
**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**
**BRNO UNIVERSITY OF TECHNOLOGY**

**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**
**FACULTY OF INFORMATION TECHNOLOGY**

**ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ**
**DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA**


# HLUBOKÉ NEURONOVÉ SÍTĚ PRO ANALÝZU MEDICÍNSKÝCH DAT

**DEEP LEARNING FOR MEDICAL IMAGE ANALYSIS**

**BAKALÁŘSKÁ PRÁCE**
**BACHELOR'S THESIS**


**AUTOR PRÁCE** **MARTIN OSVALD**
**AUTHOR**

**VEDOUCÍ PRÁCE** **Ing. MICHAL ŠPANĚL, Ph.D.**
**SUPERVISOR**



**BRNO 2021**

**Vysoké učení technické v Brně**
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2020/2021

# Zadání bakalářské práce


Student: **Osvald Martin**
Program: Informační technologie
Název: **Hluboké neuronové sítě pro analýzu medicínských dat**
**Deep Learning for Medical Image Analysis**
Kategorie: Zpracování obrazu

Zadání:

1. Seznamte se s problematikou hlubokých neuronových sítí a jejich učení.
2. Zorientujte se v metodách analýzy medicínských dat s využitím hlubokých neuronových sítí (obrazová CT data, RTG snímky, 3D modely tkání, apod.).
3. Vyberte vhodnou metodu použitelnou pro řešení zvoleného problému analýzy medicínských dat.
4. Implementujte navrženou metodu s využitím existujících nástrojů pro trénování hlubokých neuronových sítí.
5. Proveďte experimenty nad připravenou datovou sadou.
6. Porovnejte dosažené výsledky a diskutujte možnosti budoucího vývoje.
7. Vytvořte stručný plakát nebo video prezentující vaši práci, její cíle a výsledky.

Literatura:

- U-Net: Convolutional Networks for Biomedical Image Segmentation Medical Image Computing and Computer-Assisted Intervention, https://arxiv.org/pdf/1505.04597

Pro udělení zápočtu za první semestr je požadováno:

- Splnění prvních tří bodů zadaní.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Španěl Michal, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 30. července 2021
Datum schválení: 30. října 2020

# Abstrakt

Cieľom tejto práce je použitie 2D konvolučných neurónových sietí pre segmentáciu a detekciu zubov na 3D modeli čeľusti s využitím viac pohľadovej metódy. Pohľad je vyrendrovaný 2D obrázok 3D modelu. Následne na akýkoľvek 3D model zubov je možné použiť natrénované modely neurónových sietí v PyQt aplikáciach. Pri práci bol vytvorený vlastný anotačný skript na anotáciu zubov ako aj landmarkov. Táto práca rieši problém s dostupnosťou anotovaných 3D datasetov v medicínskom priemysle pomocou automatizácie v generovaní masiek z rôznych pohľadov na 3D modely.



# Abstract

The goal of this bachelor's thesis is to use the 2D convolutional neural network on the 3D model dataset by multi-view methods. The view is 2D picture of 3D model. The result are Pyqt applications, where is possible to load the 3D model of teeth and predict the location of landmarks and teeth by object segmentation and object detection. During this thesis, an annotation's script was created for the annotation of 3D models for landmarks of teeth and teeth themself. This thesis solves the problem of the small availability of annotated 3D datasets in the medical industry by automating generating binary masks from different views on 3D models.



# Kľúčové slová

Medicínske dáta, Segmentácia zubov, Detekcia Zubov Landmark, Ortodoncia , Viac-pohľadová metóda, 2D konvolučné siete, 3D objektová klasifikáciu,U-Net, Yolo v3, Mask-RCNN, Blender, STL model, Anotačný skript.



# Keywords

Medical data, Teeth segmentation, Teeth detection, Landmark, Orthodontics , Multi-view method, 2D convolutional neural network, 3D object classification, U-Net, Yolo v3, Mask-RCNN, Blender, STL model, Anotation script.



# Citácia

OSVALD, Martin. *Hluboké neuronové sítě pro analýzu medicínských dat*. Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Michal Španěl, Ph.D.

# Hluboké neuronové sítě pro analýzu medicínských dat

## Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Michala Španěla Ph.D.

Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Martin Osvald
2. augusta 2021


## Poďakovanie

Ďakujem pánovi Ing. Michalovi Španělovi Ph.D. za poskytnutie odborných rád v rámci konvolučných neurónových sietí, odporúčanie odbornej literatúry a aj prednášok na Magisterskom štúdiu.

# Obsah

# Kapitola 1

# Úvod

Počítačové videnie sa už niekoľko rokov využíva v medicínskom priemysle, avšak potreba analýzy medicínskych dát neustále narastá. V posledných rokoch pozorujeme postupný progres kvôli disciplíne Hlboké učenie ( Deep Learning ), vďaka ktorému by sme sa už neobišli v medicínskom priemysle. Základy Hlbokých neurónových sietí vznikli okolo 50-tych rokoch 20. storočia, ale aktívne sa začali používať od roku 2012. Hlboké učenie pomáha doktorom a špecialistom v medicínskom priemysle pripraviť sa na operáciu, odhaliť rôzne zápaly, zlomeniny alebo dokonca nádory v X-Ray snímkoch alebo CT snímkoch, čo si ľudské oko trénovaného špecialistu alebo doktora nemusí všimnúť. Achillovou pätou neurónových sietí je dostupnosť a veľkosť dát na trénovanie.

Cieľom tejto práce je navrhnúť, implementovať a otestovať 2D konvolučné neurónové siete na 3D modelov zubov a vytvoriť aplikáciu, ktorá využije modely neurónových sietí na vybrané zuby. Dôležité kritéria tejto práce sú presnosť neurónových sietí a vytvorenie veľkého datasetu na trénovanie s učiteľom.

Motivácia tejto práce je zautomatizovať a uľahčiť prácu zubárovi pri zákrokoch ako aj pred prípravou na zubný strojček. Zároveň ide o experimentálnu činnosť aplikovania konvolučných neurónových sietí na 3D modely pomocou rôznych pohľadov na model.

V kapitole č. 2 opisujem základné poznatky v čeľustnej ortopédií. V kapitole č. 3 vysvetľujem základné znalosti hlbokého učenia potrebné na pochopenie tejto práce. V kapitole č. 4 riešim zadaný cieľ práce ako aj návrh riešenia. V kapitole č. 5 opisujem veľmi špecificky implementáciu bakalárskej práce. Riešim nastavenia parametrov pri učení ako aj úspešnosť klasifikácie na testovacom datasete. Taktiež riešim dôvody a nutnosť implementácie aplikácie na testovanie modelov sietí. Nakoniec v kapitole č. 6 riešim jednotlivé testy a výsledky testovania.

