VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

IDENTIFIKACE ZNAKŮ NA HLOUBKOVÉM SNÍMKU PNEUMATIKY

IDENTIFICATION OF CHARACTERS ON DEPTH SCANS OF TIRES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **PAVOL TOTH VAŇO**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. MICHAL ŠPANĚL, Ph.D.**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Toth Vaňo Pavol**
Program: Informační technologie
Název: **Identifikace znaků na hloubkovém snímku pneumatiky**
Identification of Characters on Depth Scans of Tires
Kategorie: Počítačová grafika

Zadání:

1. Seznamte se s problematikou hlubokých neuronových sítí a jejich učení. Zorientujte se v možnostech analýzy 3D informace (hloubkových snímků, mračen bodů, apod.) s využitím neuronových sítí.
2. Analyzujte současný stav v oblasti detekce a rozpoznávání jednotlivých textových znaků v obraze s využitím hlubokých neuronových sítí i konvenčních přístupů.
3. Na základě získaných poznatků vyberte vhodné metody a navrhněte postup pro detekci a identifikaci písmen a číslic na hloubkovém snímku povrchu pláště pneumatiky.
4. Navržený postup implementujte v experimentálním prototypu.
5. Proveďte experimenty a diskutujte dosažené výsledky na připravené datové sadě reálných snímků pláště pneumatik. Diskutujte možnosti budoucího vývoje.
6. Vytvořte stručný plakát nebo video prezentující vaši práci, její cíle a výsledky.

Literatura:

- K. He, G. Gkioxari, P. Dollár and R. Girshick, "Mask R-CNN" in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, 2020 (https://ieeexplore.ieee.org/document/8372616).
- Dále dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění prvních tří bodů zadaní.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Španěl Michal, Ing., Ph.D.**
Konzultant: Rástočný Karol, Ing., Ph.D., ME-Inspection SK
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 2. listopadu 2021

Abstrakt

Táto práca sa zaoberá problémom detekcie, rozpoznania a segmentácie znakov na hĺbkovej snímke pneumatiky. V práci aplikovaný prístup horizontálne rozdelí vstupnú hĺbkovú snímku na prekrývajúce sa časti, v ktorých sú symboly detegované hlbokou neurónovou sieťou Mask R-CNN. Následným použitím algoritmu potlačenia nemaximálnych hodnôt sú zahodené duplicitné detekcie vznikajúce kvôli prekryvom. V práci je tiež predstavená modifikácia siete Mask R-CNN používajúca zdvojenú segmentačnú vetvu za účelom kvalitnejšej segmentácie symbolov s tenkými líniami a členitou maskou. Použitím navrhnutého prístupu dosiahla na pripravenej dátovej sade metrika mean average precision s IoU v intervale od 0,5 do 0,95 hodnotu 0,877 pre detekciu a 0,738 pre segmentáciu.

Abstract

This thesis deals with the problem of detection, recognition and segmentation of characters on the depth scan of tire. The approach applied in the thesis horizontally splits the input depth scan into overlapping parts, in which the symbols are detected by the deep neural network Mask R-CNN. The duplicate detections emerging due to the overlap are discarded by the subsequent use of non-maximum suppression. The modification of Mask R-CNN, which utilises parallel segmentation branch with the aim of improving the quality of segmentation of symbols with thin lines or complex mask, is also proposed in the thesis. Applying the proposed approach on the prepared dataset, the values 0.877 and 0.738 were obtained as the mean average precision metrics for detection and segmentation in the IoU interval from 0.5 to 0.95.

Kľúčové slová

hĺbková snímka pneumatiky, hlboké učenie, Mask R-CNN, U-Net, segmentácia obrazu, identifikácia znakov, inštančná segmentácia, sémantická segmentácia, nerovnovážna dátová sada, zdvojená segmentačná vrstva

Keywords

depth scan of tire, deep learning, Mask R-CNN, U-Net, image segmentation, identification of characters, instance segmentation, semantic segmentation, imbalanced dataset, parallel segmentation layer

Citácia

TOTH VAŇO, Pavol. *Identifikace znaků na hloubkovém snímku pneumatiky.* Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Michal Španěl, Ph.D.

Identifikace znaků na hloubkovém snímku pneumatiky

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením Ing. Michala Španěla, Ph.D.. Ďalšie informácie mi poskytli zamestnanci firmy ME-Inspection SK, spol. s r.o., najmä Ing. Karol Rástočný, Ph.D. a RNDr. Andrej Lúčny, Ph.D.. Uviedol som všetky literárne pramene, publikácie a dalšie zdroje, z ktorých som čerpal.

.....
Pavol Toth Vaňo
11. mája 2022

Poďakovanie

Rád by som poďakoval svojmu vedúcemu bakalárskej práce Ing. Michalovi Španělovi, Ph.D. za ochotu a cenné rady poskytnuté pri vypracovaní práce. Ďalej by som chcel poďakovať Ing. Karolovi Rástočnému, Ph.D. a RNDr. Andrejovi Lúčnemu, Ph.D. za poskytnutie dátovej sady a odborné rady.

Obsah

Kapitola 1

Úvod

Znaky vyrezané alebo vylisované na bočnici pneumatiky obsahujú informácie nevyhnutné pre správne a bezpečné použitie pneumatiky, preto ich čitateľnosť a kvalita patrí medzi dôležité vlastnosti kontrolované pri výrobe pneumatiky. Manuálna kontrola znakov je však časovo náročná a svoju rolu pri nej zohráva aj ľudský faktor. Z tohto dôvodu je veľmi žiadúce nahradiť ju automatizovanými metódami pracujúcimi nad hĺbkovými snímkami.

Táto práca bola vypísaná v spolupráci s firmou ME-Inspection SK, spol. s r.o., ktorá sa zaoberá vývojom a výrobou meracích a inšpekčných systémov najmä v gumárenskom a automobilovom priemysle. Firma pre svojich zákazníkov z radov výrobcov pneumatík poskytuje systém kontroly ich kvality, ktorého súčasťou je aj kontrola kvality symbolov na plášti. Prístup využívajúci klasické metódy počítačového videnia, ktorý je na identifikáciu symbolov v súčasnej dobe použitý vo firme, vyžaduje presnú definíciu a usporiadanie všetkých symbolov na pneumatike.

Hlavným cieľom práce je na základe teoretických poznatkov navrhnúť a v experimentálnom prototype implementovať metódu, ktorej výstupom bude informácia o pozícií, maske a type všetkých symbolov na vstupnej hĺbkovej snímke pneumatiky.

V rámci práce bola vytvorená dátová sada, pričom hĺbkové snímky s neúplnou anotáciou boli poskytnuté firmou. Navrhnuté riešenie problému identifikácie symbolov pozostáva z horizontálneho rozdelenia vstupnej hĺbkovej snímky na prekrývajúce sa podsnímky, z použitia neurónovej siete Mask R-CNN na detekciu symbolov v týchto podsnímkach a z následného potlačenia nemaximálnych hodnôt, ktoré zahodí duplicitné detekcie vznikajúce z dôvodu prekryvov medzi podsnímkami. Navrhnutá metóda bola implementovaná a podrobená experimentom.

Text technickej správy je štrukturovaný do 7 kapitol. V kapitole 2 sú stručne zhrnuté existujúce prístupy k detekcií a rozpoznaniu textových symbolov v obraze, podrobne je opísaný algoritmus aplikovaný vo firme ME-Inspection SK, spol. s r.o. na rozpoznanie znakov na pneumatike. Kapitola 3 sa venuje vybraným metódam segmentácie obrazu s použitím neurónových sietí, najmä architektúram Mask R-CNN a U-Net. V kapitole 4 je predstavená navrhnutá metóda na riešenie problému identifikácie symbolov na hĺbkovej snímke pneumatiky, vrátane jej modifikácie pre lepšiu segmentáciu symbolov s tenkými líniami alebo členitou maskou. Táto kapitola tiež popisuje vlastnosti pripravenej dátovej sady. Popis implementácie navrhnutej metódy a zoznam použitých nástrojov sú súčasťou kapitoly 5. Kapitola 6 obsahuje popis prevedených experimentov a ich zhodnotenie. V kapitole 7 sa nachádza zhrnutie práce a dosiahnutých výsledkov nasledované úvahou o možnostiach pokračovania práce.

Kapitola 2

Detekcia a rozpoznávanie textových znakov na snímke pneumatiky

Táto kapitola začína presnou formuláciou riešeného problému identifikácie znakov na hĺbkovej snímke pneumatiky. Ďalej obsahuje krátky prehľad existujúcich prístupov k detekcií a rozpoznávaniu znakov v obraze. Prístup, ktorý je k riešeniu problému využitý vo firme ME-Inspection SK, spol. s r.o., je popísaný podrobne.

2.1 Formulácia problému identifikácie znakov na hĺbkovej snímke pneumatiky

Znaky a symboly vyrezané na bočnici pneumatiky obsahujú dôležité informácie nevyhnutné k správnemu použitiu a identifikácií pneumatiky. Medzi informácie potrebné k použitiu patria napríklad [46]:

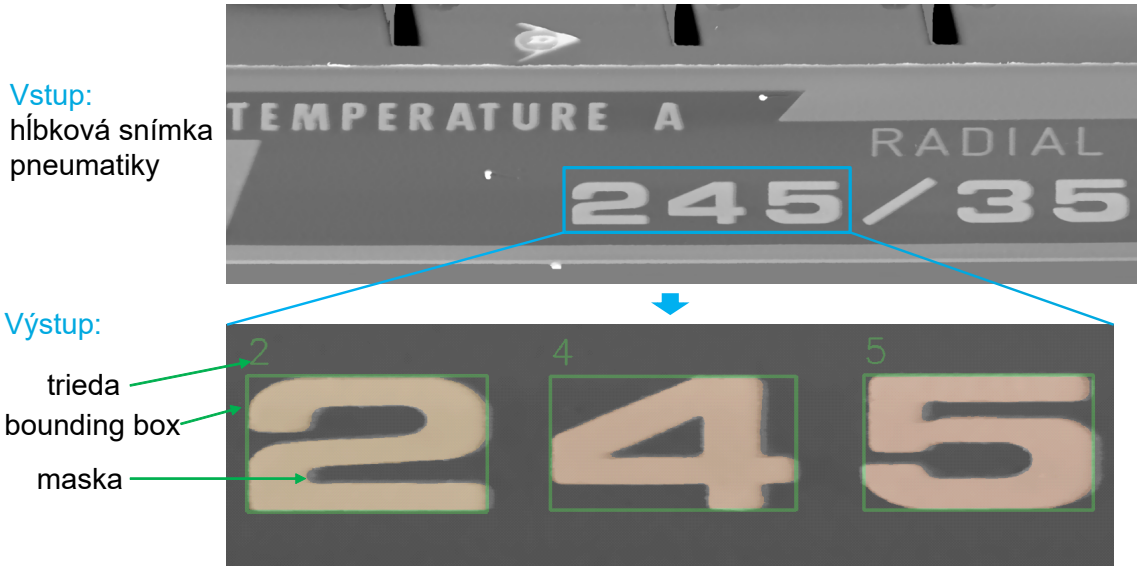
- Informácia o ročnom období a typoch vozovky, na ktoré je pneumatika určená.
- Informácia o veľkosti a ďalších fyzikálnych vlastnostiach pneumatiky.
- Informácia o roku a týždni výroby pneumatiky.
- Bezpečnostné varovania pre prácu s pneumatikou.

Ďalej sa na pneumatike nachádzajú nápisy identifikujúce výrobcu, sériu a aj konkrétnu pneumatiku.

Kvôli dôležitosti týchto informácií výrobcovia pneumatík kladú požiadavky na minimálnu kvalitu jednotlivých znakov. Požadovaná kvalita znakov sa medzi výrobcami aj medzi typmi znakov v rámci jedného výrobcu líši, pre výpočet kvality je však okrem detekcie a rozpoznania znaku nutná aj jeho maska. Preto problém identifikácie znaku riešený v tejto práci formulujeme ako problém detekcie, rozpoznania a segmentácie znakov na pneumatike.

Detekcia znakov na pneumatike je ťažkou úlohou najmä kvôli nízkemu kontrastu, keďže čierne znaky sa nachádzajú na čiernom pozadí, čo môže byť problematické aj pre ľudské oko [15]. Preto je výhodnejšie pracovať so šedotónovou hĺbkovou snímkou pneumatiky, kde

hodnota pixelu reprezentuje hĺbku povrchu pneumatiky v danom bode. Očakávaným výstupom identifikácie je potom maska, trieda a bounding box každého symbolu. Ukážka časti hĺbkovej snímky, ako aj očakávaného výstupu je na obrázku 2.1.



Obr. 2.1: Ukážka časti vstupnej hĺbkovej snímka a očakávaného výstupu. Hĺbkové snímky majú na šírku 65 536 pixelov, tu je zobrazená len časť o šírke 5 500 pixelov.

2.2 Súčasný stav v oblasti detekcie a rozpoznávania textových znakov v obraze

Detekcia a rozpoznávanie textu v obraze patria medzi tradičné problémy počítačového videnia. Kým výstupom detekcie je pozícia textu vo vstupnom obraze, výstupom rozpoznávania je reťazcová reprezentácia textu v tvare spracovateľnom počítačom [20].

Rozpoznávanie textu v obraze môže používať klasické metódy počítačového videnia alebo metódy hlbokého učenia. Na rozpoznávanie textových znakov v naskenovaných dokumentoch sa často používa OCR (optical character recognition, pozri napríklad [27]), ktoré môže byť založené na oboch týchto prístupoch.

Táto sekcia bola vypracovaná na základe [20].

2.2.1 Klasické metódy

Klasické metódy detekcie textu v obraze sa delia na 2 základné prístupy:

1. Prístup s použitím posuvného okna (angl. sliding window). Posuvné okno rôznych rozmerov je postupne prikladané na všetky pozície obrazu. Na obsah posuvného okna je aplikovaný natrénovaný klasifikátor, ktorého výstupom je informácia o tom, či sa v okne nachádza text. Tento prístup môže byť použitý na detekciu jednotlivých znakov, celých slov alebo riadkov.

2. Prístup s nájdením spojených komponentov (angl. connected components). Prístup najprv spojí pixely s určitými vlastnosťami (napr. farba, šírka ťahu písma) do kandidátnych komponentov, z ktorých sú následne odfiltrované netextové komponenty.

V porovnaní s prístupom posuvného okna je efektívnejší a robustnejší. Príkladom prístupu je metóda maximálne stabilných extrémnych regiónov (MSER) [13].

Klasifikácia textu klasickými metódami zvyčajne porovnáva ručne vytvorené príznaky alebo deskriptory. V súčasnosti sú však v oblasti detekcie aj klasifikácie textu klasické metódy nahrádzané metódami využívajúcimi hlboké učenie.

2.2.2 Metódy s použitím hlbokého učenia

Existujú 2 základné prístupy k detekcií textu s použitím hlbokého učenia:

1. Prístup s využitím sémantickej segmentácie. S využitím plne konvolučnej siete je vypočítaná binárna segmentačná mapa, ktorej následným spracovaním sú získané bounding boxy obsahujúce text.

2. Prístup na základe obecnej detekcie objektov. Často je použitá modifikovaná verzia detektorov objektov, ako sú SSD [22], YOLO [35] alebo Faster R-CNN [36].

Pri klasifikácií môže byť použitá jedna z troch úrovní – rozpoznávanie jednotlivých znakov, rozpoznávanie slov alebo rozpoznávanie sekvencií.

2.3 Súčasné riešenie vo firme

Táto sekcia popisuje, akým spôsobom vo firme ME-INSPECTION SK, spol. s.r.o. v súčasnosti prebieha detekcia symbolov na hĺbkovej snímke pneumatiky. Spracovanie pozostáva z nájdenia kontúr, výpočtu rozšíreného TAR deskriptora pre všetky nájdené objekty, porovnania deskriptora s deskriptormi všetkých objektov v existujúcich šablónach a nájdenia najlepšej zhody s objektom v šablónach. Za triedu symbolu je potom považovaná trieda objektu šablóny, ktorý má s nájdeným symbolom najlepšiu zhodu.

Celý tento proces bol publikovaný v [24]. Z tohto článku vychádza aj jeho popis v nasledujúcich podsekciách.

2.3.1 Nájdenie kontúr

Tradičný algoritmus na nájdenie kontúr vyžaduje na vstupe binárny obrázok. Na binarizáciu hĺbkového obrázka, kde hodnota pixelu v každom bode je hĺbka v danom bode, je použitý Retinex filter [32]. Filter pre obrázok vráti logaritmus pomeru hodnoty pixelu k priemernej hodnote okolia, ktorá je vypočítaná konvolúciou s Gaussovým filtrom. Výstup obrázka I po spracovaní Retinex filtrom R je potom:

$$R(I) = log\frac{I}{G * I} = logI - log(G * I), \tag{2.1}$$

kde G je 2D Gaussov filter a * je operácia konvolúcie.

Výstupom Retinex filtra ešte nie je binárny obrázok, ale väčšina pixelov takto filtrovaného obrázka má veľmi nízku alebo vysokú hodnotu. Samotná binarizácia je v experimentoch popísaných v článku prevedená pomocou iteračného algoritmu IsoData [37]. Binárny obrázok je použitý na nájdenie kontúr.

2.3.2 Výpočet TAR deskriptora

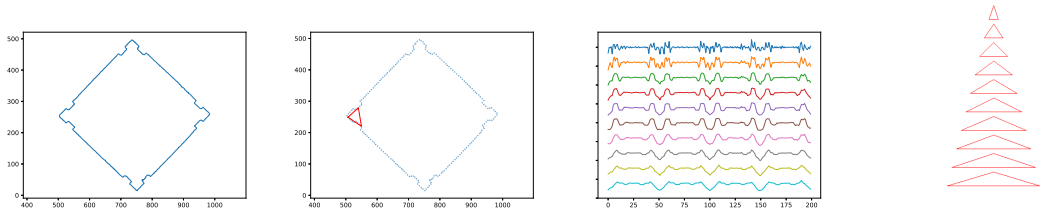
Na extrahovanie vlastností a porovnanie objektov sú často používané deskriptory tvaru. Deskriptory tvaru sa delia na deskriptory na báze kontúr (contour based descriptors) a deskriptory na báze regiónov (region based descriptors) [14]. V článku je použitý deskriptor TAR (Triangle area representation) [2], ktorý sa radí medzi deskriptory na báze kontúr.

TAR deskriptor je vypočítaný z plochy trojuholníkov tvorených bodmi kontúry. Kvôli rýchlosti spracovania sú body kontúry ekvidištančne vzorkované na fixný počet N bodov, kde N je 100 alebo 200. Označme navzorkované body kontúry $P_1, P_2, \ldots, P_N = (x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$. Plocha trojuholníka je vypočítaná pre tri postupne idúce body kontúry $P_{(n-t_s)\%N}$, P_n a $P_{(n+t_s)\%N}$:

$$TAR(n, t_s) = \frac{1}{2} \begin{vmatrix} x_{(n-t_s)\%N} & y_{(n-t_s)\%N} & 1 \\ x_n & y_n & 1 \\ x_{(n+t_s)\%N} & y_{(n+t_s)\%N} & 1 \end{vmatrix}, \tag{2.2}$$

kde $n \in [1, N]$ je index bodu kontúry a $t_s \in [1, T_s]$ je rozdiel v poradí medzi bodmi trojuholníka [2]. Typická hodnota T_s je 10.

$TAR(n, t)$ je až na znamienko totožné s plochou vymedzeného trojuholníka, znamienko závisí na reflexivite uhla pri vrchole P_n. Výpočet deskriptora TAR pre kontúru je znázornený na obrázku 2.2.



(a) **(b)** **(c)** **(d)**

Obr. 2.2: Demonštrácia výpočtu deskriptora TAR. **(a)** Nájdená kontúra; **(b)** ekvidištančne navzorkovaných 200 bodov kontúry a znázornenie trojuholníka, z ktorého je vypočítaná jedna hodnota deskriptora; **(c)** jedna krivka zobrazuje hodnoty $TAR(n, t_s)$ pre rôzne n, ale rovnaký krok t_s; **(d)** znázornenie veľkosti kroku, resp. strán trojuholníka. Prevzaté z [24].

Keďže $TAR(n, t)$ je počítaný pre všetky kombinácie hodnôt indexu n a kroku t_s, výsledkom je matica rozmeru N x T_s. Hodnoty matice normalizujeme – vydelíme ich sumou absolútnych hodnôt všetkých plôch trojuholníkov, čím sa zaručene dostanú do intervalu $[-1, 1]$.

2.3.3 Porovnanie deskriptorov

Pre porovnanie deskriptorov je použitý algoritmus Dynamic Space Warping [2] (ďalej DSW), ktorý spočíta vzdialenosť deskriptorov aj vzájomné párovanie bodov kontúr.

Nech $TAR_A(n, t_s)$ a $TAR_B(n, t_s)$ sú TAR deskriptory pre tvary A a B. Potom je vzdialenosť medzi bodom kontúry A s indexom p a bodom kontúry B s indexom q definovaná nasledovne [2]:

$$D(p, q) = \frac{1}{T_s} \sum_{t_s=1}^{T_s} |TAR_A(p, t_s) - TAR_B(q, t_s)|. \tag{2.3}$$

Algoritmus DSW využíva dynamické programovanie. Postupne počíta tabuľku $DT_{i,j}$, ktorej hodnoty predstavujú minimálnu vzdialenosť medzi prvým deskriptorom od prvého po i-tý bod a druhým deskriptorom od prvého po j-tý bod. Prvá hodnota matice je inicializovaná:

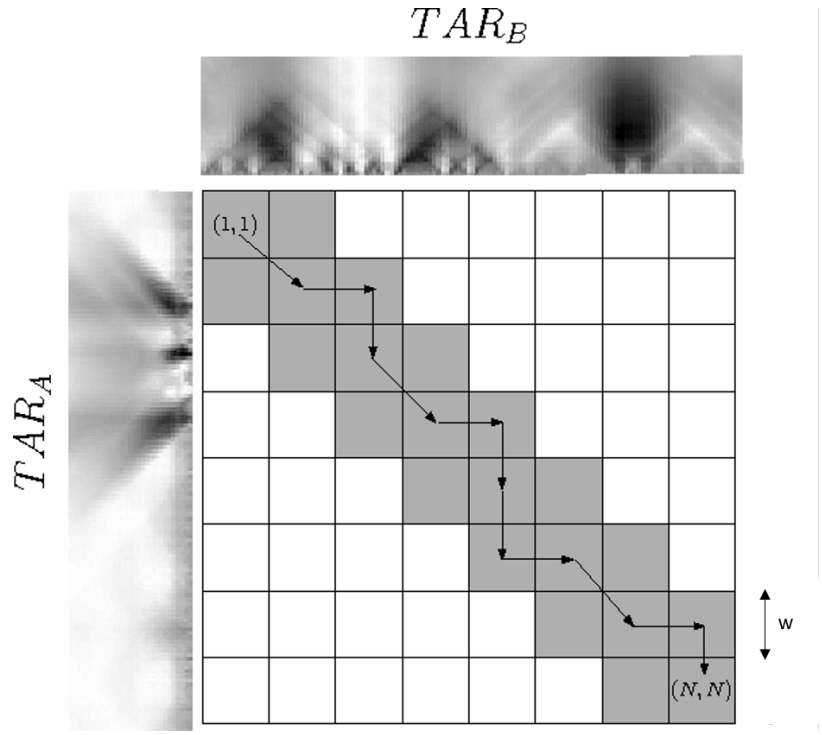
$$DT_{1,1} = D(1,1). \tag{2.4}$$

Ďalšie hodnoty sú potom vypočítané:

$$DT_{i,j} = min(DT_{i-1,j-1}, DT_{i-1,j}, DT_{i,j-1}) + D(i,j). \tag{2.5}$$

Aby bol výpočet vzdialenosti obmedzený len na tie body, ktoré si s väčšou pravdepodobnosťou korešpondujú, bol zavedený parameter w, ktorý určuje maximálnu vzdialenosť od diagonály matice DT. Hodnota matice $DT_{i,j}$ je počítaná len pre $i - w \leq j \leq i + w$, ostatné hodnoty matice sú nastavené na nekonečno.

Po vyplnení matice hodnota $DT_{N,N}$ reprezentuje vzdialenosť deskriptorov. Na základe cesty v matici k $DT_{N,N}$ je možné získať vzájomné párovanie bodov kontúr. Matica DT je znázornená na obrázku 2.3.



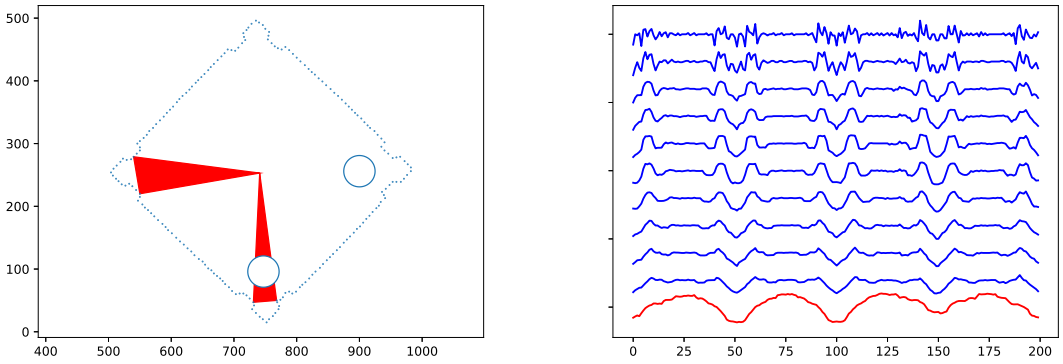
Obr. 2.3: Matica DT. Cesta od $DT_{1,1}$ ku $DT_{N,N}$ udáva párovanie bodov kontúr. Prevzaté a upravené z [2].

Je dôležité si povšimnúť, že algoritmus používajúci dynamické programovanie umožňuje, aby bol jeden bod kontúry namapovaný na viacero bodov druhej kontúry. Z toho vyplýva, že dvojíc korešpondujúcich bodov, ktoré algoritmus vyprodukuje, nemusí byť presne N.

Pri porovnávaní objektov na základe ich TAR deskriptorov je treba vziať do úvahy, že poradie bodov kontúry dvoch objektov môže byť navzájom posunuté. Preto je potrebné volať algoritmus DSW N-krát s tým, že jeden z porovnávaných deskriptorov je postupne posunutý o 0, 1, . . . , N-1 bodov.

Pri objektoch s asymetrickou kontúrou by bolo postačujúce vybrať párovanie dané algoritmom DSW, pre ktoré bola hodnota TAR vzdialenosti minimálna. Pri objektoch so symetrickou kontúrou môže cyklický algoritmus DSW vrátiť veľmi podobné hodnoty vzdialenosti pre viacero párovaní. Ak je aj vnútorná časť objektu symetrická, za správne je možné považovať ktorékoľvek z týchto párovaní. Avšak pri objektoch so symetrickou kontúrou a asymetrickým vnútrom je správne len jedno párovanie, ale TAR, ktorý patrí medzi deskriptory na báze tvaru, vnútornú časť objektu neberie do úvahy a preto nedokáže správne párovanie rozlíšiť.

Riešením tohto problému je v článku predstavené rozšírenie algoritmu TAR, ktoré popisuje vnútro objektu. Kým $TAR(n, t_s)$ počíta plochu trojuholníka tvoreného bodmi P_{n-t_s}, P_n, P_{n+t_s}, pri rozšírení $TAR_{extension}(n, t_s)$ je skúmaný trojuholník tvorený bodmi P_{n-t_s}, $center$, P_{n+t_s}, kde $center$ je ťažisko kontúry. Znázornenie rozšíreného deskriptora TAR je na obrázku 2.4. Trojuholník je samozrejme maskovaný s pixelmi, ktoré po binarizácii prislúchali objektu[1]. Plocha maskovaného trojuholníka je následne normovaná do intervalu $[0, 1]$. Rozšírený deskriptor TAR vznikne pridaním $TAR_{extension}(n, T_s)$ pre $n \in [1, N]$ k deskriptoru TAR, teda použitím najväčšej hodnoty kroku T_s.



(a) **(b)**

Obr. 2.4: Znázornenie rozšíreného deskriptora TAR. **(a)** Rozšírenie deskriptora TAR; **(b)** deskriptor TAR s rozšírením (červená krivka). Pre tento objekt s asymetrickým vnútrom obsahuje klasický TAR deskriptor 4-krát ten istý segment. Prevzaté z [24].

Kompletný pseudokód pre porovnanie tvarov sa nachádza v algoritme 1. Pri cyklickom volaní DSW pre všetky posuny sú spočítané vzialenosti, párovania a posuny pridávané do množiny *Results*. Množina *Minima* obsahuje tie prvky množiny *Results*, ktorých vzdialenosť má od minimálnej vzdialenosti odchýlku menšiu alebo rovnú ϵ. Na množinu *Minima* je aplikované potlačenie nemaximálnych hodnôt (angl. non-maximum suppression), ktoré eliminuje podobné posuny, a za najlepší je vybraný ten prvok množiny *Minima*, ktorý má najnižšiu hodnotu vzdialenosti rozšírených deskriptorov TAR (zoradených podľa vypočítaného párovania).

