VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

DETEKCE BUNĚK VE SNÍMCÍCH ZACHYCENÝCH POMOCÍ MIKROSKOPIE

DETECTION OF CELLS IN CONFOCAL MICROSCOPY IMAGES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. Michal Hubálek
AUTHOR

VEDOUCÍ PRÁCE — Ing. Pavel Škrabánek, Ph.D.
SUPERVISOR

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav automatizace a informatiky
Student:	**Bc. Michal Hubálek**
Studijní program:	Aplikovaná informatika a řízení
Studijní obor:	bez specializace
Vedoucí práce:	**Ing. Pavel Škrabánek, Ph.D.**
Akademický rok:	2021/22

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Detekce buněk ve snímcích zachycených pomocí mikroskopie

Stručná charakteristika problematiky úkolu:

Nejmodernější systémy optické mikroskopie umožňují získat velké množství obrazových dat potřebných pro biomedicínský výzkum. Jejich následné zpracování však vyžaduje aplikaci systému, který umožní jejich automatizované třízení. Požadovaný systém musí být schopen v obrazových datech detekovat zdravé buňky. Znalost jejich polohy umožní určit oblasti zájmu, pro které je potřebné získat snímky s vyšší datovou hustotou.

Cíle diplomové práce:

Student navrhne, implementuje a verifikuje algoritmus pro detekci izolovaných kardiomyocytů ve snímcích zachycených pomocí konfokální mikroskopie. Student realizuje tento algoritmus v podobě software, který bude umožňovat jak trénování detektoru na označených datech, tak i automatizované zpracování neoznačených dat. Interface software bude vytvořen dle požadavků výzkumníků v oblasti biologie.

Seznam doporučené literatury:

GONZALEZ, Rafael C. a Richard E. WOODS. Digital image processing. New York, NY: Pearson, [2018]. ISBN 978-0133356724.

PAWLEY, James, ed. Handbook of Biological Confocal Microscopy [online]. 3. US: Springer, 2006 [cit. 2019-09-03]. ISBN 978-0-387-45524-2. Dostupné z: https://www.springer.com/gp/book/9780387259215

ZHAO, Zhong-Qiu, Peng ZHENG, Shou-Tao XU a Xindong WU. Object Detection With Deep Learning: A Review. IEEE Transactions on Neural Networks and Learning Systems [online]. 2019, 30(11), 3212-3232 [cit. 2020-10-08]. ISSN 2162-237X. Dostupné z: doi:10.1109/TNNLS.2018.2876865

XU, Mengjia, Dimitrios P. PAPAGEORGIOU, Sabia Z. ABIDI, Ming DAO, Hong ZHAO, George Em KARNIADAKIS a Qing NIE. A deep convolutional neural network for classification of red blood cells in sickle cell anemia. PLOS Computational Biology [online]. 2017, 13(10) [cit. 2019-09-03]. DOI: 10.1371/journal.pcbi.1005746. ISSN 1553-7358. Dostupné z: http://dx.plos.org/10.1371/journal.pcbi.1005746.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2021/22

V Brně, dne

L. S.

doc. Ing. Radomil Matoušek, Ph.D. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

ABSTRAKT

Cílem diplomové práce bylo vytvořit aplikaci automaticky detekující zdravé buňky kardiomyocytů ze snímků zachycených konfokálním mikroskopem. Práce vznikla na základě konkrétních potřeb výzkumníků ze Slovenské akademie věd, kterým usnadní práci a ušetří čas, protože doposud musejí snímky vyhodnocovat a hledat vhodné buňky ručně. Pro detekci je použita konvoluční neuronová síť RetinaNet, která byla implementována do uživatelsky přívětivé desktopové aplikace. Aplikace také automaticky zaznamenává a ukládá souřadnice detekovaných buněk využitelné pro snímání buněk ve vyšší obrazové kvalitě. Další výhodou vytvořené aplikace je univerzálnost, která umožňuje natrénovat detekci i na jiných datech, čímž je použitelná i na dalších detekčních projektech. Výsledkem práce je funkční, samostatně spustitelná a intuitivní aplikace, která je připravena k použití výzkumníky.

ABSTRACT

The goal of the thesis was to create an application that automatically detects healthy cardiomyocytes from images captured by a confocal microscope. The thesis was created based on the specific needs of researchers from the Slovak Academy of Sciences.The application will facilitate and increase the efficiency of their research,because until now they have to evaluate the images and search for suitable cells manually. The RetinaNet convolutional neural network is used for detection and has been implemented in a user-friendly desktop application. The application also automatically records and stores coordinates of detected cells which can be used for capturing cells in higher image quality. Another advantage of the developed application is its versatility, which allows to train detection on other data, making it applicable to other projects. The result of this work is a functional, standalone and intuitive application that is ready to be used by researchers.

KLÍČOVÁ SLOVA

Hluboké učení, konvoluční neuronové sítě, detekce objektů, detekce buněk, konfokální mikroskopie, kardiomyocyty, RetinaNet

KEYWORDS

Deep learning, convolutional neural networks, object detection, cell detection, confocal microscopy, cardiomyocytes, RetinaNet

ÚSTAV AUTOMATIZACE A INFORMATIKY

2022

BIBLIOGRAFICKÁ CITACE

HUBÁLEK, Michal. Detekce buněk ve snímcích zachycených pomocí mikroskopie. Brno, 2022. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/136985. Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav automatizace a informatiky. Vedoucí práce Pavel Škrabánek.

PODĚKOVÁNÍ

Rád bych poděkoval mému vedoucímu práce Ing. Pavlu Škrabánkovi Ph.D. za věnovaný čas a ochotu při vedení mé diplomové práce. Dále patří díky mé rodině a přátelům za podporu.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že, že tato práce je mým původním dílem, vypracoval jsem ji samostatně pod vedením vedoucího práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury.

Jako autor uvedené práce dále prohlašuji, že v souvislosti s vytvořením této práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následku porušení ustanovení § 11 a následujících autorského zákona c. 121/2000 Sb., včetně možných trestně právních důsledků.

V Brně dne 20. 5. 2022 ……

Michal Hubálek

OBSAH

1 ÚVOD

Tato diplomová práce vznikla ve spolupráci se Slovenskou akademií věd na základě požadavku na zpracování a vyhodnocení velkého množství mikroskopických snímků. Pomocí moderních konfokálních mikroskopů mohou výzkumníci pořizovat velké množství obrazového materiálu v různých obrazových kvalitách, které je potřeba dále zpracovávat podle jejich aktuálních požadavků.

Dosavadní přístup výzkumníků probíhá tak, že v připraveném vzorku nejdříve hledají vhodné buňky, které v dalším kroku mohou nasnímat ve vyšší kvalitě při různých nastaveních. Vytváření snímků ve vysokém rozlišení je časově náročné, a proto je vhodnější nejprve vytvořit přehledový snímek vzorku v menší kvalitě, ve kterém se najdou souřadnice vhodných buněk pro další analýzu a vyhodnocení.

Cílem práce je co nejvíce zjednodušit zmíněný proces procházení nasnímaných dat a určit polohu buněk nutnou k dalšímu zpracování. K tomu bude sloužit detekční program využívající konvoluční neuronové sítě, který bude vytvořen podle potřeb vědců.

Požadavkem výzkumníku je, aby vytvořený program byl schopen detekovat zdravé buňky kardiomyocytů ve snímcích zachycených pomocí konfokálního mikroskopu a uložit jejich nalezenou polohu ve snímku. Detekce buněk bude probíhat za pomoci předtrénované konvoluční neuronové sítě na manuálně označených datech. Poloha detekovaných buněk bude následně vyznačena ohraničujícími obdélníky a uložena do souboru pro následné zpracování.

Dalším dílčím požadavkem je univerzálnost programu. Kromě samotné detekce by program měl být schopen konvoluční neuronovou síť jednoduše natrénovat i na jiných vstupních datech. Díky tomu by aplikace měla mít univerzální použití na rozličné úkoly.

Teoretická část obsahuje přehled a popis fungování neuronových sítí, jak vypadá základní architektura s důrazem na konvoluční neuronové sítě, které jsou vhodné pro práci s obrazovými daty a detekci objektů. Je vysvětleno názvosloví, které se používá v oblasti konvolučních neuronových sítí. Dále je v práci porovnáno několik vhodných architektur pro detekci objektů, z nichž byla následně vybrána vhodná architektura, která splňovala požadavky této práce.

V praktické části je popsán průběh vytváření uživatelské aplikace v jazyku python, grafické uživatelské rozhraní a následné trénování vybrané architektury. Zdokumentováno je několik významnějších běhů trénování, při kterých se dosahovalo nejlepších výsledků v přesnosti detekce. Součástí práce je také podrobný návod pro uživatele, který popisuje potřebné prerekvizity pro spuštění a ovládání aplikace.

2 NEURONOVÉ SÍTĚ

Historie umělých neuronových sítí jako přístupu v umělé inteligenci sahá již do 60. let 20. století, zvýšené popularity ale dosahují až v novém tisíciletí spolu se zvyšujícím se počítačovým výkonem. [1]

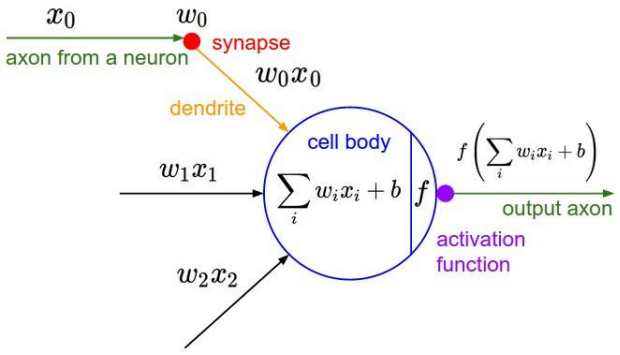
Umělá neuronová síť je výpočetní struktura vhodná pro zpracování komplexních dat a je inspirována nervovou soustavou člověka. Mezi hlavní výhody patří možnost zpracovávat velké množství informací zároveň díky paralelnímu zapojení. Neuronové sítě našly využití v mnoha rozličných aplikací.

2.1 Biologická inspirace

Nervová soustava je komplexní systém v těle člověka a jejím základním prvkem je nervová buňka – neuron. Neuron je schopný sbírat, zpracovávat a dále přenášet informace dále v nervové soustavě. Na neuronu rozlišujeme tělo, vstupní přenosové kanály – dendrity a výstupní přenosový kanál – axon. Pokud se vstupní signály – vzruchy sečtou a dojde k aktivaci neuronu, neuron vyšle signál dále do axonu. Axon se dále větví a jeho zakončení jsou spojeny synapsemi s následujícími neurony. [2]

2.2 Architektura neuronové sítě

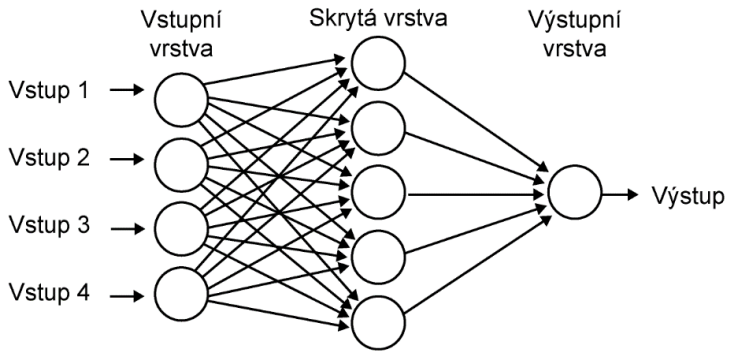
Každá umělá neuronová síť je tvořená z formálních neuronů, které jsou vzájemně propojeny. Nejjednodušší model neuronové sítě obsahuje právě jeden neuron a nazývá se Perceptron [3]. Neuron představuje bod, uzel v sítí, který přijímá vstupy od ostatních neuronů a vypočítává výstup neuronu. Vstupů může být n, ale výstup je vždy jeden. Každý vstup x je ohodnocen synaptickou váhou w, která představuje důležitost vstupu. Výstupem je tedy vážená suma součinů vstupů x a synaptických vah w, po které následuje aktivační (přechodová) funkce f, která definuje výstup z neuronu. Grafické znázornění je na obrázku 1. [3]



Obrázek 1: Model formálního neuronu [2]

Hlavní myšlenkou perceptronu je, že synaptické váhy jsou schopné učení a nastavují důležitost vstupu mezi ostatními vstupy. Pokud vážená suma přesáhne určitou hodnotu aktivační funkce, neuron se aktivuje a vyšle signál na výstup. [2]

Umělá neuronová síť je tedy složena z jednotlivých neuronů, které jsou uspořádány ve vrstvách, podobně jako na obrázku 2. Mezi vstupní a výstupní vrstvou se nachází jedna nebo více skrytých vrstev. Vstupní vrstva je označení pro první vrstvu, ve které neprobíhá výpočet a informace jsou předány do následující vrstvy. Neuron ve vstupní vrstvě předává informaci všem neuronům v následující, skryté vrstvě. Ve skrytých vrstvách jsou obsaženy neurony, ve kterých probíhá výpočet a úprava vah a jejich přesunutí do další skryté vrstvy, anebo do výstupní vrstvy. Výstupní vrstva následuje po skrytých vrstvách a na ní probíhá výsledný výpočet a aktivace nelineární aktivační funkce, která představuje výsledek z celé neuronové sítě. [2]



Obrázek 2: Struktura umělé neuronové sítě [38]

Neuronové sítě, které obsahují více skrytých vrstev, se označují jako hluboké neuronové sítě (*Deep neural networks*). S hloubkou neuronové sítě roste její výpočetní náročnost, ale také její schopnosti zpracovávat složitější a náročnější úlohy.

2.3 Aktivační funkce

Aktivační funkce určuje, zda neuron zareaguje a pošle signál dále v architektuře. Často je také označována jako přenosová funkce. Běžně používaná aktivační funkce je například sigmoid funkce. Aktivační funkce sigmoid σ je nelineární, spojitá, monotónně rostoucí a ohraničená funkce mezi nulou a jedničkou. Její průběh je na obrázku 3 a její rovnice je: [4]

$$\sigma\,(a) = \frac{1}{1 + e^{-a}}, \tag{1}$$

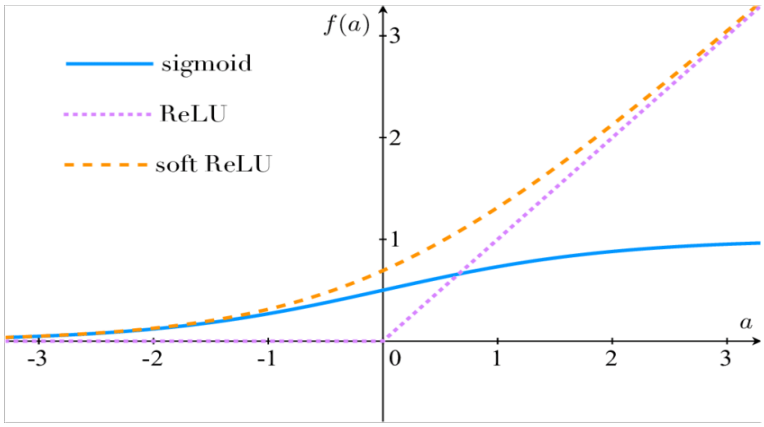
kde a je vstupní nezávislá proměnná patřící do reálných čísel a e je eulerovo číslo.

Další běžně používanou aktivační funkcí je ReLU aktivační funkce (*Rectified lienar Unit*) a její použití převažuje v konvolučních neuronových sítích. Tato funkce vrací nulu pro záporné vstupní hodnoty a pro kladné hodnoty se chová lineárně. Oproti funkci

sigmoid je jednodušší a rychlejší pro výpočet. Průběh aktivační funkce ReLU je na obrázku 3 a aktivační funkce lze vyjádřit:

$$f(a) = \begin{cases} 0 \; pro \; a < 0, \\ a \; pro \; a \geq 0. \end{cases} \tag{2}$$

Problémem funkce ReLU však je, že všechny záporné hodnoty se okamžitě stávají nulovými, což snižuje schopnost modelu správně se přizpůsobit nebo trénovat z dat. Variací na funkci ReLU je mnoho. Pokud nemá aktivační funkce ReLU ostrý přechod v nule, označuje se jako soft ReLU a je na obrázku 3.



Obrázek 3: Průběh aktivačních funkcí sigmoid, ReLU a soft ReLU [43]

Další používaná aktivační funkce je funkce SoftMax. Použití této aktivační funkce je vhodné v posledních vrstvách klasifikačních úloh, kde dochází k rozdělování vektorů do tříd. Protože SoftMax aktivační funkce normalizuje vektor hodnot na vektor pravděpodobností, jejichž součet je jedna. Rovnicí je aktivační funkce vyjádřena:

$$f(\vec{a})_i = \frac{e^{a_i}}{\sum_{n=1}^{N} e^{a_n}}, \tag{3}$$

kde $\vec{a}$ je vstupní vektor funkce SoftMax, a_i jsou prvky vstupního vektoru a mohou nabývat libovolné reálné kladné i záporné hodnoty, což není platné rozdělené pravděpodobnosti, a proto je zde využita funkce SoftMax. Výraz $\sum_{n=1}^{N} e^{a_n}$ ve spodní části vzorce je normalizační výraz. Zajišťuje, že všechny výstupní hodnoty budou mít součet jedna a každá jednotlivá z nich bude v rozsahu nula až jedna. Tím je zajištěno platné rozdělení pravděpodobnosti. [5]

2.4 Učení neuronových sítí

Snahou učení je postupné upravování vah ve vrstvách neuronové sítě tak, aby se dosahovalo co nejlepších možných výsledků. Učení probíhá postupně a váha může nabývat jak kladných, tak i záporných hodnot, tedy neurony se mnou navzájem jak podporovat, tak potlačovat.

Učení s učitelem

Učení s učitelem (*supervised learning*) znamená, že existuje množina vstupů a zároveň existuje množina správných výstupů. Výstup z umělé neuronové sítě se poté porovná s požadovaným výstupem a provede se úprava synaptických vah tak, aby docházelo ke snížení rozdílu mezi aktuálním a požadovaným výstupem. [6]

Při aplikaci neuronových sítí na zpracování obrazu se nejčastěji používá právě učení s učitelem. Je dána trénovací množina, která obsahuje správně označená data, a díky ní probíhá proces učení. Naučenou neuronovou síť je poté možné aplikovat na testovací neoznačená data. [7]

Učení bez učitele

U učení bez učitele (*unsupervised learning*) neexistují žádné trénovací, správně označená, data, podle kterých by se mohl kontrolovat správný výstup. Neuronová síť se snaží nacházet podobné vzory, relevantní příznaky, pomocí kterých je schopna vstupní data roztřídit do podobných podmnožin, shluků (*clusters*), které mají podobné vlastnosti. Tím dojde v určitých případech ke snížení variability vstupních dat a lze s nimi lépe pracovat. [7]

3 KONVOLUČNÍ NEURONOVÉ SÍTĚ

Pro potřeby strojového vidění a práci s obrazovými daty se používají konvoluční neuronové sítě (*Convolutional Neural Network*) zkratkou CNN nebo ConvNet. Jedná se o typ hluboké neuronové sítě, do které je vstupem matice představující vstupní obraz. Jeden pixel v obrazu představuje jeden neuron ve vstupní vrstvě konvoluční neuronové sítě. Důležité vstupní rozměry pro konvoluční neuronovou síť jsou tedy výška, šířka a hloubka.

Konvoluční neuronové sítě byly představeny již v 80. letech minulého století, jejich popularita začala stoupat začátkem nového tisíciletí se vzrůstajícím výpočetním výkonem. V roce 2012 byla představena architektura AlexNet [8], která se považuje za průlomovou v oblasti zpracování obrazu. Díky širokému poli uplatnění se konvoluční neuronové sítě staly pro strojové vidění nenahraditelné.

3.1 Vrstvy a pojmy v konvolučních sítích

Konvoluce je obecná matematická operace, při které se pracuje s dvěma funkcemi. Jedná se o zpracovávanou vstupní funkci f a funkci h, která označuje konvoluční jádro (*convolution kernel*). Výstupem konvoluce je tedy upravená původní funkce.

