

Západočeská univerzita v Plzni  
Fakulta aplikovaných věd  
Katedra kybernetiky

# **BAKALÁŘSKÁ PRÁCE**

Plzeň, 2022

Jan Burian

ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta aplikovaných věd

Akademický rok: 2021/2022

## ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení: **Jan BURIAN**  
Osobní číslo: **A19B0279P**  
Studijní program: **B0714A150005 Kybernetika a řídicí technika**  
Specializace: **Umělá inteligence a automatizace**  
Téma práce: **Nástroj pro detekci buněčných jader v mikroskopických histologických obrazech**  
Zadávací katedra: **Katedra kybernetiky**

### Zásady pro vypracování

1. Seznamte se s metodami detekce objektů v obraze, věnujte se zejména metodám, které jsou vhodné pro detekci buněčných jader v mikroskopických obrazech.
2. Navrhněte nástroj pro detekci jader buněk včetně webové aplikace pro trénování a zobrazení.
3. Ověřte funkčnost navrženého nástroje pomocí vhodného experimentu.

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

Seznam doporučené literatury:

- Hruda, L., Dvořák, J., Váša, L.: On evaluating consensus in RANSAC surface registration, Computer Graphics Forum, Vol. 37(5), 2019.
- Li, X., Li, W., & Tao, R. (2020). Staged Detection-Identification Framework for Cell Nuclei in Histopathology Images. IEEE Transactions on Instrumentation and Measurement, 69(1), 183-193. <https://doi.org/10.1109/TIM.2019.2894044>
- Wu, X., Sahoo, D., & Hoi, S. C. H. (2020). Recent advances in deep learning for object detection. Neurocomputing, 396, 39-64. <https://doi.org/10.1016/j.neucom.2020.01.085>
- Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2021). Dive into Deep Learning.
- Sonka, M., Hlavac, V., & Boyle, R. (2014). Image Processing, Analysis, and Machine Vision Second Edition. Thomson-Engineering.
- Tong, K., Wu, Y., & Zhou, F. (2020). Recent advances in small object detection based on deep learning: A review. Image and Vision Computing, 97, 103910. <https://doi.org/10.1016/j.imavis.2020.103910>

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

Datum zadání bakalářské práce: **15. října 2021**  
Termín odevzdání bakalářské práce: **23. května 2022**

**Doc. Ing. Miloš Železný, Ph.D.**  
děkan



**Prof. Ing. Josef Pšutka, CSc.**  
vedoucí katedry

V Plzni dne 15. října 2021

Západočeská univerzita v Plzni  
Fakulta aplikovaných věd  
Katedra kybernetiky

# **Nástroj pro detekci buněčných jader v mikroskopických histologických obrazech**

Bakalářská práce

Autor práce: Jan Burian

Vedoucí práce: Ing. Miroslav Jiřík, Ph.D.

Plzeň, 2022



## **Prohlášení**

Předkládám tímto k posouzení a obhajobě bakalářskou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni. Prohlašuji, že jsem bakalářskou vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne .....

.....

Podpis autora

## **Poděkování**

Na tomto místě bych chtěl poděkovat svému vedoucímu bakalářské práce Ing. Miroslavu Jířkovi, Ph.D. za odborné vedení mé práce, za cenné rady a připomínky a také za možnost častých konzultací, které se týkaly řešení problematiky. Dále bych chtěl poděkovat RNDr. Vladimíře Moulisové, Ph.D. za popis procesu získání biologických vzorků.

# Anotace

Tato bakalářská práce se věnuje vytvoření nástroje pro detekci buněčných jader v mikroskopických histologických obrazech, které byly obarveny barvením H&E. Při návrhu nástroje byl kladen důraz především na detekci jader jaterních buněk. Pomocí tohoto nástroje je rovněž možné natrénovat nové modely. Nástroj k predikci dat a trénování nových modelů využívá framework Detectron2, vytvořený organizací Facebook AI Research. Jako algoritmus pro detekci objektů byl zvolen algoritmus Mask R-CNN, který je součástí Detectronu2. Obecně bylo využito konvolučních neuronových sítí, jako jednoho z nástrojů počítačového vidění. Pro praktické využití nástroje byla vytvořena webová aplikace. Jako experiment bylo natrénováno šest modelů, které byly určeny přímo k detekci jaterních buněčných jader.

**Klíčová slova:** buněčná jádra, lékařství, konvoluční neuronové sítě, počítačové vidění, umělá inteligence, Mask R-CNN, Detectron2

# Annotation

The main aim of this work is to develop a tool for the detection of cell nuclei in H&E microscopic histological images. When the tool was designed, it was mainly focused on the detection of liver cell nuclei. This tool can be also used for training new models. For the prediction and the training, the tool implements Detectron2 framework, which is developed by Facebook AI Research. As an object detection algorithm Mask R-CNN was chosen which is included in Detectron2. Basically, the tool is based on the convolutional neural network, in this case neural network is used as one of the computer vision methods. Mainly for the practical usage of the tool, the web application was created. As an experiment the six models was trained for the detection of cell nuclei.

**Keywords:** cell nuclei, medicine, convolutional neural network, computer vision, artificial intelligence, Mask R-CNN, Detectron2

# Obsah

<b>Úvod</b>	<b>9</b>
<b>1 Návrh nového segmentačního nástroje pro buněčná jádra v mikroskopických obrazech</b>	<b>10</b>
1.1 Metody počítačového vidění vhodné na segmentaci objektů . . .	10
1.1.1 Prahování . . . . .	10
1.1.2 Segmentace na základě detekce hran . . . . .	11
1.1.3 Segmentace narůstáním oblastí . . . . .	12
1.1.4 K-means algoritmus . . . . .	14
1.1.5 Staged detection . . . . .	15
1.1.6 Segmentace srovnáním se vzorem . . . . .	16
1.2 Konvoluční neuronové sítě . . . . .	17
1.2.1 Architektura . . . . .	17
1.2.2 Detekce objektů pomocí CNN . . . . .	20
1.2.3 Využití . . . . .	22
1.2.4 Historie . . . . .	23
1.2.5 Významné datasety spojené s rozpoznáním objektů . . .	24
1.3 Detectron2 . . . . .	26
1.4 Mikroskopické histologické obrazy . . . . .	27
1.4.1 Zpracování fyzického vzorku . . . . .	27
1.4.2 Digitalizace vzorku . . . . .	29
1.5 ScaffAn . . . . .	29
<b>2 Návrh segmentačního nástroje</b>	<b>30</b>
2.1 Anotace dat . . . . .	30
2.2 Extrakce anotací . . . . .	30
2.3 COCO dataset . . . . .	31
2.4 Experiment . . . . .	36
2.4.1 Použitá konvoluční neuronová síť . . . . .	36
2.4.2 Popis experimentu . . . . .	37
2.4.3 MetaCentrum . . . . .	38
2.4.4 Hardwarové parametry experimentu . . . . .	40
2.4.5 Trénování modelu neuronové sítě . . . . .	41
2.5 Vyhodnocení experimentu . . . . .	42
2.5.1 Doby trénování . . . . .	42
2.5.2 Intersection over Union . . . . .	42
2.6 Schéma vytvořeného nástroje . . . . .	44
<b>3 Vytvoření webové aplikace</b>	<b>45</b>
3.1 Použité technologie . . . . .	45
3.2 Popis aplikace . . . . .	45
<b>Závěr</b>	<b>49</b>



<b>Reference</b>	<b>50</b>
<b>Seznam obrázků</b>	<b>55</b>
<b>Seznam tabulek</b>	<b>57</b>

# Úvod

V dnešní době začínají čím dál více mezi sebou spolupracovat vědci z technických a lékařských oborů, příkladem budiž např. spolupráce Fakulty aplikovaných věd a Biomedicínského centra v Plzni, spadajícího pod Lékařskou fakultu UK. Cílem této spolupráce je nalezení nových postupů a metod zaměřujících se především na léčbu orgánových onemocnění. V současné době se lékaři a vědci v Biomedicínském centru mimo jiné zabývají problematikou týkající se onkologického onemocnění lidských jater.

Pokud je pacient postižen touto nemocí, pak je možné za účelem vyléčení provést buď chirurgický zákrok, při kterém dojde k odstranění postižené části jaterní tkáně (jaterní tkáň má schopnost regenerace), anebo kompletní transplantaci jater. Operovatelnost je závislá na velikosti nádoru. K transplantaci jsou však nyní potřeba játra lidského dárce.

Z důvodu nedostatku dárců jater, se v Biomedicínském centru rozhodli realizovat výzkum, týkající se nahrazením lidských jater za játra prasečí. Prasečí játra a lidská játra jsou si velice podobná, především však strukturně.

K výzkumu jsou využívána přeštická černostrakatá prasata. Víze výzkumu spočívá, v tom, že se do prasečího decelularizovaného jaterního skeletu (scaffoldu) umístí pacientovy zdravé jaterní buňky. Tímto způsobem by tedy teoreticky bylo možné pacientovi „vypěstovat“ játra nová. Během tohoto procesu je třeba sledovat, zda pacientovy jaterní buňky správně osidlují scaffold.

Hlavním cílem této bakalářské práce je vytvoření nástroje pro detekci buněčných jader v mikroskopických histologických obrazech, sloužícího k analýze již zmíněných obrazů a zjištění přítomnosti jader jaterních buněk ve scaffoldu. Jde především o propojení technologií umělé inteligence (neuronové sítě) s lékařstvím. Histologické obrazy byly poskytnuty přímo Biomedicínským centrem. Jako rozhraní pro vytvoření tohoto nástroje byla použita aplikace *ScaffAn*, vyvíjená vedoucím této práce Ing. Miroslavem Jiříkem, Ph.D. Tato aplikace je již lékaři v Biomedicínském centru používána v praxi. Nástroj je napsaný v programovacím jazyce Python 3.6. Hlavním účelem této práce je tedy rozšíření funkcionalit tohoto nástroje. Aby bylo možné vytvořený nástroj pro detekci buněčných jader využívat v praxi, je dalším cílem vytvoření uživatelského rozhraní formou webové aplikace. V závěru práce je provedeno vyhodnocení úspěšnosti vytvořeného nástroje pomocí experimentu.

# 1. Návrh nového segmentačního nástroje pro buněčná jádra v mikroskopických obrazech

## 1.1 Metody počítačového vidění vhodné na segmentaci objektů

Cílem segmentačních metod [1] je rozdělení obrazu na části či objekty, které souvisí s reálným světem. Segmentace může být buď kompletní, anebo částečná. Výsledkem kompletní segmentace je množina disjunktních oblastí. Jednotlivé oblasti odpovídají objektům zachyceným ve vstupním obrazu. Po aplikaci částečné segmentace dostaneme výsledek, ve kterém jednotlivé oblasti obrazu neodpovídají přímo objektům v obrazu. Kompletní segmentace obrazu  $R$  je definovaná jako konečná množina oblastí  $R_1, \dots, R_S$ ,

$$R = \bigcup_{i=1}^S R_i, \quad R_i \cap R_j = \emptyset, \quad i \neq j. \quad (1.1)$$

Základní segmentační metody je možné rozdělit do tří základních skupin: (i) prahování, (ii) na základě detekce hran a (iii) na základě narůstání oblastí.

### 1.1.1 Prahování

Prahování [1] je nejjednodušší segmentační metoda, která se používá zejména k segmentaci černobílých obrazů. Mezi vlastnosti objektů v obrazu patří např. stálá odrazivost nebo absorpce světla, na základě těchto vlastností je následně možné definovat tzv. práh, pomocí něhož je možné segmentovat objekty a pozadí obrazu. Prahování je rychlá a výpočetně nenáročná metoda. Jedná se o nejstarší segmentační metodu, jež je využívána dodnes, a to především u jednoduchých segmentačních úloh. Prahování lze definovat jako transformaci vstupního obrazu  $f$  na výstupní segmentovaný binární obraz  $g$ :

$$g(i,j) = \begin{cases} 1 & \text{pro } f(i,j) > T, \\ 0 & \text{pro } f(i,j) < T, \end{cases} \quad (1.2)$$

kde  $T$  je práh,  $g(i,j) = 1$  zvýrazní objekty a  $g(i,j) = 0$  rozdělí pozadí obrazu od objektu.

Pomocí prahování lze kompletní segmentace dosáhnout zejména u jednoduchých obrazů, ve kterých se objekty vzájemně nedotýkají a zároveň jsou jasně rozpoznatelné od pozadí obrazu. Úskalím této metody se může stát zvolení správné hodnoty prahu, jelikož právě na tom je závislá úspěšnost

segmentace. Hodnota prahu může být určena např. z bimodálního histogramu a to jako minimum mezi dvěma největšími lokálními maximy.

Pouze v některých úlohách bude jediný práh pro celý obraz (globální prahování [1]) úspěšný, a to především kvůli tomu, že objekty v obrazu nejsou většinou v jednom odstínu šedé. Kvůli tomu byla zavedena segmentace používající více prahů (adaptivní prahování), v tomto případě jsou hodnoty prahu funkcí lokálních obrazových charakteristik. Globální práh je získán z celého obrazu  $f$ :

$$T = T(f). \quad (1.3)$$

Lokální prahy jsou navíc závislé na pozici v obrazu. Definice lokálního prahu je tedy následující:

$$T = T(f, f_c), \quad (1.4)$$

kde  $f_c$  je obrazová část, ve které je určen práh.

Jednotlivé lokální prahy jsou pak nezávisle určovány ze subobrazů, které vznikly rozdělením obrazu. Pokud nelze práh určit ze subobrazu, pak může být interpolován ze sousedních subobrazů. Následně je každý obraz zpracován ve vztahu s jeho předtím určeným prahem.

Prahování, definované vztahem 1.2, má mnoho modifikací. Jednou ze změn je např. segmentovat obraz do pixelových oblastí se stupni šedi z množiny  $D$ . Dostaneme modifikovanou definici prahování:

$$g(i,j) = \begin{cases} 1 & \text{pro } f(i,j) \in D, \\ 0 & \text{jinak} \end{cases} \quad (1.5)$$

Za pomocí tohoto prahování je možné např. segmentovat krevní buňky z mikroskopických obrazů [1], ve kterých je cytoplazma zbarvena jednotlivými stupni šedi a to v určitém intervalu, pozadí je v tomto případě světlejší a jádra buněk naopak tmavší. Pomocí tohoto přístupu je možné rovněž najít i hranice objektů.

### 1.1.2 Segmentace na základě detekce hran

Segmentace na základě detekce hran [1] [2] sice patří mezi nejstarší segmentační metody, avšak stále se jedná o jednu z nejdůležitějších segmentačních metod. Tento druh segmentace využívá tzv. hranových detektorů, jejichž cílem je najít místa v obrazu, ve kterých dochází k určité nespojitosti v jasu, barvě, textuře, apod. Tato místa nazýváme hranami. Obraz, na který byly aplikovány hranové detektory nelze považovat za konečný výsledek segmentace, tento obraz se nazývá obraz hran. V dalším zpracování obrazu hran je třeba spojit hrany do řetězců, které lépe odpovídají hranicím v obrazu. Hranice lze definovat jako množinu bodů, které patří do určité oblasti, ale ve svém okolí mají bod, které do dané oblasti nepatří. V této fázi je tedy hlavním cílem dosažení alespoň částečné segmentace.



