ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA BIOMEDICÍNSKÉHO INŽENÝRSTVÍ

Katedra biomedicínské informatiky

Segmentace obrazu histologie aterosklerotických plátů

Segmentation of the atherosclerotic plate histology image

Bakalářská práce

Studijní program:	Biomedicínská a klinická technika
Studijní obor:	Biomedicínská informatika

Autor bakalářské práce:	Vojtěch Kropáček
Vedoucí bakalářské práce:	Mgr. Radim Krupička , Ph.D.

Kladno 2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení:	**Kropáček**	Jméno:	**Vojtěch**	Osobní číslo:	**491422**
Fakulta:	**Fakulta biomedicínského inženýrství**				
Garantující katedra:	**Katedra biomedicínské informatiky**				
Studijní program:	**Biomedicínská a klinická technika**				
Studijní obor:	**Biomedicínská informatika**				

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Segmentace obrazu histologie aterosklerotických plátů

Název bakalářské práce anglicky:

Image segmentation of atherosclerotic plaque histology

Pokyny pro vypracování:

Cílem bakalářské práce je s využitím konvolučních neuronových sítí segmentovat obrazy histologie aterosklerotických plátů. V rámci bakalářské práce se seznamte s metodami hlubokého učení a segmentace obrazu a vyberte vhodný způsob segmentace jednotlivých tkání v histologických snímcích. Pro ověření funkčnosti algoritmu připravte ve spolupráci s FNKV trénovací a testovací množinu snímků. Definujte metodiku pro expertní anotaci a nechte anotovat vybranou množinu snímků histology z FNKV, výsledky upravte a nahrajte do systému CVAT. Na závěr zhodnoťte přesnost segmentace jednotlivých tkání.

Seznam doporučené literatury:

[1] Pytorch, Pytorch tutorials, 1.11.2021, https://pytorch.org/tutorials/
[2] Mohit Sewak, Md. Rezaul Karim, Pradeep Pujari, Practical Convolutional Neural Networks, ed. 1, Packt, 2018, ISBN 9781788392303

Jméno a příjmení vedoucí(ho) bakalářské práce:

Mgr. Radim Krupička, Ph.D.

Jméno a příjmení konzultanta(ky) bakalářské práce:

prof. MUDr. Václav Mandys, CSc.

Datum zadání bakalářské práce: **14.02.2022**
Platnost zadání bakalářské práce: **18.09.2023**

doc. Ing. Zoltán Szabó Ph.D.
vedoucí katedry

prof. MUDr. Jozef Rosina, Ph.D., MBA
děkan

PROHLÁŠENÍ

Prohlašuji, že jsem bakalářskou práci s názvem „Segmentace obrazu histologie aterosklerotických plátů“ vypracoval samostatně a použil k tomu úplný výčet citací použitých pramenů, které uvádím v seznamu přiloženém k diplomové práci.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů.

V Kladně 23.4.2022 Vojtěch Kropáček

PODĚKOVÁNÍ

Rád bych poděkoval své přítelkyni, která mi hlavně v posledních dnech psaní mé bakalářské práce dopřávala drahocenný klid, svému vedoucímu bakalářské práce Radimu Krupičkovi za jeho velkou ochotu a strpení a sousedům, kteří se rozhodli každý den v 7.00 začít pracovat na domě, a tudíž jsem místo spánku mohl psát bakalářskou práci

ABSTRAKT

Segmentace obrazu histologie aterosklerotických plátů

Cíle mé bakalářské práce jsou segmentovat obraz histologie aterosklerotických plátů na specifické oblasti (lumen, krvácení, kalcifikace, fibrózní vazivo, ateromová tkáň) a nahrání vytvořených dat do anotačního nástroje CVAT.

K automatické segmentaci byla využita konvoluční neuronovou síť DeepLabV3+. Neuronová síť byla trénována na datasetu obrazů histologií se dvěma barveními, WG(Weigert van Giesonovým) a HE(Hematxylin-eosinovým). Naučená neuronová síť se použila na segmentaci neanotovaných obrazů histologií a výsledky se nahráli do anotačního nástroje CVAT.

Pro nahrání dat do CVATu je potřeba vytvořené segmentace(anotace) převézt z obrazové formy na bodovou strukturu. Pro vyřešení toho problému jsem vytvořil metodiku na převádění obrazů anotací na body, zjednodušil jejich strukturu a vytvořil stromové rozřazování pro ukládání do XML souboru, který přijímá anotační nástroj CVAT. Parametry použité při tvoření neuronové sítě a metody algoritmu vytvoření exportních dat pro CVAT popisuji v metodách a implementaci.

Ve výsledcích práce ukazuju přesnost vytvořené neuronové sítě z hlediska funkcí na měření přesnosti, které byly součástí modelu vytvořené sítě a z hlediska přímé kontroly histologem, kterému bylo část snímků vytisknuto a byly na nich za současné kontroly originálních vzorků pod mikroskopem zaznamenávány nesrovnalosti v segmentaci.

Klíčová slova

Aterosklerotický plát

Segmentace

Neuronová síť

Algoritmus

Anotace

CVAT

ABSTRACT

Histology image segmentation of atherosclerotic plaques

The aim of my bachelor thesis is to segment the image of histology of atherosclerotic plaques into specific areas (lumen, bleeding, calcification, fibrous ligament, atheroma tissue) and upload the generated data to the CVAT annotation tool.

The DeepLabV3 + convolutional neural network was used for automatic segmentation. The neural network was trained on a dataset of histological images with two colors, WG (Weigert van Gieson) and HE (Hematxylin-eosin). The learned neural network was used to segment unannotated histology images and the results were recorded in the CVAT annotation tool.

To upload data to CVAT, it is necessary to transfer the created segmentation (annotation) from the image form to the point structure. To solve this problem, I developed a methodology for converting annotation images to the body, simplified its structure, and created a tree layout for saving to an XML file that the CVAT annotation tool accepts. The parameters used in creating the neural network and the methods of the algorithm for creating export data for CVAT are described in the methods and implementation.

The results chapter show the accuracy of the created neural network in terms of accuracy measurement functions, which were part of the model of the created neural network and in terms of direct control by a histologist, who was printed part of the images

Keywords

Atherosclerotic plaque

Segmentation

Neural network

Algorithm

Annotation

CVAT

Obsah

Seznam zkratek

Seznam zkratek

Zkratka	Význam
CVAT	Computer Vision Annotation Tool
WG	Weigert van Gieson
HE	Hematxylin-eosin
Json	JavaScript Object Notation
XML	Extensible Markup Language
FNKV	Fakultní nemocnice Královské Vinohrady

1 Úvod

Ateroskleróza, je dlouhodobě nejčastější příčinou mrtvice. Cílem projektu naší školy ve spolupráci s FNKV je vytvoření softwaru pro digitální analýzu obrazu ultrazvuku stenózy karotidy a aterosklerotického plátu a posouzení, jestli dochází k riziku prasknutí plátu, což by mohlo mít za následek mrtvici.

Výsledky projektu by měla být analýza aterosklerotického plátu a identifikace jeho stability pomocí kombinace sonografie a digitální analýzy obrazu. Ultrazvuková zobrazovací metoda, má mnoho nedostatků zejména v rozlišení obrazu a jeho kvalitě. Kvůli těmto nedostatkům se rozhodlo, že se použije metoda registrace obrazu a sonografické snímky se budou párovat se snímky ve větší kvalitě (obrazy histologií). To by mělo poskytnout přesnější informace a pomoct správně popsat strukturu ateromových plátů a stav na ultrazvukovém snímku.

Cílem mé bakalářské práce je segmentovat obraz histologie aterosklerotických plátů na specifické oblasti (lumen, krvácení, kalcifikace, fibrózní vazivo, ateromová tkáň) a nahrání vytvořených dat do anotačního nástroje CVAT. K automatizaci segmentace vytvořím konvoluční neuronovou síť. Výsledky naučené neuronové sítě budou posuzovány histologem.

V této práci se soustředím hlavně na přesnost neuronové sítě a vizualizaci výsledků dané sítě. K tomu budu využívat především model na segmentaci obrazu. Naučená síť by pak měla sloužit k zvětšení datasetu se kterým se bude pracovat při párování obrazů ultrazvukových vyšetření a obrazu histologií.

2 Přehled současného stavu

2.1 Ateroskleróza

Ateroskleróza je onemocnění cévní stěn, které je charakteristické hromaděním lipidů, krve, vaziv a tkání na vnitřní straně cévní stěny

2.1.1 Fáze vývoje

Vyvíjí se jako chronický zánět a postupně ucpává cévu, čímž zmenšuje průtok krve [1]. V případech cév vedoucích do mozku (viz. krkavice) může dojít u pacienta k mrtvici. Z hlediska morfologického vývoje můžeme rozlišit 6 stupňů rozvoje aterosklerotické léze.

Časná fáze

Stupeň I. – **Izolované pěnové buňky**. Intima je ztluštělá a najdeme v ní skupiny makrofágů, které ve své cytoplazmě obsahují kapénky lipidů.

Stupeň II. – **Tukové proužky** jsou zde makroskopicky patrné. Vzniká z nadměrného hromadění pěnových buněk. Ve stěně tepny jsou taktéž patrné makrofágy a T-lymfocyty.

Stupeň III. – **Intermediární léze**. Postupnou gradací předchozích kroků dojde k akumulaci velkého množství pěnových buněk a makrofágů, jejichž prasknutím se uvolňují lipidy do mimobuněčného prostoru.

Stupeň IV. – **Aterom**. Je to intenzivněji vyvinuté stádium Intermediární léze s hustým nahromaděním mimobuněčných lipidů, které vytvoří lipidové jádro, které obsahuje krystalky cholesterolu a zbytkové množství kalcia. Vytvoření ateromu vede k postupnému zužování vnitřního průsvitu arterie.