# Kapitola 2

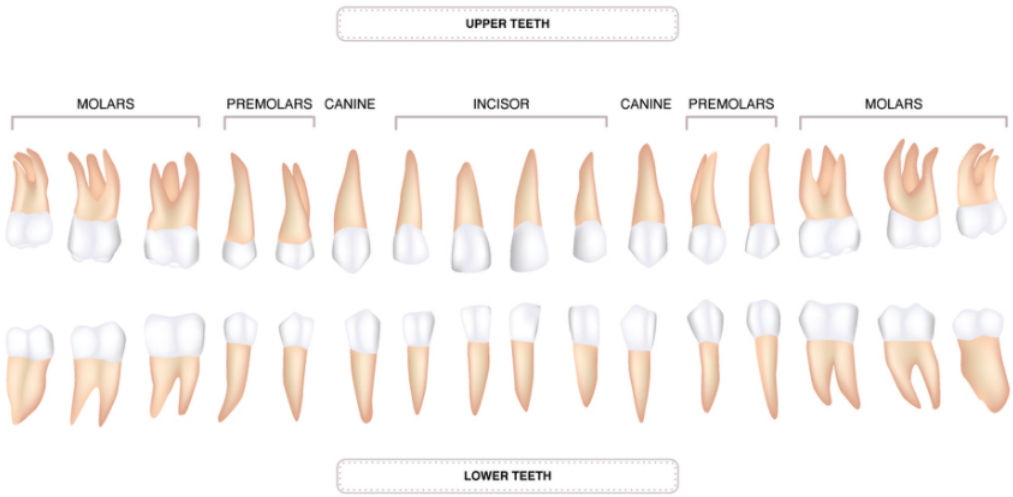
# Čelustná ortopédia

## 2.1 Zuby a anatómia zubov

Čeľustná ortopédia alebo ortodoncia je veda o postavení a funkcii zubov a čeľustí. Je jedným z nadstavbových odborov stomatológie a jej úlohou je riešiť defekty v ústnej dutine. Liečbou dokážeme ovplyvniť rast a vývoj zubov, čeľuste, a tak predísť funkčným aj estetickým komplikáciám.

Problém „krivých zubov“ môže vzniknúť na základe rôznych faktorov. Najčastejšie je príčina v zuboch samotných, ich veľkosti a postavení čeľustí alebo príčinou sú tkanivá okolo zubov, pier a jazyka. Krivé zuby môže človek získať aj zlozvykmi,

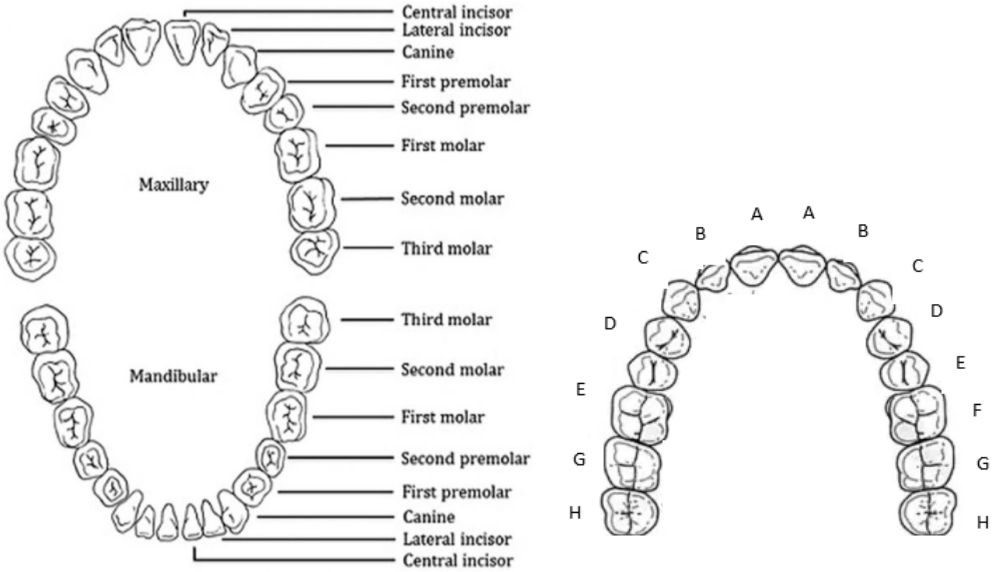
Zuby dospelého človeka (pozn. nie *mliečne* zuby) majú štyri základne typy. Rezáky ( *inscissors* ), očné zuby ( *canine* ), premolár ( *premola*r ) a molár ( *molar* ). Tieto základné typy nám tvoria horný a dolný chrup. Každý zub môžeme očíslovať alebo označiť písmenom,




Obr. 2.1: Typy zubov, obrázok je prebratý z [23].

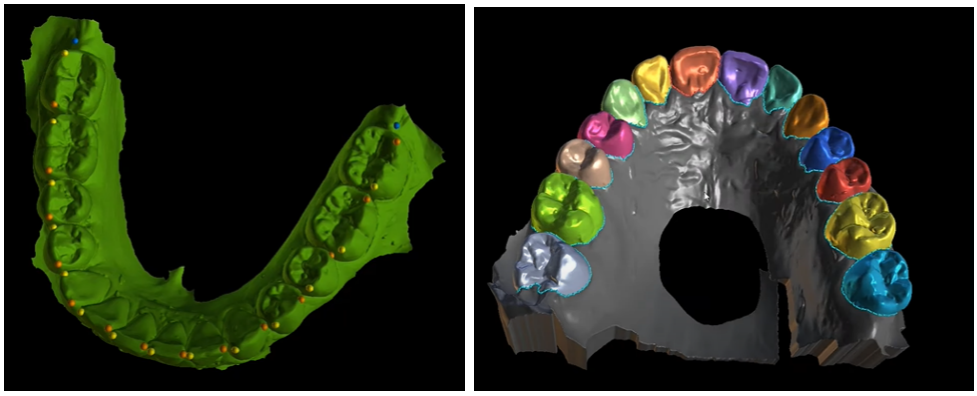
teda pridať každému zubu identifikátor. Týmto značením budeme postupovať aj pri tvorbe datasetu a pri učení neurónových sieť v kapitole 4 a 5.2 .

Neurónové siete v tejto práci nebudú rozlišovať či sa jedná o horný alebo dolný chrup, lebo z pohľadu veľkosti obrázkov pre neurónovú sieť ide o zrkadlový obraz. Každý dospelý jedinec na jednom chrupe má štyri *rezáky*, dva *očné* zuby, štyri *premoláry* a šesť *molárov*,




Obr. 2.2: Zuby rozdelené podľa typov prebraté z [24].

z toho dva sú zuby múdrosti, ktoré sa väčšinou vyberajú kvôli bolestiam chrupu a hlavy. Každý zub však má svoje špecifické body ( landmarky ). Existujú rôzne typy špecifických bodov. V tejto práci sa však budem riadiť bodmi vytvorené pre program BlueSkyBio[1]. Program v BlueSkyBio rieši segmentáciu jednotlivých zubov pomocou zadaných špecifických bodov.



Obr. 2.3: Špecifické body na zuboch v Blueskybio.

Cieľom tejto práce je použitie neurónových sietí na vyhľadávanie landmarkov a umožniť automatizáciu takéhoto zadávania, ako aj preskočenie takéhoto bodu a umožniť segmentáciu zubov priamo z pohľadu na 3D model.

[1]https://blueskybio.com/

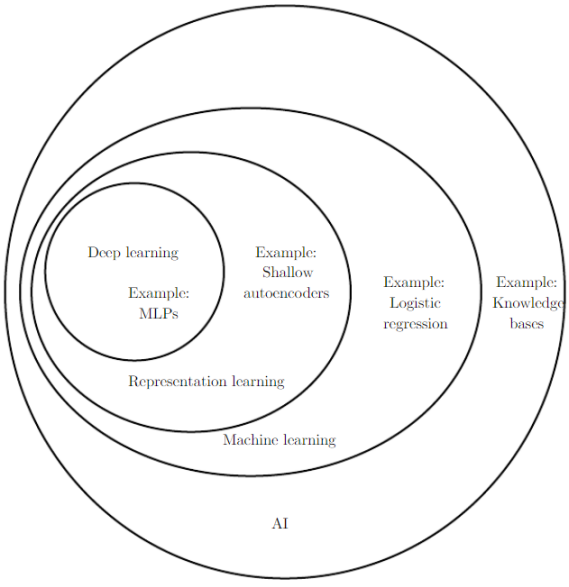
# Kapitola 3

# Hlboké učenie

## 3.1 Umelá inteligencia, Strojové učenie, Hlboké učenie

Pojem umelá inteligencia začal vznikať okolo roku 1956, keď John McCarthy ho zadefinoval na svojom seminári ako výpočtový systém, ktorý by mal byť schopný skladať hudbu formulovať alebo dokazovať vety [6].