Konkrétní metody [1] [2], spadající pod tento typ segmentace, mohou využívat apriorní informaci. Platí přímá úměra - čím více je známo apriorní informace, tím lepších výsledků segmentace lze dosáhnout. Pokud je k dispozici velká část apriorní informace, pak hrany a jejich vztahy jsou striktně určeny. Segmentace musí splňovat podmínky dané apriorní informací. Pokud není apriorní informace známa, pak segmentační metoda bere v úvahu lokální vlastnosti obrazu společně se specifickými vlastnostmi konkrétní aplikační oblasti.

Mezi nejčastější problémy segmentace na základě detekce hran patří přítomnost hran v místech, kde není hranice a nepřítomnost hrany, kde se nachází hranice. Tyto problémy, mající negativní efekt na segmentační výsledky, jsou způsobeny obrazovým šumem či nevyhovující informací v obrazu.

**Prahování obrazu hran** [1] patří mezi konkrétní metody segmentace na základě detekce hran. Obvykle je v obraze jen velmi málo míst s nulovou hodnotou velikosti hrany, což je způsobeno přítomností šumu. Použitím metody prahování hran lze odstranit tyto malé nevýrazné hrany a zároveň zachovat hrany výrazné. Zvolení vhodného globálního prahu je velice často obtížné a někdy i nemožné, proto se používá tzv. p-tile prahování.

Další metodou je např. **určení hranice s využitím znalosti její polohy** [1] [2]. Předpokladem je znalost informace o předpokládané či pravděpodobné poloze a tvaru hranice. Tato informace může být získána např. prostřednictvím znalosti vyšší úrovně, anebo jako výsledek segmentace, jež byla aplikována na obraz v nižším rozlišení. Pokud je známa přibližná poloha hranice, pak je možné určit její polohu pomocí hranových buněk, nacházejících se v blízkosti předpokládaného umístění hranice, a které mají směr blízký předpokládanému směru hranice v daném místě. Pokud je nalezen dostatečný počet obrazových bodů, které vyhovují těmto podmínkám, je těmito body proložena aproximační křivka, pomocí níž dojde ke zpřesnění hranice. Alternativně, při znalosti koncových bodů hranice a předpokladu malého šumu a malého zakřivení hranice, lze použít postupné dělení hranice a hledat nejsilnější hranu, jež se nachází na kolmicích čáry, která spojuje koncové body každé rozdělené části. Tato metoda může být opakována.

### 1.1.3 Segmentace narůstáním oblastí

Využití tohoto druhu segmentace [1] [2] je výhodné především u obrazů se šumem, u nichž je složité nalézt hranice. Důležitou vlastností oblastí je tzv. homogenita, jež je využívána jako hlavní segmentační kritérium. Hlavní myšlenkou homogenity je rozdělení obrazu do jednotlivých oblastí, které jsou vzájemně mezi sebou co nejvíce homogenní. Kritérium homogenity může být založeno např. na jasových vlastnostech .

Základní požadavky pro oblasti jsou definovány jako:

$$H(R_i) = \text{TRUE}, \quad \text{pro } i = 1, 2, \dots, S, \quad (1.6)$$

$$H(R_i \cup R_j) = \text{FALSE}, \quad i \neq j, \quad R_i \text{ sousedí s } R_j, \quad (1.7)$$

kde  $S$  je počet je počet oblastí v obraze a  $H(R_i)$  je binární vyjádření kritéria homogenity oblasti  $R_i$ .

Výsledné oblasti segmentovaného obrazu musí být homogenní a maximální. Přívlastkem „maximální“ je myšleno, že kritérium homogenity nenabude hodnoty TRUE po procesu spojení oblasti s jakoukoliv oblastí sousední. Nejjednodušší kritérium homogenity používá průměrnou hodnotu úrovně sedi v oblasti, dále pak vlastnosti týkající se barev či textury.

**Spojování oblastí** [1] [2] je nejpřirozenější metodou segmentace narůstáním oblastí. Tato metoda vychází z počátečních nezpracovaných obrazových dat, každý pixel v tomto případě reprezentuje jednu oblast. Tyto oblasti ovšem téměř jistě nesplní rovnici 1.6, tím pádem budou oblasti spojovány tak dlouho, dokud bude zůstávat splněná podmínka definovaná rovnicí 1.7. Dalším krokem je spojení dvou sousedních oblastí. Pokud tímto sloučením vznikne oblast nová, pak splňuje kritérium homogenity. Výsledek segmentace je závislý na pořadí oblastí, které jsou spojovány. Jinými slovy výsledky mohou být různé, v případě, že proces segmentace začíná ať už v levém horním či pravém dolním rohu. Nejjednodušší metody na počátku procesu segmentace využívají ke spojování oblasti o velikostech  $2 \times 2$ ,  $4 \times 4$  nebo  $8 \times 8$  pixelů. Popisy jednotlivých oblastí jsou založeny na jejich statistických jasových vlastnostech, příkladem budiž histogram jasu v oblasti. Popis oblasti je srovnáván s popisem sousední oblasti, pokud se tyto oblasti shodují, pak dojde ke spojení a vzniku nové oblasti. V opačném případě, pokud ke shodě nedojde, jsou oblasti označeny jako neodpovídající. Spojování sousedních oblastí pokračuje mezi všemi sousedy, včetně nově vytvořených. Pokud žádné dvě oblasti nemohou být spojeny, pak proces končí.

**Štěpení oblastí** [1] je opačnou procedurou spojování oblastí. Metoda začíná s obrazem reprezentovaným jako jedna oblast, která obvykle nespokojí podmínku definovanou rovnicí 1.7. Poté metoda sekvenčně rozděluje oblasti v obraze. Výsledná segmentace je rozdílná oproti výsledné segmentaci spojováním oblastí, i přesto že jsou použita stejná kritéria homogenity.

Kombinací předešlých dvou metod vznikne metoda **štěpení a spojování oblastí** [1] [2]. Tato metoda zachovává výhody obou přístupů. Je založena na základě pyramidální reprezentace obrazu, oblasti mají tvar čtverce a korespondují s elementy v odpovídající úrovni pyramidy. Pokud jakákoliv oblast v jakékoli úrovni pyramidy není homogenní, pak je oblast rozdělena do čtyř oblastí. Naopak pokud existují čtyři oblasti s přibližně stejnou hodnotou homogenity, pak jsou spojeny do jedné oblasti vyšší úrovně pyramidy. Proces segmentace lze interpretovat jako konstrukci segmentačního „čtyřstromu“, ve kterém každý list odpovídá homogenní oblasti. Štěpení a spojování koresponduje s přidáváním nebo odebráním prvků segmentačního „čtyřstromu“. Po skončení segmentačního procesu odpovídá počet listů „čtyřstromu“ počtu segmentovaných oblastí. Algoritmus štěpení a spojování oblastí končí ve chvíli, kdy nelze spojit ani rozdělit žádnou oblast.

### 1.1.4 K-means algoritmus

Jako další metoda na proces segmentace může být použit algoritmus k-means clustering [3], což je algoritmus používaný na segmentaci oblasti zájmu od pozadí obrazu. Tento algoritmus byl představený Macqueenem v roce 1967. Jedná se o jeden z nejpoužívanějších shlukovacích algoritmů. Na rozdíl třeba od hierarchického shlukování je tento algoritmus rychlejší. Obecně shlukování je proces, který rozdělí množinu dat do specifického počtu skupin (shluků). Navíc je k-means algoritmus jednoduchý a umožňuje pracovat s velkými daty. Cílem k-means algoritmu je klasifikace vstupních dat do  $k$  počtu shluků. Nevýhodou může být, že pro správné fungování algoritmu, je třeba předem určit správný počet shluků (počet  $k$  centroidů) a také správně zvolit počáteční středy centroidů. Po tomto kroku se následně iterativně zjišťují vzdálenosti jednotlivých bodů ze vstupní množiny dat k jednotlivým centroidům. K tomuto účelu je často využívána Euklidovská vzdálenost. Bod je vždy zařazen do shluku, k jehož centroidu je vzdálenost bodu minimální.

Předpokládejme obraz s rozlišením  $x \times y$ . Obraz budeme chtít rozdělit do  $k$  shluků. Necht'  $p(x,y)$  jsou vstupní pixely, které chceme shlukovat a  $c_k$  středy shluků. Algoritmus k-means lze popsat následovně: (převzato z [3])

1. Urči počet shluků  $k$  a jejich středy.
2. Pro každý pixel v obrazu spočítej Euklidovskou vzdálenost  $d$  k jednotlivým středům shluků za použití následujícího vztahu:

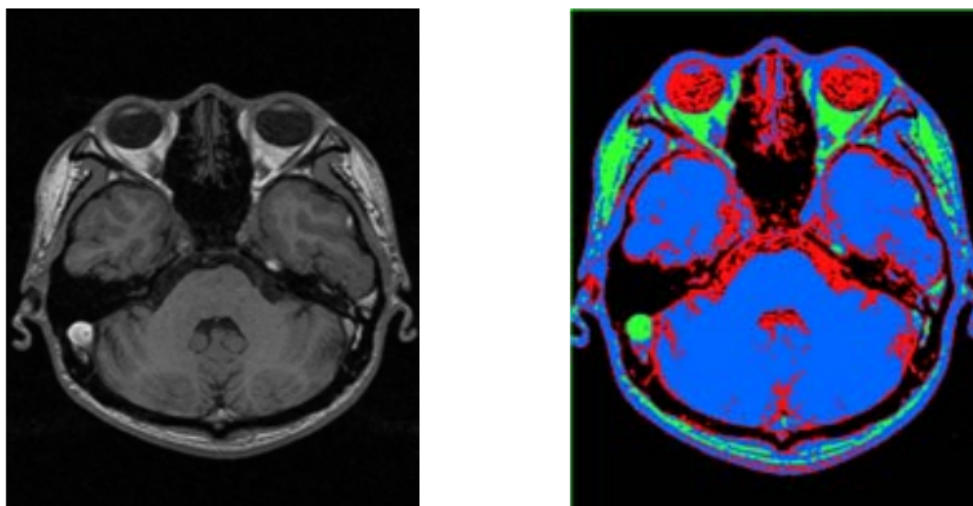
$$d = \|p(x,y) - c_k\|. \quad (1.8)$$

3. Všechny pixely přiřaď k nejbližšímu středu shluku.
4. Po zařazení všech pixelů, přepočti nové hodnoty středů  $c_k$  za použití následujícího vztahu:

$$c_k = \frac{1}{k} \sum_{y \in c_k} \sum_{x \in c_k} p(x,y). \quad (1.9)$$

5. Opakuj proces, dokud vyhovuje toleranční hodnotě.
6. Přetvoř shlukované pixely na obraz.

Tato segmentační metoda může být použita např. v lékařství [4], a to v oblasti zpracování medicínských obrazů (obr. 1.1). V tomto případě algoritmus k-means slouží k vytvoření primární segmentace vstupního obrazu. Jako vstupní data jsou použity obrazy hlavy z magnetické rezonance. Tyto obrazy se obecně skládají z oblastí reprezentující kost, měkké tkáně, tuk a pozadí. V tomto případě je tedy  $k$  rovno 4.

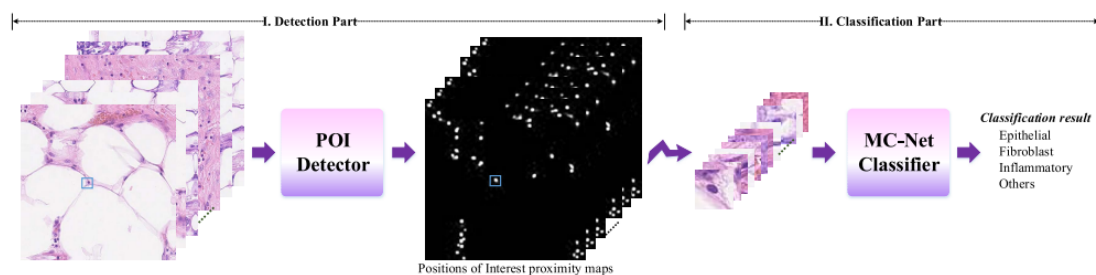


Obrázek 1.1: Vlevo je obraz z magnetické rezonance, vpravo obraz po aplikaci k-means algoritmu. Převzato z [4].

### 1.1.5 Staged detection

Staged detection [5] je vytvořený framework sloužící k automatické identifikaci a detekci buněčných jader v histopatologických obrazech, které zobrazují rakovinu tračnicku. Framework pracuje na základě konvolučních neuronových sítí. Samotný histopatologický obraz může čítat až tisíce buněk, pro pathology je tedy nemožné přesně identifikovat všechny buňky. Navíc buňky obvykle nemají jasně danou strukturu.

Staged detection je složen ze dvou stupňů - *POI CNN detektor*, jenž má za úkol využít dostupných informací o pozicích buněk společně s procesy upsamplingu a downsamplingu, a *MC-Net klasifikátor*, který automaticky získá příznaky a následně klasifikuje detekované buňky. Tento proces může být zjednodušeně vysvětlen pomocí obr. 1.2.



Obrázek 1.2: Nejprve se navržený POI detektor naučí nalézt pozice jader, následně pak dojde, pomocí MC-Netu, k identifikaci typu buňky. Převzato z [5].



### 1.1.6 Segmentace srovnáním se vzorem

Obecně nejjednodušším přístupem nalezení známého objektu v obraze je hledání jeho dokonalé kopie [1]. Tato úloha je jednoduchá v případě, že vzor objektu není nijak zmenšený, natočený či jinak zkreslený.

Jednou z metod, na měření shody mezi šablonou (známým objektem) a obrazem, je metoda součet rozdílů čtverců (SSD) [6]. Tato metoda je založena na výpočtu rozdílů intenzit pixelů obou obrazů. Dostaneme tedy následující rovnici:

$$SSD(x_a, x_b) = \sum_{i=1}^R \sum_{j=1}^C (T(i, j) - I_{x_a+i, x_b+j})^2, \quad (1.10)$$

kde  $T$  je vzor a  $I$  je obraz, ve kterém hledáme daný vzor.  $R$  je počet řádků obrazu známého objektu (šablony) a  $C$  je počet sloupců obrazu známého objektu (šablony),  $x_a$  a  $x_b$  jsou proměnné, pomocí nichž se šablona „posouvá“ po obraze ve směru os  $x$  a  $y$ .

Po úpravě rovnice 1.10 podle vzorce  $(a - b)^2 = a^2 - 2ab + b^2$ , dostaneme následující rovnici:

$$SSD(x_a, x_b) = \sum_{i=1}^R \sum_{j=1}^C (T_{i,j})^2 - 2 \sum_{i=1}^R \sum_{j=1}^C (T_{i,j} I_{x_a+i, x_b+j}) + \sum_{i=1}^R \sum_{j=1}^C (I_{x_a+i, x_b+j})^2, \quad (1.11)$$

ve které je první výraz konstanta a třetí výraz ve většině případů pouze pomalu mění svou hodnotu ve vztahu s proměnnými  $x_a$  a  $x_b$ .

