Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra kybernetiky

BAKALÁŘSKÁ PRÁCE

Plzeň, 2023 **Jan Viktora**

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Jan VIKTORA**
Osobní číslo:	**A20B0393P**
Studijní program:	**B0714A150005 Kybernetika a řídicí technika**
Specializace:	**Umělá inteligence a automatizace**
Téma práce:	**Analýza výkonu operatéra během nácviku chirurgického šití z videa**
Zadávající katedra:	**Katedra kybernetiky**

Zásady pro vypracování

1. Seznamte se s problematikou vyhodnocování událostí z videosekvencí. Zvláště se zaměřte na aplikace v medicíně a chirurgii.
2. Navrhněte a implementujte metodu hodnocení výkonu operatéra během nácviku chirurgického šití.
3. Ověřte funkci navržené metody prostřednictvím experimentu a vhodným způsobem vyhodnoťte funkčnost.

Rozsah bakalářské práce: **30-40 stránek A4**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná**

Seznam doporučené literatury:

- Čihák, R. (2016). Anatomie 3: Třetí, upravené a doplněné vydání. Grada Publishing.
- Šonka, M., Hlaváč, V., & Boyle, R. (2014). Image Processing, Analysis, and Machine Vision Second Edition. Thomson-Engineering.
- Lee, D., Won Yu, H., Kwon, H., Kong, H.-J., Eun Lee, K., & Chan Kim, H. (1964). Evaluation of Surgical Skills during Robotic Surgery by Deep Learning-Based Multiple Surgical Instrument Tracking in Training and Actual Operations. J. Clin. Med, 2020. https://doi.org/10.3390/jcm9061964
- Liška, V., Baxa, J., Beneš, J., Brůha, J., Ferda, J., Hošek, P., Jansová, M., Jiřík, M., Jonášová, A., Králíčková, M., Křečková, J., Křen, J., Lobovský, L., Lukeš, V., Mírka, H., Pálek, R., Pešta, M., Pitule, P., Rohan, E., ... Vyčítal, O. (2016). Experimental surgery (V. Liška (ed.); 1st ed.). NAVA, s. r. o.
- Goodfellow, IJ., Bengio, Y., Courville, A.: Deep Learning. Cambridge, MA, USA: MIT Press; 2016.

Vedoucí bakalářské práce: **Ing. Miroslav Jiřík, Ph.D.**
Výzkumný program 1

Datum zadání bakalářské práce: **17. října 2022**
Termín odevzdání bakalářské práce: **22. května 2023**

Západočeská univerzita v Plzni
Fakulta aplikovaných věd
L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Prof. Ing. Josef Psutka, CSc.
vedoucí katedry

V Plzni dne 17. října 2022

Prohlášení

Předkládám tímto k posouzení a obhajobě bakalářskou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne ....
Podpis

Poděkování

Chtěl bych tímto rád poděkovat panu Ing. Miroslavu Jiříkovi, Ph.D. za odborné vedení, cenné rady, trpělivost a časté konzultace. Dále bych chtěl poděkovat skupině okolo Prof. Dr. med. Uty Dahmen z Friedrich Schillerovy univerzity v Jeně, za možnost zúčastnit se jejich kurzu šití a za poskytnutí videí. A také bych chtěl poděkovat MetaCentrum (MetaVO), za přístup k výpočetním uzlům.

Abstrakt

Tato práce se zabývá tvorbou algoritmu, který bude schopen detekovat určité medicínské nástroje (jehelec, pinzetu, nůžky) z obrazu. Další algoritmus bude schopen detekovat různé objekty v obraze, tato detekce bude použita k předzpracování videa. Společně tyto detekce vytvoří pro uživatel statistiku a ohodnocení výkonu. Nejprve jsou shrnuté současné přístupy ke zpracování obrazu. Dále jsou popsány datasety potřebné k trénování neuronových sítí. Poslední část se věnuje ověření funkčnosti natrénovaných neuronových sítí.

Klíčová slova

Neuronová síť, detekce, klasifikace, metriky, ztrátová funkce, bounding box, příznaky, konvoluce, transfer learning

Abstract

The focus of this bachelor thesis is to develop an algorithm that will be able to detect certain medical instruments (needle holder, tweezers, scissors) from an image. Another algorithm will be able to detect different objects in the image, this detection will be used to pre-process the video. Together these detections will produce statistics and performance ratings for the user. First, the current approaches to image processing are summarized. Next, the datasets needed to train the neural networks are described. The last section is devoted to validating the performance of the trained neural networks.

Keywords

Neural network, detection, classification, metrics, loss function, bounding box, features, convolution, transfer learning

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

V posledních letech došlo v oblasti strojového učení k velkému posunu. Tento pokrok ovlivňuje širokou řadu oblastí, mezi ně patří i zdravotnictví. I složitá oblast jako je chirurgie, může těžit z využití technologií založených na umělé inteligenci. Použití těchto technologií přináší lepší výsledky chirurgických zákroků, větší bezpečnost pacientů nebo zlepšení tradičních přístupů. Jednou z hlavních úloh zdravotnictví je příprava a vzdělávání budoucích lékařů. Například nácvik chirurgických zákroků, které se skládají z mnoha částí jako předoperační posouzení, podání anestezie, provedení řezu, provedení zákroku, uzavření rány a další. Pro řadu těchto úkolů potřebuje chirurg umět kvalitně šít.

Běžnou praxí je, že šití učily zkušení odborníci. Tento způsob je časově velice náročný, je zapotřebí velké množství expertů a musí se uvažovat i lidský faktor, protože každý chirurg provádí šití svým vlastním stylem. Když v roce 2020 zasáhla svět pandemie COVIDu-19,, bylo nutné přesunout výuku do online prostoru. Některé obory do tohoto režimu přešly bez problému, ale obecně manuální obory nejsou pro výuku přes internet vhodné. Tým okolo Prof. Dr. med. Uty Dahmen se přesto o tento krok pokusil. Konkrétně přesunuly kurz šití na platformu ZOOM. Lektoři tedy musejí vysvětlovat látku a opravovat studenty za pomocí mobilního telefonu a malé obrazovky na počítači. Tím se náročnost vyučování ještě zvýšila. Tento program by jim měl alespoň částečně ulehčit práci.

Tato práce je součástí většího projektu mezi Fakultou aplikovaných věd Západočeské univerzity v Plzni a Friedrich Schillerovou univerzitou v Jeně, který se snaží vytvořit program na vyhodnocení chirurgického šití z video záznamu. S využitím algoritmů a modelů strojového učení by program měl zhodnotit nahrané video a uživateli poskytnout statistiky a objektivní hodnocení kvality úkonu. Úkolem této práce je natrénovat modely pro detekci nástrojů (jehelce, pinzety, nůžek) a detekci objektů, které se ve videu nachází. Druhá detekce poběží pouze na několika prvních snímcích videa a využije se k předzpracování videa, které bude spočívat v odstranění částí obrazu (rozhraní ZOOMu) nebo detekce některých objektů se použije pro výpočet poměr pixel:centimetr. Postup pro tuto práci je následující:

1. Sběr dat a tvorba datasetu pro obě úlohy
2. Výběr modelu nebo modelů pro získání informací z videa
3. Trénink a optimalizace těchto modelů
4. Vyhodnocení kvality natrénování modelů na testovací sadě

Obrázek 1: Vyučovací prostor při pandemii COVIDu-19

Struktura této práce je následující. V kapitole 2 je popis neuronových sítí, které se využívají na zpracování obrazu. Kapitola 3 popisuje úlohy z oblasti zpracování obrazu a moderní způsob řešení. Kapitola 4 popisuje dataset, použitý pro trénování modelů, které je s výsledky vysvětleno v kapitole 5.

2 Neuronové sítě

Neuronové sítě jsou algoritmus používaný v oblasti strojového učení. Za poslední desetiletí se dočkaly obrovského vývoje a v současnosti se používají v široké řadě aplikací na rozpoznávání obrazu, predikci událostí nebo zpracování přirozeného jazyka.

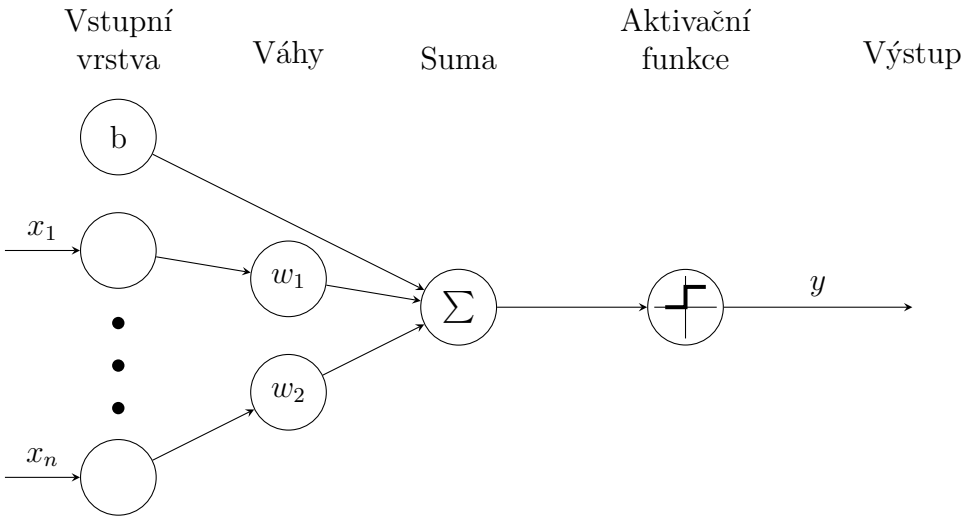
Stavebním kamenem neuronových sítí je neuron (perceptron), což je matematický model biologického neuronu. Každý umělý neuron má n vstupů, každý s jinou váhou. Hodnoty vstupů se v neuronu sečtou a použijí jako argument aktivační funkce. Hodnota aktivační funkce pak slouží jako výstup neuronu. Mezi nejčastěji používané aktivační funkce patří sigmoidální funkce nebo tzv. ReLU (Rectified Linear Unit).

V neuronové sítí jsou neurony uspořádány do vrstev a tyto vrstvy do řady za sebe. Vstupem dané vrstvy je výstup předchozí a výstup této vrstvy je vstupem následující. První vrstva se označuje jako vstup sítě a její velikost je závislá na typu vstupních dat. Poslední vrstva je výstupní a její struktura a hodnoty záleží na typu úlohy. V každé vrstvě se nachází práh, což je neuron, který nemá žádný vstup a jeho hodnota je vždy 1. Jeho výstup tedy ovlivňuje pouze jeho váha[1].

2.1 Historie neuronových sítí

Základy neuronových sítí položili Warren McCulloch a Walter Pitts v roce 1943[2], kteří se zabývali chováním neuronů v mozku a jak tyto procesy popsat matematicky. Výsledek jejich práce se považuje za první model umělé neuronové sítě. V roce 1957 Frank Rosenblatt vytvořil matematický model perceptronu[3], který dokázal rozdělit vstupní data do dvou tříd. Perceptron je model pouze jednoho lidského neuronu, který má n vstupů a binární výstup. Jeho schéma je znázorněné na obrázku 2.

Na základě modelu perceptronu začaly vznikat vícevrstvé modely s více neurony v jedné vrstvě. Tyto typy sítí se nazývají vícevrstvé dopředné sítě (feedforward networks). Během 60. a 70. let byl o neuronové sítě velký zájem. Koncem 70. let a v 80. letech vědci začali narážet na limity nejen neuronových sítí, ale také hardwaru potřebného pro práci s nimi.



Obrázek 2: Model perceptronu

Zájem tedy přibližně na 20 let upadl[4]. Ke konci prvního desetiletí 21. století se začalo experimentovat s hlubokými neuronovými sítěmi, které jsou trénovány na řádově miliónech dat. Jde o sítě, které mají několik desítek až stovek různých typů vrstev. Počet parametrů takových sítí je v miliónech, v současnosti některé sítě mají i přes miliardu parametrů[5][6]. Tento posun byl umožněn především díky vývoji v oblasti hardwaru, použití grafických karet a obrovskému množství volně dostupných dat, ze kterých vznikla jedna z nejdůležitějších databází této doby ImageNet[7]. V roce 2012 neuronová síť nazvaná AlexNet[8] vyhrála soutěž pořádanou na ImageNet datasetu zvanou ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Tato síť měla daleko lepší výsledky než její předchůdci a položila základy pro další vývoj konvolučních neuronových sítí.

2.2 Trénování neuronové sítě

To, co odlišuje neuronové sítě od ostatních algoritmů používaných v oblasti strojového učení, je jejich schopnost učit se z velkého množství dat a nacházet vzory i v komplexních datových formátech jako je obraz nebo zvuková stopa.

Proces nastavení neuronové sítě se nazývá trénování. Existují dva typy trénování: bez učitele - do sítě jsou na vstup postupně vkládány data bez jakékoliv informace o nich a síť se je snaží shlukovat podle podobnosti, a trénování s učitelem - pro každý vstup existuje také požadovaný výstup. Do sítě se tedy vloží vstup, ta vygeneruje výstup, který se porovná s požadovaným výstupem. Jejich rozdíl je zpětně propagován skrze síť. Váhy se upravují takovým způsobem aby rozdíl mezi požadovanými a vygenerovanými vstupy byl minimální.

$$W^{(l)} = W^{(l)} - \alpha \frac{\partial J}{\partial W^{(l)}} \tag{1}$$

$$b^{(l)} = b^{(l)} - \alpha \frac{\partial J}{\partial b^{(l)}} \tag{2}$$

$W^{(l)}$ jsou váhy vrstvy l, $b^(l)$ je práh vrstvy l, α je konstanta učení, která ovlivňuje rychlost změny parametrů. Velká hodnota konstanty učení může snížit čas konvergence k optimu, ale může se také stát, že algoritmus se k optimální hodnotě pouze přiblíží, nebo v krajních případech dokonce diverguje. Malá hodnota konstanty učení zajišťuje, že algoritmus nebude divergovat, ale zároveň zpomaluje proces učení a hrozí, že algoritmus se zasekne v lokálním minimu. V současné době se nepoužívá konstanta učení pouze jako jedno číslo, ale mění se během trénování k optimalizaci tohoto procesu. Výraz $\frac{\partial J}{\partial W^{(l)}}$ je gradient ztrátové funkce, kterou se snaží síť minimalizovat.

$$J(W, b) = \frac{1}{m} \sum_{i=1}^{m} ||\hat{y}_i - y_i|| \tag{3}$$

Kde $\hat{y}$ značí požadovaný výstup sítě a y_i skutečný výstup sítě. Možností, jak definovat ztrátovou funkci, je více a záleží na konkretní úloze. Mezi nejpopulárnější patří střední kvadratická chyba (MSE, L2), binární křížová entropie atd.