Pozdní fáze

Stupeň V. – **Fibroaterom**. Zvětšováním ateromu se v jeho okolí množí buňky hladkého svalstva a vaziva, které vytvářejí vazivovou vrstvu nad lipidovým jádrem. Tato léze obsahuje vystupující pojivovou tkáň, kolagen a velké množství hladkých svalových buněk. Mohou už způsobit pacientovi potíže výrazným zúžením arterie.

Stupeň VI. – **Komplikovaná léze**. Ruptura fibroateromu, tvorba hematomu s krvácením, vytvoření vředů, vytvoření krevní sraženiny (trombózy).

2.1.2 Rizikové faktory

K urychlení vytvoření Fibroateromu, nebo vůbec počátku hromadění pěnových buněk přispívá velká řada rizikových faktorů. Nejčastějším je nadměrná konzumace

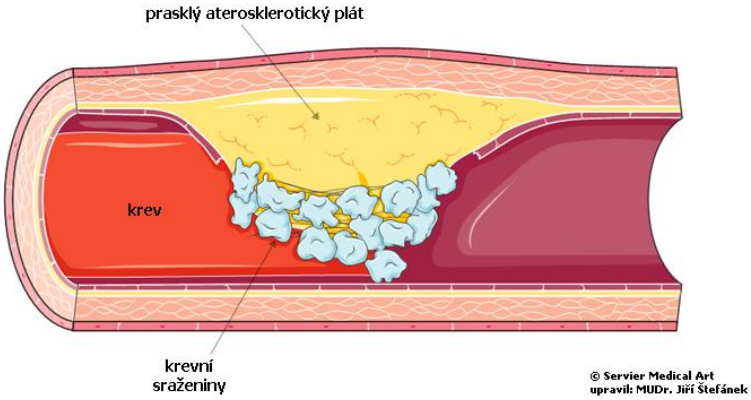
potravy, a to ne pouze tučných a mastných jídel s vysokým obsahem cholesterolu, ale i genetika. Mezi další častý rizikový faktor řadíme kouření, dlouhodobě zvýšený krevní tlak, diabetes a fyzická neaktivnost. Nově zjištěné příčiny vzniku Ateroskleróz jsou: VLDL (very low density lipoprotein) a homocystein, který vzniká při zvýšeném oxidačním stresu

2.1.3 Aterosklerotický plát

Aterosklerotický plát či plak (též aterom) je nahromadění tukové hmoty a bílých krvinek ve stěně tepny. Jedná se o příznak aterosklerózy („kornatění tepen“), onemocnění cévní soustavy, jež může za určitých okolností způsobit infarkt či mrtvici – zejména pokud aterosklerotický plak praskne a na jeho povrchu se začne srážet krev (trombóza).[2]

Rozlišujeme stabilní a nestabilní aterosklerotický plát.

- Stabilní – nachází se v něm menší počet lipidů a fibrózní obal je neporušený
- Nestabilní – v jádře plátu je velké množství lipidů, pěnových buněk a T lymfocytů. Fibrózní obal je tenký s nízkým obsahem kolagenu což evokuje větší náchylnost k ruptuře, která je častokrát odpovědná za akutní koronární příhodu(mrtvici). Zánětlivý proces, jenž probíhá v místě nahromadění makrofágů a T lymfocytů, značnou mírou přispívá k nestabilitě plátu



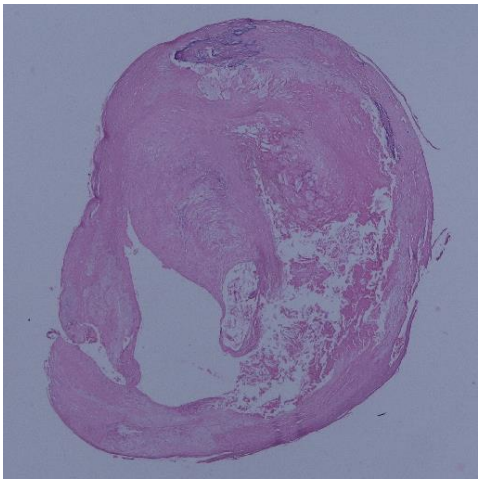
Obrázek 2.1: Prasklý aterosklerotický plát. Převzato z[3]

2.2 Histologie

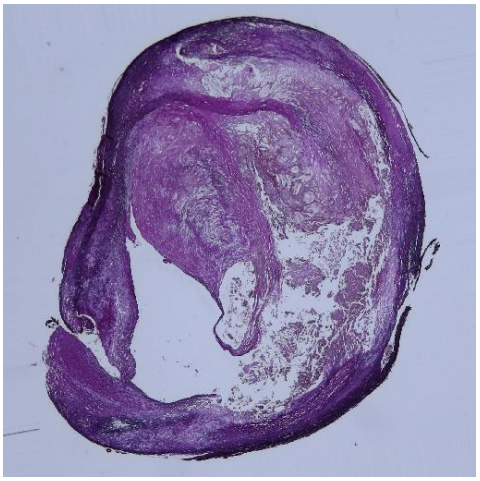
Je to vědní disciplína zabývající se mikroskopické struktury živočišných tkání. Je to invazivní metoda, při které se pacientovi odebere vzorek postižené tkáně a ten se vyšetří.

Histologie aterosklerotických plátů

Z aterosklerotického plátu se odebere část, příčný řez, kde jsou díky tomu vidět všechny části plátu. Většinou se berou vzorky po 1 cm a kvůli lepšímu a jednoduššímu určení tkání v aterosklerotickém plátu se vzorek obarví Hematxylin-eosinem, nebo Weigert van Giesonovým barvením.



Obrázek 2.2: Histologie s HE (Hematxylin-eosin) barvením



Obrázek 2.3: histologie s WG (Weigert van Gieson) barvením

2.3 Úvod do strojového učení

Strojové učení je obor zaobírajíce se způsoby, jak by se mohl systém „učit“.

2.3.1 Základní pojmy ke strojovému učení

Rozdělujeme tři základní způsoby učení:

- Učení s učitelem – *supervised learning*
- Učení bez učitele – *unsupervised learning*
- Zpětnovazebné učení – *reinforcement learning*

Každý strojový algoritmus s učitelem potřebuje vstupní data (trénovací množina) a k nim přiřazená data výstupní. Tato dvojce dat je zpracovávána algoritmem, který rozpoznává určité závislosti mezi touto dvojicí dat. Díky znalosti těchto závislostí, je pak schopen, na dalších vstupních datech předpovídat výstupy. Přesnost algoritmu se hodnotí jako odchylka očekávaného výstupu od reálného výstupu.

Strojové algoritmy bez učitele nemají k vstupním datům přiřazené žádné výstupy, a tak si musí hledat mezi daty nějakou logickou strukturu. Využívají se především při velkých množstvích vstupních dat, nebo hledáme určité pravidlo, kterým se data liší.

Zpětnovazebné učení funguje na principu zpětných vazeb. Je zapotřebí mít předem stanoveno jaké hodnocení mají jaké stavy co mohou nastat. Algoritmus prochází

problémem a po každém kroku dostává zpětnou vazbu. Finální stav má vždy nejlepší hodnocení a jeho dosažením algoritmus končí. Opakujícím se řešením problému se algoritmus učí, jak je nejlépe řešit.

Algoritmy můžeme rozdělit podle výstupu, jenž nám poskytne na:

- Regresní algoritmy
- Klasifikační algoritmy
- Shlukovací algoritmy

Klasifikační algoritmy se snaží vstupní data rozdělit do určitých tříd, jejichž počet musí být předem daný.

Výstup regresních algoritmů nejsou předem určené jako třídy v klasifikaci a produkují numerickou odpověď. Nemusí to být celá čísla, a proto se algoritmy například hodí pro ceny produktů, nebo výpočty vzdáleností.

Shlukovací algoritmy zařazují objekty do skupin s podobnými vlastnostmi. Typicky se využívají s učením bez učitele

2.3.2 Podoblasti strojového učení

Jsou různé metody rozdělování vstupních dat do jednotlivých tříd. V této kapitole vám popíši čtyři nejpoužívanějších.

1. Lineární regrese – je jednoduchý, ale účinný algoritmus s učitelem, který předpovídá spojité proměnné. Lineární regrese se snaží určit, jak se vstupní proměnná liší od výstupní proměnné. Běžně používá v problémech strojového učení k predikci spojitých proměnných, kde cílové a vstupní proměnné mají lineární vztah. [4]

2. Logistická regrese – je klasifikační, rozhodovací algoritmus, což znamená, že hledá hranice mezi dvěma třídami a zjišťuje pravděpodobnost jedné třídy. Odvozuje se od lineární regrese, ale přidává sigmoidní funkci jako další vrstvu, která zajistí, že výstup bude mezi 0 a 1. Obvykle se používá pro binární klasifikaci.

3. Naïve Bayes – Je to klasifikační technika založená na Bayesově teorému. Předpokládá nezávislosti mezi prediktory, to jest přítomnost určitého prvku ve třídě nesouvisí s přítomností jakéhokoli jiného prvku. Například ovoce je považováno za jablko, pokud je červené , 6 cm široké a kulaté. I když na sobě tyto veličiny závisí, v Bayesově algoritmu jsou brány individuálně a nezávisle na sobě.