Vezmeme tedy v potaz tedy pouze prostřední výraz, kterým získáme vztah pro vzájemnou korelaci, jež je většinou užívána jako míra souhlasu mezi vzorem a obrazem [1].

$$C_T(x_a, x_b) = \sum_{i=1}^R \sum_{j=1}^C (T_{i,j} I_{x_a+i, x_b+j}). \quad (1.12)$$

## 1.2 Konvoluční neuronové sítě

Konvoluční neuronové sítě (CNN) [7] jsou jedním z konkrétních typů neuronových sítí. Architektury založené na konvolučních neuronových sítích jsou dnes velice hojně využívány, zejména v úlohách spojených s počítačovým viděním, ať už se jedná o rozpoznání obrazů, detekci objektů nebo sémantickou segmentaci. Tento druh sítí je velice výpočetně efektivní, za prvé díky menšímu počtu parametrů ve srovnání s fully-connected architekturami (FCNN) a za druhé díky tomu, že matematické operace konvoluce je možné paralelizovat napříč jádry GPU.

### 1.2.1 Architektura

Konvoluční neuronové sítě se obvykle skládají z několika konvolučních vrstev, pooling vrstev, nelineárních aktivačních vrstev a fully connected vrstev [8]. Právě použitím konvolučních vrstev se konvoluční neuronové sítě odlišují od ostatních typů neuronových sítí.

V konvoluční vrstvě se ze vstupního obrazu (vstupní matice) za použití matematické operace zvané *konvoluce* vytvoří výstupní matice. V matematice je konvoluce dvou funkcí  $f$ ,  $g$  definovaná jako:

$$(f * g)(\mathbf{x}) = \int f(\mathbf{z})g(\mathbf{x} - \mathbf{z})d\mathbf{z}, \quad (1.13)$$

kde funkce  $f(x)$  je v našem případě vstupní obraz a funkce  $g(x)$  je nazývána jádrem, nebo také filtrem.

Vzorec pro diskrétní dvourozměrné obrazy používaný zejména v počítačové grafice je pak definován jako:

$$(f * g)(i,j) = \sum_a \sum_b f(a,b)g(i-a,j-b) \quad (1.14)$$

Na závěr lze definovat vzorec konvoluce ve zkrácené formě:

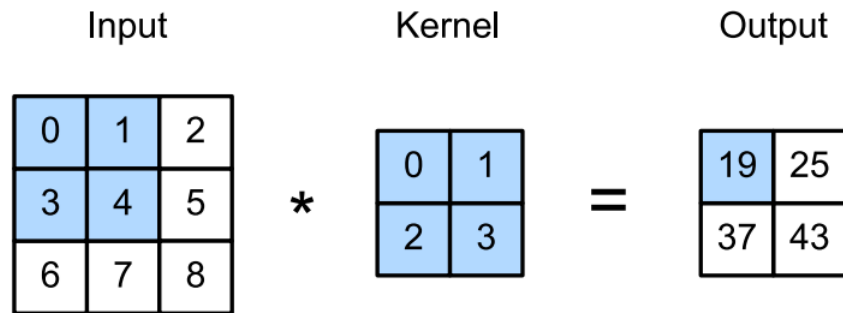
$$\mathbf{O} = \mathbf{I} * \mathbf{K}, \quad (1.15)$$

kde  $\mathbf{O}$  reprezentuje výstupní obraz,  $\mathbf{I}$  je vstupní obraz a  $\mathbf{K}$  je konvoluční jádro (maska).

Konvoluce v našem případě spočívá v přenásobení vstupního obrazu konvolučním jádrem (maskou) [7]. Tento proces je ilustrován na obr. 1.3. Tuto operaci lze volně vysvětlit jako „přikládání matice jádra na matici vstupního obrazu“. Operace začíná v levém horním rohu matice vstupního obrazu, následně je maska posunována směrem zleva doprava a shora dolů. Submatice vstupního obrazu a maska jsou mezi sebou vzájemně násobeny po prvcích. Výsledkem každého násobení je skalární hodnota, tato hodnota je pak obsažena ve výstupní matici, jejíž velikost je definovaná jako:

$$(n_h - k_h + 1) \times (n_w - k_w + 1), \quad (1.16)$$

kde  $n_h$  je počet řádků vstupní matice,  $k_h$  je počet řádků jádra,  $n_w$  je počet sloupců vstupní matice a  $k_w$  je počet sloupců jádra.



Obrázek 1.3: Dvoudimenzionální konvoluce a výpočet prvku na pozici  $o_{11}$  ve výstupní matici:  $0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 2 = 19$ . Převzato z [7].

V některých případech je však třeba použít v konvoluční vrstvě i tzv. padding [7], což je „orámování“ vstupního obrazu nulami. Zásadou paddingu nedojde ke ztrátě pixelů na okraji obrazu. Velikost obrazu je pak definována jako:

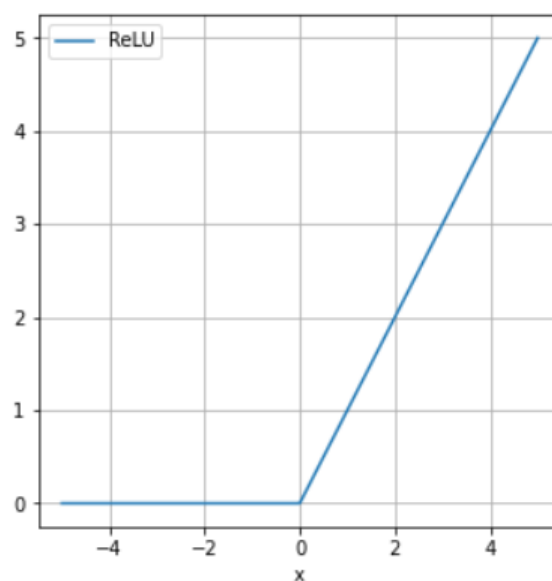
$$(n_h - k_h + p_h + 1) \times (n_w - k_w + p_w + 1), \quad (1.17)$$

kde  $p_h$  je počet řádků paddingu a  $p_w$  je počet sloupců paddingu.

Po vygenerování výstupního obrazu dojde k použití aktivační funkce. Velice často se jako aktivační funkce používá funkce ReLU (Rectified Linear Unit) [7], a to především kvůli jednoduchosti implementace a dobrého výkonu u prediktivních úloh. Definice funkce ReLU je následující:

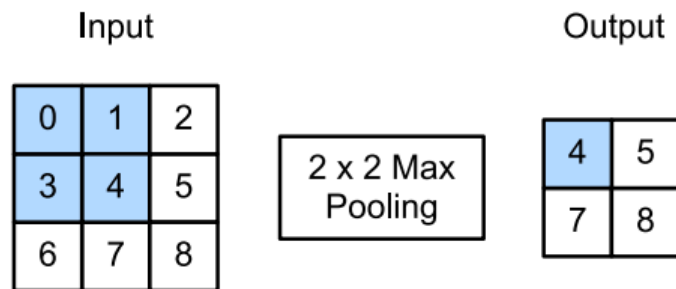
$$ReLU(x) = \max(x, 0). \quad (1.18)$$

Pomocí této funkce lze uchovat pouze kladné hodnoty, díky tomu dojde ke zbavení se záporných hodnot. Graf funkce ReLU je zobrazen na obr. 1.4.



Obrázek 1.4: Graf ReLU funkce.

V pooling vrstvě následně dochází ke zmenšení počtu vstupů a tedy i snížení výpočetních nároků [8]. V této fázi se uplatňuje buď tzv. max pooling (obr. 1.5), anebo average pooling. V případě max poolingů dochází k výběru maximální hodnoty ve vstupní matici. Velikost submatice, respektive oblast výběru maximální hodnoty je závislá na zvolené velikosti okna max poolingů. Vybraná hodnota je následně vepsána na konkrétní pozici výstupní matice. Princip je podobný jako u konvoluce. Proces opět začíná v levém horním rohu vstupní matice a následně dochází k posouvání okna max poolingů [7].

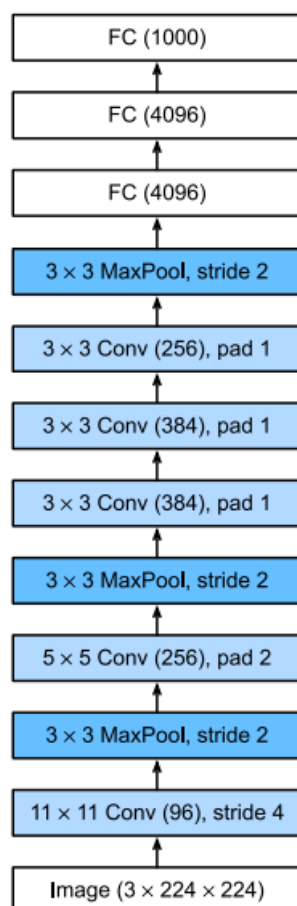


Obrázek 1.5: Schéma vysvětlující funkcionalitu max poolingů. V tomto případě byla velikost okna poolingů  $2 \times 2$ . V tomto schématu je vyznačen počáteční oblast max poolingů. Je vybíráno maximum z čísel 0, 1, 3 a 4. Převzato z [7].

Poslední vrstvou je tzv. fully connected vrstva, ve které se provádí konečná klasifikace. Neurony v této vrstvě jsou mají spojení se všemi aktivacemi v předchozí vrstvě.

Příkladem typické konvoluční neuronové sítě je AlexNet [9], který vyhrál soutěž ImageNet Large Scale Visual Recognition Challenge v roce 2012 a to s velkým náskokem. Tato síť se skládá z 5 konvolučních vrstev, 3 max pooling vrstev a 3 fully connected vrstev. Každá konvoluční vrstva je následována ReLU nelineární aktivací vrstvou [7] [8]. Struktura AlexNetu je znázorněna na obr. 1.6.





Obrázek 1.6: Schéma znázorňující strukturu AlexNetu, jež se skládá z 5 konvolučních vrstev (Conv), 3 max pooling vrstev (MaxPool) a 3 fully connected vrstev (FC). Převzato z [7].

## 1.2.2 Detekce objektů pomocí CNN

Jako jednou z metod počítačového vidění, určených k detekci objektů v obraze, může být považováno použití konvolučních neuronových sítí [8]. Obecně existují dvě možnosti detekce a to buď detekce objektů pomocí bounding boxů (vanilla object detection), anebo segmentace instance. Na rozdíl od segmentace instancí, byla detekce pomocí bounding boxů více studovaná, a tím pádem je považována za více tradiční. Cílem této detekce je lokalizace objektů v obraze a jejich následné označení obdélníkem. Segmentace instancí je naopak relativně nová metoda, která segmentuje každý objekt tzv. pixelovou maskou (pixel-wise mask). Tato metoda má vyšší nároky na zpracování informací obsažených v obraze, jelikož je citlivá na rozdílné zobrazení detekovaných objektů v obraze, např. instance stejného objektu mají v obraze rozdílnou velikost či byly vyfoceny z jiného úhlu, atp.

Detekce objektů v obraze pomocí hlubokého učení lze rozdělit do dvou skupin: two-stage detectors a one-stage detectors [7] [8]. Two-stage detectors nejprve extrahují mnoho (2000) navržených oblastí ze vstupního obrazu. Tyto oblasti jsou označeny obdélníky s různými velikostmi. Z těchto vyznačených oblastí jsou následně získány vlastnosti jednotlivých oblastí.

Získané informace jsou poté použity k predikci třídy a označení daného objektu obdélníkem. Naopak, one-stage detectors okamžitě udělají odhad kategorie objektů na každém místě vstupního obrazu bez použití klasifikačního kroku. One-stage detectors jsou navíc oproti two-stage detectors více časově efektivní a jsou lépe použitelné v reálných aplikacích. Dále se však v této části bude věnováno pouze two-stage detectors.

Two-stage detectors [8] rozdělí úlohu detekce do dvou fází. V první fázi dojde k identifikaci oblastí v obrazu, které mohou být potencionálními objekty. Ve druhé fázi se následně vytvoří predikce identifikovaných oblastí, respektive dojde k určení, do jaké třídy s největší pravděpodobností nalezený objekt náleží. Detekovaná oblast může být buď pozadím obrazu, anebo objektem jedné z předdefinovaných tříd.

Mezi konkrétními příklady two-stage detectors patří např. R-CNN, SPP-net, Fast R-CNN, Faster R-CNN či Mask R-CNN.

**R-CNN** [8] [10] je vůbec prvním two-stage detectorem představeným v roce 2014 Girshickem et al. R-CNN se skládá ze tří částí: (i) vygenerování návrhů, (ii) extrakce příznaků a (iii) klasifikace oblastí. Pro každý obraz nejprve dojde k vygenerování kolem dvou tisíc návrhů tzv. Selective Search, pomocí kterého dojde k vyřazení oblastí, jež mohou být rozpoznány jako pozadí obrazu. Poté je každý návrh oříznut, zmenšen a zakódován do příznakového vektoru (např. 4096 dimenzí). V dalším kroku je použit one-vs-all SVM klasifikátorem. Na závěr dojde k naučení regresorů bounding boxu, a to využitím extrahovaných vlastností. Jednou z výhod tohoto modelu může být to, že R-CNN na začátku odmítne velké množství easy negatives, čímž dojde ke zlepšení rychlosti učení a redukci false positives. Nevýhodou může být např. velká časová náročnost trénování a testování, ta byla způsobena mnoha duplikovanými výpočetními operacemi, způsobené separátním extrahováním vlastností.

**SPP-net** [11] byl uveden Girshickem et al. za cílem zrychlení R-CNN a naučení více diskriminativních vlastností. SPP-net je založen na základě spatial pyramid matching (SPM) [12]. Na rozdíl od R-CNN spočítá SPP-net reprezentaci celého obrazu a následně z této reprezentace získá příznakové vektory fixní délky, a to pomocí tzv. Spatial Pyramid Pooling (SPP) [8] vrstvy. Vrstva SPP umožňuje zpracovávat obrazy v různých měřítkách, bez toho aniž by došlo k jejich zmenšení. Díky tomuto tedy nedojde ke ztrátě informace a geometrickému zkreslení/distorzi. Rovněž použitím SPP-netu došlo k dosažením lepších výsledkům v porovnání s R-CNN. Na druhou stranu, trénování SPP-netu probíhalo stále ve více stupních, kvůli čemuž nemohlo dojít k end-to-end optimalizaci, bylo třeba extra paměť k ukládání extrahovaných vlastností. Další nevýhodou je pak „zmrazení“ parametrů před vrstvou SPP, to bylo způsobeno tím, že u SPP-netu nedochází ke zpětné propagaci gradientů do konvolučních jader.

**Fast R-CNN** [8] [13] byl vyvinut za cílem vyřešit výše zmíněné nevýhody SPP-netu. V prvotních fázích funguje Fast R-CNN podobně jako SPP-net. Na rozdíl od SPP-netu, však Fast R-CNN využívá ROI Pooling vrstvu na extrakci oblastí, jedná se o speciální vrstvu spadající pod SPP, která pracuje s pouze

jedním měřítkem k rozdělení návrhů na pevný počet dílů. Pomocí tohoto modelu je už možné end-to-end optimalizovat klasifikaci oblastí a regresi bounding boxů. Fast R-CNN rovněž dosáhl mnohem lepší přesnosti detekce než R-CNN nebo SPP-net. Došlo také ke zlepšení rychlosti trénování.

**Faster R-CNN** [8] [14] byl vyvinut za použití nového generátoru nazvaném Region Proposal Network (RPN). Tento generátor je konvoluční síť, jež pracuje s obrazem libovolné velikosti a generuje množinu navržených objektů na každé pozici zpracovávaného vstupního obrazu. Tento model byl schopen vytvářet predikce rychlostí 5FPS na GPU. Pomocí tohoto modelu bylo složité detekovat malé objekty. Vylepšeným následovníkem je R-FCN [15].