Trénování předchází volba architektury sítě a náhodná inicializace vah. Architektura sítě je závislá na mnoha věcech. Například s jakým typem dat se pracuje (obraz, zvuk, časová řada atd.), existují-li ke vstupní datům i požadované výstupy, kolik je trénovacích dat a mnoho dalších. Obecné doporučení pro inicializaci je použít malé náhodné hodnoty v okolí nuly. Existují i složitější metody jako He initialization[9], Xavier initialization[10] a další.

2.2.1 Transfer learning

V současné době mají neuronové sítě používané pro úlohy spojené se zpracováním obrazu desítky až stovky vrstev a desítky až stovky miliónů parametrů. Natrénovat takovéto modely od základu vyžaduje několik grafických karet a dataset s milióny obrazů. Z pochopitelných důvodu je tento přístup nepoužitelný pro velkou řadu úloh.

Tento problém se snaží řešit technika zvaná transfer learning. Nejdříve se vybere model, který byl natrénovaný na velkém datasetu a ten se použije jako výchozí model pro další trénování. To už probíhá na menším datasetu pro konkrétní úlohu. Ve většině případů se při trénování zmrazí všechny váhy původního modelu a mění se pouze ty váhy, které byly do architektury sítě přidány. Za původní výstupní vrstvu se pak přidá pouze jedna vrstva. Tato vrstva slouží jako výstupní vrstva nového modelu. Její topologie odpovídá potřebám úlohy. V některých aplikacích se nemusí zmrazit všechny váhy původního modelu a při trénování se tyto váhy dotrénují. To která část sítě je zmražená záleží na konkrétní aplikaci. Tento přístup se stal běžným v úlohách, kde velikost datasetu neumožňuje trénovat celou síť od začátku.

2.3 Architektura neuronových sítí

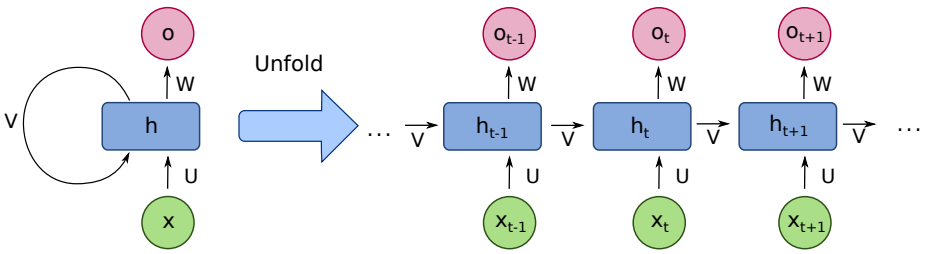
Architektura neuronové sítě popisuje kolik vrstev neuronová síť má, jak jsou vrstvy navzájem propojené, jak vrstvy vypadají a další. Mezi základní architektury patří dopředné, rekurentní a konvoluční sítě. Tyto sítě ovšem nebývají složené pouze z jednoho typu vrstev. Například konvoluční sítě mají často na začátku několik konvolučních vrstev a za ně jsou vložené dopředné vrstvy.

2.3.1 Dopředné vrstvy

Základní architekturou je dopředná síť. Což je vícevrstvá síť, kde se informace pohybuje od vstupu k výstupu skrze sérii skrytých vrstev. Počet neuronů ve vstupní vrstvě je závislý na vstupní informaci. Výstupní vrstva generuje predikci anebo klasifikuje vstup do jedné z tříd. Skryté vrstvy slouží k nalezení příznaků ve vstupních datech. Dopředné sítě se často používají jako regresory nebo klasifikátory.

2.3.2 Rekurentní vrstvy a sítě

Rekurentní vrstvy se liší tím, že se v síti nachází i zpětné vazby. Tedy výstup některých neuronů může zároveň být vstupem té samé vrstvy. Jako vstup se často používají sekvenční data a časové řady. Dopředné sítě považují jednotlivé vstupy za nezávislé, zatímco u rekurentních sítí mohou předchozí vstupy ovlivňovat současné i budoucí výstupy. Tato vlastnost umožňuje síti udržovat si jakousi paměť a tedy zachytit závislosti v čase. Proto jsou rekurentní sítě (RNNs) vhodné pro zpracování přirozené jazyka nebo na úlohy rozpoznání řeči, překladu a dalších[11]. Ilustrace práce rekurentní sítě je zobrazena na obrázku 3.

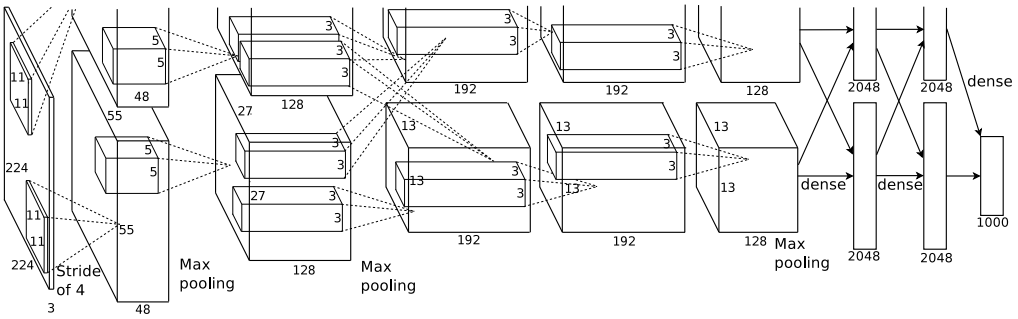


Obrázek 3: Ilustrace rekurentní sítě

2.3.3 Konvoluční vrstvy

Konvoluční vrstva se skládá z filtrů, ty často bývají reprezentovány maticí (většinou čtvercovou - 3x3, 5x5 atd.). Tyto filtry se postupně posouvají přes vstupní data a na každém místě provedou s částí vstupních dat konvoluci. Výstupem z konvoluční vrstvy je mapa příznaků reprezentována maticí. Tyto mapy zachycují lokální vzory jako hrany, tvary nebo struktury. Poskládání několika konvolučních vrstev za sebe umožňuje síti rozpoznat v datech i abstraktnější příznaky.

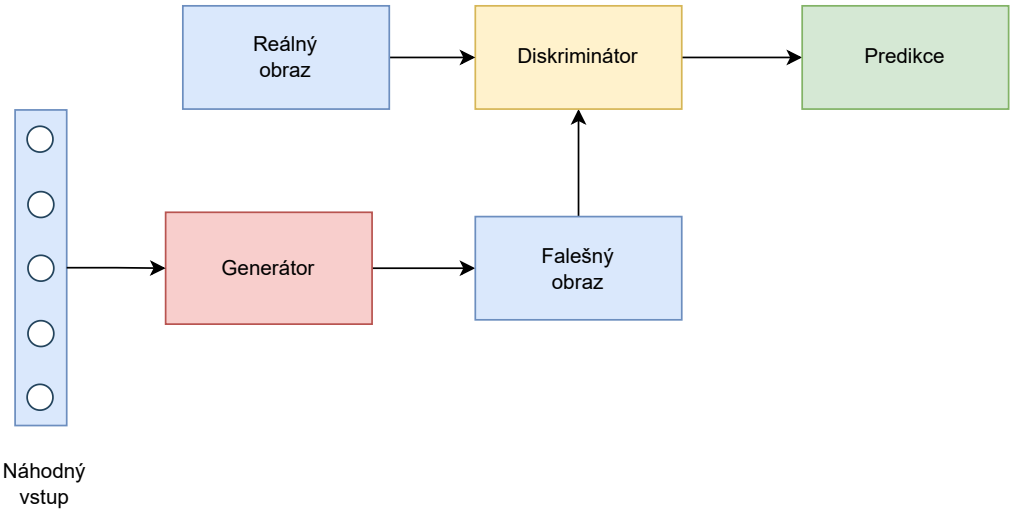
Konvoluční vrstvy jsou často používané v kombinaci s dopřednými vrstvami, kde konvoluční část sítě slouží jako generátor příznaků a dopředná část jako klasifikátor a regresor. Příznakové mapy se musí tedy upravit, aby mohli být použity jako vstup do dopředné vrstvy, která vyžaduje vektor. Nejdříve se na příznakové mapy aplikuje operace zvaná pooling (max pooling, average pooling atd.), která sníží dimenzi těchto map. Pooling často vybírá jednu hodnotu z malé oblasti matice. Například operace max pooling vybere nejvyšší hodnotu z dané oblasti. Výstup z této operace se zarovná do vektoru. Nevýhodou zarovnání je, že vygenerované příznaky částečně ztrácí prostorovou strukturu, která je například při zpracování obrazu velmi důležitá. Konvoluční sítě umožnily velký posun v oblasti zpracování obrazu. Ale používají se i v jiných aplikacích jako ve zpracování přirozeného jazyka nebo signálu[12]. Část architektury konvoluční sítě je znázorněna na obrázku 4[8]. V tomto schéma sou vidět konvoluční vrstvy, Max pooling operace a dopředná část sítě.



Obrázek 4: Část konvoluční architektury ImageNetu

2.3.4 GAN

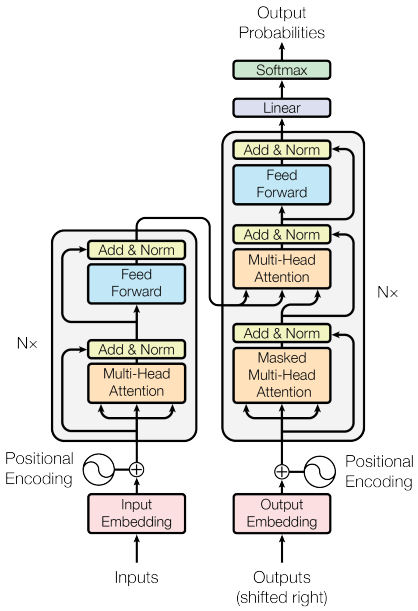
GANs - generative adversarial networks je neuronová síť složená ze dvou modelů: generátoru a diskriminátoru. Při trénování se modely trénují společně. Na vstup generátoru se přivede vektor délky n a jako výstup se vygeneruje falešný obraz. Na vstup diskriminátoru se přivede falešný obraz vytvořený generátorem anebo reálný obraz. Úkolem generátoru je vytvořit obrazy co nejpodobnější reálným, aby diskriminátor nebyl schopný rozpoznat falešný obraz od reálného. Diskriminátor se snaží rozpoznat výstup generátoru od reálného obrazu. Toto odpovídá hře minimax pro dva hráče, kde jeden model se snaží minimalizovat užitek druhého. Schéma práce GANů je na obrázku 5. GANs se používají k vytváření realistických obrazů například v oblastech, kde získat reálná data může být velice namáhavé[13][14].



Obrázek 5: Ukázka architektury GAN

2.3.5 Transformer

Transformer je architektura neuronové sítě, která má strukturu enkodér-dekodér. Podobně jako rekurentní sítě, pracuje se sekvenčními daty. Transformery částečně navazují na rekurentní sítě, ale snaží vypořádat s problémy, které při trénování a používání rekurentních sítí nastávají.



Obrázek 6: Architektury transformeru

Problémem u rekurentních sítí je to, že zapracovávají vstup prvek po prvku, snižuje se tím možnost paralelizace trénování. Klíčovou vlastností transformerů je technika zvaná self-attention, která váží důležitost všech elementů vstupní posloupnosti. Což umožňuje modelu věnovat se důležitým částem sekvence a rozpoznat závislosti na větší vzdálenosti v posloupnosti[15]. V současné době transformery předvádějí lepší výsledky než rekurentní sítě[16]. Obecnou strukturu transformerů popisuje obrázek 6[15].

Mezi další architektury patří například deep residual network (ResNet)[17], autoenkódery[18], LSTM (Long Short-Term Memory - podobné rekurentním sítím)[19] a další.

2.4 Použité technologie

V této sekci jsou zmíněny často používané technologie pro práci s neuronovými sítěmi. Zároveň byly tyto technologie použity k vypracování této bakalářské práce. K trénování neuronových sítí a přípravě dat byl použit výhradně Python. Mezi Python knihovny, které v této sekci nejsou zmíněny stojí za zmínění NumPy[20], Matplotlib[21] a Albumentation[22].

2.4.1 Python

Python je populární vysokoúrovňový programovací jazyk, který poprvé představil Guido van Rossum v roce 1991. Jeho filozofie návrhu klade důraz na čitelnost a jednoduchost kódu, což z něj činí ideální jazyk pro začátečníky i zkušené programátory. Syntaxe jazyka Python je jasná a stručná, k ohraničení bloků kódu se používají bílé znaky místo tradičních složených závorek, což usnadňuje čtení a psaní. Jeho výkonná standardní knihovna a balíčky třetích stran poskytují podporu pro širokou škálu úloh, od vývoje webových stránek a analýzy dat až po vědecké výpočty a umělou inteligenci. Obliba jazyka Python v průběhu let neustále roste a v současné době je široce používán v průmyslu, akademické sféře i ve výzkumu[23].

