VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2021 Bc. Martin Hrdý

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

SEGMENTACE OBRAZOVÝCH DAT VYUŽITÍM HLUBOKÝCH NEURONOVÝCH SÍTÍ
IMAGE DATA SEGMENTATION USING DEEP NEURAL NETWORKS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. Martin Hrdý

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Martin Kiac

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Bc. Martin Hrdý ***ID:*** 195826

Ročník: 2 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Segmentace obrazových dat využitím hlubokých neuronových sítí

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je navrhnout a realizovat neuronovou síť schopnou segmentovat objekty v obraze podle naučených vzorů. Výsledná neuronová síť by měla být schopna segmentovat jednotlivé objekty nacházející se ve scéně obrazu. Za tímto účelem je třeba vytvořit vlastní trénovací množinu, pomocí které bude segmentační neuronová síť následně trénovaná. V rámci řešení práce je také potřebné implementovat algoritmus, který by měl být schopen využitím naučeného modelu neuronové sítě analyzovat jednotlivé objekty a tak vyhodnotit různé mimořádné nebo jinak nebezpečné situace. Výsledkem řešení práce by měla být jednoduchá aplikace, která bude demonstrovat problematiku segmentace objektů v obraze využitím neuronových sítí.

DOPORUČENÁ LITERATURA:

[1] MICHELUCCI, Umberto. Applied Deep Learning: A Case-Based Approach toUnderstanding Deep Neural Networks. 1. vydání. Switzerland: Apress, 2018. 425 s. ISBN 978-1-4842-3789-2.

[2] Krizhevsky, A., Sutskever, I., Hinton G. E., ImageNet Classification with Deep Convolutional Neural Networks. In Advances in Neural Information Processing Systems 25, 2012. p. 1097-1105.

Termín zadání: 1.2.2021 ***Termín odevzdání:*** 24.5.2021

Vedoucí práce: Ing. Martin Kiac

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRAKT

Cílem této diplomové práce je seznámit se a nastudovat teorii současných segmentačních metod, které používají hluboké učení. Na základě teoretických znalostí bude navržena a vytvořena segmentační neuronová síť, která bude schopna segmentovat jednotlivé instance objektů. Segmentační neuronová síť bude zaměřena na detekci elektronických součástek na deskách plošných spojů.

KLÍČOVÁ SLOVA

Hluboké neuronové sítě, Segmentace, OpenCV, Mask R-CNN, Deska plošných spojů

ABSTRACT

The main aim of this master's thesis is to get acquainted with the theory of the current segmentation methods, that use deep learning. Segmentation neural network that will be capable of segmenting individual instances of the objects will be proposed and created based on theoretical knowledge. The main focus of the segmentation neural network will be segmentation of electronic components from printed circuit boards.

KEYWORDS

Deep neural networks, Segmentation, OpenCV, Mask R-CNN, Printed circuit board

HRDÝ, Martin. *Segmentace obrazových dat s využitím hlubokých neuronových sítí*. Brno, 2021, 77 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Martin Kiac,

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Segmentace obrazových dat s využitím hlubokých neuronových sítí“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Martinu Kiacovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

Počítačové vidění se v poslední době stává nedílnou součástí běžného života, ať se jedná o využití v medicíně, moderní geografii nebo automobilovém průmyslu. Segmentování obrazových dat je nepostradatelnou součástí počítačového vidění.

V teoretické části práce je zahrnuto téma počítačového vidění, umělých neuronových sítí, dále konvolučních neuronových sítí a rovněž sítí segmentačních. Práce popisuje rozsáhlou problematiku neuronových sítí od prostých základů, jako je například stavba neuronu, aktivační funkce nebo topologie sítí, až po komplikované architektury modelů sítí. Vysvětlena je zejména funkčnost vybraných architektur segmentačních neuronových sítí.

Praktická část se zabývá jednotlivými kroky při implementaci segmentační neuronové sítě schopné segmentovat jednotlivé instance součástek na snímcích desek plošných spojů. Pro navržení segmentační neuronové sítě bylo nutné problematiku detailně prostudovat. V textu práce je proces vytvoření vlastního rozsáhlého datasetu a princip anotace pomocí dostupných nástrojů. Práce popisuje úpravy implementace Matterport Mask R-CNN za účelem použití vlastního datasetu, zpracování JSON anotací, přidání augmentace, přidání mAP a F-score metrik nebo vytvoření HTML reportu, který obsahuje výsledky ve formě tabulek a grafů. K sestrojení sítě byly použity knihovny Keras a Tensorflow. Výstup sítě nabízí kromě výsledných segmentací také údaje o časové náročnosti. Segmentační síť je propojena se sítí typu YOLO za účelem detekce mimořádné události. Práce popisuje a zdůvodňuje rozdíly mezi páteřními archtekturami Resnet50 a ResNet101. Na závěr jsou v práci přiloženy výsledky segmentací, naměřené metriky, ztrátové funkce, časová náročnost, návod k instalaci a spuštění sítě.

1 Počítačové vidění

Počítačové vidění je transformace statických či dynamických obrazových dat buď na jinou reprezentaci, nebo na určité rozhodnutí, a to za účelem porozumění obsahu. Všechny transformace jsou provedeny za jistým účelem. Novou reprezentací obrazu může být například převedení obrazových dat na černobílé barevné spektrum či odstranění pohybu kamery z dynamického obrazu. Rozhodnutím může být například určení počtu daných automobilů na snímku [1, 2].

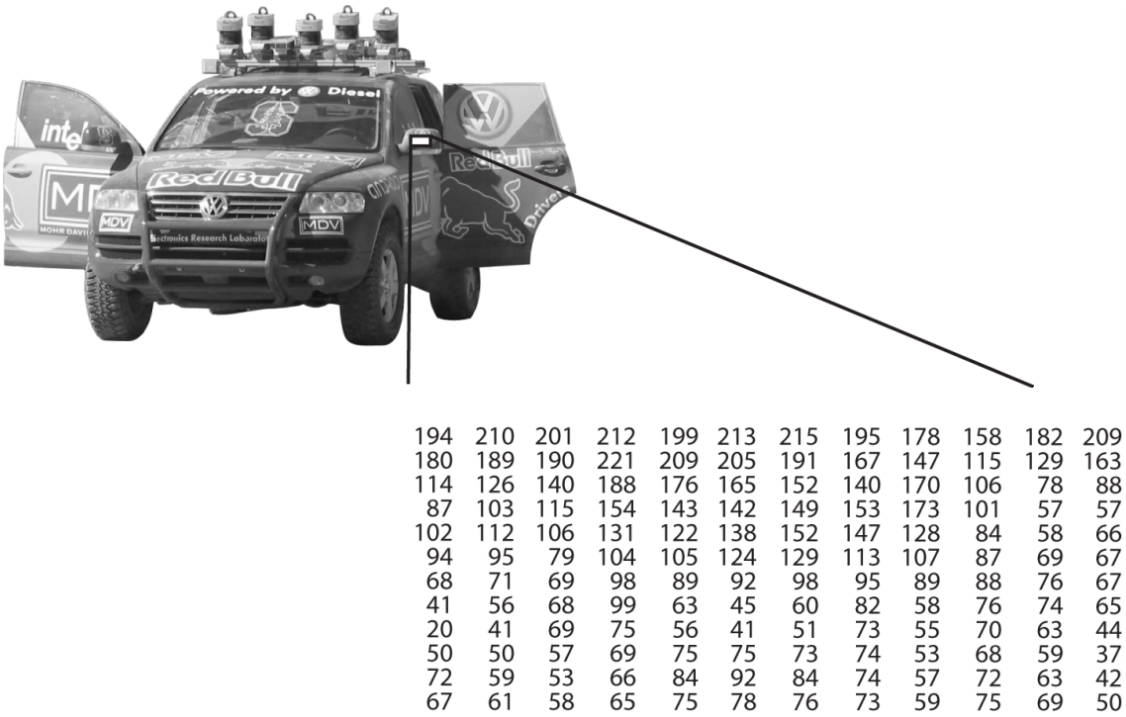
Pro člověka může být nalezení objektu v obraze považováno za velmi snadnou úlohu. Lidský mozek dokáže vizuální vstup rozdělit na nespočet odlišných typů informací, díky kterým dokáže identifikovat a investigovat důležité informace v obraze. Při pohledu na objekt si mozek dokáže vytvořit klíčové body, dle kterých objekt velmi rychle klasifikuje bez ohledu na osvětlení, perspektivu, velikost nebo orientaci. Již od mládí si mozek vytváří mnoho asociací, které postupně zdokonaluje v průběhu života [1, 2].

Porozumění dané scéně je pro stroje mnohem obtížnější. V systému počítačového vidění extrahuje počítač z dané scény pouze matici číselných hodnot. Příklad matice číselných hodnot je zobrazen na obrázku 1.1. Tato matice číselných hodnot většinou neobsahuje žádné rozpoznání vzoru a také nemá žádné asociace jako má lidský mozek. Často matice obsahuje faktory, které rozpoznání komplikují, jako například ruchy a zkreslení způsobené různými odrazy světla, pohyby objektů nebo povětrnostními podmínkami. Ze získané dvourozměrné matice hodnot reprezentující trojrozměrný svět často prakticky neexistuje způsob, jak zreprodukovat trojrozměrný signál, jelikož stejný dvojrozměrný obraz může reprezentovat nekonečně mnoho odlišných trojrozměrných scén [1, 2].

1.1 Knihovna OpenCV

OpenCV je knihovna pro počítačové vidění (*computer vision* – CV) s otevřeným zdrojovým kódem. Knihovna je nativně napsána pro programovací jazyky C a C++, nicméně vývoj je aktivní i pro jazyky Python, Java nebo MATLAB. OpenCV může být spuštěno pod operačními systémy Windows, Linux i macOS [1].

Hlavním cílem knihovny OpenCV je poskytnutí infrastruktury počítačového vidění, která je snadno použitelná a zároveň umožňuje vytvoření poměrně sofistikované aplikace. Počítačové vidění se často prolíná se strojovým učením (*machine learning* – ML), a proto OpenCV obsahuje univerzální modul pro strojové učení (modul ML). Tento modul je dostatečně univerzální pro použití na jakýkoliv problém strojového učení, nicméně prvotní zaměření bylo rozpoznání statistických vzorů [1].



194	210	201	212	199	213	215	195	178	158	182	209
180	189	190	221	209	205	191	167	147	115	129	163
114	126	140	188	176	165	152	140	170	106	78	88
87	103	115	154	143	142	149	153	173	101	57	57
102	112	106	131	122	138	152	147	128	84	58	66
94	95	79	104	105	124	129	113	107	87	69	67
68	71	69	98	89	92	98	95	89	88	76	67
41	56	68	99	63	45	60	82	58	76	74	65
20	41	69	75	56	41	51	73	55	70	63	44
50	50	57	69	75	75	73	74	53	68	59	37
72	59	53	66	84	92	84	74	57	72	63	42
67	61	58	65	75	78	76	73	59	75	69	50

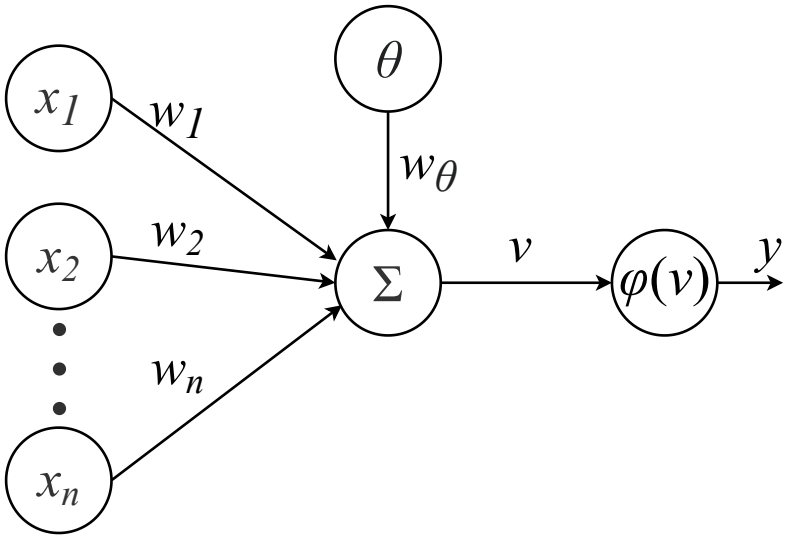
Obr. 1.1: Získaná matice číselných hodnot, převzato z [1].

Od roku 1999, kdy vyšlo první vydání knihovny, bylo OpenCV využito v mnoha aplikacích, produktech či výzkumech. Jako příklad může být uvedeno využití u vzájemného napojování satelitních snímků do sebe při vytváření webových map, zarovnávání obrazu při skenování, analýza objektů, kalibrace kamerového záznamu, automatické monitorovací a bezpečnostní systémy nebo také rozpoznání hudby [1].

2 Umělé neuronové sítě

2.1 Neuron

Obecné schéma neuronu je zobrazeno na obrázku 2.1. Vstupy neuronu jsou vynásobeny se synaptickými váhami a společně s biasem je vytvořen vnitřní potenciál neuronu, který dále slouží jako vstupní parametr pro aktivační funkci neuronu, na základě které je určen konečný výstup neuronu [3].



Obr. 2.1: Obecné schéma neuronu.

Rovnice umělého neuronu je dána vztahem:

$$y = \varphi(v) = f(\sum_{k=1}^{i} w_k x_k + \theta_k w_\theta), \tag{2.1}$$

kde y je výstup neuronu, v vnitřní potenciál, i počet vstupů, k číslo vstupu, w synaptická váha, x vstup neuronu a θ bias [3, 4].

2.2 Aktivační funkce

Aktivační funkce neuronu, značená symbolem φ, je matematická funkce, která definuje výstup neuronu y_k na základě sumy vah $\sum$ a biasu θ_k, jejichž součet tvoří vnitřní potenciál neuronu v_k. Výstupní hodnota po průchodu aktivační funkcí typicky nabývá hodnot z intervalu $[0, 1]$ nebo také $[-1, 1]$. Výběr konkrétní aktivační funkce závisí na problému, který má daná neuronová síť vyřešit. Dle funkčnosti můžeme rozlišit základní čtyři typy aktivačních funkcí:

- **Prahová funkce** – pro tuto funkci platí:

$$\varphi(v) = \begin{cases} 1 & \text{pokud } v \geq 0 \\ 0 & \text{pokud } v < 0. \end{cases} \tag{2.2}$$

  Tato funkce je také často označována jako Heavisideova funkce nebo jednotkový skok.
- **Sigmoidální funkce** – pro tuto funkci platí:

$$\varphi(v) = \frac{1}{1 + e^{av}}. \tag{2.3}$$

  Upravením parametru a lze dosáhnout změny sklonu sigmoidální funkce. Pokud by se parametr a blížil nekonečnu, sigmoidální funkce by se změnila na funkcii prahovou. Jde o nejvyužívanější typ aktivační funkce. Sigmoidální funkce je oblíbená v modelech, které jako výstup predikují pravděpodobnost, jelikož ta nabývá hodnot z itervalu 0 až 1.
- **Lineární funkce** – pro tuto funkci platí:

$$\varphi(v) = v. \tag{2.4}$$

  Jde o triviální funkci, která jednoduše vrací nezměněnou vstupní hodnotu.
- **ReLU** (*Rectified linear activation function*) – pro tuto funkci platí:

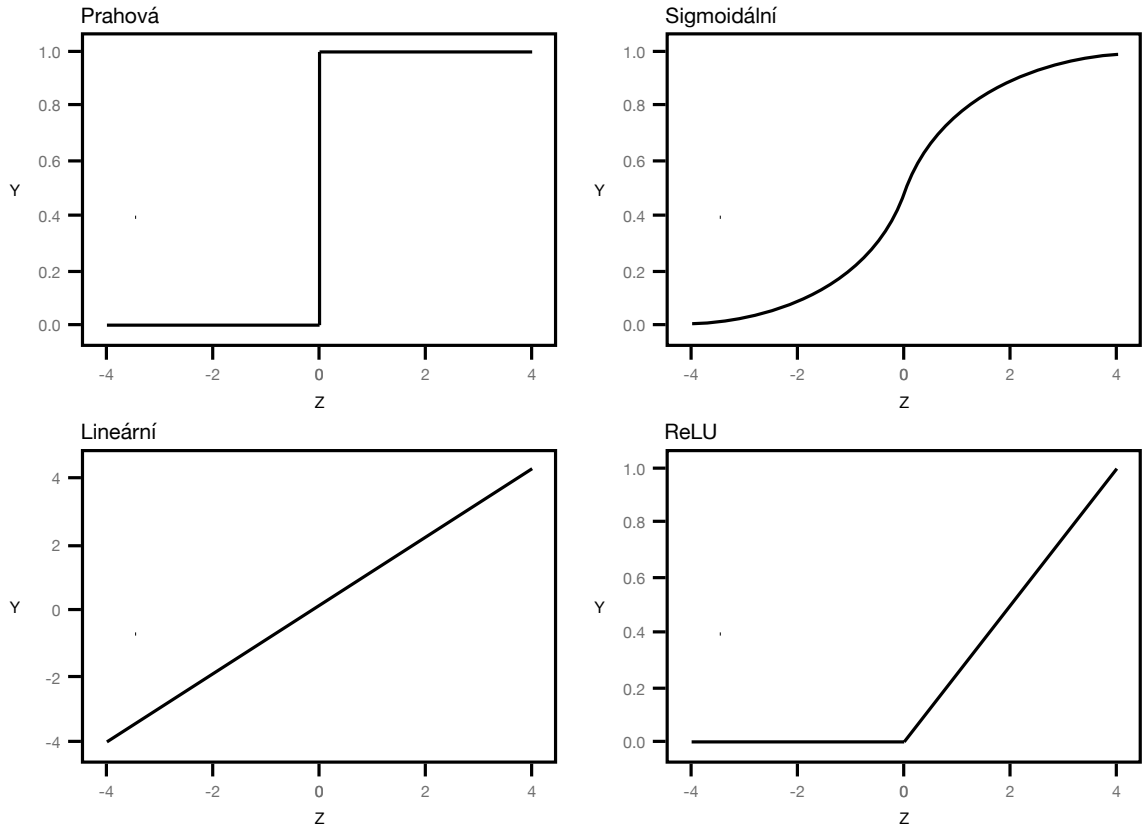
$$\varphi(v) = \begin{cases} va & \text{pokud } v > 0 \\ 0 & \text{pokud } v \leq 0, \end{cases} \tag{2.5}$$

  kde parametr a určuje sklon lineární křivky. Funkce je velmi podobná lineární funkci, nicméně pokud je vnitřní potenciál neuronu záporný, výsledná hodnota je rovna nule.

Aktivačních funkcí existuje velké množství, nicméně při reálném nasazení se využívají pouze základní typy [3, 4, 5].

2.3 Bias

Model neuronové sítě obsahuje rovněž bias (označený symbolem θ). Bias dokáže zvýšit či snížit vnitřní potenciál neuronu a tím posunout aktivační funkci. Jedná se o externí parametr neuronu, který není závislý na žádném jiném parametru sítě. Z hlediska zapojení se v podstatě jedná o další vstup neuronu, nicméně jeho hodnota je vždy nastavena na 1. Stejně jako všechny vstupy neuronu je bias spojen synaptickou váhou. Bias není povinnou součástí neuronové sítě, může být proto vynechán [3, 4].



Obr. 2.2: Základní aktivační funkce [5].

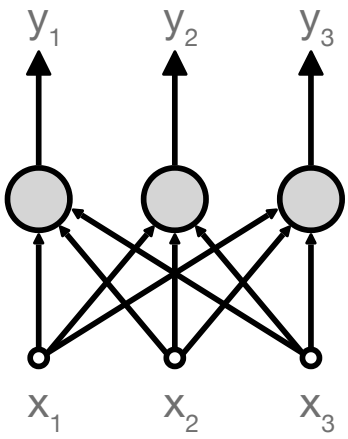
2.4 Architektura sítě

Použití jednoho neuronu, i když má mnoho vstupnů, nemusí být dostačující. Z tohoto důvodu se používá několik neuronů paralelně pracujících nezávisle vedle sebe na jedné vrstvě. Na základě těchto vrstev můžeme rozlišit tři typy sítí:

- jednovrstvé sítě,
- vícevrstvé sítě,
- zpětnovazební sítě [3, 4].

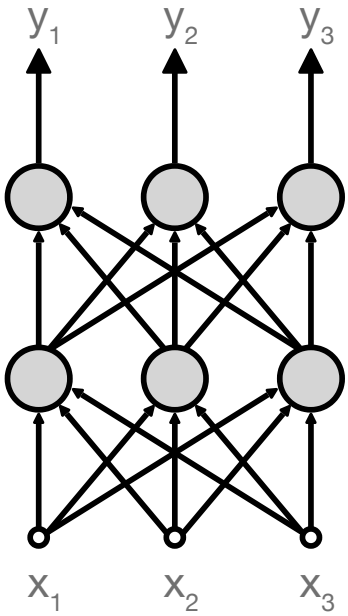
Nejjednodušší formou neuronové sítě jsou jednovrstvé sítě. Jsou dány vstupní uzly, které jsou synaptickými váhami napojeny do neuronů v jedné vrstvě. Tento typ sítě je striktně dopředný, není tedy využito zpětné vazby. Příklad jednovstvé architektury je znázorněn na obrázku 2.3 [4].

Dalším typem dopředné sítě je vícevrstvá síť, která se odlišuje použitím jedné nebo více skrytých vrstev, jejíž uzly jsou rovněž nazvány skrytými neurony. Skryté vrstvy nejsou přímo viditelné ze vstupní či výstupní vrstvy. Jejich přidáním lze docílit získání jemnějších informací ze vstupu. Výstupy první vrstvy jsou použity jako vstupy pro následující skrytou vrstvu, a výstupy této vrstvy jsou zas použity



Obr. 2.3: Jednovrstvá architektura neuronové sítě [4].

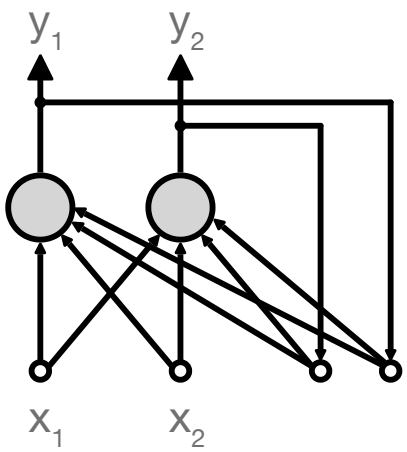
jako vstupy následující vrstvy. Pokud je každý výstup vrstvy použit jako vstup pro každý uzel vrstvy následující, hovoříme o plně propojené neuronové síti. Pokud ovšem určité propojení chybí, jde o částečně propojenou síť. Příklad této architektury je znázorněn na obrázku 2.4 [4].



Obr. 2.4: Vícevrstvá architektura neuronové sítě [4].

Zpětnovazební sítě jsou sítě, jež obsahují zpětnou smyčku. Určité výstupy neuronů mohou být použity jako vstupy jiných neuronů stejné či jiné vrstvy. V některých případech mohou být tyto sítě výkonnější než sítě dopředné a mohou jevit dočasné známky nelineárního chování. Tento typ sítě je hojně využíván ve strojovém překladu

nebo například rozpoznání řeči. Příklad zpětnovazební architektury je znázorněn na obrázku 2.5 [3, 4].



Obr. 2.5: Zpětnovazební architektura neuronové sítě [4].

2.4.1 Dropout

Technika dropout (*výpadek*) je založena na vynechání náhodných uzlů neuronové sítě. Při každé iteraci jsou vynechány jiné uzly sítě. Dropout se využívá výhradně ve fázi učení, a to z důvodu předcházení over-fittingu. Díky dropoutu získává neuronová síť robustnější vlastnosti, které jsou dobře využitelné s různými podmnožinami ostatních neuronů. Ačkoli při zavedení dropoutu je nutno použít zhruba dvakrát více iterací pro konvergenci sítě, čas učení každé epochy je výrazně snížen [5, 6].

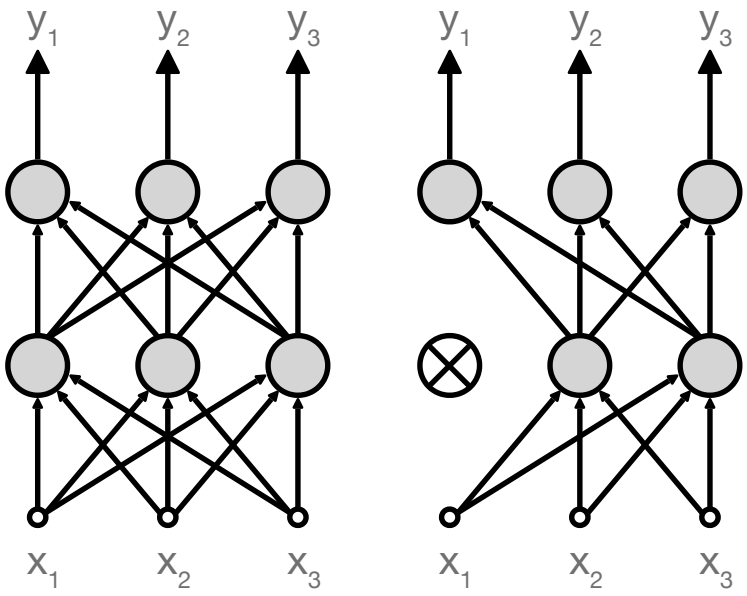
2.5 Proces učení

Učení umělé neuronové sítě není snadnou úlohou. Každá malá změna hodnoty synaptické váhy se neprojeví pouze v následujícím neuronu, ale ve všech navazujících vrstvách a tudíž i ve výstupu celé sítě. Při učení sítě není možné hledat optimální hodnotu jedné synaptické váhy. Na umělou neuronovou síť je nutno nahlížet jako na komplexní soustavu vah, u kterých je nutno najít nejvhodnější kombinaci.

