovať relevantné obrazové informácie a príznaky. V našej aplikácii sa táto metóda na vložené obrázky aplikuje automaticky. Jej vykonanie sme zabezpečili OpenCV funkciou **cvtColor()** s aktívnym prepínačom **COLOR_BGR2GRAY** v nasledovnej forme [7]:

```
1 output_image = cv2.cvtColor(input_image, cv2.COLOR_BGR2GRAY)
```

Ďalšou implementovanou metódou bolo **odšumenie**, ktoré redukuje rôzne nedokonalosti vstupného obrázku. Tento postup sa realizuje aplikovaním šumu na vyhľadanie nedokonalostí, pričom sa najčastejšie využíva gaussovský šum. Knižnica OpenCV túto metódu aplikuje funkciu **GaussianBlur()**, ktorá ponúka možnosť definovania výšky kernelu, šírky kernelu a sigmy. Finálna podoba funkcie je nasledovná [7]:

```
1 output_image = cv2.GaussianBlur(input_image, (kernel_width, kernel_height), sigma)
```

Nevyhnutnú metódu pri predspracovaní predstavuje **binarizácia**. Táto metóda sa vykonáva pomocou prahovania, kedy zvolený prah určuje hranicu transformácie každého pixelu na hodnotu 255 alebo 0, teda bielu alebo čiernu. Knižnica OpenCV ponúka dve funkcie využívajúce túto funkcionality, **adaptiveThreshold()** a **threshold()**. Pri adaptívnom prahovaní funkciou adaptiveThreshold() sa prah dopočítava adaptívne na základe vlastností okolitých pixelov. Funkcia ponúka nastavenie maximálnej hodnoty prahovania, typ výpočtu adaptívneho prahu (**ADAPTIVE_THRESH_MEAN_C**/
ADAPTIVE_THRESH_GAUSSIAN_C), typ prahovania (**THRESH_BINARY**/
THRESH_BINARY_INV), veľkosť bloku susedných pixelov pre výpočet prahovania a hodnotu C pri výpočte adaptívneho prahu. Bežné prahovanie funkciou threshold() ponúka možnosť nastavenia samotného prahu, maximálnej hodnoty prahovania a typ prahovania (**THRESH_BINARY**/
THRESH_BINARY_INV). Funkcie sú nasledovného tvaru [7]:

```
1 output_image = cv2.threshold(input_image, threshold_value,  
2                               max_pixel_value, cv2.THRESH_BINARY)
```

```
1 output_image = cv2.adaptiveThreshold(input_image, max_pixel_value,  
2                                         cv2.ADAPTIVE_THRESH_MEAN_C,  
3                                         cv2.THRESH_BINARY, block_size, c_const)
```

Ďalšiu implementovanú metódu predstavuje **Canny detekcia hrán**. Canny detekcia je viacstupňový algoritmus, ktorého výsledkom je binarizovaný obrázok s identifikovanými hranami objektov, ktoré sú reprezentované bielymi krivkami. V OpenCV je na jej realizáciu vytvorená funkcia **Canny()**. Vstup funkcie okrem vstupného obrázku tvoria dve prahové hodnoty pre Canny binarizáciu. Funkcia má štruktúru [7]:

```
1 output_image = cv2.cvtColor(input_image, cv2.COLOR_BGR2GRAY)
```

Poslednými metódami našej aplikácie sú **morfologické operácie**, konkrétnie **diletácia** a **erózia**. Pri diletácii dochádza k rozširovaniu kontúr objektov, naopak pri erózii k ich stenčovaniu. Knižnica OpenCV opäť ponúka implementácie týchto operácií, konkrétnie funkcie **dilate()** a **erode()**. Obe funkcie ponúkajú nastavenie výšky, respektívne šírky kernelu a počtu iterácií aplikovania danej operácie. Ich tvar je nasledovný [7]:

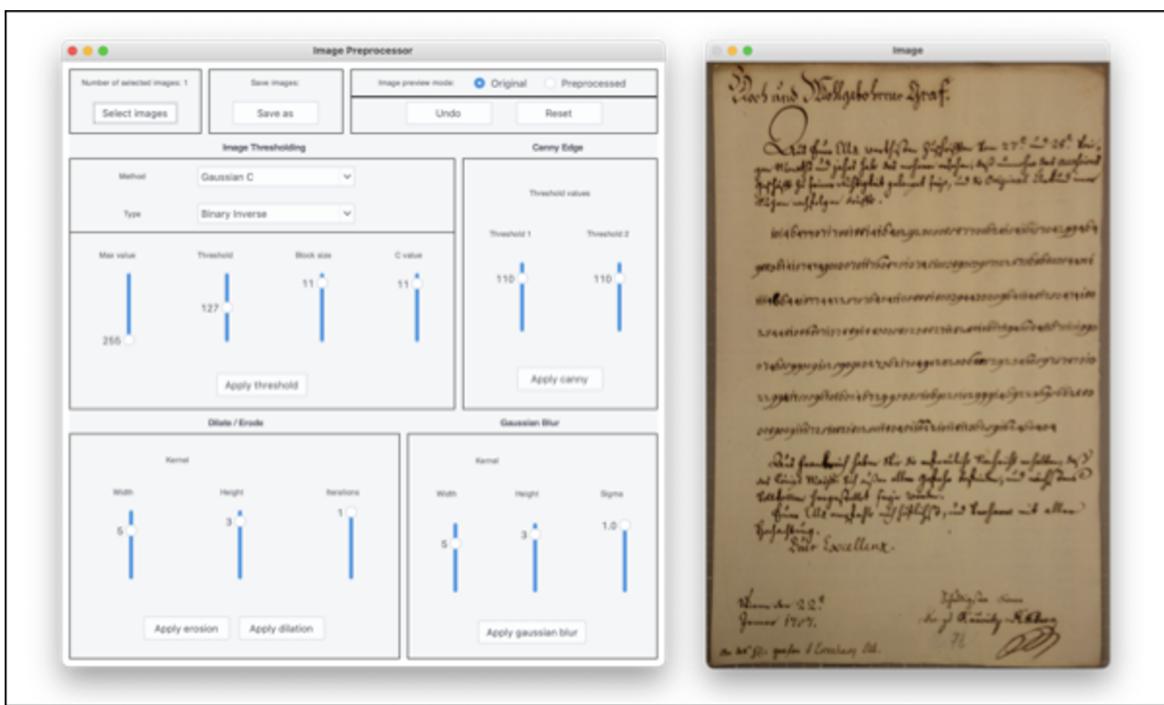
```
1 output_image = cv2.dilate(input_image, (kernel_width, kernel_height), iterations_num)

1 output_image = cv2.erode(input_image, (kernel_width, kernel_height), iterations_num)
```

5.2.2 Grafické používateľské rozhranie

Pre jednoduché používanie aplikácie bolo nutné implementovať nad logikou, ktorá využíva samotné metódy predspracovania, jednoduché a prívetivé grafické používateľské rozhranie. Pre implementáciu bola použitá Python knižnica TKinter. Táto knižnica bola zvolená pre jej jednoduchosť, prehľadnú dokumentáciu, a optimalizáciu pre všetky platформy. Knižnica ponúka niekoľko ovládacích prvkov a panelov, ktoré je možné jednoducho využiť, napríklad panely, tlačidlá, prepínače, rolovacie menu, posuvníky a mnohé iné. Samotnú štruktúru aplikácie sme kvôli prehľadnosti rozčlenili do viacerých komponentových kontajnerov. Konkrétnie ide o kontajnery pre:

- výber obrázkov,
- uloženie obrázkov,
- zmenu režimu zobrazenia náhľadu obrázku,
- vrátenie a zrušenie vykonaných zmien,
- binarizáciu,
- Canny detekciu hrán,
- diletáciu a eróziu,
- odšumenie.



Aplikácia na predspracovanie obrazu

Obr. 40: Ukážka aplikácie na predspracovanie.