2.4.2 PyTorch

PyTorch je knihovna zaměřená na strojové učení, napsaná převážně v jazyce C++. Původně ji vyvíjela společnost Facebook (nyní Meta), ale od září roku 2022 je zodpovědná za další vývoj knihovny PyTorch Foundation, dceřiná společnost neziskové organizace Linux Foundation. Hlavním záměrem tvůrců PyTorche bylo zachovat rychlost předchozích knihoven, jako jsou TensorFlow[24] a Caffe[25], zároveň však vytvořit knihovnu, která bude snadno použitelná. Tato rovnováha mezi rychlostí a jednoduchostí dělá knihovnu

PyTorch tak oblíbenou. Přestože se PyTorch často spojuje s jazykem Python, velká část knihovny je napsána v C++. Důvodem je tzv. global interpreter lock (GIL)[26], který zabraňuje běhu více vláken najednou v jazyce Python (konkrétně v CPythonu - nejrozšířenější implementace Pythonu)[27]. Při počítání derivací by GIL mohl výrazně zpomalovat běh programu, proto byl výkonově náročný kód implementován v jazyce C++[28].

2.4.3 MMDetection

MMDetection je sada nástrojů pro detekci objektů založená na PyTorchy. MMDetection vznikl z kódové základny MMDet týmu, který v roce 2018 vyhrál COCO Challenge 2018. Obsahuje nejen kódy pro trénování a inferenci, ale také váhy pro více než 200 modelů neuronových sítí. V MMDetection lze provádět širokou řadu úloh spojenou s neuronovým sítěmi jako trénování sítí od základu, fine-tunning, použití již natrénovaných modelů a další. Jeho vývoj má na starosti MMLab (Multimedia Laboratory), což je laboratoř na Čínské univerzitě v Hongkongu. MMLab vyvíjí mnoho nástrojů pro zpracování obrazových dat, například MMPose, MMClassification, MMCV a další. Výhodu má MMDetection oproti podobným nástrojům, jako je Detectron[29] nebo SimpleDet[30] v rozsahu sbírky modelů tzv. „Model ZOO“. Jedná se o sbírku všech modelů, které jsou v MMDetection implementovány[31].

3 Zpracování obrazu

Zpracování obrazů je jednou z oblastí strojového učení. Zabývá se analýzou a získáváním informací z obrazu. Často používané techniky jsou například filtrace, prahování nebo segmentace, které se používají pro získání informací, které se dále zpracovávají. V současné době se pro většinou úkolů z této oblasti používají neuronové sítě. Aplikace zpracování obrazu se nachází v robotice, zdravotnictví, dálkovém snímkování země a dalších.

3.1 Detekce objektů

Je jednou ze základních úloh počítačového vidění. Úkolem je najít objekty reálného světa v obrazu nebo ve videu a ohraničit je tzv. bounding boxem (obdélník, který ohraničuje oblast ve které se objekt nachází). Vstupem je tedy obraz a výstupem je určitý počet bounding boxů a označení, do které třídy jednotlivé bounding boxy patří. Jedná se o zásadní problém v počítačovém vidění s mnoha aplikacemi do reálného světa jako bezpečnostní

kamery, autonomní řízení nebo medicínské zobrazování. Přesná detekce může být těžký úkol kvůli měnícímu se prostředí. Změny mohou nastávat v poloze kamery (měřítku), změně v osvětlení (jas) a další. Velký posun v oblasti neuronových sítí zaručil i posun v oblasti detekce objektů.

3.2 Segmentace instancí

Segmentace instancí je další úloha z počítačového vidění. V podstatě jde o stejný úkol jako v detekci objektů, s tím rozdílem, že segmentace instancí je podrobnější. Úkolem je detekovat objekty na úrovni pixelů. Výsledkem je tedy zařazení každého pixelu do nějaké třídy. Pokud se v obraze nachází více instancí jedné třídy, tak algoritmy přiřadí každému pixelu ID, aby šlo rozpoznat ke které instanci daný pixel patří. Segmentace instancí je přesnější, ale zároveň náročnější způsob detekce objektů. Podobně jako detekce objektů má mnoho aplikací například v robotice, augmentované realitě, nebo v analýze videí.

3.3 Metriky

Pro detekci objektů bylo vyvinuto velké množství metrik, podle kterých se hodnotí kvalita detekce. Základní a nejpoužívanější metrikou je IoU (Intersection over Union), jejíž hodnota se počítá následovně:

$$\text{IoU} = \frac{S(A \cap B)}{S(A \cup B)} \tag{4}$$

$S(A \cap B)$ je obsah průniku oblastí A a B, $S(A \cup B)$ je sjednocení těchto oblastí. A je skutečná oblast ve které se objekt nachází a B je systémem odhadnutá oblast. Hodnota IoU se pohybuje v intervalu $\langle 0, 1 \rangle$, čím více se hodnota blíží k 1, tím lepší je predikce, 1 znamená úplnou shodu. Pro vyhodnocení IoU se používá práh α, pokud hodnota IoU je větší než práh α, tak se detekce objektu považuje za správnou, pokud ne považuje se predikce za nesprávnou. Jako úspěšná detekce se často považuje ta z hodnotou IoU > 0.5[32].

Pro popis dalších metrik je zapotřebí zařadit každou predikci do jedné ze čtyř kategorií:

- True Positive (TP) - model predikuje objekt a je to správná predikce
- False Positive (FP) - model predikuje objekt a je to špatná predikce
- False Negative (FN) - model objekt nepredikuje, ale měl by
- True Negative (TN) - model objekt nepredikuje a je to správná predikce

Pomocí těchto skupin se definují následující kategorie. Precision popisuje, kolik pozitivních predikcí bylo ve skutečnosti správně. Je to tedy poměr TP ku součtu TP a FP.

$$\text{Precision}_\text{n} = \frac{\text{TP}_\text{n}}{\text{TP}_\text{n} + \text{FP}_\text{n}} \tag{5}$$

n značí třídu. Tedy pro detekci pěti tříd dostaneme pět hodnot precision.

Recall ukazuje kolik objektů bylo identifikováno správně. Je to tedy poměr TP ku součtu TP a FN.

$$\text{Recall}_\text{n} = \frac{\text{TP}_\text{n}}{\text{TP}_\text{n} + \text{FN}_\text{n}} \tag{6}$$

Při optimální nastavení modelu by hodnoty FP a FN byly nulové (Precision i Recall by se rovnali jedné). Z rovnic 5 a 6 je jasné, že vysoká hodnota Precision znamená, že predikce vytvořené modelem budou ve většině případů správné. Zatímco vysoká hodnota Recall ukazuje, že pokud na obrazu bude objekt, tak ho model z velkou pravděpodobností detekuje. Ideální poměr mezi těmito dvěma hodnotami neexistuje a závisí na konkrétní aplikaci[32].

Precision a Recall se budou měnit z prahem τ. Detekce objektu je reprezentována pomocí tří atributů - třídy, polohy a confidence threshold, tedy na kolik si je model jistý svou detekcí. Při vyhodnocování práce modelu se τ nastaví na konstantu a objekty s vyšším confidence threshold se považují jako detekce. Precision-Recall křivka se získá, tak že pro několik (11, 101 aj.) hodnot τ se spočítá hodnota Recall. Poté pro každou hodnotu Recall se vybere nejvyšší hodnota Precision se stejným τ a tyto body se vynesou do grafu. Následně se interpolací lineární křivkou získá spojitá křivka, kterou je možné integrovat[33].

Další důležitou metrikou je AP (Average Precision), jejíž hodnota se získá jako obsah pod Precision-Recall křivkou.

$$\text{AP} = \int_{r=0}^{1} p(r)dr \tag{7}$$

Metrika, která se v detekčních soutěžích používá nejčastěji je AP počítaná na nějakém IoU prahu (značení AP@_α). Problém s metrikou AP nastává, když se detekuje více než jedna třída. Řešením je mean Average Precision:

$$\text{mAP} = \frac{1}{n} \cdot \sum_{i=1}^{n} \text{AP}_i \tag{8}$$

Zde se n rovná počtu tříd. Mean Average Precision se velice často využívá v dokumentacích neuronových sítí nebo detekčních soutěžích. V těchto

využitích se ale obvykle počítá několik hodnot mAP, protože se používají různé hodnoty prahu IoU pro výpočet AP@$_\alpha$[33]. Za zmínku stojí, že v COCO datasetu se AP a mAP nerozlišují, protože AP je zprůměrována přes počet tříd stejně jako mAP[34].

Stejně jako pro Precision i pro Recall existuje „průměrovací“ metrika AR, s tím rozdílem, že pro výpočet AR se nebere v úvahu confidence threshold. Pro výpočet AR je zapotřebí nejdříve nadefinovat $\mathrm{RC}_{\mathrm{IoU}}(o)$ křivku, kde o je hodnota IoU. Hodnoty této křivky se získají tak, že se zvolí libovolný práh IoU a pro něj se spočte hodnota Recall. Do grafu se na osu x vynesou prahy IoU a na osu y hodnoty křivky $\mathrm{RC}_{\mathrm{IoU}}(o)$. Hodnoty IoU se vybírají z intervalu $\langle 0.5; 1\rangle$[33][35].

$$\mathrm{AR} = \int_{0,5}^{1} RC_{\mathrm{IoU}}(o)do \tag{9}$$

Podobně jako pro mAP se mAR spočte jako průměrné AR. COCO dataset opět nerozeznává rozdíl mezi AR a mAR[34].

$$\mathrm{mAR} = \frac{1}{n} \cdot \sum_{i=1}^{n} \mathrm{AR}_i \tag{10}$$

Pro porovnání, jak dobře si vede model při detekci různě velkých objektů, se v detekčních soutěžích objevují metriky AP a AR s indexy S, M, L ($\mathrm{AP_S}$, $\mathrm{AP_M}$ a $\mathrm{AP_L}$). Tyto indexy značí, jak velké objekty se do výpočtu těchto metrik zahrnují[34].

- S (small) - velikost objektu $< 32^2$ pixelů
- M (medium) - $32^2 <$ velikost objektu $< 96^2$ pixelů
- L (large) - velikost objektu $> 96^2$ pixelů

Poslední z nejznámějších metrik je $\mathrm{F_1}$ skóre. Tato metrika popisuje poměr mezi Precision a Recall. Hodnota $\mathrm{F_1}$ skóre se pohybuje v intervalu $\langle 0; 1\rangle$. Nula značí, že alespoň jeden ze dvojice Precision Recall, se rovná nule. Jedna znamená, že i Precision i Recall se rovná jedné[32] [33].

$$\mathrm{F_1} = 2 \cdot \frac{\mathrm{Precision} \cdot \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}} \tag{11}$$

3.4 Přístup

Historicky byl problém detekce objektů řešen dvěma způsoby. První, takzvané tradiční metody, byly založeny na ručně vybraných příznacích. V těchto algoritmech se nejčastěji používalo posuvné okénko, které se postupně posouvalo přes obraz a klasifikátor poté určil, zda tato část odpovídá nějakému z hledaných objektů. Protože posuvné okénko muselo projít celý obraz a vyzkoušet různé velikosti, tak byly tyto algoritmy často časově i výpočetně náročné. Mezi nejvýznamnější algoritmy této skupiny patří Viola Jones detektor[36], HOG detektor[37] nebo Deformable Part-Based Model (DPM)[38]. Díky vzestupu hlubokého učení byly tyto detektory z velké části nahrazeny neuronovými sítěmi[39].

Druhý přístup, kterým se řeší úlohy okolo detekce objektů, je založen na hlubokém učení. Tyto metody začaly vznikat po roce 2012, kdy tři pracovníci na univerzitě v Torontu zveřejnili ImageNet[8]. Což je hluboká konvoluční neuronová síť, která v té době překonala veškeré klasifikační algoritmy. Představení této sítě bylo jedním z faktorů, který ovlivnil jakési znovuzrození hlubokých konvolučních neuronových sítí. Jejich výhodou je, že konvoluční vrstvy jsou schopné se natrénovat na konkrétní úlohu. A konvoluční vrstvy jsou schopny sami z obrazu získat příznaky, které následně slouží právě k detekci či klasifikaci.

3.5 Vývoj v posledním desetiletí

Jak bylo již zmíněno v sekci 3.4, přístup k detekci objektů se kolem roku 2012 výrazně změnil. Neuronové sítě s architekturou založenou na konvolučních sítích se staly state-of-the-art technikou pro řešení úloh v tomto oboru. Tento přístup se brzy rozdělil na dvě větve.

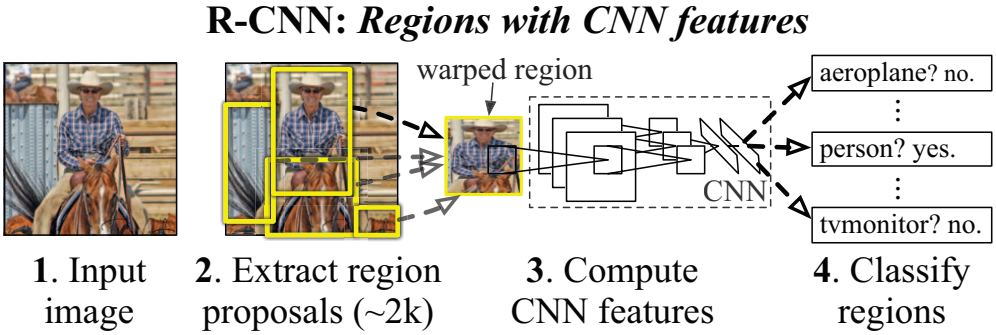
Tyto detektory se dělí na dvě skupiny. První z nich jsou dvoustupňové (vícestupňové) detektory. Tento přístup se rozděluje na více fází. V první fázi se ze vstupního obrazu získají kandidáti na objekty. Na tuto část se používají algoritmy jako Selective Search[40] a EdgeBoxes[41], nebo konvoluční sítě jako RPN (Regions Proposal Network). V druhé fázi se zlepšuje odhad těchto kandidátu a klasifikují se do tříd. Nejdříve získá konvoluční síť příznaky a následně je dopředná síť klasifikuje a vytvoří bounding boxy pro objekty. Mezi tyto detektory patří R-CNN, SPPNet[42], Fast R-CNN, Faster R-CNN.