Obecně lze proces učení neuronových sítí kategorizovat na učení s učitelem a učení bez učitele.

2.5.1 Učení s učitelem

Učení s učitelem je znázorněno na schématu 2.7. Často bývá také označováno jako učení pod dohledem (*supervised learning*). Učitel je v tomto případě entita dis-



Obr. 2.6: Porovnání klasické neuronové sítě (vlevo) se sítí využívající dropout (vpravo) [6].

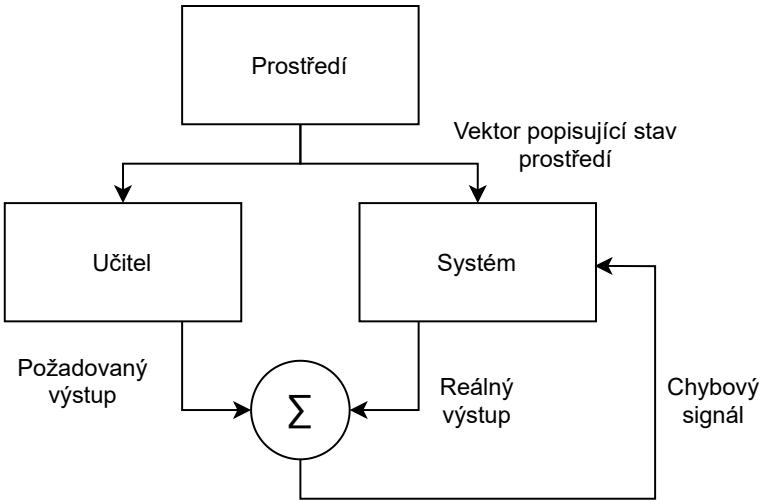
ponující znalostí prostředí, která je reprezentována sadou vstupů a odpovídajících výstupů. Pokud jsou učitel i neuronová síť vystaveni vektoru, který dané prostředí popisuje, učitel dokáže poskytnout neuronové síti požadovaný výstup vektoru. Tento požadovaný výstup reprezentuje optimální hodnotu, která by měla být výstupem neuronové sítě. Na základě chybového signálu a vstupního vektoru jsou parametry sítě upraveny. Chybový signál je definován jako rozdíl mezi požadovaným výstupem a reálným výstupem sítě. Znalost prostředí, kterou disponuje učitel, je tedy přenesena do neuronové sítě pomocí učícího procesu a uložena do formy fixních hodnot synaptických vah. Učení je ukončeno v momentě, kdy síť dosáhne požadované přesnosti, nebo je vyčerpán počet iterací [3].

2.5.2 Učení bez učitele

U metody učení bez učitele je možné rozeznat dvě kategorie – zpětnovazební učení (*reinforcement learning*) a učení bez dohledu (*unsupervised learning*).

Zpětnovazební učení

Principem zpětnovazebního učení je opakovaná interakce s prostředím za účelem získání co nejvyšší výkonnosti sítě. Zatímco učení s učitelem má jistý klíč, dle kterého na základě vstupu zná správný výstup, při zpětnovazebním učení neexistuje dataset a učení se tedy provádí na základě vlastních zkušeností z interakce s okolím [3].



Obr. 2.7: Blokové schéma učení s učitelem [3].

Učení bez dohledu

Při určitých úlohách jsou dána vstupní trénovací data, ke kterým nejsou známy odpovídající výstupy. Cílem učení bez dohledu může poté být nalezení určitých skupin s jistou mírou podobnosti (shlukování) nebo nalezení vzoru v rozpoložení dat v prostoru (odhad hustoty). Tento typ učení je výhodné použít pokud je dán obrovský dataset, který je obtížné anotovat, nebo vůbec není znám počet tříd, do kterých jsou data rozdělena. Učení bez dohledu je mnohem náročnější v porovnání s učením s dohledem a rovněž není zaručeno, že naučený systém vyhodnocuje vstup smysluplně. Mezi učení s učitelem a učení bez učitele může být také zařazeno učení s částečným dohledem (*semi-supervised learning*), při kterém figuruje učitel, který ovšem poskytuje nekompletní informace [3, 7].

2.6 Metrika mAP

V oblasti počítačového vidění je mean average precision (mAP) populární metrikou k vyhodnocení správnosti lokalizace a klasifikace objektu. Mnoho detekčních modelů, jako například Faster R-CNN, YOLO nebo MobileNet SSD použilo mAP jako hlavní metriku při publikaci výzkumů. S metrikou mAP jsou úzce spjaty pojmy *precision* a *recall*.

Precision (*přesnost*) je v oblasti statistiky vnímána jako poměr počtu skutečně pozitivních (TP - *true positive*) predikcí k celkovému počtu pozitivních predikcí, tedy skutečně pozitivních a falešně pozitivních (FP - *false positive*) predikcí dohromady:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{2.6}$$

Z výše uvedené rovnice je zřejmé, že k dosažení co nejvyšší přenosti je nutné snížit výskyt falešně pozitivních predikcí, čímž dojde ke snížení hodnoty recall.

Recall (*odvolání*) je definován jako poměr skutečně pozitivních predikcí k součtu skutečně pozitivních a falešně negativních (FN - *false negative*) predikcí:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{2.7}$$

Oba parametry – precision i recall – jsou běžně využívány s dalšími metrikami jako například F1-score, TNR (*true negative rate*) nebo ROC (*receiver operating characteristics*).

Metrika mAP je vypočítána jako průměr N sad predikcí:

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^{N} \text{AP}_i, \tag{2.8}$$

přičemž AP (*average precision*) je možné odvodit jako plochu pod křivkou parametru recall dané sady predikcí:

$$\text{AP} = \int_0^1 p(r)dr, \tag{2.9}$$

kde p označuje parametr precision.

2.7 Metrika F-score

F-score (nebo také F1-score) slouží k určení přesnosti daného datasetu. Nejčastěji je používána k vyhodnocení binárních klasifikačních systémů, které klasifikují dané vstupy jako pozitivní nebo negativní hodnotu. Rovněž může být použita k vyhodnocení vyhledávacích enginů, mnoho typů modelů strojového učení nebo v určitých systémech zpracování řeči. Metrika F-score je definována jako harmonický průměr parametrů modelu precision a recall. Jde o způsob zkombinování těchto dvou parametrů. Ideální model dosahuje hodnoty F-score rovno jedné. Matematické vyjádření metriky F-score vypadá následovně:

$$\text{F-score} = 2 \times \frac{\text{precission} \times \text{recall}}{\text{precission} + \text{recall}} = \frac{2}{\frac{1}{\text{recall}} \times \frac{1}{\text{precision}}}. \tag{2.10}$$

F-score je možno upravit tak, aby byl kladen větší důraz na parametr precission než na parametr recall, nebo naopak. Nejznámější úpravy metriky jsou F0.5-score a F2-score. Upravená rovnice má následující tvar:

$$\text{F}_\beta = (1 + \beta^2) \times \frac{\text{precission} \times \text{recall}}{(\beta^2 \times \text{precission}) + \text{recall}}, \tag{2.11}$$

kde parametr β udává důležitost recall oproti precision. Běžné F-score má parametr β roven jedné, F2-score má parametr rovno dvěma [26].

3 Konvoluční neuronové sítě

V posledních letech dosáhly konvoluční neuronové sítě významných pokroků při řešení náročných úloh v oblasti počítačového vidění, jako je klasifikace, detekce nebo segmentace na základě obrazových dat. Aplikace modelů konvolučních neuronových sítí v reálném světě jsou stále častější a jejich využití stále roste [9].

3.1 CNN

Konvoluční neuronová síť (*Convolutional Neural Network* – CNN) je vícevrstvá neuronová síť speciálně navržená k rozpoznání tvarů ve dvourozměrném prostředí, přičemž je vysoce invariantní vůči škálování, zkosení, rotaci a jiným formám zkreslení. Jde o nejjednodušší způsob detekce objektů v obraze s využitím hlubokého učení. U konvoluční neuronové síťě není vyžadováno, ve srovnání s jinými algoritmy, tak velké předzpracování vstupních obrazových dat. Obecné schéma konvoluční neuronové sítě může obsahovat následující vrstvy, z nichž každá má definovanou určitou funkci:

- konvoluční vrstva,
- pooling vrstva,
- normalizační vrstva,
- plně propojená vrstva,
- výstupní vrstva [3, 4].

Použití CNN je velmi efektivní, pokud je objekt obsažen ve značné části obrázku. V reálném světě jsou ovšem objekty umístěny na různých pozicích a v odlišném poměru stran. K vyřešení tohoto problému by bylo zapotřebí obrovské množství oblastí. Z tohoto důvodu byly vytvořeny algoritmy jako například R-CNN nebo YOLO [11].

3.1.1 Konvoluční vrstva

Jednou z hlavních komponent konvolučních neuronových sítí je konvoluční filtr (nebo také kernel). Jedná se o matici o rozměrech $A_x \times A_y$, kde A většinou zastupuje liché číslo (typicky 3 nebo 5) [5]. Konvoluční filtr K, použitý k detekci horizontálních hran může být definován touto maticí:

$$K = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix}. \tag{3.1}$$

Konvoluce

Výstup filtru aplikovaný na předchozí vrstvu se nazývá matice příznaků (*feature map*). Pokud je konvoluce prováděna nad vstupní vrstvou, jako vstup budou použity hodnoty pixelů, při provádění konvoluce v hlubších vrstvách je jako vstup použita předchozí matice příznaků [20].

Provedení konvoluce pomocí konvolučního filtru $K(m,n)$ nad vstupem $I(i,j)$ může být vyjádřeno následujícím vzorcem:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)[20]. \tag{3.2}$$

Krokování

Parametr krok (*stride*) definuje velikost kroku konvolučního filtru. Pokud je hodnota parametru rovna jedné, konvoluční filtr je posouván po vstupní matici vždy o jeden pixel. Pokud je hodnota krokování vyšší než jedna, dochází ke zmenšení výstupní matice, a to dle vzorce:

$$N_B = \frac{N_A - N_K}{S} + 1, \tag{3.3}$$

kde N_A označuje rozměr vstupní matice, N_B rozměr výstupní matice, N_K rozměr konvolučního filtru a S velikost krokování (*stride*) [5].

Nulové ohraničení

Při metodě, zvané nulové ohraničení (*zero-padding*), jsou na vstupní matici přidány okraje s nulovou hodnotou. Díky ohraničení lze ovlivnit prostorové rozměry výstupní vrstvy a lze tedy dosáhnout stejných rozměrů vstupní a výstupní matice, což je často žádoucí při práci s obrázkem. Výsledný rozměr výstupní matice po použití nulového ohraničení je dán vzorcem:

$$N_B = \frac{N_A - N_K + 2P}{S} + 1, \tag{3.4}$$

kde P označuje velikost nulového ohraničení [5].

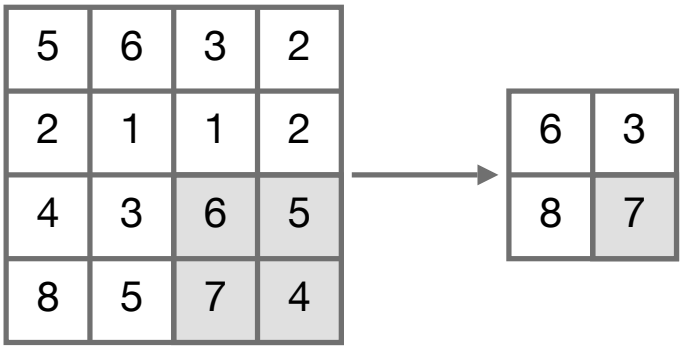
3.1.2 Podvzorkování

Podvzorkování (*pooling*) snižuje výpočetní náročnost, počet parametrů a komplexnost zmenšením dimenze konvoluční funkce. Je běžnou praxí prokládat konvoluční vrstvy s podvzorkovací vrstvou. Nejběžnější formou podvzorkování je využití konvolučního filtru o rozměru 2×2 k odstranění 75 % aktivací. Podvzorkování může rovněž sloužit k odstranění šumu. Častým problémem výstupu konvolučních vrstev je vysoká citlivost na pozici příznaků ve vstupu. Tento problém řeší právě podvzorkovací vrstva pomocí snížení dimenze (*downsampling*) [21].

Používají se dva typy podvzorkování:

- **max pooling** – výsledná hodnota je maximální hodnota z čísel v aktuální oblasti,
- **average pooling** – výsledná hodnota je průměrná hodnota z čísel v aktuální oblasti.

Jako důležité parametry podvzorkovací vrstvy mohou být uvedeny **stride** – velikost kroku a **pool size** – velikost filtru. Příklad max poolingu s parametry stride 2 a pool size 2 je uveden na následujícím obrázku:



Obr. 3.1: Příklad max poolingu.

Podvzorkovací vrstvy běžně nevyužívají nulové ohraničení [21].

V poslední době roste trend ve využívání konvolučních sítí, které omezují nebo dokonce vůbec nevyužívají podvzorkovací vrstvy. Namísto podvzorkovacích vrstev je použita architektura, ve které se vyskytují pouze konvoluční vrstvy, přičemž některé vrstvy nahrazují funkci podvzorkovacích vrstev zavedením krokvání (*stride*) [21].

3.1.3 Normalizace

Dávková normalizace (*batch normalization*) standardizuje vstupy do dané vrstvy (tj. aktivace z předešlých vrstev). Standardizace vstupů znamená stav, kdy mají vstupy do dané vrstvy přibližně nulovou střední hodnotu a rozptyl. Každá vrstva ovšem nemusí pracovat nejlépe při normalizaci kolem nulové střední hodnoty, ale může požadovat jinou průměrnou střední hodnotu. Z tohoto důvodu má normalizace dávek dva naučitelné parametry γ a β, které střední hodnotu modifikují. U každé normalizace dávky se vypočítá průměr a rozptyl této funkce v minidávce. Poté se odečte průměr a vydělí vlastnost mini-dávkovou standardní odchylkou. Normalizace často poskytuje mnohem stabilnější a rychlejší učící proces sítě. Transformace vstupu x z mini-dávky pomocí dávkové normalizace B na výstup y je popsána následujícími vzorci:

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i, \tag{3.5}$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}(x_i - \mu_B)^2, \tag{3.6}$$

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}, \tag{3.7}$$

$$y_i = \gamma\hat{x}_i + \beta = \mathrm{BN}_{\gamma,\beta}(x_i), \tag{3.8}$$

kde μ_B udává střední hodnotu mini-dávky, σ_B^2 rozptyl mini-dávky a $\hat{x}_i$ normalizovanou hodnotu. Normalizace se nepoužívá u rekurentních neuronových sítí. Implementace by byla velmi komplexní, jelikož by rekurentní neuronová síť vyžadovala odlišný parametr γ a β pro každý časový úsek v normalizační vrstvě [29].

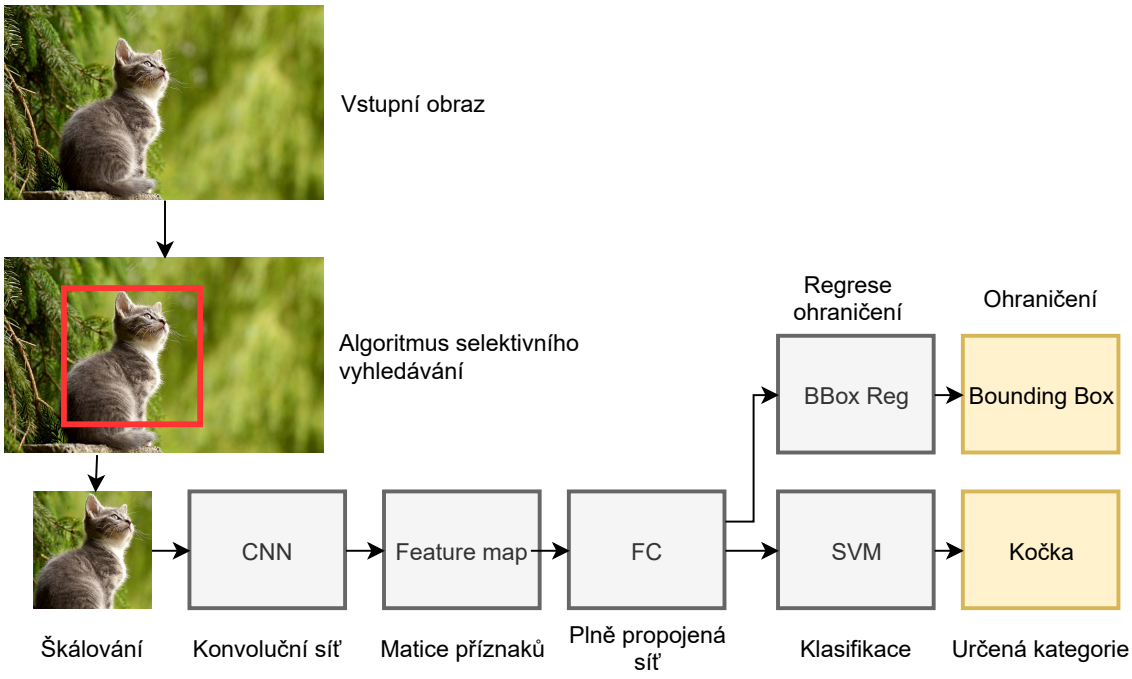
3.2 R-CNN

Namísto práce s obrovským množstvím oblastí jako v případě CNN, algoritmus konvoluční neuronové sítě založené na oblastech (*Region-Based Convolutional Neural Network* – R-CNN) navrhuje několik oblastí zájmu (*Region of Interest* – RoI) v obrázku a provádí konvoluci v těchto regionech. K získání navržených oblastí využívá R-CNN algoritmus selektivního vyhledávání (*selective search*). Navržené oblasti zájmu jsou škálovány na fixní velikost a použity jako vstup do konvoluční sítě, která produkuje matici příznaků. Tato matice příznaků je předána SVM (*Support Vector Machine*), na základě kterého je klasifikována přítomnost objektu. V posledním kroku je rovněž provedena predikce offsetu oblasti ke korekci a zpřesnění umístění bounding boxu [10].

Algoritmus R-CNN přináší mnoho nevýhod, jako například časovou náročnost. Učení neuronové sítě je velmi zdlouhavé, jelikož v jednom obrázku je nutno klasifikovat až 2000 oblastí zájmu. Z tohoto důvodu je v podstatě vyloučeno rozpoznání v reálném čase. Algoritmus selektivního vyhledávání je pevný, nepodléhá žádnému učení. S postupem času se tedy nenaučí navrhnovat pouze správné kandidáty. Obrázek 3.2 znázorňuje blokové schéma R-CNN [10, 11].

3.2.1 Selektivní vyhledávání

Algoritmus selektivního vyhledávání (*selective search*) využívá základní aspekty, které formují obraz, jako jsou barva, textura, velikost a tvar. Na základě těchto aspektů jsou části obrázku shlukovány a postupně jsou navrženy oblasti zájmu, které definují oblast potencionálního výskytu objektu. Algoritmus byl navržen jako náhrada za velmi neefektivní a výpočetně náročný algoritmus posuvného okna (*sliding window*) a pyramidového škálování (*image pyramids*) [10].



Obr. 3.2: Blokové schéma R-CNN [11].

Často vzniká mylná představa, že algoritmus selektivního vyhledávání je jakousi náhražkou za detekci objektů, nicméně tento algoritmus vytváří pouze bounding boxy, které nejsou přiřazeny žádné kategorii objektů. Rovněž není zaručeno, že bounding boxy opravdu daný objekt obsahují.

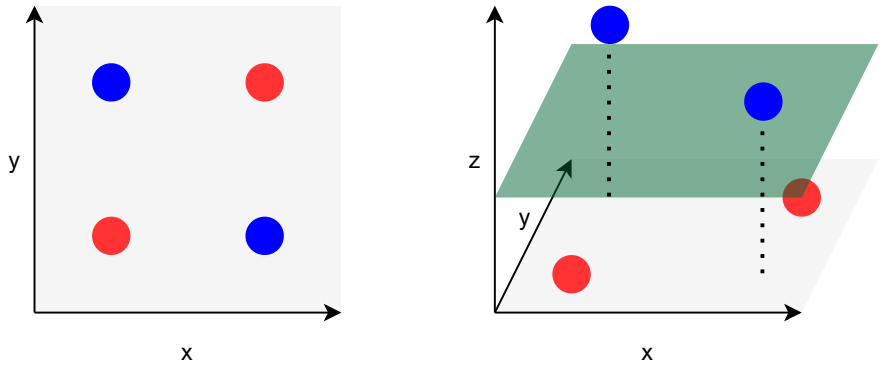
Neexistuje pouze jedna optimální strategie, kterou by bylo možno spolehlivě RoI selektovat. Nežádoucí jevy, jako je stínování objektu, mohou zvolení regionu na základě jednotné barvy velmi zkomplikovat. Z tohoto důvodu je nutné zkombinovat různé strategie a využít více typů aspektů k dosažení požadovaného shluku pixelů.

3.2.2 Algoritmus podpůrných vektorů

Hlavní úlohou algoritmu podpůrných vektorů (SVM – *Support vector machine*) je nalezení nadroviny (angl. *hyperplane*) v multi-dimenzionálním prostoru, která jednoznačně dokáže klasifikovat datové body. Pokud máme dvourozměrný prostor $\mathbb{R}^2$, lze použít k rozdělení přímku. V trojrozměrném prostoru $\mathbb{R}^3$ se jedná již o rovinu. Ačkoliv v praxi se jedná o prostory, které mají mnohem více rozměrů než tři, je velmi obtížné si tyto prostory představit [23, 24].

Nadrovina je v podstatě hranice rozhodnutí, která pomáhá s rozhodováním příslušnosti k třídě bodů. Při vytváření SVM je požadováno vytvoření takové nadroviny, která má největší možné odsazení od datových bodů všech kategorií. Pokud dané body nelze v použité dimenzi rozdělit, je možno převést prostor do dimenze

vyšší a klasifikovat datové body v nové dimenzi. Obrázek 3.3 demonstruje vytvoření nadroviny pomocí převedení prostoru z $\mathbb{R}^2$ dimenze do dimenze $\mathbb{R}^3$ [23, 24].



Obr. 3.3: Nalezení nadroviny pro klasifikaci datových bodů ve vyšší dimenzi prostoru [23].

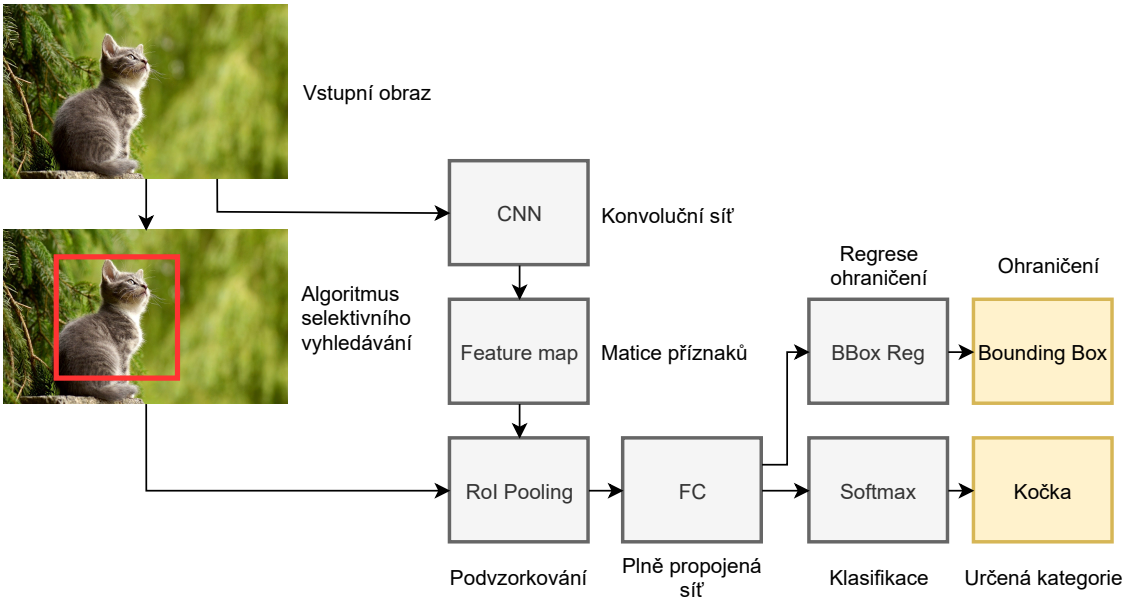
3.3 Fast R-CNN

K vyřešení některých nevýhod modelu R-CNN byl stejným autorem navrhnut model Fast R-CNN, který nabízí rychlejší detekci objektů. Na rozdíl od R-CNN modelu, kde je každý RoI zaslán do konvoluční sítě, je v případě Fast R-CNN zaslán do konvoluční sítě celý vstupní obrázek, na jehož základě je vygenerována matice příznaků. Z matice příznaků vstupního obrazu jsou identifikovány RoI, které jsou následně škálovány na stejný poměr stran. Následuje použití RoI pooling vrstvy, díky které se matice příznaků výrazně sníží a její velikost je vždy fixní. Tato matice je zaslána do plně propojené sítě a následně je použit softmax k predikci třídy objektu a také je vyhodnocen bounding box [22].