Umelá inteligencia (AI) je obecná oblasť zahrňujúca nielen neurónové siete, ale aj rôzne iné metódy a prístupy. Napríklad prvé šachové programy by sa mohli nazvať ako prvé programy umelej inteligencie, ale to bol len súbor jednoduchých pravidiel. Existujú aj logické problémy, ktorým sa nedajú určiť jednoznačné pravidlá. Preto vznikla podoblasť v umelej inteligencii, ktorú nazývame *strojové učenie*.




Obr. 3.1: Obrázok znázorňujúci vzťahy medzi umelou inteligenciou, strojovým učením a hlbokým učením z [9].

Strojové učenie je oblasť, v ktorej namiesto programovania algoritmov a pravidiel, sa algoritmy a pravidlá *trénujú*. Výsledkom týchto algoritmov sú modely. Modely na daných trénovaných vzoroch (vstupov a prípadných výstupov) učíme rozpoznávať pravidlá a vzory vstupov na požadované výstupy. Algoritmy strojového učenia delíme na:

1. Učenie s učiteľom
2. Učenie bez učiteľa
3. Semi-supervizované učenie
4. Posilované učenie

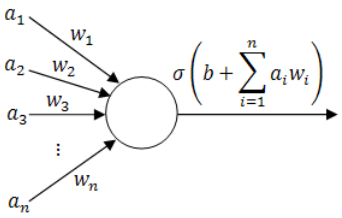
V tejto práci bude použitý spôsob učenia s učiteľom, keďže pre konvolučné neurónové siete použité v tejto práci potrebujem vstupy a výstupy.

Hlboké učenie je pod oblasťou strojového učenia, kde sa aplikujú najmä algoritmy učenia s učiteľom a algoritmy inšpirované neurónmi v ľudskom mozgu. Tieto algoritmy sa nazývajú hlboké neurónové siete. Výskum vo vedeckej komunite hlbokého učenia sa neustále vyvíja. Cieľom hlbokého učenie je pochopiť viac-dimenzionálne dáta s bohatou štruktúrou [9].

## 3.2 Neurónové siete

Neurónové siete ( Artificial Neural Network, ANN ) sa v poslednej dobe stávajú veľmi populárnym riešením na problémy v Strojovom učení. Za tento trend však môže aj dostupnosť knižníc a zvýšenie výpočtovej sily pre neurónové siete. Neurónové siete predstavujú skupinu algoritmov inšpirovaných štruktúrou neurónmi v mozgu.

Neurónové siete sa skladajú z jednotiek volané Neurón. Najjednoduchšia neurónová sieť sa volá Perceptron [14], ktorá sa skladá z jedného neurónu a z jednej výpočtovej vrstvy. Každý vstup $a$ je násobený svojou váhou $w$. Ku skalárnemu súčinu váh a vstupov sa pripočíta bias $b$.




Obr. 3.2: Znázornenie Perceptronu prevzatý z [8].

## 3.3 Konvolučné neurónové siete

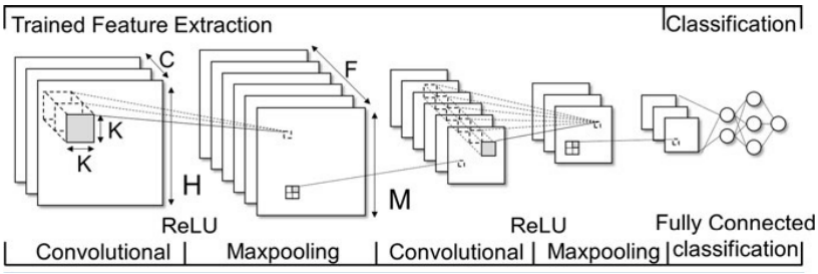
Neurónové siete nie sú ideálne na trénovanie obrázkov a 3D modelov, kvôli ťažkosti a neefektivite trénovania. Preto vznikli špeciálne siete, ktoré sa volajú konvolučné neurónové siete (CNN) [15]. Vďaka redukcii parametrov a prítomnej ekvivariancii sú konvolučné neurónové siete schopné predikovať tvary a hrany.

Konvolučné neurónové siete pracujú so špeciálnymi maticami nazývanými tensory. Tak ako matice, tak aj tensory môžu byť jedno dimenzionálne ako vektor, dvoj-dimenzionálne ako čiernobiely obrázok, troj-dimenzionálne ako obrázok s kanálmi RGB, štvor-dimenzionálne ako video pričom štvrtá os predstavuje čas.

Konvolučné neurónové sieťe sa všeobecne skladajú sa z :

- Vstupnej vrstvy (input layer)

- Skrytej vrstvy (hidden layer)
- Výstupnej vrstvy (output layer)




Obr. 3.3: Obrázok znázorňujúci konvolučnú neurónovú sieť s operáciami konvolúcia a max-pooling, prebratý z [14].

## 3.4 Operácie v konvolučných neurónových sieťach

### Konvolučná vrstva

Konvolučná vrstva (CONV) využíva filter, ktorý používa konvolučnú operáciu na vstup $I$. Hyperparametre tohto filtra majú veľkosť $F$ a stride $S$. Výsledok operácie sa volá feature map alebo activation map.

Konvolúcia je matematická operácia definovaná ako:

$$s(t) = (x * w)w(t)$$

kde $*$ označuje konvolúciu , $x$ ako vstup a $w$ ako jadro, taktiež nazývaný kernel a $s(t)$ je feature map.

Pre 2D matice má konvolúcia iný vzorec a to:

$$S(i,j) = (K * J)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n)$$

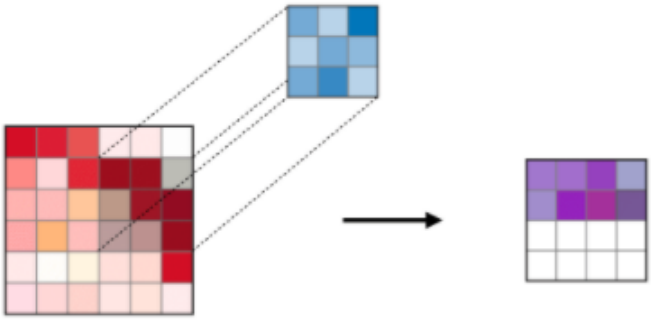
Konvolúcia a podobné operácie nájdu uplatnenie v inžinierstve, aj v matematike. Najviac sa využívajú v digitálnom spracovaní signálov, v spracovaní obrázkoch, v elektrických v obvodoch ((LTI), ale aj v spracovaní obrázkoch, čo prezentujem v tejto bakalárskej práci.

### Max-pooling

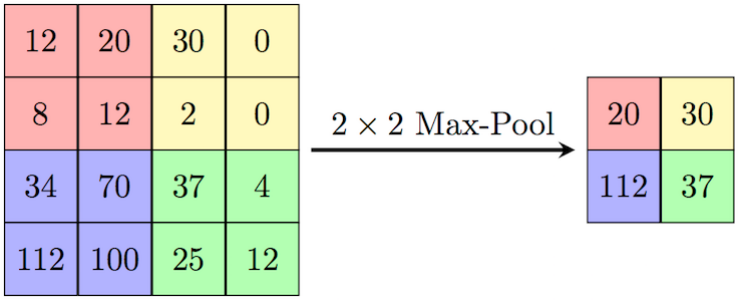
Operácia Max-pool redukuje maticu na novú maticu na základe najväčších prvkov rozdelených podľa pool-size. Operácia je znázornená na Obrázku 3.5.