Druhým přístupem jsou jednostupňové detektory. Ty vynechávají tvorbu kandidátů a rovnou lokalizují a klasifikují objekty. Výhodou těchto detektorů je jejich rychlost. Tím, že vynechávají krok generování kandidátu, který často bývá časově nejnáročnější z celého procesu, tak mají určitou výhodu oproti

dvoustupňovým detektorům. Dvoustupňové detektory mají výhodu v kvalitě detekce, jednostupňové detektory mají obecně problém s detekcí malých objekt. Mezi jednostupňové detektory patří YOLO[43], SSD[44] nebo RetinaNet[45].

3.5.1 R-CNN

V roce 2014 se objevila první síť na detekci nazvaná R-CNN (Regions with CNN)[46]. Za pomocí algoritmu nazvaného selective search se vytvoří přibližně 2000 regionů (částí obrazu). Tyto regiony slouží jako kandidáti na objekt. Následně jsou přeškálovány na fixní velikost a použity jako vstup pro konvoluční vrstvy sítě. Jejich výstup je vstupem pro SVMs (Support Vector Machines), které zjišťují zda se v daném regionu nachází nějaký objekt a pokud ano, tak do jaké třídy patří. Tento postup zpracování obrazu zobrazuje obrázek 7[46].



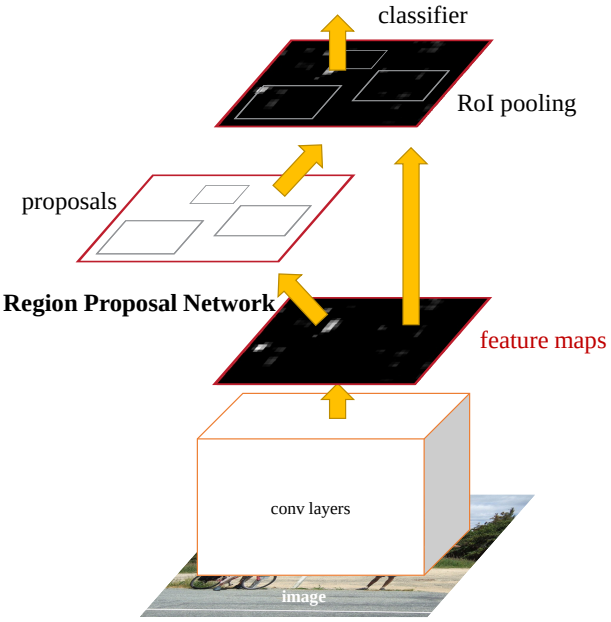
Obrázek 7: Zpracování obrazu pomocí R-CNN

3.5.2 Fast R-CNN

Jak napovídá název Fast R-CNN navazuje na R-CNN a největší změna nastala v rychlosti. Zrychlení spočívá v tom, že Fast R-CNN sdílí výpočet příznaků přes všechny kandidáty. V R-CNN se totiž tyto příznaky počítají pro jednotlivé kandidáty zvlášť. Navíc Fast R-CNN představila vrstvu zvanou Region of Interest pooling (RoI) vrstvu, která má na vstupu výstup z konvoluční části sítě a jako výstup generuje příznakový vektor fixní délky. Posledním přínosem je použití jednotné ztrátové funkce, která společně optimalizuje klasifikaci a lokalizaci bounding boxů[47].

3.5.3 Faster R-CNN

Faster R-CNN[48] navazuje na Fast R-CNN a opět přináší zrychlení. Architektura Faster R-CNN je podobná jako u Fast R-CNN. Rozdílem je, že místo externího generování kandidátů je Faster R-CNN generuje pomocí konvoluční sítě zvané Region Proposal Network (RPN), která je připojená za konvoluční vrstvy. Sdílení konvolučních výpočtů přes všechny kandidáty sice zrychlilo proces zpracování obrazu, ale nevyřešilo to problém, se kterým se potýkaly předchozí modely. Ten nastává při generování kandidátů na objekt. Tyto sítě používaly buď algoritmus zvaný selective search, který navrhne velké množství kandidátů, takže sice objekt je s velkou pravděpodobností v jednom z kandidátů, ale jeho časová náročnost je vysoká. Alternativou je algoritmus EdgeBoxes, který je kompromisem mezi rychlostí a přesností. Další nevýhodu těchto algoritmů je, že nejdou trénovat společně s neuronovou sítí na konkrétní úlohu.



Obrázek 8: Schéma Faster R-CNN

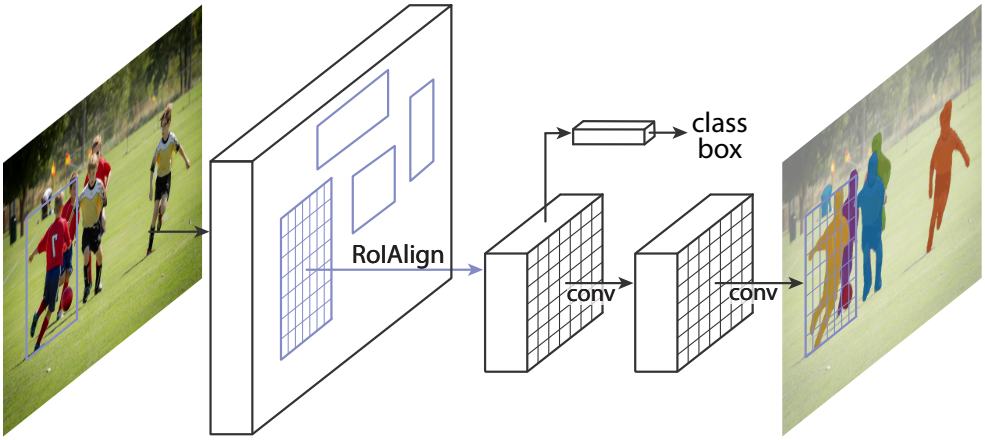
Faster R-CNN vyřešila tento problém právě pomocí RPN. Což je malá konvoluční síť, která je přidaná za konvoluční vrstvy sítě. Jako vstup má tedy příznakovou mapu, vygenerovanou konvolučními vrstvami. Jako výstup generuje kandidáty na objekty. Každý kandidát má takzvané skóre předmětnosti (objectness score), které pomocí IoU popisuje zda se v kandidátovi nachází objekt, nebo je kandidát pouze pozadí. Tato architektura řeší problém zdlouhavého generování kandidátů, protože to provádí konvoluční síť, je tedy

možné ji trénovat se zbytkem sítě. A vzhledem k tomu, že RPN pracuje s příznakovými mapami, tak je výrazně rychlejší. Výstup z RPN a konvolučních vrstev se poté zkombinuje a vloží do RoI pooling vrstvy, která je stejná jako v Fast R-CNN. RPN přináší do sítě mechanismus pozornosti, protože kandidáti z RPN říkají zbytku sítě, kam se má koukat. Na obrázku 8[48] je znázorněno jak síť nejdříve vytvoří příznakové mapy, poté vygeneruje kandidáty a po průchodu ROI pooling vrstvou se kandidáti klasifikují.

Při trénování Faster R-CNN se střídá mezi trénováním RPN a detektoru objektů (Fast R-CNN) při zafixovaných kandidátech. Celková ztrátová funkce je kombinací mezi ztrátovými funkcemi RPN a detektoru. Obě ztrátové funkce se ještě dělí na klasifikační a regresní ztrátu (lokalizace objektů). Toto schéma produkuje sjednocenou síť, která sdílí konvoluční příznaky mezi generováním kandidátu a detekcí.

3.5.4 Mask R-CNN

Mask R-CNN je rozšířením Faster R-CNN. Architektura Faster R-CNN se ponechala, pouze přibyla konvoluční část, která generuje segmentační masku. Síť má tedy tři výstupy bounding box na hrubý odhad polohy objektu, segmentační masku a označení do které třídy objekt patří, respektive pravděpodobnost s jakou si je síť jistá, že model do dané třídy patří.



Obrázek 9: Schéma Mask R-CNN

V Mask R-CNN se musí implementovat algoritmus zvaný RoIAlign. Ten zajišťuje, že konvoluční příznaky získané na výstup RoI pooling vrstvy, se zarovnají s původním obrazem. Ztrátová funkce pro tuto síť je opět kombinací ztrátových funkcí jednotlivých částí modelu.

$$L = L_{cls} + L_{box} + L_{mask} \tag{12}$$

Kde L_{cls} značí klasifikační ztrátu, L_{box} značí ztrátu způsobenou nepřesností v souřadnicích bounding boxu a L_{mask} je chyba vygenerované masky[49].

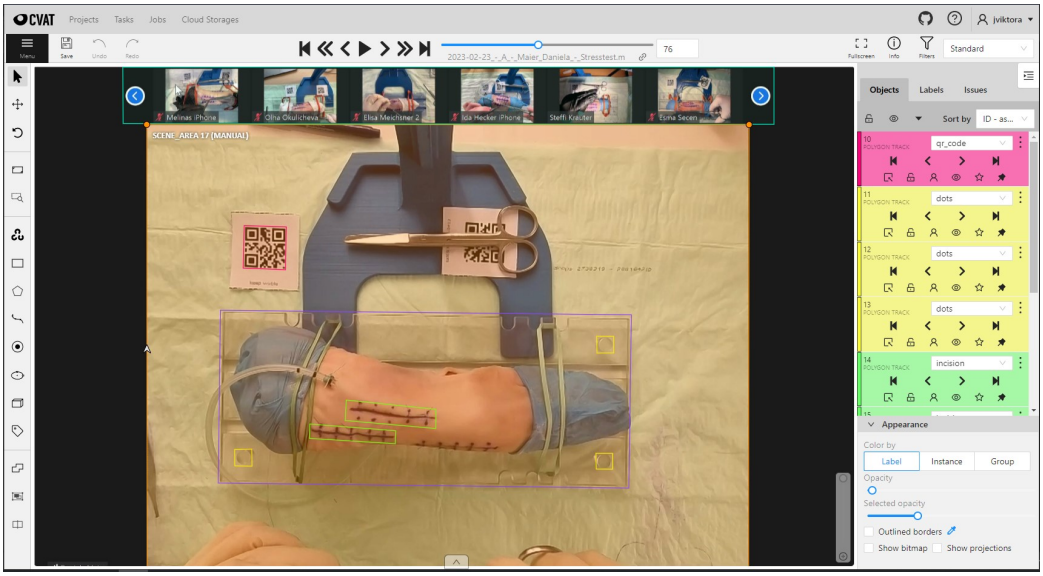
Mask R-CNN a Faster R-CNN jsou neuronové sítě použité v této práci.

4 Návrh datasetu

Pro každou úlohu byl vytvořen samostatný dataset. Jedním z důvodů je to, že v každé úloze se model snaží najít jiné objekty. Navíc se pro výpočet vzdálenosti na základě polohy objektů používá síť Mask R-CNN, která jako vstup vyžaduje masku. Oproti tomu sít Faster R-CNN používá jako vstup bounding box. Všechny datasety byly vytvořené z videí studentů Friedrich-Schillerovy univerzity v Jeně. K anotaci byl použit nástroj CVAT.

4.1 CVAT

CVAT je interaktivní nástroj pro anotování obrazových dat. Původně vyvinut v roce 2019 společností Intel. Umožňuje anotaci pomocí čtyř typů objektů: bounding box, polygon, polyline a point. Podporuje několik typů automatizací jako automatické značení, sledování atd.



Obrázek 10: Uživatelské rozhraní CVATu

Práce ve CVATu se rozděluje na projekty, ve kterých se vytváří jednotlivé úkoly. Projekty a úkoly se dají přiřadit jednotlivým uživatelům. Každý projekt má své třídy, které se dají anotovat. Jednotlivé úlohy lze mezi projekty přenášet tím, že se třídy přenášené úlohy namapují na třídy cílového projektu. CVAT podporuje téměř všechny používané formáty datasetů (COCO, KITTI, PASCAL VOC, YOLO atd.) pro segmentaci, detekci a jiné úlohy spojené se zpracováním obrazu[50].

4.2 COCO

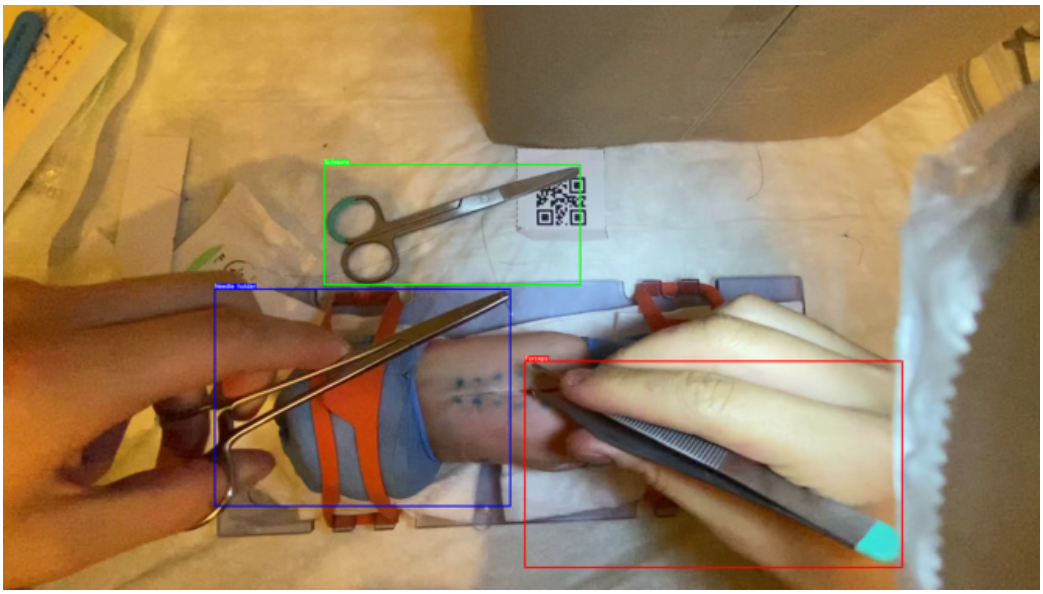
Pojem COCO může mít v oblasti zpracování obrazu dva významy. Prvním je MS COCO[51], což je dataset vyvinutý společností Microsoft v roce 2015. V datasetu se nachází přes 300 tisíc obrázku, na kterých je dohromady 2,5 miliónů objektů. Ty jsou rozděleny do 91 kategorií. Tento dataset navazuje na ImageNet[7], PASCAL VOC[52] a SUN[53] datasety. Každý z nich se soustředí na mírně odlišnou část práce s obrazem, ale všechny spojuje jejich velikost. PASCAL VOC je navíc součástí PASCAL VOC Challenge. Tato výzva zahrnovala i každoroční soutěž.