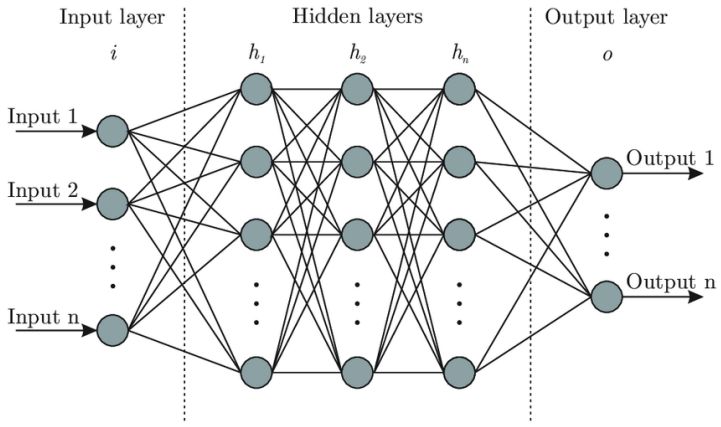
4. KNN (k nearest neighbours) – tento algoritmus využívající učení s učitelem funguje na principu hledání nejbližších k sousedů od vstupních dat. Zařazuje je do kategorie podle toho, která kategorie byla mezi k vybranými sousedy nejčastější.

2.3.3 Neuronová síť

Neuronové sítě, přesněji umělé neuronové sítě, se řadí jako podoblast strojového učení, nejčastěji pod metody učení s učitelem. Existují i sítě s učením bez učitele, které mají většinou za úkol provádět shlukové analýzy.

Základní prvky této sítě jsou neurony, které jsou navzájem propojeny a předávají si informace. Neuron může mít několik vstupů, ale pouze jednu výstupní hodnotu. Poté, co do neuronu vstoupí hodnoty, vynásobí je váhami, podle jejich významnosti pro řešený problém a výsledek sečte. Od výsledku odečte prahovou hodnotu, aplikuje aktivační funkci a pošle na výstup z neuronu.

Máme tři základní sekce neuronové sítě. Vstupní vrstvu, která zpracovává příchozí data. Skrytá vrstva, jenž se může skládat z několika vrstev neuronů a výstupní vrstva co nám prezentuje závěry ke kterým síť došla.



Obrázek 2.4 : struktura umělé neuronové sítě. Převzato z [5]

2.4 Segmentace obrazu

Segmentace, je rozdělení obrázku do různého počtu oblastí pro získání hledaných parametrů, nebo informací o snímané scéně. Různými metodami zpracování obrazu se snažíme rozdělit oblasti na scéně do tříd se stejnými vlastnostmi. Rozdělení obrazu do tříd se obvykle dělá pomocí obrazových bodů (pixelů). Mohou se kupříkladu hledat

spojitosti hodnot sousedících pixelů, nebo skokové změny v hodnotách pixelu, které značí změnu barvy nebo jasu.

Každá část obrazu by měla po dokonalé segmentaci představovat celý jeden objekt obrazu. Jelikož dokonalá segmentace není reálná, naší snahou tedy je co nejlepší segmentace částečná, ve které jednotlivé segmenty přímo nepředstavují objekty v obraze, ale oblasti obrazu, které se vyznačují vysokým stupněm homogenity. Takového druhu segmentace je možné dosáhnout pomocí operací zpracovaní obrazu. Z těchto důvodů je částečná segmentace používaná jako jeden z prvních kroků při analýze ze obrazu. [6]

2.4.1 Metody segmentace obrazu

- **Prahování**

Metoda prahování je založena na jednoduché myšlence, že objekty v pozadí mají jiné úrovně jasu. Definuje se prahová hodnota a všechny pixely s hodnotou menší, než je daná hodnota, budou považovány za pozadí. V praxi se hodnoty pozadí nastaví na 0 a ostatní hodnoty na 1. Existuje i prahování adaptivní, které neprahuje celý obraz najednou, ale pro určité části se obrazu se tvoří vlastní práh.

- **Detekce hran**

Hranou se rozumí velká změna jasu na malé oblasti. Na obraz se nahlíží jako na funkci, která nám ukazuje tyto změny jasu jako vrcholy. Funkci takto znázorněnou matematicky upravujeme, abychom zjistili, kde přesně se hrany nachází a jaké mají parametry. Jednotlivé segmenty na obraze pak můžeme jednoduše pomocí hran identifikovat.

- **Hledání oblastí**

Na rozdíl od detekce hran, jenž nejdříve hledá hranice segmentů, hledají tyto metody objekty rovnou. Pro silně zašuměné obrazy vrací lepší výsledky než metody detekce hran.

Na hledání oblastí se používají algoritmy: **Split and Merge –** obraz se postupně dělí na menší a menší oblasti, přičemž se spojují ty, které splňují kritéria homogenity. **Watershed –** obraz je „zaplavován vodou“. Jde o vytyčení bodů a od nich prohledáváním do šířky. Když se prohledávání potkají, vytvoří „bariéru“ a pokud narazí na pixel s vyšší hodnotou, tak mu ponechá původní barvu.

2.4.2 Metody strojového učení pro segmentaci obrazu

Základní metody segmentace obrazu dokážou rozdělit obraz podle barvy a kontur na ohraničené oblasti. Nicméně nedokážou přiřadit vytyčeným oblastem význam. Kdybychom například hledaly červené jablko na bílém ubruse, tak víme že oblast

s převahou červené barvy bude jablko. Problém nastane v případě, že scéna je složitá a je na ní více objektů. Když bychom například chtěli hledat jablko v misce s ovocem, už nám nepostačí pouze vědět, že je jablko červené, ale budeme potřebovat vědět, jak takové jablko skutečně vypadá.

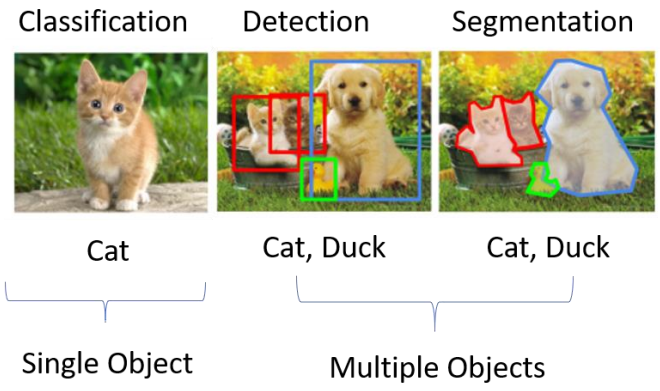
Sémantická segmentace

S problémem detekovat více objektů na jedné scéně nám pomáhá právě sémantická segmentace. Její cíl je každému pixelu obrázku přiřadit třídu. Protože se každému pixelu přiřazuje třída, je predikce nazývána jako „hloupá“. Třídy mohou zahrnovat obecné struktury jako auto, dům, člověk, nebo třeba cesta. Je to jeden z mála algoritmů, který nám dává možnost pochopit, co se na obrázku děje, jaké prostředí na obrázku je. Využívá se například u autonomních aut, pro které je pochopení okolního prostředí zásadní. Sémantická segmentace není samostatný krok v procesu úplného učení scény.

Využití neuronových sítí

První krok u určování scény je použití klasifikačního algoritmu strojového učení. Ten nám určí, jaké třídy(objekty) jsou na obrázku. Další krok je určení pozic objektů a jejich natočení v prostoru. Navazuje sémantická segmentace, která vytvoří mapu bodů se zaznačenými objekty stejně velkou jako původní obrázek. Někdy postup pokračuje ještě přes instanční segmentaci, která změní označení stejným objektům. Kdybychom například klasifikovali auta modře, tak ve výsledku by neměla žádná 2 auta na obrázku stejnou barvu. Tento komplexní proces zpracovává takzvaná neuronová síť.

Neuronové sítě zpracovávají informace podobným způsobem jako lidský mozek. Síť se skládá z velkého množství vzájemně propojených procesních prvků (neuronů) pracujících souběžně k řešení konkrétního problému. Neuronové sítě se učí příkladem. Nemohou být naprogramované k provádění konkrétního úkolu. Příklady musí být pečlivě vybírány, jinak se promarní užitečný čas, nebo dokonce v horším případě nebude síť fungovat správně. Nevýhodou je, že protože si síť zjistí, jak řešit problém sama, může její práce být nepředvídatelná.[7]



Obrázek 2.5: Kroky určování objektů na scéně. Převzato z [8]

3 Cíle práce

Cíl mé bakalářské práce je segmentovat obraz histologie aterosklerotických plátů na specifické oblasti a nahrání vytvořených dat do anotačního nástroje CVAT.

Pro automatickou segmentaci budu tvořit konvoluční neuronovou síť. Při tvoření budu používat předpřipravenou strukturu se segmentačním modelem DeepLabV3+, který nabízí nejoptimálnější řešení našeho segmentačního problému. Před vytvořením samotné sítě bude potřeba roztřídit a zařadit data(obrazy histologií). Určím trénovací a validační množinu ze které se naše síť bude učit.

K vizualizaci anotací použiji segmentační nástroj CVAT. Ten nepřijímá anotace ve formě obrazové, ale ve formě bodové. První úkol vizualizace bude tedy vytvoření logiky pro převod obrazového formátu anotace na bodový. Kvůli automatické funkci CVATu, která vyplňuje uzavřené obrazce, vyvstává úkol druhý, a to vmyslet algoritmus, který si s tímto problémem poradí.

4 Metody

V této kapitole popisuji, proč a jaké nástroje, postupy a metody jsem zvolil při tvoření náplně mé bakalářské práce.

4.1 Technická specifikace

Popis použitých technických prvků při tvoření řešení problémů.

4.1.1 Programovací jazyk

Programovací jazyk je nástroj mezi počítačem, který převádí program na fyzické prostředky a programátorem, který v daném jazyce formuluje postup řešení existujícího problému. Je to soubor pravidel pro zápis algoritmu.

Na začátku projektu bylo potřeba si určit, v jakém programovacím jazyce je dobré s daty pracovat. Učení neuronových sítí je časově a hardwarově náročná záležitost, a proto jsem si musel zvolit takový programovací jazyk co mi práci usnadní.