Aplikácia umožňuje výber viacerých obrázkov na úpravu z konkrétneho umiestnenia pomocou dialogového okna, po ktorom sa na obrázky automaticky aplikuje prevod na grayscale. Vybraný obrázok sa zobrazí v samostatnom okne. Užívateľ má možnosť plynulo prepínať medzi náhľadom upraveného a pôvodného obrázku cez prepínače, vrátiť sa o krok späť alebo vykonať zmeny úplne resetovať do pôvodného stavu. Každá metóda predspracovania ponúka nastavenie potrebných hodnôt pre vykonanie cez posuvníky, vykonanie prebieha cez tlačidlo danej metódy. Metóda binarizácie ďalej obsahuje možnosť zvolenia typu výpočtu prahu a typu prahovania cez rolovacie menu. Zvolenie miesta pre uloženie obrázkov a samotné uloženie je možné vykonať cez dialógové okno.

5.3 Detekčný model Mask R-CNN

Druhým krokom implementačnej časti našej práce bola implementácia konkrétneho detekčného systému na rozpoznávanie rukou písaných číslíc. Konkrétnie sa jednalo o systém Mask R-CNN. Mask R-CNN je implementácia dvojúrovňového detektora vychádzajúca z Faster R-CNN modelu, ktorý využíva sieť pre návrh regiónov (RPN) na lokalizáciu a konvolučnú neurónovú sieť (CNN) na klasifikáciu objektu. Mask R-CNN navyše pridáva plne konvolučnú sieť (FCN) na segmentáciu objektu. [28]

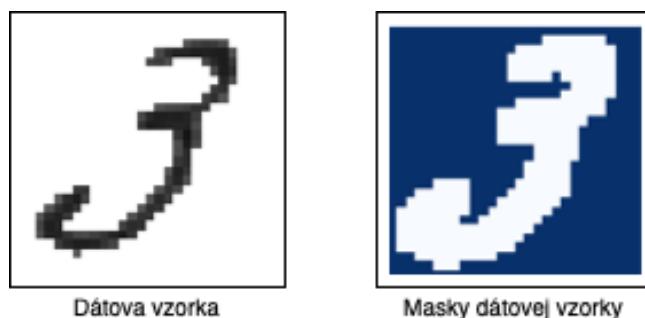
V našej práci sme sa konkrétnie rozhodli pre implementáciu využívajúcu aplikačný rámc hlbokého učenia TensorFlow. Na realizáciu experimentov bolo kvôli prehľadnosti využité webové vývojové prostredie pre Python, JupyterLab. Pri porovnaní sme sa zamerali na trénovanie Mask R-CNN na rôznych dátových množinách. Menovite sa jednalo o ARDIS, VEGA a VEGA-Cropped. Výsledky experimentov sú dostupné v elektronickej prílohe v .ipynb súboroch. Štruktúru uvádzame v prílohe A. [41]

5.3.1 ARDIS dataset

Prvým datasetom, ktorý sme využili na trénovanie a testovanie je analyzovaný dataset Ardis. Tento dataset sme zvolili pre jeho normalizovanú formu dát, ktoré zachytávajú historické číslice, čo vyhovuje nášmu zámeru detekcie číslíc v historických dokumentoch. Konkrétnie sme využili sadu ARDIS IV vo formáte .csv. Nakolko dáta v tomto formáte nekorešpondujú s formátom COCO je nutné dáta konvertovať do príslušného formátu.

Na konvertovanie formátu sme vytvorili konzolovú aplikáciu, ktorá automatizované vykoná konverziu medzi pôvodným ARDIS IV datasetom v .csv formáte na dáta vo formáte COCO. Metóda `ardis_to_coco()` má nasledovný tvar:

```
1 ardis_to_coco(source_data_ardis_path, source_labels_ardis_path,  
2                 destination_path, img_data_inverse=False, contours_examples=False)
```

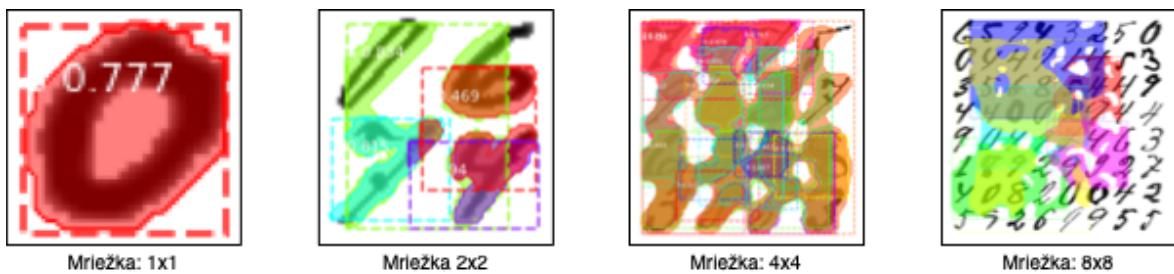


Obr. 41: Ukážka masiek dátovej vzorky: ARDIS.

Metóda pomocou knižníc Pandas, NumPy a OpenCV prevedie vstupné .csv dátá na obrázky, ku ktorým následne vytvorí anotácie v .json súbore. Ohraničujúce boxy kopírujú hranice obrázku, napokolko dátová vzorka obsahuje vždy len jednu číslicu. Segmentačné polygóny sme vytvorili pomocou získania kontúr číslice, ktoré sme následne transformovali na polia polí hodnôt reprezentujúce polygóny masiek číslic. Na obrázok sme však ešte pred získaním kontúr aplikovali diletáciu, aby sme zabezpečili správne ohraničenie číslice výslednou maskou. Posledným krokom bol prevod vzoriek z grayscale formátu do formátu RGB a zvýšenie indexu kategórii číslic o 1, napokolko hodnota 0 je pri konkrétnej Mask R-CNN implementácii rezervovaná pre pozadie.

Po upravení dát sme vykonali konfiguráciu a vytvorenie modelu na trénovanie, ktoré parametre sú dostupné v prílohe C. Transfer trénovanie prebiehalo v 80 epochách po 800 trénovacích a 50 validačných iteráciach. Vstup siete mal pevnú veľkosť 128x128 pixelov. Transfer trénovanie prebiehalo na predtrénovanej referenčnej sieti ResNet50. Trénovanie pri tejto konfigurácii trvalo 10 hodín a 18 minút.

Po natrénovaní modelu sme nakonfigurovali model na inferenciu na testovacích dátach, ktorého parametre sú dostupné v prílohe C. Prahovú hodnotu detekcie objektu sme nastavili na 0.3, vstup mal pevnú veľkosť 128x128 pixelov. Testovanie sme vykonali na vzorkách s číslicami v mriežke 1x1, 2x2, 4x4 a 8x8.



Obr. 42: Ukážka inferencie Mask R-CNN: ARDIS.

Pri výhodnotení úspešnosti modelu sme sa zamerali na metriky úspešnosti klasifikácie, presnosti detekcie, citlivosti detekcie a F1 skóre detekcie. Úspešnosť klasifikácie reprezentuje schopnosť modelu správne určiť príslušnosť objektu k triede, presnosť detekcie reprezentuje schopnosť modelu správne detektovať želané objekty, citlosť detekcie vyjadruje všeobecnú schopnosť modelu detektovať objekty a F1 skóre detekcie vyjadruje celkovú mieru presnosti modelu. Presnú definíciu metrík a ich výpočtu uvádzame v prílohe F.

Úspešnosť modelu sme vyjadrili v nasledujúcej tabuľke.

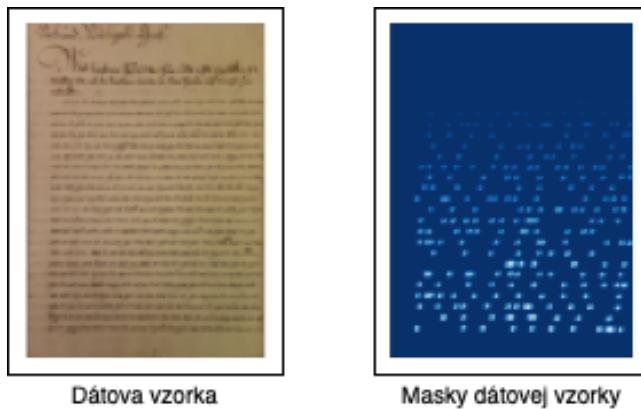
	1x1	2x2	4x4	8x8
Presnosť detekcie	98%	0%	0%	0%
Citlivosť detekcie	100%	0%	0%	0%
F1 skóre detekcie	99%	0%	0%	0%
Úspešnosť klasifikácie	94,86%	0%	0%	0%

Tabuľka 3: Úspešnosť detekcie a klasifikácie Mask R-CNN: ARDIS

Na základe vyššie uvedených zistení je zrejmé, že model vie optimálne pracovať so vstupmi v rovnejakej forme, na ktorých bol trénovaný, teda číslice v mriežke 1x1. Akonáhle len mierne zmeníme štruktúru a počet objektov na jednom obrázku zvýšime, siet má s detekciou problémy. Tento jav je zapríčinený RPN sietou pre návrh regiónov záujmu, ktorá pri trénovaní na vzorkách v mriežke 1x1 nie je vo svojej výslednej podobe schopná správne navrhovať regióny záujmu a tým pádom model nie je schopný detektovať vyšší počet objektov na obrázkoch.