Model přináší výrazné vylepšení výkonu, jelikož není nutné provádět konvoluci nad každým RoI (kterých může být až 2000). Konvoluce je prováděna pouze dvakrát za celý obrázek. Cvičení sítě upravuje vždy všechny váhy sítě. V porovnání s R-CNN je učící proces síťě Fast R-CNN až 10 krát rychlejší při použití stejných hardwarových prostředků a výsledná kvalita detekce je vyšší. Obrázek 3.4 zobrazuje blokové schéma Fast R-CNN [22].

3.3.1 Softmax klasifikace

Klasifikace softmax (nebo také softargmax) převádí vektor K reálných čísel na vektor reálných čísel, jejichž součet je roven 1. Ať je vstupní hodntota kladná, záporná, nulová či nenulová, výstup je vždy transformován do intervalu $(0, 1)$. Tento krok je



Obr. 3.4: Blokové schéma Fast R-CNN [11].

velmi podstatný pro detekci objektů, jelikož výstupem sítě jsou reálné hodnoty bez určeného měřítka. Pomocí softmax funkce lze tyto reálné hodnoty transponovat na normalizované rozdělení pravděpodobnosti. Z tohoto důvodu je ve většině případů funkce softmax použita jako poslední krok klasifikačních úloh.

Matematicky je funkce softmax vyjádřena následujícím vzorcem:

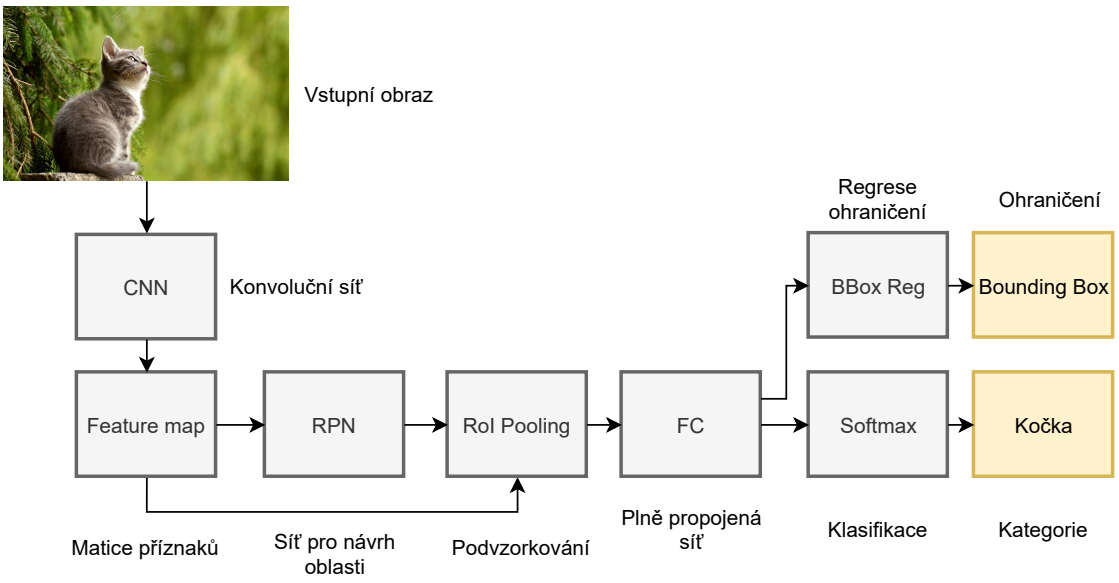
$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, \tag{3.9}$$

kde σ označuje softmax funkci, $\vec{z}$ označuje vstupní vektor hodnot, e^{z_i} standardní exponencionální funkci pro vstupní vektor, e^{z_j} standardní exponencionální funkci pro výstupní vektor a K počet tříd klasifikátoru [8].

3.4 Faster R-CNN

Oba předchozí modely sítí užívají k nalezení oblastí zájmů algoritmus selective search. Jde o výpočetně a časově náročný proces, který ovlivňuje výkon celé sítě. Z tohoto důvodu byl v architektuře Faster R-CNN selective search nahrazen za RPN (*Region Proposal Network)*. Obdobně jako ve Fast R-CNN je celý vstupní obraz zaslán do konvoluční sítě, nicméně namísto použití selective search pro určení polohy objektu je využito sítě RPN. Oblasti zájmu jsou poté škálovány pomocí RoI Pooling vrstvy. Konečný proces klasifikace objektu a upřesnění bounding boxu jsou stejné jako v architektuře Fast R-CNN. Za použití těchto úprav je rychlost algoritmu až 10 × rychlejší v porovnání s Fast R-CNN, díky čemuž lze Faster R-CNN

použít v aplikacích, které vyžadují klasifikaci objektů v reálném čase. Schéma Faster R-CNN je zobrazeno na obrázku 3.5 [11].



Obr. 3.5: Blokové schéma Faster R-CNN [11].

3.4.1 Síť pro návrh regionu

Síť pro návrh regionu (RPN – *Region proposal network*) dostává jako vstup matici příznaků. Výstupem sítě jsou návrhy oblastí objektů, přičemž u každého návrhu je uvedena i procentuální spolehlivost výskytu objektu. Ohraničení není predikováno jako konkrétní oblast, ale jako relativní odsazení k předdefinovaným oblastem, které se nazývají Anchor boxes (*kotevní boxy*). Anchor boxes je jeden z nejdůležitějších konceptů v Faster R-CNN. Tento koncept zodpovídá za poskytnutí předdefinovaných sad boxů, které mají různou velikost a poměr stran – typicky 1:1, 2:1 a 1:2. Boxy jsou určeny jako reference při určování polohy objektu. Ačkoliv RPN pracuje s maticí příznaků, anchor boxy jsou vytvořeny vzhledem k proporcím původního obrázku. Rozprostření anchor boxů je realizováno rovnoměrně přes celý vstupní obrázek [25].

Původní verze RPN používá tři velikosti a tři poměry stran. Součin počtu velikostí a poměrů stran určuje počet anchor boxů pro daný bod (parametr k). Celkový počet vytvořených anchor boxů lze určit dle vzorce:

$$a = WHk, \tag{3.10}$$

kde W je šířka matice příznaků, H výška matice příznaků, k součin počtů poměrů stran a velikostí anchor boxů pro každý bod. Hlavním důvodem k použití anchror boxes je fakt, že dochází k vyhodnocení všech predikcí zároveň. RPN přináší výrazné zrychlení, díky kterému je možná detekce objektů v reálném čase [25].

3.5 YOLO

Algoritmus YOLO (z angl. *You Only Look Once*) slouží k detekci objektů a využívá odlišný přístup než algoritmy založené na oblastech zájmu. Název je odvozen z principu algoritmu, kdy je vstupní obraz zaslán do jediné konvoluční neuronové sítě, která je schopna vytvořit jak ohraničení objektů, tak pravděpodobnost třídy objektu v každé oblasti obrázku. Algoritmus vyniká nízkou časovou náročností v porovnání s jinými detekčními algoritmy. Společně s Faster R-CNN je YOLO vhodný pro aplikace, kde je nutné zpracovávat obraz v reálném čase, jelikož je schopen vyhodnotit až 140 obrázků za sekundu (YOLOv5, 2020). Algoritmus není vhodný pro detekci menších objektů kvůli prostorovým omezením vyplývajícím z principu algoritmu [11].

V současné době je k dispozici mnoho verzí algoritmu YOLO. První verze byla publikována roku 2016 Josephem Redmonem. Rok poté vyšla další verze, která byla označena jako YOLO9000 (nebo také YOLOv2). Algoritmus YOLOv2 byl schopen zpracovat 40–90 snímků za sekundu. V roce 2018 vyšela verze YOLOv3, která vynikala možností snadné změny velikosti modelu. Změnou modelu je možno regulovat poměr mezi přesností a rychlostí zpracování snímků. Díky velmi vysoké přesnosti a skvělé rychlosti se v oblasti počítačového vidění model YOLO velmi rychle rozšířil. V roce 2020 oznámil Joseph Redmon ukončení dalšího vývoje a výzkumu YOLO kvůli obavám ze zneužití algoritmu k válečným účelům. Ukončení vývoje nicméně nezastavilo další nadšence počítačového vidění, a tak v roce 2020 vyšly verze YOLOv4 i YOLOv5. Je nutno podotknout, že tyto verze již nejsou oficiální, ale jde o volné rozšíření YOLOv3 algoritmu. Verze YOLOv4 přináší až 10% nárůst přenosti ve srovnání s verzí YOLOv3. Verze YOLOv5 je odlišná od předchozích verzí, jelikož již nevyužívá síť Darknet.

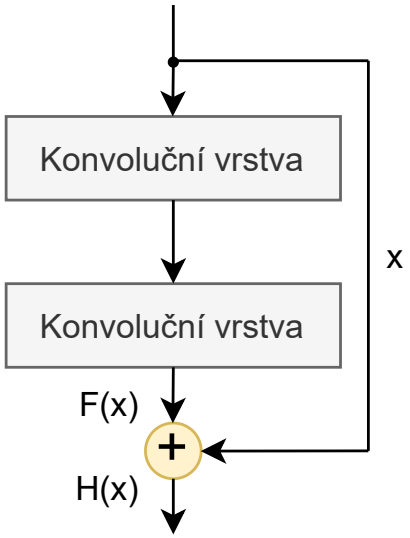
Darknet je framework pro neuronové sítě s otevřeným zdrojovém kódem, který je napsán v C a CUDA. Podporovány jsou výpočty na CPU i GPU. V algoritmu YOLO9000 bylo využito sítě Darknet-19, která obsahuje 19 konvolučních vrstev. YOLOv3 již používá Darknet-53 obsahující 53 konvolučních vrstev.

3.6 ResNet

ResNet (*Residual Network*) je velmi výkonný páteřní model, který je často využíván v úlohách týkajících se počítačového vidění. Dopředná síť o jedné vrstvě může reprezentovat jakoukoliv funkci, nicméně taková vrstva může být velmi velkých rozměrů a je náchylná na nadměrné vybavení (*overfitting*). Z tohoto důvodu je na vzestupu trend v přidávání počtu vrstev a používání hlubokých neuronových sítí. Pouhé zvyšování hloubky sítě ovšem není účinné. Učení hlubokých sítí je komplikované kvůli

známému problému mizejícího přechodu (*vanishing gradient problem*), kdy při použití algoritmu zpětného šíření chyby může opakované násobení přechod nekonečně zmenšit. Při zvýšení hloubky sítě může být tedy pozorována degradace výkonu sítě.

ResNet představil nový nápad spočívající ve vytvoření zkratky mezi vrstvami (*idenitity shortcut connection*), díky které je možné učit mnohem hlubší sítě než kdy dříve. Toto spojení umožňuje přeskočit jednu či více vrstev sítě. Reziduální blok je zobrazen na obrázku 3.6. Zkratka nemá žádné parametry a pouze přidává



Obr. 3.6: Reziduální blok použitý v modelu ResNet.

výstup předešlé vrstvy do vrstvy cílové. Pokud nemá x a $\mathrm{F}(x)$ stejné rozměry (kvůli chybějícímu paddingu při konvoluci), je mapování identity vynásobeno lineární projekcí W tak, aby bylo možné zkombinovat zkratku x a výstup vrstvy $\mathrm{F}(x)$. Výstup residuálního bloku je vypočítán následujícím vzorcem:

$$\mathrm{H}(x) = \mathrm{F}(x) + x, \tag{3.11}$$

kde $\mathrm{H}(x)$ označuje výstup residuálního bloku, $\mathrm{F}(x)$ výstup konvoluční vrstvy a x mapování identity odstraňující dvě konvoluční vrstvy. Podobný princip použil již dříve model *Highway Network* nebo *Long Term Short Memory (LSTM)*, nicméně tyto modely nemají tak uspokojivé výsledky, jako právě ResNet. V případě *Highway Network* jsou zkratky ovlivněny parametrickými branami, které určují, kolik informace projde přes spojení zkratky. V jistých případech může dojít k zavření parametrických bran, což vede k vytvoření nereziduální funkce. V případě ResNet nemůže dojít k zavření zkratek. Díky skvělým výsledkům se z ResNet modelu stal jeden z nejpoužívanějších modelů pro úlohy počítačového vidění [27, 28].

4 Segmentační neuronové sítě

Můžeme rozlišit tři typy analýzy obrazových dat:

- **klasifikace** – přiřazení obrázku do určité třídy, jako například „strom“ nebo „člověk“,
- **detekce** – detekování objektu ve vstupním obrazu a vykreslení jeho ohraničení,
- **segmentace** – přiřazení každé části obrázku k určité kategorii [14].

4.1 Původní segmentační metody

Před příchodem hlubokých neuronových sítí byly běžně používány segmentační metody, které využívaly rigidní algoritmy. Vyznačovaly se nižší spolehlivostí segmentace ve srovnání s nynějšími metodami založenými na hlubokých neuronových sítích. Mezi tyto metody patří thresholding (prahování), K-means clustering (shlukování), detekce hran nebo obrazová segmentace s využitím histogramu [14].

4.1.1 Prahování

Jednou z nejjednodušších segmentačních metod je prahování (*thresholding*). Princip metody spočívá v určení správné hodnoty prahu (*threshold*) k segmentaci obrazu. Na základě určeného prahu jsou všechny pixely v obrazu klasifikovány jako objekt, či pozadí. Rozlišujeme thresholding s jednou prahovou hodnotou (*global thresholding*) nebo s více prahovými hodnotami (*local thresholding*) [13, 14].

Otsuva binarizace

Otsův algoritmus (neboli Otsuva binarizace) slouží k rychlému a automatickému určení správné prahové hodnoty pro daný vstupní obraz. Určení prahu probíhá na základě histogramu černobílého obrazu. Algoritmus Otsovy binarizace zkouší dosadit všechny možné thresholdy t až do nejvyšší maximální hodnoty I. Uvažujeme-li klasický černobílý 8-mi bitový obrázek, bude hodnota t postupně nabývat hodnot z intervalu$(0, 256)$ a maximální hodnota I bude rovna 256 [12, 13].

Otsův algoritmus dosazuje postupně hodnoty prahů t a získává vážený rozpyl v rámci třídy (σ_w^2) pomocí vzorce:

$$\sigma_w^2 = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t), \tag{4.1}$$

kde $q_1(t)$ a $q_2(t)$ jsou pravděpodobnosti tříd a $\sigma_1^2(t)$ a $\sigma_1^2(t)$ rozptyly tříd. Pravěpodobnosti tříd $q_1(t)$ a $q_2(t)$ jsou dány vzorci:

$$q_1(t) = \sum_{i=1}^{t} P(i), \quad q_2(t) = \sum_{i=t+1}^{I} P(i), \tag{4.2}$$

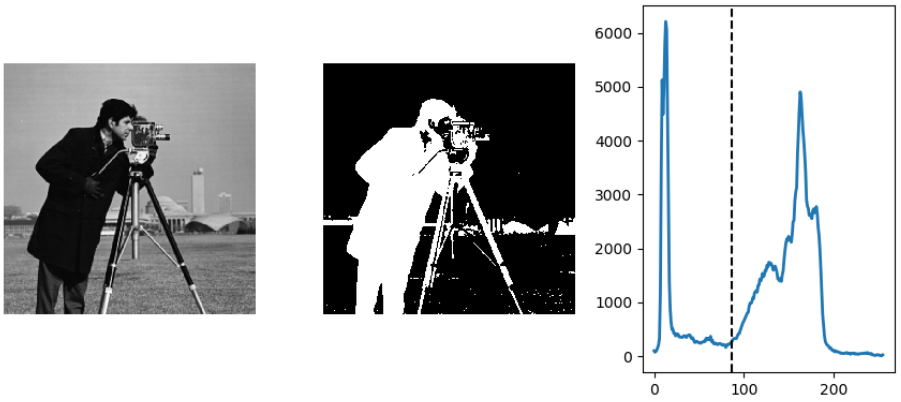
kde P označuje matici hodnot obrázku. Průměrné hodnoty tříd $\mu_1(t)$ a $\mu_2(t)$ jsou získány pomocí vzorců:

$$\mu_1(t) = \sum_{i=1}^{t} \frac{iP(i)}{q_1(t)}, \quad \mu_2(t) = \sum_{i=t+1}^{I} \frac{iP(i)}{q_2(t)} \tag{4.3}$$

Rozptyly tříd $\sigma_1^2(t)$ a $\sigma_1^2(t)$ jsou definovány vzorci:

$$\sigma_1^2(t) = \sum_{i=1}^{t} [i - \mu_1(t)]^2 \frac{P(i)}{q_1(t)}, \quad \sigma_1^2(t) = \sum_{i=t+1}^{I} [i - \mu_2(t)]^2 \frac{P(i)}{q_2(t)}. \tag{4.4}$$

Po průchodu algoritmem je určena optimální hodnota prahu t na základě minimální získané hodnoty váženého rozptylu σ_w^2 [12, 13].



Obr. 4.1: Určení prahu pomocí Otsuvy binarizace, převzato z [30].

4.1.2 Shlukování

Shlukování (*clustering*) je process sjednocování skupin dle jejich atributů. Nejrozšířenější typy shlukování jsou *K-means clustering* a *fuzzy C-means clustering* [16].

K-means shlukování

Prvním krokem algoritmu je zvolení několika náhodných centrálních bodů – tzv. *centroidů*, které jsou použity jako základ shluku – *clusteru*. V dalším kroku algoritmus opakovaně provádí kalkulace pro optimalizaci pozice centroidů. Algoritmus je ukončen, pokud se centroidy stabilizovaly, tudíž nezměnily hodnotu a shlukování bylo tedy úspěšné, nebo pokud byl vyčerpán počet definovaných iterací algoritmu [16, 17].

Cílem K-means je minimalizovat celkový rozptyl mezi clustery, který je definován následujícím vzorcem:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \|x_i^{(j)} - c_j\|^2, \tag{4.5}$$

kde J je výstupem funkce, k počet clusterů, n počet možných případů, x konkrétní případ a c označuje centroid [14, 17].

[16]

4.1.3 Obrazová segmentace pomocí histogramu

Technika využívá obrazový histogram k přiřazení skupin pixelů na základě úrovně šedé. Jednoduché obrázky mohou být složeny z jednotvárného pozadí a objektu. Pozadí má často v histogramu největší zastoupení [14]

4.2 Sémantická segmentace

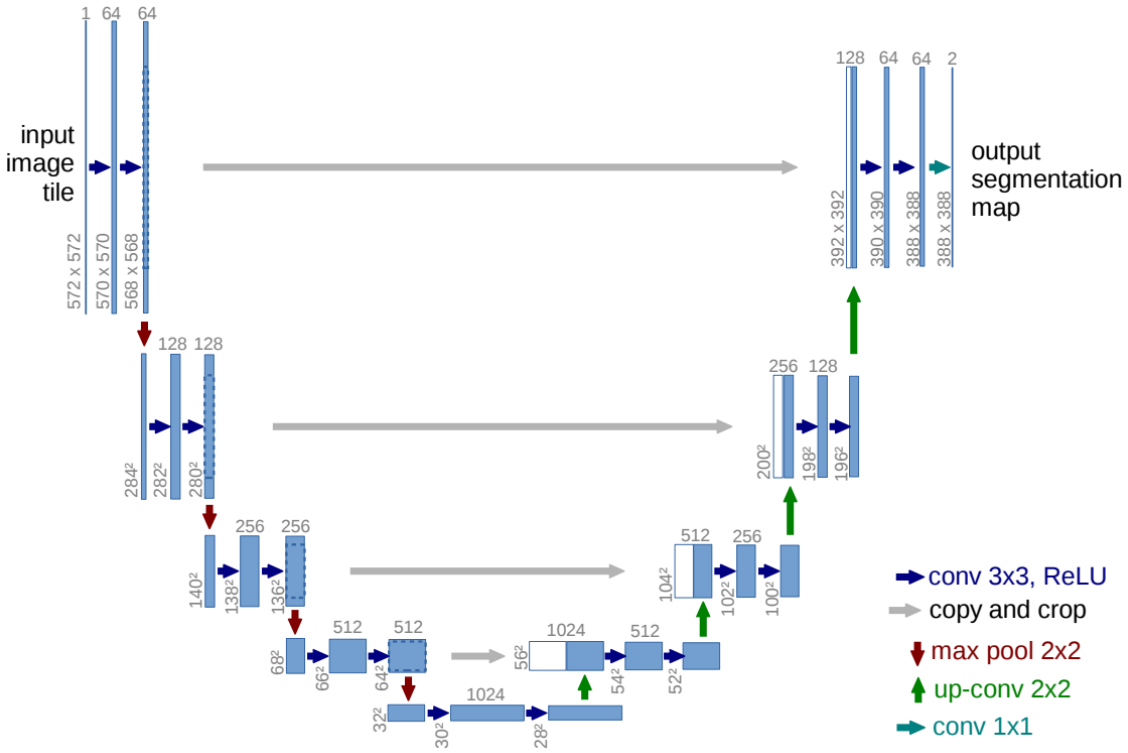
Proces přiřazení každého pixelu vstupního obrazu do určité kategorie se nazvývá sémantická segmentace. Jako příklady sémantických segmentačních sítí mohou být uvedeny: *U-Net*, *FCN*, *DialatedNet*, *DeepLab*, *Deconvnet*, *SegNet* nebo také *PSPNet* [31].

4.2.1 U-Net

Sémantická segmentační síť U-Net získala svůj název podle charakteristického tvaru. Síť se skládá ze dvou hlavních částí – levá smršťovací (*contracting*) část a pravá rozpínavá (*expansive*) část. Schéma architektury sítě je zobrazeno na obrázku 4.2. Celkově provádí sémantická segmentační síť U-Net 23 konvolucí [18].

Levá část sítě se nijak zásadně neliší od klasické architektury konvolučních neuronových sítí. Skládá se z opakovaných aplikací 3×3 konvolucí, které jsou vždy následovány aktivací typu ReLU, následně je použit max pooling s krokem 2 a velikostí filtru 2×2. Během průchodu levou částí sítě je prostorová informace redukována, ale informace v matici příznaků naopak přibývají [31, 18].

Pravá část sítě provádí nadvzorkování matice příznaků a poté 2×2 konvoluci, která redukuje počet příznaků na polovinu. Dále následuje zřetězení s odpovídající ořezanou maticí příznaků z levé části sítě. Po zřetězení je provedena dvojitá konvoluce 3×3, která je opět následována ReLU aktivací. Ořez matice je nutný kvůli ztrátě krajních pixelů při každé konvoluci [31, 18].



Obr. 4.2: Architektura sémantické segmentační sítě U-Net, převzato z[18].

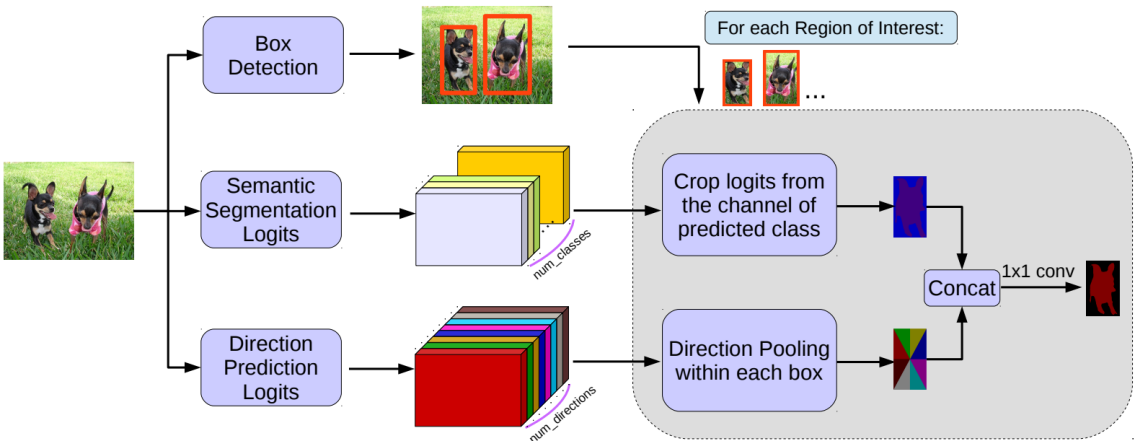
4.3 Instanční segmentace

Instanční segmentace se liší od sémantické segmentace tím, že nepřiřazuje každému pixelu kategorii, ale snaží se odlišit jednotlivé instance kategorií. Pokud jsou na obrázku tři vozidla vedle sebe, instanční segmentace dokáže identifikovat každé vozidlo zvlášť jako samostatný objekt, zatímco sémantická segmentace pouze přiřadí kategorii „vozidlo“. Stejně jako sémantická segmentace, instanční segmentace zakládá na použitelnosti CNN. Jako příklady instančních segmentačních sítí mohou být uvedeny: *MaskLab*, *SDS*, *DeepMask*, *Mask R-CNN*, *InstanceFCN*, *PANet* nebo *TensorMask* [14, 31].