### Padding

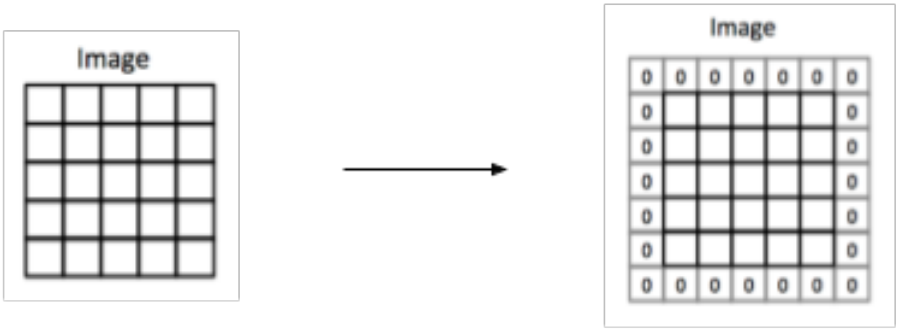
Kedže po operáciach Max-pooling a konvolúcie , matica stráca veľkosť, padding je operácia, ktorá slúži na vyplnenie okrajov danej matice. Operácia je znázornená na Obrázku 3.6.



Obr. 3.4: Operácia konvolúcie pomocou filtra pri CNN prebraté z [2].




Obr. 3.5: Znázornenie operácie Max-pooling, prebraté z [1].



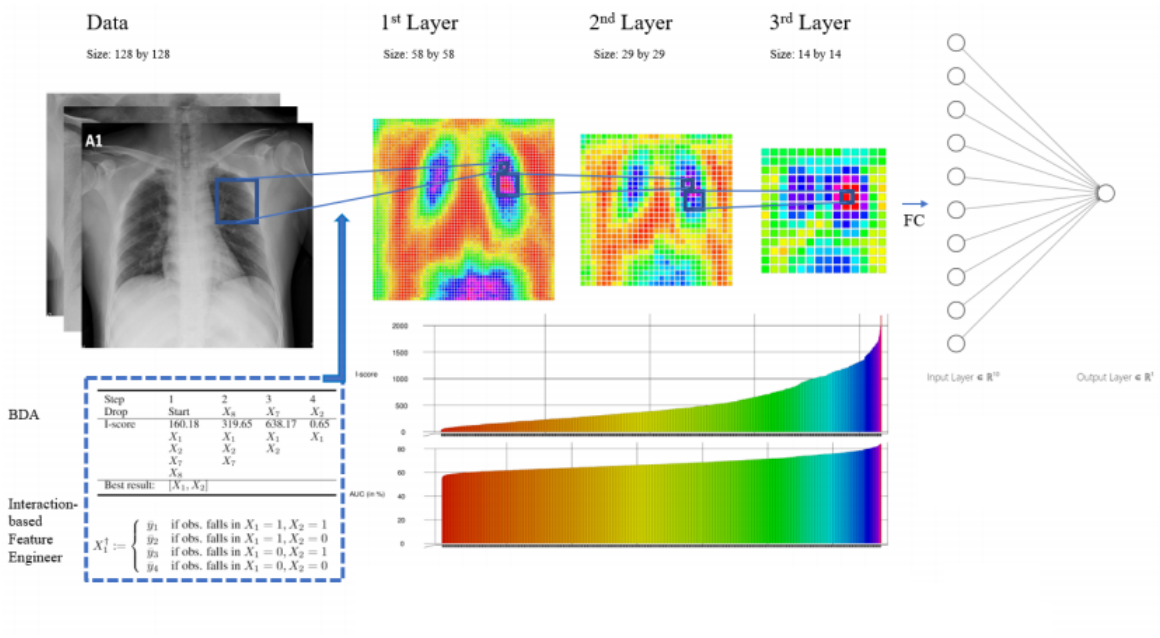

Obr. 3.6: Obrázok po aplikovaní operácie padding, upravený z [4].

Operácie s tensormi sú podporované grafickými kartami a knižnicami Pytorch[1] (vyvíjaný facebook-om[2] a Tensorflow[3] ( vyvíjaný google-om [4]). Tensor je dokonca tak významná jednotka, že je po nej pomenovaná knižnica.

## 3.5 Hlboké učenie a medicína

V posledných rokoch medicína začala veľmi napredovať a stáva sa dátovo závislá na hlbokých neurónových sieťach [5]. Neurónové siete sa bežne aplikujú na detekciu nádorov v mozgu [7], zápaloch v pľúcach[10] alebo na detekciu zlomenín [22]. Počas pandémie *Covid-19* sa začali aplikovať aj na detekciu covidu na röntgenových obrázkoch pľúc, pre rýchlosť a cenovú efektivitu oproti PCR testom [13].

Jedným veľkým problémom v medicíne je limitované množstvo anotovaných dát na trénovanie konvolučných neurónových sietí, kvôli legálnosti a súkromiu daných osôb. Preto vzniká potreba veľkého množstva dát. V tejto bakalárskej práci riešim aj tento problém v kapitole 5.2.



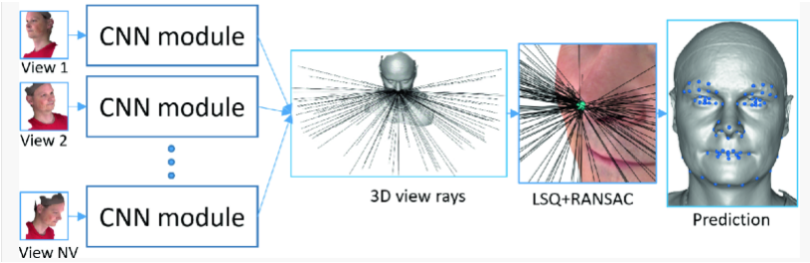

Obr. 3.7: Ilustračný obrázok na vysvetlenie CNN pri Covide 19 z [13].

## 3.6 Multi-view metódy na trénovanie

Pri rapídnom zvýšení počtu 3D modelov vďaka presným meracím prístrojom, nastáva dopyt pre analýzu v 3D doméne. Problémom však je výpočtová technika na analýzu takýchto dát. Preto sa začali používať Multi-view metódy [16]. Princíp spočíva v generovaní rôznych 2D

[1] https://pytorch.org/
[2] https://facebook.com/
[3] https://www.tensorflow.org/
[4] https://google.com/

pohľadov na 3D model, namiesto počítania a hľadania vzťahov medzi bodmi v 3D priestore. Táto metóda sa dá aplikovať na polohu landmarkov a aj celkových masiek.



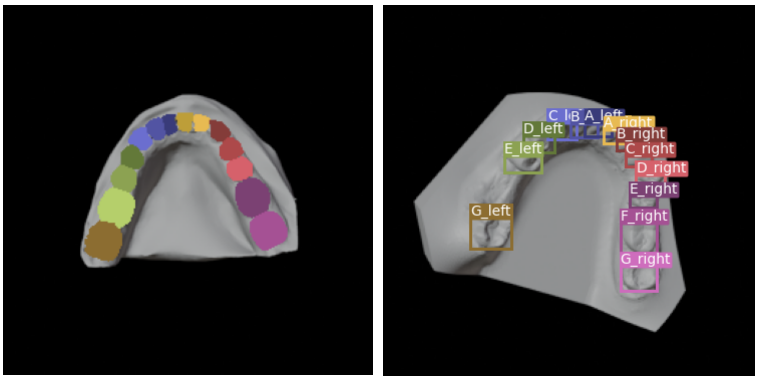

Obr. 3.8: Znázornenie použitia rôznych pohľadov, prebraté z [16].