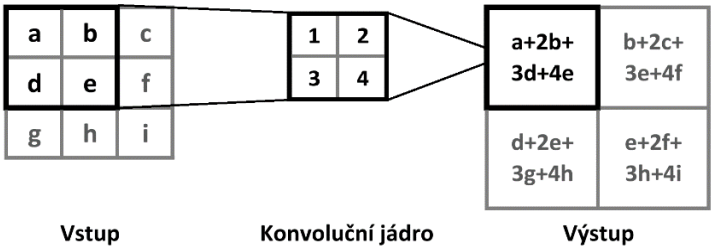
Při zpracování obrazu se používá dvoudimenzionální diskrétní konvoluce, která pracuje s diskrétními hodnotami jasu ze vstupního obrazu na dvoudimenzionálním poli (2D). Rovnice diskrétní konvoluce je:

$$f(x,y) \star h(x,y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} f(x-i, y-j) * h(i,j), \qquad (4)$$

kde $\star$ značí operátor pro konvoluci, proměnné x a y představují rozměry 2D polí.

Při diskrétní konvoluci interaguje vstupní matice s maticí, která se nazývá konvolučním jádrem nebo také konvolučním filtrem. Jádrem se prochází postupně celý vstupní obraz, jak je znázorněno na obrázku 4. Každý pixel na vstupním obrazu je vynásoben s váhou na odpovídajícím místě v jádře. Následně jsou tyto hodnoty sečteny a vytváří jednu hodnotu na výstupu. Výstupní matice je menší než matice vstupní, protože na výpočet jednoho bodu výstupní matice je na vstupu potřeba matice o velikosti jádra. [1]

Pokud je nutné, aby výstupní velikost odpovídala vstupní matici, je možné přidat na každou stranu odpovídající počet řádků a sloupců s nulovými hodnotami (*zero padding*). [1]

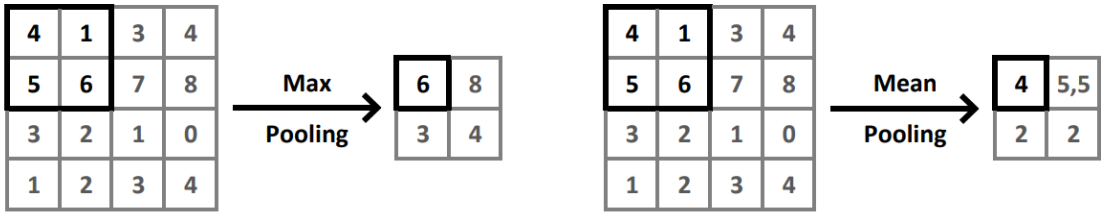


Obrázek 4: Princip výpočtu konvoluce používané při zpracování obrazu.

Konvoluční vrstva (*Convolutional layer*). Jedná se o jednu ze stěžejních vrstev, podle které mají konvoluční neuronové sítě název. V této vrstvě se provádí konvoluce a na jedné vrstvě může být aplikováno několik konvolučních jader. Průchodem konvoluční vrstvy vznikají mapy příznaků (*feature maps*), které shrnují přítomnost těchto příznaků na vstupu do vrstvy.

Konvoluční vrstvy, které jsou blízko vstupu, se učí jednoduché příznaky, jako jsou čáry a hrany. V následujících vrstvách je síť schopná se učit více abstraktně, například tvary nebo konkrétní objekty. Limitací výstupu z konvolučních vrstev je, že zaznamenávají přesnou polohu příznaků do mapy příznaků. Poté i malá změna pozice na vstupu povede k jiné mapě příznaků. Předcházením této limitace je zapojení sdružovací vrstvy. [9]

Sdružovací vrstva (*Pooling layer*). Další vrstvou v architektuře konvoluční neuronové sítě je vrstva, která slouží ke zmenšení dvoudimenzionální mapy příznaků. Díky tomu se zmenší počet příznaků a sníží se výpočetní náročnost pro následující vrstvy. Na mapu příznaků se aplikuje filtr často 2×2 pixelů s posunem o dva pixely. Z vybrané oblasti se buď vybírá největší hodnota pixelů, sdružování podle maxima (*max pooling*), která se dále přenáší do výstupní vrstvy, nebo se počítá průměrná hodnota pixelů, sdružování podle průměru (*average, mean pooling*). Druhy sdružování jsou zobrazeny na obrázku 5. Kromě zmenšení rozměrů slouží sdružovací vrstva i k z obecnění detektoru, nedochází totiž k přesnému přenosu lokace příznaku, ale k přenosu jeho relativní polohy. [1]



Obrázek 5: Princip funkce sdružování podle maxima a sdružování podle průměru

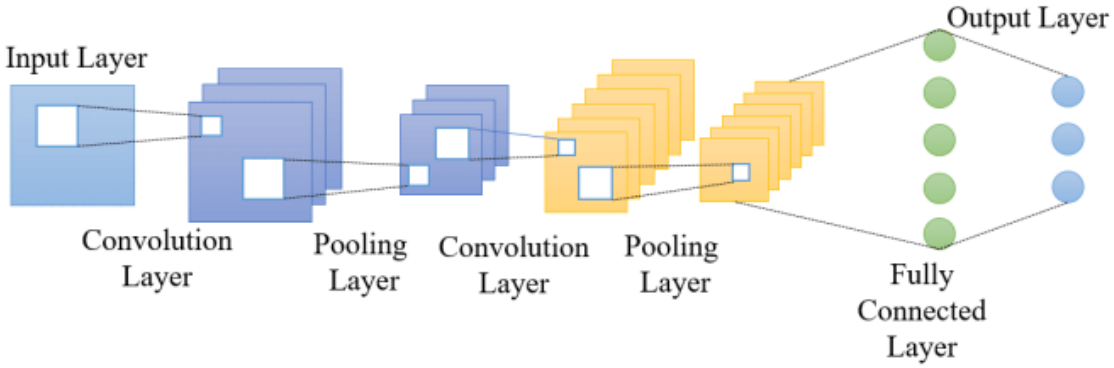
Plně propojená vrstva (*fully connected layer, dense layer*). Neurony ve vrstvě jsou všechny propojeny s neurony ve vrstvě následující. Jedná se o stejný princip jako u běžných vícevrstvých neuronových sítí. Tyto vrstvy jsou často umístěny za poslední

sdružovací vrstvou a její příznaková mapa je transformována na jednorozměrný vektor. [1]

Aktivační funkce SoftMax se vyskytuje v posledních vrstvách architektury a používá se při klasifikaci do více tříd. Funkce SoftMax vrací vektor pravděpodobností každé třídy s tím, že suma všech pravděpodobností je 1. Výsledkem je tedy určení, do které třídy objekt nejpravděpodobněji spadá.

Páteřní síť (*backbone*) – Architektura sítě pro detekci objektů často vychází ze základních architektur pro klasifikaci obrazu. Pro tuto část sítě se používá pojem páteřní síť. Často lze zvolit různé architektury, které se liší svou rozsáhlostí. Používané páteřní sítě jsou například AlexNet, VGG16 a ResNet.

Architektura konvoluční neuronové sítě – Architektura konvoluční neuronové sítě popisuje, jak jsou za sebou poskládány jednotlivé vrstvy. Architektury jsou často rozdílné a každý model má jiné počty i pořadí vrstev, ale základní struktura bývá podobná a je zobrazena na obrázku 6.



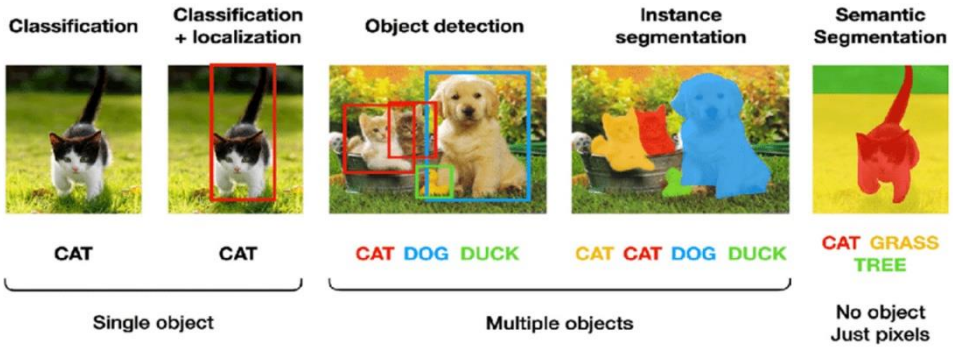
Obrázek 6: Základní architektura konvoluční neuronové sítě [39]

Konvoluční vrstvy jsou obvykle následovány aktivačními funkcemi, po bloku konvolučních vrstev následuje vrstva sdružovací. Konvoluční vrstvy a sdružovací vrstvy se navzájem střídají. Na konci sítě se nachází plně propojené vrstvy, po kterých následuje aktivační funkce SoftMax. Jak vrstvy postupují hlouběji, je běžné, že jejich rozměry se díky sdružovacím vrstvám zmenšují. [10]

3.2 Využití konvolučních neuronových sítí

Konvoluční neuronové sítě mohou obraz například klasifikovat, detekovat nebo segmentovat. Na obrázku 7 jsou jednotlivé úkony zobrazeny. Na svém začátku konvoluční neuronové sítě pracovaly s jednotlivými snímky s nízkým rozlišením a postupem času se dopracovaly až ke zpracování video obsahu v reálném čase.

Základní úlohou je klasifikace (*classification*). Vstupem je vždy obraz a v úloze klasifikace je na výstupu k obrazu přiřazena třída a pravděpodobnost, s jakou patří vstupní obraz do dané třídy. Celému obrazu je přiřazena třída, která charakterizuje vše zachycené na snímku.



Obrázek 7: Porovnání výstupu klasifikace, detekce objektů a segmentace. V obrazu se může detekovat jeden hlavní objekt nebo i více objektů. Předposlední případ je segmetnace instancí pro více objektů a v posledním případě je každý pixel přiřazen k určité třídě. [40]

Často jsou pořízené snímky více různorodé, neobsahují pouze jeden hlavní nejvýraznější objekt. Detekce objektů (*object detection*) je proto vhodnější pro identifikaci více relevantních objektů v jednom vstupním obrázku. Oproti klasifikaci je zde nutné prvně nalézt souřadnice detekovaného objektu a následně objekt klasifikovat. Výstupem je třída nalezeného objektu a souřadnice ohraničující objekt, nejčastěji pomocí obdélníku (*bounding box*).

Poslední úlohou je segmentace (*image segmentation*). Zde probíhá klasifikace každého pixelu ve vstupním obraze. Ke každému pixelu je přiřazena třída a pixely se stejnou třídou jsou v obraze viditelně odlišeny a ukazují i tvar nalezeného objektu. Díky tomu je možné rozpoznávat i pozadí snímku a celou scénu rozdělit do tříd. Segmentace může být dvou typů na segmentaci instancí (*instance segmentation*) a sémantickou segmentaci (*semantic segmentation*). Sémantická segmentace přiřadí každému pixelu třídu, ale více objektů stejné třídy bude označeno jako jeden objekt. Oproti tomu segmentace instancí rozlišuje i objekty jedné třídy. [11]

3.3 Trénování a validace

Pro vstupní data se používá pojem dataset a jedná se o označená vstupní data, která slouží pro natrénování neuronové sítě. Dataset lze obecně rozdělit na část trénovací (*training dataset*), validační (*validation dataset*) a testovací dataset (test dataset). Na trénovacím datasetu probíhá učení neuronové sítě. Validační dataset slouží pro úpravu parametrů během trénování, případně i pro validaci úspěšnosti naučené neuronové sítě. Testovací dataset je vhodný pro finální validaci úspěšnosti. Často jsou pojmy validační a testovací dataset vzájemně zaměňovány, případně je dataset rozdělován pouze na dvě části. Běžně se pro validaci používá 10 až 20 % datasetu. Tyto data jsou oddělená od trénovacího datasetu a neprobíhá na nich trénování. Pokud je nedostatek trénovacích dat, tak v takovém případě se model nemůže učit obecné vlastnosti objektů a dochází

k přeučení. Při malé velikosti validačního datasetu hrozí, že nebude dostatečně reprezentovat trénovací množinu a výsledky budou málo relevantní. [12]

Úspěšnost architektury se vyhodnocuje použitím mAP (*mean Average Precision*). Metrika mAP porovná ohraničující obdélníky s detekovanými obdélníky a vrátí skóre. Čím vyšší skóre, tím přesnější je model ve svých detekcích.

Při trénování je potřeba pracovat s několika pojmy. Trénovací proces je dělen na epochy. Epocha je jedna iterace, při které je síti předložen celý trénovací dataset. Při malém počtu epoch je síť podučená a validační chyba (*validation loss*) je vysoká. Dataset je předkládán k trénování po několika snímcích v dávkách (*batchs*), které mají určitou velikost (*batch size*). Počet kroků (*steps*) určuje kolik je potřeba dávek k dokončení jedné epochy a je dán podílem velikostí datasetu a velikostí jedné dávky. [13]

3.4 Vyhodnocování trénování

Existuje několik metrik, jak lze vyhodnocovat přesnost detekce. Jedna z používaných je metrika Average Precision (AP). Zda detekce byla přesná se používá Intersection over Union (IoU). Zde se porovnává detektorem detekovaná oblast A se skutečně anotovanou oblastí B. Jedná se o podíl průniku a sjednocení těchto dvou oblastí

$$IoU(\mathrm{A},\mathrm{B}) = \frac{A \cap B}{\mathrm{A} \cup \mathrm{B}}. \tag{5}$$

Pokud výsledek podílu je větší než 0,5, detekce je považována jako skutečně pozitivní (*true positive*), neboli označená buňka byla správně detekována. Pokud je výsledek nižší, označí se jako falešně pozitivní (*false positive*). Zde je výsledek detekce označen jako buňka i přesto, že se o označenou buňku nejedná. Poslední případ nastává, když označená buňka není vůbec detekována. Tento stav se značí jako falešně negativní (*false negative*). Pomocí těchto pojmů se mohou spočítat metriky *precision* a *recall*, které se dají přeložit jako přesnost a výtěžnost pokrytí

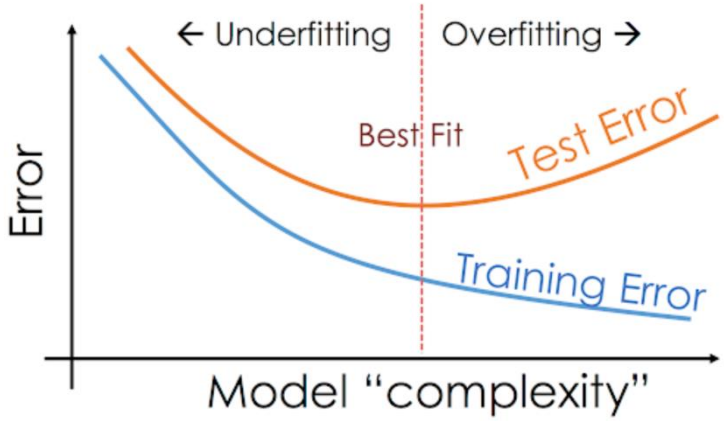
$$precision = \frac{skutečně\ pozitivní}{skutečně\ pozitivní + falešně\ pozitivní}, \tag{6}$$

$$recall = \frac{skutečně\ pozitivní}{skutečně\ pozitivní + falešně\ negativní}. \tag{7}$$

Precision určuje, jaký podíl pozitivních identifikací byl skutečně správný a *recall* představuje, jaký podíl skutečně pozitivních identifikací byl identifikován správně. AP je dále určeno tvarem křivky *precision* a *recall*. [35]

3.5 Problém přeučení a podučení

V průběhu učení sítě je snaha dosáhnou co nejvyšší přesnosti. Jak je zobrazeno na obrázku 8, ze začátku přesnost na trénovacích i validačních datech stále roste, v tom případě je síť ještě podučená (*underfitting*) a je potřeba pokračovat v trénování. Postupně přesnost na trénovacích datech během učení by stále rostla, ale přesnost na datech validačních začíná klesat a vzniká přeučení (*overfitting*). [1]



Obrázek 8: Podučení a přeučení sítě. Je zobrazena hranice, při které se síť stále zlepšuje na trénovacích datech, ale na testovacích datech dochází ke zhoršování [14]

Přeučení sítě je způsobeno tím, že se model neuronové sítě učí ze vstupních dat, které nedostatečně reprezentují obecné příznaky reálného problému. Při učení s nedostatkem různorodých vstupních dat dojde k zapamatování trénovací množiny a k naučení konkrétních znaků, které nevedou k potřebnému zobecnění. Nejjednodušším způsobem, jak problém přeučení vyřešit, je dodat dostatečný počet různorodých vstupních dat, to ale v mnoha aplikacích není možné, proto se používají metody, které problém přeučení zmírňují. [14]

Augmentace dat

Jedna z technik, jak lze zmírnit zmíněné přeučení sítě z nedostatku vstupních dat, je rozšíření dat (*data augmentation*). Jedná se o umělé zvětšení vstupního datasetu o snímky s náhodnou transformací tak, aby dataset obsahoval více různorodých snímků. [3]

Na snímky můžeme aplikovat jak základní druhy geometrických i barevných operací, tak i složitější augmentační techniky. Při učení neuronová síť bere takto upravené snímky jako jedinečné a díky tomu je dataset uměle rozšířen a může dojít k lepším výsledkům trénování.

První skupinou jsou geometrické transformace. Mezi základní operace patří změna poměru stran, rotace, vertikální i horizontální otočení, oříznutí, přiblížení či oddálení obrazu, translace a smyk. Při těchto metodách se obraz deformuje tvarem či velikostí a spolu s obrazem se deformují i ohraničující obdélníky označující objekty. [15]

Druhou skupinou jsou barevné transformace, které mění hodnoty jasu jednotlivých pixelů. Do této skupiny patří transformace, které jsou aplikovány plošně na celý obraz, jako například změna barvy a odstínu, zvýšení a snížení jasu, změna kontrastu, saturace a doostření či rozmazání. Mezi další typ operací patří přidání šumu, při kterých

jsou změněny náhodné pixely obrazu. Používané jsou Gaussův šum a šum sůl a pepř. Výhodou barevných transformací je, že se nemění pozice ohraničujících obdélníků, které označují objekty v obraze a nedochází k deformaci objektu. [15]

Ne vždy je vhodné použití všech augmentačních technik na zadaný problém. Pro ilustraci je jednoduchý příklad na datasetu obsahující auta. Ve většině případů je nevhodné takovéto snímky otáčet o 180°, protože takovéto snímky nepřidávají žádnou informační hodnotu k datasetu.

Mezi pokročilé metody augmentace lze zařadit transformace, které se aplikují pouze na část obrazu. Mezi ně například patří deformace pouze oblastí ohraničených obdélníkem nebo náhodné výřezy obrazu (*cutout*). Při výřezech jsou náhodně velké obdélníky v obraze nahrazeny černou či šedou barvou.

Výpadek (*Dropout*)

Jeden z dalších způsobů, jak předcházet přeučení je přidání vrstvy s výpadkem. Výpadek náhodně vynuluje výstupní příznaky ve vrstvě během trénování. S pravděpodobností 0,5 je možné, že příznak bude nulový a tím se naučený příznak dočasně zapomene. Do učení je tak vnášen umělý šum, který zmírňuje přeučení. [1]

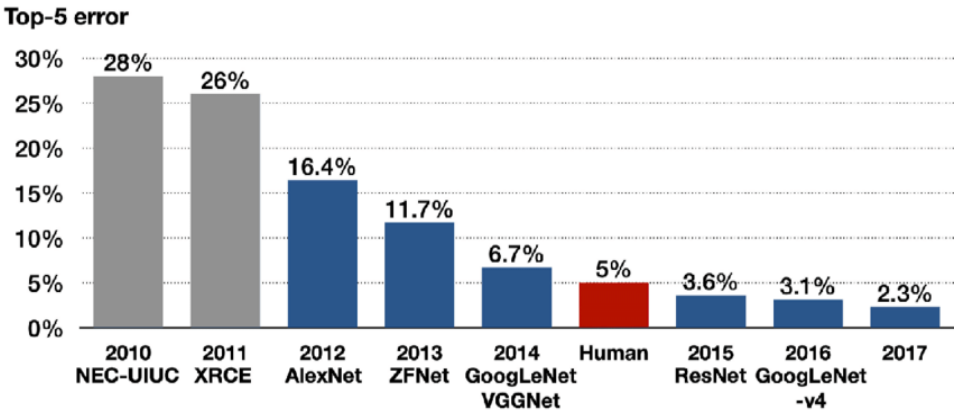
Předčasné ukončení (*Early stopping*)

Pokud se validační ztráta přestane zlepšovat, dojde k přerušení trénování a uloží se nejlepší dosavadní model. Jedná se o použití zpětného volání, které ovlivní běh trénování, přeruší ho. Před trénováním lze nastavit počet iterací trénování, při kterých nedochází ke zlepšení výsledku – zmenšení validační ztráty, po kterých dojede k přerušení. [1]

3.6 Architektury konvolučních neuronových sítí pro klasifikaci

Výzkumné týmy přicházely postupem času se stále dokonalejšími a složitějšími architekturami, které vedly k přesnějším výsledkům prvně v úlohách klasifikace, poté i v úlohách detekce či segmentace. Detektory objektů často vychází z konvolučních neuronových sítí pro klasifikaci, tvoří páteřní sít, proto jsou zde uvedeny i klasifikační detektory.