4.3.1 MaskLab

Pro docílení instanční segmentace kombinuje MaskLab prvky sémantické segmentace s technikou detekce objektů. Rovněž je vypočítána matice sématické segmentace ke kategorizaci každého pixelu vstupního obrazu. Model dále vytváří matici předpokládaných směrů jednotlivých pixelů ke středu odpovídající segmentované instance. Tyto matice jsou ořezány a spojeny dohromady pro hrubý odhad instance. Dále dochází k propojení hrubého odhadu instancí s každým RoI extrahovaným typicky

pomocí ResNet-101, díky čemuž lze segmentovat popředí a pozadí. Následuje závěrečné vyladění pomocí jednoduché trojvrstvé konvoluční sítě [19].



Obr. 4.3: Architektura instanční segmentační sítě MaskLab, převzato z [19].

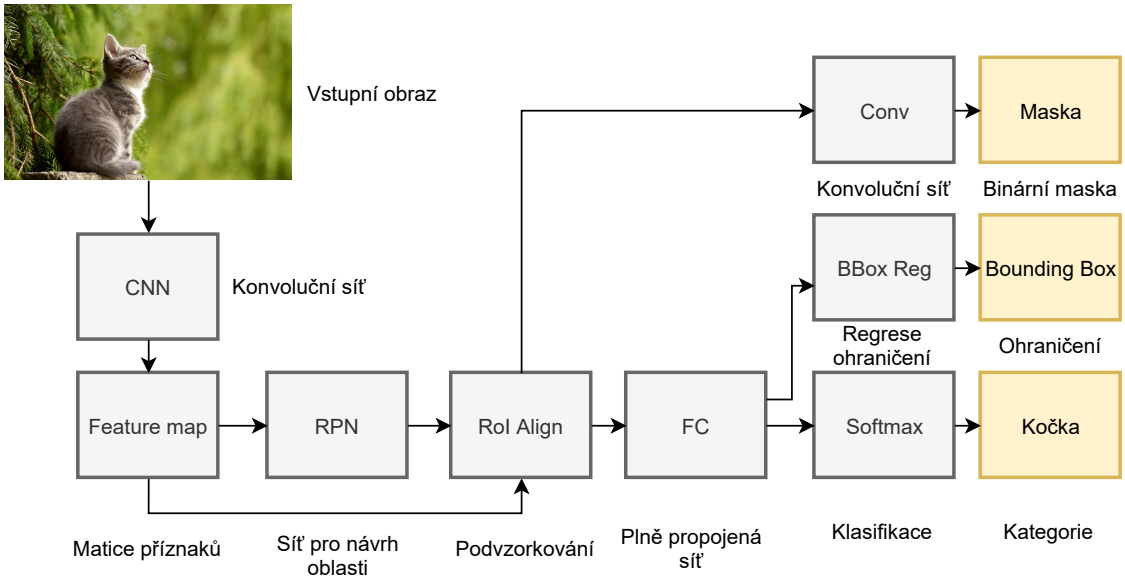
4.3.2 Mask R-CNN

Architektura sítě Mask R-CNN staví na základu architektury Faster R-CNN. Páteř konvoluční sítě může být realizována pomocí například ResNet50, ResNet101 nebo ResNext101. Výběr typu konvoluční sítě záleží na kompromisu mezi rychlostí učení a segmentace a mezi přesností segmentace. Síť Faster R-CNN nabízí dva výstupy – bounding box a příslušnou třídu. Segmentační síť Mask R-CNN k těmto základním dvěma výstupům přidává masku objektu pro každý RoI.

Počáteční kroky jsou až po RPN v Mask R-CNN plně převzaty. Nicméně při druhém kroku se paralelně k predikci tříd a vytváření bounding boxů přidává také binární maska pro každou oblast zájmu. Na rozdíl od většiny obdobných architektur zde klasifikace nezávisí na předpovědi masek.

Maska objektu kóduje prostorové rozvržení daného objektu. Na rozdíl od tříd nebo bounding boxů, které jsou transponovány do krátkých výstupních vektorů, jsou masky objektu určené přesným uspořádáním pixelů poskytnutým konvolucí. Konkrétně probíhá predikce masky o přesně dané velikosti z každého RoI pomocí FCN, tudíž každá vrstva sítě umožňuje udržet stejné prostorové rozměry bez nutnosti konverze vstupní masky do vektorové reprezentace. Z důvodu nutnosti přesného pixelového zarovnání masky objektu je nutno využít RoI align vrstvu, která hraje klíčovou roli v predikci masky.

Blokové schéma architekury Mask R-CNN je uvedeno na obrázku 4.4.



Obr. 4.4: Blokové schéma Mask R-CNN.

RoI pooling

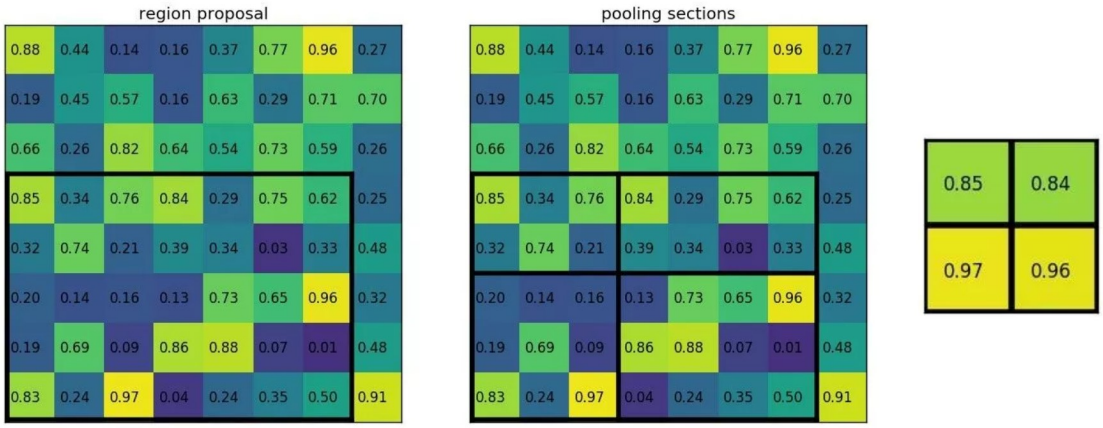
Vrstva RoI pooling škáluje získanou matici příznaků pro každý RoI na předdefinovanou velikost (typicky 7×7), čímž lze dosáhnout výrazného zlepšení výpočetní rychlosti testovací i trénovací fáze sítě, přičemž je zachována vysoká přesnost detekce. Při vygenerování vysokého počtu RoI by byla implementace detekce objektů v reálném čase bez použití RoI poolingu velmi obtížná [32].

Škálování (kvantizace) na předdefinovanou velikost probíhá pomocí rozdělení původní matice příznaků na sekce, jejichž počet odpovídá rozměru výstupní matice. Z každé této sekce je získána nejvyšší hodnota, která je následně uložena do výstupní matice. Výsupem je získání matice příznaků s fixní velikostí pro každý RoI nezávisle na jeho poměru stran či rozměrech [32].

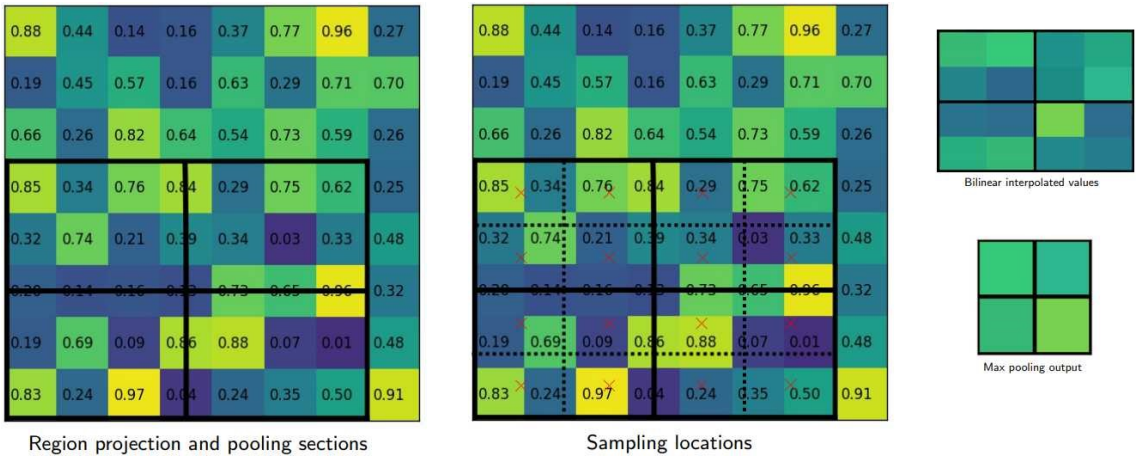
RoI align

Při procesu kvantizace u RoI poolingu dochází k vychýlení mezi RoI a maticí příznaků. Zatímco kvantizace nemusí mít žádný dopad na proces klasifikace objektu, na predikci pixelové masky je dopad velmi negativní. Z tohoto důvodu byla vytvořena RoI align vrstva, jejíž hlavním znakem je vynechání kvantizačního procesu.

Pomocí bilineární interpolace jsou určeny přesné hodnoty příznaků ze čtyř stejnoměrných oblastí v každé RoI oblasti. Z výsledků interpolací jsou získány hodnoty pomocí nejvyšší nebo průměrné hodnoty (obdoba max poolingu a average poolingu). Výsledky RoI align vrstvy nejsou závislé na přesných rozměrech či lokaci, pokud je vynechán proces kvantizace.



Obr. 4.5: Ukázka RoI poolingu z RoI o rozměrech 7 × 5 na matici příznaků o rozměrech 2 × 2, převzato z [32].



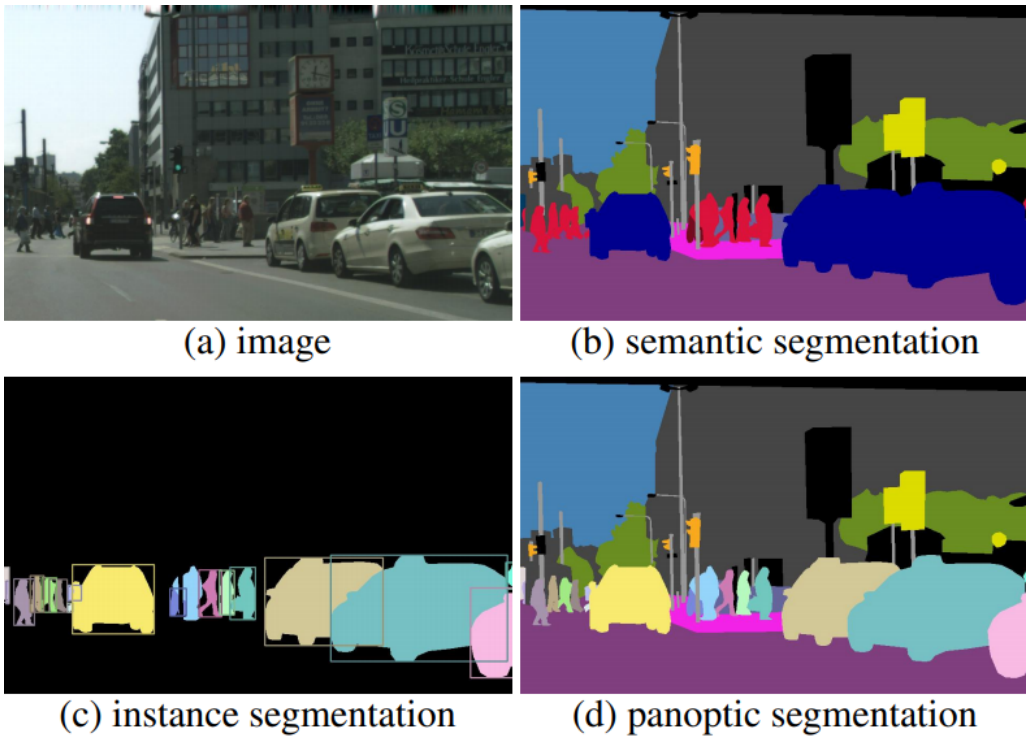
Obr. 4.6: Ukázka RoI align z RoI o rozměrech 7 × 5 na matici příznaků o rozměrech 2 × 2, převzato z [32].

4.4 Panoptická segmentace

Pojem panoptická segmentace zahrnuje využití sémantické i instanční segmentace v jednom sjednoceném celku. V současné době jde stále o málo prozkoumanou oblast. Při panoptické segmentaci dochází k přiřazení každého pixelu do určité kategorie a zároveň jsou rozeznány jednotlivé instance ve třídách. Výstupní model panoptické segmentace tedy obsahuje dva kanály pro každý pixel – kategorii a instanci. Jako zástupce panoptických segmentačních sítí mohou být uvedeny: *OANet*, *UPSNet* a *Multitask Network for Panoptic Segmentation* [31].

V současnosti existuje pouze několik volně dostupných datasetů určených k panoptické segmentaci. Mezi tyto datasety patří: *Cityscapes*, *ADE20k* a *Mapillary Vistas*. Zatímco datasety *Cityscapes* a *Mapillary Vistas* byly vyvinuty výhradně

za účelem rozpoznání scény ze silničního provozu, dataset *ADE20k* obsahuje scény z běžného života [15].



Obr. 4.7: Porovnání sémantické, instanční a panoptické segmentace, převzato z [15].

5 Implementace segmentační neuronové sítě

Segmentační neuronová síť je zaměřena na segmentaci jednotlivých instancí součástek na fotografiích desek plošných spojů. Pro vytvoření této sítě bylo nutné:

- seznámit se se zdrojovým kódem Mask R-CNN (Matterport),
- pořídit vlastní fotografie desek plošných spojů,
- vhodně anotovat součástky na fotografiích,
- přizpůsobit kód operačnímu systému Windows 10,
- přizpůsobit zdrojový kód vlastnímu datasetu a anotacím,
- přidat augmentaci datasetu,
- vytvořit skripty sloužící k manupulaci se sítí a spouštění operací,
- propojit segmentační síť se sítí YOLO pro detekci mimořádné události,
- rozšířit zdrojový kód o měření časových údajů
- rozšířit zdrojový kód o výpočet metrik,
- vytvořit modul pro export výsledného reportu obsahujícího výsledky segmentací, metriky a časové údaje.

Následující sekce popisují jednotlivé části při vytváření segmentační sítě součástek na desce plošných spojů.

5.1 Použitá síť a její prvky

Pro realizaci segmentace nad deskou plošných spojů byla zvolena architektura Mask R-CNN. Tato architektura dokáže ohraničit daný objekt bounding boxem, klasifikovat objekt a vytvořit pixelovou masku pro každou instanci objektu. Architekura je nástavbou na známou architekturu Faster R-CNN.

Jako základ byla využita implementace Mask R-CNN od Matterport postavená na Pythonu, která využívá mimo jiné frameworky Tensorflow a Keras. Matterport implementace, dostupná z [34], byla publikována pod MIT licencí. V dokumentaci je přiloženo i několik ukázek projektů z oblasi geografie či medicíny, které na této implementaci rovněž stavěly.

Jako páteřní sítě byly využity sítě ResNet50 a ResNet101. Tyto sítě se liší počtem propojených konvolučních vrstev. Zatímco ResNet50 vyniká menší časovou náročností, síť ResNet101 může dosáhnout přesnějších výsledků.

Adresář `mrcnn` zahrnuje několik Python skriptů. Soubor `config.py` obsahuje nastavení Mask R-CNN modelu. Pomocí přepsání hodnot proměnných lze upravit hodnoty jako počet kroků epochy, počet kroků v rámci epochy, typ páteřní sítě, parametry RPN sítě, parametry RoI apod. Soubor `model.py` obsahuje samotnou definici modelu, jsou zde definovány jednotlivé vrstvy. Skript `visualize.py` obsahuje funkce, které mohou být použity při vykreslování výsledků. Soubor `utils.py`

obsahuje pomocné funkce, které jsou použity v ostatních skriptech.

S implementací bylo spjato několik komplikací. Nejnovější verze Matterport R-CNN implementace (verze 2.1) byla vydána roku 2018 a tudíž je postavena na Tensorflow 1 (aktuální Tensorflow verze 2 byl vydán až koncem roku 2019). Při instalaci požadovaných balíčků pomocí PIP (package installer for Python) dochází k nekompabilitám. Původní `requirements.txt` soubor definuje pouze názvy balíčků. Verze nejsou definovány a nejvyšší verze není omezena. Rovněž není definovaná konkrétní verze Pythonu. Proto bylo nutné zkoušet různé kombinace verzí a upravovat původní zdrojový kód tak, aby byl kompatibilní s danýma verzema balíčků. V kořenovém adresáři projektu byl vytvořen soubor `requirements.txt`, který obsahuje přesné verze balíčků použité při vytváření segmentační neuronové sítě.

5.2 Vytvoření datasetu

Pro naučení segmentační neuronové sítě bylo zapotřebí vytvořit vlastní rozsáhlý dataset anotovaných fotografií. V práci nejsou použity převzaté fotografie. Všechny fotografie použité pro učení sítě byly vyfoceny výhradně pro účely této práce.

Dataset obsahuje fotografie desek plošných spojů, jednotlivých součástek nebo různých kombinací součástek. Fotky byly pořízeny z různých úhlů, z různých vzdáleností a při různém nasvícení, aby byly zajištěny nejlepší možné výsledky segmentace. Dataset zahrnuje 660 fotografií, přičemž trénovací množina obsahuje 415 snímků a dalších 219 snímků je určeno pro validační množinu. Testovací dataset pro ověření funkčnosti segmentační sítě obsahuje 26 snímků, které jsou umístěny v adresáři `test_files`. Fotografie byly pořízeny digitálním fotoaparátem Nikon D5500 a mobilním telefonem iPhone SE. Dataset obsahuje následující čtyři kategorie součástek:

- elektroluminiscenční dioda,
- relé modul,
- elektrolytický kondenzátor,
- integrovaný obvod.

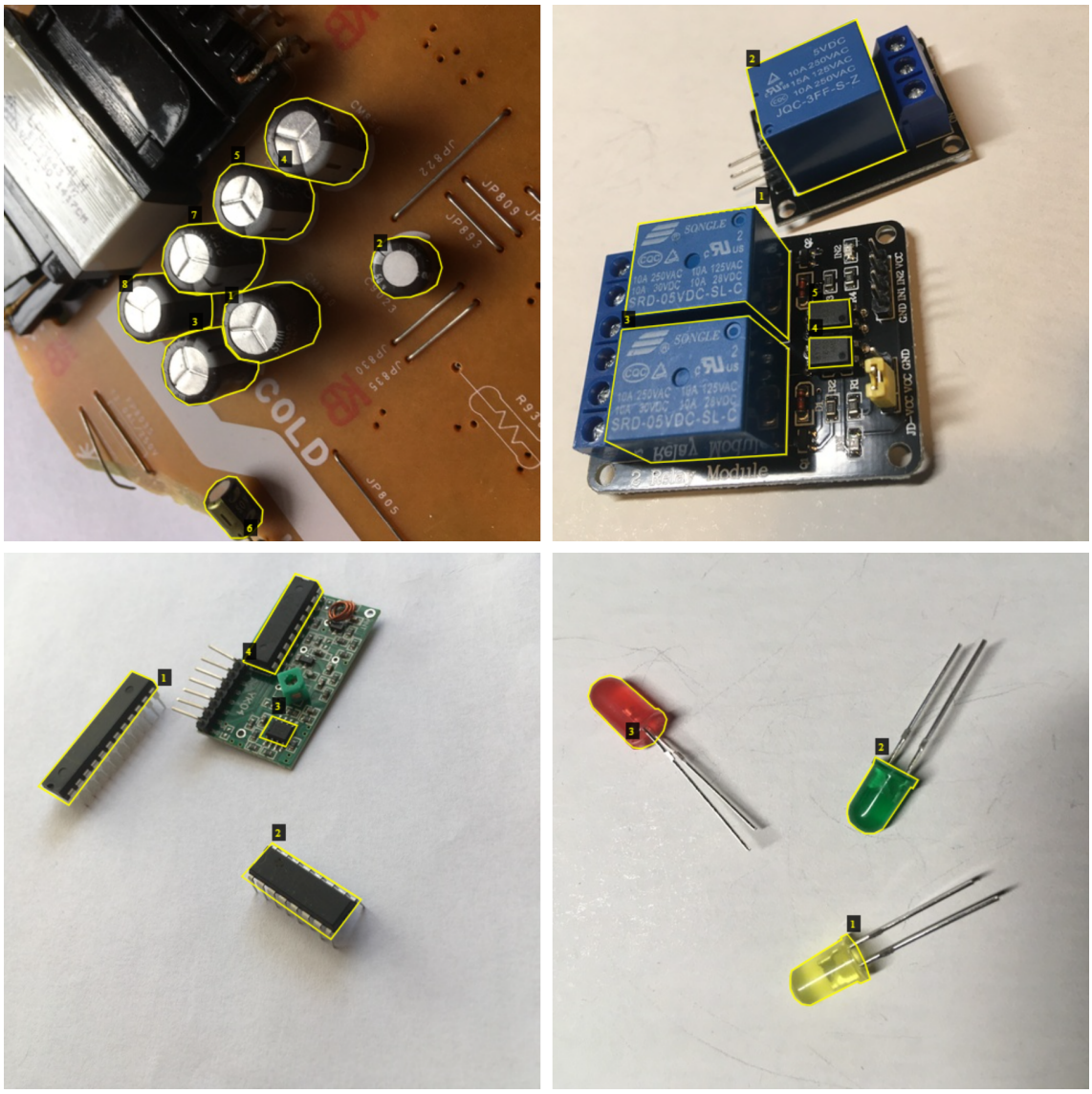
Před procesem anotace byly snímky z celého datasetu sjednoceny na pevný poměr stran 1:1. Rovněž došlo k úpravě rozlišení snímků na 512×512 pixelů pomocí jednoduchého Python skripu `PCB_resize_dataset.py`, který je obsažen v kořenovém adresáři projektu.

5.2.1 Anotace datasetu

Anotaci sady testovacích a validačních obrázků lze provést pomocí různých nástrojů. K nejznámějším patří LabelMe, RectLabel, ImageTagger, CVAT (Computer Vision

Annotation Tool) nebo VIA (VGG Image Annotator). Tyto nástroje ukládají souřadnice objektů do běžně používaných formátů, typicky JSON nebo CSV.

Vytvořený dataset byl anotován pomocí nástroje VGG Image Annotator [33]. Tento nástroj má jednoduché uživatelské prostřední a rovněž není nutná instalace, jelikož nástroj běží v rozhraní webového prohlížeče. Nástroj umožňuje export do různých formátů, a to konkrétně JSON, CSV a COCO. Jako vhodný formát anotací byl vybrát formát JSON. Ukázka anotace jedné instance objektu relé ve formátu JSON je zobrazena na obrázku 5.1:



Obr. 5.1: Ukázka anotací vytvořeného datasetu součástek.

Důležitými hodnotami u anotačního souboru jsou zejména *filename*, který udává název a příponu anotovaného obrázku, *name*, který udává název kategorie objektu

a také hodnoty *all_points_x* a *all_points_y*, které obsahují pole souřadnic jednotlivých bodů určujících ohraničení objektu. Dané hodnoty jsou z JSON souboru zpracovány a použity k určení bounding boxu a masky instance objektu.

Při manuální anotaci datasetu je nutné kolem krajů fotografií dbát na to, aby souřadnice nevystupovaly z rozměrů fotografie. Tento problém se často vyskytuje při použití funkce přiblížení ve webovém prohlížeči. Ačkoliv souřadnice anotovaného objektu může vizuálně vypadat, že spadá do ohraničení fotografie, vygenerovaný JSON soubor s anotacemi může obsahovat zápornou hodnotu (nebo hodnotu, která přesahuje výšku či šířku fotografie). V tomto případě není možné spustit proces učení neuronové sítě.

5.2.2 Augmentace datasetu

Pomocí augmentace dat lze zvýšit přesnost vyhodnocení při zachování současné velikosti datasetu. Při použití augmentace dochází k vytvoření nových fotografií v datasetu použitím jednoduché nebo komplexní transformace či úpravy. Příklad augmentace může být rotace, převrácení, přiblížení, gaussovské rozostření nebo vystřižení.