## 3.7 Image segmentation

V digitálnom spracovaní obrázkov a v procesingu a počítačovom videní, obrázková segmentácia je proces rozdelenia obrázku na viacero segmentov (skupinu pixlov). Cieľom segmentácie je zjednodušenie reprezentácie obrázka na masku, čo je viac zmysluplné a jednoduchšie na analýzu pre počítač [21]. Obrázková segmentácia je typicky používaná na lokáciu a zistenie hrán v obrázku. Špecificky obrázková segmentácia je proces pridania názvu (label) každému pixlu. Táto metóda nám dodáva granulárne porozumenie objektov v obrázku.

## 3.8 Object detection

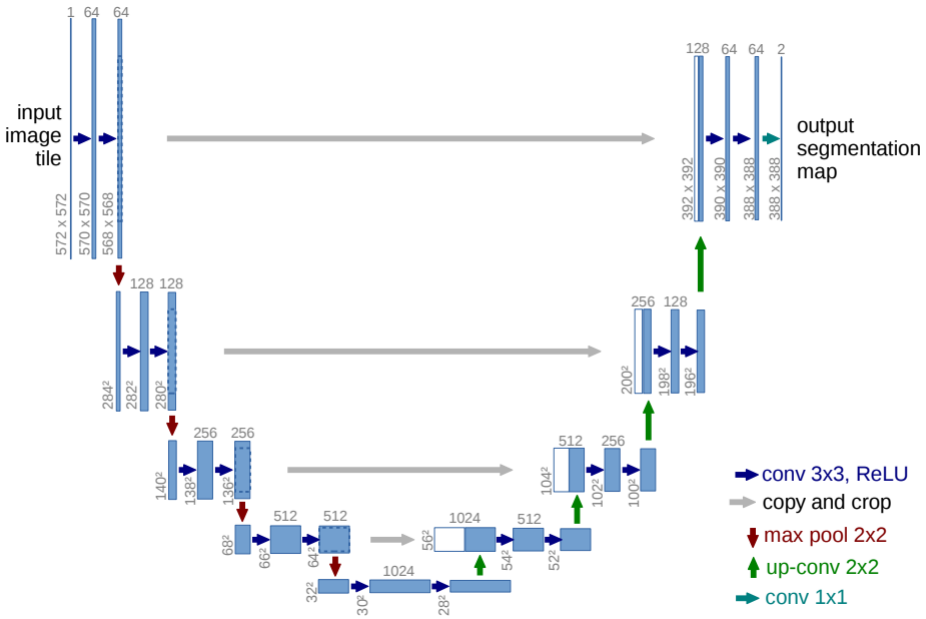
Object detection je technika, ktorá identifikuje a lokalizuje objekty v obrázku alebo vo videu. Tak ako v obrázkovej segmentácii, aj Object detection môže byť použitý na počítanie objektov v obrázku a sledovať ich polohu. Na rozdiel od obrázkovej segmentácií, Object detection vráti polohu daných objektov a nie masky ku objektom.




Obr. 3.9: Vľavo je znázornená segmentácia zubov a vpravo detekcia zubov.

## 3.9 U-Net

U-net [20] je plne konvolučná neurónová sieť špecificky spravená pre bio – medicínske dáta. Je pomenovaná podľa tvaru architektúry, ktorá pripomína písmeno U. Delí sa na ľavú (*contracting part*) a pravú časť (*expansive part*).




Obr. 3.10: Originálna architektúra prebratá z [20].

Jednotlivé šípky znázorňujú jednotlivé operácie vysvetlené v 3.4. Dôležitou časťou architektúry je, že každou operáciou do nižšej vrstvy zdvojujeme počet kanálov (zo 64 kanálov na 128 kanálov, zo 128 kanálov na 256 kanálov ...). To modelu dovoľuje v pravej časti architektúry mať veľké množstvo príznakových kanálov na propagovanie kontextovej informácie do vyšších vrstiev [20].

Je veľmi dôležité zvoliť veľkosť vstupného obrázka ku správnej segmentácii, aby bolo možné vykonať všetky 2x2 max-pool operácie. Soft-max operácia je definovaná ako:

$$p_k(\mathbf{x}) = \frac{\exp(a_k(\mathbf{x}))}{\sum_{k'=1}^{K} \exp(a_{k'}(\mathbf{x})}$$

kde $a_k(\mathbf{x})$ označuje aktiváciu v kanáli príznakov $k$, na pozícii pixla $\mathbf{x}$ pričom $\mathbf{x} \in \Omega$ a $\Omega \subset \mathbb{Z}^2$. $K$ znázorňuje počet tried.

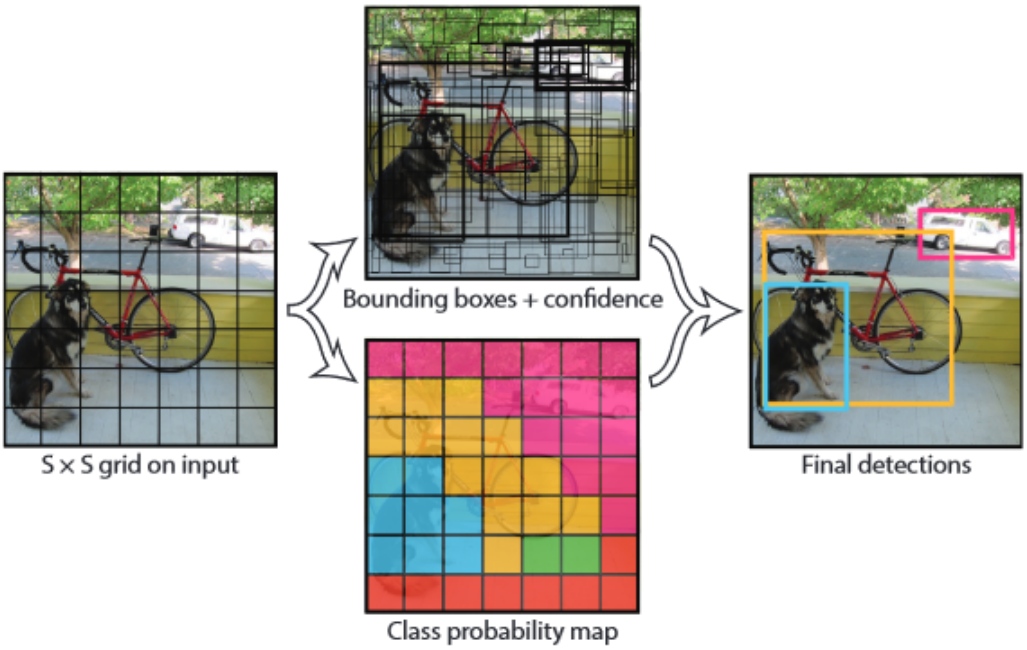
Tento model siete vyhral súťaž ISBI na sledovanie buniek v 2D malých datasetoch v roku 2015. Multi-view metóda sa dokonca používa s U-net a je dokonca o trošku lepšia ako segmentácia 3D modelov [3]. Autori siete trvdia, že sieť je veľmi vhodná na bio–medicínske dáta a preto táto sieť bola použitá v tejto práci.

## 3.10 You only look once (YOLO)

### Predchodca YOLO v1.

You only look once (YOLO) je sieť trénovaná na objektovú detekciu, nie na objektovú segmentáciu. Na objektovú detekciu sú typicky používane RCNN siete, ktoré najprv vygenerujú potenciálne bounding boxy v obrázku a až potom je použitý klasifikátor na tieto boxy. Po klasifikácii sieť upravuje predikciu oblastí na tieto boxy, eliminuje viacnásobne predikcie na ten istý objekt a dáva každému boxu nové skóre [17].