**FPN** [8] [16] je model používaný např. na detekci objektů ve videu. Pomocí tohoto modelu se podařilo dosáhnout významného zlepšení výsledků při detekci multi-scale objektů.

Použitím **Mask R-CNN** [7] [17] lze dosáhnout nejlepších výsledků při detekci objektů. Tento model je založen na Faster R-CNN. Využívá tzv. region of interest alignment (RoiAlign), na rozdíl od Faster R-CNN, který pracuje s region of interest pooling vrstvou. Vrstva region of interest alignment využívá bilineární interpolaci pro zachování prostorové informace. Mask R-CNN se používá na predikci nejen tříd a bounding boxů, ale i pro zjištění pozic objektů na úrovni pixelů a to prostřednictvím použití další plně konvoluční sítě, jinými slovy predikované objekty jsou zvýrazněny i pomocí masek. Tato konvoluční neuronová síť je vhodná především pro segmentaci instancí konkrétního objektu.

### 1.2.3 Využití

Konvoluční neuronové sítě se používají především na rozpoznávání objektů v obrazech. Mezi reálné aplikace patří např. úloha rozpoznávání chodců [18], analýza rozložení dokumentů pomocí níž je možné extrahovat informace jako např. grafy nebo strukturovaná data z tabulek [19], úloha rozpoznání obličeje [20] či rozpoznání ručně psaného textu [21]. V poslední době se konvoluční neuronové sítě začínají využívat rovněž v oblasti lékařství, ať už se jedná např. o zjišťování diagnózy Parkinsonovy choroby na základě ručních výkresů [22], automatickou diagnózu virového onemocnění COVID-19 z rentgenových snímků hrudi [23] či detekci a následnou klasifikaci rakoviny prsu prostřednictvím mamografických snímků [24].

Hluboké učení se dále může uplatňovat v úlohách týkajících se rozpoznání řeči. Pomocí konvolučních neuronových sítí je možné automaticky převést řečový signál na posloupnost slov [25]. Dále lze pomocí hlubokého učení vytvořit automatický biometrický bezpečnostní systém fungující na základě rozpoznání hlasu, jež je využitelný na identifikaci uživatelů, namísto PIN kódu nebo hesla [26].

Dalšími dvěma oblastmi použití konvolučních neuronových sítí jsou např. marketing nebo autonomní automobily. Do oblasti marketingu lze zařadit úlohu cílené reklamy na základě uživatelského vyhledávání klíčových slov po-

skytnutých službou Google Ads [27]. U autonomních automobilů levelu 2 dochází pomocí neuronových sítí k mapování pixelů získaných z kamery, následně pak může dojít k nějakému zásahu vozidla, např. vozidlo se bez zásahu řidiče vyhne překážce nebo samo změní svou rychlost [28].

#### 1.2.4 Historie

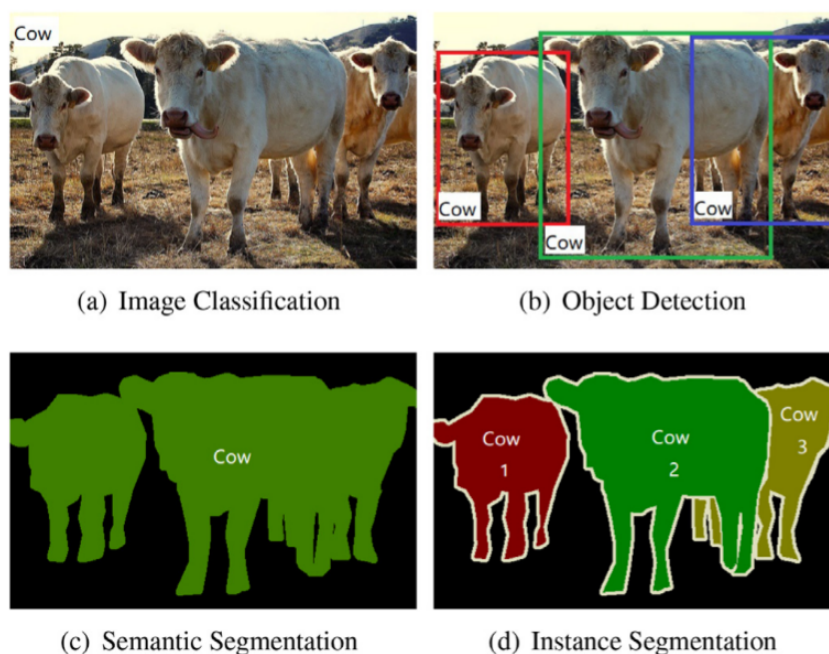
Počátky konvolučních neuronových sítí sahají do konce 60. let 20. století [29], nicméně až do roku 2009 bylo použití konvolučních neuronových sítí limitováno nedostatkem paměťové kapacity a nedostatečným výpočetním výkonem. Obecně lze epocha od roku 1980 až dodnes rozdělit do tří období. První období je charakterizováno především nedostatkem výpočetního výkonu, trénovacích dat a fyzické paměti. Ve druhém období se vědci zabývali vyšší přesností, bez ohledu na počet vrstev, parametrů a neuronů. V této době byl již výpočetní výkon dostatečný, společně s dostatečně velkou fyzickou pamětí, to vedlo k objevu prvního osmivrstvého modelu konvoluční neuronové sítě nazvaného AlexNet [9], do tohoto objevu neměl žádný model tolik vrstev. V poslední době je věnována pozornost zejména mobilním zařízením, která většinou nedisponují tak velkým výpočetním výkonem a fyzickou pamětí jako počítače. Z tohoto důvodu se vědci soustředí na velikost modelu, rychlost, energetickou účinnost a velikost fyzické paměti.

Za první model konvoluční neuronové sítě je považován *Neocognitron* [30], což byl model určený na rozpoznání tvarů. Tento model vycházel z předěšlých získaných poznatků v této oblasti. Dalším modelem byl např. model LeNet-5 [31], což byla neuronová síť navržená pro rozpoznání ručního psaného textu. Celkem se LeNet-5 skládá ze 7 vrstev a je definován jako „Gradient-based“ algoritmus.

Mezi lety 2010 a 2015 došlo poté k velkému rozvoji konvolučních neuronových sítí. Rozmach byl způsoben především tím, že v tomto období došlo k velkému pokroku zejména v oblasti počítačového výkonu. Díky tomu bylo možné vyvinout nové architektury, pomocí kterých bylo možné docílit přesných výsledků, mnohdy přesnějších, než kterých by dosáhl člověk. Novou architekturu reprezentoval již výše zmíněný *AlexNet* [9], který byl vyvinut ve frameworku ImageNet Large-Scale Visual Recognition Challenge 2012. Jednalo se o první konvoluční neuronovou síť, jež měla 60 miliónů parametrů a 500 tisíc neuronů. Celkem tato architektura obsahovala 8 vrstev. Další architekturou byl *ZFNet* [32], což byl model, který obsahoval stejně vrstev jako AlexNet [9], došlo však ke snížení chybovosti. Následně došlo ke zobecnění AlexNetu zvýšením počtu vrstev z 8 na 16 až 19 s pouze několika parametry, tím došlo k vytvoření *VGGNetu* [33]. Tímto opět došlo ke snížení chybovosti. V roce 2014 byl vyvinut *GoogLeNet* [34], což byla neuronová síť skládající se z 22 vrstev. Tato architektura byla zcela odlišná od ostatních architektur. Významným milníkem bylo vyvinutí *ResNetu* [35] s hloubkou až 152 vrstev. Pomocí této architektury se podařilo dosáhnout chybovosti pouze 3.57%, jednalo se o první případ, kdy neuronová síť měla menší chybovost než člověk. Tuto architekturu využívá např. Mask R-CNN.

## 1.2.5 Významné datasety spojené s rozpoznáním objektů

V obecném slova smyslu jsou datasety důležité pro trénování a validaci jednotlivých algoritmů spojené s počítačovým viděním. Datasety spojené s rozpoznáním objektů [36] je možné rozdělit do 4 skupin: klasifikace objektů, detekce objektů, sémantické označení objektů a segmentace instancí. Cílem klasifikace objektů je zjistit, zda se objekt nachází/nenachází v obraze. Prvotní datasety z této skupiny obsahovaly pouze jeden objekt s prázdným pozadím (MNIST handwritten digits) [37]. Úkolem detekce objektů je zařazení objektu do určité třídy a lokalizace objektu v obraze. K lokalizaci objektu se často využívá bounding box. Cílem sémantického označení objektu ve scéně je označit každý pixel scény, který je poté zařazen do konkrétní kategorie, příkladem může být např. židle, nebe, skupina lidí, atp. Úkolem segmentace instancí je označit jednotlivé rozlišitelné objekty nacházející se v obraze.



Obrázek 1.7: Porovnání jednotlivých úloh rozpoznání. (a) klasifikace objektů pouze zjišťuje, jaké objekty se v obraze nachází, (b) detekce objektů označí objekty pomocí bounding boxu, (c) sémantická segmentace predikuje kategorie jednotlivých pixelů, bez toho aniž by došlo k rozlišení jednotlivých instancí objektů, (d) segmentace instancí detekuje jednotlivé rozlišitelné objekty. Převzato z [8].

**ImageNet** je velmi rozsáhlá obrazová databáze, poprvé představená v roce 2009. ImageNet [38] využívá hierarchickou strukturu WordNetu [39], což je velká lexikální databáze anglických slov. Ve WordNetu jsou podstatná jména, slovesa, přídavná jména a příslovce sdružena do množin synonym (z angl. synsets). Celkem okolo 80 000 synonym je součástí WordNetu. Zpočátku bylo cílem autorů ImageNetu poskytnout v průměru 500 až 1000 označených obrazů reprezentujících každé synonymum, dnes už reprezentujících obrazů mohou být řádově tisíce. Do budoucna se počítá, že celkem databáze ImageNet bude čítat desítky miliónů jasně zařaditelných obrazů.

**Pascal Visual Object Classes (VOC) Challenge** [40] se skládá ze dvou částí: z každoročně konané soutěže, týkající se rozpoznání objektů v obraze a veřejně přístupného datasetu. Dataset VOC 2007 se skládá z anotovaných fotografií získaných z Flickru. Od roku 2006 byl každý rok vydán dataset se zcela novými anotacemi. Celkem se dataset skládá z 500 tisíc fotografií získaných z již zmíněného Flickru a to pomocí souvisejících klíčových slov. Celkem je možné anotace zařadit do 20 tříd, s tím, že třídy spadají pod 4 hlavní větve (vozidla, zvířata, domácnost a lidé). Každý objekt nacházející se v obraze je ohraničen bounding boxem a obsahuje název třídy, do které náleží. Poslední verze PASCAL VOC datasetu pochází z roku 2012.

**Microsoft Common Objects in COntext (MS COCO)** dataset [36] obsahuje celkem 91 kategorií, z nichž 82 zahrnuje přes 5 000 vyznačených instancí. Celkem je součástí datasetu 2,5 miliónu instancí v 328 tisících obrazech. Ve srovnání s ImageNet datasetem má COCO méně kategorií [38], ale na druhou stranu každá kategorie obsahuje více instancí. Jelikož autory tohoto datasetu zajímala zejména přesnost lokalizace instancí objektů, rozhodli se do datasetu zahrnout pouze kategorie, do kterých patří instance, které mohou být snadno označeny, např. auto, kolo, židle atp. Nebrali tedy ohled na kategorie obsahující materiály anebo na objekty bez jasných hranic, např. obloha nebo ulice. Výběr navržených kategorií nejprve probíhal tak, že autoři zkombinovali kategorie z datasetu PASCAL VOC [40] a podmnožiny 1 200 nejčastěji používaných slov, pomocí nichž jsou označovány identifikovatelné objekty [41]. V dalším kroku autoři požádali děti od 4 do 8 let, aby vyjmenovaly objekty, které vidí ve vnitřních a venkovních prostředích. Po několika dalších procesech nakonec autoři vybrali z prvotních 272 kandidátů 91. Výsledný dataset byl vytvořen pomocí AMT (Amazon Mechanical Turk). Celý proces vytvoření se skládal ze tří kroků. Nejprve byly v obraze označeny kategorie, následně byly zvýrazněny jednotlivé instance předtím vyznačených kategorií a na závěr byla provedena segmentace instancí.

## 1.3 Detectron2

Platforma Detectron [42] [43] je open source knihovna vyvinutá Facebook AI Research (FAIR), jež byla představena v roce 2018. Vývoj Detectronu však započal již v roce 2016. Jedná se o knihovnu, jejímž cílem je detekce objektů v obrazu a segmentace.

Detectron2 [43] je zcela přepracovaný původní framework Detectron. Detectron2 je na rozdíl od svého předchůdce založen na Python knihovně PyTorch. Kromě nového designu, umožňuje Detectron2 modularitu, tím pádem lze dosáhnout jasného rozdělení jádra Detectronu2 a vlastního zdrojového kódu. Součástí Detectronu2 jsou modely Faster R-CNN, Mask R-CNN, RetinaNet a DensePose (tyto modely byly rovněž součástí Detectronu), navíc jsou k dispozici modely nové jako např. Panoptic FPN nebo TensorMask. Jelikož výpočty související s trénováním probíhají na grafické kartě, došlo ke zrychlení oproti původnímu Detectronu. Navíc je možné jednoduše distribuovat trénování na několik GPU serverů, což se týká především velkých datasetů. Detectron2 podporuje úlohy spojené s detekcí objektů. Je podporována detekce objektů s bounding boxy a segmentace instancí, stejně jako predikce lidské polohy, tyto úlohy byly podporovány už původním Detectronem. Detectron2 navíc nově přidává podporu pro sémantickou segmentaci a panoptickou segmentaci, což je úloha kombinující sémantickou segmentaci společně se segmentací instancí. Jedná se o jeden z nejvíce rozšířených projektů organizace FAIR.

## 1.4 Mikroskopické histologické obrazy

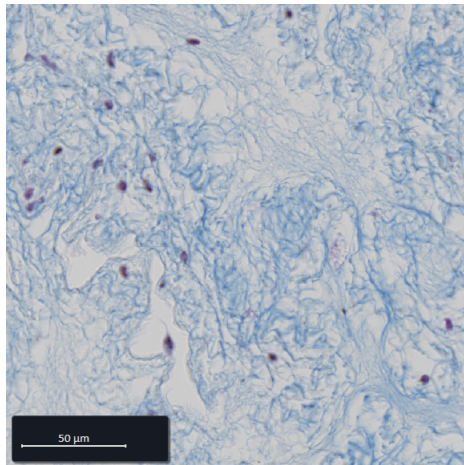
Jako výchozí data důležitá pro návrh nového segmentačního nástroje byly použity mikroskopické histologické obrazy, již byly poskytnuty Biomedicínským centrem v Plzni. Každý obraz zobrazuje decelularizovaný skelet prasečích jater (scaffold), který byl předtím implantován do tukové tkáně v břišní dutině pokusného zvířete. Celkem bylo poskytnuto 17 takovýchto obrazů.