5.3.2 VEGA dataset

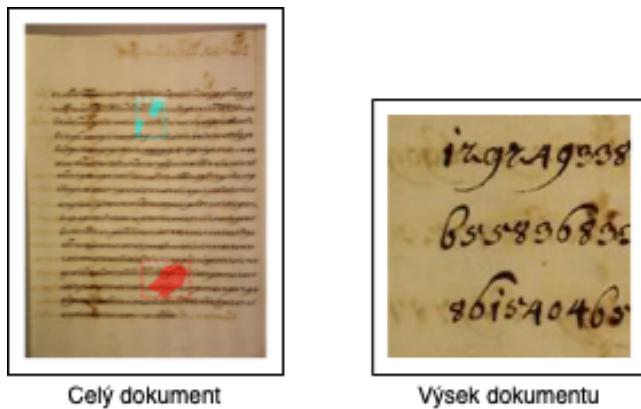
Druhým testovaným datasetom na trénovanie Mask R-CNN modelu bol dataset historických číslí VEGA. VEGA dataset bol vytvorený výskumným tímom na Fakulte elektrotechniky a informatiky Slovenskej technickej univerzity v Bratislave v rámci výskumného projektu VEGA 2/0072/20 - Moderné metódy spracovania šifrovaných archívnych dokumentov. Dataset bol vytvorený z archívnych historických dokumentov a ich transkripcíí pomocou Python knižnice na vytvorenie anotácií Labelme. Anotácie boli vytvorené ručným vyznačením masiek číslí na konkrétnom digitalizovanom dokumente. Anotácie boli následne prevedené do formátu COCO. Dataset obsahuje 9 568 číslí na 12 dokumentoch v trénovacej sade, 1 311 číslí na 3 dokumentoch vo validačnej sade a 1 554 číslí na 3 dokumentoch v testovacej sade. Veľkosť všetkých digitálnych dokumentov je 4160x6240 pixelov. Samotné vzorky (dokumenty) však boli kvôli vysokým hardvérovým nárokom na trénovanie spolu s anotáciami zmenšené na veľkosť 640x1024 pixelov. [42]



Obr. 43: Ukážka masiek dátovej vzorky: VEGA.

Konfigurácia modelu na trénovanie je uvedená v prílohe D. Transfer trénovanie pri VEGA datasete prebiehalo kvôli vysokým hardvérovým nárokom len na 1 epochu po 1660 trénovacích a 50 validačných iteráciach. Vstup siete mal maximálnu veľkosť 1024x1024 pixelov. Transfer trénovanie prebehlo na predtrénovanej sieti ResNet50. Trénovanie trvalo 4 hodiny a 13 minút.

Natrénovaný model sme následne nakonfigurovali na inferenciu na testovacích dátach. Parametre sú dostupné v prílohe D. Prahovú hodnotu detekcie objektu sme nastavili na 0.3, vstup mal pevnú veľkosť 1024x1024 pixelov. Testovanie sme vykonali na vzorkách celých dokumentov a na rozčlenených (cropped) vzorkách vytvorených z pôvodných (celých) dokumentov.



Obr. 44: Ukážka inferencie Mask R-CNN: VEGA.

Úspešnosť modelu sme rovnako vyjadrili v tabuľke. Takisto sme sa zamerali na metriky úspešnosti klasifikácie, presnosti detekcie, citlivosti detekcie a F1 skóre detekcie.

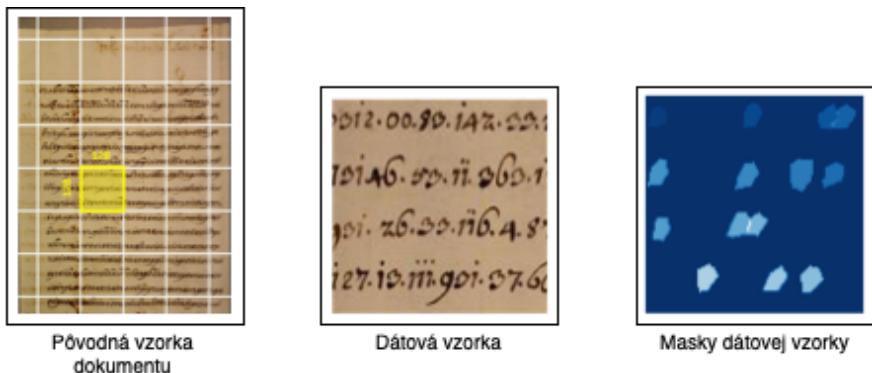
	Celý dokument	Výsek dokumentu
Presnosť detekcie	0%	0%
Citlivosť detekcie	0%	0%
F1 skóre detekcie	0%	0%
Úspešnosť klasifikácie	0%	0%

Tabuľka 4: Úspešnosť detekcie a klasifikácie Mask R-CNN: VEGA.

Na základe vyššie uvedených zistení je zrejmé, že model má napriek natrénovaní na dátach s vyšším počtom číslic na vzorke problém s detekciou na testovacích dátach zhodnej štruktúry, teda celého dokumentu. Následne došlo k otestovaniu modelu na rozčlenených častiach dokumentu, model však rovnako číslice nedokázal detektovať. Potencionálnou príčinou tohto problému môže byť veľkosť číslic na vzorkách, ktorá je pri porovnaní s veľkosťou dokumentu zanedbateľná. Z tohto dôvodu RPN sieť nedokáže optimálne lokalizovať návrhy objektov záujmu.

5.3.3 VEGA-Cropped dataset

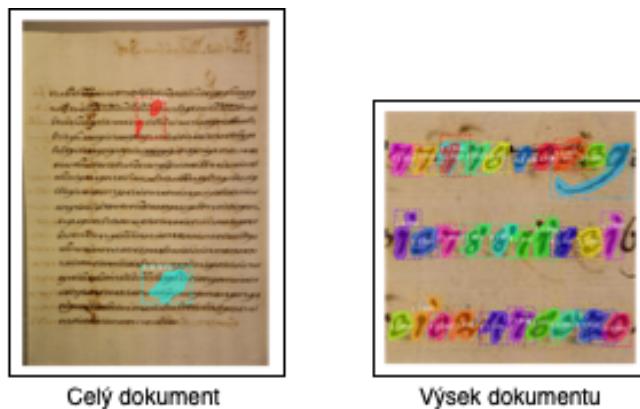
Posledným testovaným datasetom využitým na trénovanie a testovanie bola modifikovaná množina VEGA, VEGA-Cropped. VEGA-Cropped dataset vznikol ako reakcia na nízku úspešnosť systému Mask R-CNN trénovanom na vzorkách VEGA. Vzorky veľkosti 128x128 pixelov boli zhotovené pomocou rozdelenia obrázkov dokumentov v pôvodnom datasete. Na základe tohto rozdelenia boli modifikované aj ich príslušné anotácie. Dataset obsahuje 9 568 číslic na 234 obrázkoch v trénovacej sade, 1 311 číslic na 45 obrázkoch vo validačnej sade a 1 554 číslic na 54 obrázkoch v testovacej sade. Modifikácia pôvodného datasetu bola vytvorená rovnako ako pôvodný dataset výskumným tímom na Fakulte elektrotechniky a informatiky Slovenskej technickej univerzity v Bratislave v rámci výskumného projektu VEGA 2/0072/20 - Moderné metódy spracovania šifrovaných archívnych dokumentov. [42]



Obr. 45: Ukážka masiek dátovej vzorky: VEGA-Cropped.