YOLO v1. je veľmi jednoduchá sieť , skladá sa iba z jednej konvolučnej siete a predikuje súčasne bounding boxy a triedy [17]. YOLO v1. pri detekcii rozdelí obrázok na mriežku $S \times S$. Ak centroid hľadaného objektu spadá do bunky, daná bunka je zodpovedná za detekciu daného objektu. Každá bunka predikuje $B$ boxov [17].




Obr. 3.11: Obrázok znázorňujúci model z [17].

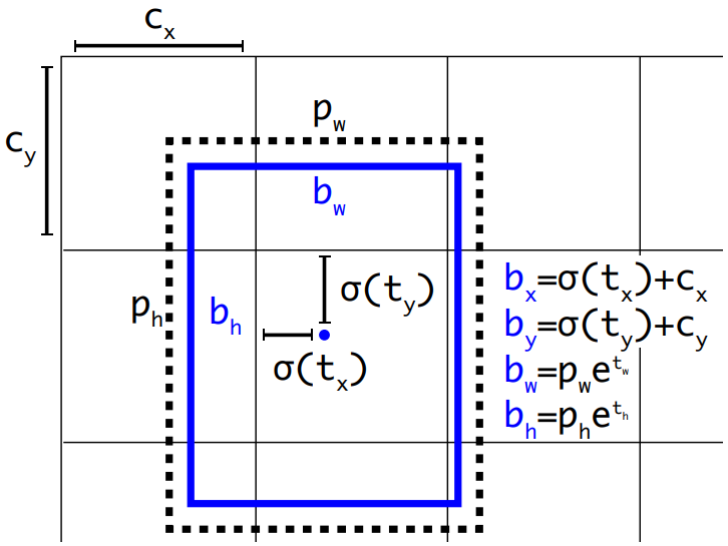
Veľká výhoda tejto siete je v tom, že trénovanie prebieha rýchlejšie ako pri RCNN sieťach, ale nevýhoda je zase v tom, že má problém detekovať malé objekty [17].

### YOLO v3.

YOLO v3. je tretia verzia pre presnejšiu a rýchlejšiu predikciu ako YOLO v1.[18]. Základná architektúra sa zmenila a skladá z Darknet-53. Darknet-53 je o dosť väčší ako Darknet-19, pretože Darknet-53 sa skladá z 53 konvolučných vrstiev o proti 19 vrstvám. Je aj viac efektívna ako Resnet-101 a Resnet-152 [18]. Architektúra ovplyvnila aj úspešnosť predikcie malých boxov. Pri každom vstupnom obrázku sa robí predikcia boundig boxu na troch rôznych veľkostiach, čo umožňuje predikovať malé , stredné a aj veľké bounding boxy. Taktiež sa používa metoda Anchor boxov oproti YOLO v1. Anchor box slúži ako kotva na predikciu hrán v bunke. Vzorec bounding-boxu sa tiež tým pádom zmenil a to:

$$b_x = \sigma(t_x) + c_x$$
$$b_y = \sigma(t_y) + c_y$$
$$b_w = p_w e^{t_w}$$
$$b_h = p_h e^{t_h}$$

kde $t_x, t_y, t_w, t_h$ je predikcia koordinátu bounding boxu kde $x$ a $y$ znázorňuje centroid objektu, $w$ predstavuje dĺžku šírky centroidu od hrany bounding boxu, $h$ výšku kde $\sigma(t_x)$ a $\sigma(t_y)$ majú hodnotu medzi 0.




Obr. 3.12: Znázornenie vzorcov na predikcii boxu, prebraté z [18].

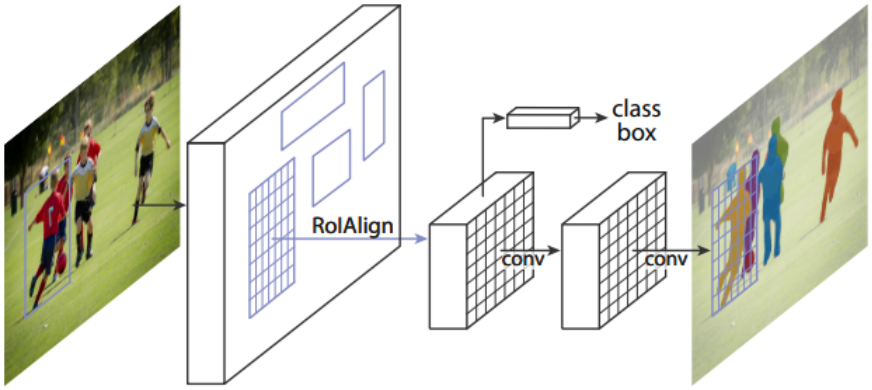
| | Type | Filters | Size | Output |
|---|---|---|---|---|
| | Convolutional | 32 | 3 × 3 | 256 × 256 |
| | Convolutional | 64 | 3 × 3 / 2 | 128 × 128 |
| 1× | Convolutional | 32 | 1 × 1 | |
| | Convolutional | 64 | 3 × 3 | |
| | Residual | | | 128 × 128 |
| | Convolutional | 128 | 3 × 3 / 2 | 64 × 64 |
| 2× | Convolutional | 64 | 1 × 1 | |
| | Convolutional | 128 | 3 × 3 | |
| | Residual | | | 64 × 64 |
| | Convolutional | 256 | 3 × 3 / 2 | 32 × 32 |
| 8× | Convolutional | 128 | 1 × 1 | |
| | Convolutional | 256 | 3 × 3 | |
| | Residual | | | 32 × 32 |
| | Convolutional | 512 | 3 × 3 / 2 | 16 × 16 |
| 8× | Convolutional | 256 | 1 × 1 | |
| | Convolutional | 512 | 3 × 3 | |
| | Residual | | | 16 × 16 |
| | Convolutional | 1024 | 3 × 3 / 2 | 8 × 8 |
| 4× | Convolutional | 512 | 1 × 1 | |
| | Convolutional | 1024 | 3 × 3 | |
| | Residual | | | 8 × 8 |
| | Avgpool | | Global | |
| | Connected | | 1000 | |
| | Softmax | | | |

Obr. 3.13: Architektúra Darknet-53, tabuľka prebratá z [18].

## 3.11 Mask-RCNN

Mask-RCNN je výsledok Facebook výskumu. Mask-RCNN je typ siete založený na Regionálnych konvolučných sieťach (Regional Convolutional Neural Network), špecificky Faster-RCNN [19]. Mask-RCNN ( novšia verzia Detectron 2 [5]) sa používa denne na sociálnych sieťach ako je Facebook a Instagram pre detekciu a segmentáciu objektov v obrázkoch.

Koncept Mask-RCNN je veľmi jednoduchý. Faster-RCNN má na každého kandidáta dva výstupy a to bounding-box a triedové označenie (class–label). Bounding-box ohraničuje takzvaný región záujmu (RoI –Region of Ineterest ). Mask-RCNN ku tomu pridáva tretí výstup a to masku segmentácie. Táto maska je generovaná nie z celého obrázku, ale z RoI.




Obr. 3.14: Obrázok znázorňujúci fungovanie architektúry Mask-RCNN, prebratý [11].

Celková stratová funkcia pri trénovaní tejto siete je veľmi jednoduchá.

$$L = L_{cls} + L_{box} + L_{mask}$$

Klasifikačná stratová funkcia $L_{cls}$ je krížová entropia [19] medzi dvoma triedami (objekt vs. nie objekt). $L_{box}$ je regresná stratová funkcia bounding-boxu [19]. $L_{mask}$ je stratová funkcia masky implementovaná ako priemer binárnej krížovej entropie. [12].

### Architektúra

Architektúra Mask-RCNN sa skladá z hlavy a z kostry. Tento prístup umožnuje použiť rôzne architektúrne kostry ako Resnet-50 alebo Resnet-101 na základe požiadavkov na hardware.