### 1.4.1 Zpracování fyzického vzorku

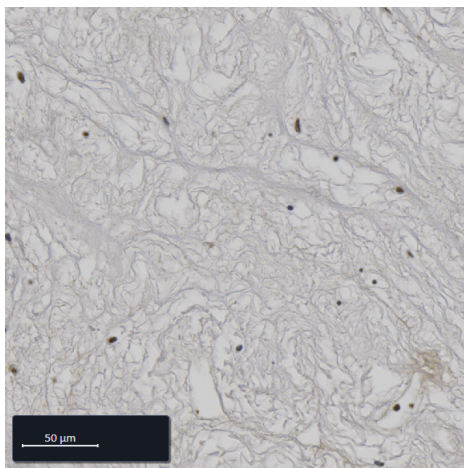
Nejprve muselo dojít k vyjmutí implantátu ze zvířete, které se nacházelo v celkové anestezii. Následně musí být vzorek fixován 4% formaldehydem, tak aby nedocházelo k autolýze [44], což je jev, kterému podléhají tkáně a orgány po zástavě přísunu kyslíku. Nežádoucí proces autolýzy je způsobován samonatrávením buněk jejich vlastními uvolněnými enzymy. Fixace je cílený děj, pomocí kterého lze zabránit procesu autolýzy. Dojde k zastavení činnosti enzymů a rovněž k usmrcení bílkovin ve tkáni pomocí denaturace. Pomocí fixace rovněž dojde k k usmrcení mikroorganismů. Dobře fixovaná tkáň je považována jako jeden z nezbytných předpokladů kvalitního histologického zpracování.

V dalším kroku je implantát zalit do parafínového bloku, který je následně nakrájen na velmi tenké řezy pomocí mikrotomu. Poté dochází k obarvení vzorku na mikroskopických sklíčkách. K tomuto účelu jsou používány tři druhy barvení (obr. 1.8): Gomori trichrom, Imunohistochemické barvení a H&E barvení. Pomocí barvení *Gomori trichrom* získáme sytě modrý skelet a fialová jádra buněk. *Imunohistochemické barvení* společně s přibarvovaným hematoxylinem způsobí tmavě modrou barvu jader buněk. Světlejší modrou barvou je obarvena okolní tkáň, v našem případě cytoplazma. Extracelulární matrix (ECM), kam patří proteiny vně buněk a decelularizovaný skelet je zbarvený velmi světle modře až šedivě. *H&E barvení* je kombinací hematoxylinu a eosinu. Jádra jsou v tomto případě tmavě modrá cytoplazma buněk je růžovo modrá (do fialova), čistě růžový je pak ECM (decelularizovaný skelet).

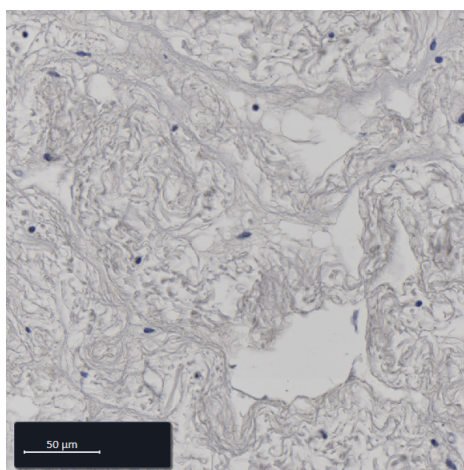




a) Barvení Gomori trichrom



b) Imunohistochemické barvení



c) H&E barvení.

Obrázek 1.8: Ukázky jednotlivých barvení.

## 1.4.2 Digitalizace vzorku

Preparát je naskenován pomocí scanneru ZEISS Axioscan, tím pádem získáme digitalizovaný fyzický vzorek, který je možné si uložit v počítači a následně s ním dále pracovat. Rozlišení získaných obrazů je velmi vysoké, např. rozlišení jednoho konkrétního obrazu je  $137045 \times 31512$  pixelů, jedná se tedy o velice kvalitní obrazy, co se detailů týče. Tím pádem jsou však tyto obrazy paměťově náročné. Jejich velikost se pohybuje v řádu stovek MB až jednotek GB. Navíc jsou digitalizované vzorky uloženy ve speciálním formátu .czi, což je proprietární formát společnosti ZEISS určený právě pro ukládání mikroskopických obrazů.

## 1.5 ScaffAn

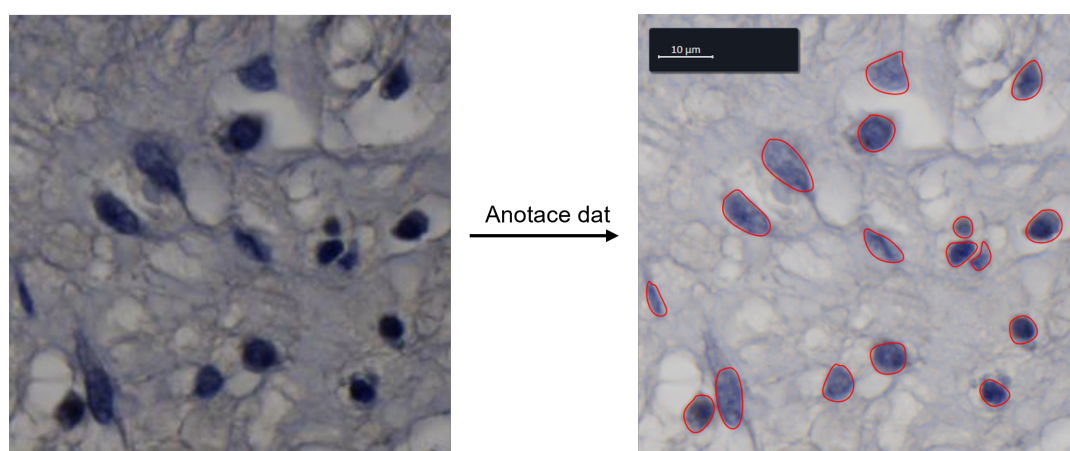
ScaffAn [45] je otevřená aplikace vytvořená, kvůli účelu posouzení mikroskopických histologických obrazů ve vysokém rozlišení. Vstupní obrazy jsou obarvené H&E, čímž dojde k rozdělení decelularizovaných částí od prasečích jater. Aplikace byla napsána v programovacím jazyku Python (verze 3.6), a to především kvůli velké rozšířenosti tohoto programovacího jazyka. Při vývoji tohoto této aplikace byly použity Python moduly jako např. Numpy, Scipy, Pandas, scikit-image a scikit-learn, které zajišťují základní výpočetní operace spojené s poli, statistiku, operace s tabulkami, zpracování obrazů a strojové učení. Autor aplikace Ing. Miroslav Jiřík, Ph.D. věří, že díky otevřenosti této aplikace, bude nástroj ScaffAn využitelný i pro analýzu decelularizovaných skeletů jiných živočišných druhů, či dokonce pro analýzu scaffoldů jiných orgánů.

Repozitář se zdrojovými kódy je volně dostupný prostřednictvím platformy Github (odkaz: <https://github.com/mjirik/scaffan>).

## 2. Návrh segmentačního nástroje

### 2.1 Anotace dat

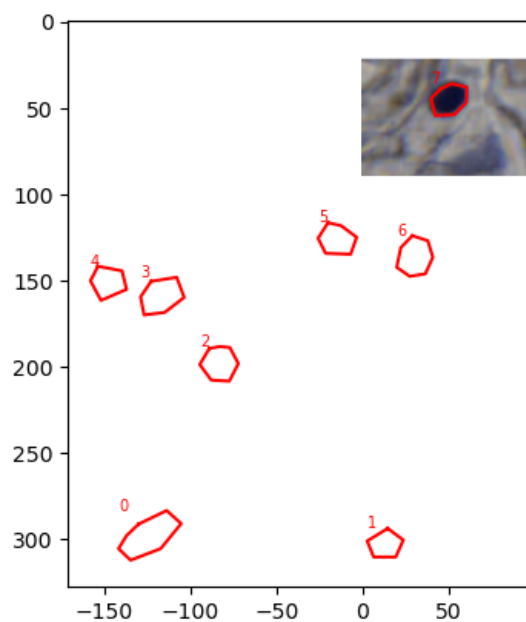
Jádra jaterních buněk byla ručně anotována v grafickém programu ZEISS ZEN (obr. 2.1). Tento program byl zvolen, kvůli již zmíněnému proprietárnímu formátu .czi, v němž byly mikroskopické obrazy poskytnuty. V programu ZEISS ZEN může uživatel anotovat data pomocí obdélníků, kruhů nebo Beziérových křivek. V našem případě byly však pro anotaci buněk zvoleny Beziérové křivky, zejména kvůli dalšímu zpracování anotovaných dat.



Obrázek 2.1: Proces anotace jader buněk v programu ZEISS ZEN.

### 2.2 Extrakce anotací

V dalším kroku bylo třeba anotovaná data zpracovat. K tomuto účelu byl v programovacím jazyce Python vytvořen pomocný skript, který nejprve načte soubor s anotacemi ve formátu .czi, ze kterého následně vytvoří metadata souboru ve formátu .xml. Soubor xml obsahuje např. informace o nastavení kamery, pomocí níž byl pořízen původní .czi soubor ještě bez anotací. Dále je však možné v xml souboru nalézt souřadnice bodů jednotlivých Beziérových křivek. Souřadnice jsou definovány v pixelech a jsou získávány pomocí Python modulu xml.dom. Díky již implementovaným metodám v nástroji *ScaffAn* a získaným bodům je možné zobrazit anotovaná data přímo v Pythonu ve formátu .jpg, a to pomocí modulu matplotlib.pyplot. Uživatel si může zvolit jakou anotaci chce zobrazit. Metody potřebné k extrakci anotací byly následně přidány do nástroje *ScaffAn*.



Obrázek 2.2: Zobrazení konkrétní anotace pomocí nástroje ScaffAn a knihovny matplotlib.

## 2.3 COCO dataset

Kvůli použití frameworku Detectron2 bylo třeba vytvořit vlastní COCO dataset, který se skládá ze dvou částí: souboru ve formátu .json a obrazů bez anotací. Všechny informace týkající se anotací jednotlivých obrazů jsou obsaženy, v již zmíněném, .json souboru. Struktura .json souboru je definována dokumentací. K vytvoření vlastního specifického COCO datasetu byly vytvořeny dva pomocné skripty, opět v programovacím jazyku Python.

Cílem prvního skriptu je převedení výstupních .czi souborů, vytvořené v programu ZEISS ZEN, na soubory ve formátu .jpg. Jak již bylo zmíněno výše, vytvořené .jpg soubory už neobsahují na rozdíl od .czi souborů žádné viditelné anotace. První část datasetu je tedy tímto připravena.

Hlavním úkolem druhého skriptu je vytvoření správné struktury .json souboru na základě dokumentace. COCO formát [46] definuje několik typů anotací - object detection, keypoint detection, stuff segmentation, panoptic segmentation, densepose a image captioning, v našem případě byly použity anotace detekující objekty (object detection). Všechny druhy anotací používají stejnou základní datovou strukturu. Základní struktura obsahuje základní informace týkající se datasetu (rok vytvoření, verze nebo autora). Součástí je dále seznam obsahující informace o jednotlivých obrazech obsahující anotovaná data. Obsah seznamu obsahující anotace je popsán níže.

```

1  {
2      "info": info,
3      "images": [image],
4      "annotations": [annotation],
5      "licenses": [license],
6  }
7
8  info{
9      "year": int,
10     "version": str,
11     "description": str,
12     "contributor": str,
13     "url": str,
14     "date_created": datetime,
15 }
16
17 image{
18     "id": int,
19     "width": int,
20     "height": int,
21     "file_name": str,
22     "license": int,
23     "flickr_url": str,
24     "coco_url": str,
25     "date_captured": datetime,
26 }
27
28 license{
29     "id": int,
30     "name": str,
31     "url": str,
32 }

```

Listing 2.1: Základní datová struktura jednotlivých typů anotací. Převzato z [46].

Jak již bylo zmíněno, v našem případě jsou používány anotace detekující objekty (object detection). Každá instance anotovaného objektu obsahuje: *id* anotovaného objektu, *id* obrazu, do kterého objekt náleží, *id* kategorie, tj. třída, do které objekt náleží (v našem případě je kategorie pouze jedna), *segmentační masku* objektu ve formátu polygonu, body polygonu jsou definovány vždy x-ovou a y-ovou souřadnicí v pixelech. Jednotlivé souřadnice polygonu segmentační masky jsou totožné s body souřadnic beziérových křivek (viz výše), jež jsou v tomto případě získané pomocí předtím implementovaných metod v nástroji *ScaffAn*. Dalšími údaji jsou pak *obsah* plochy segmentační masky, *souřadnice bodů bounding boxu* objektu. Atribut

*iscrowd* obsahuje informaci o tom, zda instance reprezentuje pouze jeden objekt nebo kolekci více objektů. Může nabývat hodnot 0 nebo 1. V závislosti na hodnotě tohoto atributu jsou buď použity polygony (*iscrowd* = 0), anebo RLE (*iscrowd* = 1).

```
1 annotation{
2     "id": int,
3     "image_id": int,
4     "category_id": int,
5     "segmentation": RLE or [polygon],
6     "area": float,
7     "bbox": [x,y,width,height],
8     "iscrowd": 0 or 1,
9 }
```

Listing 2.2: Datová struktura object detection anotací. Převzato z [46].

Následně je třeba definovat pole „categories“ jednotlivých objektů, složené z *id* kategorie, *názvu* kategorie a *superkategorie*. Naše úloha obsahuje pouze jednu kategorii s názvem „cell nuclei“.

```
1 categories[{
2     "id": int,
3     "name": str,
4     "supercategory": str,
5 }]
```

Listing 2.3: Datová struktura categories. Převzato z [46].

```
1 {
2     "info": {
3         "year": "2022",
4         "version": "1.0",
5         "description": "COCO dataset for scaffan",
6         "contributor": "Jan Burian",
7         "date_created": "31/01/2022"
8     },
9     "images": [
10         {
11             "id": 0,
12             "width": 291,
13             "height": 291,
14             "file_name": "0000.jpg"
15         },
16         {
17             "id": 1,
18             "width": 367,
19             "height": 367,
20             "file_name": "0001.jpg"
```

```

21     },
22 ],
23 "categories": [
24   {
25     "id": 1,
26     "name": "cell nuclei",
27     "supercategory": "cell nuclei"
28   }
29 ],
30 "annotations": [
31   {
32     "id": 1,
33     "image_id": 0,
34     "category_id": 1,
35     "segmentation": [
36       [
37         133.08106530508618,
38         265.72168004966426,
39         138.3076968989222,
40         262.84703267305395,
41         146.01697849983057,
42         263.36969583243774
43       ]
44     ],
45     "area": 237.313789266831,
46     "bbox": [
47       127.0,
48       262.0,
49       21.0,
50       16.0
51     ],
52     "iscrowd": 0
53   },
54   {
55     "id": 2,
56     "image_id": 0,
57     "category_id": 1,
58     "segmentation": [
59       [
60         82.17212192486797,
61         129.95733499054938,
62         87.71235141433367,
63         128.91200867178176,
64         93.04351564004672,
65         132.15252025996028
66       ]

```

```
67     ],
68     "area": 132.92235672529932,
69     "bbox": [
70         80.0,
71         128.0,
72         15.0,
73         13.0
74     ],
75     "iscrowd": 0
76 },
```

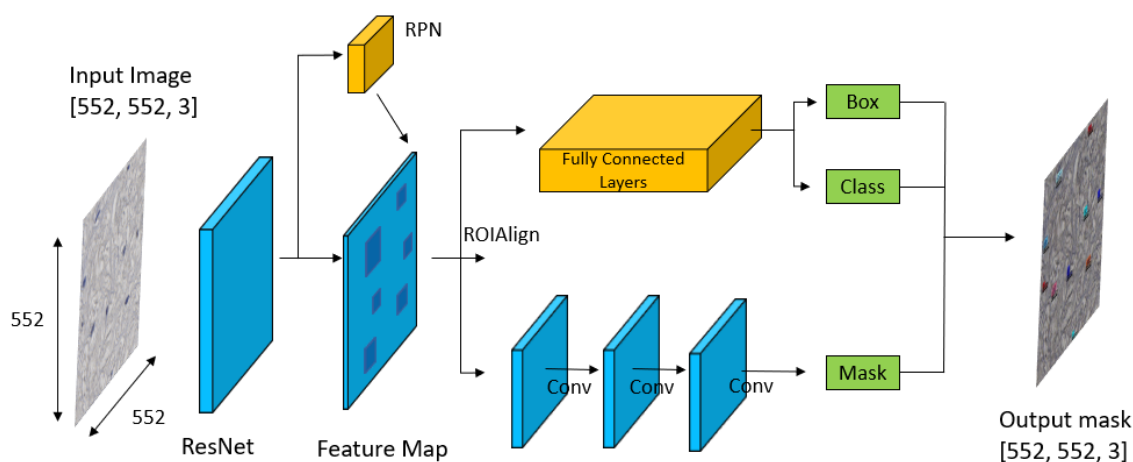
Listing 2.4: Zkrácený a ilustrační výstupní soubor .json. Dataset v tomto případě obsahuje dva obrazy, jednu kategorii a dvě anotované instance objektu v prvním obrazu.