[1] Tu použitá binarizácia sa líši od binarizácie pomocou Retinex filtru použitej na získanie kontúr, pretože tá vracia fantómové diery vnútri objektu.

Algoritmus 1: Algoritmus na porovnanie tvarov (upravené z [24])

Vstup: Rozšírené TAR deskriptory r a s
Výstup: Vzdialenosť tvarov a párovanie bodov kontúry

function *COMPARE(r,s)*:
  $Results \leftarrow \{\}$
  for $shift \leftarrow 0$ to $N-1$ **do**
    $distance, Pairing \leftarrow DSW(r_{TAR}, shiftBy(s_{TAR}, shift)$
    Pridaj $(distance, Pairing, shift)$ do $Results$
  end for
  $distance_{min} \leftarrow min\{d|(d, P, shift) \in Results\}$
  $Minima \leftarrow \{(d, P, s)|(d, P, shift) \in Results \wedge d \leq distance_{min} + \epsilon\}$
  $Minima \leftarrow$NonMaximumSuppression$(Minima)$
  $best \leftarrow$argmin$\{||r_{extension} - s_{extension}|| \mid (d, P, shift) \in Minima\}$
  $distance, Pairing, shift \leftarrow Minima[best]$
  return $distance, shiftBy(Pairing, -shift)$

2.3.4 Zarovnanie

Získané párovanie medzi bodmi dvoch kontúr môže byť použité pre výpočet transformácie, ktorá zarovná oba objekty na pôvodnej hĺbkovej mape. V tomto prípade je k bodom pridaná informácia o hĺbke z pôvodnej hĺbkovej mapy, ide teda o transformáciu postupnosti 3D bodov jedného objektu na postupnosť 3D bodov druhého objektu. Na nájdenie správnej tranformácie je iteratívne použitý Kabschov algoritmus [12]. Pre podrobnosti k iteratívnemu použitiu Kabschovho algoritmu pozri [24].

2.3.5 Integrácia

Táto podsekcia popisuje, ako sú metódy opísané v predchádzajúcich podsekciách integrované pri rozpoznávaní symbolov.

Firemná databáza obsahuje šablóny so symbolmi, ktoré je nutné rozpoznávať. Vytváranie šablón prebieha podobne ako rozpoznávanie symbolu – operátor ručne vyberie jednu z kontúr nájdených v hĺbkovej mape, pre túto kontúru je vypočítaný rozšírený TAR deskriptor a tieto informácie sú vložené do databázy. Šablóna obsahuje:

- triedu symbolu,
- zoznam bodov kontúry symbolu vrátane hĺbky (3D),
- N indexov do zoznamu bodov, ktoré predstavujú vzorkovanie kontúry,
- N x T_s hodnôt TAR deskriptora z intervalu $[-1, 1]$,
- N hodnôt TAR rozšírenia z intervalu $[0, 1]$.

Pseudokód pre rozpoznanie symbolu sa nachádza v algoritme 2. Na vstupnej hĺbkovej mape pneumatiky sú nájdené kontúry. Každá nájdená kontúra je porovnaná s každou kontúrou zo šablón. V prípade, že je najmenšia vzdialenosť deskriptora nájdenej kontúry s deskriptorom kontúry zo šablóny menšia ako zadaný prah, kontúra bola detegovaná.

Algoritmus 2: Rozpoznanie symbolov (upravené z [24])

Vstup: Hĺbková mapa pneumatiky $depthMap$
Výstup: Zoznam rozpoznaných symbolov

function *RecognizeSymbols(depthMap)*:
  $Contours \leftarrow FindContours(depthMap)$
  $Symbols \leftarrow \{\}$
  for $Contour$ in $Contours$ **do**
    $Descriptor_{TAR}, Indices \leftarrow CalculateTar(Contour)$
    $Descriptor_{Extension} \leftarrow CalculateExtension(Contour, Indices)$
    $Distance_{min} \leftarrow \infty$
    for $Template$ in $LoadAllTemplates()$ **do**
      $Class^t, Contour^t, Indices^t, Descriptor^t_{TAR}, Descriptor^t_{Extension} \leftarrow Template$
      $Distance, Pairing \leftarrow COMPARE(Descriptor, Descriptor^t)$
      if $Distance < Distance_{min}$ **then**
        $Distance_{min} \leftarrow Distance$
        $Class_{min} \leftarrow Class^t$
      end if
    end for
    if $Distance < Threshold_{distance}$ **then**
      Pridaj $(Contour, Class_{min}, Distance_{min})$ do $Symbols$
    end if
  end for
  return $Symbols$

2.3.6 Zhodnotenie súčasného riešenia

V praxi použité riešenie sa v niektorých detailoch líši od riešenia prezentovaného v [24]. V praxi použitý algoritmus:

- vyžaduje presnú definíciu, aké symboly sú na pneumatike,
- vyžaduje usporiadanie symbolov na pneumatike,
- implementácia na stroji pracuje aj s odhadovanou pozíciou (nemusí byť, ale zvyšuje robustnosť),
- vyžaduje odhad veľkosti symbolu a povolené rozpätie škálovania (nemusí byť, ale zvyšuje robustnosť),
- je časovo náročný, pretože sa skúšajú všetky kombinácie možných umiestnení v rámci sekvencie a povolených editačných vzdialeností.

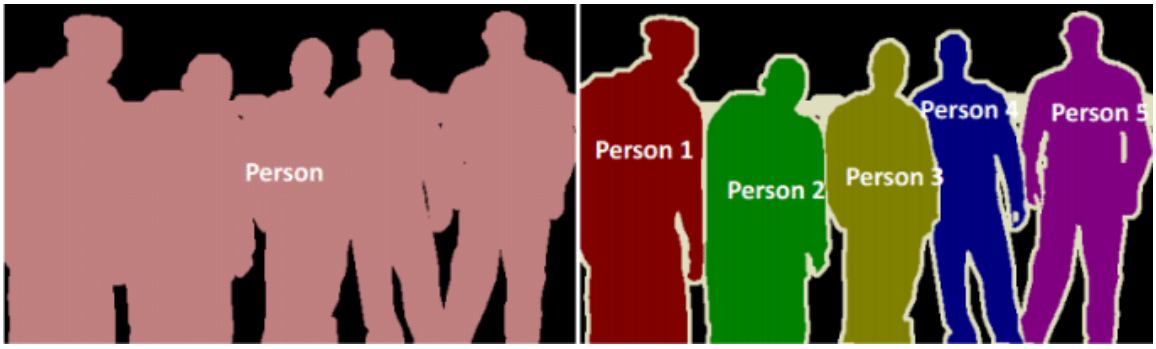
Výhodou použitého algoritmu je naopak robustnosť a jednoduchosť doplnenia nových symbolov (stačí ich maska).

Kapitola 3

Metódy segmentácie obrazu s využitím neurónových sietí

Segmentácia obrazu je proces rozdelenia obrazu na viacero segmentov alebo objektov [26]. Základnými druhmi segmentácie sú [26]:

1. Sémantická segmentácia – každému pixelu obrázka je priradená trieda. Delí sa na [9]:
   - Binárnu segmentáciu – pixelu obrázka je priradená jedna z dvoch tried. Jednou z tried býva typicky trieda pozadie.
   - Multiclass segmentáciu – jednému pixelu obrázka je priradená práve jedna trieda (z viacerých možných).
   - Multilabel segmentáciu – jednému pixelu obrázka môže byť priradených 0, 1 alebo aj viacero tried.
2. Inštančná segmentácia – v obrázku sú detegované objekty, pričom každému detegovanému objektu je priradená trieda a maska pixelov, ktoré daný objekt tvoria. Inštančná segmentácia teda narozdiel od sémantickej segmentácie dokáže od seba odlíšiť viacero inštancií objektov tej istej triedy, ako ukazuje obrázok 3.1.



Obr. 3.1: Demonštrácia rozdielu medzi (vľavo) sémantickou a (vpravo) inštančnou segmentáciou. Prevzaté z [47].

3.1 Architektúra Mask R-CNN

Architektúra Mask R-CNN [10] predstavuje v súčasnosti asi najpoužívanejšiu metódu v oblasti inštančnej segmentácie. Mask R-CNN je nadstavbou nad architektúrou Faster R-CNN, ktorá rieši problém detekcie objektov. Výstupom Mask R-CNN je navyše aj maska každého detegovaného objektu.

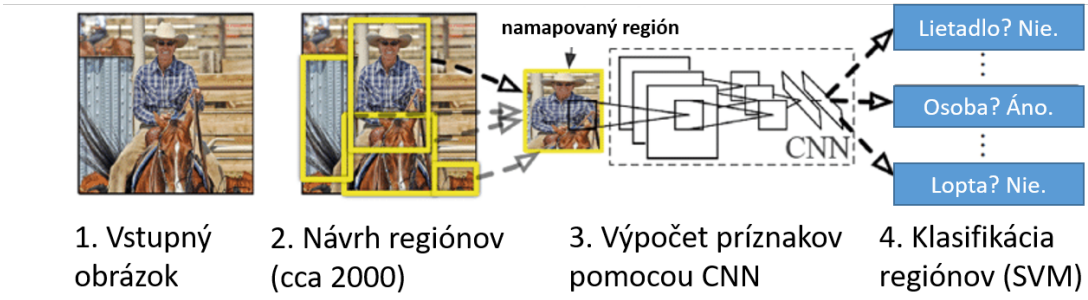
Systém Mask R-CNN vznikol postupným vývojom cez metódy R-CNN [8], Fast R-CNN [7] a Faster R-CNN [36]. Každá z týchto metód priniesla vylepšenia a črty badateľné v architektúre Mask R-CNN, preto nasledujúca pasáž obsahuje ich stručný opis vypracovaný na základe článkov, v ktorých boli predstavené.

3.1.1 R-CNN

Systém R-CNN (Regions with convolutional neural networks features), predstavený v roku 2013 [8], pozostáva z troch modulov, ako je vidieť na obrázku 3.2. Prvým z nich je modul na návrh regiónov (region proposal module). Systém R-CNN je nezávislý na konkrétnom algoritme pre návrh regiónov, v článku bol použitý algoritmus selective search [45].

Z hodnôt pixelov každého kandidátneho regiónu (angl. region of interest, ďalej RoI) je pomocou konvolučnej neurónovej siete AlexNet [17], ktorá tvorí druhý modul R-CNN, vyextrahovaný 4096-dimenzionálny vektor príznakov.

Tretí modul obsahuje lineárne stroje s podpornými vektormi (support vector machines, ďalej SVM, pozri napríklad [28]) pre každú triedu objektov, ktorú má systém detegovať. SVM natrénovaný pre konkrétnu triedu vypočíta pre vektor príznakov každého navrhnutého regiónu jeho skóre príslušnosti do danej triedy. Nakoniec nezávisle pre každú triedu prebehne potlačenie nemaximálnych hodnôt, ktoré odmietne navrhnutý región, ak má nižšie skóre ako iný región, s ktorým má IoU vyššie ako zadaný prah.



Obr. 3.2: Schéma systému R-CNN. Prevzaté a preložené z [8].

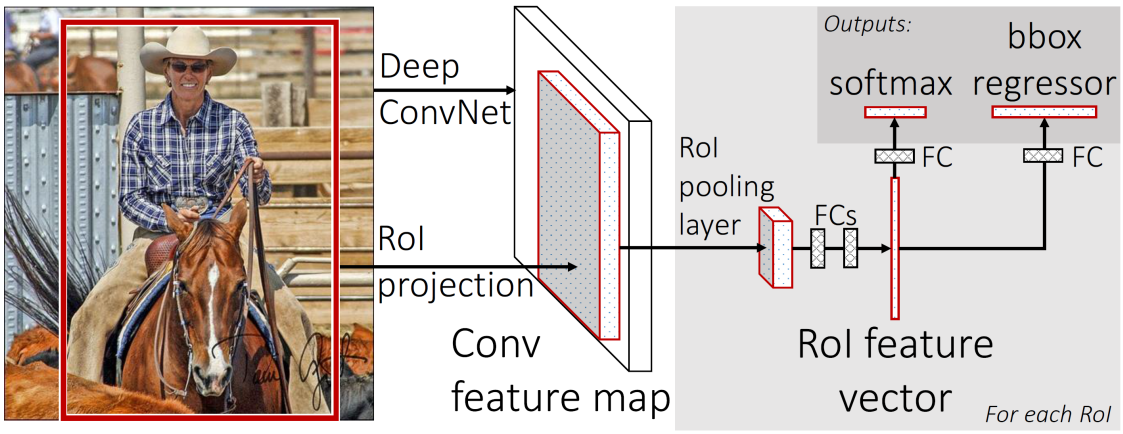
Voliteľnou štvrtou časťou systému je modul na regresiu bounding boxov, ktorý upresňuje bounding boxy vystupujúce z modulu na návrh regiónov.

Systém R-CNN sa netrénuje spolu, ale po častiach. V prvom kroku sa trénuje konvolučná sieť (extraktor príznakov) pre úlohu klasifikácie, teda so softmaxom ako poslednou vrstvou. Až po plnom natrénovaní extraktora príznakov je vrstva softmax odstránená a prebieha trénovanie lineárnych SVM pre jednotlivé triedy, ktorých vstupom sú príznaky vytvorené natrénovanou konvolučnou sieťou.

3.1.2 Fast R-CNN

V roku 2015 bola uvedená metóda Fast R-CNN [7], ktorá si kládla za cieľ vylepšiť R-CNN – zjednodušiť trénovanie, zmenšiť časové a priestorové nároky trénovania a zrýchliť samotnú detekciu objektov pri inferencii.

Vstupom Fast R-CNN je obrázok a množina navrhnutých regiónov. Obrázok je najprv spracovaný niekoľkými konvolučnými a max-pooling vrstvami, ktorých výstupom je mapa príznakov (celého obrázka). Následne vrstva RoIPool pre každý navrhnutý región vyextrahuje z mapy príznakov vektor príznakov pevnej dĺžky. Vektor príznakov je vstupom niekoľkých plne prepojených vrstiev, ktoré sa vetvia do dvoch vetiev: jedna vetva je zakončená vrstvou softmax, ktorá vracia pravdepodobnosť, že región obsahuje objekt danej triedy (triedou je aj pozadie), druhá vetva vracia pre každú triedu upresnenú pozíciu bounding boxu. Systém Fast R-CNN je znázornený na obrázku 3.3.



Obr. 3.3: Schéma systému Fast R-CNN. Prevzaté z [7].

Kľúčovú úlohu zohráva vrstva RoIPool, ktorá pre všetky regióny, ktoré sú obecne rôznych rozmerov, vyextrahuje z mapy príznakov vektor rovnakej dĺžky. Vrstva rozdelí navrhnutý región, ktorý má po namapovaní do mapy príznakov rozmery h x w, do H x W okien s približnou veľkosťou h/H x w/W. V každom z týchto okien sa vyberie maximálna hodnota (prebieha max-pooling) a tým vznikne menšia mapa príznakov fixných rozmerov H x W.

Takáto architektúra siete umožňuje trénovať celú sieť naraz. Zároveň vstupný obrázok prechádza konvolučnou časťou siete celý, čo urýchľuje trénovanie aj inferenciu.

Chybová funkcia

Chybová funkcia Fast R-CNN, ktorá je aj súčasťou chybovej funkcie Faster R-CNN a Mask R-CNN, je definovaná nasledovne:

$$L(p, u, t^u, v) = L_{cls}(p, u) + \lambda[u \geq 1]L_{loc}(t^u, v), \tag{3.1}$$

kde $u \in 0, 1, \ldots, c$ je ground-truth trieda objektu, $p = (p_0, p_1, ..., p_c)$ je sieťou vypočítaná distribúcia pravdepodobností pre $c + 1$ tried, $t^u = (t^u_x, t^u_y, t^u_w, t^u_h)$ sú ofsety vypočítané pri regresii bounding boxu pre triedu u, a v sú ground-truth ofsety pre regresiu bounding boxu. Použitá parametrizácia pre ofsety spresňujúce bounding boxy je opísaná v [8].

Chybová funkcia teda vznikne ako súčet chyby klasifikácie L_{cls} a chyby regresie bounding boxu L_{loc}. Chyba klasifikácie je daná predpisom

$$L_{cls}(p, u) = -logp_u. \tag{3.2}$$

Chyba regresie bounding boxu má predpis

$$L_{loc}(t^u, v) = \sum_{i \in \{x,y,w,h\}} L_1^{smooth}(t_i^u - v_i), \tag{3.3}$$

kde

$$L_1^{smooth}(x) = \begin{cases} 0.5x^2 & \text{ak } |x| < 1 \\ |x| - 0.5 & \text{inák} \end{cases} \tag{3.4}$$

je L_1 chyba menej citlivá voči odľahlým hodnotám.

Chyba regresie bounding boxu sa do celkovej chyby započítava len v prípade, že navrhnutý región obsahuje nejaký objekt. Podľa konvencie má trieda pre pozadie hodnotu $u = 0$. Preto je L_{loc} násobená členom

$$\lambda[u \geq 1] = \begin{cases} 1 & \text{ak } u \geq 1 \\ 0 & \text{inák} \end{cases}. \tag{3.5}$$

3.1.3 Faster R-CNN

V metóde Fast R-CNN je časovo najnáročnejšou úlohou vytvorenie návrhov regiónov, ktoré prebieha napríklad metódou selective search. Architektúra Faster R-CNN [36] z roku 2015 rieši tento problém vytvorením siete na návrh regiónov, ktorá zdieľa konvolučné vrstvy s časťou siete slúžiacou na extrakciu príznakov, a pridáva k ním niekoľko ďalších konvolučných vrstiev na získanie kandidátnych regiónov.

Faster R-CNN pozostáva z dvoch modulov. Prvým modulom je sieť na návrh regiónov (region proposal network, ďalej RPN), druhým je detektor z Fast R-CNN, ktorý využíva príznaky vypočítané extraktorom príznakov a regióny navrhnuté sieťou na návrh regiónov. Architektúra Faster R-CNN je znázornená ako súčasť Mask R-CNN na obrázku 3.5.

Okrem zväčšenia rýchlosti detekcie spôsobeného zdieľaním konvolučných vrstiev pre extrakciu príznakov a návrh regiónov je výhodou Faster R-CNN schopnosť učenia siete na návrh regiónov pre konkrétnu úlohu. V R-CNN a Fast R-CNN modul na návrh regiónov nebol schopný učiť sa.

Keďže RPN aj Fast R-CNN zdieľajú úvodnú konvolučnú časť siete a pri trénovaní modifikujú jej parametre rôznymi spôsobmi, v článku boli navrhnuté 3 techniky, ako zladiť trénovanie týchto dvoch systémov. Najčastejšie sa používa aproximované spojené trénovanie, pri ktorom RPN v každej iterácií navrhne regióny, ktoré sú pri trénovaní detektora považované za fixné. Pri spätnej propagácií potom dochádza ku skombinovaniu chyby pre RPN a Fast R-CNN. Tento druh trénovania je použitý vo väčšine dostupných implementácií vo frameworkoch ako TensorFlow či Pytorch.

Sieť na návrh regiónov

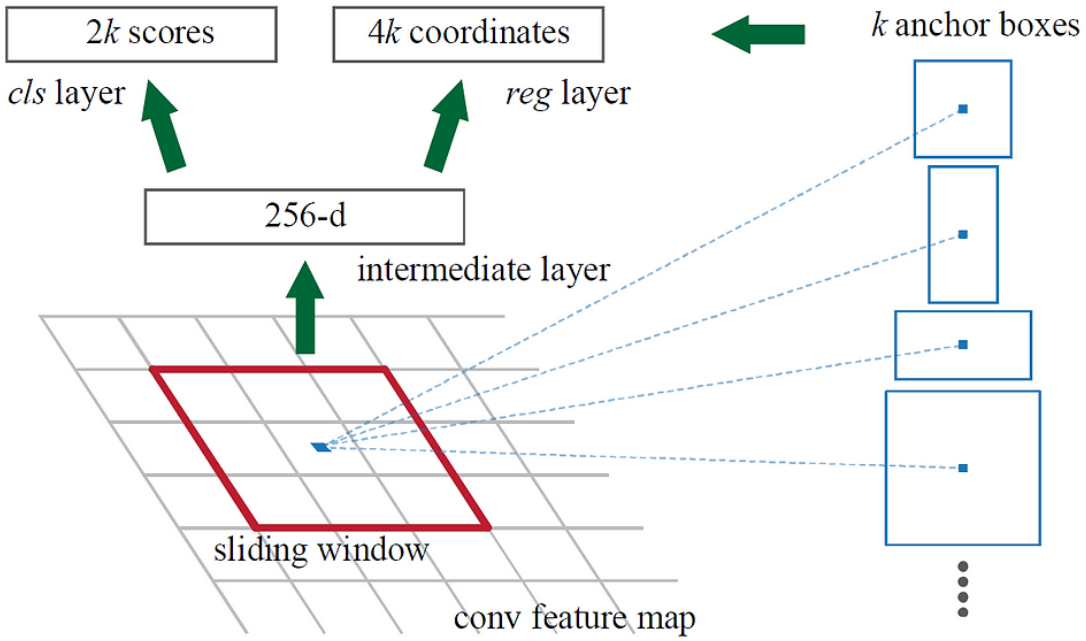
Sieť na návrh regiónov používa koncept referenčných boxov (anchors), ktoré približne zodpovedajú očakávaným rozmerom objektov. V článku sú použité referenčné boxy v troch

škálach a s tromi rôznymi pomermi strán, teda 9 rôznych referenčných boxov, ale obecne môže byť referenčných boxov k. Výstupom siete na návrh regiónov je pre každý bod mapy príznakov a každý referenčný box umiestnený stredom na tento bod pravdepodobnosť, že referenčný box obsahuje objekt, a upresnenie bounding boxu objektu.

Mapa príznakov vstupuje do konvolučnej vrstvy s veľkosťou konvolučného jadra n x n (v článku 3x3), výstupom ktorej je mapa príznakov rovnakých rozmerov, ale vyššej dimenzie (napr. 256). Táto mapa príznakov vyššej dimenzionality vstupuje paralelne do dvoch konvolučných vrstiev s veľkosťou jadra 1x1. Prvá konvolučná vrstva slúži na klasifikáciu a má $2k$ kanálov - pre každý referenčný box vráti dve hodnoty, ktoré po prechode vrstvou softmax reprezentujú pravdepodobnosť, že referenčný box obsahuje, respektíve neobsahuje objekt.

Druhá z paralelných konvolučných vrstiev slúži na regresiu bounding boxov objektu, má $4k$ kanálov a teda vráti mapu príznakov s dimenziou $4k$ (4 hodnoty spresňujúce bounding box pre každý z k referenčných boxov).

Schematické znázornenie siete na návrh regiónov a konceptu referenčných boxov je na obrázku 3.4.



Obr. 3.4: Sieť na návrh regiónov. Prevzaté z [36].

Je dôležité si všimnúť, že výpočet prebieha nad celou mapou príznakov zároveň. Neprichádza k žiadnemu prikladaniu okna, sieť sa učí rozpoznať objekty v zadaných referenčných boxoch na základe príkladov, ktoré dostáva pri trénovaní.

Pri trénovaní RPN sú za pozitívne príklady označené referenčné boxy, ktoré: (1) majú najvyššiu hodnotu IoU s niektorým ground-truth boxom objektu, alebo (2) majú s niektorým ground-truth boxom hodnotu IoU vyššiu než určený prah.

Za negatívne príklady sú označené tie referenčné boxy, ktoré majú s každým ground-truth boxom hodnotu IoU menšiu než ďalší určený prah. Ostatné referenčné boxy nie sú pri trénovaní využité.

Pri trénovaní každý mini-batch pozostáva z referenčných boxov jedného obrázka, pričom pomer medzi pozitívnymi a negatívnymi príkladmi je možné nastaviť.

Chybová funkcia

Chybová funkcia modulu RPN sa, rovnako ako chybová funkcia Fast R-CNN, skladá z chyby klasifikácie a chyby regresie bounding boxu. Chybová funkcia pre jeden obrázok je definovaná nasledovne:

$$L_{RPN}(p_i, t_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{loc}(t_i, t_i^*) \tag{3.6}$$

kde i je index referenčného boxu v mini-batchi, p_i je predpovedaná pravdepodobnosť, že referenčný box i obsahuje objekt. Ground-truth pre klasifikáciu p_i^* má hodnotu 1, ak je referenčný box pozitívny a 0, ak je negatívny.

Chyba klasifikácie L_{cls} pre dve triedy (objekt a pozadie) má predpis

$$L_{cls}(p_i, p_i^*) = -p_i^* log p_i - (1 - p_i^*) log(1 - p_i). \tag{3.7}$$

Člen L_{loc}, ktorý má rovnaký predpis ako vo Fast R-CNN, je násobený hodnotou p_i^*, čo zabezpečí, že chyba regresie bounding boxu sa do celkovej chyby započíta iba pri pozitívnych referenčných boxoch. N_{cls} a N_{loc} sú normalizačné koeficienty, λ je balančný parameter.

Chybová funkcia modulu Fast R-CNN je totožná s tou popísanou v 3.1.2.

3.1.4 Mask R-CNN

Systém Mask R-CNN [10], predstavený v roku 2017, je rozšírenie Faster R-CNN pre úlohu inštančnej segmentácie, ktoré popri triede a bounding boxe detegovaného objektu poskytuje ako výstup aj jeho masku. Maska je vypočítaná nezávisle pre každú triedu v plne konvolučnej vetve paralelnej s vetvou pre klasifikáciu a regresiu bounding boxu.

Architektúra Mask R-CNN je znázornená na obrázku 3.5.

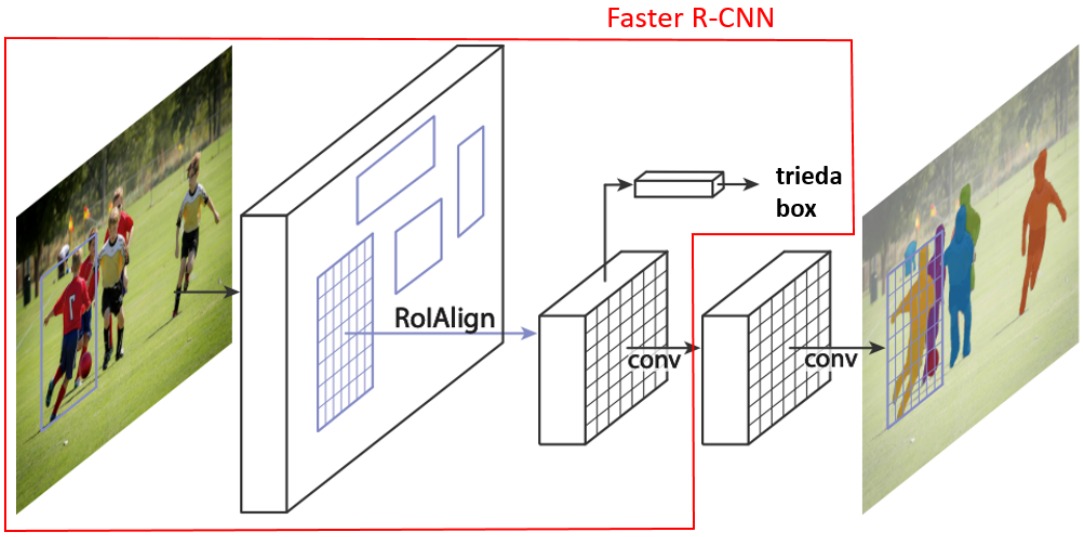
Vrstva RoIAlign

Vo Fast R-CNN bola predstavená vrstva RoIPool, ktorá z mapy príznakov pre každý navrhnutý región vyextrahovala mapu príznakov fixných rozmerov. Problémom RoIPool vrstvy je, že celkovo dvakrát vykonáva zaokrúhľovanie :

1. Pri mapovaní navrhnutého regiónu do mapy príznakov.
2. Pri rozdelení namapovanej časti mapy príznakov na menšie časti, v rámci ktorých prebieha pooling.

Toto zaokrúhľovanie síce nemusí ovplyvniť klasifikáciu, ale pri výpočte masky s presnosťou na pixely je veľmi nežiadúca.

Tento problém rieši vrstva RoIAlign, ktorá nepoužíva zaokrúhľovanie pri mapovaní ani pri rozdelení pre pooling. Mapovanie navrhnutého regiónu do mapy príznakov aj rozdelenie do okien pre pooling prebieha bez zaokrúhľovania, v každom okne pre pooling je na rovnomerne navzorkovaných 4 bodoch vypočítaná ich hodnota pomocou bilineárnej interpolácie. Na tieto hodnoty je potom aplikovaný pooling.



Obr. 3.5: Architektúra Mask R-CNN so znázornením časti, ktorá zodpovedá Faster R-CNN. Vo Faster R-CNN je namiesto RoIAlign často použitá vrstva RoIPool. Prevzaté a upravené z [10].

Chybová funkcia

Chybová funkcia Mask R-CNN sa skladá z troch častí :

$$L = L_{cls} + L_{loc} + L_{mask}. \tag{3.8}$$

Klasifikačná chyba L_{cls} a chyba regresie bounding boxu L_{loc} je identická s chybami popísanými v 3.1.2.