Kvůli flexibilitě a stabilitě jsem si zvolil pro tvorbu skriptů a neuronové sítě programovací jazyk Python. Díky oblíbenosti mezi lidmi pro jeho jednoduchost a dobré manipulaci s velkým množstvím dat, je v pythonu napsáno hodně pomocných knihoven pro práci s obrázky a polygony. Využil jsem i knihovny pro segmentaci obrazů, framework pro strojové učení a augmentace obrazů.

4.1.2 Vývojové prostředí

Vývojové prostředí (zkratka IDE, anglicky Integrated Development Environment) je software, který pomáhá při vytváření, nebo revizi kódu. Obsahuje editor zdrojového kódu, kompilátor, interpret a debugger.

Kvůli mým osobním zkušenostem jsem si vybral jako vývojové prostředí PyCharm. Architektura prostředí PyCharm dovoluje vytvoření virtuálního prostředí a díky tomu jakékoli nastavení v projektu se nepřenese do jiných projektů. PyCharm také nabízí prověření kódu, znázornění chyb v kódu a automatické opravy kódu.

4.2 Příprava dat

Způsob získání a příprava dat, které ve své práci používám.

4.2.1 Přístup k datům

Kompletní dataset obrazu histologií je uložen na síťovém úložišti CESNETU, kde jsou nahrána i data ze zbytku projektů na kterých se naše škola podílí. CESNET je sdružení vysokých škol a Akademie věd České republiky, které provozuje a rozvíjí národní e-infrastrukturu pro vědu, výzkum a vzdělávání zahrnující počítačovou síť, výpočetní mřížky, datová úložiště, prostředí pro spolupráci a nabízející širokou škálu služeb.[11]

Obrazy histologií mají přesně danou formu popisu. Ve jméně se vždy nachází číslo-biopsie_čísloŘezu_cislo-pacienta_spárovanéŘezy_typBarviva_cisloSnimku.png (3315_15_100027_1_1_HE_2054.png). Řezy jsou spárovány vždy dvojčíslím, například 1_1_HE odpovídá 1_1_VG. Všechny obrazy mají velikost 512×512 pixelů.

4.2.2 Předzpracování obrazu

V této části procesu přípravy dat se zaměřujeme na filtrování obrazu, a to manuální značení oblastí, které prováděli lékaři. Protože přesnost, se kterou je tento úkol proveden, zásadně ovlivňuje konečný výsledek, je to součást, která vyžaduje nejvyšší přesnost a optimálně více zdrojů označených(anotovaných) dat, aby se zlepšila objektivita a přesnost trénované neuronové sítě. Anotace části datasetu histologických snímků prováděl osobně pan prof. MUDr. Václav Mandys, CSc.

4.2.3 Rozdělení dat

Pro mojí práci mi bylo poskytnuto 835 neanotovaných snímků histolgií a 185 anotovaných. Snímky bylo nejdříve potřeba rozdělit do 4 skupin.

V první byly anotované histologie, ve druhé anotace k těmto histologiím. Tyto dvě skupiny dat tvořili základní trénovací množinu.

Ve třetí skupině jsem připravil neoanotované snímky jednoho barvení, ve čtvrté pak snímky barvení druhého. Rozhodl jsem se ke každému barvení udělat vlastní neuronovou síť. Na každém barvení jsou také jinak vidět různé tkáně a mohlo by docházet k falešným určením a nepřesnosti neuronové sítě.

4.3 Tvoření neuronové sítě

V tomto tématu budu popisovat, jak jsem postupoval při tvoření neuronové sítě a proč jsem zvolil právě takové postupy. Jednotlivé parametry a samotná implementace bude pospána v Implementaci 5.2.

4.3.1 Zvolení framworku strojového učení

Python nabízí hned několik knihoven, které usnadňují práci se strojovým učením. Nejpoužívanější z nich jsou Keras, PyTorch a Tensor Flow. Všechny tři knihovny jsou opensource podporované na většině velkých cloudových platformách. Model Keras se využívá při meších datasetech a nebyl tedy pro záměry mé práce vhodný. Tensor Flow a PyTorch jsou parametrově podobné knihovny, ale PyTorch je více flexibilnější, má více možností úpravy samotného procesu tvoření neuronových sítí a obsahuje více segmentačních modelů.

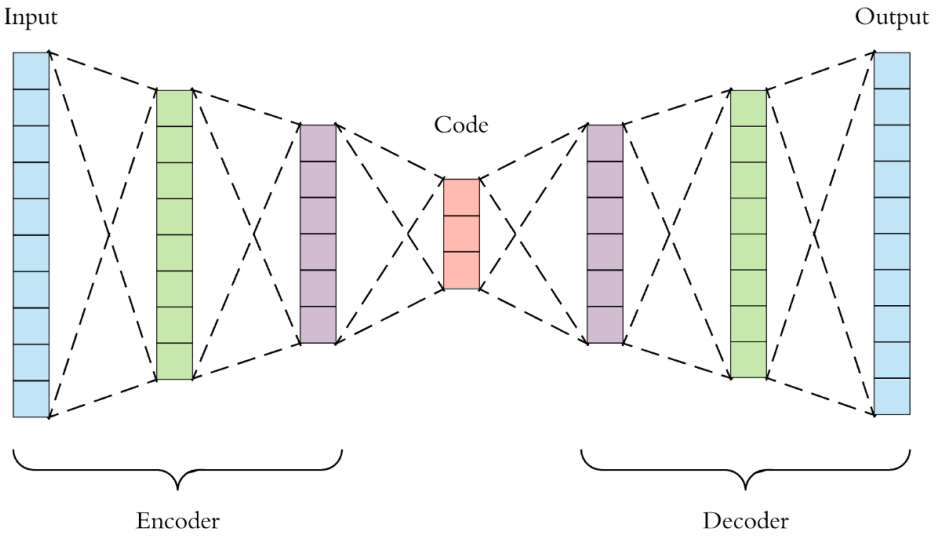
Po výběru knihovny bylo potřeba najít segmentační model, který bude tvořit kostru vytvářené neuronové sítě. Knihovna PyTorch obsahuje modul segmentations_models, který obsahuje 9 nejpoužívanějších segmentačních modelů (Unet, Unet++, Manet, Linknet, FPN, PSPNet, PAN, DeepLabV3, DeepLabV3+). Pro účely mé segmentace jsem si zvolil model DeppLabV3+, který má mezi nabízenými modely nejlepší výsledky při detekování objektů a je to současně momentálně nejpoužívanější segmentační model.

4.3.2 DeepLabV3+

Segmentační Model DeepLabV3+ provádí mnohotřídní sémantickou segmentaci (viz kapitola 2.4.2). Má plně konvoluční architekturu. To je preferované u zpracovávání obrázků, kvůli jejich velikosti. Typický obrázek má větší rozměry než 100 × 100 pixelů, což je u tří barevných kanálů více než 30 000 hodnot. Kdybychom tolik hodnot chtěli dát do plně propojené sítě tak bychom museli mít tisíce parametrů.

Konvoluční sítě řeší problém s velkými daty tak, že matematické operace aplikují jen na malých částech obrázku (např. 3 × 3 pixelů). Toto „okénko“ o 9 bodech se posunuje po celém obrázku. Takováto neuronová síť by tedy potřebovala jen 9 parametrů, ale obvykle se používá několik konvolučních vrstev za sebou, které pracují souběžně.

DeepLabV3+ na rozdíl od svých předchůdců navíc obsahuje kodér-dekodér modul. V posledních letech dosáhla konvoluční neuronová síť (CNN) pozoruhodných úspěchů v sémantické segmentaci. V dnešní době tyto metody většinou využívají architekturu kodér-dekodér jako způsob generování predikce segmentace pixel po pixelu. Kodér je určen pro extrahování map objektů a dekodér pro obnovení rozlišení mapy objektů. Díky tomu se snižuje výpočetní náročnost a zlepšují výsledky sítě.



Obrázek 4.1 : Vizualizace principu kodér-dekodér architektury. Převzato z [9]

4.3.3 Loss funkce

Loss funkce ukazuje přesnost segmentace obrazu, jako hodnotu, kterou se snažíme maximalizovat, nebo minimalizovat v závislosti na jejím typu. Nejčastěji používané loss funkce pro segmentaci obrazu jsou v současnosti cross-entropy loss funkce, která porovnává po pixelech skutečnou segmentaci s predikovanou a její varianty (vážená cross-entropy loss funkce, nebo dice loss funkce, která měří překrytí predikované segmentace a skutečných tříd. Pro moji neuronovou síť jsem tedy zvolil dice loss funkci.

$$Dice = \frac{2\,|A \cap B|}{|A| + |B|}$$

Obrzek 4.2 : Dice-loss funkce, kde A představuje predikovanou segmentaci a B reálnou třídu

4.3.4 Optimalizéry

Při učení neuronové sítě se musí nastavovat váhy, jaké budou přidávány jednotlivým vstupům do sítě. S tím, jak se učí síť, tak se mění jednotlivé váhy v modelu. K tomuto účelu nám slouží právě optimalizéry. Jsou to algoritmy nebo metody používané ke změně atributů neuronové sítě, jako jsou váhy a rychlost učení, aby se snížily ztráty. Nejpoužívanější kvůli své rychlosti a velké variabilitě je optimalizér ADAM(adaptive moment estimation), který jsem také použil ve své práci

4.3.5 Aktivační funkce

Je funkce, která pomáhá síti naučit se složité vzorce v datech. Aplikuje se v neuronech a její výsledek je informace co jde z neuronu. Normalizuje výstup sítě.

Ve své neuronové síti používám jako aktivační funkci softmax, která se využívá především při řešeních multi-třídních problémů a problémů, kde se mohou třídy na obraze překrývat.