Konfigurácia modelu pre trénovanie je uvedená v prílohe E. Transfer trénovanie na VEGA-Cropped datasete prebiehalo na 100 epochách po 1660 trénovacích a 50 validačných iteráciach. Vstup siete bol pevnej veľkosti 128x128 pixelov. Transfer trénovanie prebehlo na predtrénovanej sieti ResNet50. V práci sme využili už natrénovanú sieť zhodnej konfigurácie vytvorenú v rámci výskumného projektu VEGA 2/0072/20. [42]

Model sme následne nakonfigurovali do inferenčného módu, ktorého parametre sú rovnako uvedené v prílohe E. Prahová hodnota bola nastavená na 0.8, veľkosť vstupu na 128x128 pixelov. Testovanie bolo vykonané na vzorkách celých dokumentov a na rozčlenených (cropped) vzorkách.



Obr. 46: Ukážka inferencie Mask R-CNN: VEGA-Cropped.

Úspešnosť modelu sme zhodne ako v prvých dvoch prípadoch vyjadrili v tabuľke. Obsahuje metriky úspešnosti klasifikácie, presnosti detekcie, citlivosti detekcie a F1 skóre detekcie.

	Celý dokument	Výsek dokumentu
Presnosť detekcie	0%	~92,19%
Citlivosť detekcie	0%	~90,08%
F1 skóre detekcie	0%	~91,12%
Úspešnosť klasifikácie	0%	~96,19%

Tabuľka 5: Úspešnosť detekcie a klasifikácie Mask R-CNN: VEGA-Cropped.

Na základe vyššie uvedených metrík je zrejmé, že model ani pri trénovaní na rozdeľených vzorkách dokumentov pôvodného datasetu VEGA a lepšej schopnosti RPN siete navrhovať regióny záujmu na obrázku, nedokáže detektovať číslice na celom dokumente. Táto nedokonalosť je zapríčinená pomerom veľkosti číslice k veľkosti celého dokumentu, kde je veľkosť číslice zanedbateľná, s čím sa architektúra RPN siete nedokáže vysporiať. Pri testovaní siete na testovacích dátach štruktúry podobnej tým trénovacím, teda na výsekokoch z pôvodného dokumentu, model dosahuje veľmi dobré výsledky detekcie, klasifikácie aj segmentácie číslic. Číslice v tomto prípade zaberajú väčšiu časť vzorky, čo napomáha modelu dosiahnuť lepšie výsledky. Nevýhodou tohto riešenia však môže byť nulová alebo veľmi obmedzená schopnosť modelu detektovať číslice na hranici medzi rozdelenými vzorkami, kde je na obrázku prítomná len určitá časť pôvodnej číslice.

5.3.4 Zhrnutie

Prvým testovaným a porovnávaným detekčným systémom bola implementácia modelu Mask R-CNN. Model sme testovali na datasetoch ARDIS, VEGA a VEGA-Cropped. Model patrí medzi dvojúrovňové detektory a pracuje s dátami vo formáte COCO.

V praxi najpoužiteľnejšie výsledky sme dosiahli pri datasete VEGA-Cropped, ktorý pri natrénovaní na rozčlenených malých častiach pôvodného dokumentu dosahoval pri testovaní na dátach rovnakého formátu výborné výsledky. Tento fakt zapríčinila vhodné natrénovaná RPN siet pre návrh regiónov v kombinácii s číslicami vhodnej mierky k celkovému obrázku. Vzorky taktiež vdaka svojej menšej veľkosti mali za následok skrátenie trénovacieho času, čo viedlo k robustnejšiemu trénovaniu v priateľnom čase. Tento fakt mal za následok lepšiu úspešnosť siete pri detekcii, klasifikácii aj segmentácii.

Naopak využíte datasetov ARDIS a VEGA sa kvôli ich štruktúre a podobe vzoriek ukázali ako nevhodné.

Model natrénovaný na datasete ARDIS vykazoval výborné výsledky pri testovaní na vzorkách s jednou číslicou, no pri zvýšení počtu číslic na vzorke ich model nedokázal detektovať. Tento jav bol zapríčinený nedostatočne natrénovanou RPN sieťou. Napriek malej veľkosti vzoriek a krátkemu trénovaciemu času tento fakt výsledky modelu testovanom na vzorkách s vyšším počtom číslic neovplyvnil.

Model natrénovaný na pôvodnom datasete VEGA sa v praxi ukázal ako nepoužiteľný, napäťko jeho schopnosť detektie číslic nebola preukázaná na žiadnom nami zvolenom testovacom scenárii. Ďalšou výraznou nevýhodou tejto konfigurácie bol veľmi dlhý trénovací čas, ktorý bol zapríčinený veľkými rozmermi trénovacích a validačných vzoriek.

5.4 Detekčný model YOLO v5

Posledným krokom našej práce bola implementácia ďalšieho detekčného systému na rozpoznávanie rukou písaných číslíc. Jednalo sa o systém YOLO vo verzí 5. YOLO v5 je implementácia jednoúrovňového detektora vychádzajúca z pôvodného modelu YOLO. YOLO v5 teda na detekciu a následnú klasifikáciu využíva jednu konvolučnú neurónovú siet, čo prispieva k nízkej časovej náročnosti trénovania a aj rozpoznávania. [30]

V našej práci sme využili pôvodnú implementáciu využívajúcu aplikačný rámec hlbokého učenia PyTorch. Na realizáciu experimentov bolo použité webové vývojové prostredie JupyterLab. Pri porovnaní sme sa zamerali na trénovanie YOLO v5 na rovnakých dátových množinách ako v predošлом kroku, teda ARDIS, VEGA a VEGA-Cropped. Výsledky experimentov sú dostupné v elektronickej prílohe v .ipynb súboroch. Štruktúru uvádzame v prílohe A. [30]

5.4.1 ARDIS dataset

Prvým porovnávaným datasetom bol rovnako ako v predošлом kroku dataset ARDIS. Nakolko sú dáta v datasete ARDIS IV vo formáte .csv, bolo nutné ich takisto previesť do príslušného formátu modelu YOLO v5.

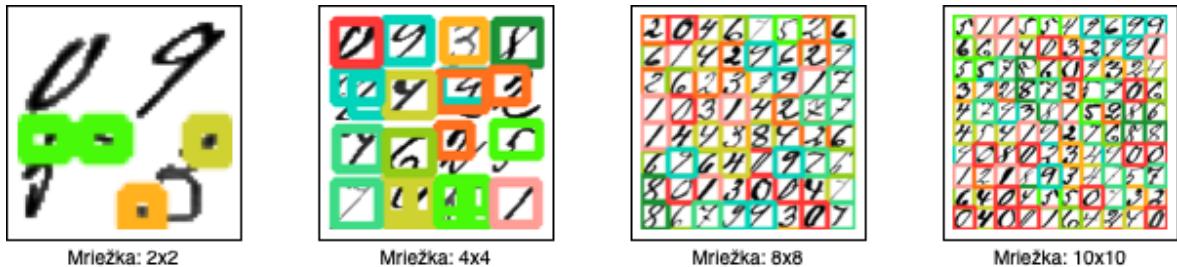
Na konverziu sme vytvorili obdobnú konzolovú aplikáciu ako v predošlej časti, ktorá konverziu na YOLO Darknet TXT formát vykonáva automatizované. Metóda `ardis_to_yolov5()` má nasledovný tvar:

```
1 ardis_to_yolov5(source_train_data_ardis_path, source_val_data_ardis_path,  
2                     source_train_labels_ardis_path, source_val_labels_ardis_path,  
3                     destination_path, img_data_inverse=False)
```

Metóda pomocou knižníc Pandas, NumPy a OpenCV prevedie vstupné .csv dátu na obrázky, ku ktorým následne vytvorí príslušné anotácie v .txt súboroch. Ohraničujúce boxy kopírujú hranice obrázku, nakoľko dátová vzorka obsahuje vždy len jednu číslicu. Informácie o triede a ohraničujúcim boxe sme následne zapísali do konkrétneho .txt súboru. Po konverzii na príslušný formát sme obrázky previedli z grayscale do RGB formátu.