Výstupom vetvy pre výpočet masky je maska v rozlíšení m x m pre každú z c tried. Na každý z c výstupných kanálov je aplikovaná vrstva sigmoida, a L_{mask} je vypočítaná ako priemerná binárna vzájomná entropia, pričom sa počíta len pre masku s indexom zodpovedajúcim skutočnej triede detegovaného objektu.

Použitím vrstvy sigmoidy a binárnej vzájomnej entropie sa Mask R-CNN líši od plne konvolučných sietí použitých pre úlohu sémantickej segmentácie, ktoré na výstup zvyčajne aplikujú softmax a kategorickú vzájomnú entropiu.

3.2 Architektúry pre sémantickú segmentáciu

3.2.1 Plne konvolučné siete

Plne konvolučná sieť neobsahuje žiadne plne prepojené vrstvy. Stavebnými prvkami plne konvolučných sietí sú konvolúcia, pooling a aktivačné funkcie. Pre všetky tieto prvky platí, že závisia len na relatívnych priestorových súradniciach, preto sú plne konvolučné siete invariantné voči posunutiu. Typy vrstiev v plne konvolučných sieťach (spolu s plne prepojenou vrstvou) sú opísané v prílohe A.

Táto podsekcia bola vypracovaná na základe článku [23].

Nahradenie plne prepojených vrstiev konvolúciou

Článok [23] predstavil spôsob, akým je možné prispôsobiť úspešné konvolučné architektúry z oblasti klasifikácie pre sémantickú segmentáciu.

Konvolučné architektúry pre klasifikáciu sú obmedzené na vstupy fixnej dimenzie. Po prechode plne konvolučnou časťou typicky na záver nasleduje niekoľko plne prepojených vrstiev, ktoré vyžadujú vstup fixnej veľkosti a nezachovávajú priestorové súradnice. Výstup klasifikačnej siete je v tvare 1 x 1 x c, kde c je počet tried.

Článok ukazuje, že plne prepojené vrstvy je možné nahradiť konvolúciou s veľkosťou konvolučného jadra rovnou priestorovým rozmerom vstupu do plne prepojenej vrstvy. Tým sa vlastne sieť stane plne konvolučnou, dokáže prijať vstup ľubovoľnej veľkosti a jej výstupom je klasifikačná mapa.

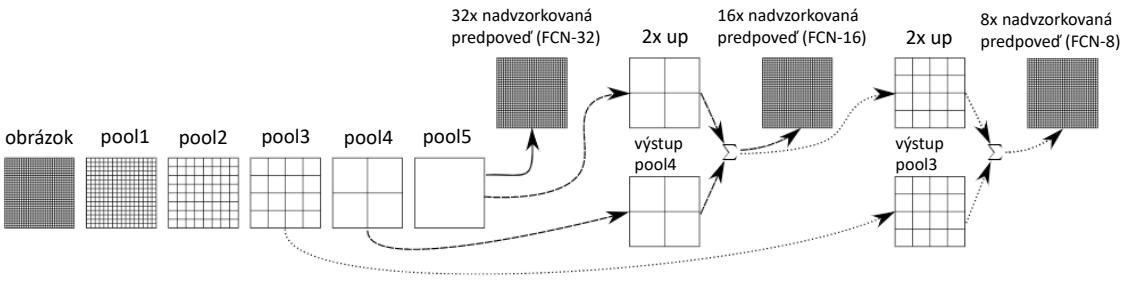
Častým prístupom k sémantickej segmentácií pred predstavením plne konvolučných sietí bolo vkladať obrázok do klasifikačnej siete po častiach fixnej veľkosti (patches) a výslednú klasifikačnú mapu zložiť z výsledkov klasifikácie pre jednotlivé časti. Výstup plne konvolučných sietí je ekvivalentný výstupu originálnych sietí na jednotlivých častiach, avšak plne konvolučný prístup je oveľa rýchlejší, keďže odstraňuje nutnosť separátneho výpočtu pre časti obrázka. Časti obrázka vkladané do siete navyše kvôli vzájomnému prekryvu obsahovali redundanciu.

Nadvzorkovanie

Priestorová veľkosť výstupu klasifikačnej siete upravenej podľa podsekcie 3.2.1 je zmenšená podvzorkovaním. Výstupom segmentácie je ale trieda pre každý pixel vstupného obrázka. Preto v architektúre plne prepojených sietí po časti vykonávajúcej podvzorkovanie (angl. subsampling) nasleduje časť vykonávajúca nadvzorkovanie (angl. upsampling).

Článok [23] navrhuje na nadvzorkovanie klasifikačnej mapy do pôvodných rozmerov obrázka použiť transponovanú konvolúciu. V článku je navyše predstavená skip architektúra, ktorá pri nadvzorkovaní spája klasifikačný výstup po podvzorkovaní obsahujúci hrubé sémantické informácie s jemnými lokálnymi informáciami o vzhľade vyextrahovanými v priebehu podvzorkovania.

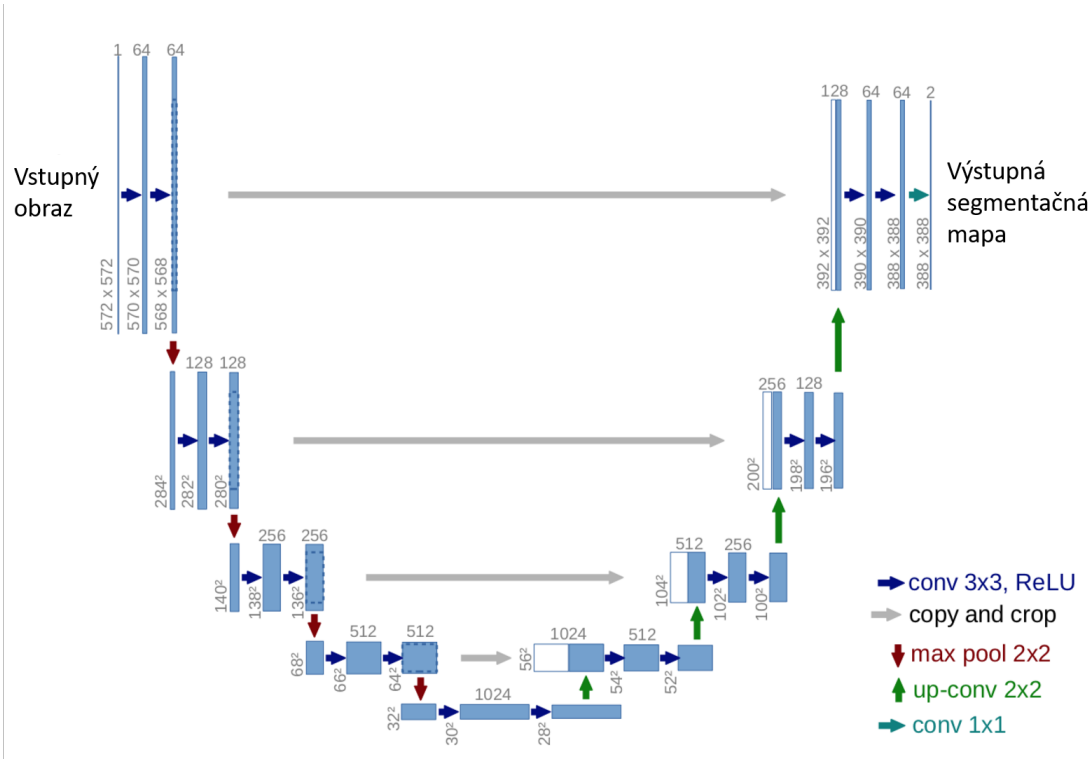
Na obrázku 3.6 sú znázornené architektúry FCN-8, FCN-16 a FCN-32 využívajúce skip prepojenia. Čísla 8, 16 a 32 udávajú, k akému nadvzorkovaniu dochádza v nadvzorkovacej časti.



Obr. 3.6: Kombinovanie lokálnych informácií získaných v priebehu podvzorkovania s globálnou sémantickou informáciou. Prevzaté a preložené z [23].

3.2.2 Architektúra U-Net

Architektúra U-Net [38] bola pôvodne navrhnutá pre segmentáciu obrázkov z oblasti biomedicíny. U-Net využíva architektúru plne konvolučných sietí, skladá sa teda z podvzorkovacej a nadvzorkovacej časti. V systéme sú tieto dve časti symetrické, preto má architektúra tvar písmena U. Schéma architektúry U-Net je na obrázku 3.7.



Obr. 3.7: Schéma architektúry U-Net. Prevzaté z [39] a preložené.

V podvzorkovacej časti sa opakovane aplikuje blok pozostávajúci z konvolúcii bez paddingu, z ktorých každá je nasledovaná jednotkou ReLu, a z max poolingu 2x2 s krokom 2, ktorý zabezpečí dvojnásobné zmenšenie priestorových dimenzií. Každý takýto blok naopak zdvojnásobí počet kanálov.

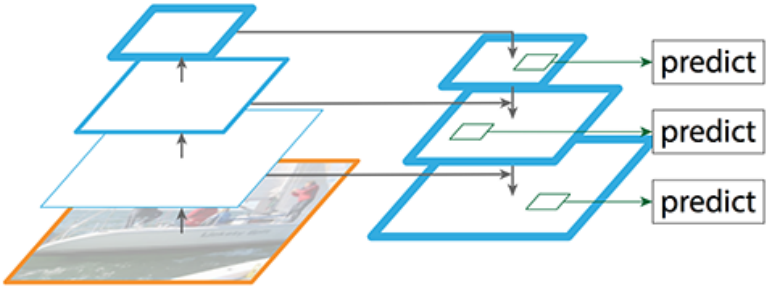
V každom kroku nadvzorkovacej časti mapa príznakov najprv prejde transponovanou konvolúciou, ktorá zdvojnásobí priestorové rozmery a naopak zmenší na polovicu počet kanálov. Výstup transponovanej konvolúcie je spojený s mapou príznakov zo zodpovedajúceho kroku podvzorkovacej časti. Dve konvolúcie, opäť každá nasledovaná jednotkou ReLU, potom znížia počet kanálov na polovicu. Na najvyššej úrovni nadvzorkovacej časti je ešte prevedená konvolúcia s jadrom 1x1, ktorej počet výstupných kanálov odpovedá počtu tried. Veľkosť vstupného obrázka je nutné vybrať tak, aby priestorové rozmery vstupu do každej operácie maxpoolingu boli párne.

V originálnej architektúre U-Net je kvôli strate krajných pixelov pri konvolúciách bez paddingu priestorový rozmer výstupu menší než veľkosť vstupného obrázka. Existujú však varianty U-Netu vracajúce segmentačnú mapu rozmeru vstupného obrázka.

3.2.3 Siete pyramíd príznakov

Siete pyramíd príznakov (anglicky feature pyramid networks), predstavené v [21], sú plne konvolučné siete uľahčujúce detekciu objektov rôznych veľkostí.

Sieť pyramíd príznakov svojou architektúrou, ktorá je znázornená na obrázku 3.8, pripomína U-Net. V jednotlivých úrovniach podvzorkovacej časti dochádza k zmenšeniu priestorových rozmerov konvolúciou. V nadvzorkovacej časti je medzi jednotlivými úrovňami mapa príznakov dvakrát zväčšená pomocou interpolácie najbližším susedom. K takto zväčšenej mape je po elementoch pripočítaná mapa zo zodpovedajúcej úrovne podvzorkovacej časti, čím dôjde k spojeniu sémanticky hodnotnej informácie z nadvzorkovacej časti s priestorovo presnejšou informáciou z podvzorkovacej časti. Rozdielom oproti plne konvolučným sieťam alebo U-Netu je to, že výstupom siete pyramíd príznakov sú mapy príznakov zo všetkých úrovní nadvzorkovacej časti.



Obr. 3.8: Architektúra siete pyramíd príznakov. Prevzaté z [21].

Sieť pyramíd príznakov sa často používa ako sieť na extrakciu príznakov v architektúre Fast R-CNN a jej nasledovníkoch Faster R-CNN a Mask R-CNN. Mapy príznakov z rôznych úrovní nadvzorkovacej časti sú použité na detekciu (a prípadne segmentáciu) objektov rôznych veľkostí.

Označme M_k tú mapu príznakov, ktorej veľkosti strán sú 2^k-krát menšie v porovnaní so stranami pôvodného obrázka. Potom podľa heuristiky uvedenej v [21] sú príznaky pre región veľkosti w x h vyextrahované z mapy príznakov M_k takej, že:

$$k = \left\lfloor k_0 + log_2 \sqrt{wh/s} \right\rfloor, \tag{3.9}$$

kde s je kanonická veľkosť regiónu a k_0 je označenie mapy, z ktorej sú extrahované príznaky pre objekt (resp. región) veľkosti s x s.

Z uvedeného vzťahu vyplýva, že príznaky pre malý región sú extrahované z mapy príznakov s veľkým rozlíšením a naopak príznaky pre veľký región z mapy s malým rozlíšením.

3.3 Chybové funkcie pre segmentáciu

Výstup modelov pre sémantickú segmentáciu je zvyčajne tenzor S v tvare C x H x W[1], kde počet kanálov C je rovný počtu tried pre klasifikáciu a W, H sú rozmery vstupného obrázka. Rovnaký tvar má aj ground-truth tenzor T. Zadefinujme si vektor $s_{i,j}$ =

[1]Poradie dimenzií závisí od použitého frameworku, vo frameworku Pytorch je to C x H x W, v TensorFlow W x H x C. Tu sa držím zápisu kompatibilného s frameworkom Pytorch.

$(S_{0,i,j}, S_{1,i,j}, \ldots, S_{C-1,i,j})$ a vektor $t_{i,j} = (T_{0,i,j}, T_{1,i,j}, \ldots, T_{C-1,i,j})$. V prípade, že ide o multiclass segmentáciu, teda pixel môže patriť len do jednej z C tried, vektor $t_{i,j}$ je v takzvanom one-hot kódovaní – práve jeden jeho prvok má hodnotu 1 a všetky ostatné majú hodnotu 0. Ak riešime problém multilabel segmentácie a pixel môže patriť naraz do viacerých tried, vektor t môže mať viacero jednotkových prvkov (obecne 0 až C).

Chybovú funkciu pre celý obrázok potom môžeme zapísať ako priemer výstupov chybovej funkcie pre každý jeho pixel :

$$L(T, S) = \frac{1}{H \cdot W} \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} L(t_{i,j}, s_{i,j}). \tag{3.10}$$

V nasledujúcich podsekciách sú popísané chybové funkcie pracujúce pre jeden pixel. Výstupný vektor $s_{i,j}$ bude zjednodušene označený ako s a ground-truth vektor $t_{i,j}$ ako t. Segmentácia je vlastne klasifikácia každého pixelu, preto majú popísané chybové funkcie zvyčajne pôvod v klasifikácií.

Táto sekcia bola vypracovaná s využitím [9].

3.3.1 Vzájomná entropia

Vzájomná entropia (cross-entropy) je chybová funkcia používaná pri klasifikácií. Je definovaná ako:

$$L_{CE}(t, s) = -\sum_{c=0}^{C-1} t_c \cdot log(s_c). \tag{3.11}$$

V prípade binárnej klasifikácie ($C = 2$) za predpokladu, že $t_1 = 1 - t_0$ a $s1 = 1 - s_0$ po dosadení do vzorca dostaneme:

$$L_{CE}(t, s) = -(t_0 \cdot log(s_0) + (1 - t_0) \cdot log(1 - s_0). \tag{3.12}$$

Funkcia sa na vektor s používa až po aplikácií aktivačnej funkcie softmax alebo sigmoida, a to spôsobom vysvetleným v podsekciách 3.3.2 a 3.3.3.

3.3.2 Kategorická vzájomná entropia

Kategorická vzájomná entropia vznikne aplikáciou funkcie softmax na vektor s a dosadením do vzťahu pre vzájomnú entropiu:

$$L_{CE}^{categorical}(t, s) = -\sum_{c=0}^{C-1} t_c \cdot log(softmax(s)_c). \tag{3.13}$$

Výstupom softmaxu aplikovaného na vektor s je vektor pravdepodobností so súčtom prvkom rovným 1. Výstupom kategorickej vzájomnej entropie je teda záporný súčet logaritmov pravdepodobností vypočítaných pre pozitívne triedy. Ak je riešeným problémom multiclass segmentácia, vektor t má len jeden jednotkový prvok a súčet preto obsahuje len jeden člen.

3.3.3 Binárna vzájomná entropia

Binárna vzájomná entropia vznikne aplikáciou funkcie sigmoida na vektor s nasledovanou výpočtom vzájomnej entropie. Výstup $sigmoid(s_c)$ vráti predpovedanú pravdepodobnosť,

že daný pixel patrí do triedy c. Vrátená pravdepodobnosť je nezávislá na všetkých ostatných hodnotách vektora s. Preto vieme problém klasifikácie pre C tried previesť na C nezávislých problémov binárnej klasifikácie, kde $1 - sigmoid(s_c)$ je predpovedaná pravdepodobnosť, že pixel nepatrí do triedy c a $1 - t_c$ je ground-truth, že pixel nepatrí do triedy c. Pre každý z týchto problémov je možné použiť vzájomnú entropiu s dvomi triedami (pozri rovnicu 3.12).

Binárna vzájomná entropia je potom daná predpisom :

$$L_{CE}^{binary}(t, s) = -\sum_{c=0}^{C-1}(t_c \cdot log(sigmoid(s_c)) + (1 - t_c) \cdot log(1 - sigmoid(s_c))). \tag{3.14}$$

3.3.4 Diceova chyba

Diceova chyba je často využívaná pri učení na dátovej sade s nerovnovážnym zastúpením tried [41]. Základom pre výpočet Diceovej chyby je Diceov koeficient, ktorý je v prípade multiclass klasifikácie pre každú triedu $c \in 0, 1, \ldots, C-1$ definovaný ako :

$$DiceCoef_c = \frac{2 \cdot \sum_{i=0}^{H-1}\sum_{j=0}^{W-1} softmax(s_{i,j})_c \cdot T_{c,i,j} + \epsilon}{\sum_{i=0}^{H-1}\sum_{j=0}^{W-1}(softmax(s_{i,j})_c + T_{c,i,j}) + \epsilon}, \tag{3.15}$$

kde ϵ je člen zavedený z dôvodu, aby sa zabránilo prípadnému deleniu nulou [41].

Diceoev koeficient, nie nepodobný koeficientu IoU, vyjadruje mieru správnosti predikcie pre danú triedu. Výraz $1 - DiceCoef_c$ potom vyjadruje Diceovu chybu pre triedu c. Celková Diceova chyba vznikne ako priemer chýb pre všetky triedy:

$$DiceLoss = \frac{1}{C}\sum_{c=0}^{C-1}(1 - DiceCoef_c). \tag{3.16}$$

To, že celková chyba je počítaná ako priemer chýb pre jednotlivé triedy, spôsobí, že v dátovej sade menej zastúpené triedy prispievajú do celkovej chyby rovnakou mierou ako triedy s vyššou frekvenciou.

3.4 Metriky pre segmentáciu

Táto sekcia predstaví najpoužívanejšie metriky v sémantickej a inštančnej segmentácií.

3.4.1 Metriky v sémantickej segmentácií

Nech C je počet tried a $N_{x,y}$ je počet pixelov patriacich do triedy x, ktorým model predikoval triedu y. Potom z pohľadu triedy x je $TP_x = N_{x,x}$ počet skutočne pozitívnych pixelov (true positives), $FP_x = \sum_{y=0,y\neq x}^{C-1} N_{y,x}$ počet falošne pozitívnych pixelov (false positives), $FN_x = \sum_{y=0,y\neq x}^{C-1} N_{x,y}$ počet falošne negatívnych pixelov (false negatives) a $TN_x = \sum_{y=0,y\neq x}^{C-1} N_{y,y}$ počet skutočne negatívnych pixelov (true negatives).

Metrika pixelová presnosť (pixel accuracy) udáva, aká časť všetkých pixelov bola správne klasifikovaná [18]:

$$PixelAccuracy = \frac{\sum_{x=0}^{C-1} N_{x,x}}{\sum_{x=0}^{C-1}\sum_{y=0}^{C-1} N_{x,y}}. \tag{3.17}$$

Táto metrika nie je vhodná v prípade nerovnovážnej dátovej sady, pretože slabo zastúpené triedy na ňu majú nedostatočný vplyv.

Metrika mIoU (mean intersection over union) vypočíta pre každú triedu pomer pixelov, ktoré do nej boli správne zaradené, k počtu pixelov, ktoré do triedy boli zaradené alebo ktoré do nej mali byť zaradené, a tieto hodnoty vypočítané pre jednotlivé triedy spriemeruje [18]:

$$mIoU = \frac{1}{C}\sum_{x=0}^{C-1}\frac{N_{x,x}}{\sum_{y=0}^{C-1}N_{x,y}+\sum_{y=0}^{C-1}N_{y,x}-N_{x,x}} = \frac{1}{C}\sum_{x=0}^{C-1}\frac{TP_x}{TP_x+FP_x+TN_x}. \quad (3.18)$$

Metrika precision pre triedu x udáva, aká časť pixelov označených za pozitívne sú skutočne pozitívne [18]:

$$Precision_x = \frac{N_{x,x}}{N_{x,x}+\sum_{y=0,y\neq x}^{C-1}N_{y,x}} = \frac{TP_x}{TP_x+FP_x}. \quad (3.19)$$

Metrika recall pre triedu x udáva, aká časť pozitívnych pixelov bola za pozitívne skutočne označená [18]:

$$Recall_x = \frac{N_{x,x}}{N_{x,x}+\sum_{y=0,y\neq x}^{C-1}N_{x,y}} = \frac{TP_x}{TP_x+FN_x}. \quad (3.20)$$

3.4.2 Metriky v inštančnej segmentácií

Metriky používané v inštančnej segmentácií zvyčajne robia vyhodnotenie na úrovni detegovaných objektov, nie jednotlivých pixelov.

V inštančnej segmentácií sa tiež používa koncept IoU. IoU hodnota masky detegovaného objektu, ktorú nazvime M_p, s maskou ground-truth objektu M_{gt}, je definovaná nasledovne [31]:

$$IoU_{mask}(M_p, M_{gt}) = \frac{area(M_p \cap M_{gt})}{area(M_p \cup M_{gt})}, \quad (3.21)$$

kde $area$ vracia počet pixelov útvaru.

Objekty v inštančnej segmentácií rovnako ako v sémantickej segmentácií delíme na skutočne pozitívne, falošne pozitívne, skutočne negatívne a falošne negatívne. Toto delenie využíva funkciu IoU_{mask}. Detegované objekty, ktoré majú hodnotu IoU_{mask} s nejakým ground-truth objektom väčšiu ako prah t, sú považované za skutočne pozitívne[2]. Detegované objekty, ktoré majú hodnotu IoU_{mask} so všetkými ground-truth objektami menšiu ako t, alebo síce majú hodnotu IoU_{mask} s ground-truth objektom vyššiu ako t, ale ich predikovaná trieda je nesprávna, sú považované za falošne pozitívne. Ground-truth objekty, ktorým neodpovedá žiaden skutočne pozitívny detegovaný objekt, sú považované za falošne negatívne.

Average precision a average recall

Výstupom každého detegovaného objektu je aj skóre istoty. Pri výpočte funkcií *Precision* a *Recall* môžeme zaviesť parameter prah istoty τ a za platné považovať len tie detekcie, ktoré majú skóre istoty väčšie než τ.

[2]Pre jeden ground-truth objekt ale môže byť len jeden detegovaný skutočne pozitívny objekt, ostatné, hoci majú hodnotu IoU_{mask} väčšiu ako t, sú považované za falošne pozitívne.

Predpisy funkcií *Precision* a *Recall* po zavedení parametra τ potom sú [31]:

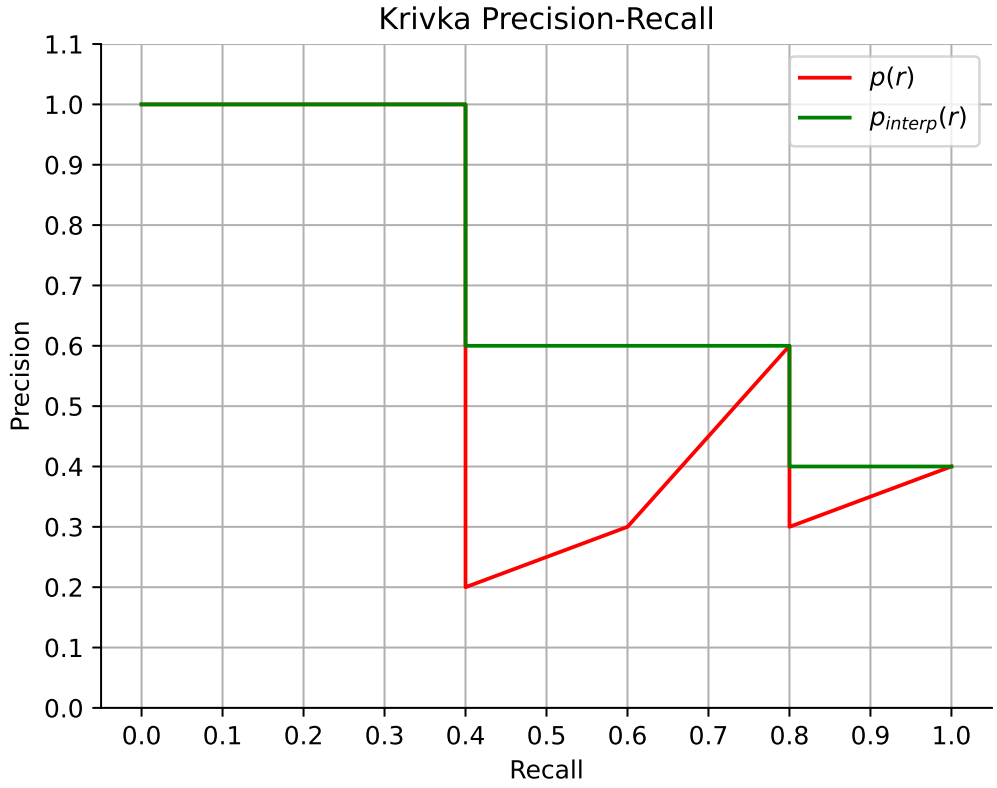
$$Precision_x(\tau) = \frac{TP_x(\tau)}{TP_x(\tau) + TP_x(\tau)} = \frac{TP_x(\tau)}{\text{všetky detekcie}(\tau)}, \tag{3.22}$$

$$Recall_x(\tau) = \frac{TP_x(\tau)}{TP_x(\tau) + FN_x(\tau)} = \frac{TP_x(\tau)}{\text{všetky ground truths}}. \tag{3.23}$$

S klesajúcim τ je funkcia $Recall_x(\tau)$ neklesajúca. Ak zobrazíme vzťah medzi $Precision_x(\tau)$ a $Recall_x(\tau)$ do grafu tak, že na x-ovej osi bude *Recall* a na y-ovej *Precision*, dostaneme tak krivku Precision-Recall (Precision-Recall curve). Označme ju $p(r)$.

Ďalej označme krivku $p_{interp}(r) = \max_{s \geq r} p(s)$ [11]. Takto definovaná krivka $p_{interp}(r)$ je nerastúca. Krivky $p(r)$ a $p_{interp}(r)$ sú zobrazené na obrázku 3.9. Metrika AP_x je potom definovaná ako určitý integrál pod krivkou $p_{interp}(r)$:

$$AP_x = \int_0^1 p_{interp}(r). \tag{3.24}$$



Obr. 3.9: Graf zobrazuje krivku Precision-Recall ($p(r)$) aj krivku $p_{interp}(r)$, ktorá vznikne nahradením hodnôt precision najvyššou hodnotou precision napravo.

Výsledná metrika AP vznikne ako priemer AP_x pre všetky triedy x:

$$AP = \frac{1}{C} \sum_{x=0}^{C-1} AP_x, \tag{3.25}$$

kde C je počet tried.

Okrem varianty, ktorá počíta IoU masiek skutočného a detegovaného objektu a vyjadruje teda kvalitu segmentácie, sa v inštančnej segmentácií využíva aj varianta s výpočtom IoU bounding boxov, vyjadrujúca kvalitu detekcie.

Metrika average recall má viacero možných definícií, tu bude uvedená tá, ktorá sa používa na vyhodnocovanie pri dátovej sade COCO. Predpis metriky average recall je:

$$AR_x = \max_{\tau | Precision_x(\tau) > 0} Recall_x(\tau), \tag{3.26}$$

teda je vybraná najvyššia hodnota recall taká, že metrika precision je pre danú mieru istoty τ nenulová [31]. Výsledná metrika AR je opäť daná ako priemer AR_x pre všetky triedy x:

$$AR = \frac{1}{C} \sum_{x=0}^{C-1} AR_x. \tag{3.27}$$

Kapitola 4

Návrh riešenia pre identifikáciu znakov na hĺbkovej snímke pneumatiky

Cieľom práce je navrhnúť a implementovať postup pre identifikáciu znakov na hĺbkovej snímke pneumatiky. Na tento účel bola pripravená dátová sada hĺbkových snímok, v ktorých je informácia o hĺbke premietnutá do stupňa šedi.