V práci bylo využito augmentace datasetu pomocí knihovny *imgaug*. Knihovna podporuje mnoho augmentačních technik a umožňuje jejich vzájemnou kombinaci. Zajištěna je nejen augmentace fotografií, ale rovněž úprava bounding boxů, heatmap nebo segmentačních masek. Pomocí této techniky dochází k mnohonásobnému navýšení velikosti datasetu při stejném množství fotografií. Implementovaná segmentační síť může použít tyto augmentační techniky (za použití parametru `-a heavy`):

- rotace (90°, 180°, 270°),
- oříznutí,
- převrácení (dle osy x či y),
- úprava jasu,
- úprava kontrastu,
- gaussovské rozostření,
- zaostření.

Použitá augmentace datasetu je podmíněna parametrem `-a` při spuštění hlavního skriptu `PCB.py`. Možné hodnoty parametru pro augmentaci jsou: *no* (bez augmentace), *light* (lehká augmentace), *heavy* (těžká augmentace).

5.3 Konfigurace a spuštění

Pro spuštění učení sítě, provedení segmentace nebo vyhodnocení kvality modelu pomocí metrik slouží Pythonový skript `PCB.py`, který se nachází v kořenovém adresáři

projektu. Průběžné informace a logy jsou vypisovány do konzolového výstupu. Jelikož celkový výstup obsahuje mimo jiné i obrázky nebo grafy, byl pro výstupní export zvolen formát HTML souboru. Při spuštění skriptu mohou být využity argumenty uvedné v tabulce 5.1.

Tab. 5.1: Argumenty skriptu PCB.py.

Přepínač	Možné hodnoty	Význam
-m	train, segmentation, metrics	mód spuštění sítě
-r	y (yes), n (no)	povolit generování výstupního souboru
-e	(číslo 1 - 500)	počet epoch při učení sítě
-s	(číslo 1 - 10000)	počet kroků jedné epochy při učení sítě
-l	(číslo 0,0001 - 0,1)	míra učení sítě
-b	resnet50, resnet101	typ páteřní sítě
-a	no, light, heavy	použitá augmentace

5.3.1 Proces učení sítě

Segmentační neuronová síť byla vyvíjena na počítači, který nedisponuje dedikovanou grafickou kartou. Proces učení bylo možné spustit tedy pouze na procesoru, který ovšem nedokázal provést náročné výpočty v požadovné době (tj. v řádu několika hodin). Z důvodu vysoké hardwarové náročnosti učícího procesu bylo nutno využít moderní grafické karty. Pro proces učení bylo tedy využito služby Google Colaboratory [36], která bezplatně nabízí možnost spuštění skriptu s využitím výkonné grafické karty. V případě aktivované grafické akcelerace je ve službě Google Colaboratory přiřazena náhodně grafická karta – většinou jde o typy NVIDIA K80, T4, P4 nebo P100. Pomocí příkazu `torch.cuda.get_device_name` je možné ověřit, zda došlo k incializaci grafické karty a určit konkrétní typ.

Náhodné přiřazení grafické karty v prostředí Google Colaboratory může zkreslovat časové náročnosti při učení sítě nebo segmentaci snímků. Z tohoto důvodu jsou výstupní HTML reporty pro učení sítě a segmentaci v záhlaví opatřeny informací o konkrétním typu grafické karty, která byla během procesu učení přiřazena.

Z důvodu vysoké časové náročnosti učícího procesu docházelo v bezplatné verzi prostředí Google Colab k výpadkům. Jestliže uživatel spustí příkaz, který vytíží grafickou kartu na několik hodin, dochází k dočasnému odpojení od grafické akcelerace, a to i často v průběhu učícího procesu. Toto dočasné odpojení může trvat různě dlouhou dobu. Při alokaci grafických karet jsou upřednostňováni uživatelé, kteří službu

nevyužívají bezplatně, ale jsou předplatiteli. Kvůli těmto limitacím byla zakoupena služba Google Colab Pro.

Výstupem učícího procesu jsou hodnoty vah, které jsou ukládány po každé epoše do HDF (*Hierarchical Data Format)* souboru s příponou h5 o velikosti cca 171 MB v případě páteřní sítě ResNet50 nebo o velikosti 244 MB v případě páteřní sítě ResNet101. Tento soubor může rovněž sloužit k navázání učícího procesu v případě, že došlo k jeho přerušení, a není tedy nutno znovu začínat od první epochy. K dosažení uspokojivých výsledků bylo zapotřebí nastavit alespoň 120 epoch učení v případě ResNet50 a 200 epoch učení v případě ResNet101, přičemž každá epocha se skládá z 300 kroků.

Proces učení sítě je možné spustit například následujícím příkazem:

```
python PCB.py -m train -e 200 -s 300 -l 0.0015
-b resnet101 -a light -r y
```

který spustí proces učení o 200 epochách, 300 krocích, mírou učení 0,0015, lehkou augmentací datasetu a za použití páteřní sítě ResNet101.

Parametry, které byly použity při procesu učení sítě, jsou zobrazeny v následující tabulce 5.2. Při vývoji neuronové sítě bylo testována mnoho konfigurací. Práce se zaměřuje zejména na uvedené dvě konfigurace.

Tab. 5.2: Použité konfigurace.

Konfigurace	Konfigurace 1	Konfigurace 2
Páteřní síť	ResNet50	ResNet101
Počet epoch	120	200
Počet kroků epochy	300	300
Počet validačních kroků	50	50
Míra učení (*learning rate*)	0,002	0,0015
Hranice důvěrnosti	0,85	0,85
Počet anchor pointů RPN / obrázek	256	256
Počet RoI / obrázek	200	200
Maximální počet instancí / obrázek	100	100
Augmentace datasetu	rotace, zrcadlení	rotace, zrcadlení

Časová náročnost

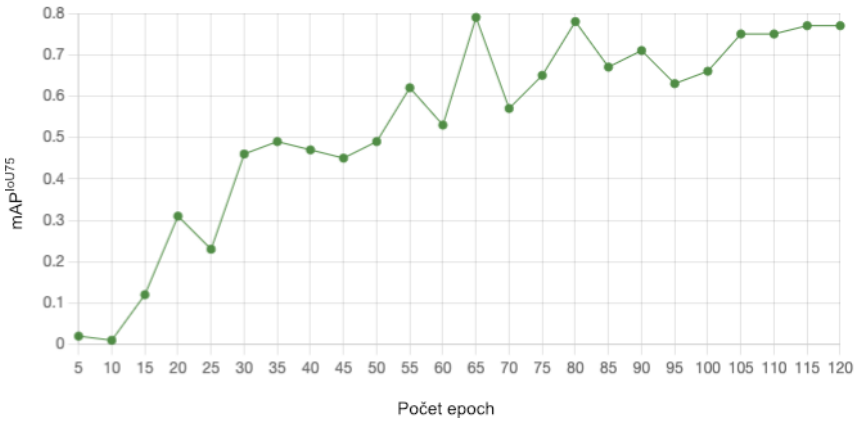
Provedení celkového učícího procesu o 120 epochách za použití páteřní sítě ResNet50 trvalo 4 hodiny a 23 minut (průměrně 2,2 minuty na jednu epochu). Při učení došlo k inicializaci grafické karty typu NVIDIA V100 16 GB.

Učící proces při použití páteřní sítě ResNet101 o 200 epochách trval 7 hodin a 36 minut (průměrně 2,3 minuty na jednu epochu). V tomto případě byla rovněž inicializována grafická karta NVIDIA V100 16 GB.

Po každé epoše dochází k uložení HDF souboru, které je při použití ResNet101 zdlouhavější než v případě ResNet50 z důvodu většího objemu ukládaných dat (obzvláště při použití pomalého povného disku).

Průběžné mezivýsledky metrik

Segmentační neuronová síť byla obohacena o průběžné výsledky metrik. Mezivýsledky metrik jsou realizovány pomocí callback funkce, která je volána po dokončení každé epochy během procesu učení. Ačkoliv je funkce volána vždy po dokončení epochy, k výpočtu metrik dochází pouze každých pět epoch z důvodu vysoké časové náročnosti výpočtu. Metriky rovněž nejsou určeny z celého validačního datasetu, ale je vybráno pouze 20 % snímků. Díky těmto opatřením je zachován vhodný poměr mezi časovou náročností učení sítě a objektivitou průběžných mezivýsledků metrik. Po dokončení učení jsou výsledky zapsány do HTML reportu, který obsahuje data v podobě tabulek i grafů. Ukázka výstupního reportu po procesu učení je uložena v adresáři `results`.

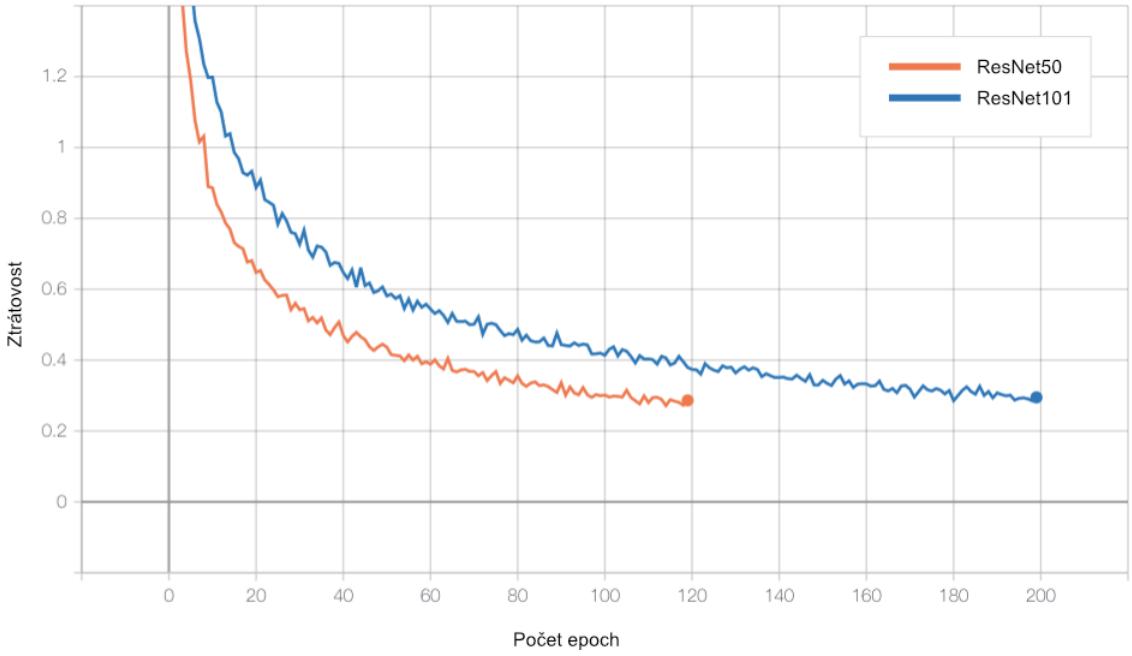


Obr. 5.2: Průběžné mezivýsledky metriky $\mathrm{mAP}^{IoU\,75}$ (ResNet50) zaznamenané v průběhu učícího procesu.

Učící proces za použití odlišných páteřních sítí

Proces učení byl testován na páteřních sítích typu ResNet50 a ResNet101. Architektura sítě ResNet101 je komplikovanější v porovnání s sítí ResNet50 a obsahuje více navazujících residuálních bloků (vrstev) a parametrů. Tento jev se projevuje na rychlosti snižování ztrátové funkce v závislosti na počtu epoch. Zatímco síť ResNet50 byla schopna snížit hodnotu ztrátové funkce pod hodnotu 0,4 za 55 epoch,

ResNet101 vyžadoval pro stejné snížení 107 epoch. Jev je zachycen na obrázku 5.3, který zobrazuje srovnání ztrátových funkcí sítí ResNet50 (oranžová) a ResNet101 (modrá) v závislosti na počtu epoch. Z tohoto důvodu bylo použito při první konfiguraci 120 epoch, zatímco při druhé konfiguraci bylo použito 200 epoch.



Obr. 5.3: Porovnání vývoje ztrátových funkcí při použití sítí ResNet50 (oranžová) a ResNet101 (modrá) v závislosti na počtu epoch.

5.3.2 Proces segmentace

Naučený model sítě dosahuje velmi uspokojivých výsledků. Jednotlivé instance kategorie jsou správně rozeznány, bounding box je správně zobrazen. Pixelová maska instance objektu je občas nepřesná, objevují se malé odchylky od požadovaného ohraničení. Odchylky závisí na úhlu natočení součástky, osvětlení a poloze objektu.

V případě, že jsou instance objektu stejné třídy umístěny tak, že zasahují do bounding boxů instance jiné, lze pozorovat nesprávné ohraničení masky objektu. Nicméně k této situaci dochází pouze za předpokladu, že jsou dvě instance objektů stejné třídy umístěny velmi blízko a zároveň je úhel kamery takový, že bounding box překryje jinou instanci.

Vyšší přesnosti výsledné masky objektů může být dosaženo zvýšením počtu epoch či jednotlivých kroků v rámci epochy.

Proces segmentace je možné spustit například následujícím příkazem:

```
python PCB.py -m segmentation -b resnet50 -r y
```

který spustí proces segmentace fotografií z adresáře `test_files` za použití páteřní sítě ResNet50 a následně vytvoří výstup v podobě HTML souboru.

Následující obrázky 5.4, 5.5, 5.6 zobrazují výsledné segmentace.

Obrázek 5.4 ukazuje rozdíly v segmentaci při použití páteřní sítě ResNet50 (nahoře) a ResNet101 (dole). Masky LED diod jsou vyplněny správně a při použití odlišných páteřních sítí není viditelný rozdíl. Relé modul je u obou typů sítí správně detekován, vykreslená maska má jen drobné odchylky. V případě páteřní sítě ResNet50 nebyl detekován elektrolytický kondenzátor, narozdíl od páteřní sítě ResNet101, která byla schopna i správně vykreslit masku objektu. Neschopnost segmentace součástky může být způsobena stejnou barvou součástky a pozadí, nebo lesklým povrchem součástky. U integrovaného obvodu limituje vykreslení požadované masky bounding box, a to obzvláště v případě ResNet50. V rámci bounding boxů je maska pro integrovaný obvod vykreslena správně a hraničí s okrajem součástky.

Obrázek 5.5 ukazuje výsledek segmentace nad LED diodama za použití páteřní sítě ResNet50. Všech šest instancí objektu je správně ohraničeno, kategorizování a pixelové masky jsou správně vykresleny.

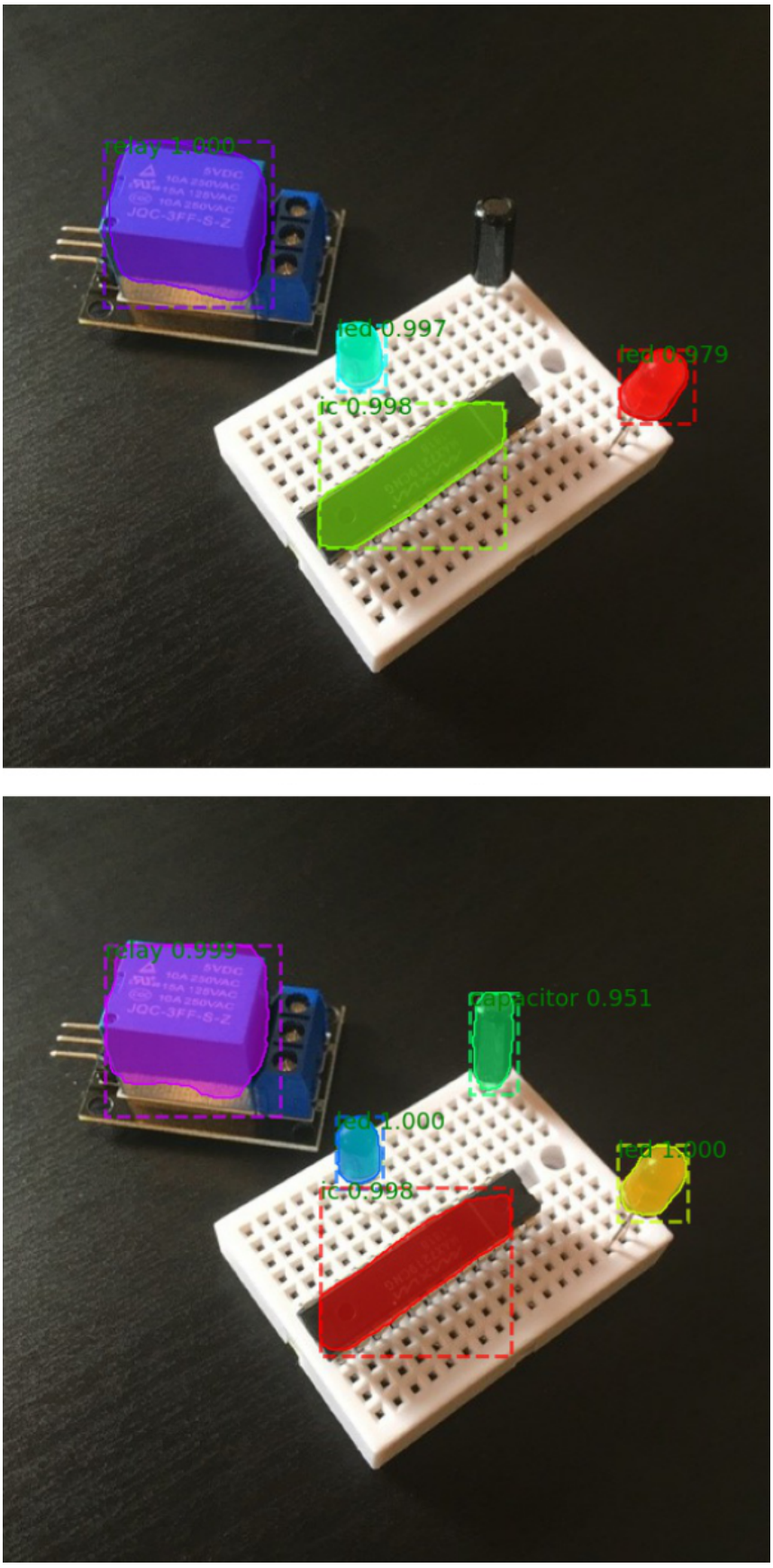
Na snímku 5.6 je zobrazena segmentace tří integrovaných obvodů. Všechny tři instance objektu jsou správně nalezeny, ohraničeny bounding boxy, kategorie objektu je správně určena. Pixelová maska správně vyplňuje objekt. Správné segmentaci napomáhá vysoký rozdíl mezi barvou součástky a barvou pozadí (podkladu).

Časová náročnost segmentace snímku

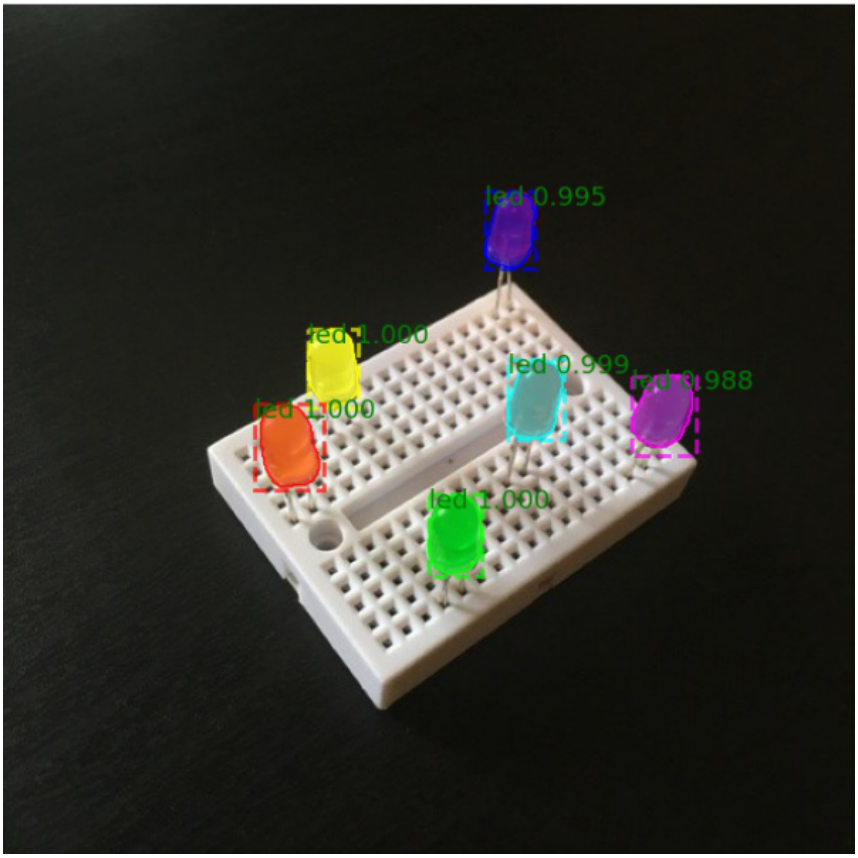
Průměrná časová náročnost při segmentaci dat je zobrazena v tabulce 5.3. Pro segmentaci byly použity snímky o rozlišení 512×512 pixelů ve formátu JPG. Segmentace byla provedena jak za použití procesoru Intel i5-6300U 2.40 GHz, tak za použití přiřazené grafické karty Tesla P100 16 GB v prostředí Google Colaboratory. Detekce rukou pomocí YOLO trvá na CPU průměrně 0,88 s, při použití GPU je časová náročnost v řádech setin sekund. Čas nutný k počátečnímu načtení vah sítě je zanedbán. Z výsledků je zřejmé, že grafická karta dosahuje mnohem lepších výsledků než procesor. Páteřní síť ResNet50 je časově méně náročná než síť ResNet101, a to průměrně o necelých 14 %.

Tab. 5.3: Průměrná časová náročnost segmentace snímku.

Páteřní síť	CPU Intel i5-6300U	GPU Tesla P100 16 GB
ResNet50	12,57 s	0,380 s
ResNet101	16,49 s	0,440 s



Obr. 5.4: Ukázka výsledné segmentace (nahoře ResNet50, dole ResNet101).



Obr. 5.5: Ukázka výsledné segmentace (použita síť ResNet50).

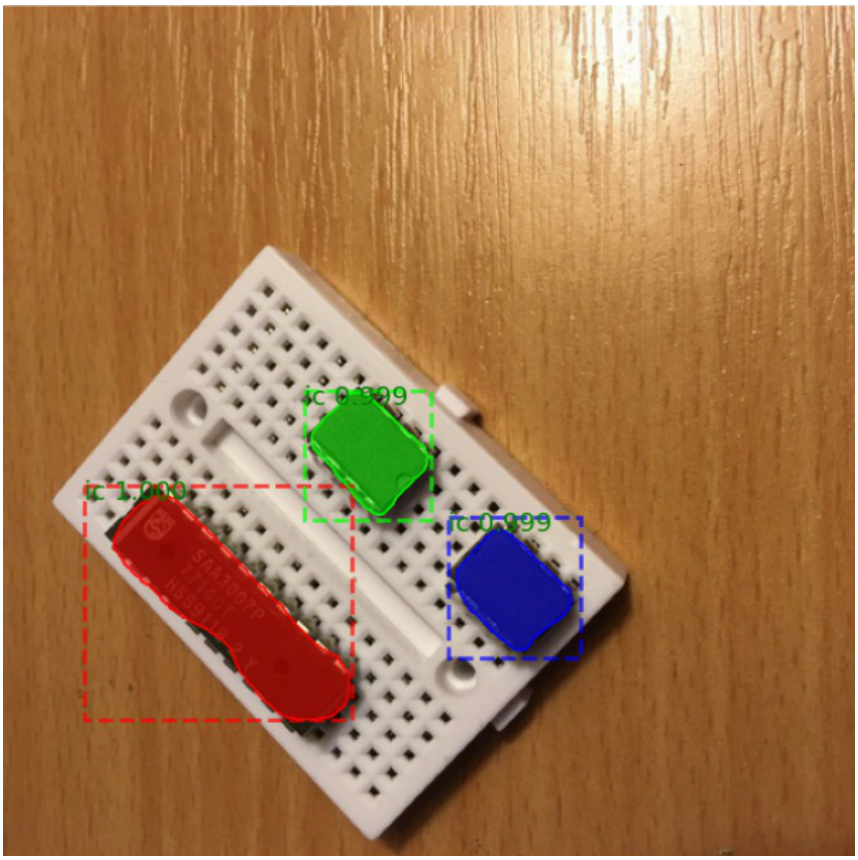
5.3.3 Detekce mimořádné události

Segmentační neuronová síť je schopna vyhodnotit mimořádnou událost. Je předpokládán scénář, kdy kamera snímá desky plošných spojů na výrobním páse. Při průjezdu každé desky plošných spojů je kontrolováno správné osazení, tedy počet instancí objektů, poloha a kategorie součástky. Na výrobním pásu není dovolena žádná manipulace se součástkou. Jako mimořádná událost je tedy vyhodnocena přítomnost ruky na snímku s deskou plošných spojů.

Instanční segmentace obrazu a vytvoření pixelové masky pro jednotlivé instance je časově mnohem náročnější proces, než pouhá detekce a klasifikace objektu. Z tohoto důvodu není pro detekci přítomnosti ruky ve snímku použita síť Mask R-CNN, ale byla zvolena síť typu YOLO. I v případě, že by pro detekci ruky ve snímku byla využita síť typu Mask R-CNN a větev pro vytvoření masky by byla vynechána, časová náročnost by byla stále vyšší než v případě YOLO.