## 2.4 Experiment

### 2.4.1 Použitá konvoluční neuronová síť

Pro potřeby tohoto experimentu byla použita v současnosti nejlepší konvoluční neuronová síť - Mask R-CNN [17] [47], která vznikla rozšířením Faster R-CNN a to tak, že byla přidána větev pro predikci masky instance, souběžně s již existující větví pro rozpoznání bounding boxu. Architektura Mask R-CNN je zobrazena na obr. 2.3. Mask R-CNN je určena především k segmentaci instancí konkrétního objektu, což je výhodné i v této úloze. V našem případě je instance reprezentována jedním buněčným jádrem. Tato síť vytváří bounding box a masku pro každou instanci objektu v daném obraze. Mask R-CNN je založeno na architektuře ResNet [35].



Obrázek 2.3: Schéma popisující architekturu konvoluční neuronové sítě Mask R-CNN.

Pro detekci objektů využívá Mask R-CNN nejprve tzv. anchor boxy (obr. 2.4). Anchor boxy jsou tvořeny množinou předem definovaných bounding boxů s určitou výškou a šířkou. Tyto boxy jsou nastaveny tak, aby zachycovaly měřítko a poměr stran konkrétních tříd objektů, které chceme predikovat. Mask R-CNN během tohoto procesu udělá tisíce predikcí, tak aby našla objekty nebo instance objektů v obraze. Finální detekce objektů vznikne odstraněním anchor boxů, které tvoří pozadí obrazu, zbylé anchor boxy jsou filtrovány na základě tzv. confidence score. Objekty jsou zvýrazněny pomocí bounding boxů.



Obrázek 2.4: Vizualizace využití anchor boxů. Převzato z [47].

Následně dojde k vygenerování masek jednotlivých detekovaných instancí objektů. Ještě před umístěním masek na správná místa v obraze, musí dojít k nastavení jejich správného měřítka. V posledním kroku je vytvořen konečný výsledek, a to na základě složení výsledků z předešlých kroků.

### 2.4.2 Popis experimentu

Experiment spočíval v tom, že vytvořený dataset, který byl vytvořen ze vzorků obarvených H&E, byl nejprve rozdělen do tří částí:

- Trénovací část
- Validací část
- Testovací část

Celkem dataset obsahoval 166 obrazů. Trénovací část tvořila 60% celkového datasetu (100 obrazů s anotacemi). Validací část tvořila stejně jako testovací 20% celkového datasetu (33 obrazů s anotacemi, respektive 33 obrazů bez anotací).

Trénovací část obsahovala celkem 1299 instancí kategorie „cell nuclei“. Validací část pak obsahovala celkem 380 instancí kategorie „cell nuclei“. Z obou těchto částí byly vytvořeny 2 COCO datasety. Testovací část byla reprezentována pouze obrazy ve formátu .jpg.

Použitím zmíněných dvou COCO datasetů (trénovací a validací) byly následně natrénovány nové modely, které byly založené na jednotlivých předtrénovaných COCO Instance Segmentation modelech z Detectron2 Zoo [48]

(tab. 2.1). Použité modely byly V experimentech byly použité modely založené na třech rozdílných páteřních (backbone) kombinacích - FPN, C4 a DC5.

Name	Ir sched	Train time (s/iter)	Inference time (s/im)	Train mem (GB)	Box AP	Mask AP
R50-C4	3x	0.575	0.111	5.2	39.8	34.4
R50-DC5	3x	0.470	0.076	6.5	40.0	35.9
R50-FPN	3x	0.261	0.043	3.4	41.0	37.2
R101-C4	3x	0.652	0.145	6.3	42.6	36.7
R101-DC5	3x	0.545	0.092	7.6	41.9	37.3
R101-FPN	3x	0.340	0.056	4.6	42.9	38.6

Tabulka 2.1: Použité předtrénované COCO Instance Segmentation modely, které využívají Mask R-CNN. Převzato z [48].

Testovací část byla použita jako předběžná kontrola správnosti predikce detekovaných buněčných jader. Na závěr byla vyhodnocována úspěšnost detekce buněčných jader jednotlivých natrénovaných modelů.

### 2.4.3 MetaCentrum

Veškeré výpočty spojené s experimentem byly prováděny na virtuálních strojích, které jsou součástí infrastruktury MetaCentra [49], spadajícího pod organizaci Cesnet. Právě pomocí infrastruktury MetaCentra byly vytvořeny nové modely, které již byly natrénované na detekci buněčných jader. Bezplatné služby MetaCentra jsou otevřeny všem akademickým pracovníkům, zaměstnancům a studentům vědeckovýzkumných institucí v České republice. Pro potřeby experimentu byly nejčastěji využívány virtuální stroje spadající pod cluster *adan.grid.cesnet.cz* [50] [51], který je díky svým parametrům určen pro úlohy z oblasti strojového učení. Celkem tento cluster obsahuje 61 uzlů, přičemž každý z nich disponuje následující hardwarovou specifikací (tab. 2.2):

<b>CPU</b>	32× Intel(R) Xeon(R) Gold 5218 CPU @ 2.30GHz
<b>RAM</b>	192 GiB
<b>Disk</b>	4× 240GB SSD
<b>GPU</b>	2× GPU nVidia Tesla T4 (BIOCEV)
<b>Net</b>	Ethernet 1Gbit/s, Omni-Path
<b>Vlastník</b>	Cesnet

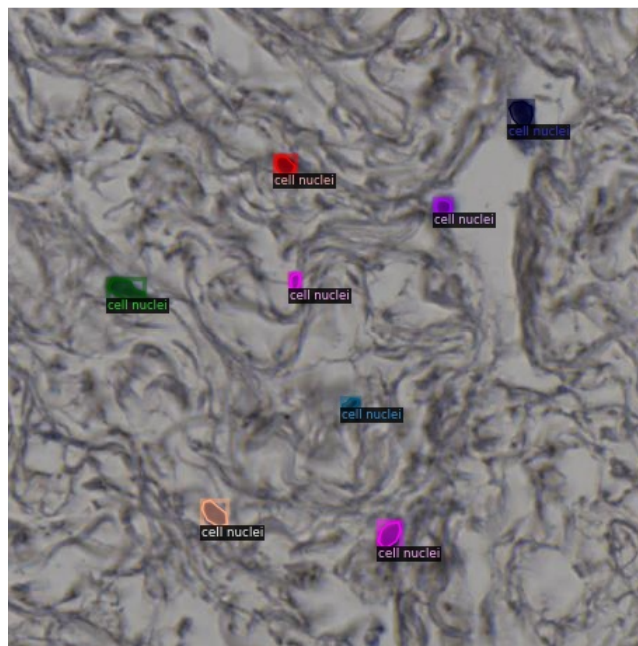
Tabulka 2.2: Hardwarová specifikace uzlu spadajícího pod *adan.grid.cesnet.cz*.

Jelikož se v našem případě jedná o dávkovou úlohu, byly kvůli pohodlnému spouštění experimentu vytvořeny dva skripty - shellový skript a Python skript.

V shell skriptu byly nejprve definovány požadavky týkající se chtěného virtuálního stroje, např. počet procesorů, počet grafických karet, velikost operační paměti, velikost fyzické paměti a doba, na kterou uživatel daný stroj potřebuje. Poté se ve skriptu nastavují cesty k jednotlivým adresářům

(LOGDIR, PROJECTDIR, DATADIR). V našem případě se tyto adresáře nachází v domovské složce uživatele MetaCentra. Do adresáře LOGDIR se následně uloží logy po ukončení úlohy. V adresáři PROJECTDIR se nachází Python skript, jež chceme na virtuálním stroji spustit. V adresáři DATADIR lze nalézt data potřebná pro trénování, validaci či testování neuronové sítě. Následně se vytváří složky v uživateli přiděleném SCRATCHDIR, což je fyzické úložiště přímo na virtuálním stroji. Poté se vytvoří nová složka v adresáři DATADIR. V dalším kroku se zkopírují data z uživatelovy složky DATADIR do adresáře SCRATCHDIR, tak aby k datům měl přístup i uživatel přidělený virtuální stroj. Následně dojde k přidání potřebných modulů. Téměř na závěr se aktivuje Python prostředí, v našem případě Python ve verzi 3.6 s předinstalovaným modulem ScaffAn. Aktivací správného Python prostředí dojde ke spuštění Python skriptu v adresáři PROJECTDIR. Na závěr už jen dojde ke kopírování dat ze SCRATCHDIR do DATADIR, odkud má uživatel přístup k výsledkům.

Python skript nejprve obsahuje importy modulů z frameworku Detectron2 i obecných modulů jazyka Python. Tyto moduly jsou potřebné pro bezchybný běh skriptu. Dále jsou do proměnných ukládány cesty k adresářům obsahující potřebná data, rovněž probíhá kontrola existence jednotlivých cest. V dalším kroku dochází k registraci COCO instancí trénovací a validační části datasetu a jsou získána metadata trénovací části. Následující řádky kódu slouží k optické kontrole správnosti anotovaných trénovacích dat, ty je totiž možné zobrazit ve formátu .jpg včetně viditelných barevných anotací (obr. 2.5). Tímto je možné zkontrolovat, zda je správně vygenerovaný .json soubor, či zda nedochází k nějaké jiné chybě.



Obrázek 2.5: Vizualizace trénovací části datasetu.

Dále se ve skriptu nastavují parametry týkající se trénování neuronové sítě a to např. jaké datasety se mají použít ke trénování či validaci, počet

iterací či jaký model z Detectron2 Model Zoo se má k trénování neuronové sítě použít.

V závěrečné části skriptu dochází pouze ke kontrolní predikci buněčných jader (obr. 2.6) a jejich následné uložení do adresáře. K predikci jader je již využíván natrénovaný model přímo na detekci buněčných jader z předchozího kroku.



Obrázek 2.6: Vizualizace predikce buněčných jader.

#### 2.4.4 Hardwarové parametry experimentu

- 1 CPU Intel(R) Xeon(R) Gold 5218 CPU @ 2.30GHz
- 1 GPU z GPU nVidia Tesla T4 clusteru
- 10 GB RAM
- 1 GB fyzické paměti

## 2.4.5 Trénování modelu neuronové sítě

Takto nastavené parametry byly použity pro trénování všech šesti modelů neuronové sítě:

- `cfg.DATALOADER.NUM_WORKERS = 2`
- `cfg.SOLVER.IMS_PER_BATCH = 2`
- `cfg.SOLVER.BASE_LR = 0.0002`
- `cfg.SOLVER.MAX_ITER = 60000`
- `cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 128`
- `cfg.MODEL.ROI_HEADS.NUM_CLASSES = 1`

Před natrénováním nového modelu musely být rovněž načteny váhy modelu, které musely být uloženy do proměnné `cfg.MODEL.WEIGHTS` a konfigurační soubor příslušného modelu ve formátu `.yaml`. Váhy modelu byly inicializovány pomocí Detectron2 Zoo ([48]). Jako základ pro každý nový model posloužil již předtrénovaný model z Detectron2 Zoo ([48]). Poté již mohlo být spuštěno trénování pomocí příkazu `DefaultTrainer(cfg).train()`. K natrénování nového modelu byly vždy použity dva COCO datasety - jeden trénovací a druhý validační (více viz 2.4.2). Nejdůležitějším výstupním souborem po trénování byl model, který už byl natrénován přímo na detekci buněčných jader.

Při trénování na rozdíl od predikce je výstupem vždy natrénovaný model. Navíc je operace trénování většinou časově náročná. U predikce už je pak možné pouze použít váhy daného natrénovaného modelu a predikovat konkrétní data, tím pádem je celý tento proces řádově rychlejší.

## 2.5 Vyhodnocení experimentu

### 2.5.1 Doby trénování

Nejprve byly změřeny doby trénování získaných modelů. Hodnoty byly zaneseny do tabulky 2.3. Všechny modely byly trénovány na strojích se stejnými předem definovanými parametry viz předešlá sekce 2.4.4. Průměrný čas trénování byl 8:44:10. Nejrychleji byl natrénován model založený na mask\_rcnn\_R\_50\_FPN\_3x.

Model	Celková doba trénování
mask_rcnn_R_101_C4_3x	10:41:32
mask_rcnn_R_101_DC5_3x	8:49:18
mask_rcnn_R_101_FPN_3x	8:30:11
mask_rcnn_R_50_C4_3x	9:17:44
mask_rcnn_R_50_DC5_3x	9:07:35
mask_rcnn_R_50_FPN_3x	5:58:39

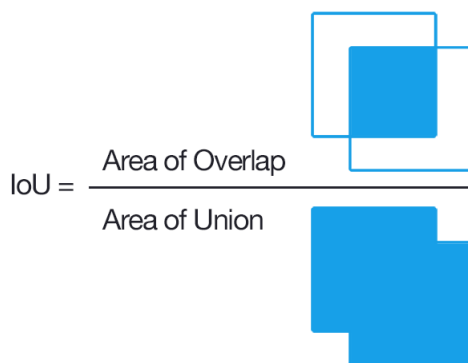
Tabulka 2.3: Doby trénování získaných modelů.

### 2.5.2 Intersection over Union

Na závěr bylo provedeno vyhodnocení jednotlivých modelů za použití metriky Intersection over Union (známé také jako Jaccard index) (obr. 2.7). Výsledky Intersection over Union každého modelu byly získány pomocí třídy COCOEvaluator, která je součástí frameworku Detectron2. IoU může být definováno pomocí následujícího vzorce:

$$IoU(A, B) = \frac{A \cap B}{A \cup B}, \quad (2.1)$$

kde  $A$  je množina predikovaných pixelů a  $B$  je množina skutečných pixelů objektu.



Obrázek 2.7: Grafická interpretace metody Intersection over Union. Převzato z [52].