V celosvětové soutěži ILSVRC (*ImageNet Large Scale Visual Recognition Challenge,* [16]) se v roce 2012 prosadila síť AlexNet a od tohoto roku všechny následující vítězné architektury jsou založeny na podobných principech konvolučních sítí. Dalším průkopníkem se stala v roce 2015 síť ResNet [17], která pokořila důležitou hranici menší chybovosti než 5 %, která představuje lidskou chybovost při klasifikaci na soutěžním datasetu. Na obrázku 9 jsou chybovosti vítězných topologií v jednotlivých ročnících a zmíněná hladina lidské chyby.



Obrázek 9: Vítězné architektury soutěže ILSVRC od roku 2010 do roku 2017 [41]

3.6.1 AlexNet

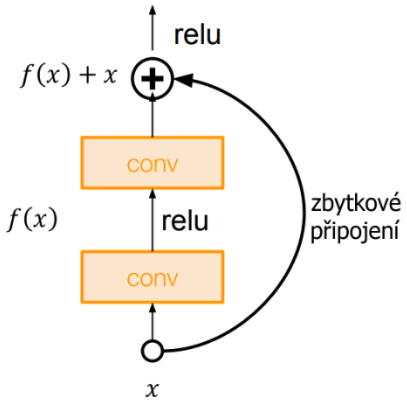
Síť AlexNet [8] z roku 2012 je označována jako průkopník na poli konvolučních neuronových sítí pro klasifikaci obrazu. Na datasetu ImageNet dosáhla lepších výsledků v přesnosti než dosavadní tradiční metody a určila, jakým způsobem se budou řešit klasifikační úlohy v budoucnu. Oproti síti LeNet-5 [18], která byla použita na datasetu číslic MNIST o několik let dříve a mohla by být považována za první použití konvoluční sítě, pracuje AlexNet s o mnoho větším počtem vstupů a klasifikuje reálné RGB fotografie v rozměru 256×256 pixelů. [8]

Na obrázku 10 je znázorněna architektura sítě AlexNet skládající se z pěti konvolučních vrstev v kombinaci se sdružovacími vrstvami podle maxima a poté následují tři plně propojené vrstvy. Po poslední vrstvě je použita funkce SoftMax. Ke zrychlení výpočtů dochází díky použití nelineární aktivační funkci ReLU, protože je jednodušší na výpočet než dosavadní sigmoid aktivační funkce.

Obrázek 10: Znázornění pořadí vrstev v architektuře sítě AlexNet [42]

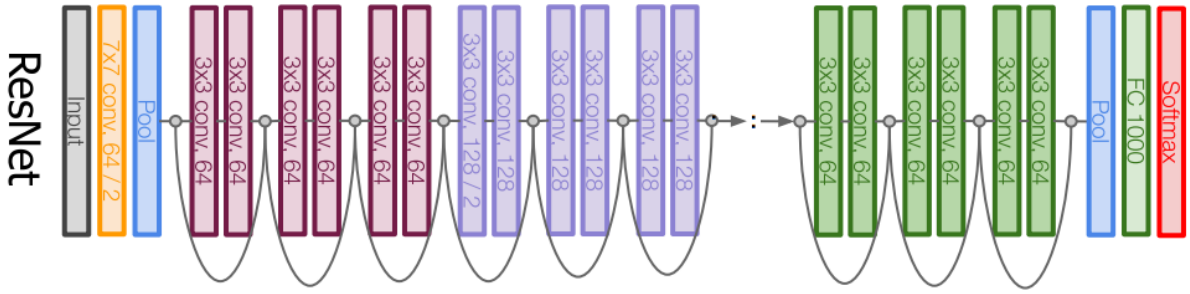
3.6.2 ResNet

Další z používaných architektur je síť ResNet (*Residual Neural Networks*, [17]), kterou vytvořil v roce 2015 výzkumný tým Microsoftu. Oproti AlexNet již obsahuje o mnohem více vrstev. Používají se varianty obsahující 34, 50, 101 či 152 vrstev. U běžné architektury však neplatí, že čím hlubší architektura, tím síť dosahuje přesnějších výsledků. S rostoucí hloubkou se zpětně propagovaný signál stává stále slabší a má stále menší a menší vliv na učení. Jedná se o problém většiny hlubokých sítí, který se označuje jako problém mizejícího gradientu (*vanishing gradient problem*, [19]). Architektura ResNet řeší tento problém prostřednictvím tzv. zbytkového modelu (*residual model*), jehož reprezentace je na obrázku 11.



Obrázek 11: Zbytkové připojení residuálního modelu [42]

Zbytkové spojení přenáší předchozí výstup x do vrstvy následující a zároveň do vrstvy, která je o několik vrstev dále, kde se přičte k novému výstupu $f(x)$. Dojde k přeskočení několika vrstev (*skip connections*). Obecně platí, že zbytkové připojení bude přínosné, pokud se použije u modelů, které mají více než 10 vrstev. Proto je tento přístup k vidění i u jiných, pozdějších architektur. Celá architektura s naznačenými residuálními spoji je k vidění na obrázku 12. [1]



Obrázek 12: Architektura sítě ResNet s residuálním spojením [42]

4 DETEKCE OBJEKTŮ V OBRAZE

Detekce objektů je součástí strojového vidění a jedná se o úkol nalezení objektu v obraze a následnou klasifikaci. Hlavní myšlenkou detekce objektů v obraze je určení souřadnic kde se objekty v obraze nachází a do které kategorie každý nalezený objekt patří.

4.1 Struktura detektoru

Model detekce lze obecně rozdělit do následujících částí.

Výběr regionů (*Region selection*). Objekty v obraze, které se mají detekovat, jsou v obraze různě rozmístěné a mohou mít odlišnou velikost a tvar, proto je potřeba prohledat celý obraz posuvným oknem s různými stupni velikostí. Z tohoto přístupu se získají všechny možné polohy objektů. Z toho také vyplývají nedostatky. Kvůli velkému počtu oken kandidátů je výpočet velmi náročný a je produkováno velké množství nadbytečných oken, které zachycují například pozadí obrazu a nemají žádnou informační hodnotu. Na druhou stranu, pokud je pouze fixní počet možných oblastí kandidátů, výsledek nalezených oblastí nemusí být uspokojující. V architekturách konvolučních neuronových sítí se používají různé způsoby, jak predikovat regiony zájmu. [10]

Extrakce příznaků (*Feature extraction*). Pro rozpoznávání různých objektů v obraze a jejich správnou klasifikaci, je potřeba extrahovat příznaky, které sémanticky reprezentují hledané objekty. Jedná se o vlastnosti obrazu, které jsou typické pro třídu objektů. Extrakce příznaků vede ke snížení počtu objektů, pro které má být klasifikace realizována. Algoritmy, které mají za úkol extrahovat příznaky z obrazu, se nazývají deskriptory příznaků (*feature descriptors*). [10]

Klasifikace (*Classification*). Poslední částí je klasifikace, která má za úkol rozřadit nalezené objekty do správných kategorií. Pro každý nalezený objekt je vypočítána pravděpodobnost, s jakou patří do přiřazené kategorie. [10]

4.2 Architektury konvolučních neuronových sítí pro detekci objektů

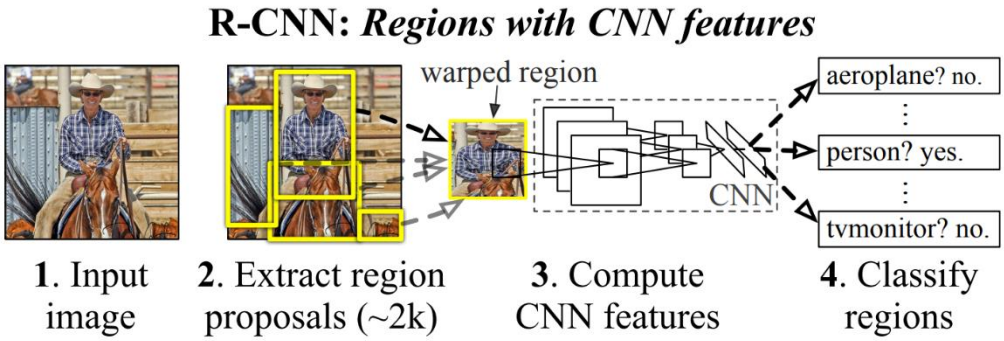
Existující detektory lze rozdělit na jednostupňové a dvoustupňové. Ve dvoustupňovém detektoru prvně probíhá návrh regionů (*region proposal*) a v druhé části probíhá klasifikace jednotlivých navržených regionů, zda objekt obsahují a do které třídy patří. Dvoustupňové detektory byly dlouhou dobu přesnější, ale zato pomalejší než jednostupňové. [10]

U jednostupňového detektoru po extrakci příznaků následuje návrh regionu a následná klasifikace v jednom kroku. Není předem dán pevný počet navržených regionů. Díky tomu jsou jednostupňové detektory rychlejší, ale nedosahovaly takových přesností jako dvoustupňové detektory. [10]

4.2.1 R-CNN

R-CNN (*Region-based Convolutional Neural Network*) je dvoustupňový detektor, který v roce 2013 způsobil výrazný posun v detekci objektů. Oproti dosavadnímu způsobu hledání podezřelých oblastí (*region proposals*) prostým posouváním okna, již podezřelé oblasti aktivně vyhledává pomocí selektivního vyhledávání (*selective search*). Jedná se o hledání a třídění podobně barevných pixelů do podmnožin, které se následně slučují do větších celků. Tímto způsobem na vstupním obrazu vznikne 2000 podezřelých oblastí. [20]

V druhé části se pro každou podezřelou oblast naleznou příznaky pomocí architektury AlexNet a následně se oblast klasifikuje do tříd pomocí klasifikátoru SVM (*Support Vector Machine*) a určí se, s jakou přesností se jedná o nalezený objekt. Schéma fungování sítě R-CNN je naznačeno na obrázku 13. [21]



Obrázek 13: Schéma fungování sítě R-CNN. 1) Vstupní obraz 2) Extrakce podezřelých oblastí 3) Výpočet příznaků pomocí CNN 4) Klasifikace regionů [20]

Stejný autor o dva roky později přichází s architekturou **Fast R-CNN** [22], která řeší některé nedostatky předchozího návrhu. Nedochází k vytváření pevného počtu regionů, ale na vstupní obraz je použita konvoluční vrstva s přechodovou funkcí SoftMax pro generování mapy příznaků. Z mapy příznaků jsou dále získány pouze regiony, které pravděpodobně obsahují umístěný objekt. Tento detektor také sdílí výpočty konvolucí mezi jednotlivými okny s objekty a díky tomu dosahuje detektor výsledků o mnoho rychleji a není tak výpočetně náročný jako první varianta. Jako páteřní síť je zde použita architektura VGG16. [21]

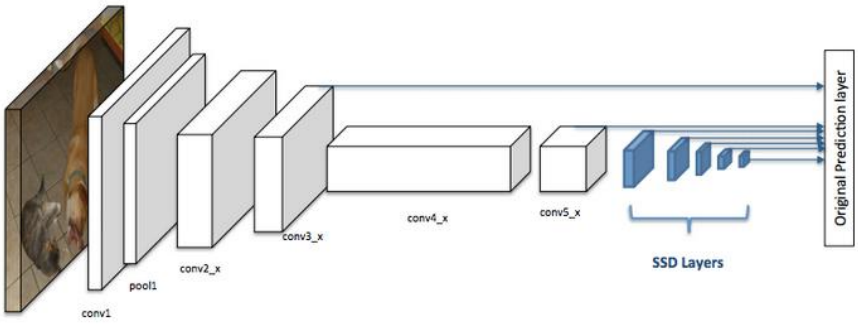
Dalším zdokonalením detektoru je **Faster R-CNN**. V této síti je zdokonaleno vyhledávání podezřelých oblastí a zmenšení jejich počtu asi na 300, které se dále klasifikují. Nepoužívá se již selektivní vyhledávací algoritmus a síť se sama učí navrhnout regiony. Celý proces detekce je zrychlený až na pět snímků za sec. [23]

4.2.2 SSD

Jednostupňová architektura SSD (*Single Shot MultiBox Detector*) je mnohem rychlejší než dosavadní dvoustupňové detektory a se schopností rozpoznat až 59 snímků za vteřinu je vhodná i pro použití v reálném čase. [24]

Obrazový vstup prochází několika konvolučními vrstvami s různými velikostmi konvolučních filtrů a díky tomu síť dokáže nezávisle dobře detekovat objekty i malých

velikostí. Pro predikci relevantních ohraničujících obdélníků slouží příznakové mapy z různých částí sítě a pro extrakci příznaků je zde použita páteřní síť VGG16. V druhé části sítě jsou konvoluční vrstvy přidány k páteřní síti a mají za úkol z příznakových map nalézt ohraničující objekty. Na obrázku 14 je naznačena architektura sítě, kde bíle označené obdélníky představují páteřní síť a na konci jsou modře zvýrazněny vrstvy SSD detektoru. [25]



Obrázek 14: Architektura původní páteřní sítě s SSD vrstvami [24]

Namísto použití posuvného okna SSD architektura rozdělí obraz pomocí mřížky a každá buňka mřížky je zodpovědná za detekci objektů v dané oblasti obrazu. Pokud buňka neobsahuje detekovaný objekt, bere se jako buňka pozadí. Ke každé buňce mřížky může být navrhnuto několik obdélníků různých tvarů, tzv, kotevních rámečků (anchor boxes), které by mohli potenciálně obsahovat objekt. Každý kotevní box je specifikován poměrem stran a úrovní zvětšení, díky tomu lze detekovat objekty různých velikostí. [24]

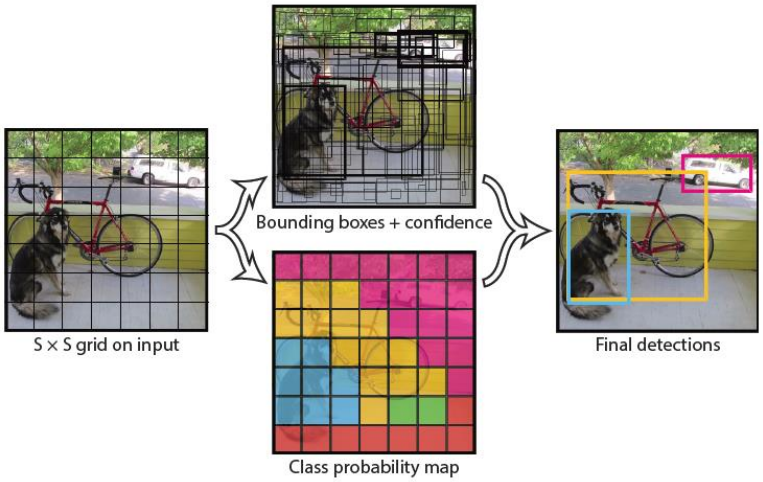
4.2.3 YOLO

Název této architektury je zkratka anglického sousloví *You Only Look Once*. Název odkazuje na průběh detekce. Jednostupňová architektura z roku 2015 předpovídá ohraničující boxy a pravděpodobnosti tříd přímo z celých obrázků během jediného průchodu. Tím je zajištěna vysoká rychlost detekce, až 45 snímků za sekundu a tím je vhodná pro použití v reálném čase. Ve verzi s nižším rozlišením se jedná až o 155 snímků za sekundu. [21]

Na obrázku 15 je zobrazen princip architektury. Architektura rozdělí vstupní obraz mřížkou $S \times S$ a pro každou buňku mřížky jsou předpovězeny ohraničující obdélníky a skóre spolehlivosti (*confidence score*), které představuje, s jakou pravděpodobností buňka obsahuje objekt. Z toho je vytvořena mapa pravděpodobností jednotlivých tříd a následně vznikají finální ohraničující obdélníky. Architektura obsahuje 24 konvolučních vrstev a dvě plně propojené na svém konci. Konvoluční vrstvy jsou předtrénované na datasetu ImageNet. [26]

Detekční síť YOLO má limity v rozpoznávání a segregaci malých objektů, které se vyskytují ve skupinách, a to kvůli rozdělení obrázku mřížkou, kde každá buňka je zodpovědná za detekci jednoho objektu. Od své první verze bylo vyvinuto mnoho,

zlepšujících variant, které předčily výchozí architekturu jak v rychlosti, tak i v přesnosti detekce.



Obrázek 15: Detekce objektů pomocí algoritmu YOLO [26]

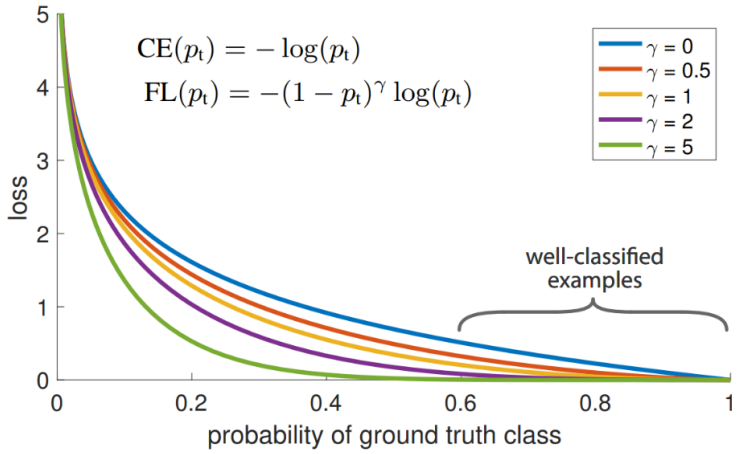
V následujících letech vzniklo několik dalších variant, které navazují na architekturu YOLO a dosahují lepších výsledků. Například v roce 2021 se vítězem každoroční detekční soutěže na datasetu COCO stala architektura YOLOR (*You Only Learn One Representation*) s výsledkem dokonce 57,3 % [27]. Jedná se o vylepšenou verzi YOLO architektury, která vytváří jednotnou reprezentaci, která slouží k plnění různých úkolů najednou. Tato architektura je poměrně nová, byla publikována v květnu roku 2021 a není zatím natolik rozšířená jako její předchozí verze.

4.2.4 RetinaNet

V roce 2018 přišel výzkumný tým *Facebook AI Research* s modelem RetinaNet. Jedná se o jednostupňový detektor, který ale svou přesností dosahuje lepších výsledků než ostatní jednostupňové detektory (SSD, YOLO, a další) a vyrovnává se dvoustupňovým detektorům (R-CNN, Fast R-CNN, Faster R-CNN, a další).