4.3.6 Doplnění datasetu

Kvůli malému počtu anotovaných obrázků jsem musel připravit a provézt augmentace obrazů a jejich anotací pro trénovací množinu. Augmentace různými metodami deformují, nebo mění obrázek, aby se jevil jako jiný. Můžou ho například otáčet, přidávat šum a zmenšovat, nebo zvětšovat jas. Předpis, jaké augmentace se provedou jsem připravil do json souboru. Tento soubor se pak v kódu naimportuje do knihovny Albumentations , která provádí samotné augmentace.

4.4 Vizualizace anotací

Anotace vytvořené neuronovou sítí bylo potřeba nějak vizualizovat, aby se s nimi dalo dobře a jednoduše pracovat.

4.4.1 CVAT

K exportu obrazů a jejich anotací jsem použil anotační nástroj CVAT(Computer Vision Annotation Tool). Je to bezplatný nástroj pro animaci digitálních obrázků s otevřeným zdrojovým kódem napsaný v Pythonu a JavaScriptu. CVAT podporuje úlohy strojového učení pod dohledem pro detekci objektů, klasifikaci obrazu, segmentaci obrazu a anotaci 3D dat.[10]

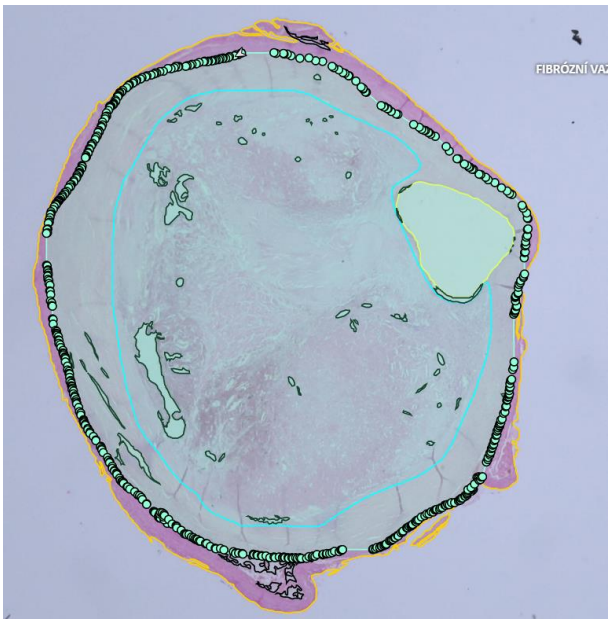
Pro vložení obrazů se nejdříve musí vytvořit takzvaný „task“. Zde si člověk určí, jak má CVAT s příchozími obrazy naložit. Důležité jsou zejména parametry: Labels – údaje

o anotované třídě (jméno, barva, ID); Z-order – seřadí třídy anotace podle pořadí které mu nastavíme; image quality – v jaké kvalitě z původního obrázku má obrázek nahrát (zadává se v procentech)

CVAT je vybavený hned několika šikovnými nástroji, které pomáhají k vizualizaci anotací. Je možno: měnit jejich průsvitnost, oddělit barevně každou instanci stejné třídy, zvýraznit jejich hranice, dát jednotlivé třídy do skupin a mnoho dalších. Mimo jiné má nástroje na ruční úpravu anotací.

4.4.2 Problém s vizualizacemi anotací

Po dokončení učení neuronové sítě, jsem nechal touto sítí anotovat všechny doposud neanotované obrazy histologií. Výsledkem byly anotace těchto histologií ve formátu PNG a velikosti 512×512 pixelů. CVAT však jako anotace přijímá soubor XML s obrazci tvořených z bodů. Sám si pak oblasti uvnitř bodů vybarví podle uvedených jmen tříd. Tímto nám vznikají dva problémy, které je nutné vyřešit. První, byl převod tříd obrazu na polygony(mnohoúhelníky), abychom mohli anotace do CVATu nahrát. Druhý problém nastává, když bychom měli dvě třídy okolo sebe. CVAT automaticky vyplňuje vnitřní prostor polygonů, takže když by se uvnitř nacházela jiná třída tak ji překryje. Druhý úkol bylo tedy vytvoření algoritmu, co by si s problémem překrývaní poradil.



Obrázek 4.3 : Anotace třídy fibrózní vazivo vložená pomocí polygonu. CVAT vybarví celou uzavřenou plochu

Obrázek 4.4 : Výsledná vizualizace anotace, kterou se snažíme dostat po vytvoření algoritmu

4.4.3 Zpracování anotace

Pro řešení obou problému je potřeba si nejdříve převést obraz na datovou strukturu abychom s ním mohli pracovat. Nejjednodušší postup a zároveň postup, který jsem použil je si načíst obrázek jako pole jednotlivých pixelů. K tomu jsem využil knihovnu OpenCV. Hlavní využití této knihovny je při strojovém učení pro detekci a klasifikaci objektů. Díky tomu je vybavena funkcionalitami, které se při naší práci budou hodit.

4.4.4 Algoritmus pro zabránění překrývání

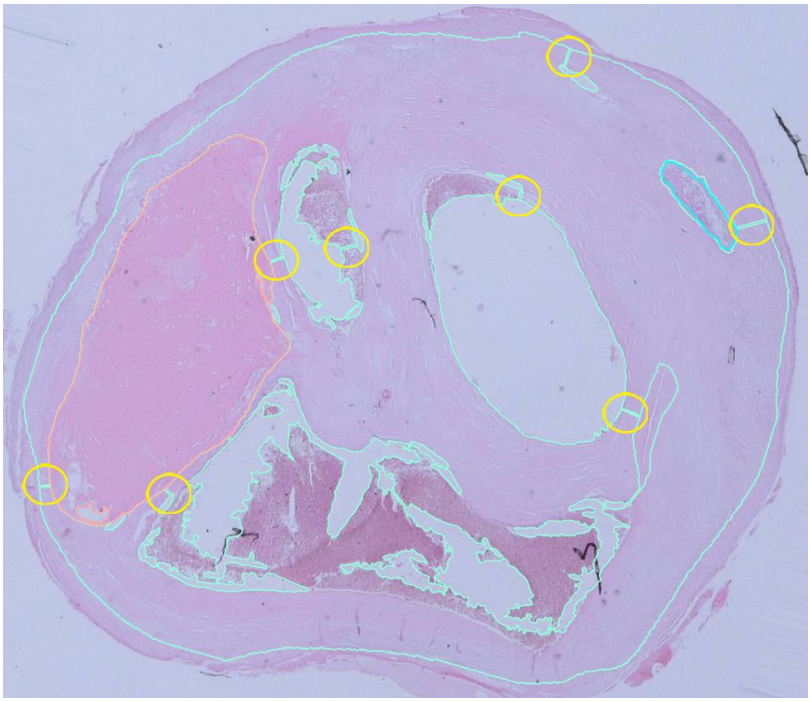
Problém nastával hlavně u tříd s velkou oblastí a nepravidelným kruhovým tvarem(viz. Obrázek 4.3). Hlavní myšlenka při tvoření algoritmu byla taková, že spojením venkovní hranice objektu třídy se všemi hranicemi vnitřních objektů, vytvoříme polygon, který bude končit těmito hranicemi a nezasáhne do vnitřních objektů. V algoritmu chceme pracovat vždy jen s jednou třídou, protože budeme chtít detekovat hranice objektů té třídy. Postupně projdeme všechny třídy a pixely co dané třídě nepatří nastavíme na 0, zbylé na 1(vytvoříme binární obraz). Při tvoření algoritmu jsem poté pokračoval následujícím způsobem:

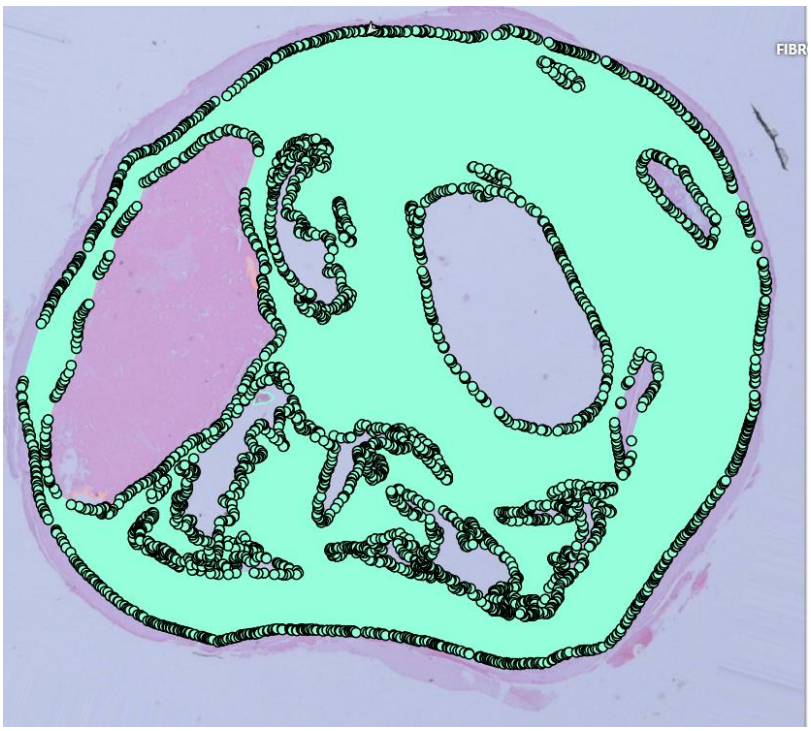
1) Nalezení počtu jednotlivých objektů dané třídy. OpenCV má právě pro tento problém vytvořenou funkci connectedComponents, která nám najde přesný počet komponent třídy na obraze.

2) Nalezení kontur. Nad každým komponentem jsem spustil funkci knihovny OpenCV findContours. Ta najde všechny kontury(obrysy) a jejich hierarchii. Hierarchii má každá kontura a odvíjí se od zvoleného parametru při použití funkce findContours. Můžeme například parametrem RETR_LIST dát všem konturám stejnou hierarchickou vrstvu, nebo použít RETRE_EXTERNAL co hledá jen venkovní kontury. Pro můj algoritmus jsem si zvolil RETRE_CCOMP, který uspořádává hierarchii do dvou kategorií, otec a potomek. Rozdělením na komponenty ve kroku 2 víme že můžeme mít jen jedenu otcovskou konturu a zbylé jsou uvnitř této kontury, tudíž potomci.