Pojem COCO lze také chápat jako formát datasetu. Ten určuje, jak se informace o anotacích uchovává. Způsobem jakým jsou objekty v datasetu popsány se liší dataset od datasetu, ale ve většině případů se údaje uchovávají v jednom souboru. Ty se odlišují typem souboru (XML, textový soubor aj.) a strukturou jakou jsou informace v souboru zaznamenány. COCO formát uchovává informace v JSON souboru s následující strukturou. První dva klíče jsou licence (licenses) a informace (info), ve kterých jsou obecné informace o souboru. V klíči obrazy (images) se uchovávají nejdůležitější informace jako výška, šířka a jméno obrazu. První tří klíče jsou v každém anotačním COCO souboru. Další klíče jsou závislé na konkrétní úloze. Pro detekci objektů jsou v souboru další dva klíče. Prvním klíčem je kategorie (categories), zde jsou do pole uspořádány všechny kategorie, které se v souboru nachází. Druhý klíč je anotace (annotations), v němž jsou opět do pole poskládány informace o jednotlivých anotacích. Obsahem je ke kterému obrazu tato anotace patří, do jaké třídy tato anotace patří a její poloha. Způsobem jakým je uchována informace o poloze, záleží na typu anotace.

Obrazy, které se používají k trénování, validování a testování modelu se ukládají do samostatného adresáře. Jména v tomto adresáři si musí odpovídat se jmény v JSON souboru. Pokud je zapotřebí rozdělit dataset na trénovací, validační a testovací část, tak se toto rozdělení musí udělat buď manuálně, anebo před vytvořením anotačního souboru.

4.3 Dataset pro detekci nástrojů

Tento dataset byl vytvořen z osmi videí a rozdělen v poměru 75% trénovací, 12.5% validační a 12.5% testovací množiny. V datasetu se nachází tři třídy, kde každá z nich reprezentuje jednotlivý nástroj (jehelec, pinzeta a nůžky). Při anotaci byl kladen důraz na ohraničení viditelné části nástroje. Člověk intuitivně odhadne, kde se nachází skrytá část nástroje. Neuronová síť použitá na tuto úlohu toho schopná není a pokud by se při trénování používaly obrazy, kde je anotovaná i ta část nástroje, která je skrytá, mohlo by se stát, že neuronová síť bude mít v testovací fázi nežádoucí chování. Například detekce rukou ve, kterých žádný nástroj není, nebo detekce výřezu, ve kterém se žádný nástroj nenachází. Důležité bylo také anotovat všechny viditelné nástroje, aby se při trénování nestalo, že penalizujeme neuronovou síť za správnou detekci nástroje.



Obrázek 11: Ukázka trénovacího obrazu

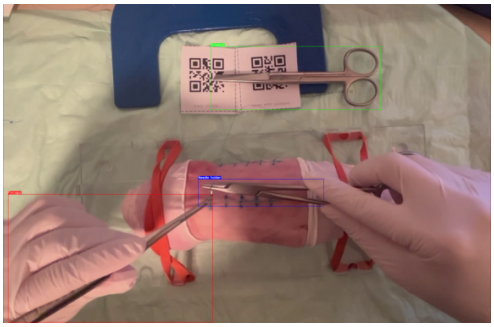
4.3.1 Augmentace datasetu

Augmentace byla zvolena z více důvodů, prvním z nich je nedostatek dat. Před použitím augmentace se v datasetu nacházelo pouze 578 obrazů. Druhým důvodem pro využití augmentace je obohacení dat. Většina lidí má dominantní pravou ruku, proto drží pinzetu v levé ruce a jehelec v pravé. V tomto případě hrozí, že se neuronová síť naučí nežádoucí chování, jako například to, že pinzeta je vždy na levé straně obrazu.

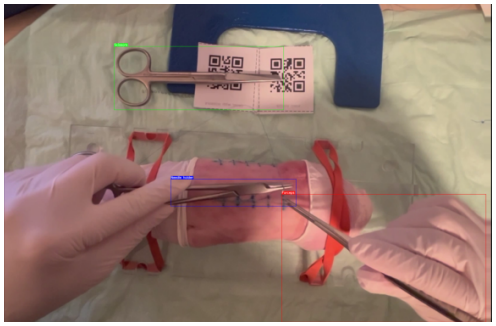
Augmentace dat je způsob, jakým lze uměle zvětšit trénovací množinu. V obraze se většinou jedná o různé typy transformací. Úkolem obecně bývá zabránit přetrénování modelu. V tomto případě se jedná o problém nerovnosti v počtu praváků a leváků. Tím, že v datasetu bude velké množství jedinců, kteří budou držet pinzetu v levé ruce, tak existuje možnost, že se neuronová síť naučí, že nástroj pohybující se v levé části obrazovky je vždy pinzeta. V této práci byly použity tři druhy augmentace.

HorizontalFlip

Je zrcadlové otočení obrazu, tedy otočení obrazu kolem svislé osy vedoucí středem obrazu. Tento typ transformace se použil na všechny původní obrazy. Trénovací množina se tedy zvětšila o 578 obrazů. A je řešením problému nerovnosti počtu praváků a leváků.



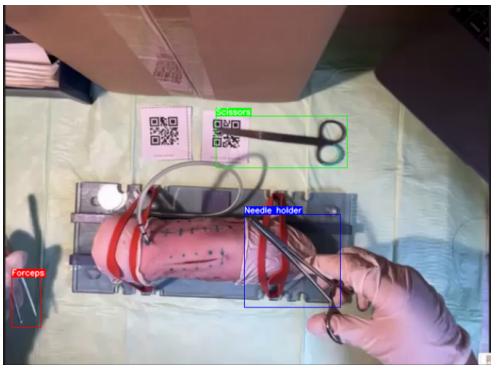
Obrázek 12: Původní obraz



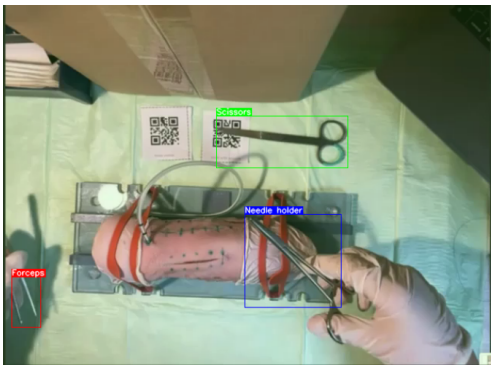
Obrázek 13: Aplikace HorizontalFlip

RGBShift

Náhodně posune hodnotu každého barevného kanálu (s nastaveným limitem). Studenti obecně nemají jednotné vybavení. Nástroje se liší pouze v detailech, ale například barva podložky, která zabírá nemalou část obrazu, je různá. Proto by tento typ augmentace měl zmírnit pravděpodobnost přetrénování a zároveň zvětšit trénovací množinu.



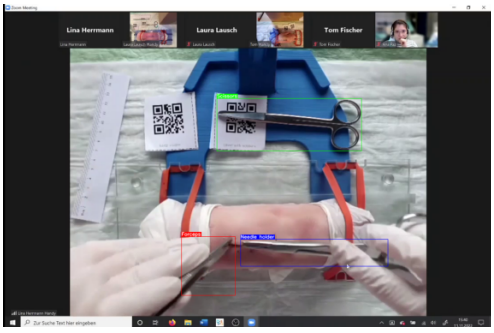
Obrázek 14: Původní obraz



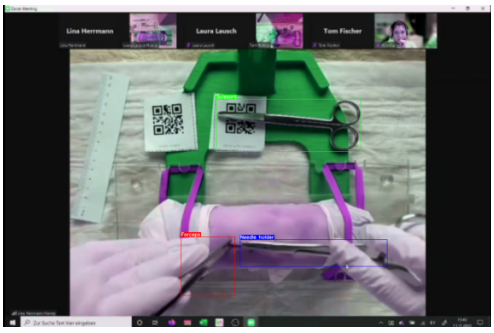
Obrázek 15: Aplikace RGBShift

Downscale, HueSaturationValue

První z nich sníží a poté zvýší kvalitu obrazu, druhá změní hodnotu odstínu a sytosti. Tyto transformace byly použity dohromady a to z důvodu různé kvality videí v trénovací množině. Podobně jako v předchozím případě podmínky každého jedince jsou rozdílné. Například vybavení pro natočení úkonu, osvětlení scény a další. Tato augmentace se tedy snaží obohatit trénovací množinu a videa s rozdílným prostředím.



Obrázek 16: Původní obraz



Obrázek 17: Obraz po augmentaci

	Před augmentací			Po augmentaci		
Množina	Trénovací	Validační	Testovací	Trénovací	Validační	Testovací
Počet původních obrazů	429	74	75	429	74	75
Horizontal Flip	0	0	0	429	74	75
RGBShift	0	0	0	429	74	75
Downscale, HSV	0	0	0	429	74	75
Celkový počet	429	74	75	1716	296	300

Tabulka 1: Počet dat v jednotlivých množinách datasetu

Po augmentaci vznikl nový dataset. Opět bylo zapotřebí jej rozdělit na trénovací, validační a testovací množinu. V trénovací množině je přibližně 75% obrazů z datasetu, 12.5% je ve validační i testovací množině. Toto rozdělení je znázorněno v tabulce 1.

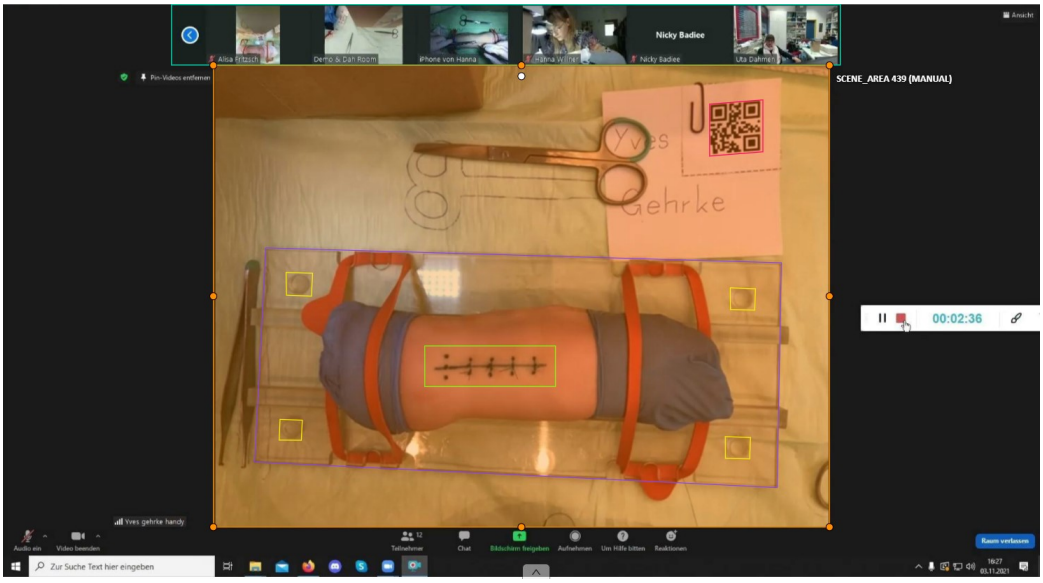
4.4 Dataset pro detekci vzdálenosti

Tento dataset byl vytvořen kombinací devíti videí. Poslední video je složeno přibližně z dvaceti videí, kde z každého videa je vybráno pouze několik úvodních záběrů. Snahou bylo do datasetu zahrnout co nejvíce tříd, které by mohla neuronová síť spolehlivě detekovat. Ve finální verzi se v datasetu nachází celkem šest kategorií:

1. řez - oblast řezu
2. operační oblast - část obrazu, ve které probíhá úkon
3. podstavec - podstava, na kterou je gumami přichycena prasečí noha
4. QR kód
5. UI videohovoru - uživatelské rozhraní ZOOM
6. výběžky - oblasti na podstavci

Tento dataset byl vytvořen pro předzpracování videa. Myšlenkou bylo, že neuronová síť natrénovaná na tomto datasetu zpracuje pouze první záběr videa. Pokud se povede detekovat operační oblast, tak se při dalším zpracování bude používat pouze tato část videa. Pokud se detekce operační oblasti nepodaří, ale povede se detekovat UI videohovoru, tak se tato část videa odstraní, protože by mohla rušit neuronovou síť, která má za úkol hledat nástroje. Pro tuto detekci jsou dva hlavní důvody. Prvním je snížení času zpracování celého videa, protože video po předzpracování má menší velikost. A druhým důvodem je, že ostatní studenti ve videohovoru mohou také provádět úkon a v tom případě by mohlo nastat, že model, co má na starosti detekci nástrojů je bude nacházet právě v UI videohovoru.

Při úspěšné detekci ostatních tříd se použijí pro výpočet reálné velikosti jednoho pixelu (velikost pixelu v centimetrech). Nález řezu a podstavce navíc umožňuje vyhodnocení pohybu nástrojů, protože v ideálním případě se nástroje vzdalují od těchto částí co nejméně.