Základom navrhnutého riešenia je neurónová sieť pre inštančnú segmentáciu Mask R-CNN, natrénovaná na pripravenej dátovej sade. Z pamäťových dôvodov riešenie počíta s vkladaním hĺbkovej snímky do siete po prekrývajúcich sa častiach a s aplikáciou algoritmu potlačenia nemaximálnych hodnôt. Očakávaným výstupom je informácia o triede, pozícií a maske všetkých symbolov na hĺbkovej mape. Voliteľnou súčasťou návrhu je aj modifikácia siete Mask R-CNN umožňujúca lepšiu segmentáciu symbolov s členitým tvarom alebo tenkými líniami. Diagram navrhnutého riešenia je na obrázku 4.1.

4.1 Popis dátovej sady

Dátová sada poskytnutá firmou ME-INSPECTION SK, spol. s.r.o. obsahuje 92 hĺbkových snímok v stupňoch sivej s bitovou hĺbkou 8 bitov, väčšinou sú to hĺbkové snímky bočnice, ale ide aj o niekoľko snímok plášťa v oblasti ochrany ráfika.

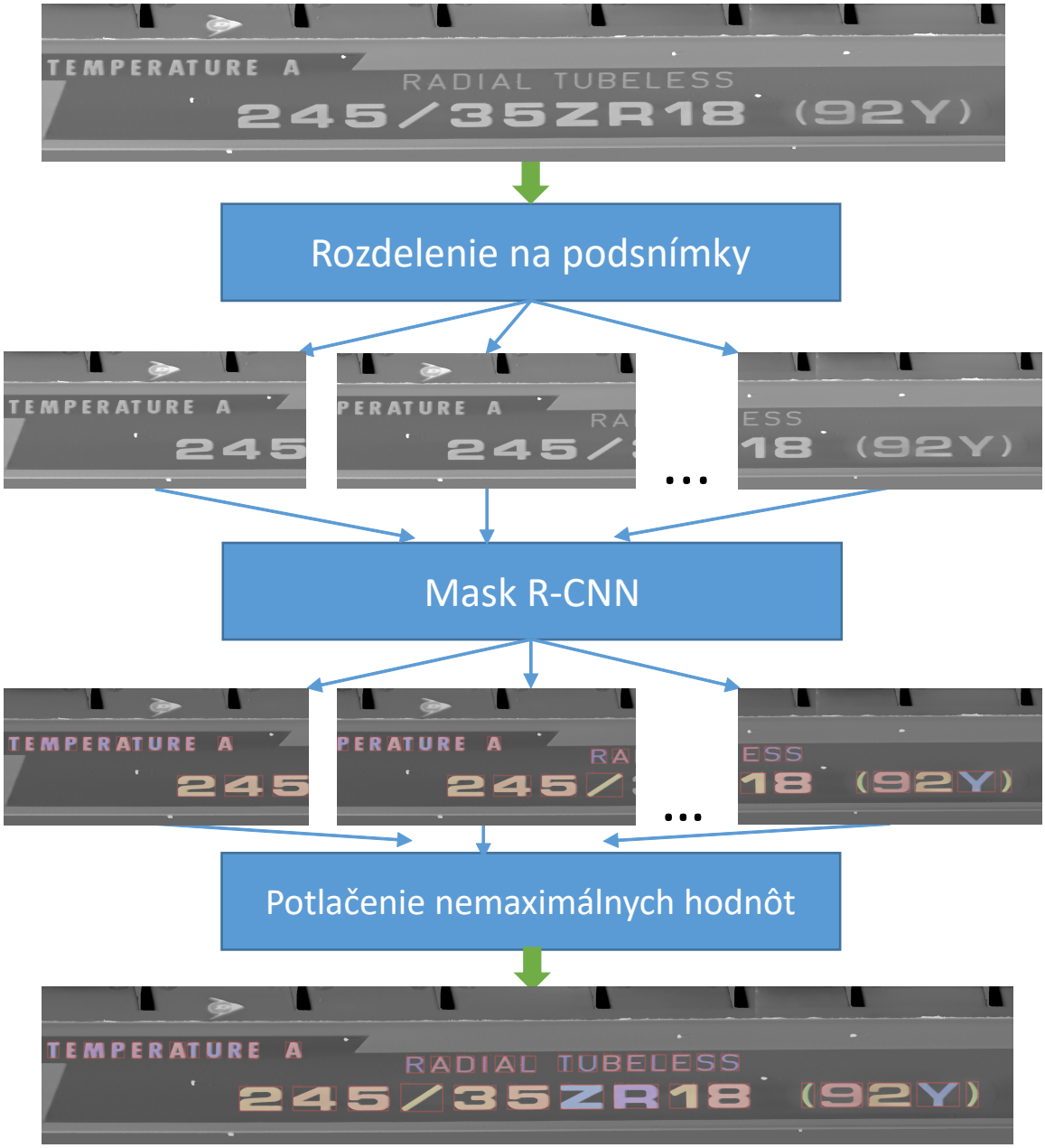
Tieto snímky vznikli spracovaním výsledkov 3D skenu, ktorého výstupom je množina 3D bodov. Najprv došlo k interpolácií neznámych bodov, následne boli 3D body zarovnané do 2D roviny a ich hĺbka bola premenená na stupeň šedi.

Ku každej snímke dátová sada navyše obsahuje anotáciu každého symbolu nachádzajúceho sa na pneumatike[1]. Popis formátu anotácií sa nachádza v prílohe B.

Anotácia dátovej sady poskytnutej firmou mala viacero problémov vrátane toho, že nebola kompletná. Popis dokončenia anotácie dátovej sady sa nachádza v podsekcií 5.1.2.

V dátovej sade sa nachádza celkovo 21 typov pneumatík. Snímky v rámci jedného typu majú rovnaké rozloženie nápisov a symbolov, môžu sa líšiť konkrétnymi číselnými hodnotami a tiež môžu byť navzájom cyklicky posunuté.

[1] Niektoré symboly, ktoré firma nepotrebuje detegovať, neboli po konzultácií s firmou zaradené do anotácie.



Obr. 4.1: Diagram navrhnutého riešenia.

4.1.1 Rozdelenie dátovej sady na trénovaciu a testovaciu množinu

Dátová sada bola po dokončení anotácie rozdelená na dve časti – trénovaciu a testovaciu sadu. Vzhľadom na malú veľkosť dátovej sady som sa rozhodol nevytvárať validačnú sadu. Trénovanie prebieha na trénovacej sade, na testovacej sade sú validované výsledky počas trénovania. Vyhodnotenie naučeného modelu prebieha na testovacej sade.

Pri rozdelení dátovej sady bol najprv ručne vybraný jeden typ pneumatiky, ktorý sa skladal len zo symbolov nachádzajúcich sa aj na iných typoch pneumatiky, a pneumatiky tohto typu boli zaradené len do testovacej sady. Dôvodom bolo, aby v testovacej sade bol typ pneumatiky, ktorý pri trénovaní sieť vôbec nevidela. Ostatné pneumatiky boli do dvoch sád vybrané skriptom, a to nasledovne:

- Pokiaľ z daného typu bola v dátovej sade len jedna snímka, tá bola zaradená do trénovacej sady.
- Ak bolo z daného typu v dátovej sade 2 a viac snímok, $\lceil \frac{1}{5} \rceil$ z nich bola zaradená do testovacej sady a zvyšné snímky boli zaradené do trénovacej sady.

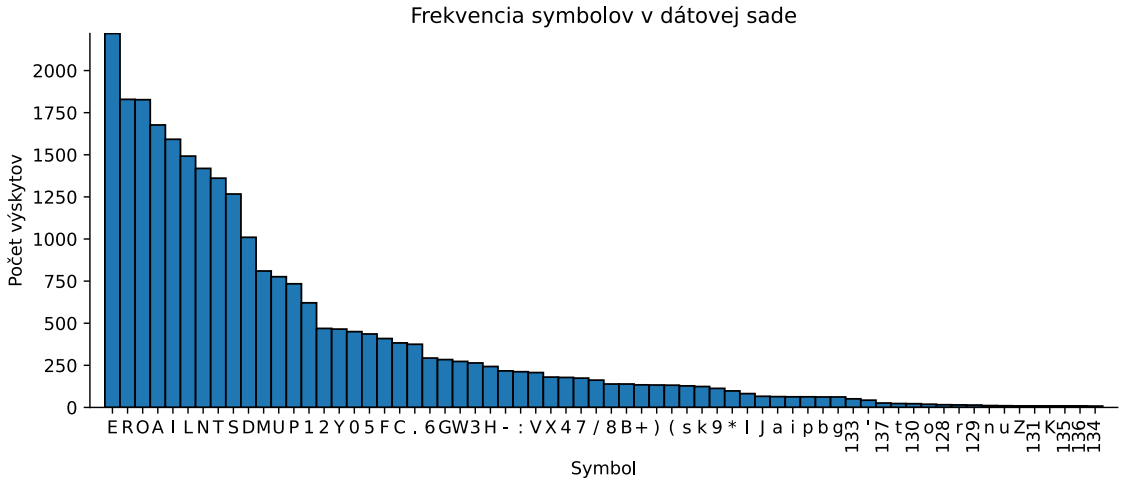
Po takomto rozdelení mala trénovacia sada 66 snímok a testovacia sada 26 snímok.

4.1.2 Problém nerovnováhy tried v dátovej sade

V dátovej sade poskytnutej firmou bol veľký rozdiel v početnosti jednotlivých tried/typov symbolov. Najčastejším znakom bolo 'E', ktoré sa v dátovej sade vyskytlo celkovo 2220-krát. Naopak s veľmi nízkou frekvenciou sa v dátovej sade vyskytovali najmä špeciálne symboly, z ktorých najmenej zastúpený bol symbol slnka (letnej pneumatiky) s 8 výskytmi.

Histogram na obrázku 4.2 znázorňuje frekvenciu jednotlivých symbolov v dátovej sade. Špeciálne (netextové) symboly sú v histograme označené trojmiestnym číselným kódom a ich popisky sú otočené. Bližší popis špeciálnych symbolov a kódov, ktorými sú označené, sa nachádza v prílohe C.

Ako vidieť na histograme, veľmi malé zastúpenie mali okrem špeciálnych symbolov aj malé písmená, ale napríklad aj písmená 'K' alebo 'Z'.

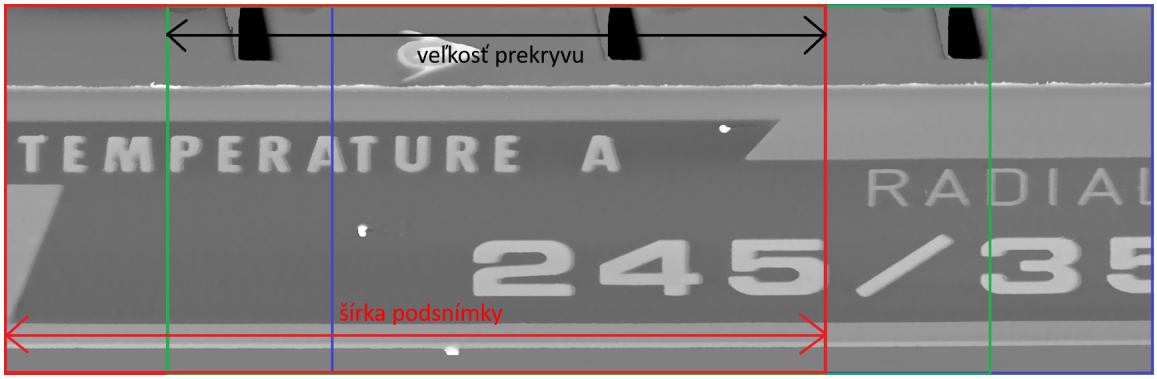


Obr. 4.2: Histogram frekvencie symbolov v dátovej sade. Špeciálne (netextové) symboly sú označené trojmiestnym číselným kódom. Ich bližší popis sa nachádza v prílohe C.

4.2 Návrh riešenia problému identifikácie znakov

Na riešenie problému identifikácie znakov z firemnej dátovej sady som sa rozhodol použiť architektúru Mask R-CNN. Každému typu symbolu v dátovej sade odpovedá jedna trieda v modeli Mask R-CNN a model navyše vyžaduje špeciálnu triedu pre pozadie.

Keďže vstupná hĺbková mapa je najmä na šírku veľmi veľká (65536 pixelov), celá sa na použitom počítači (8 GB) nezmestí na GPU. Preto sú hĺbkové snímky ešte pred trénovaním či inferenciou rozdelené na menšie podsnímky s výškou totožnou s výškou pôvodnej snímky a s vhodne zvolenou šírkou a veľkosťou prekryvu medzi susediacimi podsnímkami. Rozdelenie snímky na podsnímky znázorňuje obrázok 4.3.



Obr. 4.3: Ukážka, akým spôsobom je snímka rozdelená na menšie časti. Pre lepšiu názornosť je zobrazená len časť hĺbkovej snímky.

Tento prístup pripomína klasický prístup s použitím posuvného okna (sliding window). Kým posuvné okno býva zvyčajne používané kvôli fixnej veľkosti vstupu alebo preto, aby algoritmus prešiel možné polohy a často aj rozmery objektu a na časť obrázku nachádzajúcu sa vo vnútri okna aplikoval klasifikátor, v prípade tejto práce sú dôvodom rozdelenia snímky na podsnímky len pamäťové nároky. Nevýhodou navrhnutého prístupu je samozrejme redundancia prameniaca z opakovaného vkladania tej istej časti obrázka do siete.

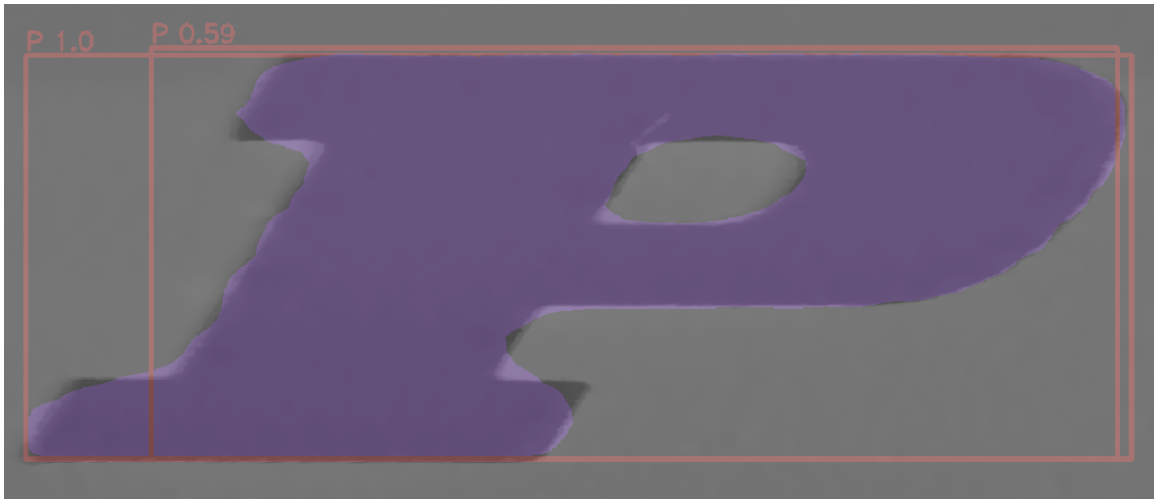
Dôležité je poznamenať, že rozdelenie na podsnímky prebieha ešte pred trénovaním, podsnímky aj ich anotácie sa uložia na disk.

Hĺbková snímka je teda vo fáze trénovania, testovania aj inferencie do siete vkladaná po prekrývajúcich sa podčastiach. Z toho vyplýva nutnosť zloženia predikcií pre jednotlivé podsnímky do celkovej predikcie veľkosti pôvodnej hĺbkovej mapy.

Pri metóde Mask R-CNN môžu byť niektoré inštancie objektu detegované na viacerých podsnímkach. Preto pri tvorbe celkového výstupu vo veľkosti pôvodnej hĺbkovej snímky používam algoritmus potlačenia nemaximálnych hodnôt (angl. non-maximum suppression, pozri napríklad [33]), ktorý zahodí detegované objekty, ktorých bounding box má s bounding boxom nejakého iného detegovaného objektu (rovnakej či rôznej triedy) skóre IoU väčšie než daný prah, ale skóre istoty menšie než skóre istoty objektu, s ktorým sa prekrýva. Príklad, kedy je treba použiť potlačenie nemaximálnych hodnôt, je na obrázku 4.4.

V úvodnej fáze riešenia práce bolo cieľom vykonať aj experimenty s použitím modelu U-Net pre sémantickú segmentáciu. Návrh sa v prípade použitia U-Netu líši len tým, že algoritmus potlačenia nemaximálnych hodnôt prebieha na úrovni jednotlivých pixelov, nie celých objektov.

Pri metóde U-Net sa navyše výrazne prejavoval problém nerovnováhy tried, a to nie len medzi jednotlivými typmi symbolov, ale najmä medzi jednoznačne najzastúpenejšou



Obr. 4.4: Obrázok znázorňuje dve detekcie toho istého symbolu 'P' z dvoch rôznych podsnímok. Detekcie sa navzájom prekrývajú a keďže pravá detekcia má nižšie skóre istoty, bude zahodená algoritmom potlačenia nemaximálnych hodnôt.

triedou pozadie a ostatnými triedami. Navrhnutým riešením je použitie Diceovej chybovej funkcie popísanej v podsekcií 3.3.4, ktorá zabezpečí rovnako veľký vplyv všetkých tried na výslednú chybu.

4.2.1 Modifikácia Mask R-CNN pre segmentáciu symbolov s komplikovaným tvarom

Nevýhodou siete Mask R-CNN je to, že ňou predikované masky majú fixný rozmer bez ohľadu na veľkosť detegovaného objektu alebo komplikovanosť jeho tvaru, a až následne sú bilineárnou interpoláciou prevzorkované na rozmery detegovaného objektu.

Fixná veľkosť masky je daná veľkosťou výstupu operácie RoIAlign, ktorá podľa veľkosti regiónu predpovedaného sieťou na návrh regiónov z jednej z úrovní pyramídy príznakov vyextrahuje tenzor fixnej veľkosti.

V experimentoch popísaných v článku o Mask R-CNN [10] je veľkosť výstupu vrstvy RoIAlign 14 x 14. Keďže v segmentačnej vetve dochádza k 2-násobnému zväčšeniu tenzora pomocou transponovanej konvolúcie, predikované masky majú fixnú veľkosť 28 x 28. Pri výpočte chybovej funkcie je skutočná maska objektu prevzorkovaná na veľkosť 28 x 28 a je spočítaná binárna vzájomná entropia medzi predikovanou a skutočnou maskou.

V mnou prevedených experimentoch (pozri podsekciu 6.1.5) sa ukázalo, že predikcia a výpočet chyby na takomto nízkom rozlíšení nie sú dostatočné pre segmentáciu symbolov s tenkými líniami alebo veľkou členitosťou, ako sú napríklad symboly s kódmi 128, 129 či 133 (pozri prílohu C).

V práci navrhujem a experimentálne overujem dva spôsoby riešenia tohto problému:

1. Zväčšenie výstupu vrstvy RoIAlign pre segmentáciu. Týmto zväčšením sa v rovnakom pomere zväčší aj predikovaná maska.

2. Vytvorenie zdvojenej segmentačnej vrstvy. Nevýhodou predchádzajúceho spôsobu je zvýšená pamäťová náročnosť modelu prameniaca zo zväčšenia veľkosti masky aj pre tie triedy, s ktorých segmentáciou nebol problém. Alternatívou je preto vytvorenie

zdvojenej segmentačnej vrstvy, kde v jednej vetve budú v štandardnej veľkosti predikované masky symbolov s jednoduchším tvarom a v druhej s väčším rozlíšením masky symbolov s tenkými líniami a veľkou členitosťou. Schéma zdvojenej segmentačnej vrstvy je na obrázku 4.5

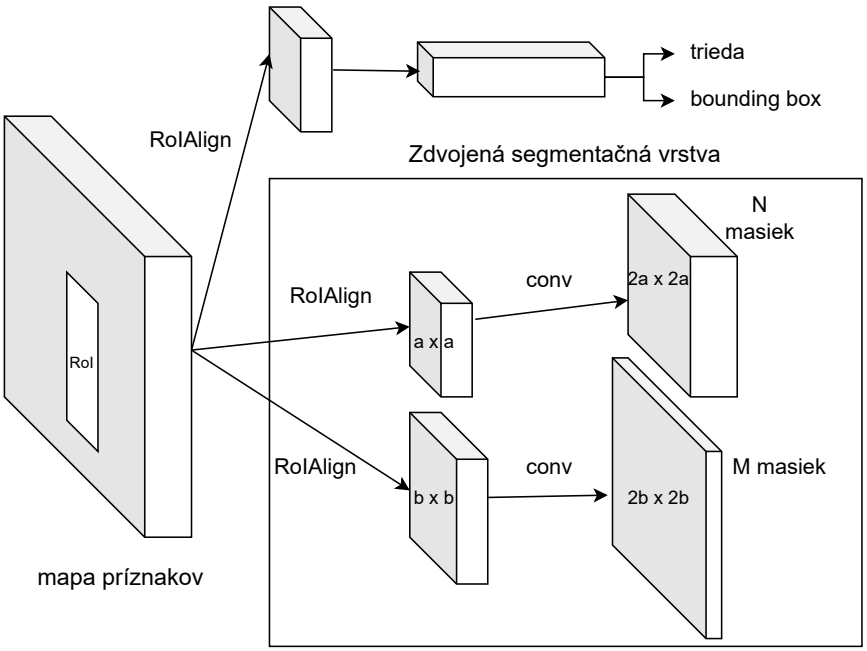
Maska pre každú triedu symbolov bude teda počítaná v práve jednej z dvoch segmentačných vetiev. Pri rozdelení tried do vetiev je nutné okrem kvality segmentácie prihliadať aj na:

- kvalitu detekcie - ak sa triedu nedarí detegovať, kvalita segmentácie (ktorá od detekcie závisí) sa so zväčšením výstupnej masky nemusí výrazne zvýšiť,
- veľkosť symbolov danej triedy - nie je vhodné, aby mala predikovaná maska výrazne väčšiu veľkosť ako skutočná maska.

Rozdelenie tried do vetiev v rámci tejto práce prebiehalo manuálne na základe uvedených kritérií.

Pozitívom tohto prístupu je aj oddelené trénovanie segmentácie jednoduchých a náročných symbolov. Chyba segmentácie, ktorá je počítaná ako vážený súčet chýb pre jednotlivé vetvy, je propagovaná len do tej vetvy, do ktorej patrí objekt podľa jeho skutočnej triedy. Váhy sú priamo úmerné počtu objektov patriacich do danej vetvy podľa skutočnej triedy a sú normalizované tak, aby ich súčet bol 1. Pri inferencii je výsledná maska vybraná z jednej z vetiev podľa predikovanej triedy symbolu.

Alternatívou k zdvojenej segmentačnej vrstve by bolo trénovanie dvoch rôznych sietí. Veľkou výhodou siete so zdvojenou segmentačnou vrstvou je fakt, že môže byť trénovaná naraz a okrem segmentačnej vrstvy sú všetky výpočty zdieľané pre všetky triedy.



Obr. 4.5: Schéma znázorňuje zdvojenú segmentačnú vrstvu. Vrstva pre klasifikáciu a regresiu bounding boxov zostáva nezmenená. Vytvorené pomocou nástroja diagrams.net [6].

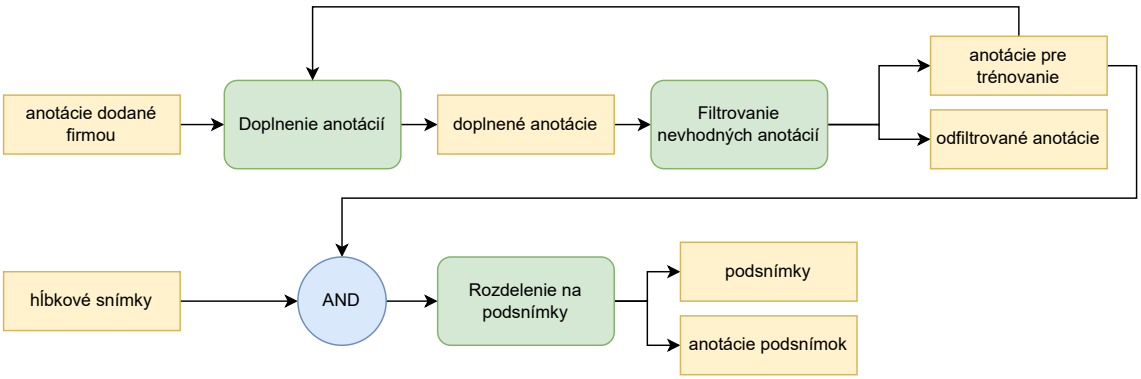
Kapitola 5

Implementácia

Zdrojový kód tejto práce bol implementovaný v jazyku Python s využitím viacerých frameworkov a knižníc, z ktorých najdôležitejšie boli Pytorch, pandas, NumPy, OpenCV, PytorchLightning a Segmentation Models Pytorch. Popis jednotlivých adresárov, do ktorých bol rozdelený zdrojový kód, je v prílohe D.

5.1 Práca s dátovou sadou

Nasledujúca sekcia popisuje implementované funkcie a skripty pre prácu s dátovou sadou. Práca s dátovou sadou pozostávala v prvej fáze riešenia z vytvorenia umelého generátora na testovacie účely, v nasledovných fázach po dodaní dátovej sady bolo treba doplniť a filtrovať anotácie a následne rozdeliť snímky a ich anotácie na časti, ako je znázornené na obrázku 5.1.



Obr. 5.1: Diagram znázorňuje kroky, z ktorých pozostávala príprava dátovej sady. Symbol krúžku s textom 'AND' značí, že proces má viacero vstupov. Vytvorené pomocou nástroja diagrams.net [6].

5.1.1 Umelý generátor dátovej sady

V období predtým, ako firma poskytla dátovú sadu pozostávajúcu z hĺbkových snímok, masiek a čiastočnej anotácie, som mal k dispozícií len malú množinu masiek symbolov. Táto množina masiek bola využitá na vytvorenie jednoduchej umelej dátovej sady, ktorej účelom

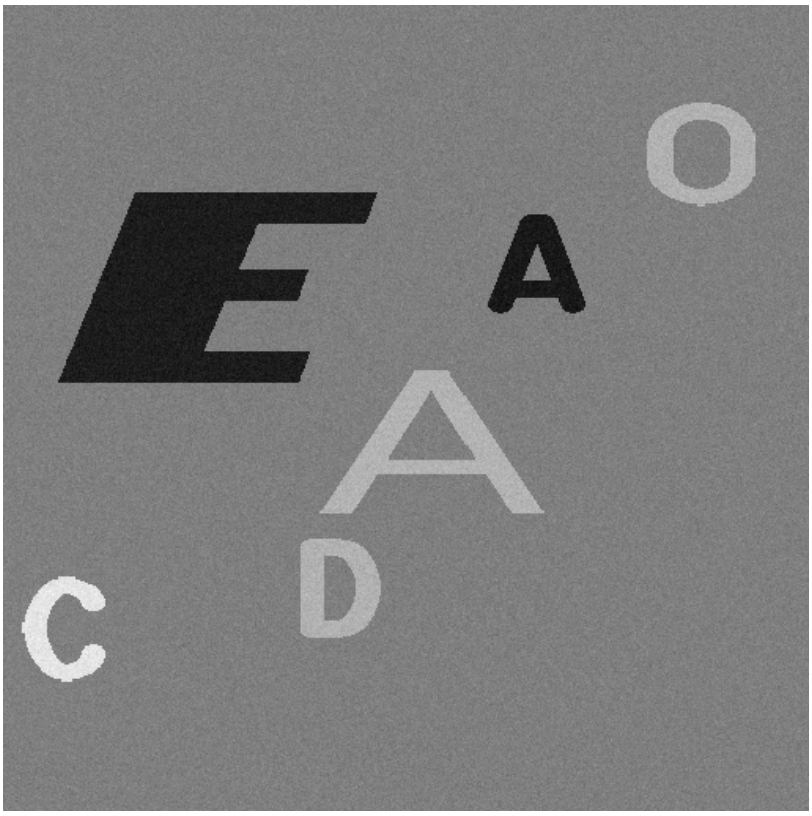
malo byť overiť vhodnosť architektúry Mask R-CNN pre úlohu detekcie a segmentácie symbolov.

Na vytvorenie umelej dátovej som implementoval jednoduchý generátor, ktorý prijíma cesty k adresárom obsahujúcim masky symbolov, a z každého adresára pridá do obrázka zadaný počet symbolov.

V prvom kroku je vygenerované pozadie obrázka istej farby (v odtieňoch sivej) s jemným šumom. Na toto pozadie sú potom svetlejšou alebo tmavšou farbou kladené jednotlivé symboly takým spôsobom, aby sa neprekrývali. Veľkosť symbolu aj jeho farba sú dané normálnym rozložením, ktorého rozptyl je, podobne ako veľkosť šumu v rámci symbolu, parametrom generátora.