3) Nalezení nejkratší vzdálenosti mezi konturami. Informace o pozicích kontur jsou zapsány v bodech. Vezmeme vždy otcovskou konturu a konturu prvního potomka a uděláme jejich kartézský součin. Tím nám vzniknou všechny možné kombinace bodů mezi těmito dvěma konturami. Použitím vzorce (4.7), který plyne z Pythagorovi věty vypočítáme vzdálenosti těchto bodů a vybereme tu nejkratší.

4) Spojení nejbližších kontur. K tomu jsem znovu použil knihovnu OpenCV, konkrétně její metodu line, která kreslí linku do obrazů a circle, která vytváří kruh. Metodě line jsem zadal body, které od sebe měli nejkratší vzdálenost a nakreslil čáru. Protože se však pozice v bodech zaokrouhlují a namísto bodu o souřadnicích [x = 20,45; y = 20,85] nám program vypíše bod [20 20], tak metoda line nedosáhne až na dané body. Proto bylo potřeba na každém konci této čáry udělat jedno-pixelový vyplněný kruh. Tímto zásahem do obrazu se obě kontury spojí a vznikne z nich jedna.





Obrázek 4.5 : Žlutými kruhy vyznačené spoje kontur vytvořené algoritmem
Obrázek 4.6 : Zobrazení výsledné anotace po aplikování algoritmu ve CVATu

5) Odstranění pixelových chyb. Kvůli dotyku čáry jiné kontury nám může nechtěně vzniknout nová kontura o velikosti jednoho pixelu. Je potřeba vždy po zakreslení zkontrolovat, jestli se v obraze tato chyba nevyskytuje. Vytvořil jsem si proto funkci, která hledá pomocí masky tyto jedno-pixelové kontury a maže je.

6) Opakování algoritmu. Algoritmus se vrací na začátek. Opětovně se spustí metoda findContours, která bude zobrazovat o konturu, kterou jsme připojili k otcovské méně. Celý cyklus se takto provádí, dokud není pouze jedná velká otcovská kontura.

Při náhledu výsledné podoby zakreslené anotace jsou na obraze vidět spojovací čáry (viz. Obrázek 4.5). Díky tomu, že nahrané obrazy mají sloužit jen pro kontrolu práce neuronové sítě, nemá zakreslená čára negativní vliv na prezentaci a kontrolu vytvořených anotací.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Obrázek 4.7 : Vzorec vycházející z Pythagorovi věty, kterým se vypočítá vzdálenost bodů ve 2D. x a y jsou souřadnice bodů

4.4.5 Vytvoření polygonů

V další části projektu jsem z upravených anotací vytvářel polygony. Jako vstupní parametr funkce bude třída a k ní její upravená anotace.

1) Stejně jako u algoritmu pro zabránění překrývání začínáme metodou findContours, abychom našli všechny kontury.

2) U každé nalezené kontury zkontrolujeme, že má více než 2 body, aby z ní mohl být vyroben polygon. Pokud ne tak se zbytek funkce přeskočí.

3) Na samotné vytvoření polygonu jsem použil knihovnu Shapely, která slouží pro manipulaci a analýzu rovinných geometrických objektů. V ní se nachází třída geometry s funkcí Polygon. Tato funkce bere jako parametr body, které má spojit a vrací vytvořený polygon.

4) Takto vytvořený polygon by obsahoval velké množství bodů a výsledný soubor se všemi polygony by obsahoval obrovské množství dat. Tento problém jsem vyřešil aplikováním funkce simplify knihovny Shapely. Ta vezme polygon a aplikuje na něj Douglas-Peucker algoritmus, který mění křivku složenou z úseček na podobnou křivku s méně úsečkami ubráním spojových bodů. Díky aplikaci této metody nám vznikne tvarově podobný polygon s význačně menším počtem bodů.

5) Informace o všech polygonech se ukládají do listu jako objekty. Následně jsem využil knihovnu lxml, která pomáhá s převodem objektů do XML struktury.

6) Vytvořený XML soubor jsem nahrál k do CVATu k předem vytvořenému tasku (viz 4.5.1). Pro nahrání anotací je nutné, aby jména tříd a obrazů v XML souboru byla stejná jako u nahraných obrazů ve CVATu.

5 Implementace

Celý projekt je napsán v Python scriptech. Pomocné scripty, které definují často používané metody jsou uložené v jednom podadresáři, přičemž větší algoritmy mají vytvořené svoje podadresáře pro lepší přehlednost. Celý projekt je uložen v jedné složce a kvůli tomu, že je napsaný ve scriptech je možné všechny funkce a algoritmy spouštět samostatně bez jakýchkoli větších úprav. Nainstalované knihovny jsou uloženy v adresáři virtuálního prostředí, které se vytvoří pomocí venv modulu, jenž má každá verze Python nainstalovaný.

5.1 Algoritmus zabránění překrývání + vytvoření polygonů

Celý algoritmus je rozdělen ve čtyřech scriptech. Spouští se jednou main() metodou a jako vstupní data se mu vloží cesta k obrazům bez anotací a obrazům s anotacemi. Po načtení obrázku a jeho odpovídající anotace se zkontroluje, jestli jsou stejně velké a anotace ve formě listu z pixelů se pošle jako parametr do funkce convert_mask_to_cvat. V té se načítá list všech tříd, které se mohou vyskytovat v anotaci a všechny se postupně prochází. Při každém průchodu se vezme třída na anotaci se stejnou barvou a oddělí se od zbytku tříd do nového obrazu.

Ten vezmeme jako vstupní parametr pro funkci makeline na zabránění překrývání popsanou v 4.5.4, jejíž návratová hodnota bude upravený obraz s jednou třídou.

Upravený obraz anotace jedné třídy se jako parametr předá funkci get_polygons, která vytváří polygony, popsána v 4.5.5. Výsledek funkce je list polygonů a ten se připojí do druhého listu, který po všech průchodech algoritmem bude obsahovat polygony všech tříd.

Vytvoří se objekt třídy Image definované v jiném scriptu s innit parametry, které odpovídají názvu obrázku a jeho rozměrům v prostoru.

Všechny polygony tříd se předají Image objektu a ten vrátí jako výsledek funkce convert_mask_to_cvat. Celý postup se opakuje pro všechny obrazy na definované cestě. Vytvořené objekty s informacemi o polygonech se ukládají do listu.

Vytvoříme si nový objekt Anotations, jehož innit parametr je list objektů s polygony. Objekt vložíme jako parametr funkce map_to_xml, která pomocí funkcí etree.Element a etree.SubElement knihovny lxml vytvoří z objektu XML stromovou strukturu a vrátí ho.

Otevřeme pomocí funkce open() XML soubor a zapíšeme do něj navrácenou hodnotu z předchozí funkce. Zápis kvůli XML stromové struktuře musí být zapsán následovně.

```
write(etree.tostring(´stromová struktura xml´, pretty_print=True).decode())
```

Kód 1: Zápis XML stromové struktury do XML souboru

Rozepsané funkce k algoritmu jsou v příloze č1.

5.2 Neuronová síť

Všechny scripty k neuronové síti jsou uložené v jedné složce. Je v nich hlavní script s main() metodou, kterým se celé trénování spouští. Třída, do které se na začátku musí zadat všechny údaje k neuronové síti a metoda, ve které je vytvořený celý trénovací proces a další pomocné scripty, které například pomáhají k ukládání logů a grafů.

V jednom ze scriptů je definovaná třída, do které se zadává veškeré nastavení neuronové sítě až na optimalizér, jenž se nastavuje až ve trénovací funkci.

```
def __init__(self,
             model_id,
             histology_segmentation_root_str,
             dataset_name,
             sub_dataset_path_str,
             augmentation_file_name,
             model_name,
             encoder_name,
             encoder_weights,
             activation,
             in_channels,
             batch_size,
             n_epochs,
             loss_function,
             metrics,
             log_lre,
             log_lrd,
             ):
```

Kód 2: vstupní parametry třídy nastavující neuronovou síť

-**model_id, histology_segmentation_root_str, dataset_name, sub_dataset_path_str** jsou názvy složek kde jsou buď uložená data pro trénink neuronové sítě, nebo kam se mají uložit její výstupy. Všechny se zapisují ve formě string.

-**augmentation_file_name** je cesta k json souboru se zapsanými augmentacemi (viz 4.4.6)

-**model_name** je použitý segmentační model. Zadává se jako string a v našem případě to je 'deeplabv3+'. V rámci třídy mám vytvořené mapovaní kde podle zadání

parametru model_name se vybere daný model z knihovny segmentation_models_pytorch.

-**encoder_name, encoder_weights** je název použitého kodér-dekodér modulu a modulu pro jeho váhy. Zadává se ve stringu. Stjeně jako u model_name mám pro ně vytvořené mapování ke knihovně segmentation_models_pytorch. Pro moji práci jsem zvolil jako encoder_name ‘resnet34’ a pro encoder_weights ’imagenet’.

-**activation** je aktivační funkce, která se aplikuje v neuronech při učení. Zadává se ve formě stringu a je namapována jako argument na vybraný segmentační modul. Jako parametr jsem zvolil ´softmax´ (viz. 4.4.5)

-**in_channels** je počet vstupních kanálů. Zapisuje se jako integer. Barevné obrázky(RGB) mají 3 vstupy, takže jsem hodnotu nastavil na 3 a namapoval jako argument na segmentační modul.