V práci je použita síť YOLO verze 3. Pro účely detekce ruky bylo využito předtrénovaných vah dostupných z [35]. Váhy byly vytvořeny pomocí trénování na třech různých datasetech – CMU Hand DB, Egohands a cross-hands.

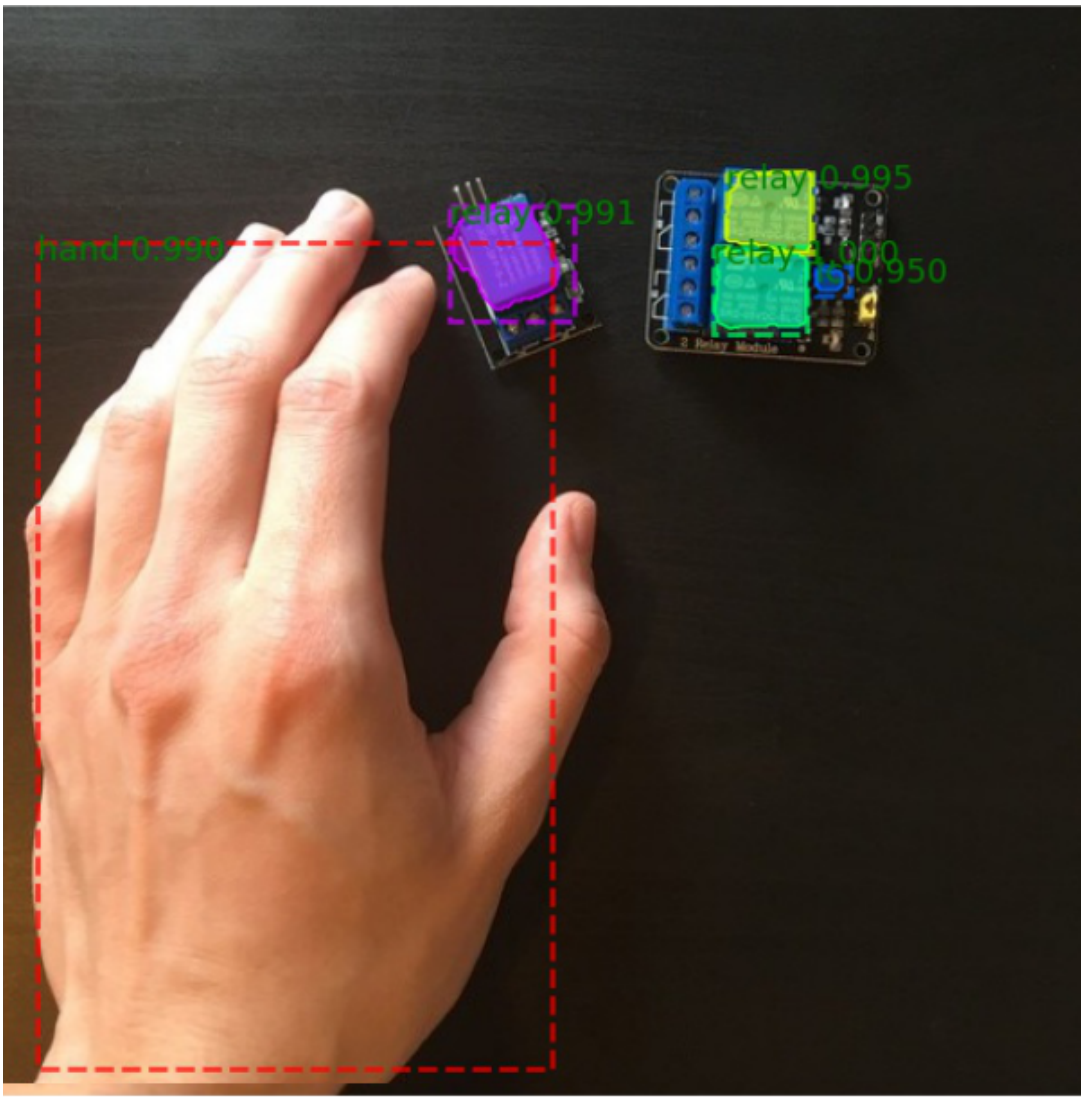
Síť YOLO pro detekci rukou byla zakomponována do stávající sítě Mask R-CNN



Obr. 5.6: Ukázka výsledné segmentace (použita síť ResNet101).

pro detekci součástek. Při spuštění skriptu `PCB.py` v segmentačním módu dochází před provedením segmentace součástek k uložení cesty k adresáři s testovacími fotografiemi. Tato cesta je zaslána do objektu ve skriptu `PCB_yolo.py` a dochází k provedení detekce rukou ve všech fotografiích v daném adresáři. Objekt si uchovává seznam detekcí v podobě seznamu polí, které obsahuje název fotografie, souřadnice levého horního rohu bounding boxu, výšky a šířky bounding boxu. Po provedení detekce rukou následuje detekce součástek pomocí Mask R-CNN sítě. Jakmile je provedena segmentace seoučástek na snímku, síť Mask R-CNN se dotáže na bounding boxy z detekce rukou. Pokud jsou boundung boxy pro patřičnou fotografii nalezeny, dochází k jejich podvržení při vizualizaci snímku. Tímto způsobem mohou být vykresleny nejen segmentace součástek, ale zároveň i ohraničení rukou.

Mimořádná událost se rovněž projeví ve výstupním HTML souboru segmentace. Uživatel je o výskytu této události patřičně informován chybovou zprávou. Ukázka detekce mimořádné události je zobrazena na obrázku 5.7.



Obr. 5.7: Ukázka mimořádné situace.

5.3.4 Vyhodnocení modelu

Metriky

K vyhodnocení kvality modelu lze využít několik metrik. Pro získání údajů lze použít následující příkaz:

```
python PCB.py -m metrics -b resnet50 -r y
```

Na základě výše uvedeného příkazu je vygenerován HTML soubor, který přehledně zobrazuje vypočtené metriky ve formě tabulky i grafu. Každý řádek tabulky obsa-

huje název zpracovaného obrázku, metriku Average Precision, F1-Score a F2-Score. Metrika Average Precision je vypočítána třikrát pro různé IoU (konkrétně 0,25, 0,50 a 0,75). V patičce jsou uvedeny průměrné metriky celého testovacího datasetu, tedy mAP, Mean F1-Score a Mean F2-Score. Hodnoty metrik jsou rovněž zobrazeny v grafech. Pro vykreslení grafů v HTML souborech bylo využito javascriptové knihovny `Chart.js`.

Tabulka 5.4 zobrazuje průměrné metriky naměřené na testovacím datasetu při použití páteřní sítě ResNet50 a ResNet101. Při použití páteřní sítě ResNet101 bylo dosaženo vyšších hodnot metrik v porovnání se sítí ResNet50 (vyjma $\text{mAP}^{\text{IoU }75}$). Lepší výsledky jsou rovněž viditelné při vizuálním srovnání výsledků segmentací, zejména u pixelových masek. Vyšší přesnosti by bylo možné dosáhnout delším procesem učení či úpravou rychlosti učení.

Tab. 5.4: Získané hodnoty metrik.

Páteřní síť	$\text{mAP}^{\text{IoU }25}$	$\text{mAP}^{\text{IoU }50}$	$\text{mAP}^{\text{IoU }75}$	F1 score	F2 score
ResNet50	0,83	0,81	0,72	0,40	0,30
ResNet101	0,89	0,87	0,65	0,44	0,33

Ztrátové funkce

Tabulka 5.5 zobrazuje postupný vývoj hodnot ztrátových funkcí pro trénovací dataset za použití páteřní sítě ResNet101. V příloze B jsou obsaženy tabulky zobrazující vývoj hodnot ztrátových funkcí pro trénovací i validační dataset za použití páteřních sítí ResNet50 i ResNet101. Spojnice trendu mají u všech zobrazených grafů klesající tendence. Hodnoty ztrátových funkcí konvergují k optimální hodnotě. Z hodnot je zřejmé, že se v prvních 200 epochách nevyskytuje nežádoucí overfitting.

Celková ztrátová funkce L je definována jako součet ztrátové funkce masky, ohraničení a klasifikace:

$$L = L_{mask} + L_{bbox} + L_{class}. \tag{5.1}$$

Ztrátová funkce klasifikace L_{class} je definovaná následujícím vzorcem:

$$L_{class}(p, u) = -\log p_u, \tag{5.2}$$

kde u určuje skutečnou třídu objektu ($u \in 0, 1, ..., K$), symbol K určuje počet tříd, přičemž třída 0 označuje typicky pozadí, p je diskrétní rozdělení pravděpodobnosti (každé oblasti zájmu zvlášť) přes $K+1$ tříd ($p = (p_0, p_1, ..., p_K)$), vypočítané softmax funkcí přes $K + 1$ výstupů plně propojené vrstvy (*Fully connected layer – FCL*).

Ztrátová funkce bounding boxu L_{bbox} je definovaná jako:

$$L_{bbox}(t^u, v) = \sum_{i \in x,y,w,h} L_s(t_i^u - v_i), \tag{5.3}$$

přičemž funkce Ls je rovna:

$$Ls(x) = \begin{cases} 0{,}5x^2 & \text{pokud } |x| < 1 \\ |x| - 0{,}5 & \text{pokud } |x| \geq 1, \end{cases} \tag{5.4}$$

kde v označuje skutečný bounding box ($v = (v_x, v_y, v_w, v_h)$) a t^u označuje odchylku predikovaného bounding boxu ($t^u = (t_x^u, t_y^u, t_w^u, t_h^u)$). Symbol x značí horizontální souřadici levého horního rohu bounding boxu, y vertikální souřadici levého horního rohu bounding boxu, w šířku bounding boxu a h výšku bounding boxu.

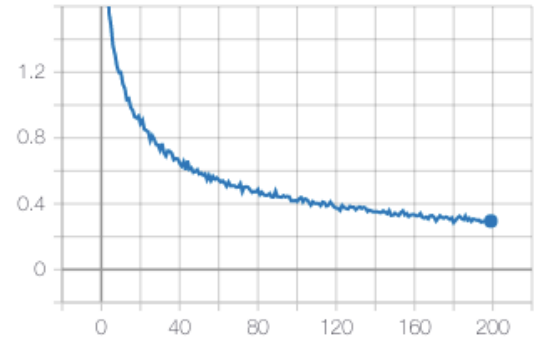
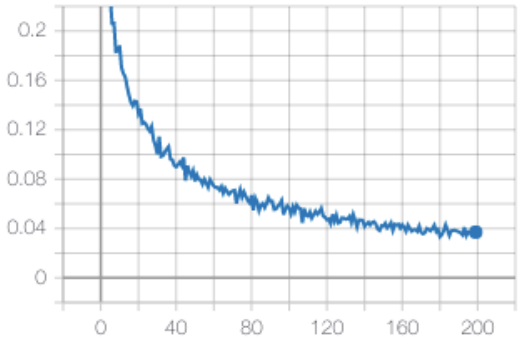
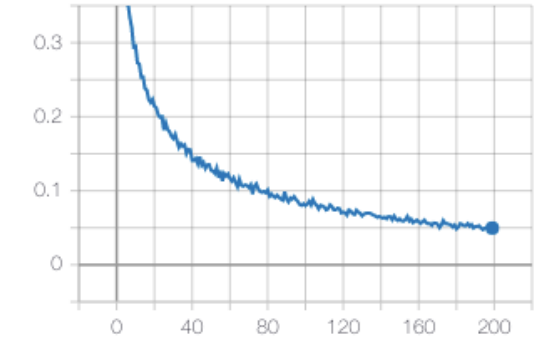
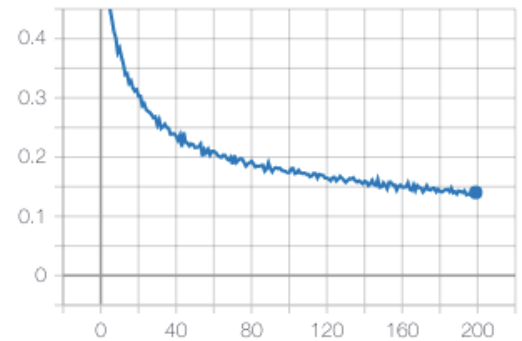
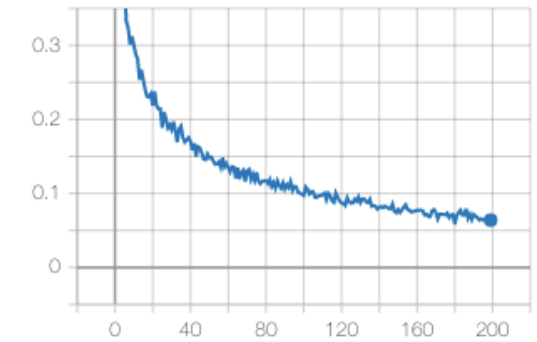
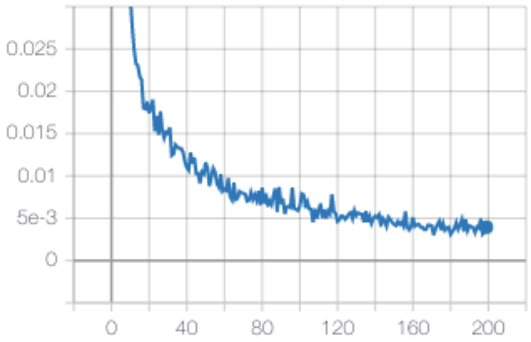
Ztrátová funkce masky L_{class} je definována jako:

$$L_{mask} = -\frac{1}{m^2} \sum_{1 \leq i,j \leq m} [y_{ij} \log\, y_{ij}^k + (1 - y_{ij}) \log\, (1 - y_i^k j)], \tag{5.5}$$

kde m je rozměr masky v pixelech, y_{ij} je reálná hodnota masky v daném pixelu, y_{ij}^k je predikovaná hodnota masky v daném pixelu.

Během 200 epoch učení za použití ResNet101 se u trénovacího datasetu snížila celková ztrátová funkce na 0,295, ztrátová funkce klasifikace na 0,037, ztrátová funkce bounding boxu na 0,049 a ztrátová funkce masky na 0,149. U validačního datasetu se snížila celková ztrátová funkce na 0,591, ztrátová funkce klasifikace na 0,062, ztrátová funkce bounding boxu na 0,109 a ztrátová funkce masky na 0,183.

Tab. 5.5: Průběh hodnot ztrátových funkcí (trénovací dataset – ResNet101).

Loss (L) 	Mask R-CNN class loss (L_{class}) 
Mask R-CNN bounding box loss (L_{bbox}) 	Mask R-CNN mask loss (L_{mask}) 
RPN bounding box loss 	RPN class loss 
Horizontální osa udává počet epoch, vertikální osa udává hodnotu ztrátové funkce	

Závěr

V rámci diplomové práce byla rozebrána problematika segmentačních neuronových sítí. Teoretická část práce se zabývá komplexní tématikou neuronových sítí od základních stavebních prvků sítí, jako je stavba neuronu, aktivační funkce nebo bias, až po komplikované architektury segmentačních neuronových sítí jako json například U-Net, MaskLab či Mask R-CNN. Práce popisuje postupy při výpočtu metrik mAP a F-score a jejich varianty.

V praktické části byl vytvořen vlastní rozsáhlý dataset obsahující 660 fotografií desek plošných spojů. Dataset se konkrétně zaměřuje na čtyři typy součástek: relé modul, elektrolytický kondenzátor, elektroluminiscenční dioda a integrovaný obvod. Fotografie byly rozděleny na trénovací, validační a testovací množinu, přičemž zastoupení trénovací množiny vzhledem k validační množině je v poměru 2:1. Jednotlivé instance součástek byly pro každou fotografii anotovány a vyexportovány pomocí volně dostupného webového nástroje VGG annotator tool. Pro snadné zpracování anotací byl zvolen formát JSON.

Jako základ pro vytvoření segmentační neuronové sítě pro segmentaci součástek na desce plošných spojů byla využita implementace od Matterport, která je založena na modelu Mask R-CNN. Na základě teoretických znalostí byla tato síť upravena tak, aby bylo možné použít vlastní dataset s vlastními anotačními soubory. Z důvodu navýšení množství fotografií datasetu byla přidána augmentace dat datasetu. Při implementaci sítě bylo řešeno několik komplikací, které byly spjaty například s nekompatibilitou balíčků, konkrétní verzí pythonu nebo použitím OS Windows namísto linuxové distribuce. Z důvodu vývoje na počítači, který nedisponuje dedikovanou grafickou kartou, bylo využito služby Google Colaboratory, která výrazně snížila časovou náročnost při učícím či segmentačním procesu. Nicméně i tato služba přinesla několik komplikací, jako například přiřazovaní náhodné grafické karty nebo odpojování při běhu učícího procesu. Z důvodu vysoké časové náročnosti učícího procesu byla zvolena placená služba Google Colaboratory Pro, nicméně tato služba rovněž nemá neomezenou dobu běhu skriptu, ale pouze tuto dobu navyšuje a k odpojování dochází sporadicky.

Vytvořená segmentační neuronová síť je schopna segmentovat jednotlivé instance součástkek na desce plošných spojů. Kvalitu segmentace lze vyhodnotit díky několika metrikám, konkrétně $\text{mAP}^{\text{IoU}\,25}$, $\text{mAP}^{\text{IoU}\,50}$, $\text{mAP}^{\text{IoU}\,75}$, F1 Score a F2 Score, které byly do kódu přidány. Metriky jsou vypočítány jednotlivě pro každý obrázek a rovněž jako průměr celého datasetu. Průběžné metriky jsou také počítány během učícího procesu sítě. Z důvodu vysoké časové náročnosti jsou metriky počítány pouze pro každou pátou epochu a rovněž pouze pro vybranou množinu z datasetu, což přináší vhodný poměr mezi rychlostí učení sítě a relevantností naměřených hodnot met-

rik. V práci je srovnáno použití dvou páteřních sítí ResNet50 a ResNet101. Práce zdůvodňuje výsledky segmentací za použití obou pátečních sítí, časovou náročnost, ztrátové funkce i naměřené metriky. Pro detekci mimořádné události došlo k propojení segmentační neuronové sítě Mask R-CNN s další sítí typu YOLOv3. Z procesu učení sítě, segmentace a vyhodnocení metrik jsou generovány výstupy v podobě HTML reportů. Reporty obsahují informace v podobě přehledných tabulek či grafů. V záhlaví výstupního reportu jsou zobrazeny informace o parametrech při spuštění a o použité grafické kartě. Při segmentaci obsahuje report výsledné snímky segmentací, časovou náročnost pro vybraný snímek a celkovou časovou náročnost pro celou množinu segmentovaných snímků.

V práci jsou přiloženy ukázky výsledných segmentací, grafy získaných ztrátových funkcí pro obě konfigurace a návod na instalaci a spuštění sítě včetně ukázky příkazů.

Literatura

[1] KAEHLER, Adrian a Gary R. BRADSKI. *Learning OpenCV 3: computer vision in C++ with the OpenCV library*. Sebastopol, CA: O'Reilly Media, [2017]. ISBN 978-1-491-93799-0.

[2] JOSHI, Prateek, David Millán ESCRIVÁ a Vinícius GODOY. *OpenCV By Example: Enhance your understanding of Computer Vision and image processing by developing real-world projects in OpenCV 3*. Birmingham: Packt Publishing, 2016. ISBN 978-1-78528-094-8.

[3] HAYKIN, Simon. *Neural networks and learning machines*. 3rd ed. Upper Saddle River, N.J.: Pearson, c2009. ISBN 978-0-13-147139-9.

[4] T. HAGAN, Martin, Howard B. DEMUTH, Mark HUDSON BEALE a Orlando DE JESÚS. *Neural Network Design: (2nd edition)* [online]. 2nd edition. Oklahoma: Martin Hagan, 2014. ISBN 0971732116.

[5] MICHELUCCI, Umberto. *Applied Deep Learning: A Case-Based Approach to Understanding Deep Neural Networks*. Dübendorf, Switzerland: Apress, 2018. ISBN 978-1-4842-3789-2.

[6] BUDHIRAJA, Amar. Dropout in (Deep) Machine learning. *Medium* [online]. 2016 [cit. 2020-10-25]. Dostupné z: `https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5`

[7] MISHRA, Sanatan. Unsupervised Learning and Data Clustering. *Towards Data Science* [online]. 2017 [cit. 2020-12-03]. Dostupné z: `https://towardsdatascience.com/unsupervised-learning-and-data-clustering-eeecb78b422a`

[8] WOOD, Thomas. Softmax function. *DeepAI* [online]. [cit. 2020-11-21]. Dostupné z: `https://deepai.org/machine-learning-glossary-and-terms/softmax-layer`

[9] KŮRKOVÁ, Věra, Yannis MANOLOPOULOS, Barbara HAMMER, Lazaros ILIADIS a Ilias MAGLOGIANNIS, ed. *Artificial Neural Networks and Machine Learning – ICANN 2018: 27th International Conference on Artificial Neural Networks Rhodes, Greece, October 4–7, 2018*. Springer Nature Switzerland, 2018. ISBN 978-3-030-01417-9.

[10] SHARMA, Pulkit. A Step-by-Step Introduction to the Basic Object Detection Algorithms. *Analytics Vidhya* [online]. 2018 [cit. 2020-10-18]. Dostupné z: https://www.analyticsvidhya.com/blog/2018/10/a-step-by-step-introduction-to-the-basic-object-detection-algorithms-part-1/

[11] GANDHI, Rohith. R-CNN, Fast R-CNN, Faster R-CNN, YOLO – Object Detection Algorithms: Understanding object detection algorithms. *Towards Data Science* [online]. 2018 [cit. 2020-10-18]. Dostupné z: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

[12] NELLI, Fabio. *OpenCV & Python – The Otsu's Binarization for thresholding. Meccanismo Complesso* [online]. Meccanismo Complesso, 2017 [cit. 2020-10-21]. Dostupné z: https://www.meccanismocomplesso.org/en/opencv-python-the-otsus-binarization-for-thresholding/

[13] GURUNG, Amit a Sangyal Lama TAMANG. Image Segmentation using Multi-Threshold technique by Histogram Sampling. *Cornell University* [online]. Cornell University, 2019 [cit. 2020-10-21]. Dostupné z: https://arxiv.org/abs/1909.05084

[14] Image Segmentation in Deep Learning: Methods and Applications: Computer vision. *MissingLink* [online]. [cit. 2020-10-21]. Dostupné z: https://missinglink.ai/guides/computer-vision/image-segmentation-deep-learning-methods-applications/

[15] KIRILLOV, Alexander, Kaiming HE, Ross GIRSHICK, Carsten ROTHER a Piotr DOLLÁR. *Panoptic Segmentation* [online]. 2019 [cit. 2020-10-24]. Dostupné z: https://arxiv.org/abs/1801.00868

[16] GURUSAMY, Vairaprakash a Subbu KANNAN. *Review on image segmentation techniques* [online]. In: . 2014 [cit. 2020-10-24]. Dostupné z: https://www.researchgate.net/publication/273127438_REVIEW_ON_IMAGE_SEGMENTATION_TECHNIQUES

[17] ASHOUR, Amira a Ahmed REFAAT HAWAS. K-Means Clustering. *Science Direct* [online]. 2019 [cit. 2020-10-24]. Dostupné z: https://www.sciencedirect.com/topics/computer-science/k-means-clustering

[18] RONNEBERGER, Olaf, Philipp FISCHER a Thomas BROX. *U-Net: Convolutional Networks for Biomedical Image Segmentation* [online]. In: . Freiburg: University of Freiburg, 2015 [cit. 2020-10-24]. Dostupné z: https://arxiv.org/pdf/1505.04597.pdf

[19] CHEN, Liang-Chieh, Alexander HERMANS, George PAPANDREOU, Florian SCHROFF, Peng WANG a Hartwig ADAM. *MaskLab: Instance Segmentation by Refining Object Detection with Semantic and Direction Features* [online]. 2017 [cit. 2020-10-24]. Dostupné z: https://arxiv.org/abs/1712.04837

[20] KHANDELWAL, Renu. Convolutional Neural Network(CNN) Simplified. *Medium* [online]. 2018 [cit. 2020-10-25]. Dostupné z: https://medium.com/datadriveninvestor/convolutional-neural-network-cnn-simplified-ecafd4ee52c5

[21] SAHA, Sumit. A Comprehensive Guide to Convolutional Neural Networks. *Towards data science* [online]. 2018 [cit. 2020-11-21]. Dostupné z: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[22] GIRSHICK, Girshick. Fast R-CNN [online]. 2015 [cit. 2020-11-21]. Dostupné z: https://arxiv.org/abs/1504.08083

[23] STECANELLA, Burno. An Introduction to Support Vector Machines (SVM). *Monkey Learn* [online]. 2017 [cit. 2020-11-21]. Dostupné z: https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/

[24] GANDHI, Rohith. Support Vector Machine: Introduction to Machine Learning Algorithms. *Towards data science* [online]. 2018 [cit. 2020-11-21]. Dostupné z: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithm-934a444fca47

[25] Faster R-CNN: Using Region Proposal Network for Object Detection. *Alegion* [online]. [cit. 2020-12-05]. Dostupné z: https://www.alegion.com/faster-r-cnn

[26] WOOD, Thomas. F-Score: What is the F-score? *DeepAI* [online]. [cit. 2021-01-18]. Dostupné z: https://deepai.org/machine-learning-glossary-and-terms/f-score

[27] SHORTEN, Connor. Introduction to ResNets. *Towards data science* [online]. 2019 [cit. 2021-04-04]. Dostupné z: https://towardsdatascience.com/introduction-to-resnets-c0a830a288a4

[28] FENG, Vincent. An Overview of ResNet and its Variants. *Towards data science* [online]. 2017 [cit. 2021-04-04]. Dostupné z: https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035

[29] VIJAYRANIA, Nilesh. Different Normalization Layers in Deep Learning: Weight Standardization — An Alternative to Batch Normalization. *Towards data science* [online]. 2020 [cit. 2021-04-04]. Dostupné z: https://towardsdatascience.com/different-normalization-layers-in-deep-learning-1a7214ff71d6

[30] Scipy Lecture Notes. In: *Scipy Lecture Notes* [online]. [cit. 2020-10-21]. Dostupné z: http://scipy-lectures.org/packages/scikit-image/aut_examples/plot_threshold.html

[31] SULTANA, Farhana, Abu SUFIANA a Paramartha DUTTA. *Evolution of Image Segmentation using Deep Convolutional Neural Network* [online]. In: Elsevier BV, 2020 [cit. 2020-10-21]. ISSN 0950-7051. Dostupné z: https://arxiv.org/abs/2001.04074

[32] GREL, Tomasz. Region of interest pooling explained. *Deepsense.ai* [online]. 2017 [cit. 2020-11-08]. Dostupné z: https://deepsense.ai/region-of-interest-pooling-explained/#:~:text=Region%20of%20interest%20pooling%20(also,pedestrians%20in%20a%20single%20image.&text=In%20the%20second%20case%20it,all%20objects%20in%20an%20image.