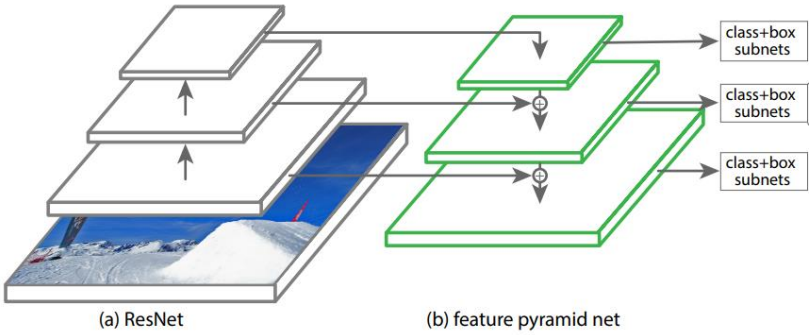
Jednostupňové detektory se během trénování potýkají s problémem nerovnováhy tříd (*Class imbalance*). Detektory uvažují velké množství nalezených oblastí, uvádí se 10^4 až 10^5, ale pouze několik oblastí nakonec objekt skutečně obsahuje, zbytek je možné klasifikovat jako pozadí. To způsobuje neefektivitu trénování. Lepších výsledků je zde dosaženo pomocí nové chybové funkce Focal Loss, která je vylepšením pro dosud používanou chybovou funkci křížové entropie (*Cross Entropy loss*). Obě chybové funkce jsou na obrázku 16. [28]

Chybová funkce Focal Loss řeší problém nerovnováhy tříd přiřazením větších vah složitým, nebo jednoduše zaměnitelným příkladům nalezených oblastí a snížením váhy jednoduše rozpoznatelným oblastem. Díky tomu se nesčítá velké množství jednoduchých příkladů, které by zahlcovaly učení. [29]



Obrázek 17: Porovnání Cross Entropy Loss (CE) Focal Loss (FL). K chybové funkci Focal Loss je přidána část s parametrem γ. Pokud je za proměnnou γ dosazena nula, výsledkem je chybová funkce Cross Entropy. U modelu RetinaNet je zvoleno γ=2. Při hodnotě pravděpodobnosti vyšší než 0,6 se výsledek bere jako dobře klasifikovaný. [28]

Pro extrakci příznaků slouží páteřní síť ResNet a na ní navazuje struktura FPN (*Feature Pyramid Network*). Struktura sítě je navrhnuta na obrázku 17. FPN rozšiřuje standardní konvoluční síť o dráhu shora dolů (*top-down)* a boční připojení, takže síť efektivně konstruuje bohatou, vícerozměrnou pyramidu příznaků ze vstupního obrazu. A každá úroveň pyramidy může být použita pro detekci objektů v jiném měřítku. [28]



Obrázek 16: Zobrazení páteřní sítě ResNet a FPN [28]

5 DETEKOVÁNÍ KARDIOMYOCYTŮ

Cílem praktické části diplomové práce je vytvoření programu pro detekci buněk kardiomyocytů podle požadavků výzkumníků ze Slovenské Akademie Věd (SAV). Je žádoucí, aby aplikace detekovala dospělé buňky na velkém množství mikroskopických dat a byla schopná uložit jejich polohu pro další následné zpracování. Díky tomu nebude nutné manuální procházení a kontrola nasnímaných dat. Výzkumníci mohou se znalostí polohy následně vytvořit nové snímky konkrétních buněk již ve velkém rozlišení. Pro větší univerzálnost aplikace mohou výzkumníci detekční síť natrénovat jak na nových snímcích kardiomyocytů, tak i na vlastních označených datech i pro jiné úlohy detekce objektů.

Konvoluční sítě se zdají jako vhodné řešení, protože snímky neobsahují pouze zdravé buňky, ale je potřeba je oddělit od neživých buněk či ostatních nečistot, nebo od buněk, které nejsou správně zaostřeny, a jsou nevhodné pro další zpracování. Pokud by úkolem bylo detekovat vše bez větší znalosti, pravděpodobně by rychlejším a jednodušším řešením bylo předzpracování obrazu a následné použití jiných jednodušších detekčních aplikací strojového vidění.

5.1 Kardiomyocyty

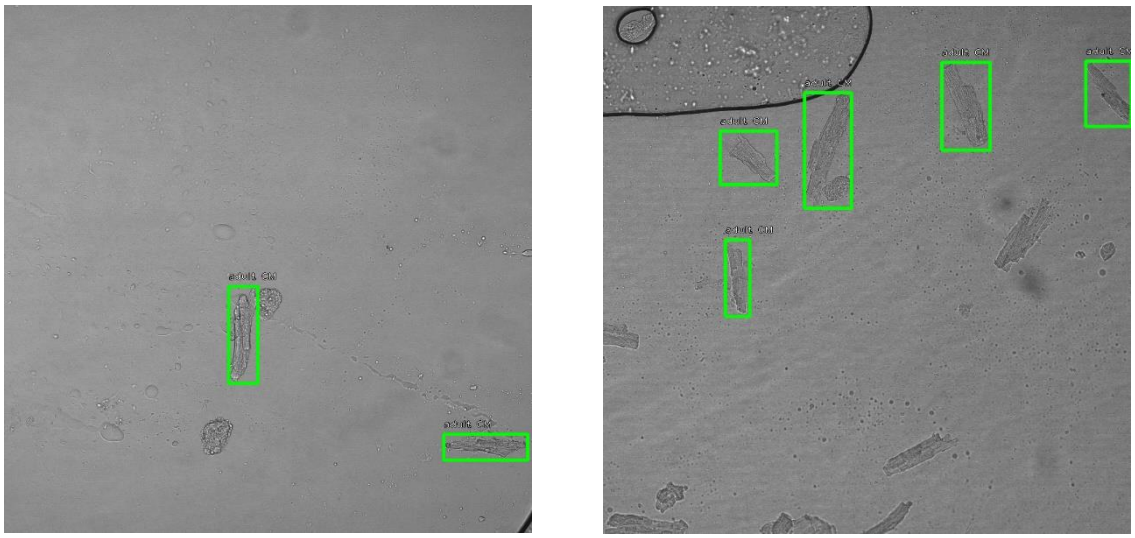
V programu se detekují izolované buňky kardiomyocytů. Kardiomyocyty jsou svalové buňky tvořící tkáň srdeční svaloviny. Jejich tvar je obdélníkový, nebo připomínající písmeno Y a délka je přibližně 85 až 110 μm. V případě výzkumu nejrůznějších patologických stavů na modelech laboratorních zvířat se jedná o svalové buňky srdce. Pomocí fluorescence mohou být na buňkách označeny proteiny a/nebo organely a z jejich záznamů se dají získat změny způsobené patologií nebo změny přirozenými procesy, například růstem. Sledováním těchto změn lze například zjistit, zda léčba dosahuje očekávaných výsledků. [30]

Pro snímání izolovaných kardiomyocytů se používá konfokální mikroskop. Jedná se o druh optického mikroskopu, který je díky svému principu schopný získat vysoce kvalitní snímek z žádané roviny bez rušivých vlivů pozadí. Jako zdroj světla se používá paprsek laseru. Přes bodovou clonu a objektiv je laserem osvětlována rovina zaostření vzorku. Během jednoho kroku snímání se získávají informace právě o jednom konkrétním bodu. V dalších krocích je paprsek postupně pomocí clony zaměřen na další body v rovině a jednotlivé body se skládají do výsledného obrazu. Zmíněný postup je časově náročný a existují kompromisy mezi rychlostí snímání, rozlišením a zorným polem. Malé zorné pole může být zobrazeno rychleji, ale větší vzorky vyžadují více času na skenování. Další možností použití konfokálního mikroskopu je prosvětlovat různé roviny vzorku a z nich výsledně vytvářet 3D obrazy.

Doba, po kterou je možné skenovat závisí na možnostech mikroskopu. Pokud mikroskop obsahuje komoru pro udržování vhodné teploty, obsahu vlhkosti a složení plynů, lze se vzorkem pracovat v řádu hodin až dní.

5.2 Trénovací dataset pro detekci kardiomyocytů

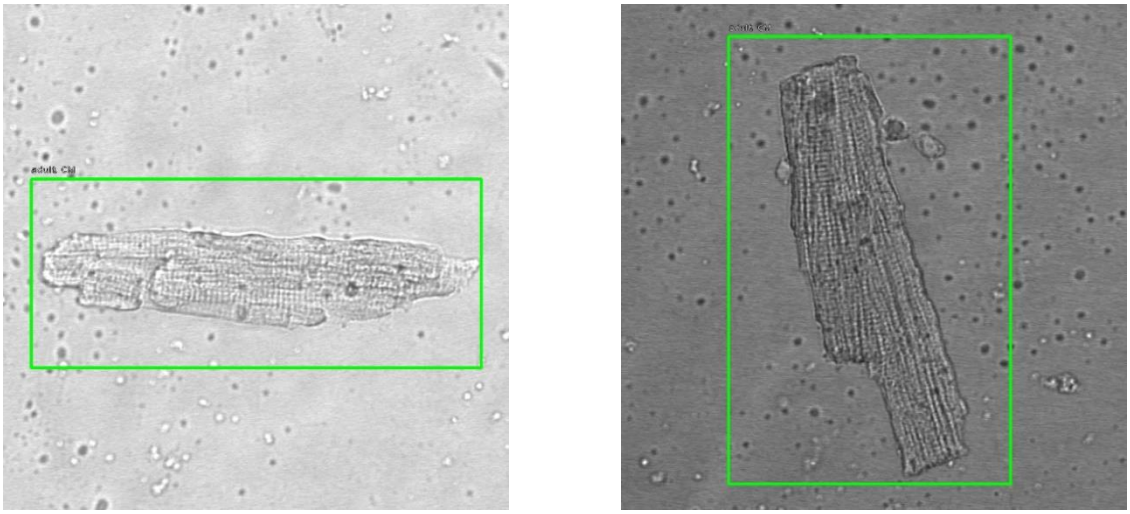
Dataset je tvořen přehledovými snímky z konfokálního mikroskopu v běžném režimu. Vytváření snímků bylo časově náročné, proto bylo výzkumníky z SAV vytvořeno pouze několik snímků, které byly vhodné pro další použití. Běžně se pro trénování používají stovky a tisíce snímků a obecně lze tvrdit, že s velkým množstvím snímků lze dosahovat žádaných výstupů jednodušeji. Dataset by měl obsahovat snímky které jsou reprezentativní, generalizující a vystihují podstatu učeného problému. Snímky obsažené v datasetu by se neměly typově lišit od snímků, na kterých bude prováděna následná detekce. Ukázka snímků z vytvořeného datasetu je na obrázku 18 a na snímcích jsou zobrazeny anotace buněk.



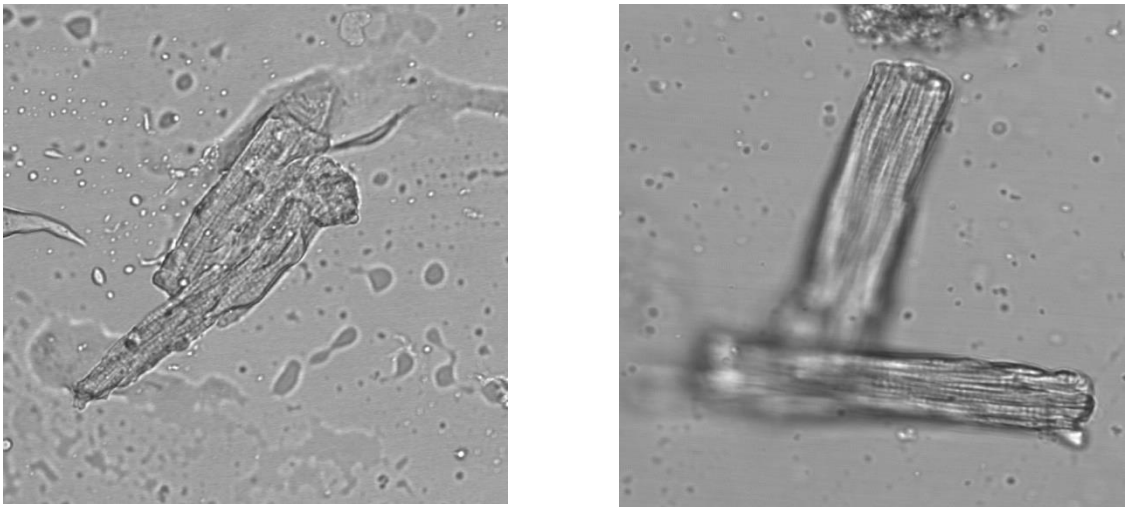
Obrázek 18: Ukázka označených kardiomyocytů na snímcích zachycených konfokální mikroskopií v běžném režimu

Snímky mají rozměry 4096×4096 pixelů a jejich velikost je 16 MB. Vybraná architektura RetinaNet pracuje se snímky v rozlišení 800×800 pixelů a jednotlivé snímky jsou po vybrání předem zpracovány na zmíněnou velikost, proto není nutné snímky předzpracovávat zvlášť. Při plném rozlišení by síť pracovala s o mnoho více vstupními daty a trénování by bylo časově náročnější.

Ne všechny buňky na vzniklém přehledovém snímku jsou pro výzkumníky v oblasti zájmu, a proto nejsou vhodné pro následné kvalitnější snímání. I pro výzkumníky je v některých případech složité správně rozřadit buňky obsažené na snímku. Buňky, které není žádoucí detekovat, jsou nejčastěji deformované, zborcené, nebo jsou částečně rozostřené. I při nejlepší přípravě vzorku se takovéto buňky v obraze vyskytují. Ukázka vhodných a nevhodných buněk je na obrázku 19 a 20.



Obrázek 19: Ukázka správně označených živých dospělých kardiomyocytů a jejich anotačních obdélníků



Obrázek 20: Ukázka nevhodných buněk, které síť nemá umět detekovat. Na levém snímku je buňka již poškozená a na pravém snímku jsou buňky špatně zaostřené a nejsou vhodné pro další zpracování.

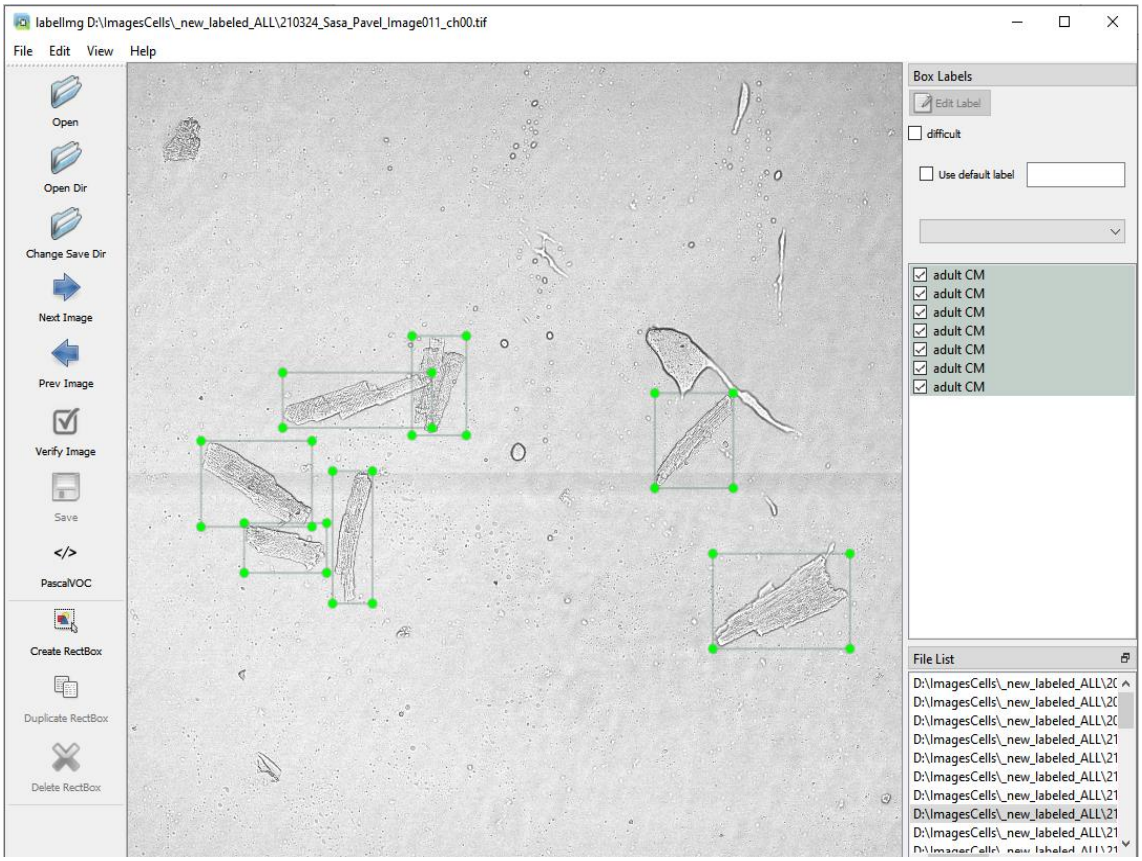
Označená data je nutné rozdělit na trénovací dataset a validační dataset. Validační dataset tvoří přibližně 10 % snímků náhodně vybraných z označených dat a slouží pro určení mAP, jak naučená síť obstojí na jiných datech než trénovacích. Tímto parametrem se určuje kvalita sítě.

Trénovací dataset tedy obsahuje 30 mikroskopických snímků, obsahující 126 označených buněk a validační dataset je tvořen třemi snímky, které obsahují 13 označených buněk. S větším trénovacím datasetem by mohl být větší i validační dataset a tím by byla zajištěna větší správnost.

5.2.1 Anotační nástroj LabelImg

Označenou trénovací sadu je nutné prvně manuálně vytvořit. Každá buňka se ohraničí obdélníkem, který určuje polohu buňky ve snímku. Existuje několik aplikací, které usnadňují označování dat, například jednoduchý nástroj *LabelImg* [31], který je používán pro vytvoření datasetu v této práci. Jeho prostředí je na obrázku 21. Prostředí je intuitivní, uživatelsky přívětivé a lze ho ovládat klávesovými zkratkami.

V některých případech je i pro odborníka náročné rozpoznat, zda se jedná o živou či neživou buňku, proto je důležité při vytváření anotací dbát zvýšené pozornosti na označování správných buněk. Vnášení chyby již v procesu označování povede k nižší úspěšnosti natrénování.



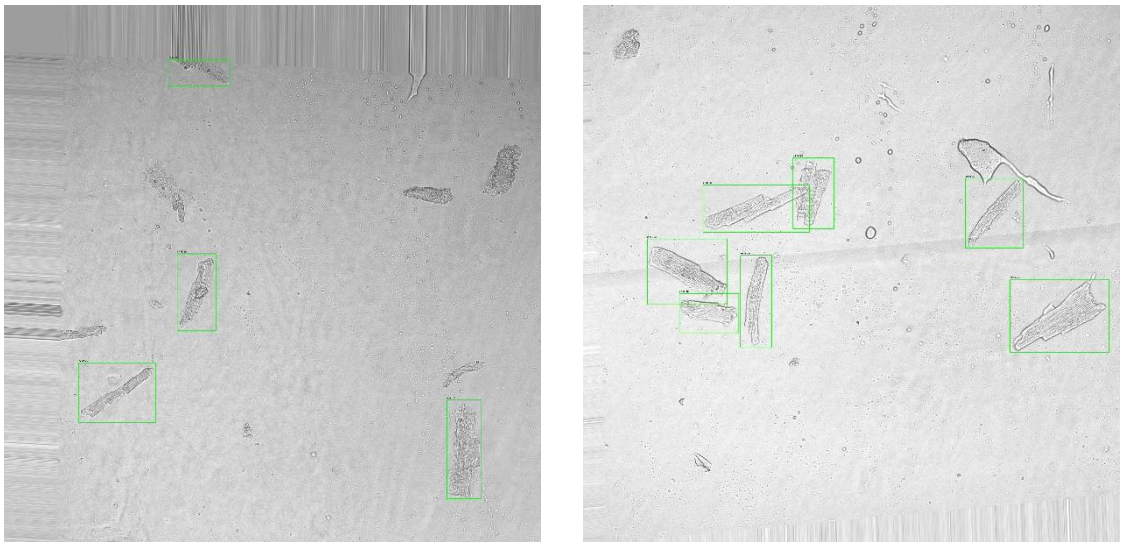
Obrázek 21: Ukázka prostředí nástroje labelImg na konkrétním snímku s kardiomyocyty

Nástroj *LabelImg* ukládá souřadnice buněk, které jsou určeny ohraničujícími obdélníky. V tomto případě se k ukládání souřadnic využívá formát *PascalVOC*, který ke každému snímku vytvoří anotační soubor ve formátu *.xml*, který obsahuje název snímku, jména tříd a souřadnice jednotlivých objektů. Souřadnice obdélníku ve snímku jsou určeny dvěma body, a to levým horním bodem s parametry $xmin\ a\ ymin$ a pravým dolním rohem s parametry $xmax\ a\ ymax$. Je nutné si uvědomit, že souřadnicový systém tohoto formátu začíná v levém horním rohu snímku.