Po úprave dát sme spustili proces trénovania. Transfer trénovanie prebiehalo v 3000 epochách po 8 vzorkách v jednej dávke. Vstup siete mal pevnú veľkosť 128x128 pixelov. Transfer trénovanie prebiehalo na predtrénovanej referenčnej sieti YOLOv5s a bolo ukončené po čase 1 hodina a 32 minút po 236 epoche.

Po natrénovaní modelu sme nakonfigurovali model na inferenciu na testovacích dátach. Prahovú hodnotu detekcie objektov sme nastavili na 0.5. Testovanie sme vykonali na vzorkách s číslicami v mriežke 2x2, 4x4, 8x8, 10x10, 12x12, 14x14, 16x16 a 18x18.



Obr. 47: Ukážka inferencie YOLO v5: ARDIS.

Úspešnosť modelu sme vyjadrili v tabuľke. Opäťovne sme sa zamerali na metriky úspešnosti klasifikácie, presnosti detekcie, citlivosti detekcie a F1 skóre detekcie.

	2x2	4x4	8x8	10x10	16x16	18x18
Presnosť detekcie	0%	21,03%	99,69%	99,53%	95,76%	68,97%
Citlivosť detekcie	0%	26,97%	99,84%	99,76%	61,72%	18,52%
F1 skóre detekcie	0%	23,63%	99,76%	99,45%	75,06%	29,19%
Úspešnosť klasifikácie	0%	97,56%	99,53%	99,37%	71,52%	41,67%

Tabuľka 6: Úspešnosť detekcie a klasifikácie YOLO v5: ARDIS

Na základe hodnôt uvedených v tabuľke je zrejmé, že model trénovaný na dátovej množine ARDIS dokáže veľmi spoľahlivo detektovať a klasifikovať aj viac čísl na jednej vzorke napriek natrénovaniu na vzorkách len s jednou číslou. Tento fakt je zapríčinený architektúrou siete modelu YOLO v5. Limitácie fungovania konkrétneho riešenia však nachádzame pri velmi nízkom, respektívne veľmi vysokom počte čísl na vzorkách, ktoré schopnosť detekcie absolútne eliminujú. Výhodou tohto riešenia je však veľmi krátky čas trénoania modelu a jeho následná schopnosť úspešnej detekcie vo väčšine potencionálnych scenároch použitia.

5.4.2 VEGA dataset

Ďalším porovnaným datasetom pri implementácii modelu YOLO v5 bol dataset VEGA. Nám dostupná verzia tohto datasetu je uchovaná vo formáte COCO, čo vedie opäťovne k potrebe konverzie, v tomto prípade anotácie, do formátu YOLO Darknet TXT. [42]

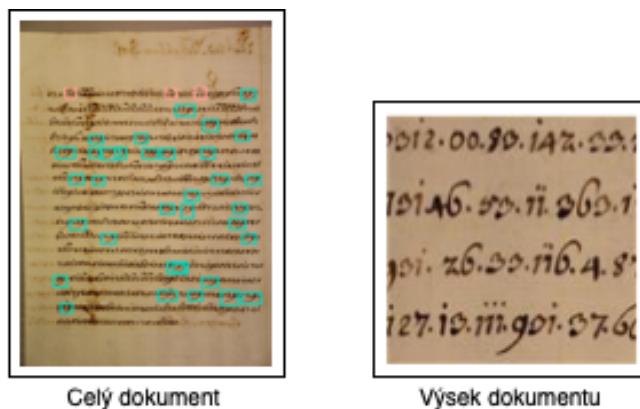
Na konverziu bol využitý modifikovaný voľne dostupný skript na prevod medzi formátmi anotácií datasetov obraz spracujúcich systémov. Metóda vykonávajúca tento typ konverzie **convert_coco_json_to_yolo_txt()** je v tvare [43]:

```
1 convert_coco_json_to_yolo_txt(output_path, json_file)
```

Metóda vykoná prevod .json anotácie formátu COCO do .txt anotácie YOLO formátu pre každú dátovú vzorku. Obrázky, respektívne dátové vzorky nemeníme, nakolko sú natívne v správnom formáte. Ich veľkosť je 640x1024 pixelov.

Po konverzií anotácie nasledoval proces trénovania. Transfer trénovanie prebiehalo v 3000 epochách po jednej vzorke v jednej dávke s veľkosťou vstupu 1024x1024 pixelov. Transfer trénovanie prebiehalo zhodne na predtrénovanej referenčnej sieti YOLOv5s. K ukončeniu trénovania došlo v 211 epoche po 17 minútach.

Po natrénovaní modelu sme nakonfigurovali model na inferenciu na testovacích dátach. Prahovú hodnotu detekcie objektov sme nastavili na 0.25. Testovanie bolo vykonané na vzorkách celých dokumentov a na rozčlenených (cropped) vzorkách.



Obr. 48: Ukážka inferencie YOLO v5: VEGA.

Úspešnosť modelu sme vyjadrili nižšie. Uvedené sú metriky úspešnosti klasifikácie, presnosti detekcie, citlivosti detekcie a F1 skóre detekcie.

	Celý dokument	Výsek dokumentu
Presnosť detekcie	~83,87%	0%
Citlivosť detekcie	~3,83%	0%
F1 skóre detekcie	~7,3%	0%
Úspešnosť klasifikácie	~89,52%	0%

Tabuľka 7: Úspešnosť detekcie a klasifikácie YOLO v5: VEGA.

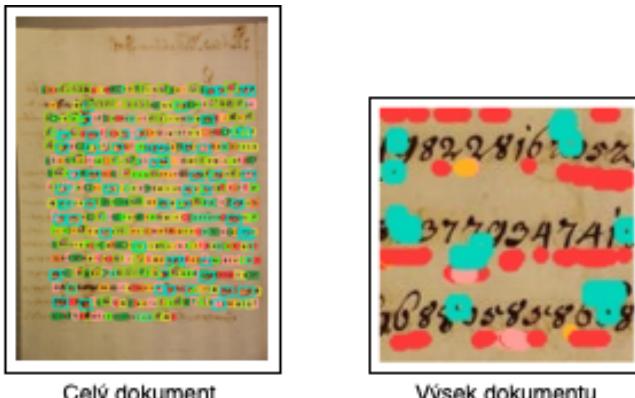
Podľa tabuľkových metrík je badateľná veľmi obmedzená schopnosť modelu detektovať číslice na celých dokumentoch, čo je opäťovne spôsobené architektúrou siete modelu. Konfigurácia modelu taktiež nie je schopná detektovať číslice na výsekokach z pôvodných vzoriek.

5.4.3 VEGA-Cropped dataset

Posledným datasetom určeným na testovanie bola modifikovaná verzia datasetu VEGA, VEGA-Cropped. VEGA-Cropped dataset sa skladá z výsekov častí dokumentov pôvodného datasetu. Nám dostupná verzia tohto datasetu je uchovaná vo formáte COCO, čo vedie opäťovne k potrebe konverzie, v tomto prípade anotácií do formátu YOLO Darknet TXT. Na konverziu bol využitý rovnako ako pri pôvodnom datasete VEGA modifikovaný voľne dostupný skript na prevod medzi formátmi anotácií datasetov obraz spracujúcich systémov. [42, 43]

Transfer trénovanie prebiehalo v 3 000 epochách po ôsmych vzorkách v jednej dávke s veľkosťou vstupu 128x128 pixelov. Transfer trénovanie prebiehalo zhodne na predtrénovanej referenčnej sieti YOLOv5s. K ukončeniu trénovania došlo v 429 epoche po 1 hodine a 27 minútach.

Po natrénovaní modelu sme nakonfigurovali model na inferenciu na testovacích dátach. Prahovú hodnotu detekcie objektov sme nastavili na 0.25. Testovanie bolo vykonané na vzorkách celých dokumentov a na rozčlenených (cropped) vzorkách.



Obr. 49: Ukážka inferencie YOLO v5: VEGA-Cropped.