Generátor vytvorí zadaný počet obrázkov zadaných rozmerov. Spolu s každým obrázkom vytvorí aj pixelovú masku inštancií symbolov a pixelovú masku tried. Obrázky aj masky oboch typov (tiež vo formáte obrázka) sú uložené do zadaného adresára.

Príklad vygenerovaného obrázka je na obrázku 5.2.



Obr. 5.2: Príklad obrázka vytvoreného umelým generátorom dátovej sady.

5.1.2 Príprava dátovej sady

Ako bolo spomenuté vyššie, anotácia k dátovej sade poskytnutej firmou mala viacero nedostatkov. Táto podsekcia bližšie popíše tieto nedostatky a spôsob ich riešenia.

Prvým a najvážnejším problémom boli symboly, ktoré kompletne chýbali v anotácii, t.j. nebol o nich záznam v CSV súbore a chýbal aj PNG súbor s ich maskou.

Na znázornenie, ktoré symboly sa v anotácii nachádzali, som implementoval skript, ktorý vygeneroval obrázky vzniknuté dokreslením bounding boxov okolo anotovaných symbolov.

S týmito obrázkami som ďalej pracoval pomocou anotačného nástroja LabelImg [44], ktorý z vyskúšaných nástrojov ako jediný umožňoval anotáciu takto veľkých obrázkov bez ich zmenšenia a zároveň umožňoval prácu s dátami uloženými na lokálnom počítači[1].

Chýbajúce symboly sa často nachádzali v zhlukoch. V nástroji LabelImg som tieto zhluky symbolov voľne označil do bounding boxov, ako je vidieť na obrázku 5.3. Obrázok tiež v červených štvorcoch zobrazuje symboly nachádzajúce sa v pôvodnej anotácii.

Obr. 5.3: Anotácia chýbajúcich symbolov pomocou nástroja LabelImg.

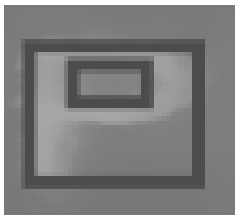
Samotný popis typu symbolu, vytvorenie jeho masky a bounding boxu už prebiehali pomocou mnou implementovaného skriptu. Skript zobrazí používateľovi časť hĺbkovej mapy zodpovedajúcu bounding boxu zhluku. Používateľ pomocou posúvadla (slidera) vyberie prah odlišujúci symboly od pozadia a pri posúvaní posúvadla sa mu zobrazuje výsledok prahovania s daným prahom, ako je vidieť na obrázku 5.4. Manuálny výber prahu mal zmysel najmä pri šrafovaných symboloch, pri klasických symboloch zvyčajne postačil automatický výber metódou Otsu [30]. Po vybraní prahu a binarizácii prahovaním sú v binárnom obrázku nájdené kontúry pomocou funkcie `findContours` knižnice OpenCV. Používateľovi sú postupne zobrazované bounding boxy okolo nájdených kontúr. Ak je nájdená kontúra symbolom, používateľ zadá jeho znak (ak je možné ho zadať na klávesnici) alebo ASCII kód.

Maska symbolu vznikne vyplnením oblasti ohraničenej kontúrou. Problém vzniká pri symboloch, ktoré obsahujú diery, ako je napríklad písmeno 'P' na obrázku 5.5. Kontúra diery sa dá získať z hierarchie kontúr vrátenej funkciou `findContours`. Na rozlíšenie, kedy ide o dieru symbolu, ktorej oblasť do masky nepatrí a preto ju treba vyplniť farbou pozadia, a kedy o malú oblasť pixelov, ktorá vznikla kvôli prahovaniu a v skutočnosti do masky symbolu patrí, sa ako dostatočné ukázalo použiť fixný prah na šírku a výšku kontúry.

Niektoré symboly, ako napríklad znaky ':' a 'i', sú zložené z viacerých komponentov. Pre tieto symboly má používateľ možnosť po výbere prahu a binarizácii preskočiť krok hľadania

[1]Kvôli licenčným podmienkam firmy nebolo možné, aby dáta opustili firemný počítač.

Obr. 5.4: Manuálny výber prahu predchádzajúci nájdeniu kontúr.



Obr. 5.5: Nájdené kontúry v prípade symbolu s dierou.

kontúr a za masku je považovaný jednoducho výsledok binarizácie (obmedzený na bounding box binarizovaných pixelov).

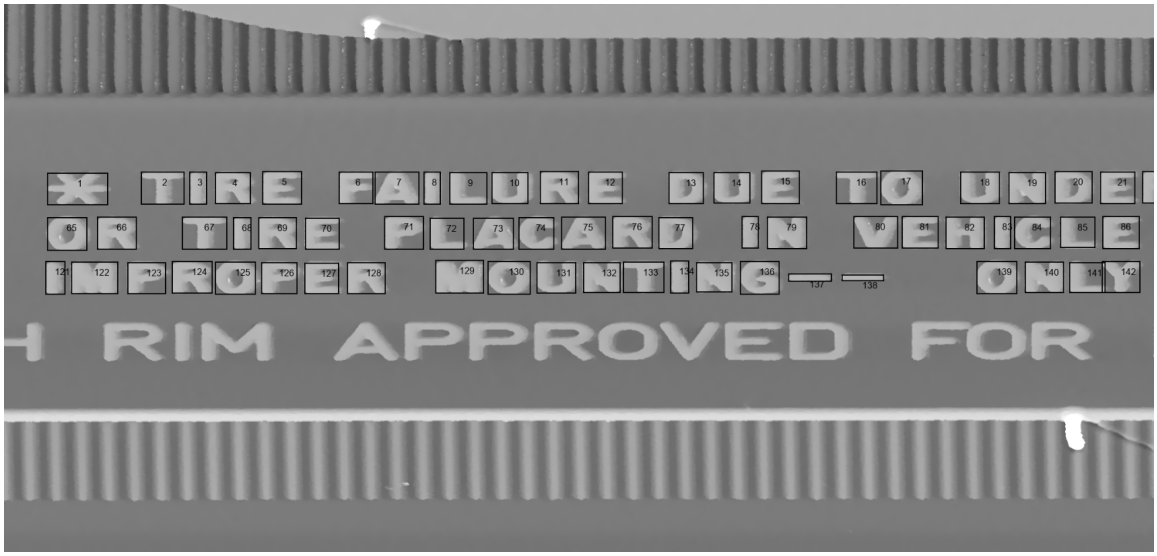
Druhým problémom dátovej sady boli znaky, ktorých maska a informácie o pozícii v anotácii boli, lenže ich kód triedy v stĺpci Unicode CSV súboru nebol správny. Typicky išlo o súvislé nápisy a názvy výrobcu pneumatiky. Na anotáciu týchto znakov bol tiež použitý skript, ktorý znázornil bounding box symbolu na hĺbkovej mape a vyzval používateľa, aby zadal znak alebo jeho ASCII kód.

Zvláštnym prípadom takéhoto nápisu bolo bezpečnostné varovanie o dĺžke 182 znakov, ktoré sa opakovalo na mnohých pneumatikách. Poradie symbolov (stĺpec Component index) sa medzi snímkami menilo, preto nebolo možné jednoduchým spôsobom pridať predpripravené značenie jednotlivých symbolov hromadne pre všetky výskyty tohto nápisu. Na riešenie tohto problému som využil fakt, že text bezpečnostného varovania sa zvyčajne nachádzal v troch riadkoch pod sebou, a na základe pozícií symbolov bolo možné symboly zoradiť tak, aby zodpovedali predchystanému značeniu. Zoradené symboly bezpečnostného varovania, ktoré mohli byť jednoducho oanotované pomocou predpripraveného značenia, sú na obrázku 5.6.

Tretím a posledným výrazným problémom dátovej sady boli symboly, ktoré boli v anotácii rozdelené na jednotlivé komponenty, ale detegované mali byť ako jeden objekt. Príkladom je grafický symbol topánky s krídlom skladajúci sa zo siedmich spojitých komponentov. Tieto symboly boli pomocou skriptu spojené do jedného symbolu a bola vypočítaná nová maska a bounding box takto spojeného symbolu.

5.1.3 Filtrácia dátovej sady

Filtrácia bola v procese prípravy dátovej sady použitá buď na úplné odstránenie vybraných symbolov z anotácie, alebo na ich odfiltrovanie do iného súboru a neskoršie doanotovanie pomocou metód popísaných v predchádzajúcej podsekcii.

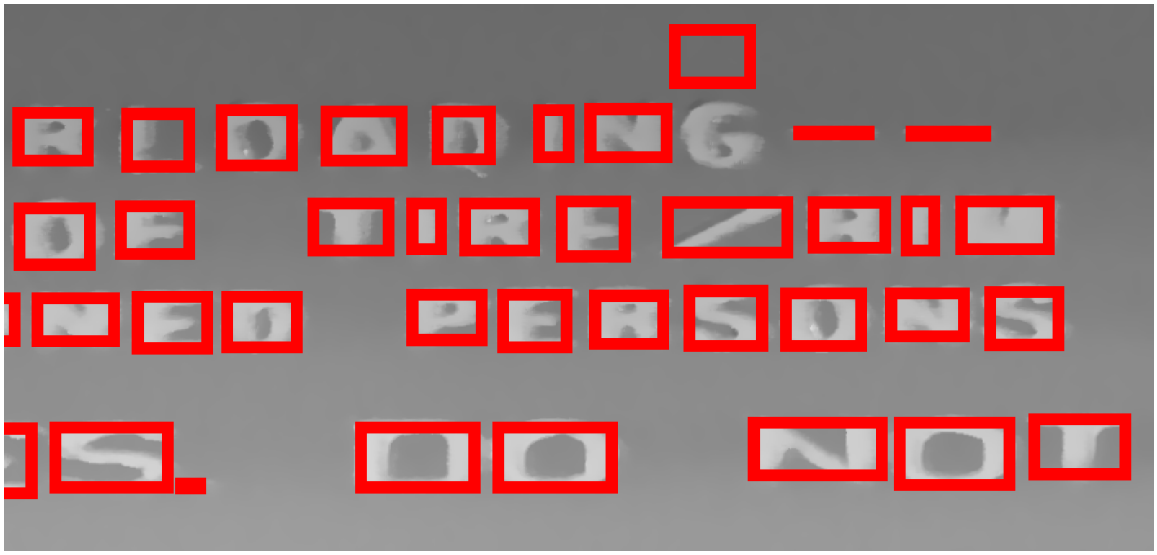


Obr. 5.6: Zoradené symboly bezpečnostného varovania (v bounding boxe sa nachádza poradové číslo symbolu).

Implementované funkcie a triedy umožňujú filtrovať dátovú sadu na základe rôznych kritérií. CSV súbory s anotáciou sú pri filtrácii načítané do DataFrame knižnice *pandas*.

Funkcia vykonávajúca filtráciu anotácie všetkých hĺbkových snímok v dátovej sade prijíma ako argument zoznam filtrov a postupne na všetky CSV súbory aplikuje všetky filtrácie. Funkcia uloží filtrovanú anotáciu každej hĺbkovej snímky do novo vytvoreného CSV súboru.

Príkladom použitia filtrácie je odfiltrovanie chybných symbolov z anotácie. Ide o symboly, ktoré sa buď na danom mieste vôbec nevyskytujú, ako je znázornené na obrázku 5.7, alebo je nesprávne určená ich trieda. V oboch prípadoch bolo možné tieto symboly odfiltrovať na základe nízkej hodnoty v stĺpci 'Correlation' CSV súbora s anotáciou (pozri prílohu B).



Obr. 5.7: Chybná anotácia - na danom mieste sa nenachádza žiaden symbol.

Pri anotovaní dátovej sady bol využitý aj filter na odfiltrovanie symbolov, ktoré sú súčasťou súvislých nápisov a majú nesprávny kód, ako bolo spomenuté v podsekcií 5.1.2. Symboly tvoriace jeden takýto nápis bolo možné identifikovať na základe toho, že mali rovnakú hodnotu v stĺpci Complex glyph index. Symboly boli filtrované do zvláštneho súboru a nanovo anotované.

Z anotácií boli odstránené aj symboly, ktoré boli na hĺbkovej snímke rozdelené hranicou obrázka – ich časť sa nachádzala na pravom okraji obrázka a druhá časť na ľavom okraji.

5.1.4 Získanie informácií o dátovej sade

Priložený zdrojový kód tiež obsahuje funkcie, ktorých výstupom sú celkové informácie o dátovej sade, ako je napríklad frekvencia jednotlivých druhov symbolov a maximálna šírka symbolu. Informácia o maximálnej šírke symbolu bola použitá pri určení veľkosti prekryvu medzi jednotlivými časťami obrázka vstupujúcimi do neurónovej siete. Zastúpenie jednotlivých typov symbolov v trénovacej a testovacej množine bolo graficky znázornené využitím knižnice Matplotlib.

5.1.5 Rozdelenie hĺbkových snímok a ich anotácií na časti

Funkcia, ktorá rozdeľuje hĺbkové snímky a ich anotácie na časti, prijíma ako argumenty šírku podsnímky a veľkosť prekryvu medzi susediacimi podsnímkami. Pre každú vzniknutú podsnímku je nutné vytvoriť anotáciu na základe anotácie pôvodnej snímky a polohy podsnímky v pôvodnej snímke. Prah, ktorý je ďalším argumentom funkcie, určuje, aká časť symbolu sa musí nachádzať v podsnímke, aby bol symbol zaradený do anotácie podsnímky[2]. Funkcia upraví masku symbolu, aby zodpovedala len časti symbolu nachádzajúcej sa v podsnímke, a prepočíta polohy symbolov v anotácii vzhľadom k novému súradnicovému systému, ktorého počiatkom je začiatok podsnímky. Funkcia na záver uloží podsnímku, masky symbolov aj anotáciu podsnímky na disk do adresára podsnímky, v ktorého názve je obsiahnutá informácia o x-ovej súradnici začiatku podsnímky, neskôr využitá pri inferencii.

Funkcia tiež umožňuje vynechať pri rozdeľovaní hĺbkovej snímky tie podsnímky, ktoré neobsahujú žiadny symbol, a na týchto podsnímkach netrénovať model. Experimentovaním som ale zistil, že aj snímka bez symbolov obsahuje pre model cennú informáciu o neexistencií symbolu, preto boli pri trénovaní použité všetky podsnímky.

5.2 Metóda Mask R-CNN

Na experimentovanie s metódou Mask R-CNN bol použitý framework *Pytorch*. Knižnica *torchvision*, ktorá je súčasťou Pytorchu, obsahuje triedu `MaskRCNN` implementujúcu model Mask R-CNN. Táto knižnica tiež umožňuje použitie predtrénovaných váh či už pre celý model, alebo pre jeho úvodnú konvolučnú časť slúžiacu na extrakciu príznakov.

Pri používaní modelu Mask R-CNN som vychádzal z tutoriálu na webovej stránke frameworku Pytorch [43].

5.2.1 Načítanie dát do modelu

Na transformáciu podsnímok a ich anotácií do formátu požadovaného triedou `MaskRCNN` bola implementovaná trieda `MaskRCNN_dataset` dediaca od štandardnej triedy `Dataset`

[2]Pri trénovaní aj vyhodnotení modelu boli do anotácie podsnímky zaradené len symboly, ktoré sa celou plochou nachádzali v podsnímke

frameworku Pytorch. Trieda `MaskRCNN_dataset` pri požiadavke na konkrétnu vzorku z dátovej sady načíta podsnímku a jej anotáciu v CSV súbore a prostredníctvom knižníc *pandas*, *NumPy* a *torch* transformuje informácie o typoch symbolov, ich maskách a bounding boxoch do očakávaného tvaru. V prípade, že je použitá dynamická augmentácia, je na vstupný obrázok, masky a bounding boxy ešte v rámci `MaskRCNN_dataset` aplikovaná požadovaná transformácia.

Iná trieda dediaca od triedy `Dataset` bola použitá na transformáciu obrázkov a anotácií vytvorených umelým generátorom do správneho tvaru.

Štandardná trieda `DataLoader` frameworku Pytorch potom spojí N výstupov z triedy `MaskRCNN_dataset` do jedného batcha, kde N je veľkosť batcha. Výstup triedy `DataLoader` je vstupom modelu v každom kroku.

5.2.2 Dynamická augmentácia

Dynamická augmentácia bola implementovaná s využitím knižnice *Albumentations*. Výber použitých transformácií je opísaný v podsekcií 6.1.4. Všetky vybrané transformácie, aplikované s istou pravdepodobnosťou, boli zložené do jednej transformácie odohrávajúcej sa počas načítania dát do modelu.

5.2.3 Implementácia trénovacieho, validačného a testovacieho cyklu

Kód zodpovedný za trénovanie modelu, jeho validáciu a testovanie bol zapúzdrený do triedy `MaskRCNNModel` dediacej od triedy `LightningModule` frameworku *PytorchLightning*. PytorchLighnting pracuje nad frameworkom Pytorch, vynucuje jednotnú štruktúru modelov a zjednodušuje trénovanie. Triedy dediace od `LightningModule` reprezentujúce model musia implementovať metódy, ako napríklad `training_step`, `validation_step` a `test_step`, ktoré určujú chovanie modelu pri jednom trénovacom/validačnom/testovacom kroku. Podobné metódy určujú chovanie modelu na konci trénovacej/validačnej/testovacej epochy. Framework PytorchLightning zabezpečí volanie týchto metód v správny čas.

Ďalšou dôležitou metódou reimplementovanou v `MaskRCNNModel` je metóda `configure_optimizers`, ktorá nastaví optimalizačný algoritmus a prípadne aj plánovač rýchlosti učenia (learning rate scheduler) použitý pri trénovaní.

Pre trénovanie potom stačí vytvoriť inštanciu triedy `Trainer` frameworku PytorchLightning, predať jej parametre trénovania, a vyvolať jej metódu `fit` so zapuzdreným modelom ako argumentom.

PytorchLightning poskytuje možnosť nastaviť viacero parametrov trénovania. Jedným z nich je počet krokov pre prípadnú akumuláciu gradientu, pri ktorej sú parametre modelu aktualizované až po prechode viacerých batchov modelom[3].

Chovanie pri trénovaní je ďalej možné modifikovať nastavením callbackov. Použité callbacky zabezpečujú napríklad výpočet metrík počas validačnej epochy, skoré ukončenie trénovania či uloženie kontrolných bodov po každej epoche. Kontrolný bod obsahuje ako dovtedy natrénované parametre modelu, tak aj informácie o hodnotách sledovaných metrík a chýb, ktorých priebeh bol neskôr vizualizovaný a analyzovaný v nástroji *TensorBoard*.

5.2.4 Výpočet metrík

Na výpočet metrík average precision a average recall bola použitá knižnica *pycocotools*.

[3]Zdrojový kód obsahuje aj vlastnú implementáciu akumulácie gradientov.

Adresár `MaskRCNN/torchvision_scripts` obsahuje súbory prevzaté[4] z nedistribuovanej časti repozitára torchvision [42], ktoré boli použité na transformáciu dát z formátu aplikovaného v modeli Mask R-CNN v Pytorchi do tvaru očakávaného knižnicou pycocotools. Tieto súbory boli modifikované tak, aby výstupom vyhodnotenia bola aj informácia o hodnotách metrík pre jednotlivé triedy symbolov.

5.2.5 Inferencia a zloženie predikcií na podsnímkach

Kód implementujúci inferenciu na podsnímkach, zloženie predikcií pre podsnímky do predikcie pre celú hĺbkovú mapu, vizualizáciu predikovaných symbolov na hĺbkovej mape a prípadné vyhodnotenie metrík pre celú hĺbkovú mapu sa nachádza v súbore `MaskRCNN/inference.py` v triede `Inference`.

Pri inferencii na jednej hĺbkovej mape najprv prebehne predikcia na všetkých podsnímkach, na ktoré bola daná hĺbková mapa rozdelená. Následne prebehne algoritmus potlačenia nemaximálnych hodnôt, ktorý bol implementovaný na základe článku [33].

V ďalšom kroku môže voliteľne nasledovať vizualizácia predikovaných symbolov na hĺbkovej mape. Masky symbolov sú do pôvodnej hĺbkovej mapy dokreslené priehľadne, vďaka čomu zostávajú viditeľné aj pôvodné symboly a teda je možné vizuálne ohodnotiť kvalitu masky. Farba masky je daná typom symbolu.

Do obrázka môžu byť voliteľne dokreslené aj bounding boxy okolo detegovaného symbolu, informácia, o aký symbol ide, a skóre istoty predikcie. Obrázok môže byť zobrazený používateľovi alebo uložený do súboru.

Ďalšou možnosťou je uloženie informácií o detegovaných symboloch na disk. Informácie o symboloch sú ukladané v podobnom formáte, v akom boli pôvodné anotácie, teda masky symbolov sú uložené ako obrázok a ostatné informácie o všetkých symboloch detegovaných na hĺbkovej mape ako jeden CSV súbor. V CSV súbore je okrem informácií o type symbolu a jeho pozícií uložená aj miera istoty detekcie.

Trieda `Inference` tiež dokáže vypočítať COCO metriky pre celú pneumatiku berúc do úvahy len symboly po aplikácií potlačenia nemaximálnych hodnôt. Vtedy je nutné tejto triede zadať meno CSV súboru s anotáciou celej pneumatiky. Na transformovanie anotácií do formátu, s ktorým vedia pracovať triedy a funkcie na výpočet metrík z pycocotools, bola implementovaná trieda `MaskRCNN_evaluation_dataset` v rovnomennom súbore. Predikované masky symbolov aj masky symbolov z anotácie sú pred predaním do funkcií pycocotools komprimované kódovaním run-length encoding.

Implementácia zdvojenej segmentačnej vetvy

V implementácií Mask R-CNN vo frameworku Pytorch sa segmentačná vrstva skladá z troch častí, pričom všetky sú uložené v objekte `roi_heads` triedy `RoIHeads`, ktorý je atribútom triedy `MaskRCNN`:

- Z objektu uloženého v atribúte `mask_roi_pool`. Implicitne je týmto objektom objekt triedy `RoIAlign`, ktorá z výstupu extraktora príznakov pre každý navrhnutý región vyextrahuje mapu príznakov zadanej fixnej veľkosti.
- Z objektu uloženého v atribúte `mask_head`. Implicitne je ním objekt triedy `MaskRCNNHeads`, ktorá prijíma výstup z `mask_roi_pool` a aplikuje naň niekoľko plne kon-

[4] Prevzatie a prípadná modifikácia týchto súborov sú odporúčané v [43].

volučných vrstiev s jadrom 3x3 s rôznym počtom kanálov (nasledovaných jednotkou ReLU), zachovávajúcich priestorové rozmery spracovávaného tenzora.

- Z objektu uloženého v atribúte `mask_predictor`. Implicitne je ním objekt triedy `MaskRCNNPredictor`, ktorá prijíma výstup z `MaskRCNNHeads`, aplikuje naň transponovanú konvolúciu s jadrom 2x2, ReLU a na záver konvolúciu s jadrom 1x1, ktorej počet výstupných kanálov odpovedá počtu tried.

V rámci segmentačnej vrstvy sú postupne volané metódy `forward()` objektov uložených v `mask_roi_pool`, `mask_head` a `mask_predictor`, pričom výstup jedného objektu je vstupom nasledujúceho. Najjednoduchší spôsob implementácie zdvojenej segmentačnej vetvy (s čo najmenším zásahom do pôvodného zdrojového kódu Pytorchu) je vytvoriť triedy `MaskRCNNParallelRoiAlign`, `MaskRCNNParallelHeads` a `MaskRCNNParallelPredictor`, ktoré všetky budú obsahovať po dve inštancie zodpovedajúcich pôvodných tried. V metóde `forward()` potom tieto novo vytvorené triedy vyvolajú metódu `forward()` oboch inštancií a oba výstupy vrátia ako dvojicu, ktorú si nasledujúci objekt rozbalí a na prvý, resp. druhý prvok dvojice aplikuje metódu `forward()` prvej, resp. druhej inštancie, a tak ďalej.

Objekty týchto nových tried stačí potom nastaviť ako argumenty inicializátora modelu `MaskRCNN`. Z pohľadu kódu využívajúceho triedy `MaskRCNNParallelRoiAlign`, `MaskRCNNParallelHeads` a `MaskRCNNParallelPredictor` ide len o jeden prechod dát objektmi týchto tried, zdvojenosť segmentačnej vrstvy je implementovaná zdvojením inštancií pôvodných objektov vnútri týchto tried.

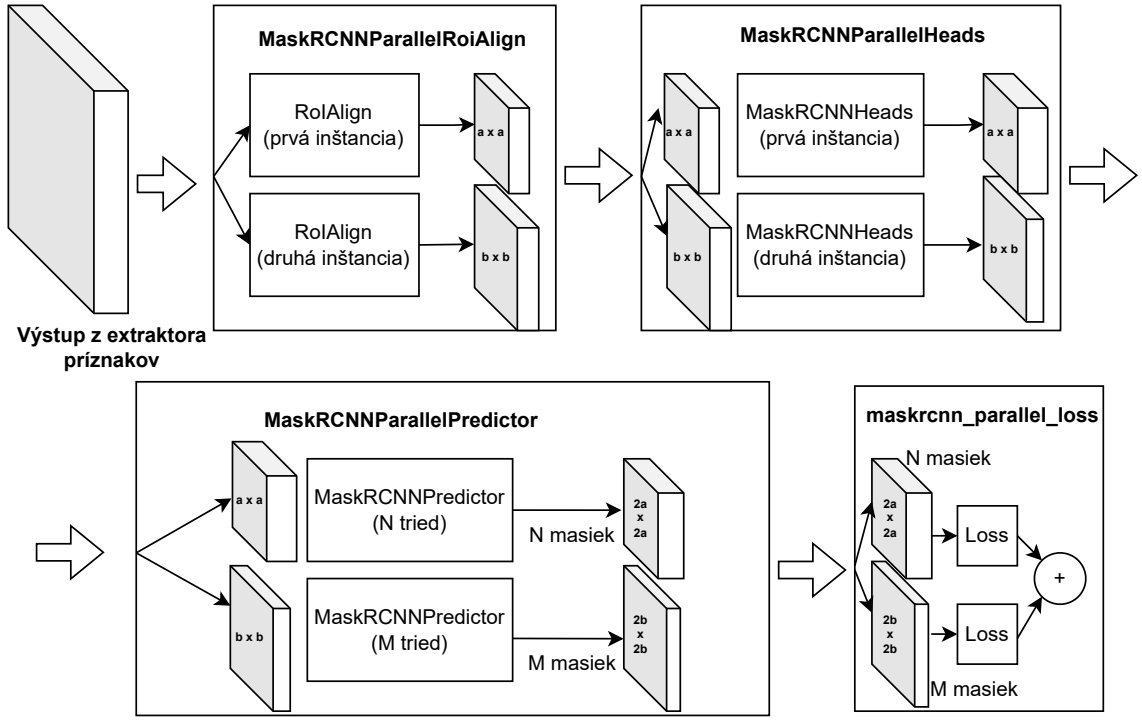
Tiež bolo potrebné implementovať chybovú funkciu `maskrcnn_parallel_loss()`, ktorá prijíma výstupnú dvojicu z `MaskRCNNParallelPredictor`, polohy predikovaných objektov, masky a triedy skutočných objektov a párovanie, ktoré každému predikovanému objektu priradí najpravdepodobnejší skutočný objekt hľadiac len na jeho pozíciu. Funkcia namapuje skutočné masky na predikované objekty podľa mapovania tak, aby sa veľkosť skutočnej masky rovnala veľkosti výstupu z tej segmentačnej vetvy, v ktorej sú počítané masky triedy zodpovedajucej skutočnej triede objektu. Následne je pre obe vetvy vypočítaná binárna vzájomná entropia medzi predikovanými a skutočnými maskami. Výsledná chyba je daná váženým súčtom chýb z oboch vetiev, pričom váhy chýb sú dané pomerom počtu predikovaných objektov patriacich do danej vetvy k počtu všetkých predikovaných objektov.

Schematické znázornenie implementácie zdvojenej segmentačnej vrstvy je na obrázku 5.8.

5.3 Metóda U-Net

Pri experimentovaní s metódou U-Net bola využitá knižnica *Segmentation Models Pytorch* [48], ktorá obsahuje implementáciu viacerých modelov pre sémantickú segmentáciu, vrátane možnosti použitia predtrénovaných váh. Tiež sú v nej implementované najčastejšie používané metriky a chybové funkcie pre sémantickú segmentáciu.

Podobne ako pri Mask R-CNN, aj v prípade metódy U-Net bolo treba implementovať triedu (dediacu od triedy `Dataset` Pytorchu), ktorá transformuje hĺbkové mapy a ich anotácie do formátu prijímaného modelom. Architektúra U-Net kvôli spojeniu máp príznakov z podvzorkovacej a nadvzorkovacej časti vyžaduje, aby rozmery vstupného obrázka boli deliteľné číslom 32. V opačnom prípade by sa priestorové rozmery máp príznakov vstupujúcich do spojenia (angl. concatenation) v nadvzorkovacej časti nezhodovali. Preto sú



Obr. 5.8: Schéma implementácie paralelnej segmentačnej vrstvy. Vytvorené pomocou nástroja diagrams.net [6].

vstupné obrázky aj masky prevzorkované do rozmerov deliteľných 32, a to tak, aby došlo k čo najmenšej interpolácií.