Obrázek 18: Ukázka z datasetu pro výpočet vzdálenosti

	Řez	Operační oblast	Podstavec	QR kód	UI videohovoru	Výběžky
Trénovací	504	975	232	1566	270	1802
Testovací	264	335	61	295	167	620
Celkem	768	1310	293	1861	437	2422

Tabulka 2: Četnost jednotlivých tříd v datasetu

5 Výsledky

Pro vyhodnocení kvality natrénování neuronové sítě se používala testovací množina. Validační množina slouží ke kontrole zda trénování probíhá správně. V MMDetection lze nastavit parametr *evaluation_interval*, ten ovlivňuje jak často se bude trénování validovat.

Metriky použité pro vyhodnocení experimentů jsou detailněji popsané v kapitole 3.3. Po natrénování sítě pomocí funkce *train_detector()* MMDetection spočte několik metrik (AP - Average precision, AR - Average Recall, mAP - mean AP a mAR - mean AR) na validační a testovací množině (na validační množině jsou tyto hodnoty počítány i v průběhu trénování). Pro AR byly použity pouze metriky AR_m a AR_l, protože pro výpočet AR_s nebylo v datasetu dostatek hodnot, které by patřili do kategorie malé (velikost objektu $< 32^2$ pixelů). Při výpočtu těchto metrik je zapotřebí IoU práh, v MMDetection se používají hodnoty v intervalu $\langle 0,5; 0,95 \rangle$. Důvodem proč

se používá IoU práh až od 0.5 je ten, že hodnotu IoU pod 0.5 se považuje jako nepřesná lokalizace objektu.

Pro vyhodnocení IoU byla použitá vlastní implementace IoU distribuce. Která pro každý testovací obraz vezme anotované bounding boxy a výsledky detekce, spočte jejich IoU a nakonec vykreslí rozložení pro celou testovací množinu. Vodorovná osa je rozdělena do deseti intervalů ($\langle 0; 0,1)$, $\langle 0.1; 0,2)$ atd.), y hodnota značí procentuální počet v daném intervalu.

5.1 Detekce nástrojů

Pro tuto úlohu byla použita neuronová síť s názvem Faster R-CNN. Podrobný popis jejího fungování je v kapitole 3.5.3. Vybraná byla proto, že je rychlejší než její předchůdci a zároveň má lepší detekční schopnosti.

5.1.1 Trénování bez použití augmentovaných dat

Augmentace by teoreticky měla zlepšit kvalitu natrénování sítě. V praxi je třeba natrénovat síť nejdříve na originálních datech a poté na augmentovaném datasetu a výsledky porovnat. Trénování proběhlo s následujícími parametry:

- learning rate: 1.75e−3
- evaluation metric: bbox_loss
- learning rate warmup: None
- number of epochs: 30
- evaluation interval: 5

Ke konci trénování loss (ztráta) konvergovala k 0.08, nakonec skončila na hodnotě 0.0869 a accuracy (přesnost) konvergovala k hodnotě 99 a skončila na 99.0723. Tato ztrátová funkce hodnotí nastavení celého modelu jako celku. Existují ale detailnější ztrátové funkce, které hodnotí pouze část neuronové sítě. Ty závisí na komponentách, ze kterých se síť skládá. Pro Faster R-CNN mezi ně patří: loss_rpn_cls, loss_rpn_bbox, loss_cls, loss_bbox.

Ztrátové funkce, které mají v názvu „rpn“, se vztahují pouze na RPN část neuronové sítě. Funkce **loss_rpn_bbox** popisuje chybu, kterou udělala RPN část při generování souřadnic kandidátů, **loss_rpn_cls** popisuje chybu klasifikace kandidátů do tříd. Zbylé dvě funkce počítají ztrátu pro část odpovídající Fast R-CNN, **loss_cls** je ztráta klasifikace objektu a **loss_bbox** je chyba odhadu souřadnic daného objektu[48].

Rozdíl mezi těmito dvěma typy ztrátových funkcí je v tom, že funkce pro Fast R-CNN se zabývají už finálním odhadem sítě. Zatímco „rpn“ chyby se počítají pro velké množství kandidátů, které slouží pouze jako návrhy, kde by objekty mohly být.

	loss_rpn_cls	loss_rpn_bbox	loss_cls	loss_bbox	acc	loss
5. epocha	0.0014	0.0131	0.0353	0.0776	98.6621	0.1274
10. epocha	0.0005	0.0081	**0.0222**	**0.0482**	**99.1895**	**0.0789**
15. epocha	0.0009	0.0088	0.0284	0.0575	98.9062	0.0956
20. epocha	**0.0003**	0.0104	0.0244	0.0552	99.0137	0.0904
25. epocha	0.0004	**0.0071**	0.0290	0.0557	98.8574	0.0922
30. epocha	0.0008	**0.0071**	0.0251	0.0539	99.0723	0.0869

Tabulka 3: Průběh ztrát a přesnosti při trénování bez augmentace

Jak bylo zmíněno výše pro vyhodnocení natrénování neuronové sítě použijeme několik metrik, které MMDetection počítá v průběhu trénování. Před trénováním byl parametr *evaluation interval* nastavený na hodnotu 5. To zaručuje, že vždy po proběhnutí pěti epoch se vyhodnotí nastavení sítě. Pro 30 trénovacích epoch to znamená, že síť bude šestkrát ohodnocena během tréninku. Tyto metriky společně s hodnotami ztrát z tabulky 3 slouží ke kontrole průběhu trénování.

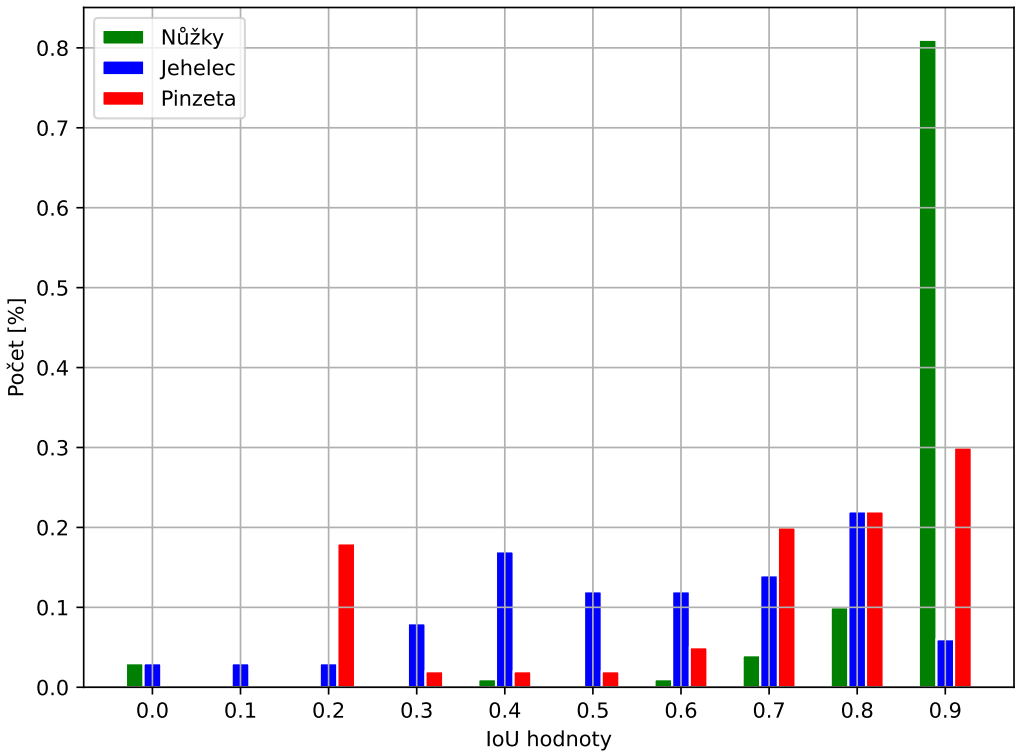
		mAP_{50}	mAP_{75}	mAR_m	mAR_l
Validační množina	5. epocha	**0.9612**	0.6180	0.4594	0.6319
	10. epocha	0.9579	0.6682	**0.4895**	**0.6758**
	15. epocha	0.9542	0.6896	0.4884	0.6750
	20. epocha	0.9534	0.6957	0.4863	0.6675
	25. epocha	0.9527	0.6904	0.4874	0.6687
	30. epocha	0.9527	**0.7003**	0.4880	0.6662
Testovací množina	Po trénování	0.7396	0.4994	0.4632	0.6007

Tabulka 4: Průběh metrik při trénování bez augmentace

Posledním typem ohodnocení natrénování sítě je graf IoU rozložení. Pro každý objekt v testovací množině se spočte IoU s příslušícím detekovaným objektem. Tyto hodnoty se rozdělí do jednoho z deseti intervalů ($\langle 0; 0,1), \langle 0.1; 0,2)$ atd.). V každém intervalu se sečtou hodnoty pro každý typ objektu a výsledek se převede na procentuální podíl z celé testovací množiny. Tyto hodnoty slouží jako výška sloupců v grafu na vodorovné ose jsou vynesena minima intervalů.

Min. intervalu	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Nůžky	0.03	0.00	0.00	0.00	0.01	0.00	0.01	0.04	0.10	0.81
Jehelec	0.03	0.03	0.03	0.08	0.17	0.12	0.12	0.14	0.22	0.06
Pinzeta	0.00	0.00	0.18	0.02	0.02	0.02	0.05	0.20	0.22	0.30

Tabulka 5: IoU distribuce



Obrázek 19: IoU distribuce

Jako postačující detekce se ve většině případů udává hodnota IoU s ground truth (anotovaným objektem), která má hodnotu > 0.5. Nůžky byly detekovány úspěšně v 96% případů, jehelec v 66% a pinzeta v 78% případů. Celkem tedy síť detekovala úspěšně 80% objektů.

5.1.2 Použití augmentovaných dat

Jak bylo zmíněno výše, pro zjištění zda augmentace byla pro experiment prospěšná, je zapotřebí nejdříve natrénovat síť bez augmentovaných dat a poté s nimi. Aby dával experiment smysl, tak byla použitá stejná neuronová síť, stejné parametry pro trénování a stejný poměr pro rozdělení dat na trénovací, validační a testovací množinu.

	loss_rpn_cls	loss_rpn_bbox	loss_cls	loss_bbox	acc	loss
5. epocha	0.0010	0.0079	0.0296	0.0508	98.8574	0.0893
10. epocha	0.0016	0.0062	0.0189	0.0453	99.1602	0.0719
15. epocha	0.0004	0.0045	0.0168	0.0343	99.3359	0.0560
20. epocha	0.0004	**0.0043**	**0.0142**	0.0393	**99.4141**	0.0581
25. epocha	0.0004	0.0058	0.0147	**0.0340**	99.4043	**0.0549**
30. epocha	**0.0002**	0.0065	0.0166	0.0364	99.3652	0.0596

Tabulka 6: Průběh chyb a přesnosti při trénování

Použité ztrátové funkce jsou stejné jako při vyhodnocení natrénování augmentovaných dat v tabulce 3. Je zapotřebí si uvědomit, že tyto hodnoty jsou ovlivněné počtem obrazů použitých při trénování. Protože při použití augmentovaných dat se zvětší počet obrazů, které se při každé epoše použije k trénování. Zvětšení trénovací množiny ale bylo jedním z cílů augmentace.

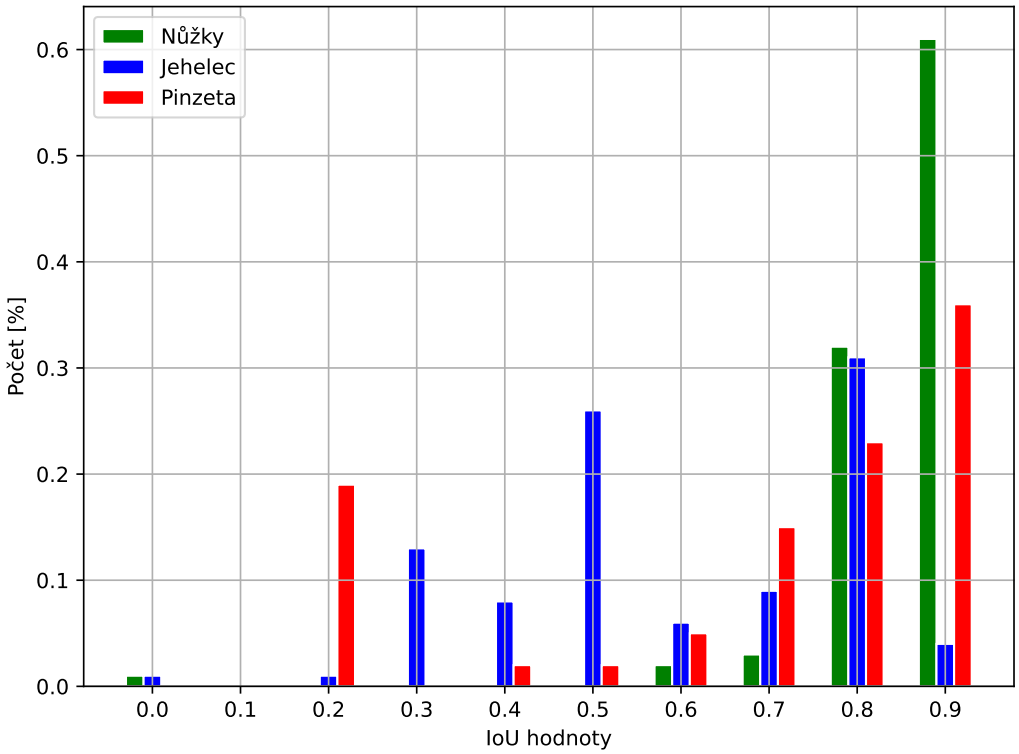
		mAP_{50}	mAP_{75}	mAR_m	mAR_l
Validační množina	5. epocha	**0.9362**	**0.7003**	0.5102	0.7109
	10. epocha	0.9332	0.6734	0.5097	0.7088
	15. epocha	0.9344	0.6755	0.5104	0.7116
	20. epocha	0.9334	0.6751	0.5099	**0.7117**
	25. epocha	0.9331	0.6761	**0.5110**	0.7082
	30. epocha	0.9342	0.6792	0.5106	0.7093
Testovací množina	Po trénování	0.7085	0.5083	0.4704	0.6636

Tabulka 7: Průběh metrik při trénování

Porovnání výsledků natrénování sítí na datasetu bez a s augmentovanými daty jsou popsány v kapitole 5.3. Zvýrazněné hodnoty v tabulkách 3, 4, 6 a 7 ukazují nejlepší hodnotu dané ztrátové funkce nebo metriky.