Úspešnosť modelu sme zhodne ako v ostatných prípadoch vyjadrili v tabuľke. Obsahuje metriky úspešnosti klasifikácie, presnosti detekcie, citlivosti detekcie a F1 skóre detekcie.

	Celý dokument	Výsek dokumentu
Presnosť detekcie	~90,96%	0%
Citlivosť detekcie	~82,79%	0%
F1 skóre detekcie	~86,69%	0%
Úspešnosť klasifikácie	~96,24%	0%

Tabuľka 8: Úspešnosť detekcie a klasifikácie YOLO v5: VEGA-cropped.

Podľa zistených hodnôt možno odpozorovať nulovú schopnosť modelu detektovať číslice na výsekokoch z pôvodného dokumentu testovacej sady VEGA, napriek ich štruktúre, ktorá priamo korešponduje s trénovacou sadou konkrétneho modelu. Naopak pri detekcií na pôvodnom dokumente model dosahuje priateľnejšie výsledky. Za problémy konkrétneho modelu je opäť zodpovedná samotná architektúra daného riešenia.

5.4.4 Zhrnutie

Druhým testovaným a porovnávaným detekčným systémom bola implementácia modelu YOLO v5. Model sme rovnako ako Mask R-CNN testovali na datasetoch ARDIS, VEGA a VEGA-Cropped. Model patrí medzi jednoúrovňové detektory a pracuje s dátami vo formáte YOLO Darknet TXT.

V praxi použiteľné výsledky sme dosiahli pri trénovaní na datasetoch ARDIS a VEGA-Cropped.

Model trénovaný na datasete ARDIS s jednou číslicou na vzorke preukazoval výborné výsledky pri detekcií a klasifikácii v rôznych mriežkach. Limitácie modelu sa prejavili pri scenároch s veľmi vysokým a nízkym počtom číslíc na vzorke. Tento fakt má za následok samotná architektúra systému YOLO. Pozitívom tejto konfigurácie bol vzhľadom na výsledky prijateľný čas trénovania.

Prijateľné výsledky boli dosiahnuté aj pri trénovaní modelu na dátovej množine VEGA-Cropped. Konfigurácia mala značné problémy pri detekcii číslíc na výsekokoch pôvodných VEGA vzoriek, no pri detekcií na celých dokumentoch dosahovala uspokojivé výsledky. Úspešnosť klasifikácie bola v oboch prípadoch veľmi vysoká.

Model trénovaný na pôvodnom VEGA datasete číslice nedokázal úspešne detektovať ani v jednom nami testovanom scenári.

5.5 Porovnanie a zhrnutie - Mask R-CNN, YOLO v5

Po implementácii a otestovaní detekčných modelov Mask R-CNN a YOLO v5 v rôznych scenároch použitia sme dospeli k nasledujúcim záverom. Kompletné výsledky meraní úspešnosti modelov uvádzame v prílohe G.

Mask R-CNN model sme testovali na datasetoch ARDIS, VEGA a VEGA-Cropped. Mask R-CNN implementácia dosiahla najlepšie výsledky pri datasete VEGA-Cropped, najhoršie výsledky sme naopak zaznamenali pri datasete VEGA. Z týchto skutočností je možné usúdiť, že model má výrazné problémy s detekciou objektov, ktoré zaberajú malú časť vzorky, čo často vedie k potrebe rozdelenia vzoriek na menšie časti. Ďalším problémom systému je nutnosť trénovať na dátach podobnej formy a štruktúry, ako budú reálne vstupy. Táto požiadavka vychádza z potreby natrénovania RPNA siete pre optimálny návrh regiónov záujmu. Ďalším negatívom je časová náročnosť pri samotnom trénovaní. Pozitívom je ale schopnosť vhodne navrhnutého modelu spoloahlivo detektovať a klasifikovať objekty na vhodne zvolenom vstupe. Mask R-CNN sa teda vyznačuje veľmi dobrými výsledkami pri samotnej činnosti, sú však priamo závislé od úspešnej konfigurácie, štruktúry datasetu a v neposlednom rade od dĺžky trénovacieho procesu.

Model YOLO v5 sme zhodne testovali na datasetoch ARDIS, VEGA a VEGA-Cropped. YOLO v5 dosiahol najlepšie výsledky na datasetoch ARDIS a VEGA-Cropped. Nulové výsledky sme zaznamenali pri datasete VEGA. Z výsledkov testovania je možné vyvodíť niekoľko záverov. Model YOLO v5 vykazoval menšiu mieru závislosti fungovania systému na samotnej konfigurácii modelu, štruktúre dátových vzoriek a čase trénovania ako model Mask R-CNN. Jeho ďalšou výhodou je niekoľko násobne nižšia časová náročnosť pri samotnej detekcii a klasifikácii. Konfigurácia modelu a jeho trénovanie je teda veľmi užívateľsky prívetivé, pri detekcii však napriek tomu ponúka relevantné výsledky. Pri zložitejších prípadoch využitia však na správne nakonfigurovanú a vhodne natrénovanú siet Mask R-CNN stráca. Model ma navyše problém detektovať malé a veľké objekty na vzorkách, túto skutočnosť však nie je možné odstrániť, nakoľko je spôsobená samotnou architektúrou siete na pozadí.

Záver

V tejto práci sme sa oboznámili so štruktúrou kryptosystému nomenklátor a s dostupnými metódami a riešeniami HTR a HCR na rozpoznávanie rukou písaných číslíc.

Po analýze fungovania a štruktúry týchto systémov sme sa zamerali na ich implementáciu a porovnanie.

Prvým krokom bola implementácia oknovej aplikácie na predspracovanie obrazu, ktorá dokáže aplikovať rôzne metódy na úpravu a modifikáciu obrazových dát pre zlepšenie fungovania výsledného HCR systému.

Po tomto kroku sme zozbierali voľne dostupné vyhovujúce datasety obsahujúce rukou písané číslice, ktoré sme podrobili porovnaniu, analýze a normalizácii.

Tieto poznatky sme využili pri implementácii detekčných systémov Mask R-CNN a YOLO v5, ktoré sme po úprave formátu dát natrénovali na najvhodnejších nami získaných dátových množinách. Ich funkčnosť a schopnosť detektovať objekty sme následne otestovali, vyhodnotili a porovnali.

Z výsledkov našich meraní je zrejmé, že každé riešenie prináša pozitíva aj negatíva. Konkrétny model a jeho konfiguráciu je nutné personalizovať a prispôsobiť scenárom a podmienkam, v akých bude fungovať.

Zoznam použitej literatúry

1. FISCHER, Andreas, LIWICKI, Marcus a INGOLD, Rolf. *Handwritten Historical Document Analysis, Recognition, And Retrieval: State Of The Art And Future Trends*. World Scientific, 2020. ISBN 978-9811203237.
2. ANTAL, Eugen a GROŠEK, Otokar. *Moderná kryptoanalýza klasických šifier*. Diz. pr. Slovenská technická univerzita v Bratislave, 2017.
3. GONO, Tomáš a ANTAL, Eugen. *Digitalizácia a spracovanie nomenklátorových šifrovacích kľúčov*. Dp. pr. Slovenská technická univerzita v Bratislave, 2021.
4. KAHN, David. *The Code-Breakers: The Story of Secret Writing*. The Macmillan Company, 1974. ISBN 978-0025604605.
5. DUNIN, Elonka a SCHMEH, Klaus. *Codebreaking: A Practical Guide*. Robinson, 2020. ISBN 978-1472144218.
6. ANTAL, Eugen a ZAJAC, Pavol. *Analýza Rabenhauptovo zašifrovaného dopisu*. Crypto-World, 2013. Č. 11-12. ISSN 1801-2140. Dostupné tiež z: http://crypto-world.info/casopis15/crypto1112_13.pdf.
7. DOMORÁK, Martin, ĎURFINA, Jaroslav, KISS, Mátyás, MAJTÁN, Patrik, MICHALE, Martin, ŠTRBA, Martin a TÖRÖK, Patrícia. *Porovnanie metód a nástrojov na rozpoznávanie rukopisov: Shougao*. Slovenská technická univerzita v Bratislave, 2021.
8. BROWNLEE, Jason. *Deep Learning for Computer Vision: Image Classification, Object Detection, and Face Recognition in Python*. Machine Learning Mastery, 2019.
9. HUANG, Thomas Shi-Tao. *Computer Vision: Evolution and Promise*. University of Illinois at Urbana-Champaign, 2009. Dostupné tiež z: <https://cds.cern.ch/record/400313/files/p21.pdf>.
10. ROSYDA, Salma Shofia a PURBOYO, Tito Waluyo. A Review of Various Handwriting Recognition Methods. *International Journal of Applied Engineering Research*. 2018, roč. 13, č. 2, s. 1155–1164. ISSN 0973-4562. Dostupné tiež z: https://www.ripublication.com/ijaer18/ijaerv13n2_44.pdf.
11. JOHNSON, Stephen. *On Digital Photographyv*. O'Reilly Media, 2006. ISBN 978-0596523701.