Samotný model bol, podobne ako pri Mask R-CNN, zapúzdrený do triedy odvodenej od triedy `LightningModule` frameworku PytorchLightning, a v metódach tejto triedy sa nachádza implementácia správania modelu pri trénovaní, validácií a testovaní.

Ďalej bolo implementované spojenie predikcií na podsnímkach pri inferencii do jednej snímky rozmerov pôvodnej hĺbkovej mapy. Pre celkovú predikciu bola implementovaná vizualizácia aj výpočet metrík, porovnávajúci predikciu s anotáciou.

Kapitola 6

Experimenty a ich vyhodnotenie

Táto kapitola najprv detailne popisuje a vyhodnocuje experimenty prevedené s metódou Mask R-CNN. Kapitola obsahuje aj stručný popis experimentov s metódou U-Net vykonaných v úvodných fázach riešenia práce. Všetky experimenty boli prevedené na firemnom počítači s pamäťou GPU o veľkosti 8 GB.

6.1 Experimenty s metódou Mask R-CNN

S modelom Mask R-CNN boli prevedené nasledujúce experimenty:

1. Experiment s predtrénovaným modelom. Váhy v celom modeli sú predtrénované na dátovej sade COCO. Cieľom bolo využiť predtrénované váhy na rýchlejšie učenie a zlepšenie generalizácie modelu.

2. Experiment s predtrénovaným extraktorom príznakov. Váhy v rámci použitého extraktora príznakov sú predtrénované na dátovej sade ImageNet. Cieľom bolo využiť predtrénované váhy v extraktore príznakov na rýchlejšie učenie a zlepšenie generalizácie.

3. Experiment s použitím dynamickej augmentácie. Cieľom bolo obohatiť dáta vstupujúce do modelu a tým zabezpečiť väčšiu schopnosť modelu generalizovať.

4. Experiment s väčším počtom pomerov strán referenčných boxov v RPN. Cieľom bolo zlepšiť detekciu symbolov s nerovnomerným pomerom strán.

5. Experiment s väčším rozmerom predikovanej masky pre všetky symboly. Cieľom bolo zlepšiť segmentáciu symbolov s tenkými líniami.

6. Experiment s paralelnou segmentačnou vrstvou. Masky symbolov s nízkou kvalitou segmentácie sú predikované s väčším rozlíšením v separátnej vetve. Cieľom bolo ešte viac zlepšiť segmentáciu symbolov s tenkými líniami za prijateľných pamäťových nárokov.

Všetky experimenty od tretieho (vrátane) používajú predtrénovaný model a dynamickú augmentáciu. Konkrétny popis použitých metrík, jednotlivých experimentov a ich vyhodnotenie sa nachádza v nasledujúcich podsekciách.

6.1.1 Vyhodnocované metriky

Vyhodnocované metriky pri experimentoch s Mask R-CNN boli až na nižšie uvedené výnimky totožné s metrikami používanými pre dátovú sadu COCO. Použité metriky boli dvoch typov:

- Metriky pre detekciu, ktoré pri výpočte IoU brali do úvahy bounding boxy objektov.
- Metriky pre segmentáciu, ktoré pri výpočte IoU brali do úvahy masky objektov.

V ďalšom texte bude $AP[t]$ značiť hodnotu average precision počítanú s IoU prahom t a $AP[x:s:y]$ značiť priemer hodnôt $AP[t]$ pre $t = x, x+s, x+2s, \ldots, y$. Metrika average recall bude označená až na použitie skratky AR rovnako.

Objekty sú v dátovej sade COCO podľa veľkosti (plochy bounding boxu) delené na malé objekty s plochou menšou než 32^2 pixelov (v mnou získanej dátovej sade interpunkčné znamienka a písmená 'i', 'I', 'l'), stredne veľké objekty s plochou medzi 32^2 a 96^2 pixelov (písmená a čísla ako súčasť varovaní a informácií o pneumatike) a veľké objekty s plochou väčšou než 96^2 pixelov (veľké nápisy identifikujúce výrobcu alebo sériu). Metriky pre malé, stredné a veľké objekty budú v ďalšom texte značené ako AP_s/AR_s, AP_m/AR_m, AP_l/AR_l. Pri testovaní boli vyhodnocované nasledovné metriky:

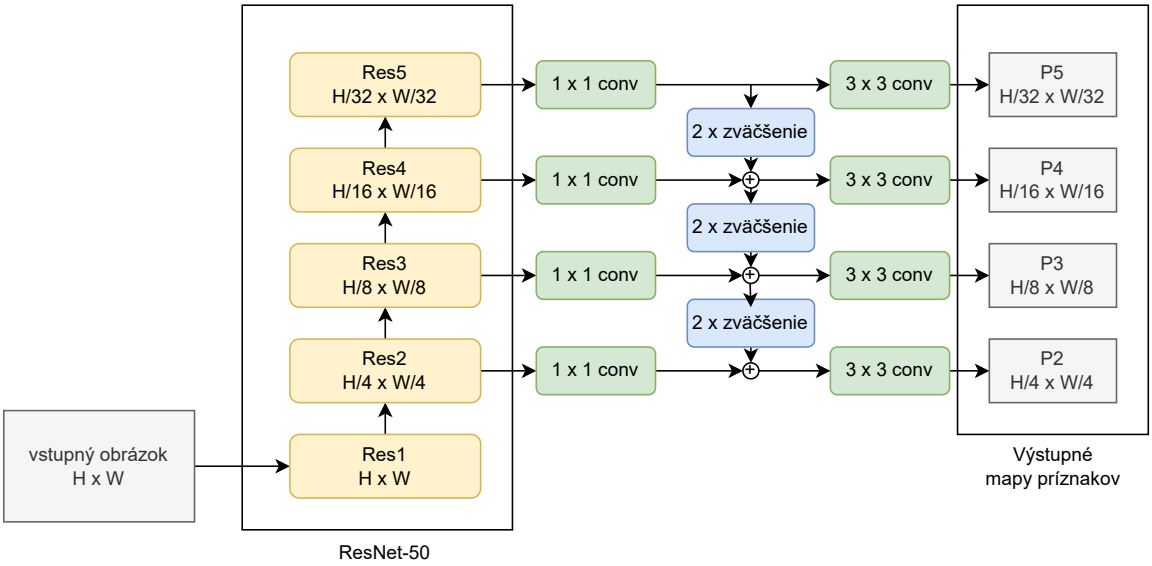
- AP/AR $[0,5:0,05:0.95]$
- AP/AR $[0,5]$,
- AP/AR $[0,75]$,
- AP_s/AR_s $[0,5:0,05:0.95]$,
- AP_m/AR_m $[0,5:0.05:0.95]$,
- AP_l/AR_l $[0,5:0,05:0.95]$,

kde zápis AP/AR značí, že na daných prahoch boli vyhodnocované obe tieto metriky. Metriky boli vyhodnocované vo variante pre detekciu aj vo variante pre segmentáciu. Narozdiel od dátovej sady COCO bolo obmedzenie na maximálny počet detekcií nastavené na 600, čo je viac než je maximálny počet symbolov na pneumatike. Metrika AP bola vo variante pre detekciu aj segmentáciu na všetkých troch IoU prahoch počítaná aj pre jednotlivé typy symbolov.

6.1.2 Spoločné vlastnosti experimentov s Mask R-CNN

Pred prevedením experimentov s metódou Mask R-CNN boli vstupné hĺbkové mapy rozdelené na podsnímky so šírkou 2500 pixelov a prekryvom 2000 pixelov medzi susediacimi podsnímkami. Tieto hodnoty boli zvolené na základe toho, že najširší symbol dátovej sady mal šírku 1890 pixelov. Hodnota prekryvu väčšia než šírka najširšieho symbolu zabezpečí, že každý symbol sa prinajmenšom na jednej podsnímke bude nachádzať v celej svojej šírke.

Hoci v úvodnej fáze riešenia problému boli prevedené experimenty aj s menšou sieťou *ResNet-18*, v experimentoch popísaných v tejto sekcií je ako sieť na extrakciu príznakov (chrbticová sieť, anglicky backbone) zhodne použitá sieť *ResNet-50*, ktorá produkovala lepšie výsledky. Konkrétne bola použitá jej varianta so sieťou pyramíd príznakov (pozri podsekciu 3.2.3), ktorá pri použití v Mask R-CNN podľa [10] vykazuje lepšie výsledky v porovnaní s klasickou sieťou ResNet-50. Schematické znázornenie siete ResNet-50 vo variante s pyramídou príznakov je na obrázku 6.1.



Obr. 6.1: Schéma siete ResNet-50 vo variante s pyramídou príznakov. Žlté bloky zobrazujú 5 etáp siete ResNet-50 a rozmery výstupu jednotlivých etáp. Sivé bloky zobrazujú výstupné mapy príznakov. Schéma bola vytvorená pomocou nástroja diagrams.net [6] na základe podobnej schémy v [19].

Za optimalizačný algoritmus bol zvolený *stochastic gradient descent* (SGD) s hodnotou hyperparametra momentum 0,9 a hodnotou útlmu váhy (anglicky weight decay) 0,0005. Hodnota rýchlosti učenia (learning rate) bude popísaná pri jednotlivých experimentoch. V úvodných experimentoch boli vyskúšané aj iné optimalizačné algoritmy, ale SGD sa ukázal ako najvhodnejší.

Kvôli pamäťovým obmedzeniam sa batch skladal len z jedného obrázka. Bola ale použitá akumulácia gradientu po 16 batchoch, efektívna veľkosť batcha bola teda 16, rovnako ako v experimentoch v článku o Mask R-CNN.

Trénovanie prebiehalo minimálne 15 a maximálne 30 epoch. Po každej trénovacej epoche prebehla validačná epocha vyhodnocujúca sledované metriky na testovacej množine, pričom pre rýchlosť a jednoduchosť vyhodnotenie prebiehalo na podsnímkach (bez skladania predikcií pre podsnímky jednej pneumatiky a použitia potlačenia nemaximálnych hodnôt). V prípade, ak sa v troch po sebe idúcich epochách nezlepšila hodnota metriky $AP[0,5:0,05:0,95]$ pre segmentáciu, rýchlosť učenia bola 10-násobne zmenšená[1]. Ak sa táto metrika nezlepšila v piatich po sebe idúcich epochách, trénovanie bolo zastavené.

Po skončení každej trénovacej epochy boli aktuálne natrénované váhy modelu uložené na disk. Za najlepší model bol po ukončení trénovania považovaný výstup tej trénovacej epochy, po skončení ktorej prebehla validačná epocha s najmenšou dosiahnutou hodnotou $AP[0,5:0,05:0,95]$ pre segmentáciu. Pri testovaní nad testovacou množinou boli predikcie tohto modelu na jednotlivých podsnímkach každej hĺbkovej snímky zložené algoritmom potlačenia nemaximálnych hodnôt do celkových predikcií, nad ktorými prebiehal výpočet metrík. Metriky vypočítané pri testovaní tak vykazovali podľa očakávania lepšie hodnoty ako metriky pri validácií, keďže algoritmus potlačenia nemaximálnych hodnôt zahodil predikcie s menšou mierou istoty.

[1]Hodnota rýchlosti učenia dosiahnutá týmto zmenšovaním nemohla byť nižšia ako 0,0005

Zároveň s výpočtom metrík boli predikcie uložené na disk vo formáte podobnom formátu anotácií a takisto bola uložená vizualizácia predikcií, ktorá bola neskôr predmetom skúmania a hľadania nedostatkov.

6.1.3 Experimenty s predtrénovanou sieťou alebo časťou siete

Prvým navrhnutým experimentom bolo porovnať model, ktorý bol celý predtrénovaný na dátovej sade COCO pre úlohu inštančnej segmentácie, s nepredtrénovaným modelom, ktorého chrbticová sieť (extraktor príznakov) resnet50 bola ale predtrénovaná na dátovej sade ImageNet.

Použitie predtrénovanej chrbticovej siete má zmysel, pretože táto sieť sa dokáže (najmä v prvých vrstvách) naučiť extrahovať príznaky typické pre všetky objekty, ako sú napríklad hrany. Váhy naučené na inej dátovej sade teda môžu byť použité ako prvotné hodnoty pre trénovanie. V oboch prípadoch boli prvé dve z piatich vrstiev chrbticovej siete zmrazené a ich parametre sa už pri trénovaní neaktualizovali.

Pri predtrénovanom modeli bola rýchlosť učenia nastavená na hodnotu 0,015. Pri nepredtrénovanom modeli s predtrénovanou chrbticovou sieťou bola rýchlosť učenia nastavená na vyššiu hodnotu 0,02. Myšlienkou za tým bolo, že predtrénovaná sieť nepotrebuje meniť hodnoty váh tak rýchlo ako náhodne inicializovaná sieť. Kvôli numerickej stabilite trénovania bolo ale nutné použiť pri nepredtrénovanej sieti orezanie gradientu (gradient clipping) na maximálnu hodnotu 10.

Vývoj metriky $AP[0,5:0,05:0,95]$ pre detekciu počas trénovania oboch modelov je znázornený v grafe na obrázku 6.2. Je vidieť, že v úvodných epochách bola hodnota metriky pri predtrénovanom modeli výrazne vyššia a hoci sa postupne rozdiel znižoval, aj po skončení trénovania dosahoval model inicializovaný predtrénovanými váhami lepšie výsledky.

Tabuľka 6.1 ukazuje dosiahnuté hodnoty metrík $AP[0,5:0,05:0,95]$ pre detekciu a segmentáciu oboch modelov. Predtrénovaný model dosiahol lepšie výsledky vo všetkých 24 sledovaných metrikách, preto som sa rozhodol vo všetkých ďalších experimentoch vychádzať z predtrénovaného modelu.

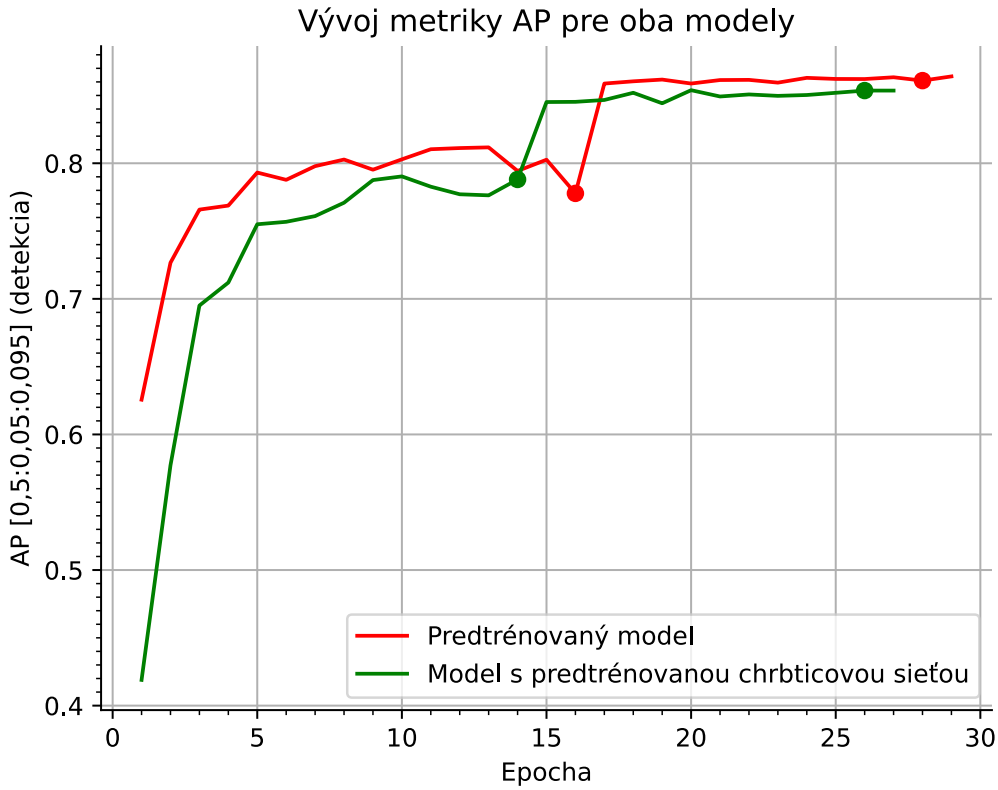
Metrika	Predtrénovaný model	Predtrénovaná chrbtica
$AP[0,5:0,05:0,95]$ pre detekciu	**0,876**	0,867
$AP[0,5:0,05:0,95]$ pre segmentáciu	**0,724**	0,709

Tabuľka 6.1: Tabuľka zobrazuje metriky dosiahnuté pri použití predtrénovaného modelu a modelu s predtrénovanou chrbticou.

6.1.4 Experiment s dynamickou augmentáciou

Charakter dátovej sady a riešenej úlohy výrazne obmedzoval možné zmysluplné druhy augmentácie. Použité transformácie totiž museli spĺňať nasledovné vlastnosti:

- Transformácia aplikovaná na obrázok musela mať svoju obdobu aj pre bounding boxy a masky.



Obr. 6.2: Graf zobrazuje vývoj metriky AP pre detekciu pri validácií počas trénovania predtrénovaného modelu a modelu s predtrénovanou chrbticovou sieťou. Tučnými bodmi je znázornené, kedy došlo k zníženiu rýchlosti učenia.

- Keďže početnosť niektorých typov symbolov v dátovej sade bola veľmi nízka, transformácia by nemala zahadzovať symboly alebo spôsobiť, že symboly nebudú v transformovanom obrázku v celom rozmere.
- Obrázok po aplikovaní transformácie musí byť zmysluplný a nemal by obsahovať objekty a javy, ktoré v pôvodnej dátovej sade neboli.

Transformácie, často používané v oblasti detekcie, ako napríklad zrkadlové otočenie, rotácia, zmena veľkosti alebo výber obdĺžnikového výseku mohli byť teda použité len v obmedzenej miere alebo vôbec. Vzhľadom na to, že dátová sada pozostáva zo snímok v odtieňoch sivej, ani operácie pracujúce s farebnou škálou obrázka nemali veľký zmysel.

Nakoniec bola zvolená augmentácia zložená z troch častí:

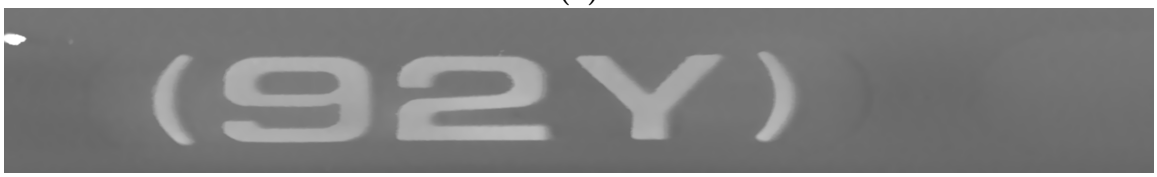
1. V prvom kroku bol s pravdepodobnosťou 60% aplikovaný výber obdĺžnikového výseku garantujúci, že výsek bude plne obsahovať všetky objekty pôvodného obrázka. S pravdepodobnosťou 20% bola aplikovaná operácia zväčšenia a rotácie, pričom zväčšenie mohlo byť maximálne 1,1-násobné v porovnaní s pôvodným obrázkom a limit pre rotáciu bol 1 stupeň. S pravdepodobnosťou 20% nebola v prvom kroku aplikovaná žiadna transformácia.

2. V druhom kroku bol s pravdepodobnosťou 25% aplikovaný blur obrázka s veľkosťou jadra 3x3, s pravdepodobnosťou 25% gaussovský blur s jadrom 3x3, a s pravdepodobnosťou 50% nebola aplikovaná žiadna transformácia.

3. V treťom kroku bol s pravdepodobnosťou 75% aplikovaný gaussovský šum s hodnotou rozptylu 1[2].

Je zrejmé, že až na výber obdĺžnikového výseku ide o veľmi jemnú augmentáciu. Príklad podsnímky pred a po augmentácií je na obrázku 6.3. Na overenie, aký vplyv má takáto dynamická augmentácia, bol prevedený experiment využívajúci predtrénovanú sieť, kde jediný rozdiel oproti trénovaniu predtrénovanej siete opísanému vyššie spočíval v použití augmentácie pred načítaním dát do modelu.

(a)



(b)

Obr. 6.3: Vstupný obrázok **(a)** pred a **(b)** po augmentácií.

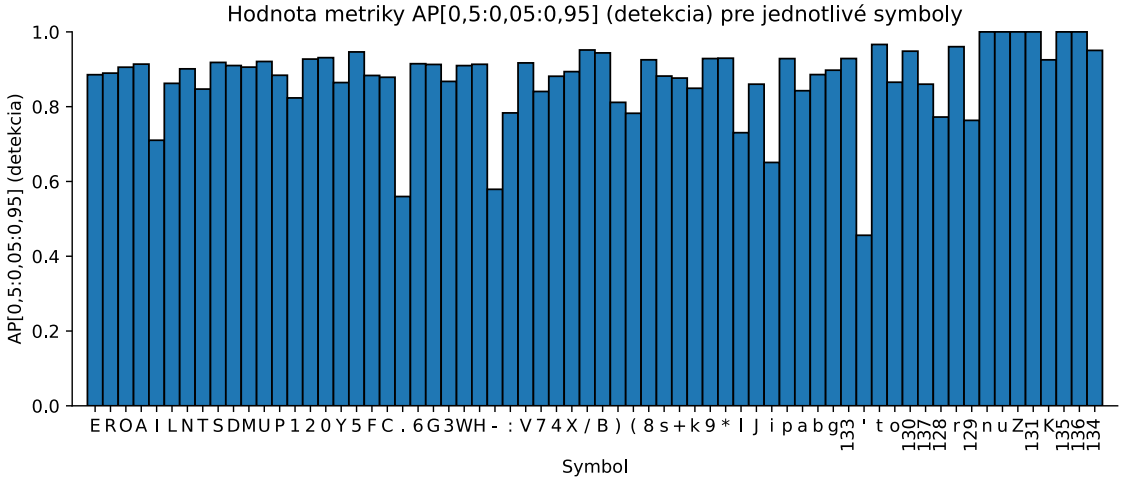
Z výsledných metrík vyplynulo, že takáto augmentácia má na schopnosť modelu učiť sa zanedbateľný vplyv. Metrika $AP[0,5:0,05:0,95]$ pre detekciu klesla o jednu stotinu na 0,875, hodnota $AP[0,5:0,05:0,95]$ pre segmentáciu zostala na hodnote 0,724. Rozdiely vo všetkých ostatných metrikách boli tiež minimálne.

6.1.5 Analýza kvality detekcie a segmentácie pre jednotlivé druhy symbolov

Táto podsekcia obsahuje analýzu metrík spočítaných pre jednotlivé druhy symbolov, ktoré boli dosiahnuté trénovaním predtrénovaného modelu s použitím dynamickej augmentácie.

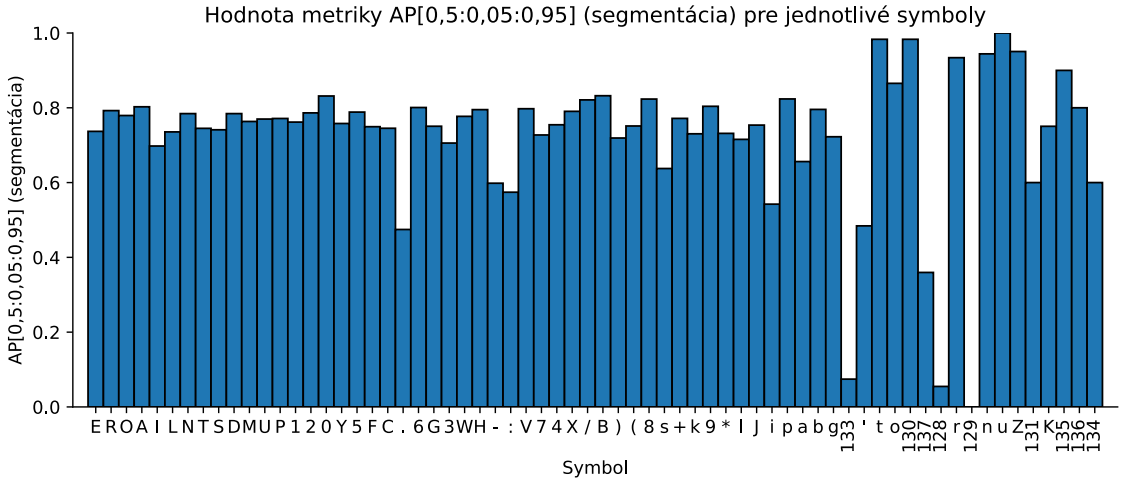
[2]Hodnoty pixelov v rámci jedného symbolu alebo v rámci pozadia boli v dátovej sade väčšinou veľmi vyhladené, väčšia hodnota rozptylu pre šum preto produkovala obrázky nepodobné tým z dátovej sady.

Histogram na obrázku 6.4 ukazuje hodnotu metriky $AP[0,5:0,05:0,95]$ pre detekciu. Symboly sú v histograme zoradené podľa ich početnosti v dátovej sade. Podľa očakávania dosahujú najmenšiu presnosť detekcie symboly bodka, pomlčka a apostrof, ktoré majú veľmi malú plochu, podobajú sa na náhodnú chybu na pneumatike a niekedy je náročné ich rozpoznať aj voľným okom. Nízku presnosť detekcie majú aj symboly 'I', 'i', 'l', '(', ')', ':', ktorých spoločnou vlastnosťou je veľký pomer výšky ku šírke. Zaujímavým pozorovaním je, že presnosť detekcie sa zdá byť nezávislá od frekvencie symbolu v dátovej sade.



Obr. 6.4: Metrika $AP[0,5:0,05:0,95]$ pre detekciu pre jednotlivé typy symbolov dosiahnutá v experimente s dynamickou augmentáciou.

Kvalita segmentácie, znázornená na obrázku 6.5, samozrejme závisí na kvalite detekcie. Najvýraznejším záverom vyplývajúcim z histogramu je nízka kvalita segmentácie špeciálnych symbolov, najmä však tých s kódmi 133, 137, 128 a 129, ktorých spoločnou vlastnosťou je tenkosť línií, členitosť a prípadne malá veľkosť komponentov, ktoré ich tvoria.



Obr. 6.5: Metrika $AP[0,5:0,05:0,95]$ pre segmentáciu pre jednotlivé typy symbolov dosiahnutá v experimente s dynamickou augmentáciou.

Na základe tejto analýzy boli navrhnuté ďalšie experimenty, popísané v nasledujúcich podsekciách.

6.1.6 Experiment s väčším počtom pomerov strán referenčných boxov

Na riešenie problému s detekciou symbolov s nerovnomerným pomerom strán, popísaného v predchádzajúcej podsekcií, bol navrhnutý experiment, v ktorom sieť na návrh regiónov používa referenčné boxy s viacerými pomermi strán.

Prednastavené pomery strán referenčných boxov v Pytorchi, zhodujúce sa s pomermi strán v [10], sú 1:2, 1:1 a 2:1. V tomto experimente boli pridané ďalšie dva pomery strán 1:4 a 4:1, ktoré mali uľahčiť detekciu symbolov s nerovnomerným pomerom strán ako 'I' či '-'. Pridanie ďalších pomerov strán však znamenalo, že v časti generujúcej referenčné boxy nemohli byť použité predtrénované váhy.

Ako vidieť v tabuľke 6.2, pridanie pomerov strán 1:4 a 4:1 neviedlo k lepšej detekcií symbolov s nerovnomerným pomerom strán. Presnosť detekcie symbolu s kódom 129 dokonca klesla na nulu. Klesla aj presnosť detekcie spriemerovaná pre všetky triedy, a to z 0,875 na 0,856.

	$AP[0,5:0,05:0,95]$ pre detekciu	
Symbol	3 pomery strán	5 pomerov strán
'('	**0,782**	0,777
')'	**0.812**	0,787
'-'	0,579	**0,627**
'I'	**0,710**	0,707
'i'	**0,651**	0,633
'l'	0,730	**0,749**
129	**0,763**	0,000

Tabuľka 6.2: Tabuľka zobrazuje metriky pre symboly s nerovnomerným pomerom strán pri použití 3 a 5 pomerov strán referenčných boxov.