5.2.2 Augmentace datasetu

Zmíněná implementace architektury *RetinaNet* má ve svém základu augmentační generátor, který je schopen náhodně aplikovat základní augmentace. Geometrické transformace jsou v základu rotace, translace, zkosení, přiblížení a převrácení po ose. Barevné operace, které jsou implementovány v generátoru, jsou saturace, změna kontrastu, světlosti a odstínu. Hodnoty změn jsou nastaveny tak, aby docházelo pouze k mírným změnám a nezměnila se podstata datasetu.

Pro potřeby správné detekce je v této úloze nevhodná jakákoliv operace, která deformuje tvar buňky. Proto byl augmentační generátor upraven tak, aby neobsahoval deformující operace. Z tohoto důvodu nebyla použita operace zkosení a operace přiblížení byla nastavena tak, aby se strany zvětšovaly, či zmenšovaly zároveň, a ne docházelo k deformaci tvaru. Na obrázku 22 jsou označené snímky s aplikovanou augmentací.



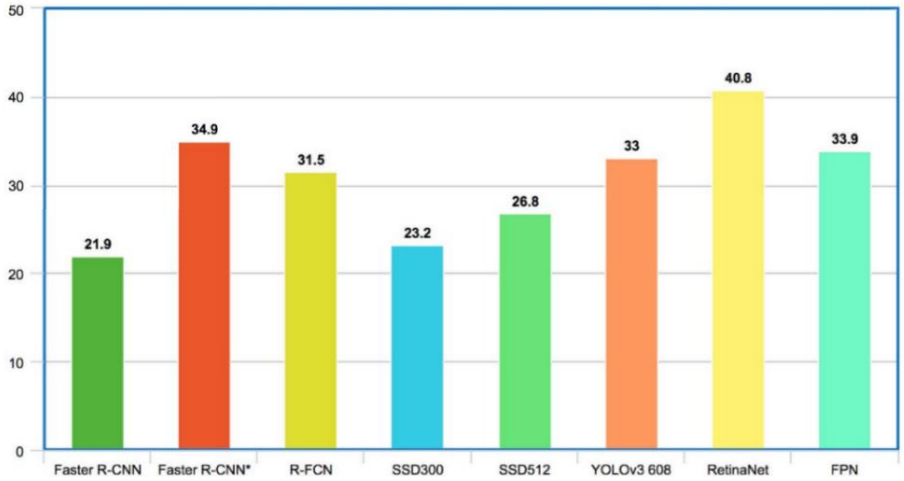
Obrázek 22: Vstupní označená argumentovaná data pro trénování

V implementaci lze příkazem použít všechnu augmentaci, nebo zvolit trénování bez augmentace. Pokud augmentace není použita, snímky v datasetu jsou pouze zrcadlově otočeny jak kolem horizontální, tak i vertikální osy s 50 % pravděpodobností. Tato operace nemá destruktivní účinky a rozšiřuje dataset o nerušivé snímky, které jsou barevně i tvarově nezměněné.

5.3 Výběr vhodné architektury pro implementaci

Pro potřeby aplikace bylo potřeba vybrat vhodnou architekturu konvoluční neuronové sítě. Jednotlivé architektury se mohou posuzovat podle dvou hlavních parametrů, přesnosti a rychlosti detekce. Protože se jedná o detektor, který bude sloužit výzkumníkům pro další práci s výsledky, je žádoucí, aby natrénovaná síť dosahovala co nejpřesnějších výsledků. Není nutné, aby detekce probíhala v reálném čase.

Při porovnání výsledků jednotlivých architektur na datasetu COCO z roku 2018 je na obrázku 23 zobrazeno, že nejlepších výsledků přesnosti mAP dosáhla síť RetinaNet. Od svého vzniku je stále velmi používanou a podporovanou architekturou, která našla uplatnění v mnoha aplikacích. Proto byla vybrána jako vhodná a dostatečně přesná architektura pro následnou implementaci detektoru kardiomyocytů a zároveň její univerzální použití je vhodné i pro trénování na vlastních datech. [32]



Obrázek 23: Porovnání přesností z roku 2018 na datasetu COCO [32]

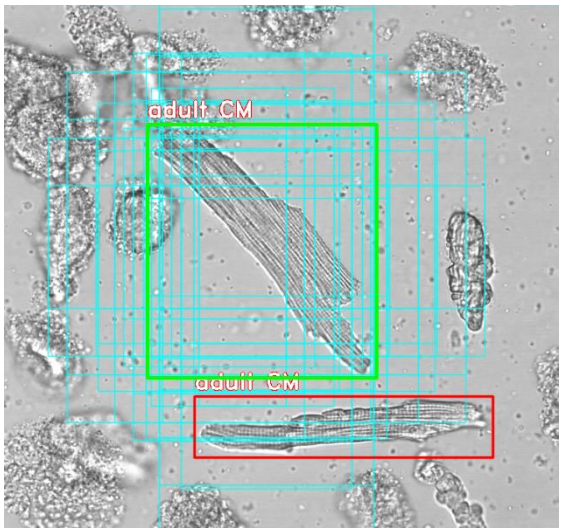
5.4 Implementace RetinaNet

Vybraná architektura *RetinaNet* byla použita v implementaci od firmy *Fyzir* a její zdrojový kód je volně dostupný jako *open source* na stránkách GitHub [33]. Jedná se o dostupné řešení napsané v programovacím jazyku Python verze 3.7.3, s využitím knihovny *Keras* ve verzi 2.4 a knihovny *TensorFlow* verze 2.3.0. Je nutné dbát zvýšené pozornosti, aby byly verze doplňků spolu kompatibilní a předcházelo se tím prvotním nevysvětlitelným problémům. V implementaci je jako páteřní síť použita *ResNet50* a optimalizační algoritmus je použit Adam, který je založen na algoritmu gradientního sestupu podle knihovny *Keras*. Zmíněnou implementaci nebylo nutné před použitím upravovat a její skript *train.py* byl využit pro trénování část ve vytvořené aplikaci.

Knihovna *Keras* je rozšířením pro knihovnou *TensorFlow* a tyto knihovny se používají pro práci s modely hlubokého učení. Knihovna *Keras* se vyznačuje svým uživatelsky přívětivým prostředím, pohodlným vytvářením modelů neuronových sítí, ale zároveň i vysokou funkcionalitou. Jedná se o dvě z nejvíce populárních knihoven pro práci s hlubokým učením.

Implementace také obsahuje skript *debug.py*, který slouží pro odzkoušení a kontrolu, zda trénovací i validační dataset je ve správném tvaru a zda se ohraničující obdélníky zobrazují na správných místech. Dochází k vizuální kontrole datasetu a tím lze předejít prvotním nevysvětlitelným problémům.

K zobrazení anotací slouží parametr *--annotations*. Anotace jsou podbarveny zeleně, pokud jsou k dispozici kotvy (*anchors*), a červeně, pokud kotvy k dispozici nejsou. Kotvy jsou předem dané možné tvary a velikosti objektů pro každou pozici a záleží na implementaci použité architektury. Pokud anotace nemá k dispozici kotvy, znamená to, že nepřispěje k trénování. Nejčastějším problémem je, že anotace jsou příliš malé nebo příliš nevhodně tvarované – roztažené, jak je zobrazeno na obrázku 24.



Obrázek 24: Zobrazení dvou buněk pomocí skriptu debug.py. Zeleně je označená správná anotace se všemi možnými kotvami. Červeně je označená anotace, která není použita pro učení, protože není dostatečně široká.

Díky zobrazení správných a špatně označených anotací pomocí skriptu *debug.py* bylo možné dataset s kardiomyocyty náležitě opravit a tím zlepšit následné trénování. Ve vytvořené aplikaci je uživateli nabídnuta stejná možnost tak, aby uživatel byl schopný vylepšit označený dataset.

5.5 Trénování

Pro trénování je potřeba poměrně velký výpočetní výkon počítače. Pokud počítač tímto výkonem nedisponuje, lze využít služby Google Colab [34], která je používaná pro úlohy hlubokého učení.

Trénování na stolním PC

První možností, jak lze natrénovat neuronovou síť je použití stolního počítače. Zde ale rychlost trénování závisí na komponentech počítače, hlavně na grafické kartě (grafická procesorová jednotka, GPU), která je schopná provádět výpočty paralelně. Vhodné grafické karty jsou od společnosti NVIDIA, která investovala do rozvoje v oblasti hlubokého učení a vytvořila podporovanou sadu nástrojů CUDA [35] pro své grafické karty. Trénovat lze i bez vhodné grafické karty, pomocí procesoru (centrální procesorová jednotka, CPU), ale zde se nevyužívá výhoda paralelního přístupu k výpočtům a trénování je proto časově náročnější.

Program i trénování bylo vytvářeno na běžném kancelářském notebooku Lenovo IdeapPad 720. Mezi hlavní specifikace patří grafická karta RX Radeon 560. Notebook neobsahuje grafickou kartu NVIDIA, proto samotné trénování probíhalo pouze na procesoru Intel Core i5-8250U, bez využití výhod grafické karty.

Prostředí Google Colab

Prostředí Colab, od slova kolaborace, je produkt společnosti Google, který umožňuje v prohlížeči psát a spustit programy napsané v jazyce Python. Toto prostředí je vhodné pro úlohy hlubokého učení, analýzy a zpracování dat. Hlavní výhodou je použití externího výkonného hardware pro potřeby trénování. Tím je zaručena vysoká rychlost trénování bez nutnosti vlastnit výkonný počítač. Potřeba je připojení k internetu a práce se soubory probíhá pomocí Google účtu. Aktuálně je toto prostředí zdarma k použití, s omezením na trénování na maximálně 12 hodin. Pro náročné uživatele existuje služba Colab Pro a Colab Pro+ s měsíčním předplatným, která již neobsahuje další omezení. [34]

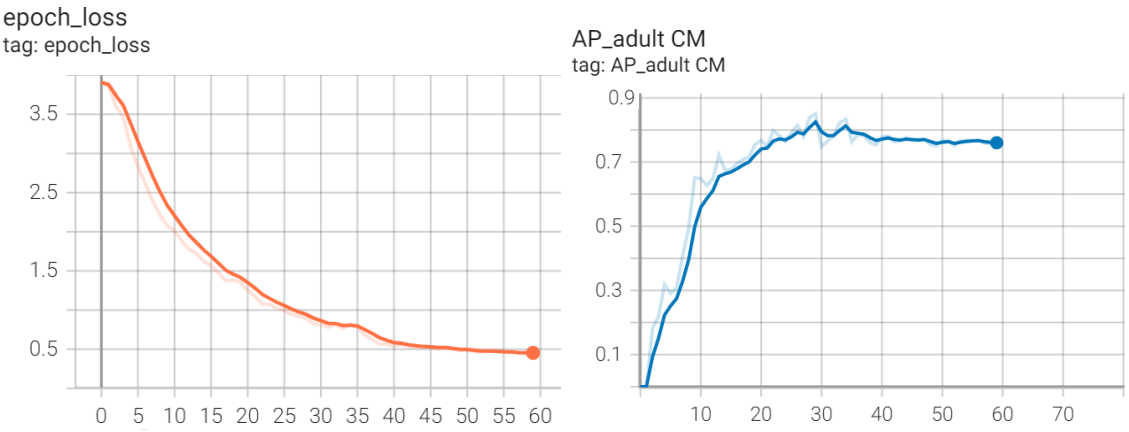
5.6 Proces trénování

Pomocí skriptu *train.py* je model natrénován na vlastním označeném datasetu a výstupem je natrénovaný model konvoluční neuronové sítě s koncovkou ve tvaru *.h5*, který je vhodný pro použití v následující detekci. Pro trénování je potřeba označený dataset a je žádoucí nastavit několik parametrů, které jsou nutné pro spuštění trénování. Mezi ně patří hlavně parametry *--batch-size, --steps a --epochs.* Nastavováním těchto parametrů se dosahuje horších či lepších výsledků přesnosti a parametry je nutné nastavit empirickým přístupem s ohledem na vstupní data. Počet epoch určuje, po jakou dobu bude probíhat trénování. V průběhu trénování se snižuje *learning rate* a pokud nedochází delší dobu ke zlepšení validační ztráty, dojde k předčasnému ukončení trénování.

Během trénování je snaha dosáhnout co nejlepších výsledků následné detekce v závislosti na nastavení parametrů pro trénování. Na konkrétním datasetu s kardiomyocyty byly prováděny běhy trénování s různými poměry velikosti dávky a počtu kroků. Níže jsou porovnány nejúspěšnější běhy a vliv přenosového učení a augmentace na výsledky validačního datasetu. Před provedenými experimenty bylo předpokladem, že jak přenosové učení, tak i augmentace povede k vyšší úspěšnosti detekce kardiomyocytů.

Trénování bez přenosového učení

Při trénování bez přenosového učení se dosahuje nižší přesnosti během většího počtu epoch. Na obrázku 25 jsou průběhy trénování při nastavení sítě, kde velikost dávky je dva a počet kroků je 15. Nejlepších výsledků se dosahuje kolem 30. epochy, kde výsledky jsou 0,85 mAP a se ztrátou (*loss*) 0,89. Při 35. epoše se dosáhlo mírně nižší přesnosti 0,83 mAP s ještě nižší ztrátou 0,76.

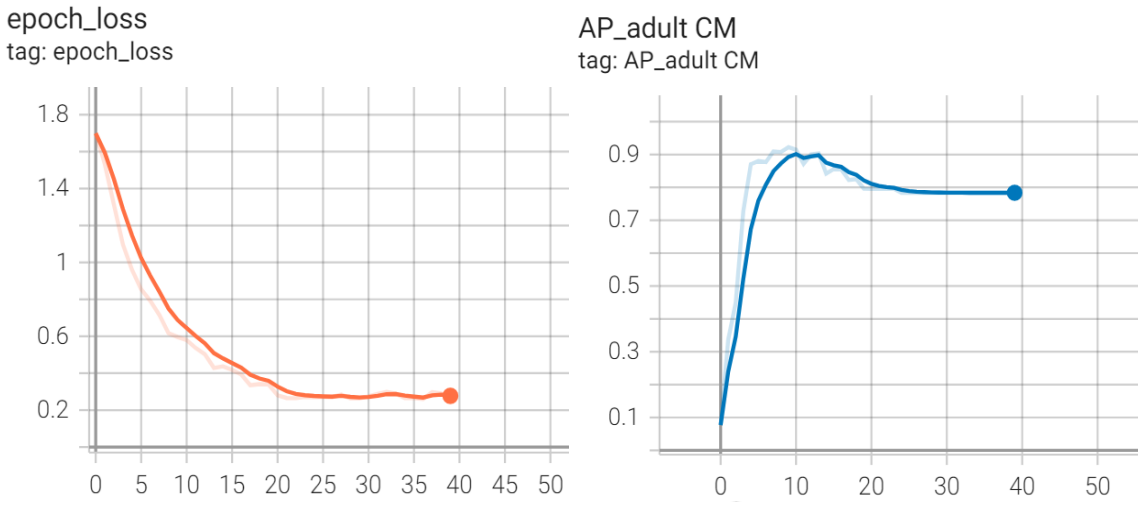


Obrázek 25: Průběh trénování bez přenosového učení.

Trénování s přenosovým učením

Dalším způsobem, jak zlepšit účinnost sítě bylo použít metody přenosového učení (*transfer learning*). Jedná se o použití vah z jiné natrénované detekční sítě a díky tomu lze zlepšit parametry učení. Byly použity předtrénované váhy z rozsáhlého COCO datasetu, který obsahuje třídy reálných fotografií.

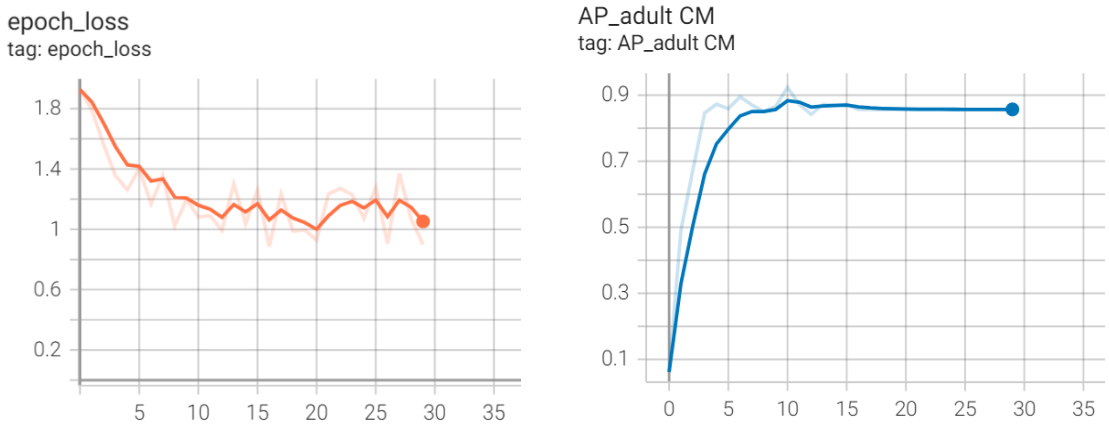
Průběh trénování je zobrazen na obrázku 26. Díky použití předtrénované sítě se dosáhne přesnějších výsledků již v dřívější 10. epoše. Konkrétně se dosáhne přesnosti 0,92 mAP při ztrátě 0,6. Při tomto běhu je počet kroků v jedné epoše stejný jako počet snímků v datasetu, protože velikost dávky je jedna.



Obrázek 26: Průběh trénování s přenosovým učením

Trénování augmentovaného datasetu

Při použití augmentovaného datasetu by mělo dojít ke zlepšení celkových výsledků. Během testování různých nastavení se neprokázalo výrazné zlepšení výsledků. Podle níže uvedených grafů na obrázku 27 se dosahuje podobných přesností jako při trénování na datasetu bez augmentace. Nejlepších výsledků se dosahuje kolem 15. epochy, kde výsledné mAP je 0,87 ale během trénování nedochází k ustálení ztráty a hodnota ztráty se nedostane pod jedna. Nejlepších výsledků se dosáhlo při použití přenosového učení a nastavení velikosti dávky jedna a počtu kroků 30. Doba trénování jedné epochy byla přibližně třikrát delší než při trénování bez použití augmentace.



Obrázek 27: Graf klasifikační ztráty a průměrné přesnosti při trénování s augmentací

6 UŽIVATELSKÁ APLIKACE

Při vytváření aplikace byla snaha o jednoduché přívětivé uživatelské rozhraní tak, aby uživatel pouze se základní znalostí problematiky hlubokých neuronových sítí byl schopen podle přiloženého návodu použít natrénovanou síť na snímcích kardiomyocytů, nebo aby byl uživatel schopný síť natrénovat na vlastních označených datech a využít detekční aplikaci v budoucích projektech. Při konzultacích se SAV docházelo k upřesňování dílčích funkcionalit a dosáhlo se uspokojivé varianty vytvořené aplikace splňující požadované funkce výzkumníků.

6.1 Funkcionalita

Detekce kardiomyocytů

Vytvořená aplikace byla pojmenována CM Detection (zkratka pro anglický název kardiomyocytů, *cardiomyocytes*) a má pomoci výzkumníkům s automatickým zpracováním velkého množství dat z konfokálního mikroskopu tak, aby se dala určit oblast zájmu pro další kvalitnější snímání. Pro výzkumníky požadovaným výstupem jsou souřadnice nalezených buněk uložené v souboru, společně s přesností a absolutní cestou k detekovanému obrázku.

Jedna z dílčích funkcionalit aplikace je zobrazení detekovaných buněk na jednom snímku, tak aby výzkumník byl schopen sám posoudit, zda aktuální výstup z detektoru je podle požadavků vhodný, nebo je potřeba zvětšit, či upravit trénovací dataset, případně změnit nastavení pro trénování.