12. JOHNSON, Hans J., MCCORMICK, Matthew M. a IBÁNEZ, Luis. *The ITK Software Guide: Introduction and Development*. 2021. Dostupné tiež z: <https://itk.org/ItkSoftwareGuide.pdf>.
13. BORAD, Anand. Understanding Object Localization with Deep Learning [online]. 2021 [cit. 2021-12-18]. Dostupné z : <https://www.einfochips.com/blog/understanding-object-localization-with-deep-learning/>.
14. ROSEBROCK, Adrian. Sliding Windows for Object Detection [online]. 2015 [cit. 2021-12-18]. Dostupné z : <https://pyimagesearch.com/2015/03/23/sliding-windows-for-object-detection-with-python-and-opencv/>.
15. GIRSHICK, Ross, DONAHUE, Jeff, DARRELL, Trevor a JITENDRA, Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. 2013. Dostupné tiež z: <https://arxiv.org/pdf/1311.2524.pdf>.
16. UIJLINGS, Jasper R. R., GEVERS, Theo, SMEULDERS, Arnold W. M. a SANDE, Koen van de. Selective Search for Object Recognition. *International Journal of Computer Vision* [online]. 2013, roč. 2, č. 104, s. 154–171 [cit. 2021-12-19]. Dostupné z : <https://ivi.fnwi.uva.nl/isis/publications/2013/UijlingsIJCV2013/UijlingsIJCV2013.pdf>.
17. SHANDILYA, Shivam. Image Segmentation With Felzenszwalb's Algorithm [online]. 2021 [cit. 2021-12-19]. Dostupné z : <https://www.analyticsvidhya.com/blog/2021/05/image-segmentation-with-felzenszwalbs-algorithm/>.
18. GIRSHICK, Ross. Fast R-CNN. 2015. Dostupné tiež z: <https://arxiv.org/pdf/1504.08083.pdf>.
19. REN, Shaoqing, HE, Kaiming, GIRSHICK, Ross a SUN, Jian. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. 2016. Dostupné tiež z: <https://arxiv.org/pdf/1506.01497.pdf>.
20. KARMARKAR, Tanay. Region Proposal Network (RPN): Backbone of Faster R-CNN [online]. 2018 [cit. 2022-04-11]. Dostupné z : <https://medium.com/egen/region-proposal-network-rpn-backbone-of-faster-r-cnn-4a744a38d7f9>.
21. REDMON, Joseph, DIVVALA, Santosh, GIRSHICK, Ross a FARHADI, Ali. You Only Look Once: Unified, Real-Time Object Detection. 2016. Dostupné tiež z: <https://arxiv.org/pdf/1506.02640.pdf>.
22. GREGOR, Michal, NEMEC, Dušan, HRUBOŠ, Marián a SPALEK, Juraj. *Umelá inteligencia 2: Hlboké učenie*. CEIT, a.s., 2017. ISBN 978-80-89865-03-1.

23. GREGOR, Michal. *Umelá inteligencia 1*. CEIT, a.s., 2014. ISBN 978-80-971684-1-4.
24. KOUTSANTONIS, L. Image Convolution with GPU and CUDA [online]. 2021 [cit. 2022-05-28]. Dostupné z : <https://ulhpc-tutorials.readthedocs.io/en/latest/cuda/exercises/convolution/>.
25. Max-pooling / Pooling [online]. 2018 [cit. 2022-05-29]. Dostupné z : https://computersciencewiki.org/index.php/Max-pooling_-_Pooling.
26. BANDYOPADHYAY, Hmrishav. An Introduction to Image Segmentation: Deep Learning vs. Traditional [online]. 2022 [cit. 2022-04-15]. Dostupné z : <https://www.v7labs.com/blog/image-segmentation-guide#image-segmentation>.
27. LONG, Jonathan, SHELHAMER, Evan a DARRELL, Trevor. Fully Convolutional Networks for Semantic Segmentation. 2015. Dostupné tiež z: https://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Long_Fully_Convolutional_Networks_2015_CVPR_paper.pdf.
28. HE, Kaiming, GKIOXARI, Georgia, DOLLA'R, Piotr a GIRSHI, Ross. Mask R-CNN. 2018. Dostupné tiež z: <https://arxiv.org/pdf/1703.06870.pdf>.
29. LIN, Tsung-Yi, MAIRE, Michael, BELONGIE Serge Bourdev, Lubomir, GIRSHICK, Ross, HAYS, James, PERONA, Pietro, RAMANAN, Deva, ZITNICK, C. Lawrence a DOLLÁR, Piotr. Microsoft COCO: Common Objects in Context. 2015. Dostupné tiež z: <https://arxiv.org/pdf/1405.0312v3.pdf>.
30. JOCHER, Glen. YOLOv5: Documentation [online]. 2020 [cit. 2022-05-19]. Dostupné z : <https://docs.ultralytics.com>.
31. PONNUSAMY, Arun. Preparing Custom Dataset for Training YOLO Object Detector [online]. 2019 [cit. 2022-05-19]. Dostupné z : <https://www.visiongeek.io/2019/10/preparing-custom-dataset-for-training-yolo-object-detector.html>.
32. HOCHULI, A. G., BRITTO JR, A. S., BARDDAL, J. P., OLIVEIRA, L.S. a SABOURIN, R. An End-To-End Approach for Recognition of Modern and Historical Handwritten Numeral Strings. *IEEE International Joint Conference on Neural Networks*. 2020.
33. KUSETOGULLARI, H., YAVARIABDI, A. a CHEDDAD, A. ARDIS: a Swedish historical handwritten digit dataset. 2020. Dostupné tiež z: <https://doi.org/10.1007/s00521-019-04163-3>.

34. CAMPOS, Teófilo E. de, BABU, Bodla Rakesh a VARMA, Manik. Character recognition in natural images. 2009. Dostupné tiež z: http://personal.ee.surrey.ac.uk/Personal/T.Decampos/papers/decampos_etal_visapp2009.pdf.
35. KUSETOGULLARI, Huseyin, YAVARIABDI, Amir, HALL, Johan a LAVESSON, Niklas. DIGITNET: A Deep Handwritten Digit Detection and Recognition Method Using a New Historical Handwritten Digit Dataset. *Big Data Research*. 2020.
36. KUSETOGULLARI, Huseyin, YAVARIABDI, Amir, HALL, Johan a LAVESSON, Niklas. DIDA: The largest historical handwritten digit dataset with 250k digits [online]. 2021 [cit. 2022-05-17]. Dostupné z : <https://github.com/didadataset/DIDA/>.
37. LECUN, Yann, BOTTOU, Léon, BENGIO, Yoshua a HAFFNER, Patrick. Gradient-Based Learning Applied to Document Recognition. 1998. Dostupné tiež z: <http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf>.
38. COHEN, Gregory, AFSHAR, Saeed, TAPSON, Jonathan a SCHAIK, André van. EMNIST: an extension of MNIST to handwritten letters. 2017. Dostupné tiež z: <https://arxiv.org/pdf/1702.05373.pdf>.
39. TACTILE SRL Brescia, Italy. Semeion Handwritten Digit Data Set. 1994.
40. HULL, J.J. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1994, roč. 16, č. 5, s. 550–554. Dostupné z DOI: 10.1109/34.291440.
41. ABDULLA, Waleed. Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow [online]. 2017 [cit. 2022-05-18]. Dostupné z : https://github.com/matterport/Mask_RCNN.
42. ANTAL, Eugen a MARÁK, Pavol. Historical handwritten document processing using modern methods. In: *Proceedings of the 22nd Central European Conference on Cryptography*. CECC, 2022.
43. TAN, Haobin. Annotation Conversion: COCO JSON to YOLO Txt [online]. 2022 [cit. 2022-05-20]. Dostupné z : <https://haobin-tan.netlify.app/ai/computer-vision/object-detection/coco-json-to-yolo-txt/#reference>.