Získané výsledky byly zaneseny do tabulek 2.4 a 2.5. Hodnoty AP (Average Precision) byly vypočteny jako  $IoU = .50 : .05 : .95$ , což znamená, že byly použity prahy od 0.5 do 0.95 s krokem 0.05, celkem tedy bylo vypočteno 10 hodnot, ze kterých byl vypočten průměr. Hodnoty AP50 jsou definovány jako  $IoU = .50$ , hodnota prahu je v tomto případě fixována na hodnotě 0.5. Stejným způsobem jako hodnoty AP50 byly získány hodnoty AP75. Hodnoty APs reprezentují přesnost detekce pro malé objekty. Definice jednotlivých hodnot AP převzaty z [53].

<b>Model</b>	<b>AP</b>	<b>AP50</b>	<b>AP75</b>	<b>APs</b>
mask_rcnn_R_101_C4_3x	<b>47.221</b>	89.519	43.697	47.221
mask_rcnn_R_101_DC5_3x	44.882	90.283	36.528	44.890
mask_rcnn_R_101_FPN_3x	45.356	89.294	42.831	45.384
mask_rcnn_R_50_C4_3x	46.097	91.110	42.249	<b>46.141</b>
mask_rcnn_R_50_DC5_3x	46.046	91.500	<b>43.216</b>	46.063
mask_rcnn_R_50_FPN_3x	45.276	<b>91.916</b>	36.967	45.285

Tabulka 2.4: Hodnoty Intersection over Union pro bounding boxy.

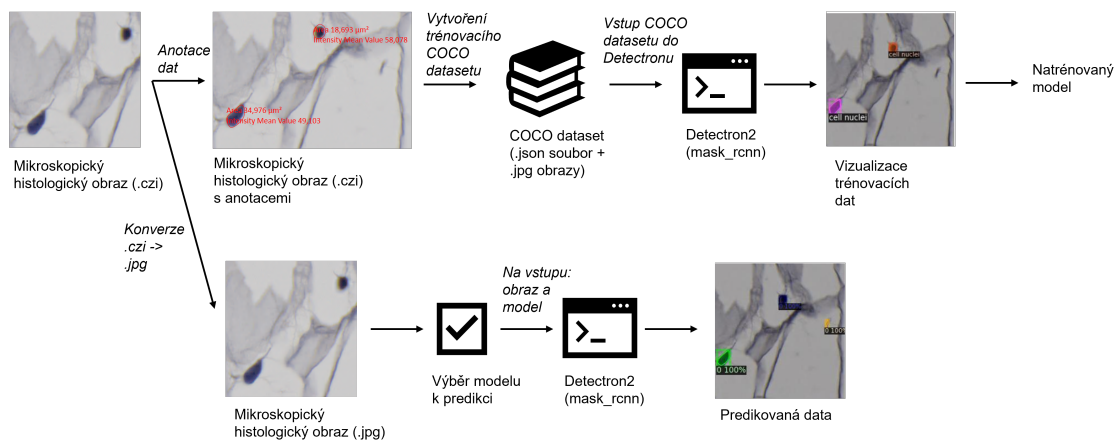
<b>Model</b>	<b>AP</b>	<b>AP50</b>	<b>AP75</b>	<b>APs</b>
mask_rcnn_R_101_C4_3x	44.597	89.705	<b>40.237</b>	44.597
mask_rcnn_R_101_DC5_3x	43.517	91.191	38.398	43.517
mask_rcnn_R_101_FPN_3x	42.875	88.472	33.890	42.875
mask_rcnn_R_50_C4_3x	43.640	89.662	36.711	43.640
mask_rcnn_R_50_DC5_3x	43.379	91.443	36.891	43.379
mask_rcnn_R_50_FPN_3x	<b>45.705</b>	<b>91.878</b>	39.603	<b>45.705</b>

Tabulka 2.5: Hodnoty Intersection over Union pro segmentaci.

Z tabulky 2.4 vyplývá, že nelze zcela jednoznačně určit jeden nejlepší model, avšak největší hodnotu AP má model mask\_rcnn\_R\_101\_C4\_3x. Ze druhé tabulky 2.5 lze naopak usoudit, že nejlepším modelem je model mask\_rcnn\_R\_50\_FPN\_3x, který dosáhl nejvyšších hodnot u AP, AP50 i APs. Pouze u AP75 byl lepší model mask\_rcnn\_R\_101\_C4\_3x.



## 2.6 Schéma vytvořeného nástroje



Obrázek 2.8: Grafické schéma zobrazující proces fungování vytvořeného nástroje.

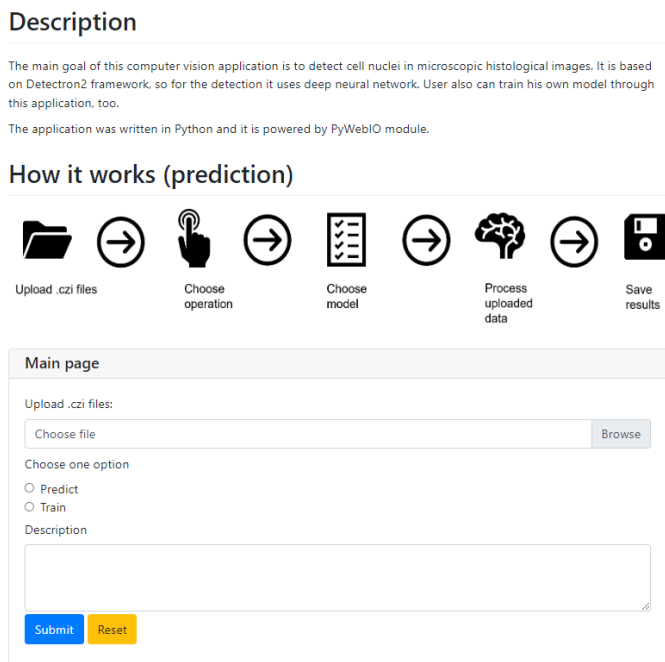
# 3. Vytvoření webové aplikace

## 3.1 Použité technologie

Aplikace, určená pro praktické využití nového nástroje pro detekci buněčných jader, byla vytvořena interaktivní webová aplikace. byla vytvořena v programovacím jazyce Python s využitím modulu PyWebIO, pomocí kterého bylo vytvořeno uživatelské rozhraní. Tato knihovna je vhodná především pro vytváření jednoduchých interaktivních aplikací. Vývojář při vytváření aplikace pomocí tohoto nástroje nemusí mít znalosti HTML ani JavaScriptu. Avšak při vývoji této aplikace byl přece jenom značkový jazyk HTML použit. Pro vytvoření aplikace byly rovněž použity části již dříve popsaných na sobě nezávislých Python skriptů (skript na vytvoření COCO datasetu a skript použitý k výpočtům na MetaCentru). Mezi nejdůležitější Python knihovny, které aplikace využívá, patří: PyWebIO, ScaffAn a Detectron2.

## 3.2 Popis aplikace

Uživatel nejprve nahraje soubory ve formátu .dzi. Následně vybere, zda chce data použít k trénování nového modelu či k predikci. Tyto dva kroky jsou zobrazeny na obr. 3.1. K trénování je možné použít pouze data, která obsahují anotované objekty. Tyto úkony uživatel provede na úvodní stránce aplikace.



Obrázek 3.1: Úvodní stránka webové aplikace, na které uživatel nahraje data a vybere operaci (predikce nebo trénování).

Pokud je zvolena operace „predikce“, je uživatel vyzván k výběru modelu z dostupných modelů, který má být k predikci použit (obr. 3.2). Následně na pozadí probíhají výpočetní operace. Jakmile jsou získány výsledky, dojde k zobrazení obrazů s predikcemi (obr. 3.3). Součástí výsledků je rovněž soubor ve formátu .json, ve kterém jsou obsaženy např. souřadnice v pixelech jednotlivých predikovaných bounding boxů nebo počet nalezených instancí konkrétní třídy v daném obrazu. Uživatel si může získané výsledky stáhnout a uložit.

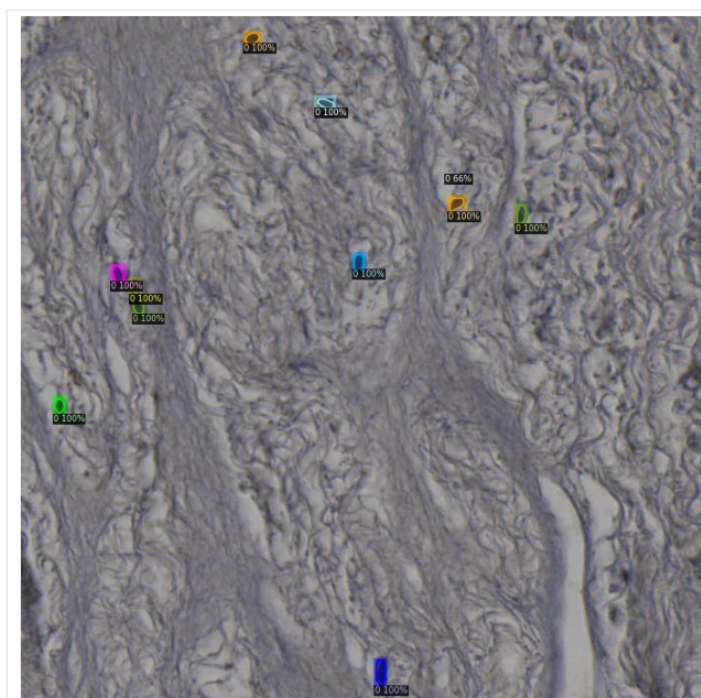
Choose one of pretrained models:

mask\_rcnn\_R\_50\_C4\_3x.pth

Submit Reset

Obrázek 3.2: Vyzvání uživatele k výběru modelu, který se má použít k predikci. Uživatel zvolí jeden model prostřednictvím rozbalovacího seznamu.

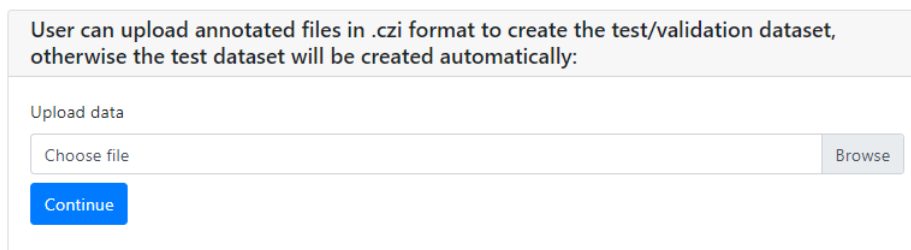
Predicted data visualization



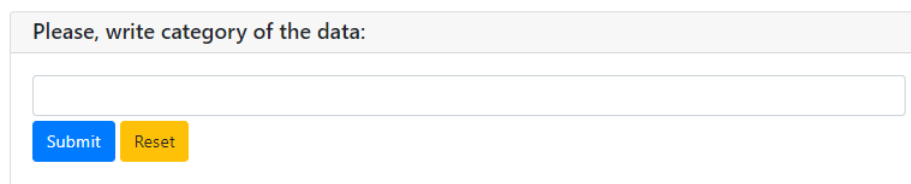
[Download results.](#)

Obrázek 3.3: Konečná stránka, na které si uživatel může prohlédnout obraz s predikcemi. Rovněž si může stáhnout výsledky, které obsahují obrazy s predikcemi. Součástí výsledků je i soubor outputs.json, který např. obsahuje počet nalezených instancí v jednotlivých obrazech nebo souřadnice v pixelech predikovaných bounding boxů.

Pokud je vybrána operace „trénování“, má uživatel v dalším kroku možnost nahrát testovací/validační data (obr. 3.4). Validační data, stejně jako trénovací, musí obsahovat anotované objekty. Pokud uživatel nenahraje vlastní testovací data, pak jsou tato data vytvořena automaticky z dat trénovacích. Poté je uživatel vyzván k napsání názvu kategorie dat (obr. 3.5), pro ilustraci byla v tomto příkladu zvolena kategorie s názvem „cell nuclei“.

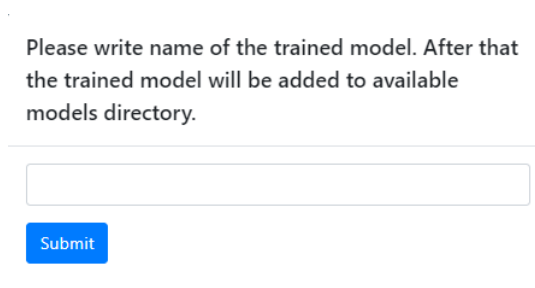


Obrázek 3.4: Možnost nahrání vlastních validačních dat.



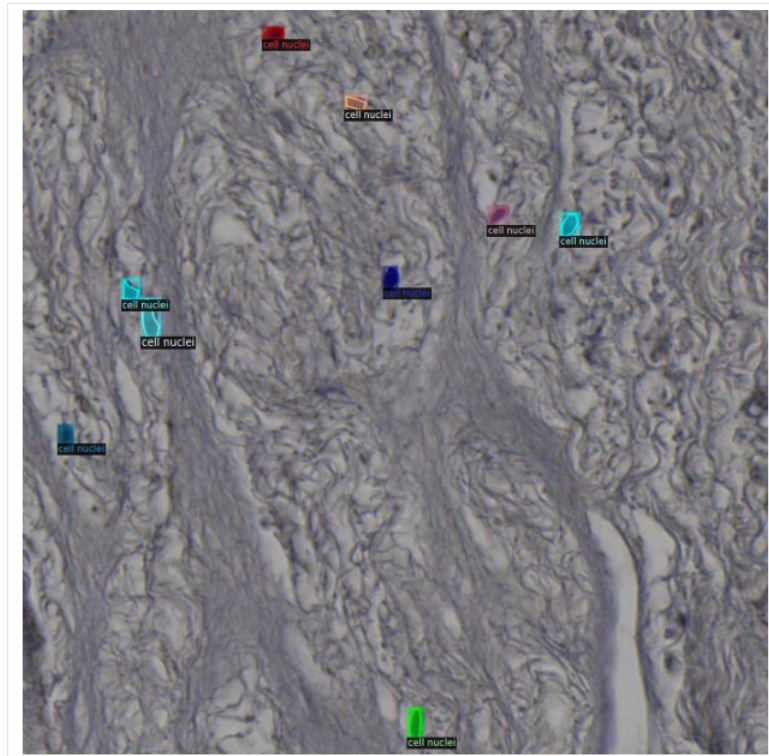
Obrázek 3.5: Určení názvu kategorie dat.

Z trénovacích a validačních dat se následně vytvoří dva COCO datasety. Aplikace k tomuto využívá již dříve popsané skripty. Poté dojde ke kontrolnímu zobrazení trénovacích dat s anotacemi (obr. 3.7), a to včetně názvu příslušné kategorie dat, který uživatel vyplnil o jeden krok dříve. Na pozadí mezitím probíhá trénování modelu. Po natrénování modelu je uživatel vyzván prostřednictvím vyskakovacího okna k pojmenování právě natrénovaného modelu (obr. 3.6). Nyní, může být model přidán k ostatní dostupným modelům určených k predikci. Na konci si může uživatel stáhnout výsledná data (použitá COCO datasety, natrénovaný model a také trénovací data se zvýrazněnými anotacemi ve formátu .jpg) (obr. 3.7).



Obrázek 3.6: Vyskakovací okno, pomocí kterého uživatel pojmenuje právě natrénovaný model.