Min. intervalu	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Nůžky	0.01	0.00	0.00	0.00	0.00	0.01	0.02	0.03	0.33	0.60
Jehelec	0.01	0.00	0.01	0.12	0.08	0.26	0.07	0.09	0.3	0.06
Pinzeta	0.00	0.00	0.19	0.00	0.02	0.02	0.05	0.15	0.22	0.36

Tabulka 8: IoU distribuce



Obrázek 20: IoU distribuce

Stejně jako v experimentu bez augmentovaných dat i zde se považuje jako postačují hodnota IoU > 0.5. Nůžky byly detekovány úspěšně v 99% případů, jehelec v 78% a pinzeta v 79% případů. Celkem tedy 85.33% detekcí bylo úspěšných.

5.2 Detekce vzdálenosti

Na rozdíl od detekce nástrojů pro tuto úlohu byla použita neuronová síť s názvem Mask R-CNN. Důvodem pro použití této sítě místo Faster R-CNN je ten, že Mask R-CNN je schopna pracovat z takzvanou maskou, zatímco Faster R-CNN pracuje pouze s bounding boxy. Maska na rozdíl od bounding boxu může mít libovolný tvar. Proto se tedy hodí více na úlohy, kde objekt není obdélníkového tvaru. Což je pro tento úkol důležité, protože v mnoha případech nemají objekty v této úloze rovnoběžný tvar. A čím přesnější bude lokalizace těchto objektů, tím přesnější bude odhad vzdálenosti.

Nevýhodou při použití Mask R-CNN je, že se déle trénuje a déle zpracovává obrazy při provozu. Vzhledem k tomu, že by se v tomto projektu měla používat na několik málo snímků, tak ztráta na rychlosti zpracování by

neměla být velká. Při trénování modelu Mask R-CNN se používají stejné ztrátové funkce jako u modelu Faster R-CNN. Navíc je přidána funkce loss_mask, protože Mask R-CNN má tři výstupy - bounding box, confidence threshold a masku - zatímco Faster R-CNN má pouze dva. Funkce loss_mask charakterizuje chybu tvorby masky.

	loss_rpn_cls	loss_rpn_bbox	loss_cls	loss_bbox	loss_mask	acc	loss
5. epocha	0.0159	0.0305	0.1001	1.3660	0.0739	97.3975	1.5865
10. epocha	**0.0112**	**0.0175**	0.0731	0.3740	0.0513	**98.1624**	0.5270
15. epocha	0.0131	0.0185	**0.0718**	**0.2679**	**0.0495**	98.1315	**0.4208**
20. epocha	0.0120	0.0181	0.0805	0.2931	0.0551	97.9720	0.4588
25. epocha	0.0134	0.0189	0.0789	0.3043	0.0527	97.8630	0.4682
30. epocha	0.0135	0.0184	0.0761	0.3185	0.0532	97.9720	0.4797

Tabulka 9: Průběh chyb a přesnosti při trénování

Stejně jako Faster R-CNN má Mask R-CNN na výstupu bounding boxy a informaci do jaké třídy patří. Liší se v tom, že jako další výstup má i segmentační masku. Bounding box ohraničuje oblast, ve které se objekt nachází. Maska má za úkol určit, které pixely z oblasti určené bounding boxem jsou daným objektem.

Každý výstup sítě se hodnotí samostatně. Pokud se zjistí, že se síť nenatrénovala dobře nebo nefunguje vůbec, je snadnější nalézt problém.

		mAP_{50}	mAP_{75}	mAR_m	mAR_l
Validační množina	5. epocha	0.8007	0.6248	0.7021	0.7394
	10. epocha	0.8036	0.5968	0.7355	0.7030
	15. epocha	0.8023	0.6016	0.7053	0.7411
	20. epocha	**0.8108**	0.6235	0.7314	0.7400
	25. epocha	0.7989	0.6235	**0.7716**	0.7191
	30. epocha	0.8001	**0.6603**	0.7308	**0.7486**
Testovací množina	Po trénování	0.7359	0.6534	0.6936	0.6997

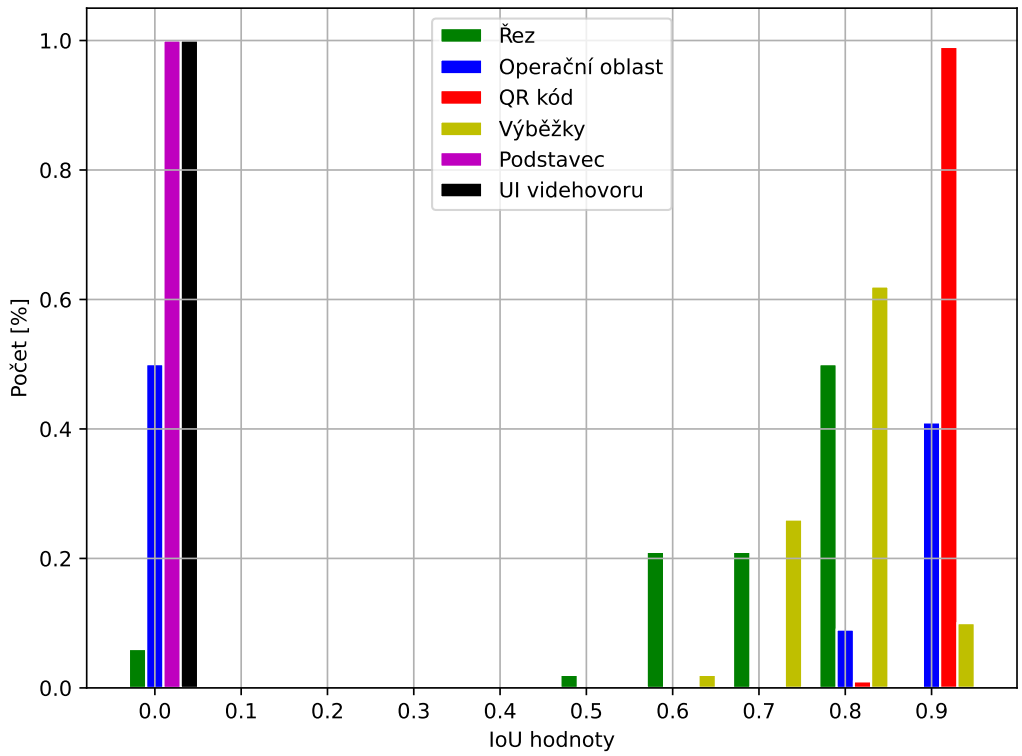
Tabulka 10: Průběh metrik bounding boxu při trénování

Následující tabulka používá stejné metriky jako předchozí. Rozdíl je v tom, že se metriky počítají na masce a ne na bounding boxu jako v předchozích případech.

		mAP_{50}	mAP_{75}	mAR_m	mAR_l
Validační množina	5. epocha	0.7904	**0.7112**	0.7201	0.7660
	10. epocha	**0.8036**	0.6138	0.7139	0.7240
	15. epocha	0.8023	0.6235	0.6757	0.7599
	20. epocha	0.8015	0.6378	0.7068	0.7572
	25. epocha	0.7989	0.6428	0.7462	0.7466
	30. epocha	0.8001	0.6791	**0.7539**	**0.7692**
Testovací množina	Po trénování	0.7420	0.6144	0.7521	0.7600

Tabulka 11: Průběh metrik masky při trénování

Nevýhodou těchto metrik i výše zmíněných ztrátových funkcí je to, že z nich nelze usoudit kvalitu detekce jednotlivých tříd. Počet objektů v jednotlivých třídách tohoto datasetu není vyvážený. Hrozí riziko, že model nebude schopný dobře detekovat objekty, které patří do méně zastoupených tříd. IoU distribuce může ukázat zda model opravdu detekuje některé typy objektů hůře.



Obrázek 21: IoU distribuce pro bounding boxy

Min. intervalu	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Řez	0.06	0.00	0.00	0.00	0.00	0.02	0.21	0.21	0.50	0.00
Operační oblasti	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.09	0.41
QR kód	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.99
Výběžky	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.26	0.62	0.10
Podstavec	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
UI videohovoru	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Tabulka 12: IoU distribuce pro bounding boxy

Pokud se použije stejný IoU práh pro hodnocení správné a špatné detekce jako v předchozí úloze, tak vyjde, že 42.67% detekcí je špatných. Pokud se, ale nezapočítají třídy Podstavec a UI videohovoru, tak vychází, že 86% detekcí je správných.

5.3 Interpretace výsledků

Pro vyhodnocení natrénování neuronových sítí byly použity ztrátové funkce popsané v kapitole 5.1.1, metriky zmíněné v kapitole 3.3 a graf IoU distribuce, který má popisovat schopnost sítě detekovat jednotlivé třídy.

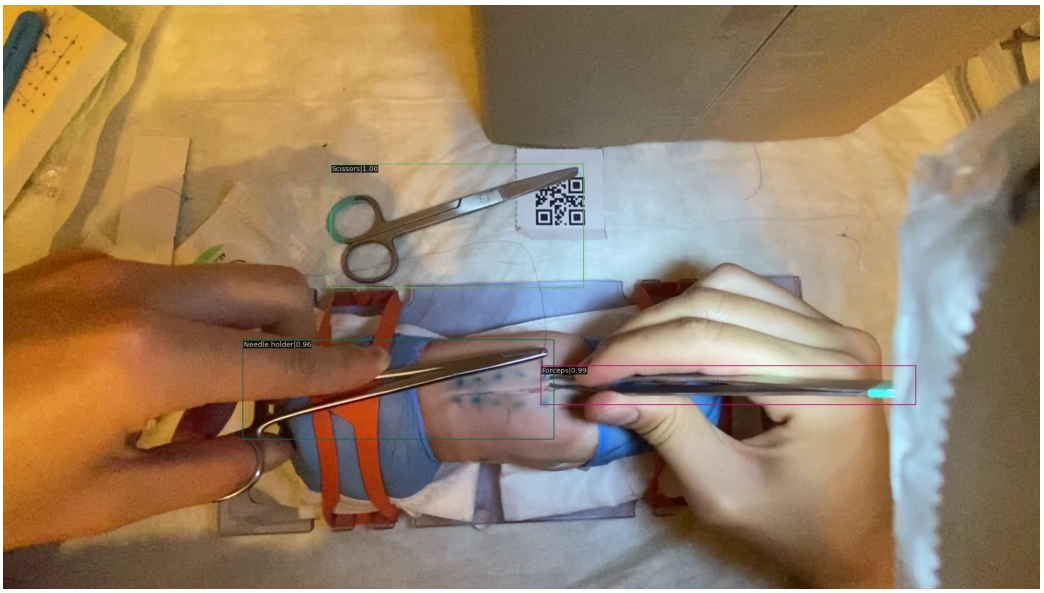
5.3.1 Výsledky pro detekci nástrojů

Pro úlohu detekce nástrojů byly natrénovány dvě neuronové sítě. Prvních z nich byla natrénovaná na originálním datasetu, druhá na augmentovaném. Pro účel porovnání výsledků byla použita stejná neuronová síť - Faster R-CNN a stejné parametry pro trénování. Průběh ztrátových funkcí je znázorněn v tabulkách 3 a 6. Z pohledu ztrátových funkcí se dá pokládat augmentace za přínosnou pro experiment, protože minima všech těchto funkcí jsou v tabulce 6 nižší než v tabulce 3.

Z tabulek 4 a 7 pro průběh metrik je vidět, že model trénovaný na augmentovaném datasetu má vyšší AR, ale nižší AP. Vysoké AR znamená, že model málokdy nedetekuje objekt, i když by měl. Vysoké AP znamená, že model zřídka predikuje objekt, který v obraze není. Což znamená, že augmentovaný model mine málo objektů, ale hrozí, že bude detekovat i objekty které na obrazu nejsou. Na testovací sadě je u prvního modelu vyšší pouze hodnota mAP_{50}, ale výrazný rozdíl je pouze v hodnotě metriky mAR_l. Z pohledu metrik přináší augmentace pouze malé zlepšení.

Grafy IoU distribuce ukazují schopnost modelů detekovat jednotlivé třídy. Detekce nůžek je pro model zdaleka nejjednodušší, pravděpodobně kvůli tomu, že nůžky jsou téměř celou dobu na stejném místě. Model trénovaný na originálním datasetu úspěšně detekoval 80% objektů z trénovací množiny.

Model trénovaný na augmentovaném datasetu měl úspěšnost 85.33%. Tedy i z tohoto pohledu přinesla augmentace užitek. Hodnoty 85.33% i 80% jsou uspokojivé. Model má problém například pokud jsou snímky rozmazané anebo je velká část nástroje zakrytá.