Hlava má rovnakú implementáciu ako Fast-RCNN [12]. Hlava predikuje bounding-boxy. Mask-RCNN sa skladá z dvoch základných stavov [11]. Prvý stav sa nazýva Region Proposal Network (RPN). Tento stav navrhne kandidátov na bounding boxy. V druhom stave paralene Mask-RCNN predikuje binárnu masku a triedu pre daného kandidáta. Toto je rozdiel oproti iným sieťam, ktoré predikujú triedu na základe binárnej masky.

[5] https://github.com/facebookresearch/detectron2

# Kapitola 4

# Návrh konvolučných neurónových sietí pre segmentáciu a detekciu zubov na 3D modeli čeľuste

Táto kapitola opisuje cieľ a definíciu práce, návrh anotačného programu a aplikácie. Poukazuje, aký je rozdiel v existujúcich riešeniach v rámci 3D modelov.

## 4.1 Cieľ práce

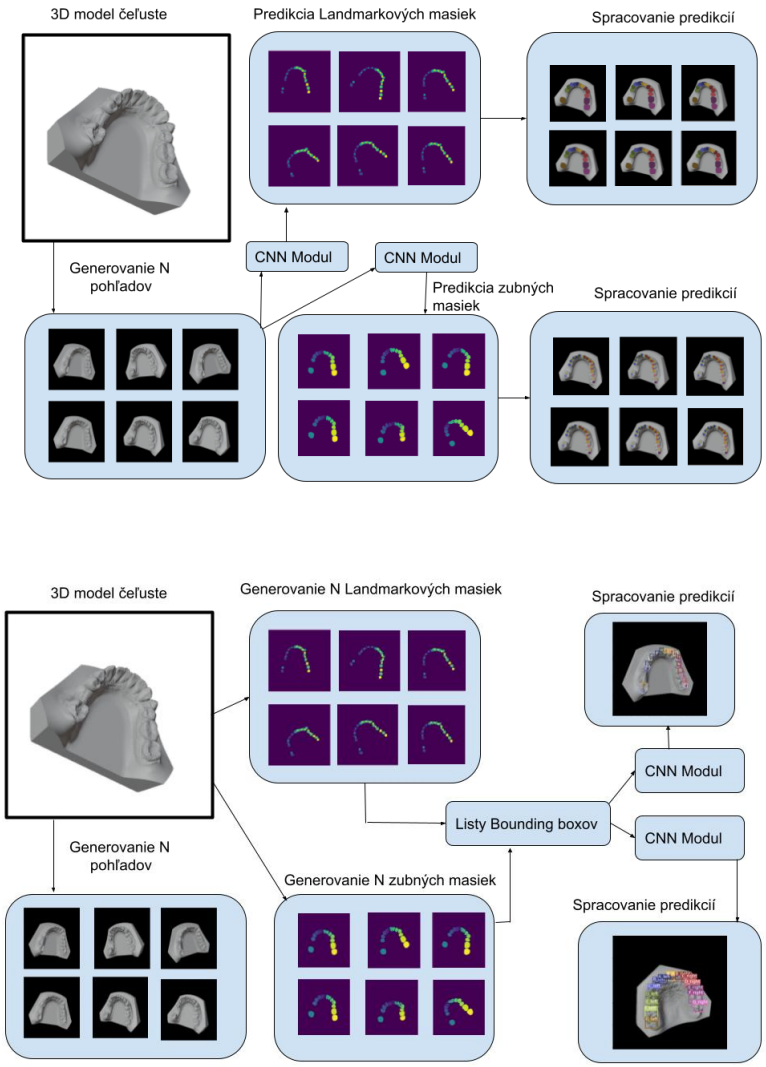
Hlavným cieľom práce je zautomatizovanie procesu označovania pozície zubov a landmarkov na 3D modely zubov používaných v čeľustnej ortopédií (ortodoncia). Tento cieľ má tri hlavné použitia v priemysle:

- zautomatizovanie označovania zubov a landmarkov, čo skráti čas a zjednoduší prácu pre zubára pred prípravou na operáciu poprípadne prípravy na strojček
- experimentálne testovanie Multi–view metód v medicínskom priemysle.
- porovnávanie Image segmentation a Object detection v medicínskych dátach.

Cieľom práce je aj implementovať aplikáciu na analýzu 3D modelov zubov. Veľkou nevýhodou datasetov je to, že sú omedzené. Aj pri použití anotačného nástroja na generovanie prakticky nekonečného datasetu, vždy bude dataset omedzený. Až po anotácií datasetu prídeme na veci, ktoré by stáli za vyskúšanie. Ktoré pohľady, uhly, ale aj nezrovnalosti v modeloch robia sieťam problém. Preto je dobrým riešením manuálne testovanie modelov pre generovanie nových vstupov a tým pádom generovanie nových situácii ako zareaguje sieť.

Aplikácia by nemala neslúžiť ako náhrada ku profesionálnemu programu ako je BlueSkyBio. Aplikácia je vytvorená len na testovacie účely a zistenie chýb, kde všade modely neurónových sietí robia chyby.

Anotačný nástroj by mal byť schopný vygenerovať $N$ pohľadov na daný model s $N$ binárnymi maskami, ktoré budú použité pri trénovaní konvolučných neurónových sietí. Analogicky dané binárne masky sa dajú ohraničiť a vytvoriť bounding-boxy, čo bude slúžiť ako výstup do sietí na detekciu.



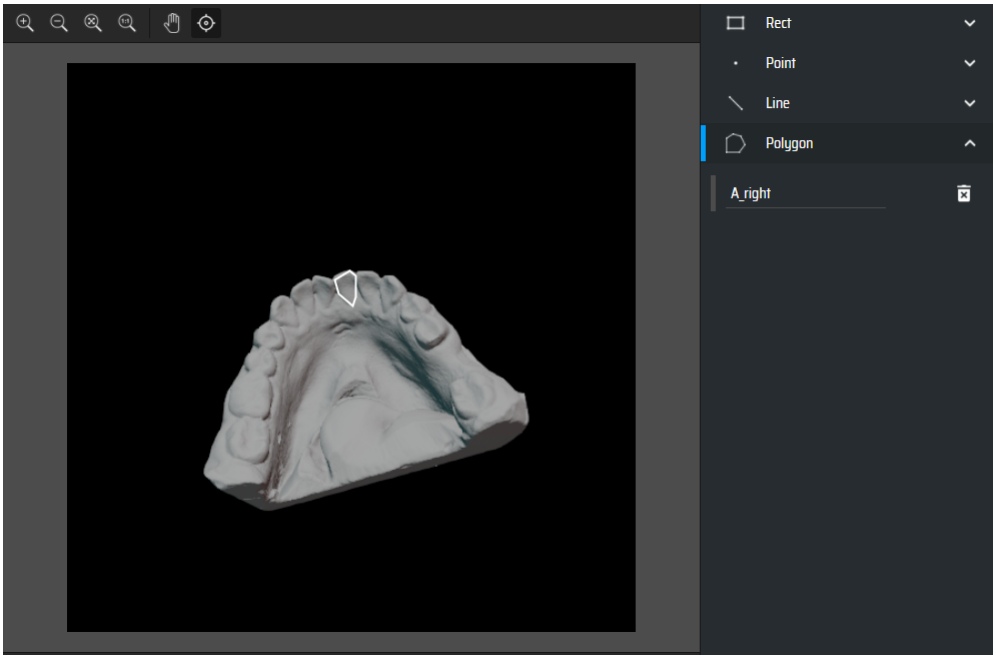

Obr. 4.1: Znázornenie v rámci segmentácii a detekcie zubov a landmarkov po aplikovaní konvolučnej neurónovej sieťe CNN.