### Annotated training data visualization



[Download trained model/s.](#)

[Download image data with annotations.](#)

[Download complete custom COCO dataset.](#)

[Download training COCO dataset.](#)

[Download validation COCO dataset.](#)

Obrázek 3.7: Konečná stránka, na které jsou vizualizovány trénovací data a také si uživatel může stáhnout výsledná data (použití COCO datasety, natrénovaný model a také trénovací data se zvýrazněnými anotacemi ve formátu .jpg).

Zdrojové kódy k aplikaci jsou dostupné na adrese: [https://github.com/janburian/Web\\_App\\_PyWebIO](https://github.com/janburian/Web_App_PyWebIO)

# Závěr

V předkládané bakalářské práci se podařilo vytvořit nástroj pro detekci buněčných jader v mikroskopických histologických obrazech. Mikroskopické histologické obrazy byly poskytnuty Biomedicínským centrem v Plzni. Tento nástroj využívá frameworku Detectron2, k detekci buněčných jader byla využita konvoluční neuronová síť Mask R-CNN.

K ověření funkčnosti vytvořeného nástroje byl proveden experiment. Nejprve byla buněčná jádra ručně anotována v grafickém programu Zeiss Zen, čímž byly vytvořeny trénovací a validační části datasetu. Celkem dataset obsahoval 166 obrazů. Trénovací část obsahovala 100 obrazů s anotacemi, validační část obsahovala 33 anotovaných obrazů a testovací část čítala taktéž 33 obrazů, ale bez anotací. Pomocí vytvořeného skriptu bylo možné z trénovací a validační části vytvořit dva COCO datasety, které sloužily jako vstupy do trénovací procedury Detectronu2. Pro tento skript jsou důležité především funkce knihovny ScaffAn, právě některé z nich musely být, do této knihovny během vypracování této práce, doprogramovány. Pomocí zmíněných dvou COCO datasetů bylo natrénováno celkem šest modelů, které sloužily k detekci buněčných jader. Hlavní metrikou, která byla použita k vyhodnocení experimentu byla metoda Intersection over Union. Jednotlivé dílčí výsledky této metody byly získány jako výstup z trénování modelu. Metoda vyhodnocovala úspěšnost predikce u bounding boxů a segmentovaných masek. U bounding boxů nebyl žádný model natolik významný, tak aby bylo možné usoudit, že se jedná o právě nejlepší model. Naopak u segmentovaných masek lze říci, že nejlepším modelem se stal model `mask_rcnn_R_50_FPN_3x`.

Pro praktické využití tohoto nástroje byla vytvořena webová aplikace, která slouží ke trénování nových modelů nebo k vizualizaci predikovaných dat (v našem případě buněčných jader). Aplikace byla napsána v programovacím jazyce Python s použitím knihovny PyWebIO a byla vyvinuta tak, aby mohla být využitelná, co nejvíce univerzálně, je tedy možné pomocí ní detekovat jakékoliv objekty, pokud budou uživatelem nahrávaná data ve formátu `.czi`.

Pokud se na závěr zaměříme na důležitost výsledných dat, tak pro lékaře z Biomedicínského centra, pro které je tento vytvořený nástroj určen především, je podstatné zejména vizuální obkreslení detekovaných buněčných jader a případně jejich počet. Naopak pro člověka, zabývající se strojovým učním, budou hodnotnější spíše souřadnice predikovaných jader, které mohou být použity jako základ pro nějaké další zpracování.

# Reference

- [1] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis, and Machine Vision*. Cengage Learning, 2014.
- [2] Miloš Železný. Učební text k předmětu USVP, May 2020.
- [3] Nameirakpam Dhanachandra, Khumanthem Manglem, and Yambem Jina Chanu. Image Segmentation Using K-means Clustering Algorithm and Subtractive Clustering Algorithm. *Procedia Computer Science*, 54:764–771, 2015.
- [4] H.P. Ng, S.H. Ong, K.W.C. Foong, P.S. Goh, and W.L. Nowinski. Medical Image Segmentation Using K-Means Clustering and Improved Watershed Algorithm. In *2006 IEEE Southwest Symposium on Image Analysis and Interpretation*, pages 61–65, 2006.
- [5] Xiang Li, Wei Li, and Ran Tao. Staged Detection-Identification Framework for Cell Nuclei in Histopathology Images. *IEEE Trans. Instrum. Meas.*, 69(1):183–193, 2020.
- [6] M.B. Hisham, Shahrul Nizam Yaakob, R.A.A Raof, A.B A. Nazren, and N.M. Wafi. Template Matching using Sum of Squared Difference and Normalized Cross Correlation. In *2015 IEEE Student Conference on Research and Development (SCOReD)*, pages 100–104, 2015.
- [7] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. Dive into Deep Learning. *CoRR*, abs/2106.11342, 2021.
- [8] Xiongwei Wu, Doyen Sahoo, and Steven C. H. Hoi. Recent Advances in Deep Learning for Object Detection. *Neurocomputing*, 396:39–64, 2020.
- [9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Commun. ACM*, 60(6):84–90, may 2017.
- [10] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.
- [11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1904–1916, 2015.
- [12] Jim Kleban, Xing Xie, and Wei-Ying Ma. Spatial pyramid mining for logo detection in natural scenes. In *2008 IEEE International Conference on Multimedia and Expo*, pages 1077–1080, 2008.

- [13] Ross Girshick. Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.
- [14] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *CoRR*, abs/1506.01497, 2015.
- [15] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN: Object Detection via Region-based Fully Convolutional Networks. *CoRR*, abs/1605.06409, 2016.
- [16] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature Pyramid Networks for Object Detection, 2017.
- [17] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.
- [18] Tianrui Liu and Tania Stathaki. Faster R-CNN for Robust Pedestrian Detection Using Semantic Segmentation Network. *Frontiers in Neurobotics*, 12, 10 2018.
- [19] Dário Augusto Borges Oliveira and Matheus Palhares Viana. Fast CNN-Based Document Layout Analysis. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 1173–1180, 2017.
- [20] S. Lawrence, C.L. Giles, Ah Chung Tsoi, and A.D. Back. Face recognition: a convolutional neural-network approach. *IEEE Transactions on Neural Networks*, 8(1):98–113, 1997.
- [21] Junqing Yang, Peng Ren, and Xiaoxiao Kong. Handwriting Text Recognition Based on Faster R-CNN. In *2019 Chinese Automation Congress (CAC)*, pages 2450–2454, 2019.
- [22] Pedram Khatamino, İsmail Cantürk, and Lale Özyılmaz. A Deep Learning-CNN Based System for Medical Diagnosis: An Application on Parkinson’s Disease Handwriting Drawings. In *2018 6th International Conference on Control Engineering Information Technology (CEIT)*, pages 1–6, 2018.
- [23] Gonçalo Marques, Deevyankar Agarwal, and Isabel de la Torre Díez. Automated medical diagnosis of COVID-19 through EfficientNet convolutional neural network. *Applied Soft Computing*, 96:106691, 2020.
- [24] Fung Fung Ting, Yen Jun Tan, and Kok Swee Sim. Convolutional neural network improvement for breast cancer classification. *Expert Systems with Applications*, 120:103–115, 2019.



- [25] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. Convolutional Neural Networks for Speech Recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(10):1533–1545, 2014.
- [26] Pankaj H. Chandankhede, Abhijit S. Titarmare, and Sarang Chauhvan. Voice Recognition Based Security System Using Convolutional Neural Network. In *2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, pages 738–743, 2021.
- [27] Jing Liang, Haotian Yang, Jiajia Gao, Caitong Yue, Shilei Ge, and Boyang Qu. MOPSO-Based CNN for Keyword Selection on Google Ads. *IEEE Access*, 7:125387–125400, 2019.
- [28] Akhil Agnihotri, Prathamesh Saraf, and Kriti Rajesh Bapnad. A Convolutional Neural Network Approach Towards Self-Driving Cars. In *2019 IEEE 16th India Council International Conference (INDICON)*, pages 1–4, 2019.
- [29] Azeddine Elhassouny and Florentin Smarandache. Trends in deep convolutional neural Networks architectures: a review. In *2019 International Conference of Computer Science and Renewable Energies (ICCSRE)*, pages 1–8, 2019.
- [30] Kunihiko Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural Networks*, 1(2):119–130, 1988.
- [31] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [32] Matthew D. Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. *CoRR*, abs/1311.2901, 2013.
- [33] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR*, abs/1409.1556, 2015.
- [34] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.
- [35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

- [36] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. *CoRR*, abs/1405.0312, 2014.
- [37] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. <http://yann.lecun.com/exdb/mnist/>, 2010.
- [38] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pages 248–255. IEEE Computer Society, 2009.
- [39] George A. Miller. Wordnet: A Lexical Database for English. *Commun. ACM*, 38(11):39–41, nov 1995.
- [40] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *Int. J. Comput. Vis.*, 88(2):303–338, 2010.
- [41] R. Sitton, B. Hanno, and D. Bernard. *Rebecca Sitton’s Spelling Sourcebook for 2nd Grade Teachers: Level 2*. Egger Publishing, Incorporated, 2001.
- [42] Ross Girshick. Facebook open sources Detectron. <https://research.facebook.com/blog/2018/01/facebook-open-sources-detectron/>, January 2019. [Online; accessed February 11, 2022].
- [43] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2: A PyTorch-based modular object detection library. <https://ai.facebook.com/blog/-detectron2-a-pytorch-based-modular-object-detection-library-/>, October 2019. [Online; accessed February 11, 2022].
- [44] Petr Ferczadi. *Učební text k předmětu histologická technika*. SZŠ Plzeň, 2019.
- [45] Vladimíra Moulisová, Miroslav Jiřík, Claudia Schindler, Lenka Červenková, Richard Pálek, Jáchym Rosendorf, Janine Arlt, Lukáš Bolek, Simona Šušová, Sandor Nietzsche, Václav Liška, and Uta Dahmen. Novel morphological multi-scale evaluation system for quality assessment of decellularized liver scaffolds. *Journal of Tissue Engineering*, 11, 5 2020.
- [46] Data format. <https://cocodataset.org/#format-data>. [Online; accessed February 11, 2022].
- [47] Inc Matterport. Mask R-CNN for Object Detection and Segmentation. [https://github.com/matterport/Mask\\_RCNN](https://github.com/matterport/Mask_RCNN), 2017. [Online; accessed May 19, 2022].

- [48] facebookresearch. Detectron2 Model Zoo and Baselines. [https://github.com/facebookresearch/detectron2/blob/main/MODEL\\_ZOO.md](https://github.com/facebookresearch/detectron2/blob/main/MODEL_ZOO.md), July 2021. [Online; accessed: May 13, 2022].
- [49] Cesnet. O Metacentru. <https://metavo.metacentrum.cz/cs/about/index.html>, December 2021. [Online; accessed February 11, 2022].
- [50] Cesnet. Cluster adan.grid.cesnet.cz - 1 952 CPU. <https://metavo.metacentrum.cz/pbsmon2/resource/adan.grid.cesnet.cz>. [Online; accessed February 11, 2022].
- [51] Cesnet. Cluster Adan. [https://wiki.metacentrum.cz/wiki/Cluster\\_Adan](https://wiki.metacentrum.cz/wiki/Cluster_Adan). [Online; accessed February 11, 2022].
- [52] Adrian Rosebrock. Intersection over Union (IoU) for object detection. <https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/>, 2016. [Online; accessed May 17, 2022].
- [53] <https://cocodataset.org/#format-data>. [Online; accessed May 17, 2022].

# Seznam obrázků

1.1	Vlevo je obraz z magnetické rezonance, vpravo obraz po aplikaci k-means algoritmu. Převzato z [4]. . . . .	15
1.2	Nejprve se navržený POI detektor naučí nalézt pozice jader, následně pak dojde, pomocí MC-Netu, k identifikaci typu buňky. Převzato z [5]. . . . .	15
1.3	Dvoudimenzionální konvoluce a výpočet prvku na pozici $o_{11}$ ve výstupní matici: $0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 2 = 19$ . Převzato z [7].	18
1.4	Graf ReLU funkce. . . . .	18
1.5	Schéma vysvětlující funkcionalitu max pooling. V tomto případě byla velikost okna pooling $2 \times 2$ . V tomto schématu je vyznačen počáteční oblast max pooling. Je vybíráno maximum z čísel 0, 1, 3 a 4. Převzato z [7]. . . . .	19
1.6	Schéma znázorňující strukturu AlexNetu, jež se skládá z 5 konvolučních vrstev (Conv), 3 max pooling vrstev (MaxPool) a 3 fully connected vrstev (FC). Převzato z [7]. . . . .	20
1.7	Porovnání jednotlivých úloh rozpoznání. (a) klasifikace objektů pouze zjišťuje, jaké objekty se v obrazu nachází, (b) detekce objektů označí objekty pomocí bounding boxu, (c) sémantická segmentace predikuje kategorie jednotlivých pixelů, bez toho aniž by došlo k rozlišení jednotlivých instancí objektů, (d) segmentace instancí detekuje jednotlivé rozlišitelné objekty. Převzato z [8]. . . . .	24
1.8	Ukázky jednotlivých barvení. . . . .	28
2.1	Proces anotace jader buněk v programu ZEISS ZEN. . . . .	30
2.2	Zobrazení konkrétní anotace pomocí nástroje ScaffAn a knihovny matplotlib. . . . .	31
2.3	Schéma popisující architekturu konvoluční neuronové sítě Mask R-CNN. . . . .	36
2.4	Vizualizace využití anchor boxů. Převzato z [47]. . . . .	37
2.5	Vizualizace trénovací části datasetu. . . . .	39
2.6	Vizualizace predikce buněčných jader. . . . .	40
2.7	Grafická interpretace metody Intersection over Union. Převzato z [52]. . . . .	42
2.8	Grafické schéma zobrazující proces fungování vytvořeného nástroje. . . . .	44
3.1	Úvodní stránka webové aplikace, na které uživatel nahraje data a vybere operaci (predikce nebo trénování). . . . .	45
3.2	Vyzvání uživatele k výběru modelu, který se má použít k predikci. Uživatel zvolí jeden model prostřednictvím rozbalovacího seznamu. . . . .	46

3.3	Konečná na stránka, na které si uživatel může prohlédnout obraz s predikcemi. Rovněž si může stáhnout výsledky, které obsahují obrazy s predikcemi. Součástí výsledků je i soubor outputs.json, který např. obsahuje počet nalezených instancí v jednotlivých obrazech nebo souřadnice v pixelech predikovaných bounding boxů. . . . .	46
3.4	Možnost nahrání vlastních validačních dat. . . . .	47
3.5	Určení názvu kategorie dat. . . . .	47
3.6	Vyskakovací okno, pomocí kterého uživatel pojmenuje právě natrénovaný model. . . . .	47
3.7	Konečná stránka, na které jsou vizualizovány trénovací data a také si uživatel může stáhnout výsledná data (použité COCO datasety, natrénovaný model a také trénovací data se zvýrazněnými anotacemi ve formátu .jpg). . . . .	48

# Seznam tabulek

2.1 Použité předtrénované COCO Instance Segmentation modely, které využívají Mask R-CNN. Převzato z [48]. . . . .	38
2.2 Hardwarová specifikace uzlu spadajícího pod adan.grid.cesnet.cz. . . . .	38
2.3 Doby trénování získaných modelů. . . . .	42
2.4 Hodnoty Intersection over Union pro bounding boxy. . . . .	43
2.5 Hodnoty Intersection over Union pro segmentaci. . . . .	43