[33] DUTTA, Abhishek, Ankush GUPTA a Andrew ZISSERMAN. *The VIA Annotation Software for Images, Audio and Video*. Visual Geometry Group [online]. 2019 [cit. 2021-03-12]. Dostupné z: https://www.robots.ox.ac.uk/~vgg/software/via/

[34] Mask R-CNN for Object Detection and Segmentation. *Github* [online]. 2018, 20.03.2018 [cit. 2021-03-20]. Dostupné z: https://github.com/matterport/Mask_RCNN

[35] YOLO-Hand-Detection. *Github* [online]. [cit. 2021-03-22]. Dostupné z: https://github.com/cansik/yolo-hand-detection

[36] *Google Colaboratory* [online]. California, USA [cit. 2021-5-5]. Dostupné z: https://colab.research.google.com/notebooks/intro.ipynb

Seznam symbolů, veličin a zkratek

AP *Average Precision* – Průměrná přesnost

CNN *Convolutional neural network* – Konvoluční neuronová síť

CPU *Central processing unit* – Centrální procesorová jednotka

CV *Computer vision* – Počítačové vidění

FCL *Fully connected layer* – Plně propojená vrstva

FCN *Fully convolutional network* – Plně konvoluční síť

FN *False negative* – Falešně negativní predikce

FP *False positive* – Falešně pozitivní predikce

HDF *Hierarchical Data Format* – Datový formát

HTML *Hypertext Markup Language* – Jazyk pro tvorbu webových stránek

JSON *JavaScript Object Notation* – Datový formát

LED *Light-Emitting Diode* – Elektroluminiscenční dioda

LTSM *Long Term Short Memory* – Rekurentní neuronová síť

mAP *Mean Average Precision* - Metrika užívaná v oblasti počítačového vidění

MIT Svobodná licence

ML *Machine learning* – Strojové učení

PIP *Package installer for Python* – instalátor balíčků pro Python

R-CNN *Region-based convolutional neural network* – Konvoluční neuronová síť založená na oblastech

ReLU *Rectified linear activation function* – Usměrněná lineární aktivační funkce

ResNet *Residual network* – Reziduální síť

ROC *Receiver operating characteristics* – Metrika užívaná v oblasti počítačového vidění

RoI *Region of interest* – Oblast zájmu

RPN *Region Proposal Network* – Síť návrhů oblasti

TNR *True negative rate* – Metrika užívaná v oblasti počítačového vidění

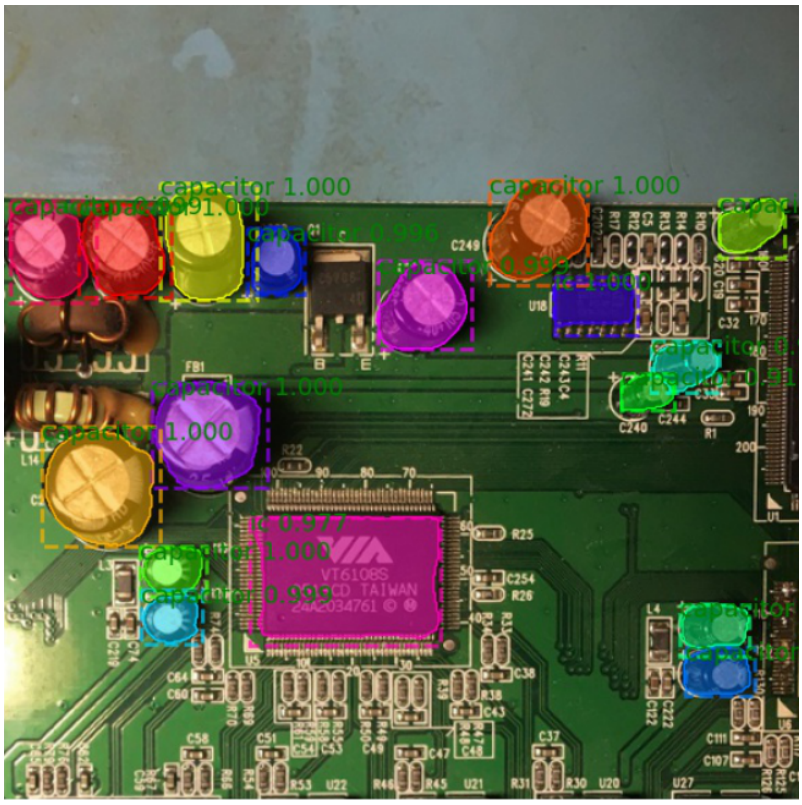
TP *True positive* – Skutečně pozitivní

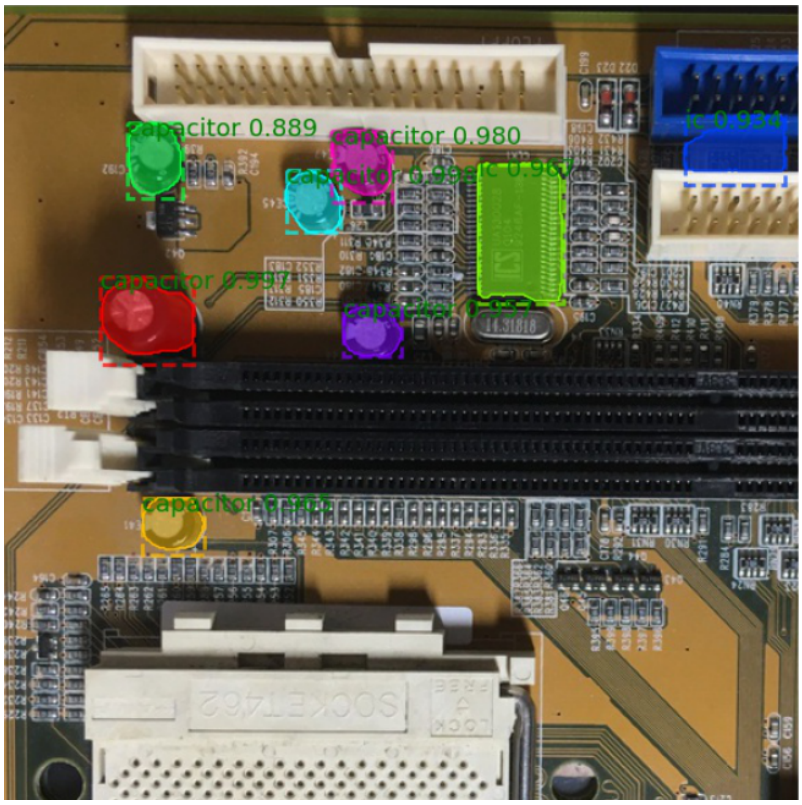
SVM *Support Vector Machine* – Metoda podpůrných vektorů

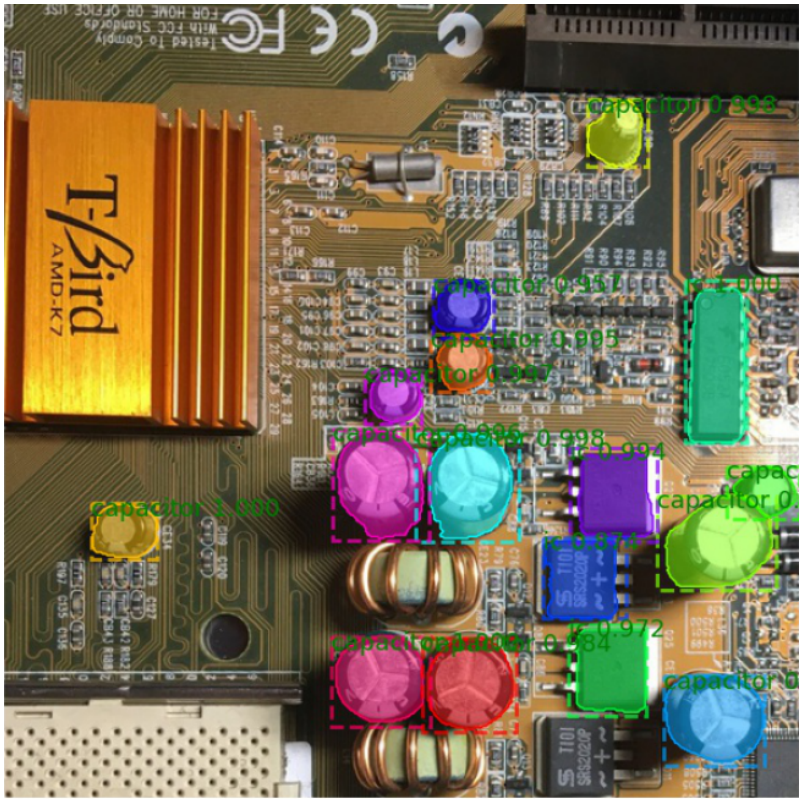
YOLO *You only look once* – Algoritmus „Nahlížíš pouze jednou“

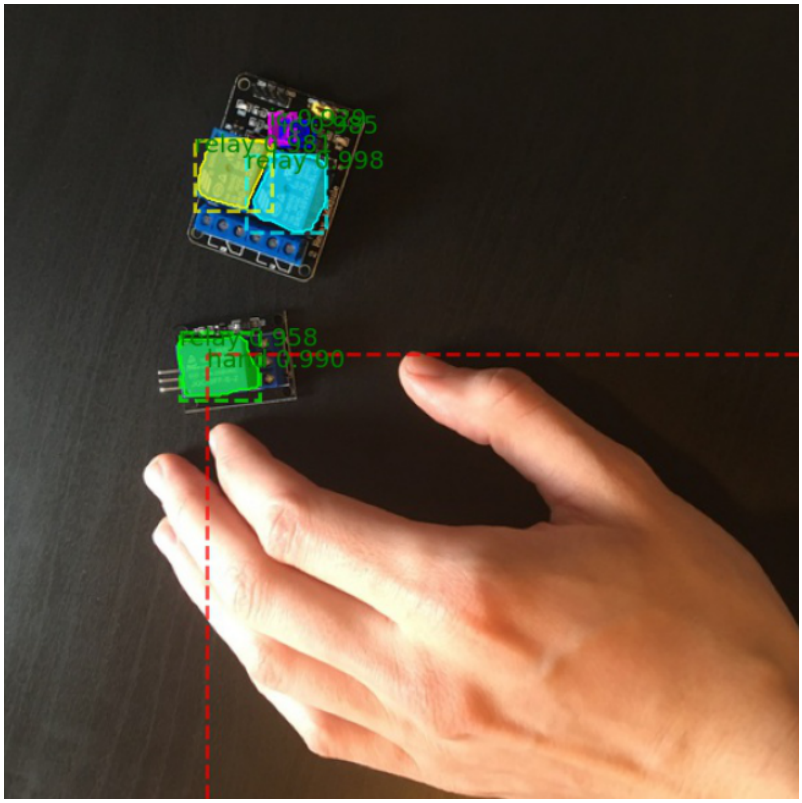
Seznam příloh

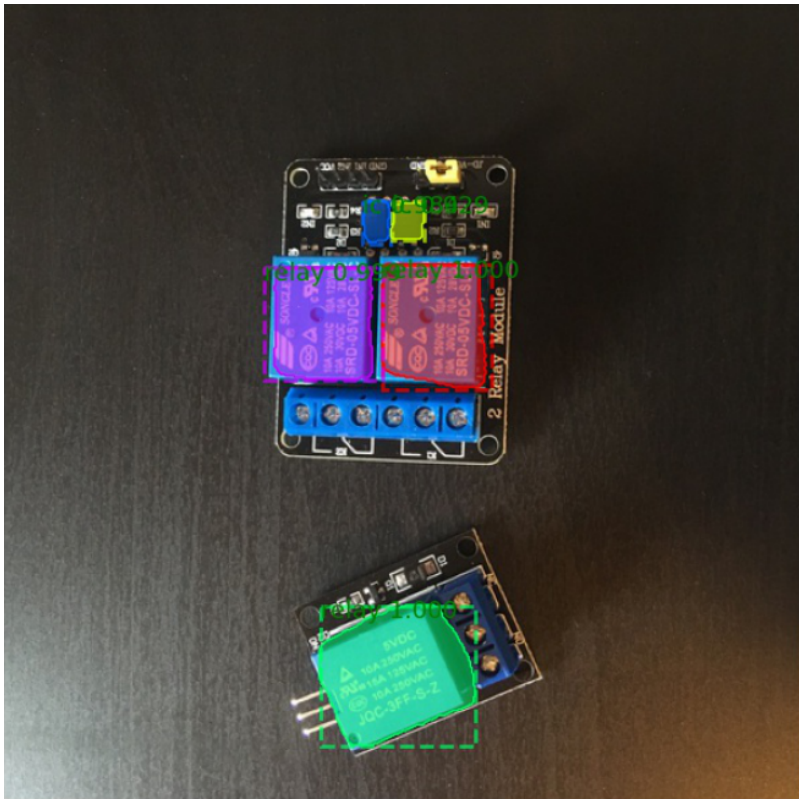
A Ukázky výsledné segmentace součástek

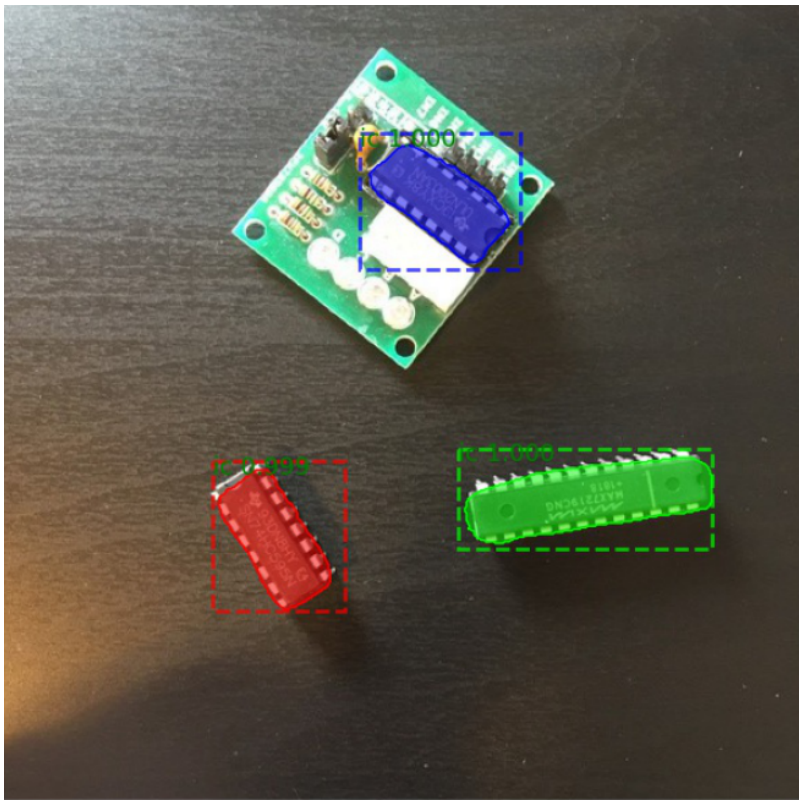


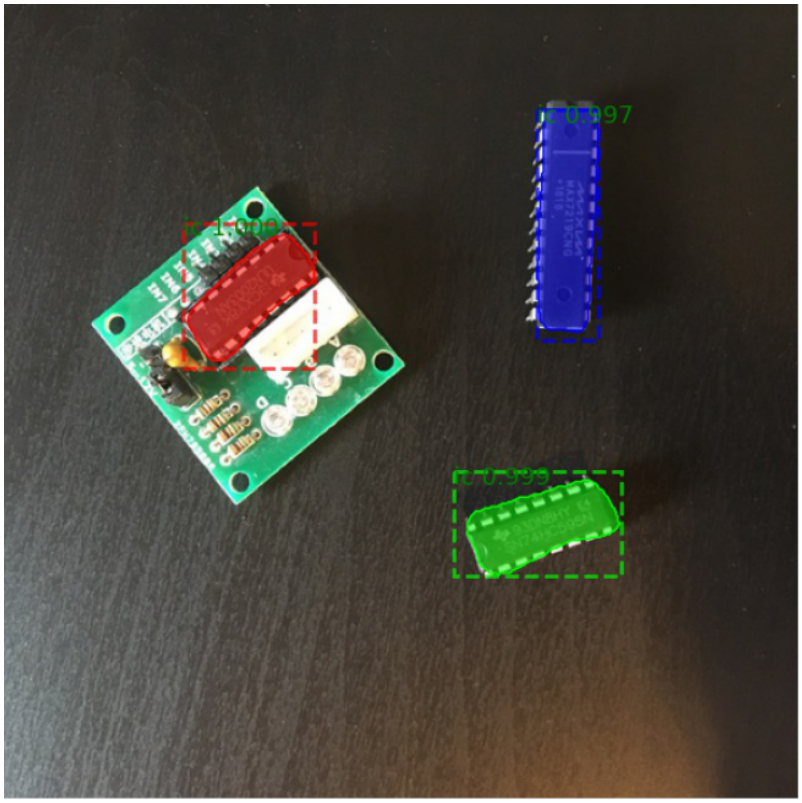


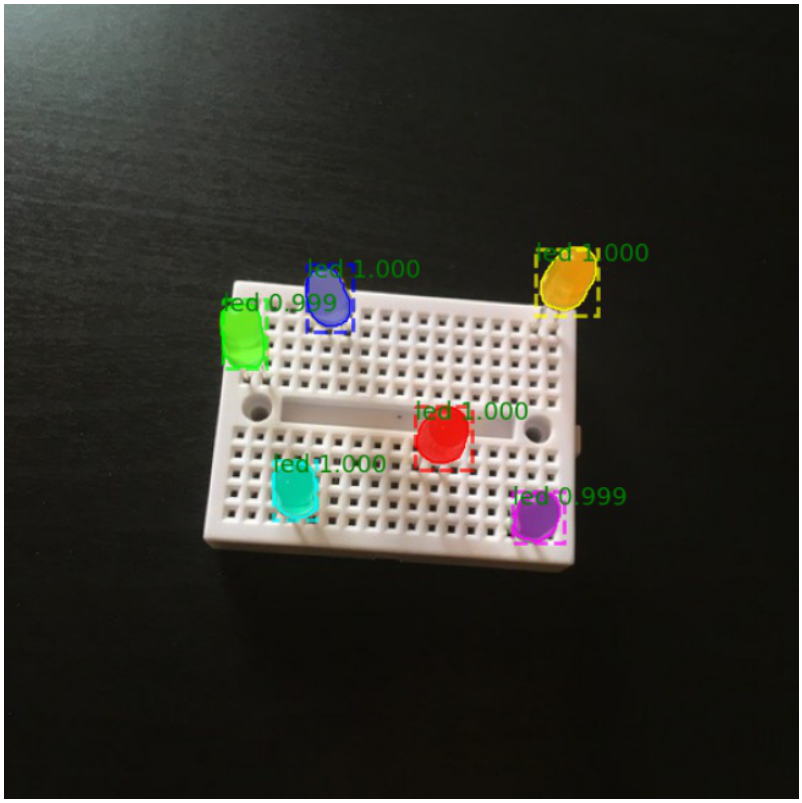

T-Bird
AMD-K7
capacitor 0.998
capacitor 0.995
capacitor 1.000
capacitor 0.972

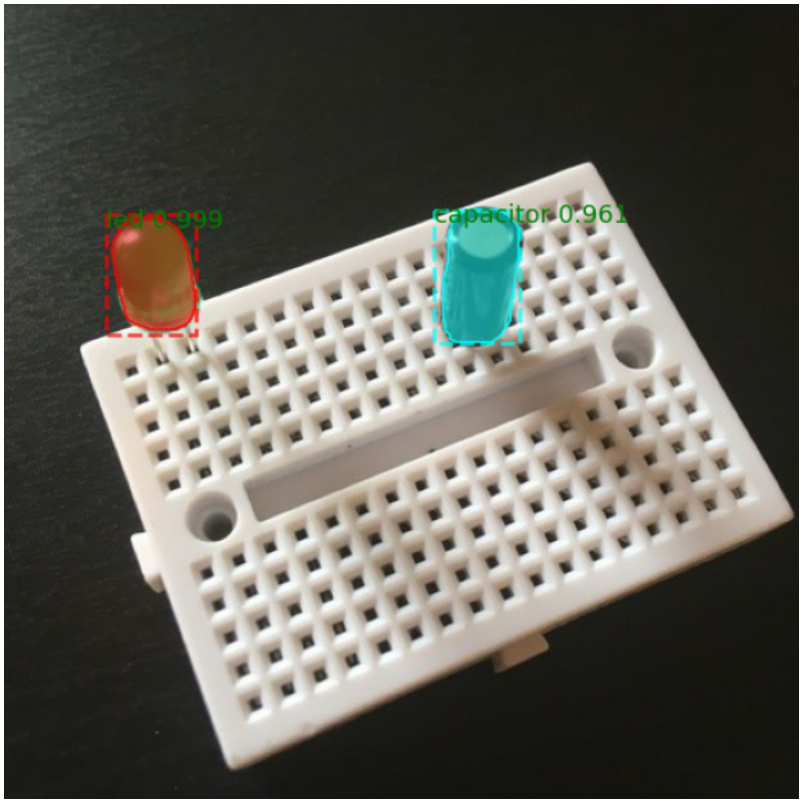

relay 0.998
relay 0.958
relay 0.990

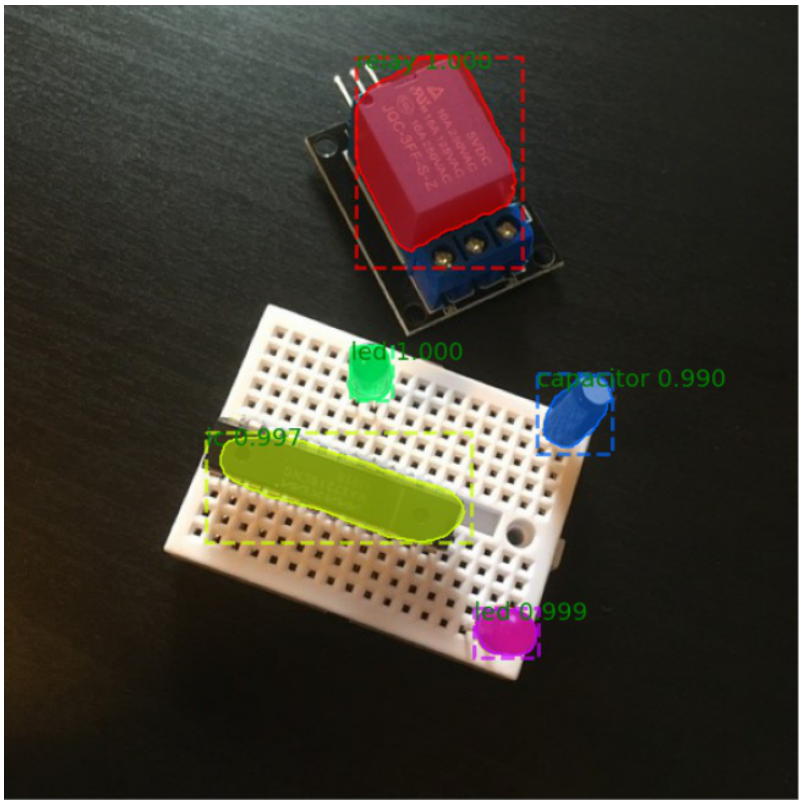

relay 0.999
relay 1.000
2 Relay Module
relay 1.000
5VDC
10A 250VAC
10A 125VAC
10A 250VAC
JQC-3FF-S-Z