-**batch_size** je počet obrázků co se bude zpracovávat najednou. Udává se jako integer a kvůli výpočetní náročnosti jsem tuto hodnotu nastavil na 8. Toto číslo se společně s tréninkovými a validačními datasety předává funkci DataLoder knihovny PyTorch, která předpřipraví data pro učení.

-**n_epochs** počet kolikrát celý dataset projde trénováním. Udává se jako integer. Počet epoch jsem nastavil na 800, jelikož při více než 800 epochách se nezvedá f-skóre a síť začíná být přetrénována.

-**loss_function** použitá loss funkce. Tuto hodnotu jsem nemapoval, ale zadávám jí přímo jako funkci třídy losses knihovny segmentation_models_pythorch. Zvolil jsem si DiceLoss funkci (viz 4.4.3).

-**metrics** jsou funkce co určují přesnost sítě. Jako hodnotu zadávám funkce přímo ze třídy metrics knihovny segmentation_models_pythorch. Používám Fscore metodu, která počítá f-skóre a Accuracy, jenž počítá přesnost.

-**log_lre, log_lrd** jsou rychlosti s jakými se budou neurony učit. Zadává se ve formě integeru a jdou jako argumenty do optimalizovací funkce Adam knihovny PyTorch. Zvolil jsem si číslo -4, které má nejlépe optimalizovat rychlost učení (viz [12]).

Po spuštění main() metody, si vytvoříme objekt ModelInput s těmito parametry a ten uložíme.

Spustíme funkci train_model a jako parametr jí vložíme objekt ModelInput. Načteme dva textové soubory, jeden se jmény obrázků trénovací množiny, druhý s validační. Vytvoříme dva objekty train_dataset a valid_dataset, kterým do innit metody dáme cestu ke všem obrázkům, jména obrázků ze souborů(jeden trénovací, druhý validační), preprocesing, což je encoder funkce a u train_dataset přidávám načtený soubor augmentací funkcí load knihovny albumentations.

Inicializuji si objekty train_loader a valid_loader pomocí třídy Loader knihovny PyTorch a jako innit parametry použiju: Dataset = objekty train_dataset a valid_dataset, batch_size = batch size z ModelInput, shuffle = True, num_workers = 8

Vytvořím si další dva objekty train_epoch a valid_epoch jež patří pod třídu TrainEpoch a ValidEpoch knihovny segmentation_models_pytroch. Jako innit hodnota se jim zadává loss funkce, metriky a optimalizér

Ve for cyklu volám funkci objektů train_epoch a valid_epoch run(), která jako parametr bere objekty train_loader a valid_loader a vrací dictionary s hodnotami dice_loss, fscore a accuracy. V další části kódu se zkontroluje jestli je fscore větší než největší fscore, pokud ano uloží se váhy modelu pomocí funkce save knihovny PyTorch. .Cyklus se volá, dokud číslo nedosáhne nastaveného počtu epoch.

Po skončení cyklu se uloží váhy, grafy a logy.

Odkaz na projekt:

https://github.com/vojtakropi/segmentation/tree/main/kropacek/code/python

6 Výsledky

V následující kapitole zhodnotím přesnost neuronové sítě. Měření přesnosti se provádělo pomocí metrik(viz 5.2) a dice loss funkce. Druhá část kontroly přesnosti neuronové sítě spočívala v ruční kontrole histologem.

6.1 Hodnocení přesnosti pomocí metrik

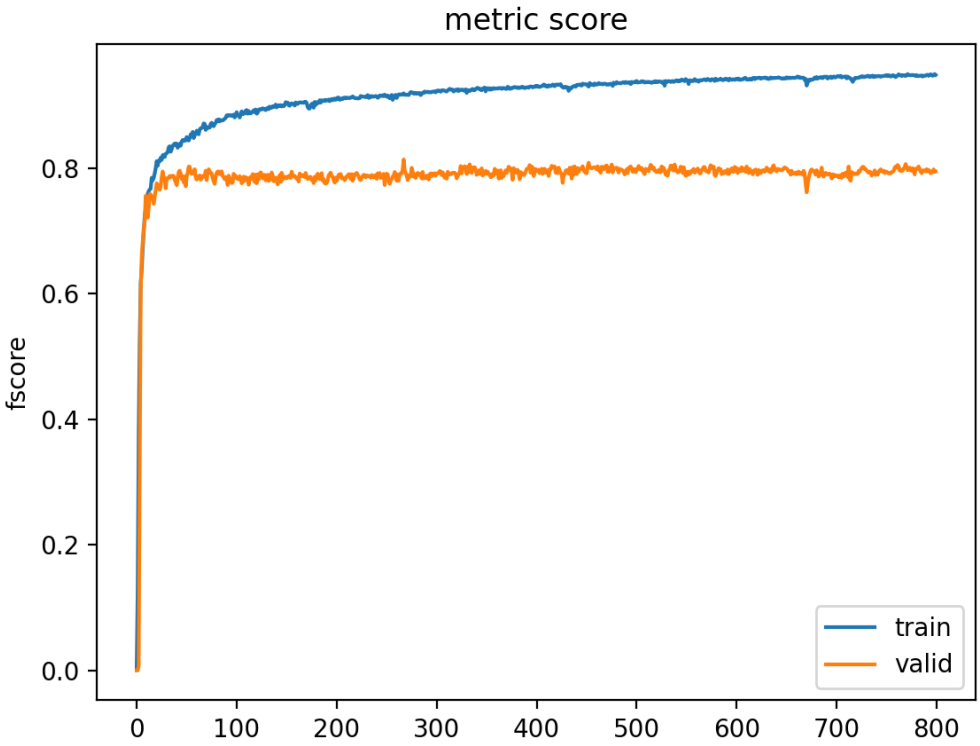
Takovéto hodnocení je automaticky generováno a počítá se po každém průchodu trénovacími daty(epochou). Ke kontrole se používají validační data a zjišťuje se, jak moc se neuronová síť při vytváření anotace zmýlila oproti originální anotaci.

6.1.1 F-skóre

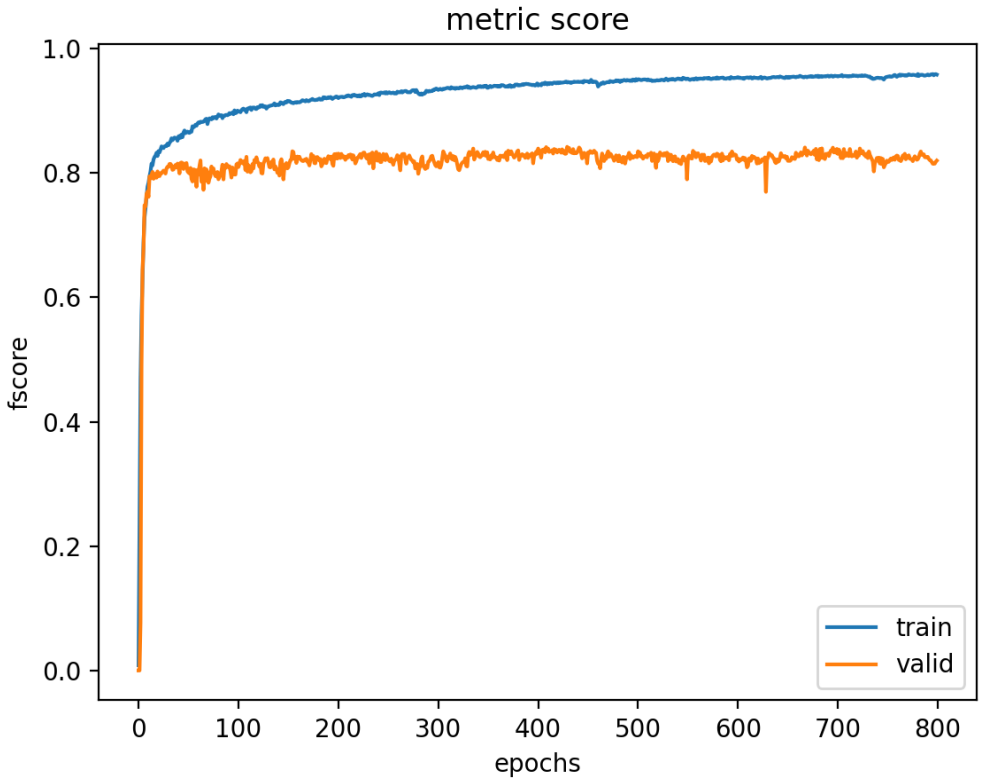
F-skóre se počítá z přesnost a Sensitivita. Přesnost je počet true positive hodnot podělených hodnotami true positive + false negative. Sensitivita je počet true positive hodnot podělených hodnotami true positive + false positive.

$$\text{F1-score} = 2.\frac{\text{Precision}.\,\text{Recall}}{\text{Precision} + \text{Recall}}$$

Obrázek 6.1 : Vzorec výpočet F-skóre



Obrázek 6.2 : Obrázek ukazuje narůstající f-skóre(přesnost) s uběhlými epochami v mé neuronové síti pro barvení HE



Obrázek 6.3 : Obrázek ukazuje narůstající f-skóre(přesnost) s uběhlými epochami v mé neuronové síti pro barvení WG

6.1.2 Číselné výstupy

Po každé epoše se měřila dice loss funkce, f-skóre funkce a obecná přesnost sítě. Logy s výsledky se ukládali do json souboru(viz. Příloha1). Finální navrácené hodnoty osmisté epochy spočítané funkcemi byli:

Pro barvení HE - "dice_loss": 0.0645, "fscore": 0.948, "accuracy": 0.989

Pro barvení WG - "dice_loss": 0.0524, "fscore": 0.958, "accuracy": 0.991

6.2 Ruční kontrola

Panu prof. MUDr. Václavu Mandysovi bylo pro ruční kontrolu vytištěno a poskytnuto 92 obrázků anotovaných neuronovou sítí. Jednalo se o shodných 46 obrázků

jednou s anotované s barvením HG, jednou s barvením WG. Přesnost anotací kontroloval porovnáním originálního vzorku prohlíženého pod mikroskopem a poskytnutých anotací na papíře.