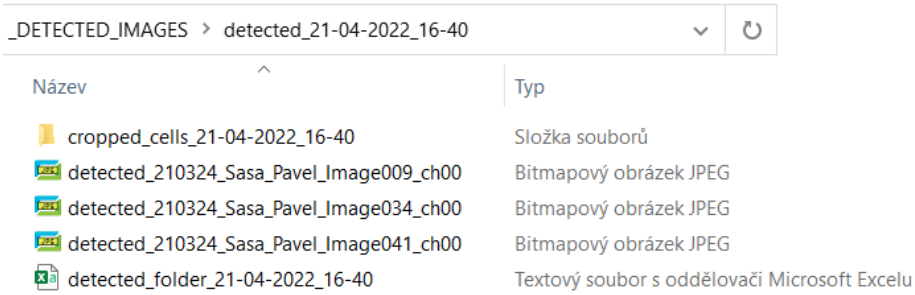
Pokud je uživatel s výstupem spokojen, může načíst složku s daty a spustit detekci pro celou složku. Po proběhlé detekci je vytvořena složka, která obsahuje soubor se souřadnicemi nalezených buněk, výsledný snímek s ohraničujícími obdélníky, a i výřezy jednotlivých detekovaných buněk pro možnou vizuální kontrolu. Navíc byla přidána možnost výřezy jednotlivých buněk neukládat a tím šetřit místo ve výsledné složce. Pokud je tato možnost vybrána, tak výsledkem je soubor se souřadnicemi a přehledový snímek s označenými buňkami.

Pro zvýšení uživatelského komfortu při používání aplikace jsou po najetí na tlačítka zobrazeny komentáře s nápovědou. Jedná se pouze o krátké výstižné zprávy, které mají za úkol doplnit podrobnější informace v návodu, nebo v textu diplomové práce a pomoci tak uživateli s ovládáním aplikace.

Možnosti uložení výstupu detekce

Uživatel si může zvolit, zda chce, aby souřadnice buněk byly uloženy pro každý detekovaný snímek zvlášť, nebo aby všechny snímky byly uloženy do jednoho souboru. V každém případě je vytvořena složka v aktuálním adresáři *_DETECTED_IMAGES* ve tvaru *detected_datum_čas*, kde datum a čas představuje identifikátor, kdy byla detekce provedena. Na obrázku 28 je jako příklad uvedena struktura vytvořené složky po detekci na třech snímcích, pokud uživatel zvolil uložení společně do jedné složky. Výřezy všech

nalezených buněk ze všech snímků jsou uloženy ve složce *cropped_cells_datum_čas*. Detekované snímky s ohraničujícími obdélníky jsou uloženy ve tvaru *detected_název obrázku*.



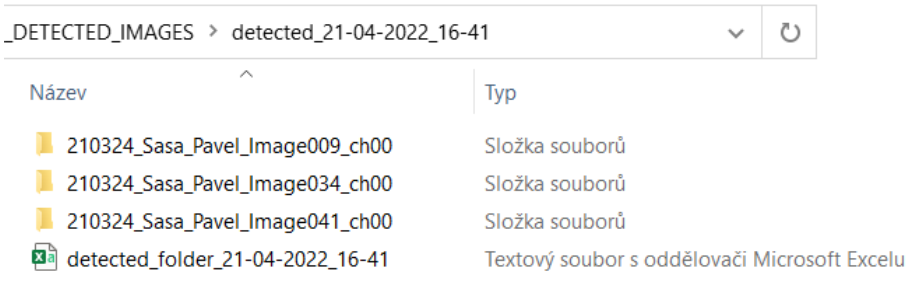
Obrázek 28: Struktura vytvořené složky, pokud uživatel zvolí uložit do jedné složky

Poslední a nejdůležitější soubor *detected_folder_datum_čas.csv* obsahuje souřadnice detekovaných buněk spolu s přesností, indexem třídy a umístěním snímku. Na obrázku 29 je zobrazena zmíněná struktura uloženého souboru. Jako oddělovač mezi parametry je použita čárka. Ohraničující obdélník je určen souřadnicemi levého horního rohu a pravého dolního rohu. Druhý uložený parametr je pravděpodobnost, s jakou si naučená síť myslí, že se jedná o buňku. Poté následuje index třídy, do které nalezený objekt náleží. Na datasetu s kardiomyocyty se detekovala právě jedna třída. Posledním parametrem je absolutní cesta k detekovanému snímku.

```
[2911 1696 3199 2463],0.827,0,C:/Users/micha/PycharmProjects/CMDetection_only_training/dataset-val/210324_Sasa_Pavel_Image009_ch00.tif
[ 872 3167 1599 3375],0.687,0,C:/Users/micha/PycharmProjects/CMDetection_only_training/dataset-val/210324_Sasa_Pavel_Image009_ch00.tif
[ 724 1305 1237 1866],0.623,0,C:/Users/micha/PycharmProjects/CMDetection_only_training/dataset-val/210324_Sasa_Pavel_Image009_ch00.tif
[3346 1650 3700 2598],0.502,0,C:/Users/micha/PycharmProjects/CMDetection_only_training/dataset-val/210324_Sasa_Pavel_Image009_ch00.tif
[1982  592 2371 1477],0.501,0,C:/Users/micha/PycharmProjects/CMDetection_only_training/dataset-val/210324_Sasa_Pavel_Image009_ch00.tif
[1598 2327 2390 2617],0.920,0,C:/Users/micha/PycharmProjects/CMDetection_only_training/dataset-val/210324_Sasa_Pavel_Image034_ch00.tif
[ 821 3084 1303 3469],0.840,0,C:/Users/micha/PycharmProjects/CMDetection_only_training/dataset-val/210324_Sasa_Pavel_Image034_ch00.tif
[2279 3037 2934 3497],0.726,0,C:/Users/micha/PycharmProjects/CMDetection_only_training/dataset-val/210324_Sasa_Pavel_Image034_ch00.tif
[ 226  443  498 1131],0.612,0,C:/Users/micha/PycharmProjects/CMDetection_only_training/dataset-val/210324_Sasa_Pavel_Image034_ch00.tif
[1612 2417 2335 2733],0.761,0,C:/Users/micha/PycharmProjects/CMDetection_only_training/dataset-val/210324_Sasa_Pavel_Image041_ch00.tif
[2185 2084 2923 2403],0.587,0,C:/Users/micha/PycharmProjects/CMDetection_only_training/dataset-val/210324_Sasa_Pavel_Image041_ch00.tif
```

Obrázek 29: Ukázka uložených souřadnic do .csv souboru, obsahující souřadnice. Ve třech snímcích bylo nalezeno 11 buněk. Výsledky jsou zapsány sestupně podle přesnosti pro každý snímek.

Na obrázku 30 je druhá možnost uložení výsledných souborů. Zde se výsledky detekce uloží do podsložek pro jednotlivé snímky zvlášť. Podsložka je tvořena názvem snímku a obsahuje soubor se souřadnicemi, přehledový snímek s vizuálně označenými nalezenými buňkami a jednotlivé výřezy nalezených buněk.



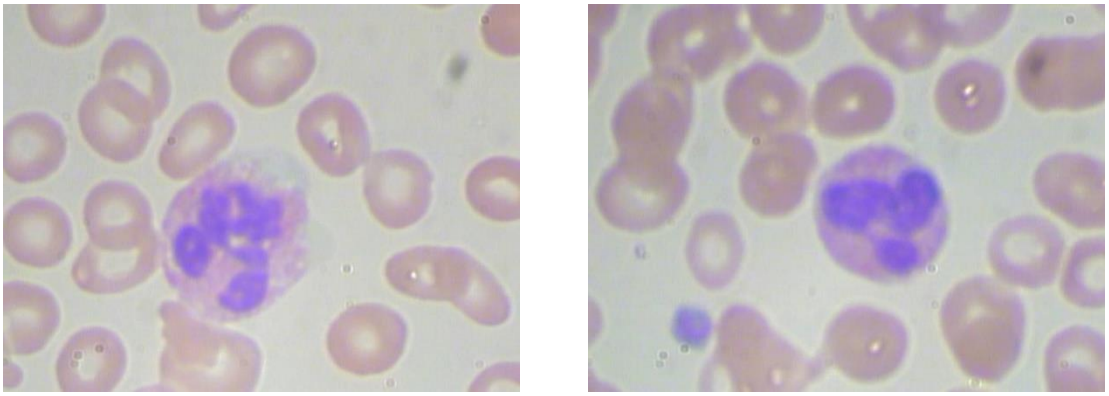
Obrázek 30: Struktura vytvořené složky, pokud uživatel zvolí uložit pro každý snímek zvlášť

Trénování na vlastních datech

Další požadovaná funkce aplikace je schopnost trénování implementované architektury. Je možnost spustit trénování na datasetu s kardiomyocyty a natrénovat síť podle vlastního nastavení, nebo lze model natrénovat na vlastních označených datech. Zde je potenciál v univerzálnosti vytvořené aplikace. Uživatel s pomocí této aplikace je schopný podle přiloženého návodu natrénovat model na vlastních datech a natrénovaný model použít pro vlastní detekci.

Pro trénování se volá skript *train.py* a k trénování je nutný označený dataset, které uživatel vybírá pomocí vytvořeného GUI. Dále si uživatel nastavuje základní parametry trénovacího běhu, které jsou nutné pro spuštění. Výsledkem je natrénovaná síť, která lze použít v detekční části.

Pro ověření funkcionality, zda aplikace je použitelná i na vlastních datech, byla architektura natrénována i na jiném datasetu než na datasetu s kardiomyocyty. Byl vybrán dataset *Complete Blood Count (CBC) Dataset* [36], který je typově podobný datasetu kardiomyocytů. Jedná se o mikroskopické snímky, které obsahují tři druhy krevních buněk. V datasetu jsou označeny červené krvinky, bílé krvinky a krevní destičky. Na obrázku 31 jsou zobrazeny dva ukázkové snímky z datasetu.



Obrázek 31: Ukázka datasetu krevních buněk CBC. Největší buňka s modrým středem je bílá krvinka. Na pravém snímku malá modrá buňka je krevní destička. Ostatní buňky jsou červené krvinky. [36]

Pro trénování bylo použito 150 snímků a validace byla prováděna na 60 snímcích. Z důvodu většího počtu snímků bylo trénování časově náročnější než na datasetu s kardiomyocyty. Natrénovat jednu epochu trvalo přibližně 12 minut. V 35. epoše se dosáhlo nejvyšší celkové přesnosti validačního datasetu 0,86 mAP. Při dosažení nejlepších výsledků byla velikost dávky byla šest a počet kroků byl 25.

6.2 Vytvoření GUI

Pro naprogramování grafického uživatelského prostředí (*graphical user inferface*, GUI) v jazyce Python existují různé knihovny, například *Tkinter, Kivy*, nebo *PyQT*. Právě poslední zmíněná byla použita v této úloze. Aktuální verze *PyQT5 5.9.2* se řadí mezi populární knihovny s rozsáhlým množstvím použitelných prvků a aktivní uživatelskou podporou. Dále byl použit nástroj *Qt designer*, který slouží pro vytvoření GUI v prostředí

editoru a usnadňuje tak návrh a úpravy prvků. Výstupem z nástroje *Qt designer* je soubor ve formátu *.ui*, se kterým dále pracuje vlastní program v prostředí *Pycharm*.

Jedním z požadavků bylo, aby aplikace byla samostatně spustitelná. Aby uživatel nemusel k aplikaci přistupovat skrze vývojové prostředí, či spouštěním skriptu. K tomu byl použit nástroj *pyInstaller* [37]. Jedná se o kompilátor, sloužící k vytváření samostatně spustitelných *standalone* aplikací s koncovkou .exe. Nástroj projde Python skript a všechny potřebné moduly a knihovny shromáždí do jedné složky, tak aby aplikace šla spustit bez dalších zásahů.

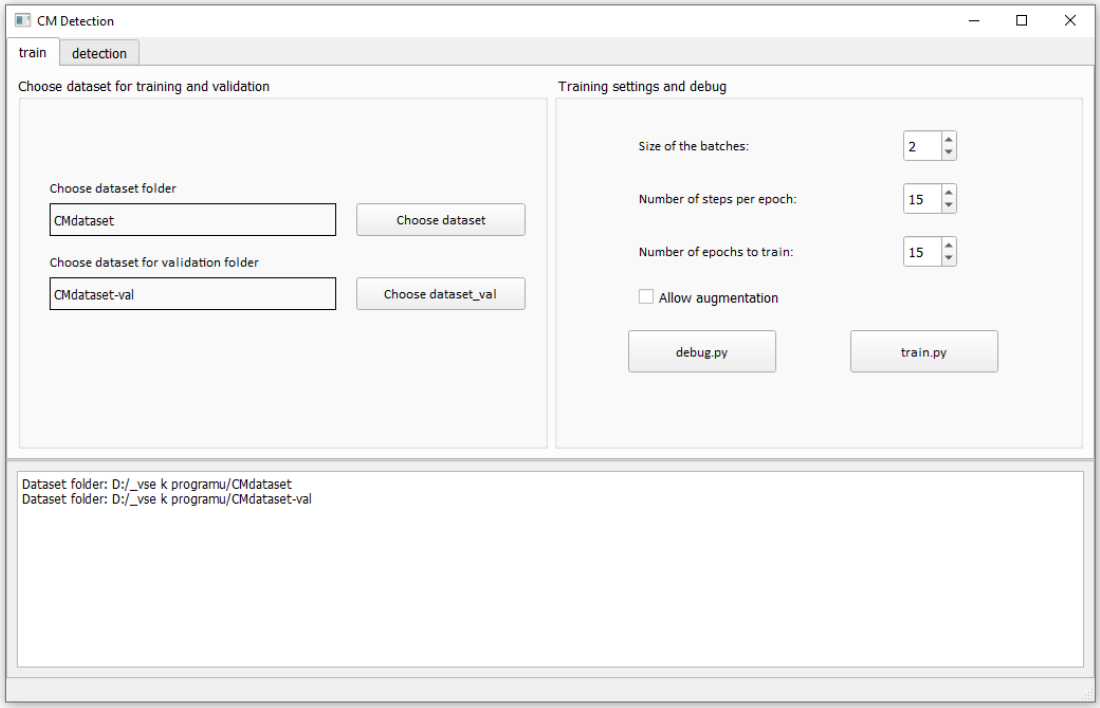
Byla použita nejnovější dostupná verze *pyInstaller 5.0.1*, která podporuje Python verzi 3.6 a vyšší. Program je schopný vytvářet aplikace jak pro operační systém Windows 8 a vyšší, tak pro Linux i Mac OS. Omezením je, že nástroj není vhodný pro převádění mezi různými platformami, tedy pro vytvoření aplikace pro Windows je potřeba použít *pyInstaller* také na Windows.

6.3 Trénovací část

V trénovací části na obrázku 32 je na levé polovině potřeba vybrat složku s trénovacím datasetem pomocí tlačítka *Choose dataset* a složku s validačním datasetem tlačítkem *Choose dataset_val*. Po kliknutí na tlačítko se otevře dialogové okno s aktuálním adresářem a uživatel je vyzván k vybrání složky. Název vybrané složky se zobrazí v rámečku a cesta k vybrané složce se zobrazí v dolní části okna, kde se zobrazují všechny důležité informace.

V pravé části okna uživatel může spustit tlačítkem *debug.py* odzkoušení, zda dataset je správně označen a případné nedostatky datasetu před trénováním opravit. Pohyb mezi jednotlivými snímky je zajištěn klávesovými šipkami doleva a doprava a vyskakovací okno se zavře tlačítkem *Esc*. Dále se v pravé části nastavují parametry pro trénování sítě. Nastavení *Size of batches* a *Number of steps per epoch* závisí na velikosti datasetu, kde počet snímků se rovná počtu kroků (*steps*) děleno velikostí dávky (*batch size*). Další parametr je počet epoch, *Number of epochs to train*, který nastavuje počet iterací, po které se bude model učit. A naposledy si uživatel může zvolit, zda při trénování bude použita augmentace či nikoliv. I přesto, že se s augmentací na datasetu s kardiomyocyty nedosáhlo zlepšení, v jiných případech by augmentace mohla být nápomocná, proto zde byla ponechána možnost augmentaci si zvolit.

V dolní části aplikace je informační okno, ve kterém se zobrazují základní informace pro kontrolu uživatelem, zda vybral žádané soubory. Dále se zde zobrazují základní chybové hlášky. Například uživateli není dovoleno, aby spustil trénování, aniž by vybral trénovací i validační dataset. Informační okno se společné jak pro trénovací, tak i pro detekční část aplikace.



Obrázek 32: Okno aplikace v trénovací části

Poté je zde tlačítko *train.py*, po jeho kliknutí se spustí trénování. Před spuštěním se otevře okno se zprávou, zda uživatel chce opravdu začít trénovat, aby nedocházelo k nechtěnému předčasnému překliknutí a samovolnému spuštění. Trénování lze spustit pouze tehdy, pokud jsou vybrány datasety pro trénování i validaci.

Součástí aplikace je i informační okno příkazového řádku, kde se v reálném čase zobrazuje průběh trénování se všemi náležitostmi. V příkazovém okně se zobrazují všechny informace, které by pro uživatele nebyly důležité a nemusely by být zobrazovány, ale zároveň, díky tomu má uživatel možnost sledovat průběh trénování v reálném čase, a proto je okno příkazového řádku ponecháno. Na obrázku 33 je jako příklad zobrazen průběh 14. epochy, u kterého je zobrazena průběžná validační přesnost mAP, doba trvání i další parametry vyskytující se během trénování.

```
Epoch 14/60
Running network: 100% (3 of 3) |##########| Elapsed Time: 0:00:01 Time:  0:00:01
Parsing annotations: 100% (3 of 3) |######| Elapsed Time: 0:00:01 Time:  0:00:01
13 instances of class adult CM with average precision: 0.7214
mAP: 0.7214

Epoch 14: saving model to ./snapshots/resnet50_csv_14.h5
15/15 [==============================] - 16s 1s/step -
loss: 1.7214 - regression_loss: 1.5055 - classification_loss: 0.2159 - mAP: 0.7214 - lr: 1.0000e-05
```

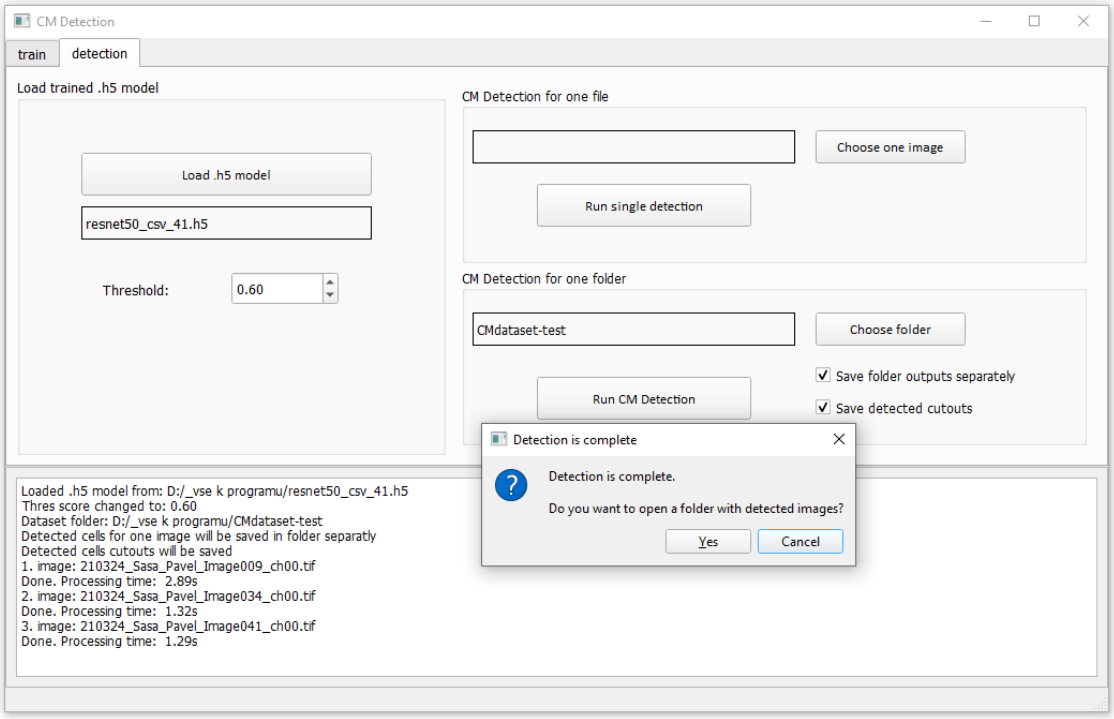
Obrázek 33: Zobrazení průběhu trénování 14. epochy

Výsledkem trénování je natrénovaný model sítě ve formátu *.h5*, který je vhodný pro následnou detekci. Model se trénuje postupně a po každé epoše je uložen v aktuálním adresáři ve složce */snapshots* ve tvaru *resnet50_csv_čísloEpochy.h5*, kde resnet50, představuje použitou páteřní síť. Uživatel má možnost z uložených souborů vybrat ten, který bude nejvíce vhodný pro jeho aplikaci. Je potřeba pamatovat na dostatek místa v aktuálním adresáři, protože Průběžné ukládání natrénovaných modelů bere relativně mnoho místa. Po konci natrénování a výběru vhodné sítě je možné ostatní průběžné modely odstranit.