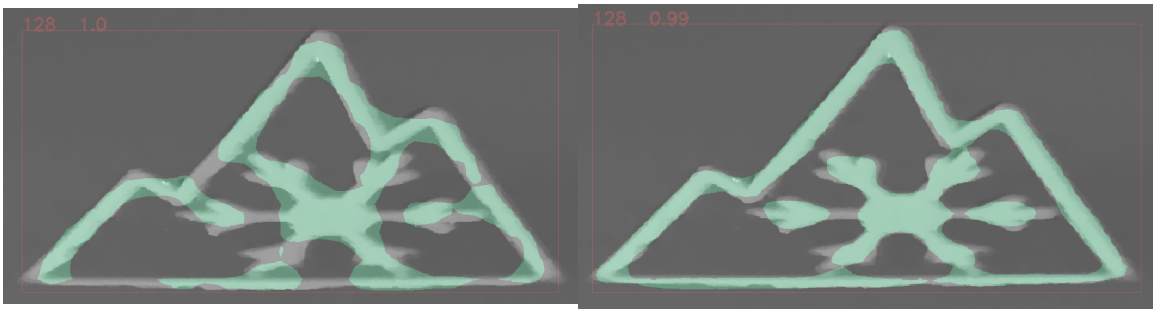
6.1.7 Experiment s väčším rozmerom predikovanej masky pre všetky symboly

Ako bolo uvedené v podsekcií 6.1.5, symboly s najnižšou kvalitou segmentácie boli špecifické tenkými líniami a členitosťou. Predpokladaným dôvodom nízkej kvality segmentácie týchto symbolov bolo malé rozlíšenie výstupnej masky. Preto som navrhol experiment, v ktorom predikovaná maska má väčšie rozmery, čo je zabezpečené nastavením väčších želaných rozmerov výstupu operácie RoIAlign.

Rovnica 3.9 na základe veľkosti regiónu navrhnutého sieťou RPN udáva, z ktorej vrstvy siete pyramíd príznakov sú pre daný región extrahované príznaky. V článku [10] aj vo frameworku Pytorch majú konštanty k_0 a s v rovnici 3.9 hodnoty 4, respektíve 224. Dôvod je taký, že k-tá vrstva siete pyramíd príznakov podvzorkuje obrázok 2^k-násobne. Teda ak je kanonická veľkosť s vstupného obrázku 224 a k je 4, potom obrázok má na výstupe k-tej vrstvy veľkosť strany $s/2^k = 224/16 = 14$, teda operácia RoIAlign s požadovanou veľkosťou strany výstupu 14 nemusí aplikovať žiadnu interpoláciu. Pri experimente s požadovaným

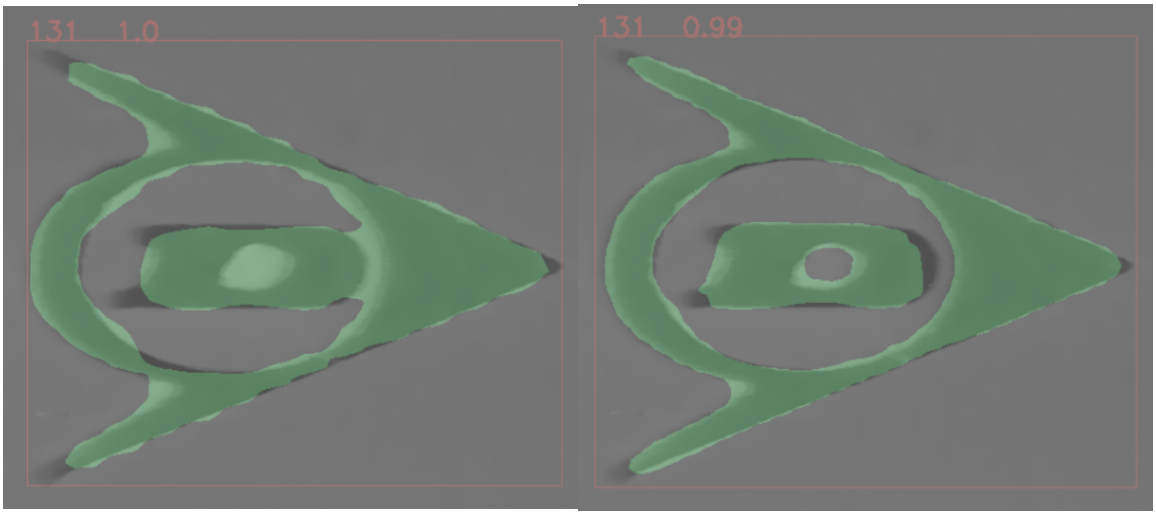
výstupom vrstvy RoIAlign pre segmentáciu o veľkosti 28 x 28 bolo teda nutné pozmeniť hodnotu k_0 na 3, aby pri operácií RoIAlign dochádzalo k čo najmenšej interpolácií.

Hodnota $AP[0,5:0,05:0,95]$ pre segmentáciu takto upraveného modelu stúpla z 0,724 na 0,738. Dokonca došlo aj k drobnému zvýšeniu $AP[0,5:0,05:0,95]$ pre detekciu (na 0,877). Ako vidieť v tabuľke 6.3, zlepšila sa segmentácia problémových symbolov s kódmi 128, 131, 134, 136 a 137. Ukážka masiek symbolov s kódmi 128 a 131 pred a po použití väčšieho výstupu segmentácie je na obrázkoch 6.6 a 6.7. Kvalitu segmentácie symbolov s kódmi 129 a 133 sa zlepšiť nepodarilo.



(a) **(b)**

Obr. 6.6: Ukážka predikovanej masky symbolu s kódom 128 **(a)** pred a **(b)** po použití väčšieho výstupu segmentácie.



(a) **(b)**

Obr. 6.7: Ukážka predikovanej masky symbolu s kódom 131 **(a)** pred a **(b)** po použití väčšieho výstupu segmentácie.

6.1.8 Experiment s paralelnou segmentačnou vrstvou

Ďalšie zväčšovanie veľkosti výstupu pre všetky triedy by už z pamäťových dôvodov na použitom počítači nebolo možné. Preto druhým navrhnutým riešením problému nízkej kvality segmentácie niektorých špeciálnych symbolov bolo vytvorenie zdvojenej segmentačnej

vrstvy, pričom v jednej segmentačnej vetve boli predikované masky problémových špeciálnych symbolov (konkrétne symbolov s kódmi 128, 129, 131, 133, 134, 136, 137) a v druhej masky všetkých ostatných. Masky problémových symbolov boli predikované s rozlíšením 112 x 112 pixelov, teda veľkosť výstupu RoIAlign bola 56 x 56 a kanonická úroveň k_0 z rovnice 3.9 mala hodnotu 2. Počet konvolučných vrstiev v prvej vetve aj ich hĺbka bola z pamäťových dôvodov znížená. Veľkosť výstupu druhej vetvy bola ponechaná na implicitnej hodnote 14 x 14.

V tabuľke 6.3 je vidieť, že vytvorenie paralelnej segmentačnej vetvy s väčšou veľkosťou výstupu výrazne zlepšilo kvalitu masiek najmä tých symbolov, u ktorých sa dovtedy blížila nule. Hodnoty $AP[0,5:0,05:0,95]$ pre predikciu, resp. segmentáciu dosiahnuté v tomto experimente boli 0.876, resp. 0.729. Obrázok 6.8 ukazuje segmentáciu symbolu s kódom 129, ktorého predikovaná maska bola bez použitia zdvojenej segmentačnej vetvy prázdna.

$AP[0,5:0,05:0,95]$ pre segmentáciu

Symbol	zákl. model	s väčším výstupom segmentácie	so zdvojenou segm. vetvou
128	0,055	0,510	**0,549**
129	0,0	0,0	**0,289**
131	0,6	**0,8**	0,7
133	0,074	0,045	**0,412**
134	0,6	**0,8**	0,7
136	0,8	**0,9**	0,8
137	0,360	**0,428**	0,328

Tabuľka 6.3: Tabuľka zobrazuje metriku $AP[0,5:0,05:0,95]$ pre segmentáciu základného modelu (z experimentu s dynamickou augmentáciou), modelu so zväčšeným výstupom segmentácie pre všetky triedy a modelu s použitím zdvojenej segmentačnej vetvy.

Obr. 6.8: Ukážka segmentácie symbolu s kódom 129 s použitím zdvojenej segmentačnej vetvy. Bez jej použitia bola predikovaná maska symbolu prázdna.

6.2 Experimenty s metódou U-Net

Alternatívou k sieti Mask R-CNN, ktorá bola v rámci práce vyskúšaná, bola metóda U-Net, tá sa však už od úvodných experimentov javila ako výrazne menej vhodná na problém identifikácie symbolov.

Experimenty boli prevedené s variantou U-Netu, ktorá v podvzorkovacej časti používa sieť ResNet-18 alebo ResNet-50. Pamäťová náročnosť modelu bola v porovnaní s Mask R-CNN vyššia, preto musela byť hĺbková snímka rozdelená na podsnímky so šírkou menšou

ako pri Mask R-CNN a s prekryvom menším, než je maximálna šírka symbolu v dátovej sade. Počet podsnímok bol teda väčší, z čoho pramení aj dlhšia doba trénovania.

V rámci prevedených experimentov boli vyskúšané rôzne rýchlosti učenia a rôzne verzie siete ResNet, napriek tomu model nedokázal prekonať problém nerovnováhy medzi jednotlivými triedami, keďže sa v najlepšom prípade naučil rozpoznávať len niektoré triedy s najväčšou frekvenciou, ako znázorňuje obrázok 6.9[3].

Obr. 6.9: Výsledok inferencie pri metóde U-Net. Je vidieť, že model sa naučil rozpoznávať len v dátovej sade najviac frekventované triedy (pozri histogram frekvencie symbolov na obrázku 4.2), a aj pri nich je segmentácia pixelov na hranách symbolov nepresná.

Hoci je pravdepodobné, že ďalšou optimalizáciou hyperparametrov a prípadným vyriešením problému nerovnováhy tried by bolo možné dosiahnuť lepších výsledkov, rozhodol som sa radšej sústrediť na experimentovanie s výrazne sľubnejšou metódou Mask R-CNN.

6.3 Celkové vyhodnotenie experimentov

Z prevedených experimentov dosiahol najlepšie hodnoty metrík model Mask R-CNN, ktorý predikoval väčšiu masku (56 x 56 pixelov) pre všetky triedy. Hodnoty všetkých metrík vypočítaných pre tento model sú v tabuľke 6.4. V tabuľke vidieť, že na IoU prahu 0,5 metriky AP aj AR pre detekciu aj segmentáciu presahujú hodnotu 0,95. Metriky dosiahnuté pre jednotlivé triedy a ukážky výstupu tohto modelu sú v prílohe E. Inferencia na všetkých podsnímkach, na ktorú bola jedna pneumatika rozdelená, spolu s následným potlačením nemaximálnych hodnôt trvali na použitom počítači približne 50 sekúnd.

	Detekcia		Segmentácia	
Metrika	AP	AR	AP	AR
AP/AR $[0,5:0,05:0,95]$	0,877	0,905	0,738	0,772
AP/AR $[0,5]$	0,979	0,984	0.952	0.961
AP/AR $[0,75]$	0,953	0,968	0.867	0.893
AP_s/AR_s $[0,5:0,05:0,95]$	0,396	0,430	0,358	0,395
AP_m/AR_m $[0,5:0,05:0,95]$	0,856	0,892	0,729	0,769
AP_l/AR_l $[0,5:0,05:0,95]$	0,938	0,951	0,827	0,839

Tabuľka 6.4: Tabuľka zobrazuje dosiahnuté metriky pri použití modelu s väčším rozlíšením predikovanej masky.

[3]Obrázok je výstupom modelu natrénovaného s hodnotou rýchlosti učenia 0,005, s veľkosťou batcha 1, akumuláciou gradientu po 4 batchoch a trénovaní po dobu 15 epoch. Chybovou funkciou bola Diceova chyba.

Zaujímavé výsledky produkoval aj model so zdvojenou segmentačnou vrstvou. V prípade, že cieľom je dosiahnuť aspoň určitú minimálnu kvalitu segmentácie pre všetky typy symbolov a zväčšenie výstupu segmentácie pre všetky triedy z pamäťových alebo iných dôvodov nie je vhodné, môže byť model so zdvojenou segmentačnou vrstvou dobrým riešením.

Kapitola 7

Záver

Cieľom tejto práce bolo na základe naštudovaných znalostí navrhnúť, implementovať a experimentálne overiť postup pre identifikáciu znakov na hĺbkovej snímke pneumatiky.

Práca začína analýzou prístupov k detekcií a rozpoznaniu znakov v obraze. Podrobne je opísaná metóda, ktorá je v súčasnej dobe aplikovaná vo firme ME-Inspection SK, spol. s r.o.. Ďalšia časť práce sa venuje neurónovým sieťam používaným pre úlohu segmentácie obrazu. Keďže vstupnými dátami boli 2D hĺbkové snímky, práca sa nezaoberá možnosťami analýzy 3D informácie. V tejto časti sú popísané architektúry Mask R-CNN a U-Net, s ktorými boli prevedené experimenty.

Metóda navrhnutá na riešenie problému identifikácie znakov na pneumatike pozostáva z rozdelenia širokej hĺbkovej snímky na časti, využitia siete Mask R-CNN na identifikáciu symbolov na týchto častiach, a zloženia čiastkových predikcií do celkovej predikcie prostredníctvom potlačenia nemaximálnych hodnôt. Na riešenie problému segmentácie symbolov s tenkými líniami alebo členitou maskou je navrhnuté použitie zdvojenej segmentačnej vrstvy, kde výpočet masiek pre triedy s komplikovanou maskou prebieha v separátnej vetve a na vyššom rozlíšení.

V rámci práce bola v spolupráci s firmou vytvorená dátová sada hĺbkových snímok pneumatík. Tá bola použitá na experimentálne vyhodnotenie navrhnutého postupu, ktorý bol implementovaný vo frameworku Pytorch. Experimenty sú zamerané najmä na rôzne varianty siete Mask R-CNN, ktorá je v porovnaní so sieťou U-Net výrazne menej citlivá na nerovnováhu tried.

Pri použití predtrénovaného modelu so zväčšenou veľkosťou výstupu segmentačnej vrstvy dosiahli metriky $AP[0,5 : 0,05 : 0,95]$ pre detekciu, resp. segmentáciu hodnôt 0,877, resp. 0,738. Aplikácia zdvojenej segmentačnej vrstvy sa ukázala ako vhodné a pamäťovo úsporné riešenie pre segmentáciu symbolov s komplikovanou maskou. Výhodou navrhnutého prístupu voči metóde použitej vo firme je to, že nevyžaduje šablónu symbolov na pneumatike, nevýhodou je naopak nižšia presnosť masky.

Veľkým benefitom pre prípadnú budúcu prácu by bolo získanie väčšej dátovej sady. Masky symbolov získané navrhnutou metódou by bolo ďalej možné upresniť použitím metódy PointRend [16], ktorá prijíma menší výstup segmentácie z Mask R-CNN a mapu príznakov z extraktora príznakov a vyprodukuje väčšiu masku, pričom pri jej výpočte nerobí predikcie na rovnomernej mriežke, ale len pre vybrané body na hraniciach objektu. Ďalšou nezodpovedanou otázkou je, akým spôsobom dotrénovať model pri pridaní nového symbolu alebo novej grafickej reprezentácie už známeho symbolu do dátovej sady.

Literatúra

[1] Aggarwal, K. *Transpose convolution explained for up-sampling images* [online]. Paperspace Blog, 9. apríla 2021 [cit. 2022-04-25]. Dostupné z: https://blog.paperspace.com/transpose-convolution/.

[2] Alajlan, N., El Rube, I., Kamel, M. S. a Freeman, G. Shape retrieval using triangle-area representation and dynamic space warping. *Pattern Recognition*. 2007, zv. 40, č. 7, s. 1911–1920. DOI: https://doi.org/10.1016/j.patcog.2006.12.005. ISSN 0031-3203. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0031320306005188.

[3] Albawi, S., Mohammed, T. A. a Al Zawi, S. Understanding of a convolutional neural network. In: *2017 International Conference on Engineering and Technology (ICET)*. 2017, s. 1–6. DOI: 10.1109/ICEngTechnol.2017.8308186.

[4] Convolutional Neural Networks (CNNs / ConvNets). *CS231N convolutional neural networks for visual recognition* [online]. [cit. 2022-04-25]. Dostupné z: https://cs231n.github.io/convolutional-networks/.

[5] *ConvTranspose2d* [online]. PyTorch 1.10 documentation, 2019 [cit. 2022-04-25]. Dostupné z: https://pytorch.org/docs/stable/generated/torch.nn.ConvTranspose2d.html.

[6] diagrams.net. *Diagram software and flowchart maker* [online]. [cit. 2022-04-25]. Dostupné z: https://www.diagrams.net/.

[7] Girshick, R. Fast R-CNN. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, s. 1440–1448. DOI: 10.1109/ICCV.2015.169.

[8] Girshick, R., Donahue, J., Darrell, T. a Malik, J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. November 2013. DOI: 10.1109/CVPR.2014.81.

[9] Gómez, R. *Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss, Logistic Loss, Focal Loss and all those confusing names* [online]. Raúl Gómez blog, 23. mája 2018 [cit. 2022-04-25]. Dostupné z: https://gombru.github.io/2018/05/23/cross_entropy_loss/.

[10] He, K., Gkioxari, G., Dollár, P. a Girshick, R. Mask R-CNN. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, s. 2980–2988. DOI: 10.1109/ICCV.2017.322.

[11] Hui, J. *MAP (mean average precision) for object detection* [online]. Medium, 7. marca 2018 [cit. 2022-04-25]. Dostupné z: https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173.

[12] Kabsch, W. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A*. Sep 1976, zv. 32, č. 5, s. 922–923. DOI: 10.1107/S0567739476001873. Dostupné z: https://doi.org/10.1107/S0567739476001873.

[13] Karaoglu, S., Fernando, B. a Tremeau, A. A Novel Algorithm for Text Detection and Localization in Natural Scene Images. In:. December 2010, s. 635–642. DOI: 10.1109/DICTA.2010.115.

[14] Kazmi, I. K., You, L. a Zhang, J. J. A Survey of 2D and 3D Shape Descriptors. In: *2013 10th International Conference Computer Graphics, Imaging and Visualization*. 2013, s. 1–10. DOI: 10.1109/CGIV.2013.11.

[15] Kazmi, W., Nabney, I., Vogiatzis, G., Rose, P. a Codd, A. An Efficient Industrial System for Vehicle Tyre (Tire) Detection and Text Recognition Using Deep Learning. *IEEE Transactions on Intelligent Transportation Systems*. 2021, zv. 22, č. 2, s. 1264–1275. DOI: 10.1109/TITS.2020.2967316.

[16] Kirillov, A., Wu, Y., He, K. a Girshick, R. B. PointRend: Image Segmentation As Rendering. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, s. 9796–9805.

[17] Krizhevsky, A., Sutskever, I. a Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. Red Hook, NY, USA: Curran Associates Inc., 2012, s. 1097–1105. NIPS'12.

[18] Lateef, F. a Ruichek, Y. Survey on semantic segmentation using deep learning techniques. *Neurocomputing*. 2019, zv. 338, s. 321–348. DOI: https://doi.org/10.1016/j.neucom.2019.02.003. ISSN 0925-2312. Dostupné z: https://www.sciencedirect.com/science/article/pii/S092523121930181X.

[19] Lee, I., Kim, D., Wee, D. a Lee, S. An Efficient Human Instance-Guided Framework for Video Action Recognition. *Sensors*. December 2021, zv. 21, s. 8309. DOI: 10.3390/s21248309.

[20] Lin, H., Yang, P. a Zhang, F. Review of Scene Text Detection and Recognition. *Archives of Computational Methods in Engineering*. Marec 2020, zv. 27, s. 433–454. DOI: 10.1007/s11831-019-09315-1.

[21] Lin, T.-Y., Dollár, P., Girshick, R. B., He, K., Hariharan, B. et al. Feature Pyramid Networks for Object Detection. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, s. 936–944.

[22] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S. E. et al. SSD: Single Shot MultiBox Detector. In: *ECCV*. 2016.

[23] Long, J., Shelhamer, E. a Darrell, T. Fully convolutional networks for semantic segmentation. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, s. 3431–3440. DOI: 10.1109/CVPR.2015.7298965.

[24] Lucny, A., Dillinger, V., Kacurova, G. a Racev, M. Shape-Based Alignment of the Scanned Objects Concerning Their Asymmetric Aspects. *Sensors*. 2021, zv. 21, č. 4. DOI: 10.3390/s21041529. ISSN 1424-8220. Dostupné z: https://www.mdpi.com/1424-8220/21/4/1529.

[25] *MaxPool2d* [online]. PyTorch 1.10 documentation, 2019 [cit. 2022-04-25]. Dostupné z: https://pytorch.org/docs/stable/generated/torch.nn.MaxPool2d.html.

[26] Minaee, S., Boykov, Y. Y., Porikli, F., Plaza, A. J., Kehtarnavaz, N. et al. Image Segmentation Using Deep Learning: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2021, s. 1–1. DOI: 10.1109/TPAMI.2021.3059968.

[27] Mittal, R. a Garg, A. Text extraction using OCR: a systematic review. In: IEEE. *2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)*. 2020, s. 357–362.

[28] Noble, W. S. What is a support vector machine? *Nature biotechnology*. Nature Publishing Group. 2006, zv. 24, č. 12, s. 1565–1567.

[29] Nwankpa, C., Ijomah, W., Gachagan, A. a Marshall, S. Activation Functions: Comparison of Trends in Practice and Research for Deep Learning. In:. December 2020.

[30] Otsu, N. A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics*. 1979, zv. 9, č. 1, s. 62–66. DOI: 10.1109/TSMC.1979.4310076.

[31] Padilla, R., Lobato Passos, W., Dias, T., Netto, S. a Silva, E. da. A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit. *Electronics*. Január 2021, zv. 10, s. 279–306. DOI: 10.3390/electronics10030279.

[32] Petro, A.-B., Sbert, C. a Morel, J.-M. Multiscale Retinex. *Image Processing On Line*. Apríl 2014, zv. 4, s. 71–88. DOI: 10.5201/ipol.2014.107.

[33] Prakash, J. *Non maximum suppression: Theory and implementation in Pytorch* [online]. LearnOpenCV, 2. Jun 2021 [cit. 2022-04-25]. Dostupné z: https://learnopencv.com/non-maximum-suppression-theory-and-implementation-in-pytorch/.

[34] Ramachandran, P., Zoph, B. a Le, Q. V. Searching for Activation Functions. *ArXiv*. 2018, abs/1710.05941.

[35] Redmon, J., Divvala, S., Girshick, R. a Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, s. 779–788. DOI: 10.1109/CVPR.2016.91.

[36] Ren, S., He, K., Girshick, R. a Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. Cambridge, MA, USA: MIT Press, 2015, s. 91–99. NIPS'15.

[37] Ridler, T. a Calvard, S. Picture Thresholding Using an Iterative Selection Method. *IEEE Transactions on Systems, Man, and Cybernetics*. 1978, zv. 8, č. 8, s. 630–632. DOI: 10.1109/TSMC.1978.4310039.

[38] Ronneberger, O., Fischer, P. a Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Navab, N., Hornegger, J., Wells, W. M. a Frangi, A. F., ed. *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Cham: Springer International Publishing, 2015, s. 234–241. ISBN 978-3-319-24574-4.

[39] Ronneberger, O., Fischer, P. a Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Navab, N., Hornegger, J., Wells, W. M. a Frangi, A. F., ed. *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Cham: Springer International Publishing, 2015, s. 234–241. ISBN 978-3-319-24574-4.

[40] Sharma, S., Sharma, S. a Athaiya, A. ACTIVATION FUNCTIONS IN NEURAL NETWORKS. *International Journal of Engineering Applied Sciences and Technology*. Máj 2020, zv. 04, s. 310–316. DOI: 10.33564/IJEAST.2020.v04i12.054.

[41] Sudre, C. H., Li, W., Vercauteren, T. K. M., Ourselin, S. a Cardoso, M. J. Generalised Dice overlap as a deep learning loss function for highly unbalanced segmentations. *Deep learning in medical image analysis and multimodal learning for clinical decision support : Third International Workshop, DLMIA 2017, and 7th International Workshop, ML-CDS 2017, held in conjunction with MICCAI 2017 Quebec City, QC,...* 2017, zv. 2017, s. 240–248.

[42] torchvision. *Object detection reference training scripts* [online]. GitHub, 2022. Dostupné z: https://github.com/pytorch/vision/tree/main/references/detection.

[43] *Torchvision object detection finetuning tutorial* [online]. PyTorch, 2021 [cit. 2022-04-25]. Dostupné z: https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html.

[44] Tzutalin. *LabelImg* [Free Software: MIT License]. GitHub, 2015 [cit. 2022-04-25]. Dostupné z: https://github.com/tzutalin/labelImg.

[45] Uijlings, J., Sande, K., Gevers, T. a Smeulders, A. Selective Search for Object Recognition. *International Journal of Computer Vision*. September 2013, zv. 104, s. 154–171. DOI: 10.1007/s11263-013-0620-5.

[46] VanderWerp, D. *What do the numbers on a tire mean?* [online]. Car and Driver, 6. marca 2019 [cit. 2022-04-25]. Dostupné z: https://www.caranddriver.com/features/a16580427/how-to-read-a-tire-sidewall/.

[47] Varatharasan, V., Shin, H.-S., Tsourdos, A. a Colosimo, N. Improving Learning Effectiveness For Object Detection and Classification in Cluttered Backgrounds. In:. November 2019, s. 78–85. DOI: 10.1109/REDUAS47371.2019.8999695.

[48] Yakubovskiy, P. *Segmentation Models Pytorch* [online]. GitHub, 2020 [cit. 2022-04-25]. Dostupné z: https://github.com/qubvel/segmentation_models.pytorch.

[49] Yamashita, R., Nishio, M., Do, R. a Togashi, K. Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*. Jún 2018, zv. 9. DOI: 10.1007/s13244-018-0639-9.

Príloha A

Typy vrstiev v konvolučných neurónových sieťach

Táto príloha obsahuje stručný popis vrstiev používaných v konvolučných neurónových sieťach.

Konvolučná vrstva

Konvolúcia je typ lineárnej operácie, kde malé pole čísel, nazývané konvolučné jadro alebo kernel, je aplikované na vstup, ktorým je pole čísel nazývané tenzor [49]. Nech vstupný tenzor má dimenzie CxHxW, kde C je počet kanálov (resp. hĺbka), H je výška a W šírka. Potom kernel musí mať rozmery CxMxN, kde C je počet kanálov kernela, ktorý je totožný s počtom kanálov vstupného tenzora, a M a N sú výška, respektíve šírka kernela. Ako veľkosť kernela sa zvyčajne udávajú len priestorové rozmery, teda M x N. Typická veľkosť kernela je 3 x 3, menej často 5 x 5 alebo 7 x 7 [49].

Konvolučná vrstva môže obsahovať obecne K kernelov totožných rozmerov. Počet kernelov udáva počet kanálov výstupu.

Každý kernel je priložený na každú priestorovú pozíciu tenzora. Hodnota výstupu na tejto pozícií je daná súčtom súčinov odpovedajúcich si hodnôt kernelu a tenzora.

Zapísané matematicky, hodnota na priestorovej pozícií (i, j) v k-tom kanáli výstupu je vypočítaná ako:

$$G_{k,i,j} = (\sum_{c=0}^{C-1} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} h_{k_{c,m,n}} \cdot f_{C-c,i-m,j-n}) + b_k, \tag{A.1}$$

kde h_k je k-tý kernel, b_k je bias pre k-tý kernel a f je vstupný tenzor.

Dôležitou vlastnosťou konvolúcie je zdieľanie parametrov pre všetky priestorové pozície. Vďaka tomu má vrstva menší počet parametrov – počet parametrov konvolučnej vrstvy je $C \cdot M \cdot N \cdot K$ (+ K, ak je použitý bias) – a je invariantná voči posunutiu, čo je pre obrázok kľúčové.

Takto definovaná konvolúcia neumožňuje priloženie stredu konvolučného jadra (väčšieho než 1x1) k hraničným prvkom vstupného tenzora, pretože pozície mimo hraníc tenzora nemajú definovanú hodnotu. Tým pádom sú priestorové rozmery výstupu menšie než priestorové rozmery vstupu a stráca sa informácia z okrajových pixelov [3]. Na riešenie tohto problému sa používa padding – k hraniciam obrázka je pridaný istý počet riadkov a stĺpcov. Najčastejšie sa používa nulový padding.

Ďalším parametrom konvolučnej vrstvy je veľkosť kroku. Je to vlastne vzdialenosť medzi dvoma po sebe idúcimi pozíciami, na ktoré je konvolučné jadro priložené. Zvýšenie veľkosti kroku má za následok zmenšenie výstupu.

Majme dáta rozmeru N x N. Potom veľkosť výstupu konvolučnej vrstvy s veľkosťou kernela F x F, paddingom P na každej strane a krokom S je [3] :

$$O = 1 + \left\lfloor \frac{N + 2P - F}{S} \right\rfloor . \tag{A.2}$$

Transponovaná konvolúcia

Transponovaná konvolúcia (tiež známa ako dekonvolúcia) je často používaná pri nadvzorkovaní, pretože zvyšuje rozmery výstupnej mapy. Majme opäť vstup veľkosti N x N, veľkosť kernela F x F, padding P na každej strane a navyše parameter R značiaci výstupný padding. Potom veľkosť výstupu transponovanej konvolúcie je [1] :

$$O = (N - 1) \cdot S + F - 2P + R. \tag{A.3}$$

Vidíme, že pri rovnakých parametroch pre klasickú a transponovanú konvolúciu sú ich vstupné a výstupné rozmery inverzné[1].