Tabulka 7.1 : Měření přesnosti neuronové sítě ruční kontrolou

Sledovaná třída	TP	FP	TN	FN	Přesnost	Sensitivita	Fskóre
Kalcifikace (WG)	10 + 3	2	21	10	0,87	0,56	0,68
Kalcifikace (HE)	16 + 2	2	21	5	0,9	0,78	0,84
Krvácení (WG)	20 + 6	1	18	1	0,96	0,96	0,96
Krvácení (HE)	14 + 5	4	13	8	0,83	0,7	0,76
Ateromová tkáň (WG)	29 + 4	5	6	2	0,87	0,94	0,90
Ateromová tkáň (HE)	13 + 12	7	4	10	0,78	0,71	0,74
Lumen (WG)	35 + 5	3	2	1	0,93	0,98	0,95
Lumen (HE)	35 + 6	4	1	0	0,91	1	0,95
Fibrózní vazivo (WG)	30 + 16	0	0	0	1	1	1
Fibrózní vazivo (HE)	26 + 20	0	0	0	1	1	1

-true positive hodnoty jsou správně anotované třídy, „+ *číslo*“ značí počet obrazů s malou odchylkou, kdy například chyběl kousek třídy

-false positive jsou hodnoty kde byla třída zakreslena, ale neměla tam být

-true negative značí kde třída být neměla a ani tam nebyla

-false negative hodnoty označují kolikrát tam třída být měla zakreslena, ale nebyla

7 Diskuse

Použití neuronové sítě pro segmentaci obrazu se ukázalo jako dobrý krok vzhledem k pozitivním zpětným vazbám na její výsledky. Takto natrénovaná neuronová síť se dá použít k segmentaci obrazu histologie aterosklerotických plátů a je vcelku přesná. Výstupní hodnoty pro měření přesnosti neuronové sítě se pohybovali kolem 99 % a f-skóre kolem 95 %. Při ruční kontrole histologem se zjistili následující poznatky. Anotace barvení WG se ve 42 z 46 případech jevily jako přesnější. Kalcifikace se lépe projevovala u HE barvení. Krvácení a ateromová tkáň byla přesněji označena u barvení WG. Lumen byl špatně označen ve 8 případech napříč oběma barveními, přičemž se jednalo většino o detekci sekundárního lumenu v místě, kde měl plát štěrbinu.

Přesnost sítě by se ale dala zlepšit, kdyby pro trénovací množinu bylo více anotovaných obrazů. Augmentace obrazů byl nejspíše důvod často chybně označených děr jako lumen. Nastavení sítě, které jsem zvolil by mělo zajistit dobré výsledky, ale nemuselo to být nastavení, které nám poskytne výsledky nejlepší. Při trénování jsem kvůli patrné odlišnosti rozdělil data, aby měla neuronová síť jasnější rozdělení tříd při anotování, ale tento krok mohl způsobit, že neuronová síť pro dané barvení nebude tak zobecňovat. Což je to samé, jako kdybychom přetrénovali síť.

Algoritmus, který jsem vytvořil pro převod obrazové struktury do bodové, fungoval jak bylo zamýšleno a anotace se tvořili přesně podle zadaných předloh. Spojením vnitřních kontur všech oblastí vznikali uzavřené polygony, které díky tomu nepřekrývali jiné třídy na obraze.

Při spojování nejbližších bodů otce a potomka se musí nakreslit čára, která má na konci kruh, který v případě jednoho pixelu je spíše ve tvaru kříže. Někdy při spojích vznikají jednopixelové objekty, které se „odříznou“ ze spojovaného objektu. Algoritmus musí proto provádět funkci na mazání těchto děr. Jistě by se našel lepší způsob, jakým jde daný problém vyřešit bez vzniku jednopixelových objektů. Před vytvořením polygonů se aplikuje funkce, co zjednoduší polygon, ten maže některé body a v perfektním případě by se ponechávali všechny body.

8 Závěr

Cílem mé bakalářské práce bylo segmentovat obraz histologie aterosklerotických plátů na specifické oblasti a nahrání vytvořených dat do anotačního nástroje CVAT. K dosažení tohoto cíle jsem si vytvořil konvoluční neuronovou síť pro automatickou segmentaci obrazů histologií, která byla díky své přesnosti histologem uznána za dobrou a tím jsem splnil první cíl mé práce.

K vizualizaci a práci s vytvořenými anotacemi jsem měl nahrát vytvořené obrazy do anotačního nástroj CVAT. Pro nahrání dat se však nejdříve musely z obrazů udělat bodové struktury. Vymyslel jsem logiku řešení pro tento problém a vytvořil z obrazů XML soubor, který v sobě obsahoval polygony vytvořené pro všechny třídy k celému datasetu.

Anotační systém CVAT automaticky vyplňuje středy spojených objektů a bylo proto potřeba vložit do algoritmu pro tvoření polygonů funkci, která bude řešit tento problém. Algoritmus jsem sestavil a povedlo se mi vytvořit dataset, který šlo nahrát do CVATu.

Moje práce by měla sloužit pro zvětšení počtu obrazů se kterými se budou párovat snímky z ultrazvuku. Je potřeba u nich správně určit jejich stabilita, tudíž jejich složení. Toho bez anotovaných snímků, se kterými se párují lze jen těžce dosáhnout. Prozatím je anotovaných snímků jen malé množství, ale použitím mé vytvořené neuronové sítě by se jich mohlo anotovat nespočetně mnoho.

Seznam použité literatury

[1] REITSMA, W. D., ed. *Atherosclerosis*. Amstredam: Excerpta Medica, 1979. Jonxis lectures. ISBN 0-444-90075-6. (přeloženo)

[2] HANSSON, G. K.; HERMANSSON, A. The immune system in atherosclerosis. Nat Immunol. 2011, roč. 12, čís. 3, s. 204–12. Dostupné online. ISSN 1529-2916. (přeloženo)

[3] ŠTEFÁNEK, Jiří. Častý mechanizmus akutního tepenného uzávěru. *Medicína, nemoci, studium na 1. LF UK* [online]. 2011 [cit. 2022-05-12]. Dostupné z: https://www.stefajir.cz/uzaver-tepen

[4] ADAMEC, Václav. *Applied statistics*. V Brně: Mendelova univerzita, 2010. ISBN 978-80-7375-455-6. (přeloženo)

[5] FACUNDO, Bre. An efficient metamodel-based method to carry out multi-objective building performance optimizations. *ResearchGate* [online]. [cit.2022-05-12]. Dostupné z: https://www.researchgate.net/publication/337011190_An_efficient_metamodel-based_method_to_carry_out_multi-objective_building_performance_optimizat

[6] STRAKA, Stanislav. *Segmentace obrazu* [online]. Brno, 2009. Dostupné z: https://is.muni.cz/th/tzp80/. Diplomová práce. Masarykova univerzita, Fakulta informatiky. Vedoucí práce Radka POSPÍŠILOVÁ.

[7] M. S. B. M. M. P. W. *"Research Paper on Basic of Artificial Neural Network". International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 2, no. 1, Jan. 2014, pp. 96-100, doi:10.17762/ijritcc.v2i1.2920. (přeloženo)

[8] KARPATHY, Andrej. CS224d: Deep Learning for Natural Language ProcessingCS224d: Deep Learning for Natural Language Processing. *The Natural Language Processing Group at Stanford University* [online]. [cit. 2022-05-12]. Dostupné z: https://cs224d.stanford.edu/index.html

[9] DERTAT, Arden. Applied Deep Learning - Part 3: Autoencoders. *Towards Data Science* [online]. 2017 [cit. 2022-05-12]. Dostupné z: https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798

[10] VISO, AI. Computer Vision Annotation Tool (CVAT) – 2022 Overview. *Viso.ai* [online]. 2022 [cit. 2022-05-12]. Dostupné z: https://viso.ai/computer-vision/cvat-computer-vision-annotation-tool/

[11] O nás. *Cesnet* [online]. 2021 [cit. 2022-05-12]. Dostupné z: https://www.cesnet.cz/sdruzeni

[12] JORDAN, Jeremy. Setting the learning rate of your neural network. *Jeremy Jordan* [online]. March 2018 [cit. 2022-05-12]. Dostupné z: https://www.jeremyjordan.me/nn-learning-rate

[13] Mohit Sewak, Md. Rezaul Karim, Pradeep Pujari, *Practical Convolutional Neural Networks*, ed. 1, Packt, 2018, ISBN 9781788392303

[14] SMITH, S. M. and BRADY, J. M.: SUSAN. *A new approach to low level image processing*. International Journal of Computer Vision, 1997

[15] SONKA, M., HLAVAC, V., BOYLE. R.: *Image Processing, Analysis, and Machine Vision (2nd ed.)*. 1999, ISBN 053495

Příloha D: Obsah přiloženého CD

Cesta k hlavním scriptům je přes složky segmentation/kropacek/code/python/. Zde se nachází 5 složek.

Ve složce cvat jsou funkce na práci daty pro CVAT. Skript, který jsem použil pro změnu anotací na polygony se jmenuje covert_mask_to_cvat_xml.py.

Model neuronové sítě se nachází ve složce deeplearning. Hlavní spouštěcí skript se jmenuje main.py. Do složky jsem připojil trénovací a validační logy a váhy naučené neuronové sítě.

Ve složce images, jsou obrazy trénovacích množin a jejich anotace. Podsložky jsou označené názvy barvení, které na ně bylo použito.

Složka model má script, který inicializuje třídu pro vstupní parametry modelu neuronové sítě.

Ve složce utils jsou pomocné scripty pro pomoc při různých částech bakalářské prác