6.4 Detekční část

V detekční části, která je zobrazena na obrázku 34 uživatel pomocí tlačítka *load .h5 model* nahraje model natrénované sítě v podporovaném formátu *.h5*. Uživatel má možnost vizuálně zkontrolovat výsledek detekce natrénované sítě na jednom snímku. K tomu slouží tlačítko *Choose one image*, které uživatele vyzve k nahrání snímku a tlačítko *Run single detection*, které uloží obraz s detekovanými buňkami. Uživatel si nastavováním hodnoty *Threshold score* může určit hranici, práh spolehlivosti pro zobrazení detekovaného objektu. Číslo mezi 0 a 1 odpovídá procentům a čím vyšší číslo, tím méně objektů bude pravděpodobně nalezeno a zobrazeno.

Pokud je uživatel se zobrazeným výstupem spokojen, může spustit detekci pro celou složku snímků. K tomu slouží tlačítka v druhé části. Uživatel si vybere složku tlačítkem *Choose folder*, program bude pracovat se všemi snímky, které jsou ve správném formátu *.tif*, nebo *.jpg*. Před spuštěním detekce má uživatel možnost si zvolit dvě možnosti. Pokud uživatel zaškrtne *Save folder outputs separately*, pro každý snímek bude vytvořena vlastní složka obsahující samostatný .csv soubor. Při zaškrtnutí druhého políčka *Save detected cutouts* se navíc uloží do složky výřezy nalezených buněk. Tuto možnost není nutné využívat vždy, proto má uživatel možnost volby, aby nepoužité snímky nezabíraly místo na disku. Tlačítkem *Run CM Detection* se spustí detekce. Detekce lze spustit pouze tehdy, pokud je vybraný předtrénovaný model a složka obsahující snímky k detekování. Operace detekce už zabere nějaký čas, záleží na počtu snímků, které je potřeba detekovat a jak je počítač výkonný. Při testování na běžném notebooku bez GPU se jedná přibližně o 2 vteřiny na snímek.



Obrázek 34: Okno aplikace v detekční části po dokončené detekci na složce CMDataset-test

Po skončení detekce je otevřeno vyskakovací okno. Zde je uživatel informován o dokončené detekci a je dotázán, zda si přeji otevřít umístění složky s výslednou detekcí. Díky tomu má uživatel možnost zkontrolovat výsledek detekce.

7 ZHODNOCENÍ A DISKUZE

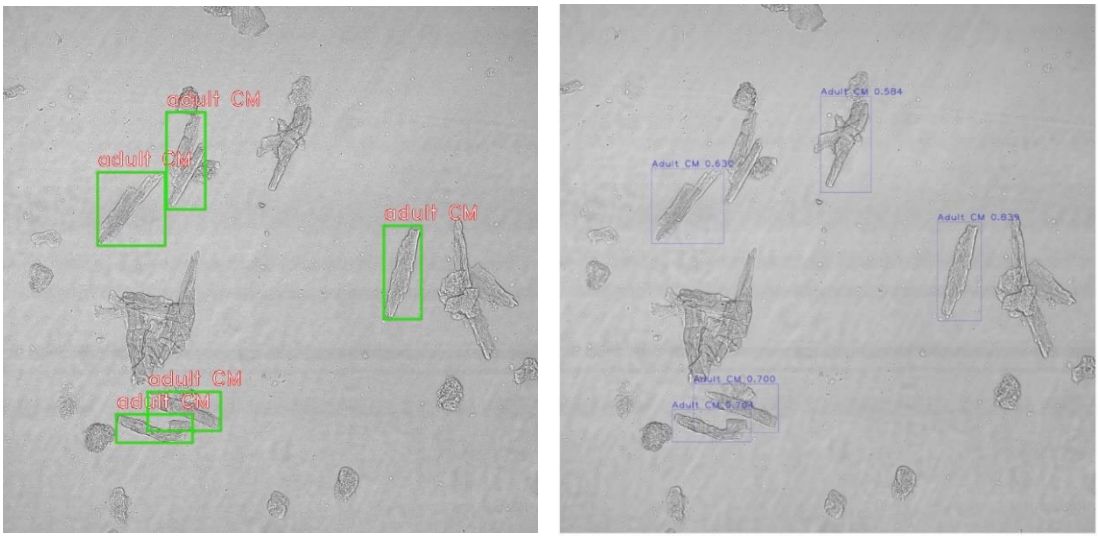
Vyhodnocení detekce kardiomyocytů

Několik významnějších běhů je zobrazeno v tabulce na obrázku 35. Nejlepších výsledků průměrné dosažené přesnosti mAP i klasifikační ztráty na validačním datasetu se dosáhlo v 15. epoše při velikosti dávky dvě a počtu kroků 15 a to s využitím přenosového učení. Proto ve výsledné aplikaci probíhá trénování s přenosovým učením a nastavení ostatních parametrů je ponecháno na uživateli podle velikosti datasetu.

batch size	steps	přenosové učení	epochy	nejlepší epocha	mAP	class_loss
1	30	NE	50	28	0,86	0,69
2	15	NE	60	35	0,85	0,89
3	10	NE	43	29	0,82	0,99
1	30	ANO	40	10	0,92	0,6
3	10	ANO	50	15	0,95	0,62
2	**15**	**ANO**	**45**	**15**	**0,96**	**0,52**

Obrázek 35: Výsledky několika významnějších běhů. Poslední řádek obsahuje trénování s nejlepším výsledkem

Validační dataset obsahuje 13 označených buněk kardiomyocytů. Na obrázku 36 je porovnání jednoho snímku s označenými anotacemi (vlevo) s výstupem z detekční aplikace (vpravo). V tomto případě jsou správně detekovány čtyři buňky jako skutečně pozitivní, jedna buňka je označena špatně jako falešně pozitivní a jedna buňka je falešně negativní, protože není detekována, přestože by měla být.



Obrázek 36: Porovnání snímku s označenými anotacemi (vlevo) a snímku s detekovanými buňkami (vpravo)

V celém validačním datasetu je detekováno 12 buněk se spolehlivostí vyšší než 0,5. Z toho je 11 buněk detekováno správně a jedna buňka je označena jako falešně pozitivní i přesto, že je částečně deformovaná, a správně by neměla být nalezena. Dále dvě buňky jsou označeny jako falešně negativní, protože jsou v těsné blízkosti s nevhodnými buňkami, a proto nebyly nalezeny. Detekci tedy unikla jedna buňka a dvě buňky byly označeny navíc. Pro výzkumníky není významný problém, pokud nebudou nalezeny všechny buňky, které by měly být nalezeny. Pokud bude ale nalezeno co nejméně falešně pozitivnách buněk, které by prodlužovali výslednou analýzu. Pro tento

přístup lze zvýšit práh spolehlivosti *Threshold* a výsledkem bude méně objektů označených jako buňka.

Při použití augmentace se nedosahovalo lepších výsledků a ztráta se nebyla schopná během trénování ustálit a neklesla pod jedna. Na jiném datasetu, který by obsahoval více snímků by augmentace mohla být přínosná a mohlo by se dosáhnout přesnějších výsledků. Proto ve finální aplikaci má uživatel možnost si zvolit, zda při trénování využije augmentaci či nikoliv. Důvodem, proč nedochází k výraznému zlepšení, může být právě velmi malá velikost datasetu, kde dochází k výrazným rozdílům.

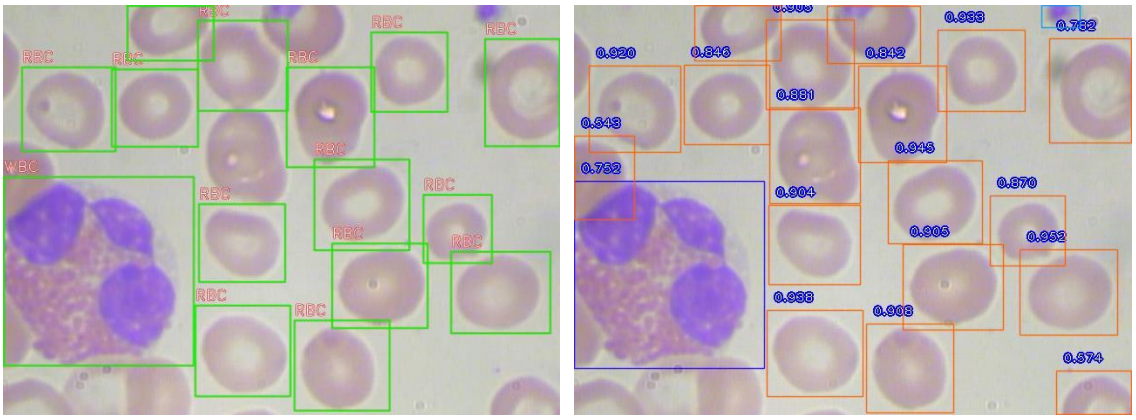
Vyhodnocení trénování na vlastních datech

Bylo nutné ověřit, zda aplikace lze trénovat i na jiných označených datech než na snímcích s kardiomyocyty. Testování bylo zaměřeno více na samotnou funkcionalitu procesu trénování než na snahu dosáhnout nejlepších výsledků. Během experimentů se dosáhlo nejlepších výsledků během 35. epochy. Při nejúspěšnějším trénování velikost dávky byla šest a počet kroků byl 25 a bylo aplikováno přenosové učení. Na obrázku 37 je celková přesnost i přesnosti pro jednotlivé druhy buněk v porovnání s přesností jednotlivých buněk. Autoři [36] tohoto datasetu byli zaměřeni na dosáhnutí co nejvyšší přesnosti na tomto konkrétním datasetu a v některých případech s různým nastavením a při použití různých detekčních architektur se autorům povedl dosáhnout mírně lepších výsledků detekce.

typ buňky	počet	dosažená AP	AP podle [36]
bílé krvinky	63	0,95	86,89
krevní destičky	49	0,85	96,36
červené krvinky	700	0,79	96,09
celková přesnost mAP:	812	**0,86**	

Obrázek 37: Vlastní výsledky validace datasetu CBC ve 35. epoše z vytvořené aplikace

Na obrázku 38 je porovnání anotačního snímku a výsledku detekce. Anotační snímky v datasetu neobsahují anotace krajních buněk, a proto v detekovaném snímku je nalezeno více buněk než v anotačním snímku. Konkrétně levý snímek obsahuje 14 označených buněk červených krvinek, přičemž na pravém snímku bylo nalezeno o čtyři červené krvinky více. Navíc je na horním okraji nalezena jedna krevní destička, které nebyla anotována. Tyto navíc nalezené buňky jsou brány jako falešně pozitivní a zhoršují celkový výsledek sítě.



Obrázek 38: Porovnání anotací pro trénování (vlevo) a výsledné detekce (vpravo).

Během trénování se dosáhlo mírně horších výsledků, nicméně výsledky se dají považovat stále za úspěšné s přihlédnutím na možnou univerzálnost aplikace. Obecně platí, že pro zaručení správného výsledku, nelze zvolit jedno identické nastavení pro různé datasety. Proto má uživatel možnost nastavovat alespoň základní parametry před trénováním a testováním různých nastavení je schopný nalézt nejvhodnější řešení daného problému.

V průběhu řešení diplomové práce bylo nutné zaměřit se na několik menších problémů. Dataset s kardiomyocyty obsahuje velmi malé množství snímků pro trénování a následnou validaci, a proto reálnou přesnost bude možné výzkumníky ověřit až se získáním většího množství snímků vhodných pro trénování. Díky natrénování sítě na rozsáhlejším datasetu, který obsahoval jiná data, bylo možné porovnat dosažené výsledky s výsledky autorů datasetu. Podle toho lze očekávat, že dosažené přesnosti trénování jsou dostatečné pro úspěšnou detekci. Vždy bude záležet na kvalitě a velikosti označeného datasetu.

Při testování aplikace několika uživateli byly postupně opravovány drobné nalezené nedostatky. Pro spuštění trénovací části je nutný nainstalovaný python, protože se volají skripty *train.py* a *debug.py*, které pro svůj běh jazyk potřebují. I přes použitý kompilátor je před prvním spuštěním potřeba doinstalovat podle návodu některé knihovny bez kterých jsou hlášeny chyby. Při dalším spuštění již aplikace běží bez dalších zásahů.

Pokud se aplikace použije na detekci jednoho druhu objektu, nenastává žádný problém. Při detekování více tříd objektů se v označeném přehledovém snímku nezobrazují jednotlivé třídy správně. Zde se nepodařilo spárovat index třídy s jejími názvy a nalezené objekty tříd jsou odlišeny pouze jinou barvou rámečku. Jedná se pouze o vizuální problém zobrazení a indexy třídy jsou správně uloženy do souboru a pro následnou práci se se souborem se nejedná o žádnou komplikaci.

Pro další přínos výzkumníkům je potřeba na vzniklý program navázat s další prací. Vytvořený program této diplomové práce je dílčím řešením, kde výstupem jsou souřadnice nalezených objektů v přehledovém snímku. Následně se tyto souřadnice musí dále předat konfokálnímu mikroskopu, aby mohly vzniknout zmíněné kvalitnější snímky přiblížených buněk.

8 ZÁVĚR

Výstupem diplomové práce je funkční aplikace, která bude nápomocná výzkumníkům a ušetří jejich čas při zpracování velkého množství dat. Vytvořená aplikace je schopná po načtení natrénovaného modelu detekovat buňky ve snímcích a ukládat jejich souřadnice do souboru. Druhou hlavní funkcí je možnost trénování modelu na vlastních datech.

Vytvořená aplikace je naprogramována pomocí jazyku Python a grafické uživatelské prostředí bylo vytvořeno za pomoci nástroje Qt Designer s knihovnou PyQT5. Z rešeržní části byla vybrána konvoluční neuronová síť RetinaNet v implementaci od firmy Fyzir, která je vhodnou architekturou pro potřeby aplikace. Nejlepších výsledků detekce se dosáhlo při trénování s přenosovým učením bez augmentace. Schopnost zvolené architektury obstát v úlohách detekce byla otevřena i na rozsáhlejším datasetu obsahujícím krevní buňky. Oproti výsledkům autorů datasetu bylo dosaženo mírně horších výsledků, celková úspěšnost detekce s přihlédnutím na univerzálnost aplikace je ale dostatečná.

Aplikace je samostatně spustitelná a její fungování je ověřeno na operačním systému Windows. Aplikace obsahuje pouze nezbytné části a tlačítka tak, aby byla co nejvíce uživatelsky přívětivá. Například po najetí kurzorem myši na jednotlivá tlačítka se zobrazí krátká slovní nápověda pro připomenutí funkcionality. Podrobný návod k ovládání aplikace a seznam všech nutných prerekvizit k trénování je rozepsán v návodu, který je součástí diplomové práce jako příloha.

Aplikace je také schopná trénovat model na vlastních datech a poté s pomocí natrénovaného modelu detekovat i jiné typy buňky. Díky možnosti trénování je aplikace univerzální a je vhodná pro použití i v jiných detekčních úlohách. Model se trénuje přímo v aplikaci, s využitím výkonu počítače, proto trénování může být časově náročné. Průběh trénování je zobrazován v reálném čase v přidruženém okně příkazového řádku a uživatel vidí, v jaké části se trénování nachází a jakých se dosahuje průběžných výsledků. V detekční části aplikace je možné spustit detekci, kde si uživatel volí práh, s jakou přesností bude aplikace buňky detekovat. Hlavním výstupem jsou uložené souřadnice detekovaných objektů, které jsou vhodné pro následné zpracování. Vzniklá aplikace je plně funkční a připravena k použití výzkumníky ze Slovenské akademie věd.

SEZNAM POUŽITÉ LITERATURY

[1] CHOLLET, François. *Deep learning v jazyku Python: knihovny Keras, Tensorflow* [online]. Praha: Grada Publishing, 2019 [cit. 2021-03-14]. Knihovna programátora (Grada). ISBN ISBN978-80-271-2750-4.

[2] FUMO, David. *A Gentle Introduction To Neural Networks Series* [online]. In: . 2017 [cit. 2021-04-10]. Dostupné z: https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc

[3] VOLNÁ, Eva. *Neuronové sítě 1* [online]. 2. vyd. Ostravská univerzita v Ostravě, 2008 [cit. 2021-03-25]. Dostupné z: https://web.osu.cz/~Volna/Neuronove_site_skripta.pdf

[4] SAEED, Mehreen. A Gentle Introduction To Sigmoid Function. In: *Machine learning mastery* [online]. [cit. 2022-04-20]. Dostupné z: https://machinelearningmastery.com/a-gentle-introduction-to-sigmoid-function/

[5] WOOD, Thomas. What is the Softmax Function?. In: *Deep AI* [online]. 2019 [cit. 2022-04-20]. Dostupné z: https://deepai.org/machine-learning-glossary-and-terms/softmax-layer

[6] DA SILVA, Ivan, Danilo HERNANE SPATTI, Rogerio ANDRADE FLAUZINO, Luisa LIBONI a Silas DOS REIS ALVES. *Artificial Neural Networks*. Cham: Springer International Publishing AG, 2016. ISBN 3319431617. Dostupné z: doi:10.1007/978-3-319-43162-8

[7] HAYKIN, Simon. *Neural networks: a comprehensive foundation*. 2nd ed. Upper Saddle River: Prentice Hall, 1999. ISBN 01-327-3350-1.

[8] KRIZHEVSKY, Alex, Ilya SUTSKEVER a Geoffrey HINTON. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*. 2017, **60**(6), 84-90. ISSN 0001-0782. Dostupné z: doi:10.1145/3065386

[9] BROWNLEE, Jason. A Gentle Introduction to Pooling Layers for Convolutional Neural Networks. In: *Machine learning mastery* [online]. 2019 [cit. 2021-04-21]. Dostupné z: https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/

[10] ZHAO, Zhong-Qiu, Peng ZHENG, Shou-Tao XU a Xindong WU. Object Detection With Deep Learning: A Review. *IEEE Transactions on Neural Networks and Learning Systems*. 2019, **30**(11), 3212-3232. ISSN 2162-237X. Dostupné z: doi:10.1109/TNNLS.2018.2876865

[11] Instance vs. Semantic Segmentation. In: *Keymakr* [online]. Keymakr Inc. [cit. 2022-03-20]. Dostupné z: https://keymakr.com/blog/instance-vs-semantic-segmentation/

[12] KUMAR, Ajitesh. Machine Learning: Training, Validation & Test Data Set. In: *Data Analytics* [online]. [cit. 2022-05-10]. Dostupné z: https://vitalflux.com/machine-learning-training-validation-test-data-set/

[13] SHARMA, Sagar. Epoch vs Batch Size vs Iterations. In: *Towards Data Science* [online]. [cit. 2022-03-20]. Dostupné z: https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9

[14] SAXENA, Sharoon. Underfitting vs. Overfitting (vs. Best Fitting) in Machine Learning. In: *Analytic Vidhya* [online]. 2020 [cit. 2021-04-16]. Dostupné z: https://www.analyticsvidhya.com/blog/2020/02/underfitting-overfitting-best-fitting-machine-learning/

[15] AGARWAL, Shivang, Jean TERRAIL a Frédéric JURIE. *Recent Advances in Object Detection in the Age of Deep Convolutional Neural Networks* [online]. In: . 2019. 2019 [cit. 2022-05-20]. ffhal-01869779v2f.