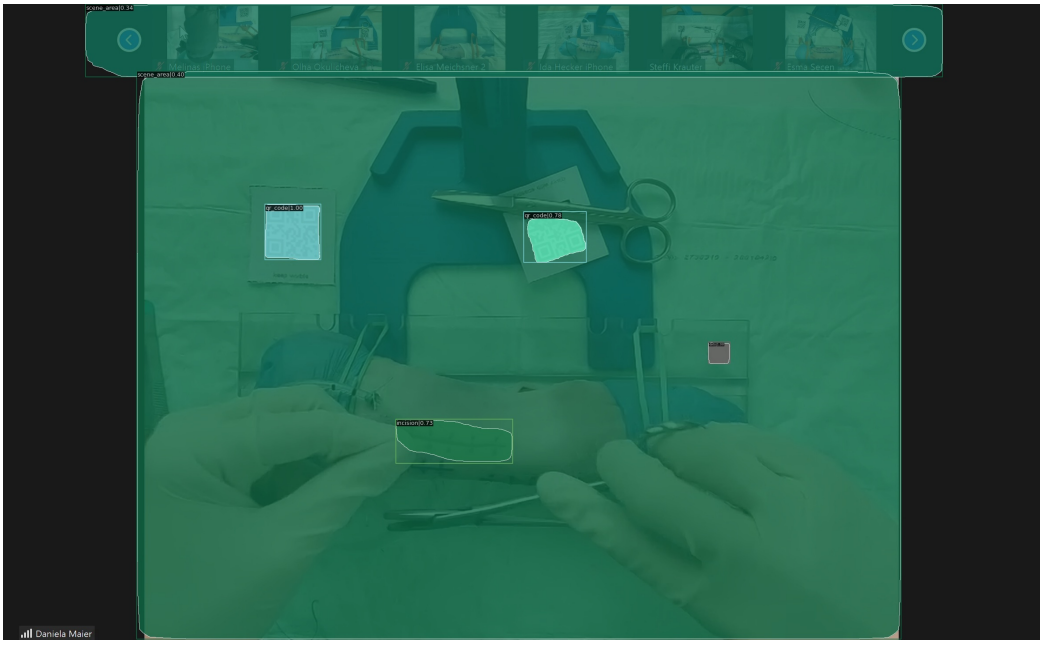


Obrázek 22: Výsledek detekce nástrojů

5.3.2 Výsledky pro detekci objektů

Při porovnaní průběhu ztrátových funkcí u detekce objektů s detekcí nástrojů je patrné, že model pro detekci objektů se nedostane na hodnoty jako se dostaly modely pro detekci nástrojů. Což je ale pochopitelné, protože jde o těžší úlohu. Za prvé Mask R-CNN je složitější síť než Faster R-CNN a objekty, které se v této úloze detekují, bývají často zakryté nebo je na obrazu vidět pouze jejich část. O výsledcích této úlohy dobře vypovídá obrázek 21. Z něho je patrné, že síť nebyla schopná detekovat podstavec a UI videohovoru. Je vidět, že má problém s detekcí operační oblasti, protože ji správně detekuje pouze v 50% případů. Toto číslo je nízké z toho důvodu, že si síť plete UI videohovoru právě s operační oblastí. Zbylé tři třídy je model schopný detekovat téměř bez problémů.

Hodnota IoU přes 0.9 pro QR kód je 99%, QR kód je tedy optimální objekt k výpočtu vzdálenosti na videu.



Obrázek 23: Výsledek detekce objektů

6 Závěr

Cílem této práce bylo vytvořit nastroj pro podporu vyhodnoceni výkonu studenta při nácviku chirurgického šití. Tento nástroj detekuje chirurgické nástroje (jehelec, pinzetu, nůžky) využívané k provedení výkonu a několik objektů, které pomohou s jeho vyhodnocením.

I přestože jsou podmínky za kterých studenti vykonávají šití téměř neměnné, objevují se určité výchylky v datech. Zároveň pro kvalitní natrénování neuronové sítě je zapotřebí velké množství dat. Oba problémy se snaží řešit datová augmentace, která zvětší počet trénovacích dat a zároveň použitím různých transformací odstraní určité výchylky v datech jako je nevyváženost v počtu lidí s dominantní pravou a počtu lidí s dominantní levou rukou.

V detekci nástrojů byla síť natrénovaná na augmentovaném datasetu schopna detekovat správně přes 85% objektů z testovací množiny a dosáhnout hodnoty $\text{mAP}_{50} = 0.9362$. Síť pro detekci nástrojů tedy funguje spolehlivě a je schopná ve videu sledovat nástroje. Neuronová síť pro detekci objektů sice není schopná dobře detekovat všechny třídy, ale objekty jako QR kód nebo řez detekuje s velkou úspěšností. Tyto detekce se budou dále dát využít pro výpočet vzdálenosti nebo pro hodnocení polohy nástrojů (nástroje by se v ideálním případě měli vzdalovat od řezu co nejméně). Budoucí práce leží ve zlepšení detekce některých objektů pro lepší předzpracování videa. Také ve vývinu algoritmu co s detekce například QR kódu bude schopen získat přesnou velikost jednoho pixelu v centimetrech.

Zdroje

[1] Ian Goodfellow, Yoshua Bengio a Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[2] Warren S Mcculloch a Walter Pitts. *A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY*. Tech. zpr. 1943.

[3] Frank Rosenblatt. „rosenblatt-1957“. In: (led. 1957). URL: https://blogs.umass.edu/brain-wars/files/2016/03/rosenblatt-1957.pdf.

[4] Larry Hardesty. *Explained: Neural networks*. MIT News, dub. 2017. URL: https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414 (cit. 07. 05. 2023).

[5] Geoffrey E. Hinton, Simon Osindero a Yee-Whye Teh. „A Fast Learning Algorithm for Deep Belief Nets“. In: *Neural Computation* 18.7 (2006), s. 1527–1554. DOI: 10.1162/neco.2006.18.7.1527.

[6] G. E. Hinton a R. R. Salakhutdinov. „Reducing the dimensionality of data with neural networks“. In: *Science* 313 (5786 čvc. 2006), s. 504–507. ISSN: 00368075. DOI: 10.1126/science.1127647.

[7] Jia Deng et al. „ImageNet: A large-scale hierarchical image database“. In: Institute of Electrical a Electronics Engineers (IEEE), břez. 2010, s. 248–255. DOI: 10.1109/cvpr.2009.5206848.

[8] Alex Krizhevsky, Ilya Sutskever a Geoffrey E Hinton. *ImageNet Classification with Deep Convolutional Neural Networks*. Tech. zpr. URL: http://code.google.com/p/cuda-convnet/.

[9] Kaiming He et al. „Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification“. In: (ún. 2015). URL: http://arxiv.org/abs/1502.01852.

[10] Xavier Glorot a Yoshua Bengio. „Understanding the difficulty of training deep feedforward neural networks“. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Ed. Yee Whye Teh a Mike Titterington. Sv. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, květ. 2010, s. 249–256. URL: https://proceedings.mlr.press/v9/glorot10a.html.

[11] Afshine Amidi a Shervine Amidi. *CS 230 - Recurrent Neural Networks Cheatsheet*. Stanford.edu, 2019. URL: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks (cit. 07. 05. 2023).

[12] *CS231n Convolutional Neural Networks for Visual Recognition*. Stanford.edu, 2012. URL: https://cs231n.github.io/convolutional-networks/ (cit. 14. 05. 2023).

[13] Ian J. Goodfellow et al. „Generative Adversarial Networks“. In: (čvn. 2014). URL: http://arxiv.org/abs/1406.2661.

[14] Antonia Creswell et al. „Generative Adversarial Networks: An Overview“. In: *IEEE Signal Processing Magazine* 35 (1 led. 2018), s. 53–65. ISSN: 10535888. DOI: 10.1109/MSP.2017.2765202.

[15] Ashish Vaswani et al. „Attention Is All You Need“. In: (čvn. 2017). URL: http://arxiv.org/abs/1706.03762.

[16] Surafel M Lakew et al. *A Comparison of Transformer and Recurrent Neural Networks on Multilingual Neural Machine Translation*. Tech. zpr., s. 641–652.

[17] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[18] Mark A Kramer. *Nonlinear Principal Component Analysis Using Autoassociative Neural Networks*. Tech. zpr.

[19] Sepp Hochreiter a Jürgen Schmidhuber. „Long Short-Term Memory“. In: *Neural Computation* 9.8 (lis. 1997), s. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. eprint: https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[20] Charles R. Harris et al. „Array programming with NumPy“. In: *Nature* 585 (2020), s. 357–362. DOI: 10.1038/s41586-020-2649-2.

[21] J. D. Hunter. „Matplotlib: A 2D graphics environment“. In: *Computing in Science & Engineering* 9.3 (2007), s. 90–95. DOI: 10.1109/MCSE.2007.55.

[22] Alexander Buslaev et al. „Albumentations: Fast and Flexible Image Augmentations“. In: *Information* 11.2 (2020). ISSN: 2078-2489. DOI: 10.3390/info11020125. URL: https://www.mdpi.com/2078-2489/11/2/125.

[23] Python. *Welcome to Python.org*. Python.org, květ. 2019. URL: https://www.python.org/ (cit. 27. 04. 2023).

[24] Martín Abadi et al. „TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems“. In: (břez. 2016). URL: http://arxiv.org/abs/1603.04467.

[25] Yangqing Jia et al. „Caffe: Convolutional Architecture for Fast Feature Embedding“. In: (čvn. 2014). URL: http://arxiv.org/abs/1408.5093.

[26] *Glossary — Python 3.9.5 documentation*. docs.python.org, dub. 2023. URL: https://docs.python.org/3/glossary.html%5C#term-global-interpreter-lock (cit. 26. 04. 2023).

[27] *GlobalInterpreterLock - Python Wiki*. wiki.python.org, pros. 2020. URL: https://wiki.python.org/moin/GlobalInterpreterLock (cit. 26. 04. 2023).

[28] Adam Paszke et al. „PyTorch: An Imperative Style, High-Performance Deep Learning Library“. In: (pros. 2019). URL: http://arxiv.org/abs/1912.01703.

[29] Yuxin Wu et al. *Detectron2*. https://github.com/facebookresearch/detectron2. 2019.

[30] Yuntao Chen et al. „SimpleDet: A Simple and Versatile Distributed Framework for Object Detection and Instance Recognition“. In: (břez. 2019). URL: http://arxiv.org/abs/1903.05831.

[31] Kai Chen et al. „MMDetection: Open MMLab Detection Toolbox and Benchmark“. In: (čvn. 2019). URL: http://arxiv.org/abs/1906.07155.

[32] Ravpreet Kaur a Sarbjeet Singh. „A comprehensive review of object detection with deep learning“. In: *Digital Signal Processing: A Review Journal* 132 (pros. 2022). ISSN: 10512004. DOI: 10.1016/j.dsp.2022.103812.

[33] Rafael Padilla et al. „A comparative analysis of object detection metrics with a companion open-source toolkit“. In: *Electronics (Switzerland)* 10 (3 ún. 2021), s. 1–28. ISSN: 20799292. DOI: 10.3390/electronics10030279.

[34] Microsoft. *COCO - metrics*. Srp. 2021. URL: https://cocodataset.org/%5C#detection-eval.

[35] Jan Hosang et al. „What makes for effective detection proposals?“ In: (ún. 2015). DOI: 10.1109/TPAMI.2015.2465908. URL: http://arxiv.org/abs/1502.05082%20http://dx.doi.org/10.1109/TPAMI.2015.2465908.

[36] P. Viola a M. Jones. „Rapid object detection using a boosted cascade of simple features“. In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. Sv. 1. Pros. 2001, s. I–I. DOI: 10.1109/CVPR.2001.990517.

[37] N. Dalal a B. Triggs. „Histograms of oriented gradients for human detection“. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Sv. 1. Čvn. 2005, 886–893 vol. 1. DOI: 10.1109/CVPR.2005.177.

[38] Pedro Felzenszwalb, David McAllester a Deva Ramanan. „A discriminatively trained, multiscale, deformable part model“. In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. Čvn. 2008, s. 1–8. DOI: 10.1109/CVPR.2008.4587597.

[39] Zhengxia Zou et al. „Object Detection in 20 Years: A Survey“. In: (květ. 2019). URL: http://arxiv.org/abs/1905.05055.

[40] J. R.R. Uijlings et al. „Selective search for object recognition“. In: *International Journal of Computer Vision* 104 (2 zář. 2013), s. 154–171. ISSN: 09205691. DOI: 10.1007/s11263-013-0620-5.

[41] C Lawrence Zitnick a Piotr Dollár. *Edge Boxes: Locating Object Proposals from Edges*. Tech. zpr.

[42] Kaiming He et al. „Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37 (9 zář. 2015), s. 1904–1916. ISSN: 01628828. DOI: 10.1109/TPAMI.2015.2389824.

[43] Joseph Redmon et al. *You Only Look Once: Unified, Real-Time Object Detection*. 2016. arXiv: 1506.02640 [cs.CV].

[44] Wei Liu et al. „SSD: Single Shot MultiBox Detector“. In: *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, s. 21–37. DOI: 10.1007/978-3-319-46448-0_2. URL: https://doi.org/10.1007%5C%2F978-3-319-46448-0_2.

[45] Tsung-Yi Lin et al. *Focal Loss for Dense Object Detection*. 2018. arXiv: 1708.02002 [cs.CV].

[46] Ross Girshick et al. „Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation“. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. Čvn. 2014, s. 580–587. DOI: 10.1109/CVPR.2014.81.

[47] Ross Girshick. „Fast R-CNN“. In: (dub. 2015). URL: http://arxiv.org/abs/1504.08083.

[48] Shaoqing Ren et al. „Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks“. In: (čvn. 2015). URL: http://arxiv.org/abs/1506.01497.

[49] Kaiming He et al. „Mask R-CNN“. In: (březen 2017). URL: http://arxiv.org/abs/1703.06870.

[50] CVAT.ai Corporation. *Computer Vision Annotation Tool (CVAT)*. Ver. 2.2.0. 2022. DOI: 10.5281/zenodo.4009388. URL: http://cvat.ai/.

[51] Tsung-Yi Lin et al. „Microsoft COCO: Common Objects in Context“. In: (květ. 2014). URL: http://arxiv.org/abs/1405.0312.

[52] Mark Everingham et al. „The pascal visual object classes (VOC) challenge“. In: *International Journal of Computer Vision* 88 (2 čvn. 2010), s. 303–338. ISSN: 09205691. DOI: 10.1007/s11263-009-0275-4.

[53] Jianxiong Xiao et al. „SUN database: Large-scale scene recognition from abbey to zoo“. In: 2010, s. 3485–3492. ISBN: 9781424469840. DOI: 10.1109/CVPR.2010.5539970.