Prílohy

A	Štruktúra elektronickej prílohy s praktickým výstupom	II
B	Štruktúra COCO JSON anotácií	IV
C	Mask R-CNN - konfigurácia trénovania a inferencie na datasete ARDIS	V
D	Mask R-CNN - konfigurácia trénovania a inferencie na datasete VEGA	VI
E	Mask R-CNN - konfigurácia trénovania a inferencie na datasete VEGA-Cropped	VII
F	Metriky na vyhodnotenie úspešnosti detekčných systémov	VIII
G	Vyhodnotenie úspešnosti detekčných systémov	IX

A Štruktúra elektronickej prílohy s praktickým výstupom

/implementation

/datasets_analysis

/ardis_ds.ipynb

/chars74k_ds.ipynb

/dida_ds.ipynb

/emnist_ds.ipynb

/mnist_ds.ipynb

/semeion_ds.ipynb

/usps_ds.ipynb

/image_preprocessing_app

/image_preprocessing_app.py

/model.py

/view.py

/image_recognition

/datasets_formats

/jupyter_lab_images

/coco_img1.png

/coco_img2.png

/yolo_img1.png

/coco_ds_format.ipynb

/coco_to_yolo.py

/datasets_format_converter.py

/yolo_ds_format.ipynb

/mask_rcnn_ardis.ipynb

/mask_rcnn_vega_cropped.ipynb

/mask_rcnn_vega.ipynb

/yolo_v5_rcnn_ardis.ipynb

/yolo_v5_vega_cropped.ipynb

/yolo_v5_vega.ipynb

B Štruktúra COCO JSON anotácií

```
1     {info: {
2         "year" : int,
3         "version" : str,
4         "description" : str,
5         "contributor" : str,
6         "url" : str,
7         "date_created" : datetime
8     },
9     licenses: [{
10        "id" : int,
11        "name" : str,
12        "url" : str
13    ],
14    categories: [{
15        "id" : int,
16        "name" : str,
17        "supercategory" : str
18    ],
19    images: [{
20        "id" : int,
21        "width" : int,
22        "height" : int,
23        "file_name" : str,
24        "license" : int,
25        "flickr_url" : str,
26        "coco_url" : str,
27        "date_captured" : datetime
28    ],
29    "annotations": [
30        "id" : int,
31        "image_id" : int,
32        "category_id" : int,
33        "segmentation" : RLE or [polygon],
34        "area" : float,
35        "bbox" : [x,y,width,height],
36        "iscrowd" : 0 or 1
37    ]}
```

C Mask R-CNN - konfigurácia trénovania a inferencie na datasete ARDIS

```
1 class TrainDigitsConfig(Config):
2     NAME = "HandwrittenDigits"
3
4     GPU_COUNT = 1
5     IMAGES_PER_GPU = 8
6
7     NUM_CLASSES = 10 + 1
8     STEPS_PER_EPOCH = 800
9     VALIDATION_STEPS = 50
10    LEARNING_RATE = 0.0001
11
12    BACKBONE = "resnet50"
13    RPN_ANCHOR_SCALES = (8, 16, 32, 64, 128)
14    POST_NMS_ROIS_INFERENCE = 500
15    POST_NMS_ROIS_TRAINING = 1000
16
17    IMAGE_RESIZE_MODE = "square"
18    IMAGE_MIN_DIM = 128
19    IMAGE_MAX_DIM = 128
20    IMAGE_CHANNEL_COUNT = 3
21    MEAN_PIXEL = np.array([143.62647976011993, 143.62647976011993, 143.62647976011993])
```



```
1 class InferenceDigitsConfig(TrainDigitsConfig):
2     GPU_COUNT = 1
3     IMAGES_PER_GPU = 1
4
5     IMAGE_MIN_DIM = 128
6     IMAGE_MAX_DIM = 128
7
8     DETECTION_MIN_CONFIDENCE = 0.3
```

D Mask R-CNN - konfigurácia trénovania a inferencie na datasete VEGA

```
1 class TrainDigitsConfig(Config):
2     NAME = "HandwrittenDigitsVega"
3
4     GPU_COUNT = 1
5     IMAGES_PER_GPU = 1
6
7     NUM_CLASSES = 10 + 1
8     STEPS_PER_EPOCH = 1660
9     VALIDATION_STEPS = 50
10    LEARNING_RATE = 0.0001
11
12    BACKBONE = "resnet50"
13    RPN_ANCHOR_SCALES = (8, 16, 32, 64, 128)
14    POST_NMS_ROIS_INFERENCE = 500
15    POST_NMS_ROIS_TRAINING = 1000
16
17    IMAGE_RESIZE_MODE = "square"
18    IMAGE_MAX_DIM = 1024
19    IMAGE_CHANNEL_COUNT = 3
20
21 class InferenceDigitsConfig(TrainDigitsConfig):
22     GPU_COUNT = 1
23     IMAGES_PER_GPU = 1
24
25     IMAGE_MIN_DIM = 1024
26     IMAGE_MAX_DIM = 1024
27
28     DETECTION_MIN_CONFIDENCE = 0.3
```

E Mask R-CNN - konfigurácia trénovania a inferencie na datasete VEGA-Cropped

F Metriky na vyhodnotenie úspešnosti detekčných systémov

Presnosť detektie:

$$Precision = \frac{TP}{TP + FP}$$

kde:

- $Precision$ – presnosť detektie,
- TP – správne detekované objekty,
- FP – nesprávne detekované objekty.

Citlivosť detektie:

$$Recall = \frac{TP}{TP + FN}$$

kde:

- $Recall$ – citlivosť detektie,
- TP – správne detekované objekty,
- FN – nedetekované objekty.

F1 skóre:

$$F1 = 2 * \frac{(Recall * Precision)}{(Recall + Precision)}$$

kde:

- $F1$ – F1 skóre,
- $Precision$ – presnosť detektie,
- $Recall$ – citlivosť detektie.

Úspešnosť klasifikácie:

$$Accuracy = \frac{TP}{TP + FP}$$

kde:

- $Accuracy$ – úspešnosť klasifikácie,
- TP – správne klasifikované objekty,
- FP – nesprávne klasifikované objekty.

G Vyhodnotenie úspešnosti detekčných systémov

	Presnosť detektie	Citlivosť detektie	F1 skóre detektie	Úspešnosť klasifikácie
Mask R-CNN				
ARDIS - 1x1	98%	100%	99%	94,86%
ARDIS - 2x2	0%	0%	0%	0%
ARDIS - 4x4	0%	0%	0%	0%
ARDIS - 8x8	0%	0%	0%	0%
VEGA - celý dokument	0%	0%	0%	0%
VEGA - výsek dokumentu	0%	0%	0%	0%
VEGA-Cropped - celý dokument	0%	0%	0%	0%
VEGA-Cropped - výsek dokumentu	~92,19%	~90,08%	~91,12%	~96,19%
YOLO v5				
ARDIS - 2x2	0%	0%	0%	0%
ARDIS - 4x4	21,03%	26,97%	23,63%	97,56%
ARDIS - 8x8	99,69%	99,84%	99,76%	99,53%
ARDIS - 10x10	99,53%	99,76%	99,45%	99,37%
ARDIS - 16x16	95,76%	61,72%	75,06%	71,52%
ARDIS - 18x18	68,97%	18,52%	29,19%	41,67%
VEGA - celý dokument	~83,87%	~3,83%	~7,3%	~89,52%
VEGA - výsek dokumentu	0%	0%	0%	0%
VEGA-Cropped - celý dokument	~90,96%	~82,79%	~86,69%	~96,24%
VEGA-Cropped - výsek dokumentu	0%	0%	0%	0%