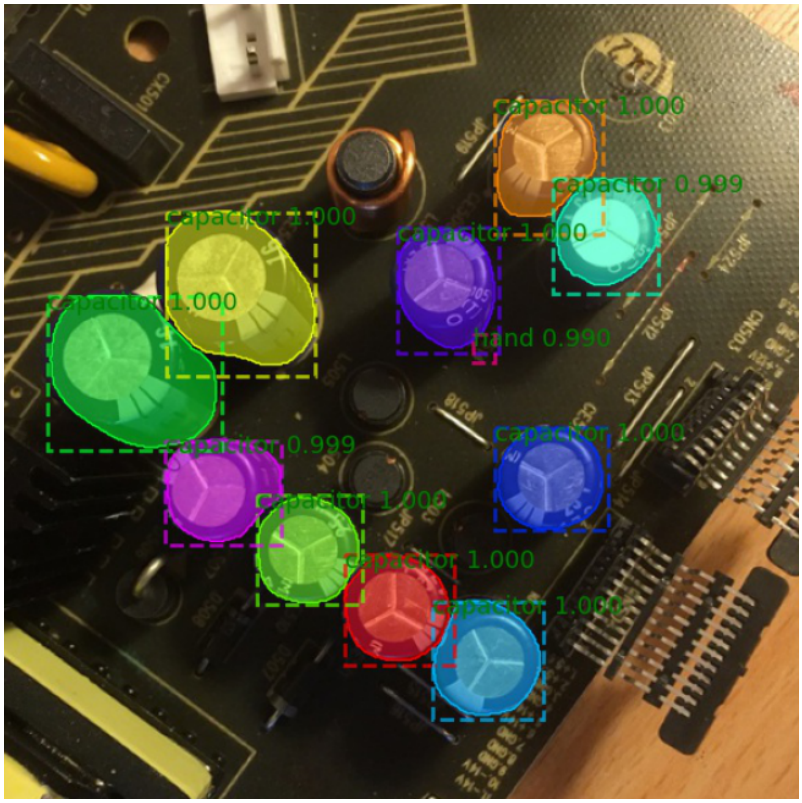

ic 1.000
ic 0.999
ic 1.000

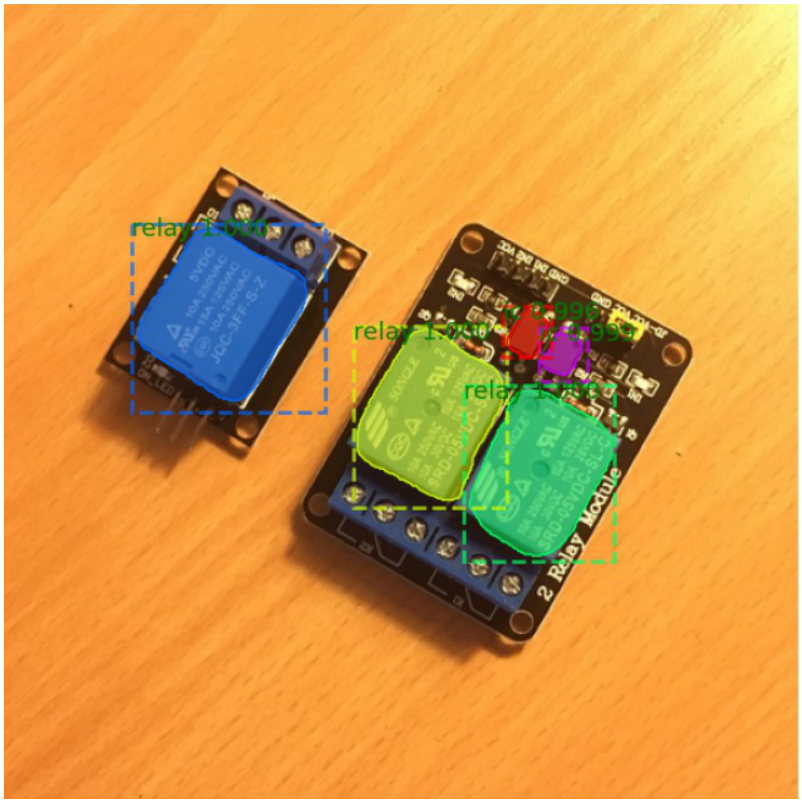



led 1.000
led 0.999
led 1.000
led 1.000
led 1.000
led 0.999


capacitor 0.961

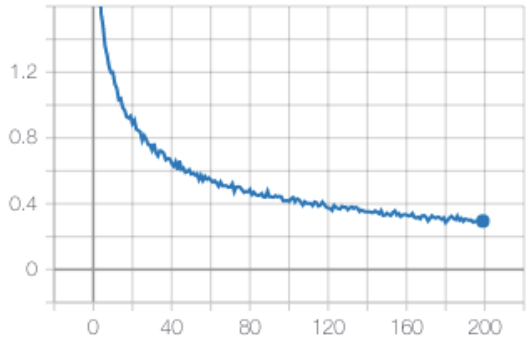
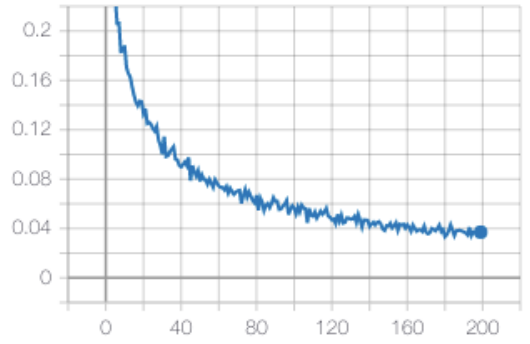
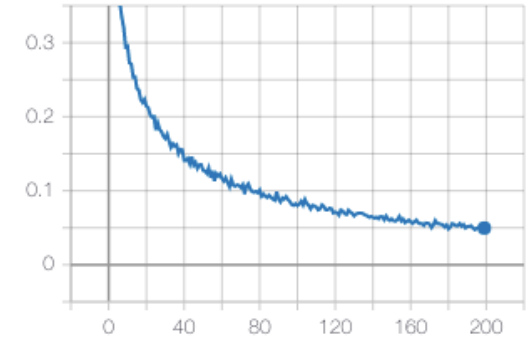
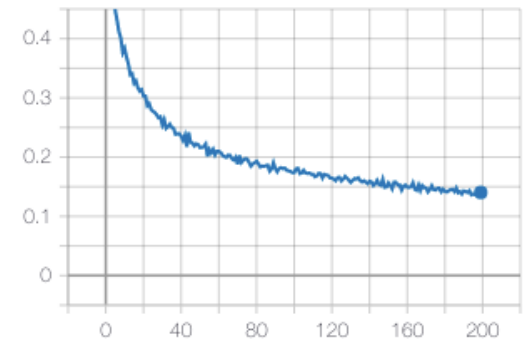
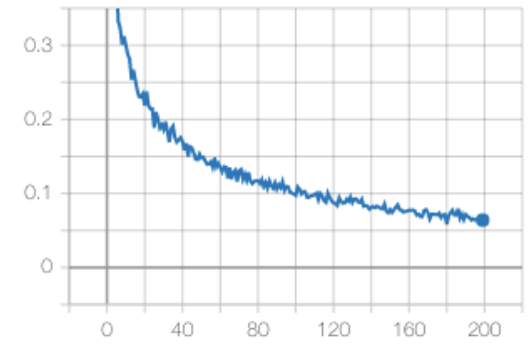
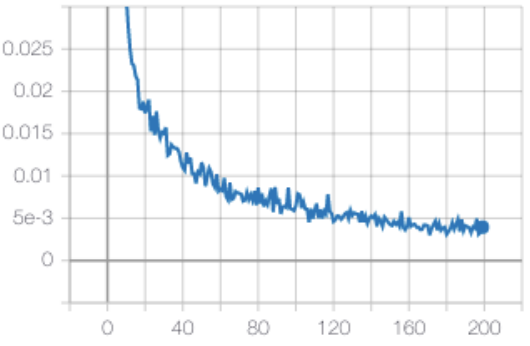

led1.000
capacitor 0.990
led 0.999


capacitor 1.000
capacitor 0.999
capacitor 1.000
capacitor 1.000
capacitor 1.000
hand 0.990
capacitor 1.000
capacitor 0.999
capacitor 1.000
capacitor 1.000
capacitor 1.000


relay 1.000
relay 1.000
relay 1.000
2 Relay Module

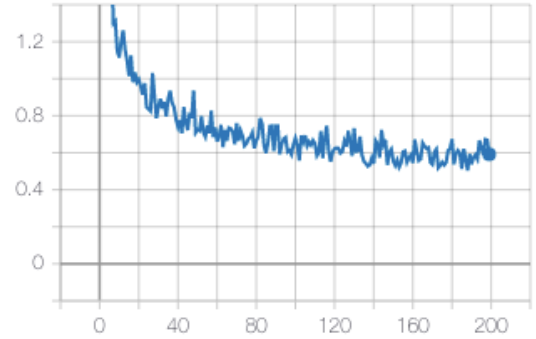
B Hodnoty ztrátových funkcí

Tab. B.1: Průběh hodnot ztrátových funkcí (trénovací dataset – ResNet101).

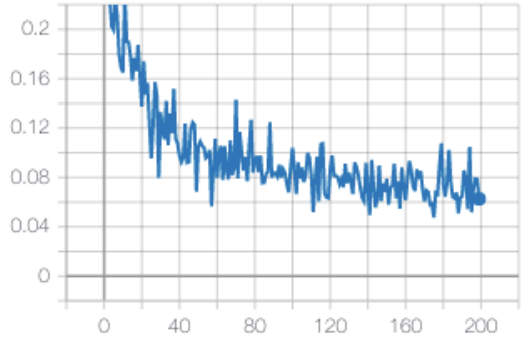
Loss (L) 	Mask R-CNN class loss (L_{class}) 
Mask R-CNN bounding box loss (L_{bbox}) 	Mask R-CNN mask loss (L_{mask}) 
RPN bounding box loss 	RPN class loss 
Horizontální osa udává počet epoch, vertikální osa udává hodnotu ztrátové funkce	

Tab. B.2: Průběh hodnot ztrátových funkcí (validační dataset – ResNet101).

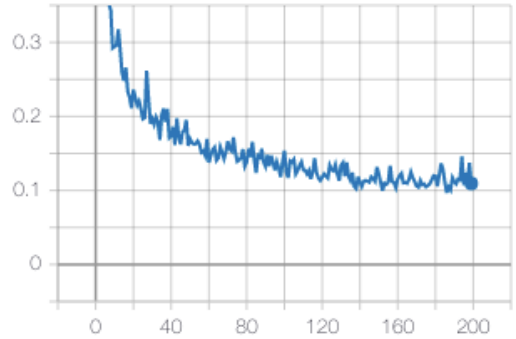
Loss (L)



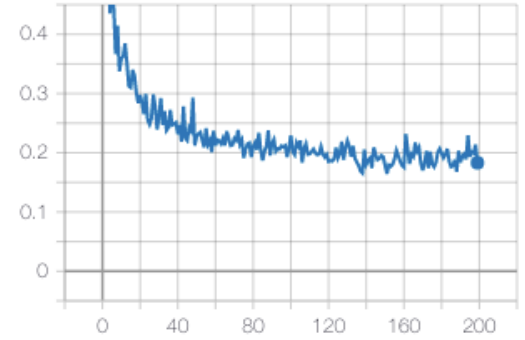
Mask R-CNN class loss (L_{class})



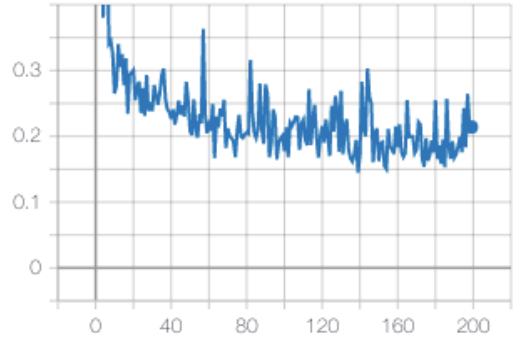
Mask R-CNN bounding box loss (L_{bbox})



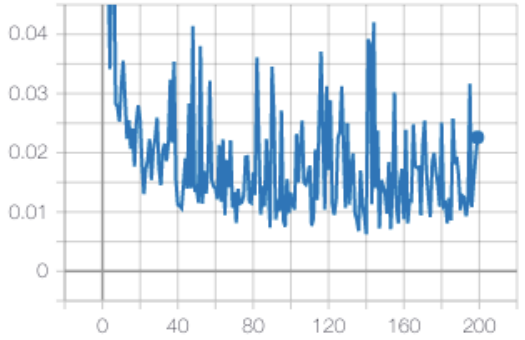
Mask R-CNN mask loss (L_{mask})



RPN bounding box loss

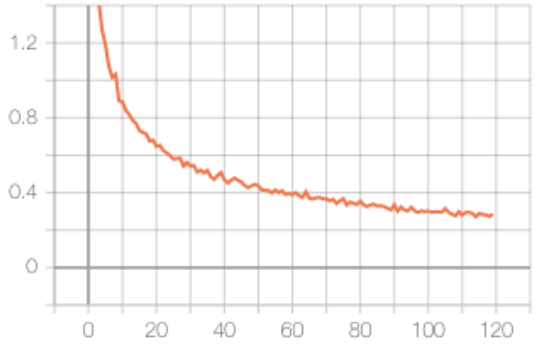
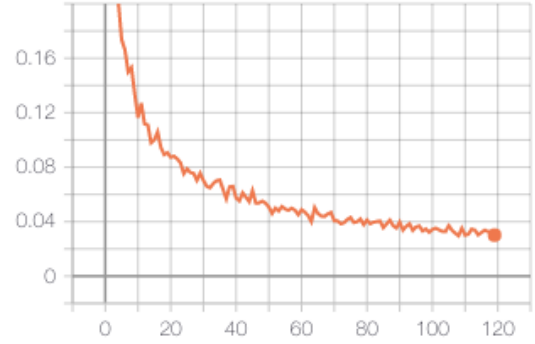
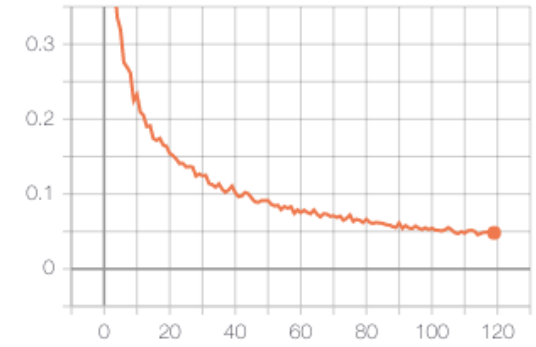
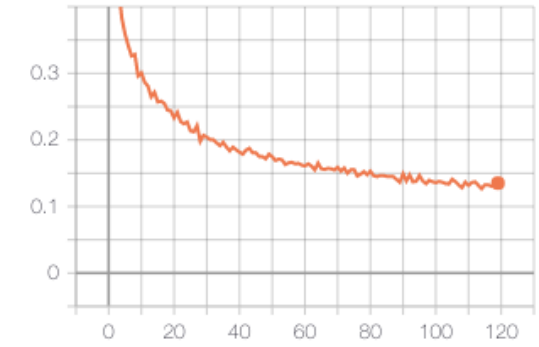
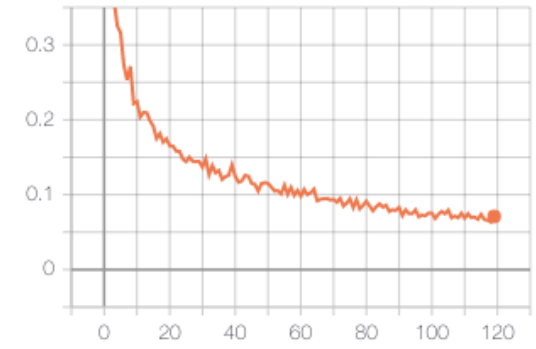
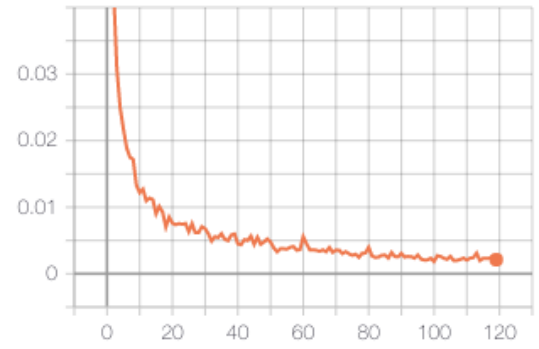


RPN class loss



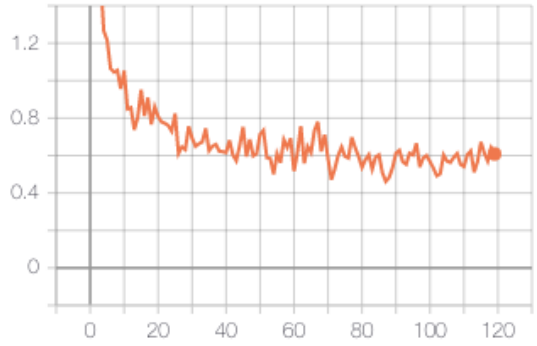
Horizontální osa udává počet epoch, vertikální osa udává hodnotu ztrátové funkce

Tab. B.3: Průběh hodnot ztrátových funkcí (trénovací dataset – ResNet50).

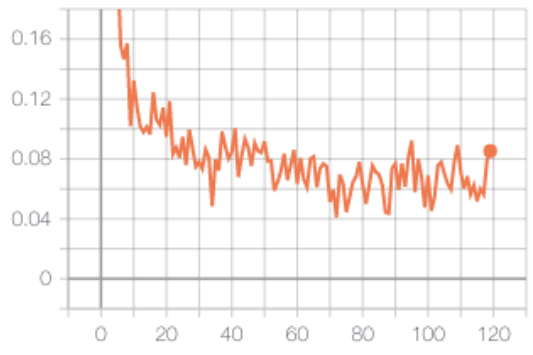
Loss (L) 	Mask R-CNN class loss (L_{class}) 
Mask R-CNN bounding box loss (L_{bbox}) 	Mask R-CNN mask loss (L_{mask}) 
RPN bounding box loss 	RPN class loss 
Horizontální osa udává počet epoch, vertikální osa udává hodnotu ztrátové funkce	

Tab. B.4: Průběh hodnot ztrátových funkcí (validační dataset – ResNet50).

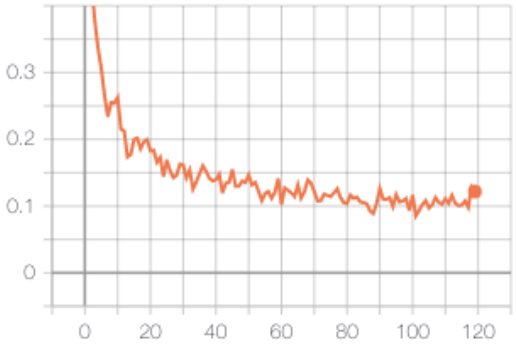
Loss (L)



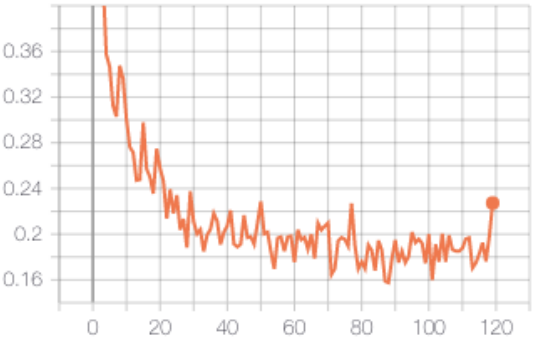
Mask R-CNN class loss (L_{class})



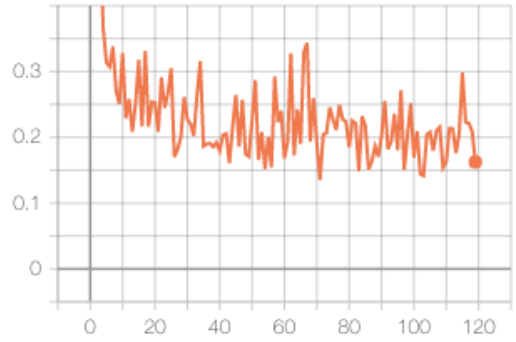
Mask R-CNN bounding box loss (L_{bbox})



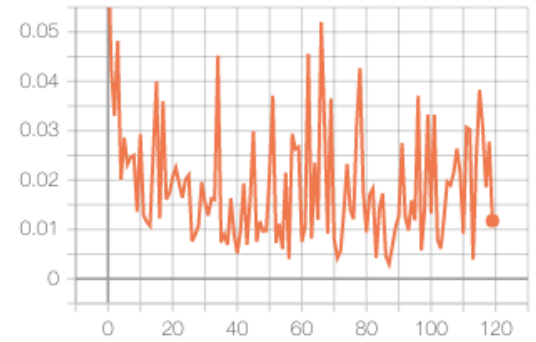
Mask R-CNN mask loss (L_{mask})



RPN bounding box loss



RPN class loss



Horizontální osa udává počet epoch, vertikální osa udává hodnotu ztrátové funkce

C Příkazy pro manipulaci se segmentační sítí

- Příklad spuštění učícího procesu sítě:

```
python PCB.py -m train -e 200 -s 300 -l 0.0015
-b resnet101 -a light -r y
```

- Příklad spuštění segmentace dat:

```
python PCB.py -m segmentation -b resnet101 -r y
```

- Příklad spuštění výpočtu metrik:

```
python PCB.py -m metrics -b resnet101 -r y
```

- Příklad úpravy rozlišení datasetu

```
python PCB_resize_dataset.py -dim 512 -path_in
folder_A -path_out folder_B
```

D Návod na instalaci a spuštění sítě

1. Spuštění segmentační sítě vyžaduje následující prerekvizity:
   - OS Windows 10 (nebo prostředí Google Colaboratory),
   - Python 3 (testováno na verzi 3.6.1),
   - CUDA Toolkit 10.0.
2. Přiložené soubory přemístíme z CD do disku počítače.
3. Do kořenového adresáře stáhneme předučené váhy pro ResNet50 dostupné z: `https://github.com/fchollet/deep-learning-models/releases/download/v0.2/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5`.
4. V hlavním souboru `PCB.py` upravíme cesty v sekci *Default configuration* dle vlastní potřeby.
5. Provedeme build Mask R-CNN přesunutím do adresáře `main Mask_RCNN` a spuštěním příkazu `python setup.py install`.
6. Pomocí příkazu `pip install -r requirements.txt` nainstalujeme potřebné balíčky.
7. Soubory určené k segmentaci nahrajeme do adresáře `test_files`.
8. Pro manipulaci se segmentační sítí využijeme příkazy uvedené v příloze C.

E Obsah přiloženého CD

- `/`
  - `PCB_Mask_RCNN` ..... kořenový adresář projektu
    - `examples` ..... ukázky získaných výsledků
      - `RN50` ..... ResNet50 (Konfigurace 1)
        - `segmentation` ..... výsledky segmentace
          - `...jpg`
          - `results.html`
        - `learning.html` ..... výsledky učení
        - `metrics.html` ..... vyhodnocení metrik
      - `RN101` ..... ResNet101 (Konfigurace 2)
        - `segmentation` ..... výsledky segmentace
          - `...jpg`
          - `results.html`
        - `learning.html` ..... výsledky učení
        - `metrics.html` ..... vyhodnocení metrik
    - `log_files` ..... logovací a h5 soubory
    - `main`
      - `dataset`
        - `training` ..... trénovací dataset
          - `...jpg`
          - `annotations.json` ..... anotace trénovacího datasetu
        - `validation` ..... validační dataset
          - `...jpg`
          - `annotations.json` ..... anotace validačního datasetu
        - `testing` ..... testovací dataset
          - `...jpg`
          - `annotations.json` ..... anotace testovacího datasetu
      - `Mask_RCNN` ..... Matterport projekt
      - `YOLO` ..... YOLO projekt
    - `PCB.py` ..... Hlavní skript
    - `PCB_metrics.py` ..... Skript k měření metrik
    - `PCB_resize_dataset.py` ..... Skript pro úpravu datasetu
    - `PCB_results.py` ..... Skript pro generování HTML výstupu
    - `PCB_yolo.py` ..... Skript k detekci rukou
    - `requirements.txt` ..... Seznam nutných balíčků
    - `results` ..... výsledky predikcí a metrik
    - `templates` ..... šablony pro HTML reporty
      - `training_template.html` ..... šablona pro HTML report učení sítě
      - `results_template.html` ..... šablona pro HTML report segmentace
      - `metrics_template.html` ..... šablona pro HTML report metrik
    - `test_files` ..... testovací obrázky pro segmentaci

Z důvodu malé kapacity CD nejsou výsledné váhy a soubory Matterport projektu a YOLO projektu přiloženy a je nutné je stáhnout z uvedených zdrojů. Vytvořený dataset je dostupný na adrese `https://github.com/xhrdym03/PCB_dataset`.