Pri výpočte transponovanej konvolúcie je celý kernel vynásobený jednou hodnotou vstupu a umiestnený do výstupnej matice. Toto je opakované pre každú hodnotu vstupu. Krok pri transponovanej konvolúcií určuje veľkosť kroku vo výstupnej matici, nie vo vstupe ako pri klasickej konvolúcií. Prekrývajúce sa hodnoty vo výstupnej matici sa sčítajú.

Poolingová vrsta

Poolingová vrstva slúži na zmenšenie priestorových dimenzií mapy príznakov pri zachovaní počtu kanálov. Vrstva nemá žiadne trénovateľné parametre.

Parametre poolingovej vrstvy sú podobné ako pri konvolúcií. Poolingová vrstva rozdelí vstup na menšie časti odpovedajúce veľkosti kernela a v rámci týchto častí pre každý kanál nezávisle vykoná operáciu, ako napríklad nájdenie minima alebo výpočet priemeru. Veľkosť kroku opäť udáva vzdialenosť dvoch po sebe idúcich pozícií, na ktorú je filter priložený.

Pre zjednodušenie predpokladajme, že veľkosť kernela F a veľkosť kroku S je v oboch priestorových dimenziách rovnaká. Potom môžeme výstup max poolingu (pooling vykonávajúci operáciu maximum) matematicky zapísať nasledovne [25]:

$$Maxpool(c, i, j) = \max_{0 \leq m \leq F-1,\ 0 \leq n \leq F-1} f(c, S \cdot h + m, S \cdot w + m), \tag{A.4}$$

kde c je kanál a (i, j) sú priestorové súradnice vo výstupnom tenzore.

Plne prepojená vrstva

V plne prepojenej vrstve je každý neurón prepojený s každým výstupom predchádzajúcej vrstvy. Výstup neurónu sa teda počíta ako skalárny súčin hodnôt vstupu s váhami, ku

[1] Kvôli zaokrúhľovaniu nadol pri výpočte výstupných rozmerov konvolúcie je viacero vstupných tvarov namapovaných na rovnaký výstupný tvar. Na špecifikovanie, ktorý z týchto tvarov má mať výstup transponovanej konvolúcie, treba nastaviť parameter výstupný padding [5].

ktorému je pripočítaný bias [4]. Ak N je počet výstupov predchádzajúcej vrstvy a K počet výstupov plne prepojenej vrstvy, vrstva má $N \cdot K + K$ parametrov.

Plne prepojenú vrstvu je možné nahradiť konvolúciou s K kernelmi veľkosti 1 x 1.

Nelineárne aktivačné funkcie

Bez použitia nelineárnej aktivačnej funkcie by bola výstupom modelu lineárna funkcia (polynóm stupňa jeden), a to bez ohľadu na počet lineárnych vrstiev použitých v modeli. Vďaka nelineárnym funkciám má model schopnosť naučiť sa a rozpoznať aj komplexnejšie vzťahy medzi vstupom a očakávaným výstupom [40].

ReLU

Aktivačná funkcia ReLU (rectified linear unit) je podľa [34] v súčasnosti najpoužívanejšia aktivačná funkcia v hlbokom učení. Jej predpis pre vektor x na vstupe je [29]:

$$ReLU(x) = max(0, x_i) = \begin{cases} x_i & \text{ak } x_i \geq 0 \\ 0 & \text{ináč} \end{cases} \tag{A.5}$$

Softmax

Funkcia softmax prijíma ako vstup vektor reálnych čísel a na výstup vracia vektor pravdepodobností rovnakej dĺžky, ktorého prvky sú v intervale $[0, 1]$ a ich súčet je 1. Predpis funkcie softmax je [29]:

$$softmax(x) = \frac{e^{x_i}}{\sum_j e^{x_j}}. \tag{A.6}$$

Zvyčajne sa vyskytuje vo výstupných vrstvách multiclass modelov, kde vracia pravdepodobnosť pre každú triedu [29].

Sigmoida

Funkcia sigmoida prijíma na vstup reálne číslo x a vráti pravdepodobnosť z intervalu $[0, 1]$. Je daná vzťahom [29]:

$$sigmoid(x) = \frac{1}{1 + e^{-x}}. \tag{A.7}$$

Používa sa v binárnej klasifikácií a pri logistickej regresii.

Príloha B

Popis formátu anotácie hĺbkových snímok v dátovej sade

Anotácia každej hĺbkovej snímky pozostáva z 8-bitových obrázkov formátu PNG s maskou každého symbolu nachádzajúceho sa na danej hĺbkovej snímke a z CSV súboru, v ktorom každý riadok obsahuje informáciu o jednom symbole. Štruktúra a význam stĺpcov CSV súboru sú popísané v tabuľke B.1.

Názov stĺpca	Význam stĺpca
ID	identifikátor symbolu
Unicode	kód typu symbolu (väčšinou totožný s ASCII kódom)
Complex glyph index	pôvodne unikátny index zloženého symbolu
Component index	pôvodne index komponentu v rámci symbolu[1]
X	x-ová súradnica ľavého horného bodu bounding boxu symbolu
Y	y-ová súradnica ľavého horného bodu bounding boxu symbolu
Width	šírka bounding boxu symbolu
Height	výška bounding boxu symbolu
Mask	názov PNG súboru s maskou symbolu
Correlation	miera istoty, s akou sa symbol na danej pozícií nachádza
hatched	informácia, či je symbol šrafovaný (hodnoty 1, resp. 0)

Tabuľka B.1: Popis položiek CSV súbora s anotáciou

[1] Pri anotovaní dátovej sady boli všetky symboly skladajúce sa z viacerých symbolov (napríklad dvojbodka alebo znak 'i'), zložené do jedného, stĺpce Complex glyph index a Component index teda stratili význam.

Príloha C

Popis symbolov v dátovej sade

Medzi symboly nachádzajúce sa v dátovej sade patria písmená, čísla, interpunkčné znamienka a špeciálne symboly. V CSV súboroch s anotáciou sú v stĺpci 'Unicode' písmená, čísla a interpunkčné znamienka označené svojim ASCII kódom.

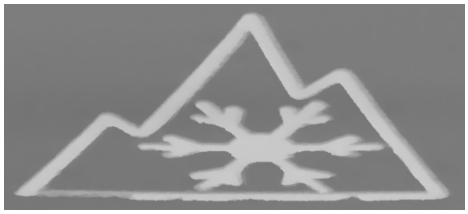
Keďže špeciálne symboly nemajú svoj ASCII kód, bola im pri anotácií pridelená číselná hodnota, začínajúc číslom 127.

Tabuľka C.1 popisuje špeciálne symboly, ich kódy v anotácií a informáciu o tom, či boli použité pri trénovaní v rámci popísaných experimentov. Sieť bola trénovaná tak, aby nápisy 'E13' a 'CCC', oba uprostred krúžku, rozpoznávala po častiach, teda aby rozpoznala jednotlivé písmená, čísla a symbol kruhu. Rozpoznanie kompletného symbolu môže byť implementované spracovaním výsledkov inferencie a spojením jednotlivých detegovaných komponentov.

Kód	Obrázok	Popis symbolu	Natrénované
127		symboly, ktoré nepatria do žiadnej inej kategórie	nie
128	C.1	symbol zimnej pneumatiky (vločka v kopcoch)	áno
129	C.9	dvojitá šípka	áno
130	C.2	zdobené písmeno 'F'	áno
131	C.3	písmeno 'D' v šípke	áno
132	C.4	nápis 'E13' v kruhu	nie (po častiach)
133	C.4, C.10	kruh	áno
134	C.5	symbol letnej pneumatiky (slniečko)	áno
135	C.6	symbol pneumatiky na mokrý povrch (3 kvapky)	áno
136	C.7	symbol zimnej pneumatiky (vločka)	áno
137	C.8	ozdobná topánka s krídlom	áno
138	C.10	nápis 'CCCs' v krúžku	nie (po častiach)

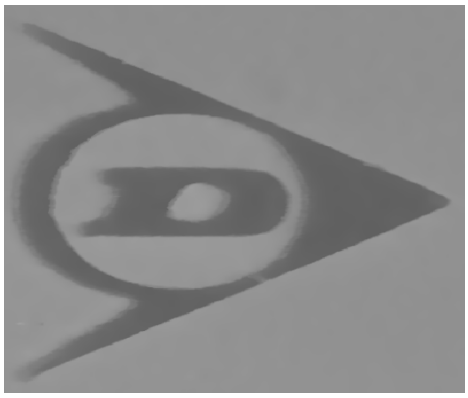
Tabuľka C.1: Popis špeciálnych symbolov a ich kódov v anotácií.

Na nasledujúcich stranách sa nachádzajú obrázky jednotlivých špeciálnych symbolov. Symbol krúžku s kódom 133 sa v dátovej sade nevyskytuje samostatne, ale je súčasťou symbolov na obrázkoch C.4 a C.10.

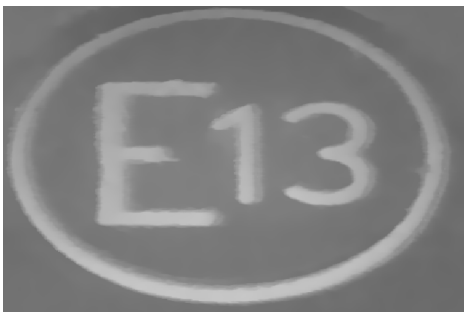


Obr. C.1: Vločka v kopcoch - kód 128

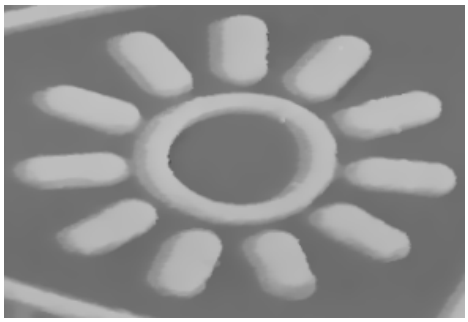
Obr. C.2: Zdobené 'F' - kód 130



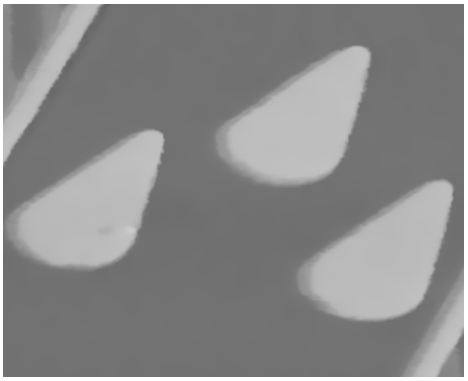
Obr. C.3: 'D' v šípke - kód 131



Obr. C.4: Nápis 'E13' - kód 132

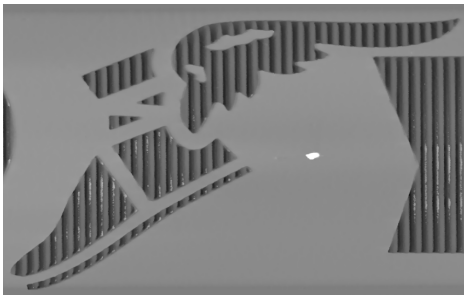


Obr. C.5: Slniečko - kód 134



Obr. C.6: Tri kvapky - kód 135

Obr. C.7: Vločka - kód 136



Obr. C.8: Topánka - kód 137

Obr. C.9: Dvojitá šípka - kód 129. Písmená nápisu 'RIM PROTECTOR' sú v anotácii samostatne.

Obr. C.10: Nápis 'CCCs' - kód 138. Samostatným symbolom je aj krúžok s kódom 133.

Príloha D

Obsah priloženého pamäťového média

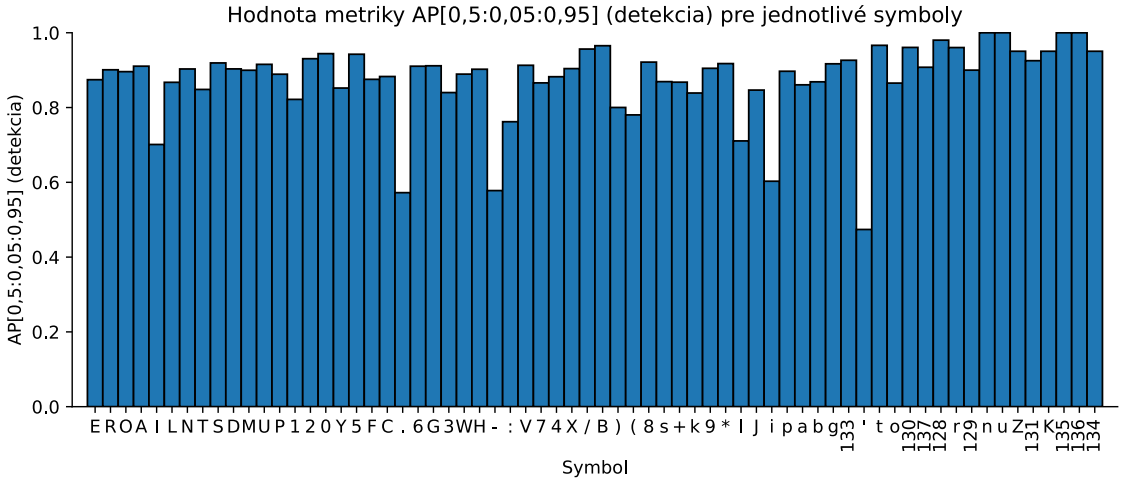
```
source ..........................................................Zdrojový kód
    dataset ....................................Kód pre prácu s dátovou sadou
    MaskRCNN ..............................Kód pre prácu s modelom Mask R-CNN
        notebooks .................Adresár s pripravenými Jupyter notebookmi
        experiments .......................Súbory s jednotlivými experimentmi
        torchvision_scripts ...............Súbory prevzaté z torchvision[42]
    UNET ....................................... Kód pre prácu s modelom U-Net
    other ..................................................Zvyšný kód
    setup.py ............................Súbor umožňujúci inštaláciu balíčka
    requirements.txt Popis požadovaných knižníc
dataset ..................................................Ukážka dátovej sady
    TrainingSet ...........Ukážka snímok a anotácií použitých pri trénovaní
    TestingSet ............Ukážka snímok a anotácií použitých pri testovaní
technicka_sprava_src .....................Zdrojový tvar technickej správy
README.txt .......................Popis obsahu média a návod na inštaláciu
technicka_sprava.pdf ............................PDF s bakalárskou prácou
video_prezentacia.mp4 ..............................Videoprezentácia práce
best_model.pth ....................................Váhy najlepšieho modelu
```

Príloha E

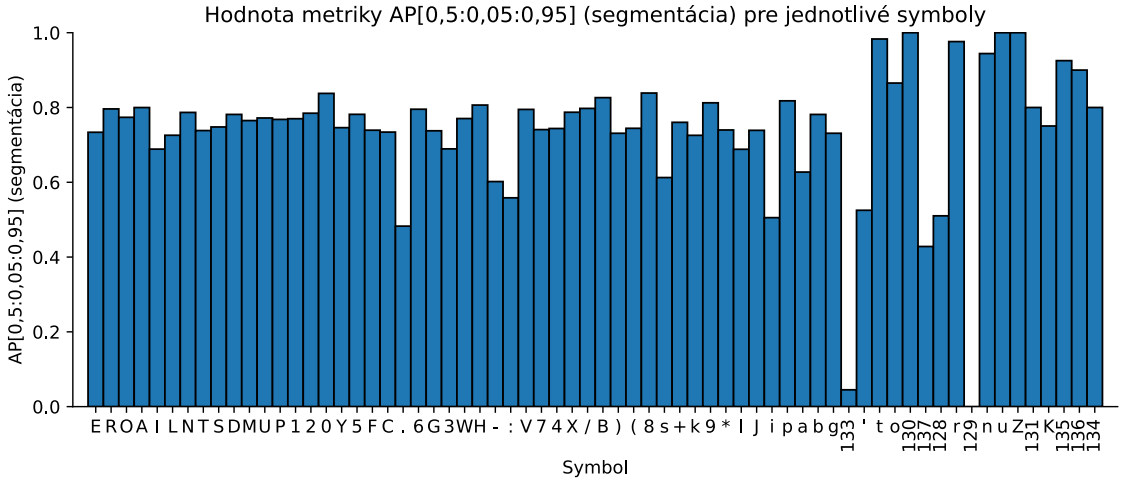
Metriky a ukážka výstupu najlepšieho modelu

Ako najlepšia z variant skúmaných v tejto práci sa ukázala tá s predtrénovaným modelom Mask R-CNN, ktorého výstup pre segmentáciu bol zväčšený na rozmer 56 x 56. Táto príloha obsahuje metriky vypočítané pre tento model a ukážky jeho výstupu. Grafy na obrázku E.1 znázorňujú hodnoty $AP[0,5:0,05:0,95]$ pre detekciu a segmentáciu dosiahnuté pre jednotlivé triedy.

Segmentácia dosiahnutá na type pneumatiky, ktorý sa nachádzal len v testovacej množine, je ukázaná na obrázkoch E.2 a E.3. Občasné zlyhanie detekcie, ktorého predpokladaným dôvodom je menšia odolnosť natrénovaného modelu voči zmene škálovania symbolov, je možné vidieť na obrázku E.4.

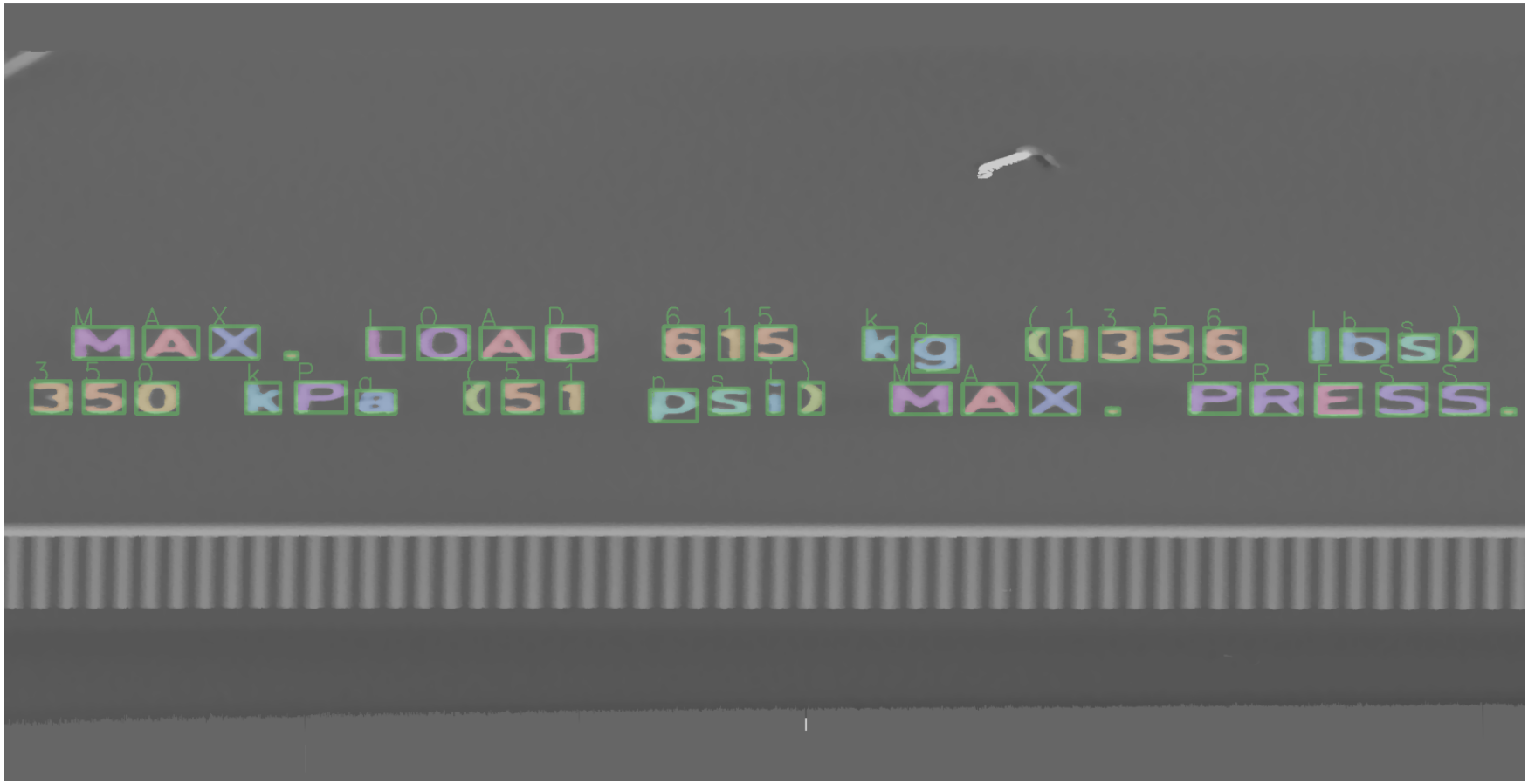


(a)

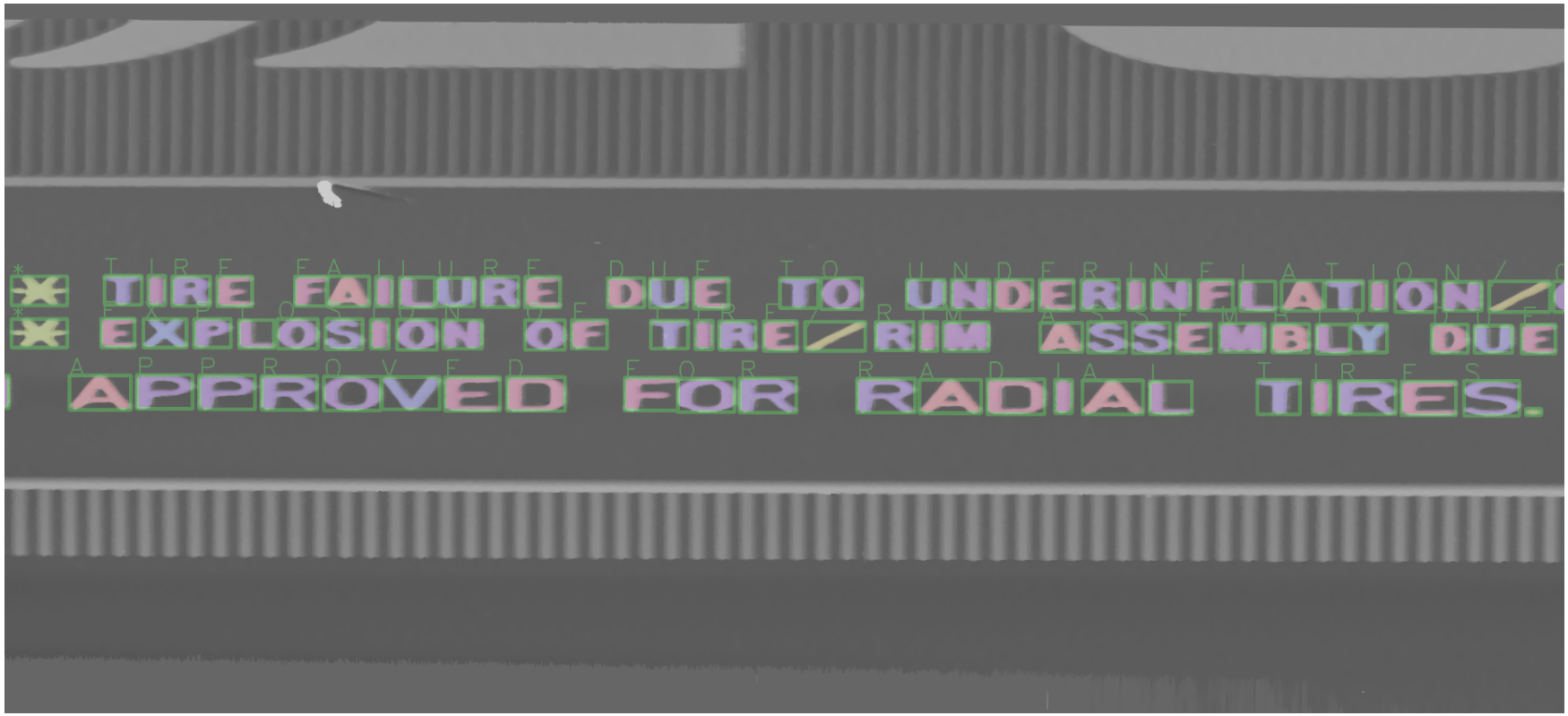


(b)

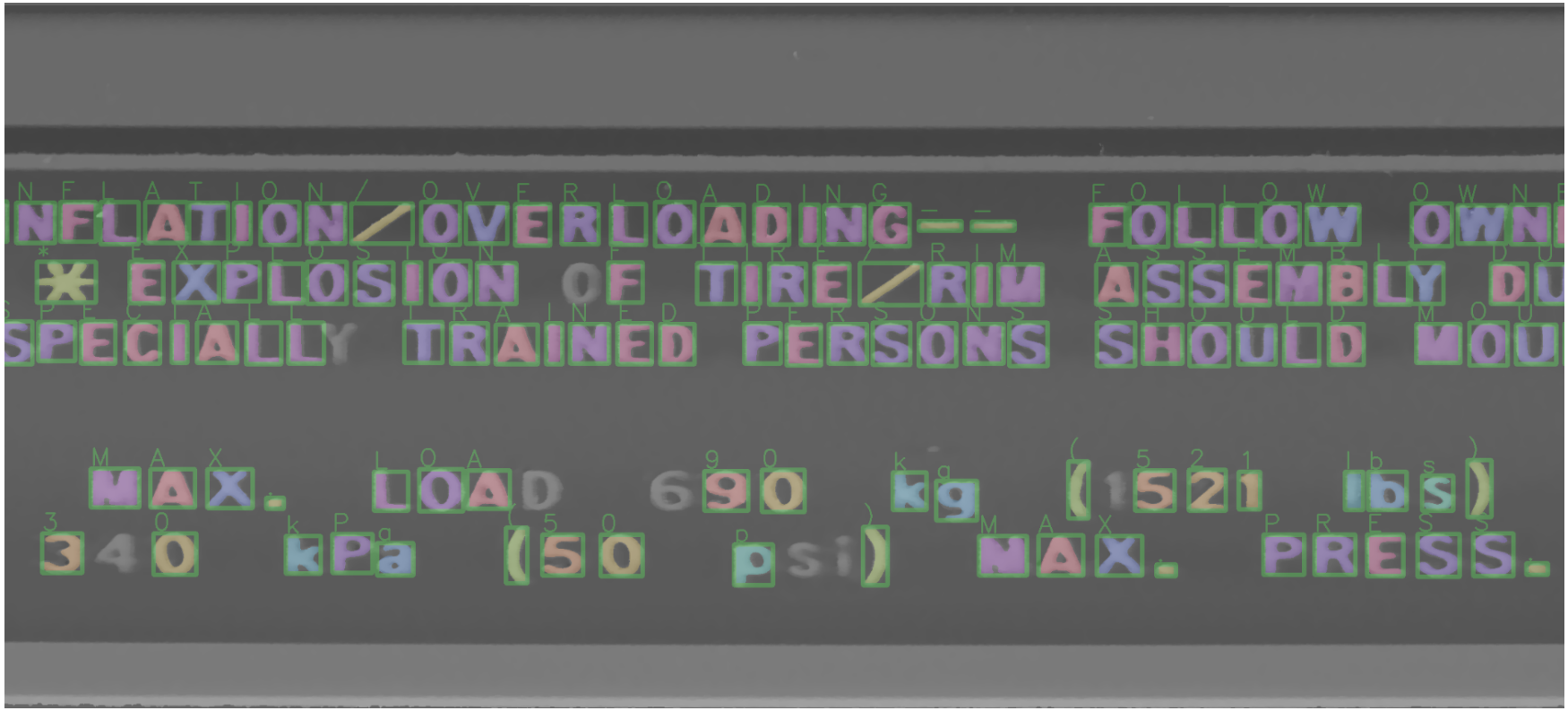
Obr. E.1: Metrika $AP[0,5:0,05:0,95]$ pre **(a)** detekciu a **(b)** segmentáciu pre jednotlivé typy symbolov. Zobrazené metriky boli vypočítané pre model, ktorý pre všetky triedy vráti masku veľkosti 56 x 56.



Obr. E.2: Ukážka dosiahnutej segmentácie na nevidenom type pneumatiky. Výstup bol vyprodukovaný natrénovaným modelom so zväčšeným rozlíšením masky pre všetky triedy.



Obr. E.3: Ďalšia ukážka dosiahnutej segmentácie na nevidenom type pneumatiky. Výstup bol vyprodukovaný natrénovaným modelom so zväčšeným rozlíšením masky pre všetky triedy.



Obr. E.4: Ukážka nedetekovania niektorých symbolov. Zlyhanie detekcie sa v takomto rozsahu prejavovalo len na jednom type pneumatiky. Dôvodom je prevdepodobne menšia odolnosť natrénovaného modelu voči zmene škálovania symbolov. Výstup bol vyprodukovaný natrénovaným modelom so zväčšeným rozlíšením masky pre všetky triedy.