[16] *ImageNet Large Scale Visual Recognition Challenge (ILSVRC)* [online]. Stanford University: Stanford Vision Lab, 2020 [cit. 2022-03-08]. Dostupné z: https://image-net.org/challenges/LSVRC/

[17] HE, Kaiming, Xiangyu ZHANG, Shaoqing REN a Jian SUN. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, 770-778. ISBN 978-1-4673-8851-1. Dostupné z: doi:10.1109/CVPR.2016.90

[18] LECUN, Y., L. BOTTOU, Y. BENGIO a P. HAFFNER. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*. 1998, **86**(11), 2278-2324. ISSN 00189219. Dostupné z: doi:10.1109/5.726791

[19] WANG, Chi-Feng. The Vanishing Gradient Problem. In: *Towards Data Science* [online]. [cit. 2022-03-13]. Dostupné z: https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484

[20] GIRSHICK, Ross, Jeff DONAHUE, Trevor DARRELL a Jitendra MALIK. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition* [online]. IEEE, 2014, 580-587 [cit. 2022-03-14]. ISBN 978-1-4799-5118-5. Dostupné z: doi:10.1109/CVPR.2014.81

[21] GANDHI, Rohith. R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms. In: *Towards Data Science* [online]. [cit. 2022-03-14]. Dostupné z: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

[22] GIRSHICK, Ross. Fast R-CNN. *2015 IEEE International Conference on Computer Vision (ICCV)* [online]. IEEE, 2015, 1440-1448 [cit. 2022-03-14]. ISBN 978-1-4673-8391-2. Dostupné z: doi:10.1109/ICCV.2015.169

[23] REN, Shaoqing, Kaiming HE, Ross GIRSHICK a Jian SUN. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* [online]. 2017, **39**(6), 1137-1149 [cit. 2022-03-14]. ISSN 0162-8828. Dostupné z: doi:10.1109/TPAMI.2016.2577031

[24] LIU, Wei, Dragomir ANGUELOV, Dumitru ERHAN, Christian SZEGEDY, Scott REED, Cheng-Yang FU a Alexander BERG. SSD: Single Shot MultiBox Detector. *Computer Vision – ECCV 2016* [online]. Cham: Springer International Publishing, 2016, 21-37 [cit. 2022-03-14]. Lecture Notes in Computer Science. ISBN 978-3-319-46447-3. Dostupné z: doi:10.1007/978-3-319-46448-0_2

[25] HUI, Jonathan. SSD object detection: Single Shot MultiBox Detector for real-time processing. In: *Medium* [online]. 2018 [cit. 2022-03-10]. Dostupné z: https://jonathan-hui.medium.com/ssd-object-detection-single-shot-multibox-detector-for-real-time-processing-9bd8deac0e06

[26] REDMON, Joseph, Santosh DIVVALA, Ross GIRSHICK a Ali FARHADI. You Only Look Once: Unified, Real-Time Object Detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* [online]. IEEE, 2016, 779-788 [cit. 2022-04-21]. ISBN 978-1-4673-8851-1. Dostupné z: doi:10.1109/CVPR.2016.91

[27] Real-Time Object Detection on COCO. In: *Papers with code* [online]. 2021 [cit. 2022-04-21]. Dostupné z: https://paperswithcode.com/sota/real-time-object-detection-on-coco

[28] LIN, Tsung-Yi, Priya GOYAL, Ross GIRSHICK, Kaiming HE a Piotr DOLLAR. *Focal Loss for Dense Object Detection* [online]. [cit. 2022-03-15]. Dostupné z: doi:10.1109/TPAMI.2018.2858826

[29] TSANG, Sik-Ho. Review: RetinaNet — Focal Loss (Object Detection). In: *Towards Data Science* [online]. 2018 [cit. 2022-03-15]. Dostupné z: https://towardsdatascience.com/review-retinanet-focal-loss-object-detection-38fba6afabe4

[30] WOODCOCK, Elizabeth a Scot MATKOVICH. *Cardiomyocytes structure, function and associated pathologies* [online]. 2005, **37**(9), 1746-1751 [cit. 2022-05-09]. ISSN 13572725. Dostupné z: doi:10.1016/j.biocel.2005.04.011

[31] *LabelImg* [online]. GitHub, 2016 [cit. 2021-04-24]. Dostupné z: https://github.com/tzutalin/labelImg

[32] HUI, Jonathan. Object detection: speed and accuracy comparison: (Faster R-CNN, R-FCN, SSD, FPN, RetinaNet and YOLOv3). In: *Medium* [online]. [cit. 2022-03-16]. Dostupné z: https://jonathan-hui.medium.com/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359

[33] *Implementace Fyzir/keras-retinanet* [online]. 2019 [cit. 2021-04-24]. Dostupné z: https://github.com/fizyr/keras-retinanet

[34] *Google Colab* [online]. In: . [cit. 2021-05-02]. Dostupné z: https://research.google.com/colaboratory/faq.html

[35] *CUDA Toolkit* [online]. In: . NVIDIA [cit. 2021-05-02]. Dostupné z: https://developer.nvidia.com/cuda-toolkit

[36] ALAM, Mohammad a Mohammad ISLAM. Machine learning approach of automatic identification and counting of blood cells. *Healthcare Technology Letters* [online]. 2019, **6**(4), 103-108 [cit. 2022-04-27]. ISSN 2053-3713. Dostupné z: doi:10.1049/htl.2018.5098

[37] CORTESI, David. *PyInstaller: 5.0.1* [online]. In: . [cit. 2022-04-13]. Dostupné z: https://pyinstaller.org/

[38] BLAHA, Milan. Koncept umělé neuronové sítě. In: *Matematická biologie* [online]. Lékařská fakulta Masarykovy univerzity [cit. 2022-03-01]. Dostupné z: https://portal.matematickabiologie.cz/index.php?pg=analyza-a-hodnoceni-biologickych-dat--umela-inteligence--neuronove-site-jednotlivy-neuron--uvod-do-neuronovych-siti--koncept-umele-neuronove-site

[39] GU, Hao. Basic architecture CNN. In: *Research gate* [online]. 2019 [cit. 2022-03-20]. Dostupné z: https://www.researchgate.net/figure/Basic-architecture-of-CNN_fig3_335086346

[40] KANG, Dae-Young, Hieu DUONG a Jung-Chul PARK. Application of Deep Learning in Dentistry and Implantology. *The Korean Academy of Oral and Maxillofacial Implantology*. 2020, **24**(3), 148-181. ISSN 1229-5418. Dostupné z: doi:10.32542/implantology.202015

[41] KANG, Dae-Young. *Winners of ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2010-2016* [online]. In: . 2020 [cit. 2022-03-08]. Dostupné z: https://www.researchgate.net/figure/Algorithms-that-won-the-ImageNet-Large-Scale-Visual-Recognition-Challenge-ILSVRC-in_fig2_346091812

[42] LI, Fei-Fei, Justin JOHNSON a Serena YEUNG. *CNN Architectures*. Stanford University, 2019. Dostupné také z: http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture09.pdf

[43] ZHANG, Chao. *Joint Training Methods for Tandem and Hybrid Speech Recognition Systems using Deep Neural Networks* [online]. Cambridge, 2017 [cit. 2022-04-18]. Dostupné z: https://www.researchgate.net/publication/320402742_Joint_Training_Methods_for_Tandem_and_Hybrid_Speech_Recognition_Systems_using_Deep_Neural_Networks. Dissertation. University of Cambridge.

SEZNAM OBRÁZKŮ

SEZNAM ZKRATEK

µm	mikrometr
2D	Dvoudimenzionální
3D	trojdimenzionáln
AP	Average Precision
CBC	complete blood count
CE	Cross Entropy
CM	Cardiomyocytes
CNN, ConvNet	Convolutional neural network
COCO	Common Objects In Context
CPU	Central Processing Unit
Esc	Escape
FL	Focal Loss
FPN	Feature Pyramid Network
GPU	Graphics Processing Unit
GUI	Graphic User Interface
IoU	Intersection over Union
mAP	mean Average Precision
MB	megabyte
R-CNN	Region Based Convolutional Neural Networks
ReLu	Rectified Linear Unit
ResNet	Residual neural network
SAV	Slovenská Akademie Věd
SSD	Single Shot MultiBox Detector
SVM	Support vector machines
YOLO	You Only Look Once

SEZNAM PŘÍLOH

Struktura přílohy DP_Hubalek_Michal_prilohy.zip
Příloha Návod k ovládání
DP_Hubalek_Michal_prilohy.zip

STRUKTURA PŘÍLOHY .ZIP

DP_Hubalek_Michal_prilohy.zip

- Source_appCMDetection
    - Keras-retinanet-main – složka obsahující implementaci RetinaNet
    - App_CMDetection.py – hlavní skript
    - GUI_CMDetection.ui – nutný soubor pro zobrazení GUI
    - Readme.txt – bodový návod
    - Requierements.txt – seznam prerekvizit k instalaci podle návodu
- Návod k ovládání.pdf

PŘÍLOHA NÁVOD K OVLÁDÁNÍ

Použití editoru LabelImg

Editor LabelImg slouží pro označování datasetu pro trénování.

Instalace LabelImg

Nutno stáhnout a nainstalovat Python verze 3.7.3

https://www.python.org/downloads/release/python-373/

1 Aktuální verze pythonu lze zkontrolovat v příkazovém řádku pomocí příkazu:

```
python --version
```

2 Stáhnout zdrojový kód pro labelImg z https://github.com/tzutalin/labelImg do zvoleného adresáře (např. d:\lableImg)

3 V příkazovém řádku (cmd) zadat následující příkazy (instalace knihoven):

```
pip install PyQt5
pip install lxml
```

přejít do adresáře s editorem, pro d:\lableImg zadej příkazy:

```
d:
cd labelImg
pyrcc5 -o libs/resources.py resources.qrc
```

Spuštění LabelImg

Editor se pouští z příkazového řádku (cmd) příkazem:

```
python labelImg.py
```

Je nutné být přepnutý v adresáři, kde se nachází zdrojový kód editoru (v našem příkladu to je d:\labelImg).

Ovládání LabelImg

Po spuštění se otevře přehledné okno. Složka, která se chce označit se vybere pomocí tlačítka **Open Dir**. Šipky **Next Image** a **Prev Image** jsou pro procházení snímků. Formát souřadnic má být nastaven na **Pascal/VOC**. Tlačítkem **Create RectBox** uživatel může vytvářet ohraničující obdélníky. Po vytvoří se obdélníku přiřadí název třídy a všechny objekty jsou zobrazeny v tabulce vpravo. Tímto způsobem se označí celý dataset. Takto označený dataset je vhodný pro následné trénování v trénovací části aplikace CMDetection.

Vytvoření .exe pomocí nástroje PyInstaller

Pro SAV byla vytvořena samostatně spustitelná aplikace, která není součástí příloh. Přílohou je skript app_CMDetection.py, ze kterého je možné spustitelnou aplikaci vytvořit. Pokud lze aplikace spustit v prostředí python, lze vytvořit .exe:

```
pip install pyinstaller
pyinstaller path_to/app_CMDetection.py --onedir –clean
```

Do vytvořené složky se musí přesunout soubory:

`Snapshots/_pretrained_model.h5` (pro trénování lze stáhnout zde. Je nutné přesunout a přejmenovat)

`GUI_CMDetection.ui`

`Složka keras-retinanet-main`

Prvotní instalace CMDetection

Pro trénování je nutné nainstalovat python 3.7.3
(https://www.python.org/downloads/release/python-373/)
A nainstalovat prerekvizity v příkazovém řádku (cmd):

```
python -m install pip –upgrade pip
```

`pip install -r requirements.txt` (být ve stejném adresáři jako requirements.txt pomocí příkazu cd)
Případně doinstalovat další knihovny podle chybových hlášek konkrétního počítače

Spuštění aplikace CMDetection

- Po stažení rozbalit .zip soubor do složky s dostatečným místem.
- Není doporučena jakákoliv manipulace se složky a soubory.
- Aplikace se spustí pomocí zástupce CMDetection.
- Spolu s aplikací se otevře i informativní příkazový řádek, kde se zobrazují informace z aplikace.
- Před spuštěním trénovací části je nutná instalace několika kroků.

Popis aplikace CMDetection

Trénovací část

Pro trénování je potřeba:

- Označený dataset (LabelImg) rozdělený na trénovací dataset a validační dataset (cca 90 % a 10 %)
- Nainstalovaný python 3.7.3

- Dostatečný prostor v adresáři pro ukládání průběžně natrénovaných modelů

Po dokončení tréninku jsou průběžně natrénované modely uloženy do složky:

CMDetectionApp/snapshots

Detekční část

Pro detekci je potřeba:

- Natrénovaný model ve formátu .h5
- Složku se snímky, které je žádoucí detekovat.

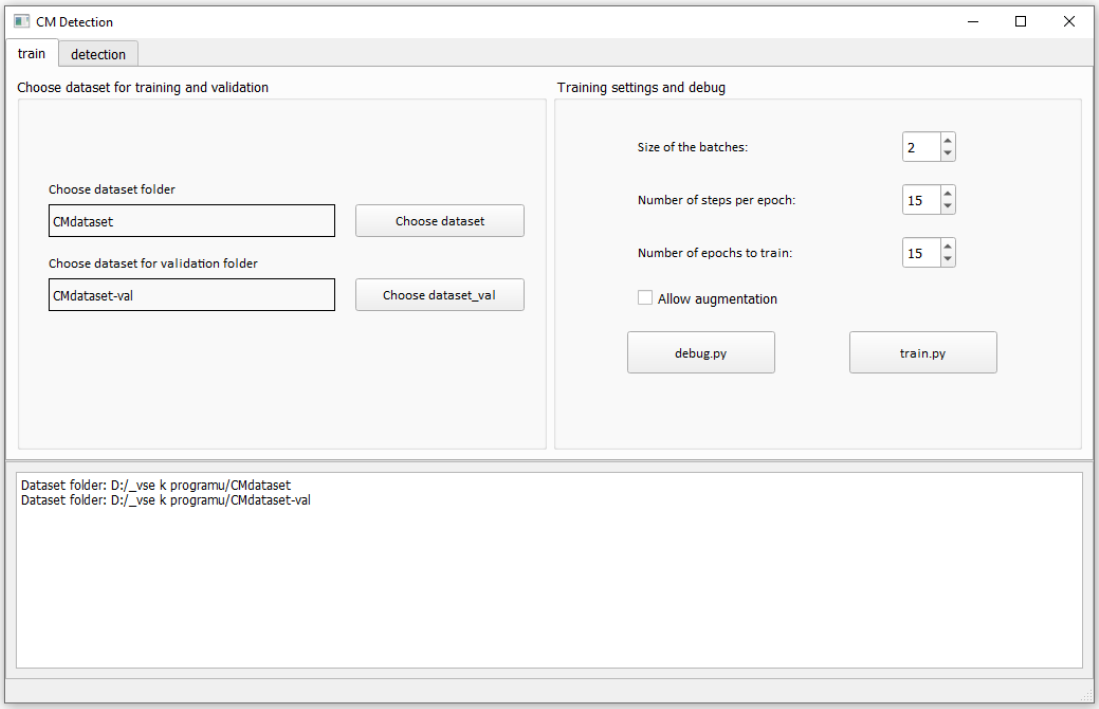
Po dokončení detekce je vytvořená složka s aktuálním časem obsahující souřadnice, přesnost a adresou ke snímku uložena do:

CMDetectionApp/_DETECTED_IMAGES

Ovládání aplikace CMDetection

Při spuštění a používání aplikace se průběžně načítají knihovny, proto v některých případech po kliknutí aplikace reaguje později.

Ovládání trénovací části



Tlačítko **Choose dataset** – Uživatel je vyzván k vybrání složky se snímky, na kterých chce model sítě trénovat. Dataset musí být předem označený pomocí nástroje LabelImg.

Tlačítko **Choose dataset_val** – Uživatel je vyzván k vybrání složky s validačními snímky. Dataset musí být předem označený pomocí nástroje LabelImg.

Tlačítko **debug.py** – Kontrola vstupních dat před započnutím trénování. Zobrazí se snímky s vyznačenými anotacemi, na kterých následně proběhne detekce.

Výběrové pole **Size of batches** – Slouží k nastavení velikosti dávky pro trénování jedné epochy.

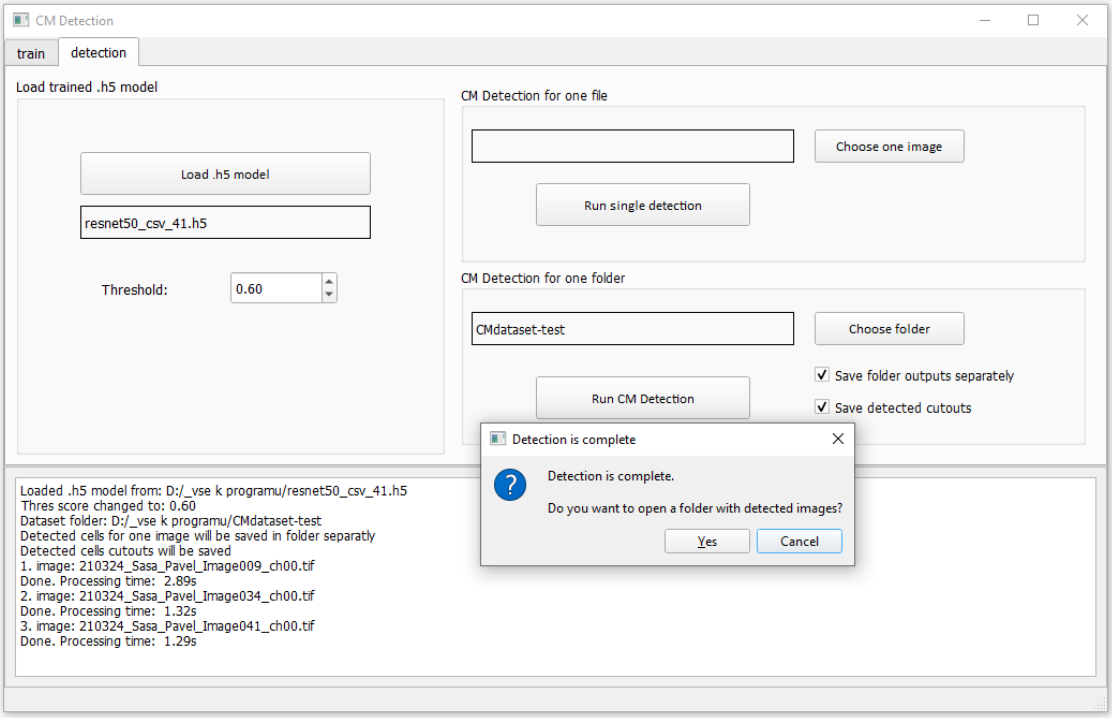
Výběrové pole **Number of steps per epoch** – Slouží k nastavení počtu kroků pro trénování jedné epochy. Součin velikosti dávky a počtu kroků se musí rovnat počtu snímkům v trénovacím datasetu.

Výběrové pole **Number of epochs to train** – Nastavení, po kolik epoch bude probíhat trénování. Počet epoch pro konkrétní dataset je nutné zjistit experimentálně.

Možnost **Allow augmentation** – Možnost, zda bude při trénování využita augmentace datasetu či nikoliv.

Tlačítko **train.py** – Nutný vybraný trénovací i validační dataset. Po kliknutí se zobrazí potvrzovací okno, zda uživatel opravdu chce spustit trénování.

Ovládání detekční části



Tlačítko **Load .h5 model** – Po kliknutí je uživatel vyzván k načtení předučeného modelu s koncovkou .h5.

Výběrové pole **Treshold** – Parametr může nabývat hodnot od nuly až do jedné a představuje práh minimální přesnosti, s jakou budou detekované buňky označeny.

Tlačítko **Choose one Image** – Uživatel je vyzván pro vybrání jednoho snímku s koncovkou .tif nebo .jpg.

Tlačítko **Run single detection** – Pro vybraný snímek je spuštěna detekce. Tato část slouží pro kontrolu výstupu ještě před spuštěním detekce pro celou složku.

Tlačítko **Choose folder** – Uživatel je vyzván k vybrání celé složky obsahující snímky s koncovkou tif nebo .jpg.

Tlačítko **Run CM Detection** – Pro vybranou složku je spuštěna detekce. Po skončení je zobrazena informace o dokončení detekce a dotaz, zda uživatel chce otevřít umístění výsledných souborů.

Možnost **Save folder outpusts separately** – Výstup z detekce bude uložen do jednoho csv souboru. pro každý snímek zvlášť

Možnost **Save detected outputs** – Možnost uložení jednotlivých výřezů ze snímku obsahující detekované buňky.