## 4.2 Anotačné nástroje

K učeniu s učiteľom 3.1 je potrebný dataset so vstupom a výstupom na trénovanie konvolučných neurónových sietí. Google a Kaggle[1] má najväčšiu databázu anotovaných a aj ne -anotovaných datasetov. Je veľmi veľa anotačných nástrojov na anotovanie 2D obrázkov. Medzi najznámejšie patrí Makesense.ai[2].

Avšak pri tejto práci bola poskytnutá ne-anotovaná množina 3D .stl modelov zubov, ktoré boli z pracovného prostredia firmy Tescan. Pri anotácii 3D modelov nastáva problém. Dá sa vygenerovať $N$ pohľadov na 3D model, čo budú 2D obrázky a potom každý obrázok ručne anotovať pomocou makesense.ai. To nie je problém, ak chceme mať 15 pohľadov na jeden 3D model a anotovať landmark na zube. Ak chceme anotovať celý zub, vzniká nám polygón, ktorý je časovo náročnejší ako vytvorenie jedného bodu. Časovo náročné je anotovať ten istý zub 15-krát z pätnástich pohľadov. Pre každý zub je to oveľa náročnejšie.

Keďže pre trénovanie neurónovej siete 15 pohľadov na model je veľmi málo, je potrebných aspoň 100 pohľadov. Časovo to tvorí 400 minút na jeden model. Ak by sme týmto spôsobom chceli anotovať každý model z množiny ( 419 modelov ), tak len samotné anotovanie bez trénovania modelov by trvalo 46 dní, čo je extrémne stále veľa a pritom máme relatívne málo pohľadov na modely. Existujú proprietárne programy na anotáciu 3D modelov, avšak tieto programy sú konštruktérskeho zamerania ako napríklad Autodesk[3]. Vzniká teda potreba vytvoriť časovo efektívny anotačný nástroj/program/skript na 3D modely.




Obr. 4.2: Anotovanie v programe Makesense.ai.

[1] https://www.kaggle.com/
[2] https://www.makesense.ai/
[3] https://www.autodesk.com/

# Kapitola 5

# Realizácia

V tejto kapitole opisujem použité technológie pre vytvorenie datasetu, implementácie neurónových sietí a aplikácie. Všetky technológie boli spustiteľné na Windows 10 verzia 10.0.19043 Build 19043 s procesorom Intel i7-10750H a s grafickou kartou Nvidia 2070 RTX.

## 5.1 Technológie

### Python 3.7

Pre túto prácu som ra rozhodol pre programovací objektový jazyk Python. Hlavným dôvodom pre zvolenie tohto jazyka je prenositeľnosť kódu medzi operačnými systémami ako Windows a Linux a podpora Deep Learning frameworkov na trénovanie neurónových sietí. Vybral som si konkrétnu verziu Python 3.7 , ktorá je najviac stabilná pre frameworky.

### Anaconda

Anaconda je prostredie, ktoré automaticky rieši konflikty medzi jednotlivými balíkmi a frameworkmi v Pythone. Prostredie Anaconda je využívané dátovými vedcami pre podporu a stabilitu prostredí v Python-e. V kóde je príloha na vytvorenie rovnakých balíkov na replikovanie trénovania a spustenie aplikácie.

### Blender 2.93

Blender je program na modelovanie , upravovanie a vytváranie scén a animácii s 3D modelmi. Je to open-source projekt podporovaný v každom operačnom systéme. Najväčšiu výhodu má jeho dokumentácia ku API. Blender je napísaný celý v Pythone a autori sa rozhodli podporovať automatizáciu pomocou skriptov a pluginov v Pythone.

### Pytorch 1.8

Pytorch je výsledok práce Facebooku. Je to open-source knižnicia ktorá podporuje operácie pre neurónové siete. Je výhodný v tom, že podporuje aj výpočet operácii pre balík CUDA, čo sú Nvidia grafické karty.

### Tensorboard

Tensorboard je vizualizačná serverová aplikácia na zobrazenie grafov pri trénovaní neurónových sietí. Aplikácia je vyvíjaná pre framework Tensorflow, ale má aj obmedzenú podporu pre framework Pytorch 1.8.

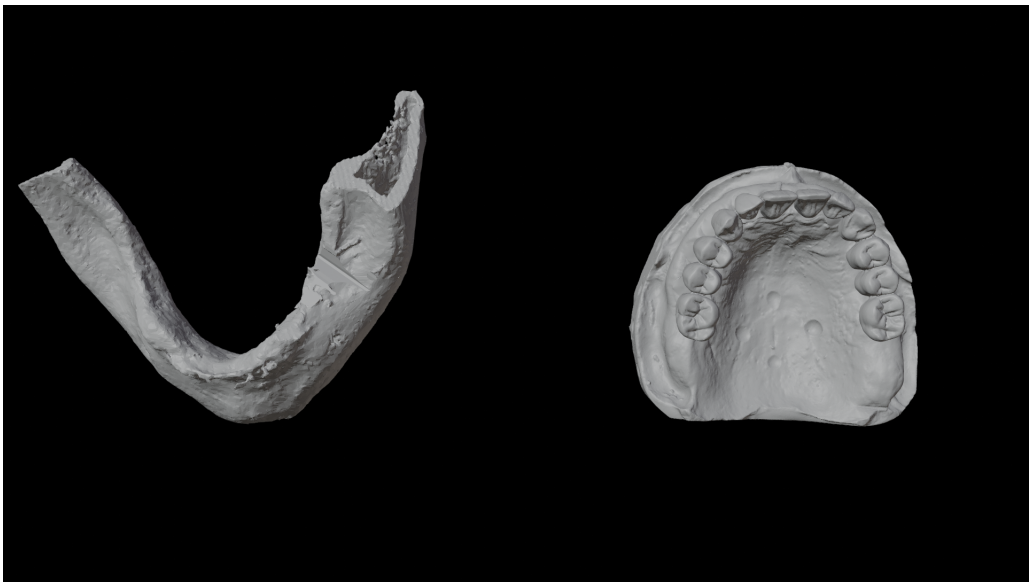
### PyQT 5

Je grafická knižnica pre tvorbu GU I(Graphical User Interface) aplikácií. Je pod licenciou GNU GPL v3, čo znamená, že je voľne dostupná, avšak sa nemôže pomocou nej zarábať, ak sa nezaplatí licencia.

## 5.2 Dataset

Dataset poskytnutý k bakalárskej práci sa skladá z **neanotovaných** STL modelov zubov. Tieto modely su bežne využívané v medicínskom priemysle dentistami v čelustnej ortopédii. STL (Standard Triangle Language) je formát na ukladanie súradnic trojuholníkov v modeli. Ako som spomenul v Návrhu riešenia v 4.2, je potreba eliminovať časovú náročnosť pri anotácii 3D modelov.

V datasete sa nachádzajú aj nepoužiteľné modely, ktoré by spôsobovali šum pri trénovaní neurónových sietí. Pri anotácií boli použité modely, ktoré by takýto šum nespôsobovali.



Obr. 5.1: Porovnavnie zlého a dobrého modela zubov.

### Blender

Blender je grafická aplikácia na úpravu a modelovanie 3D modelov , generovaní animácii a scén. Má podporu API pre Python. Blender sa skladá z dvoch hlavných častí. Z módu a z pracovných sekcií. V objektovom móde sa dá manipulovať s pozíciou objektov na scéne. Vo Vertex móde sa každému modelu dá nastaviť vertex farba. Nie je to však rovnaké ako maľovanie pomocou textúry v textúrovom móde. V pracovných sekciách môžeme nájsť pracovné

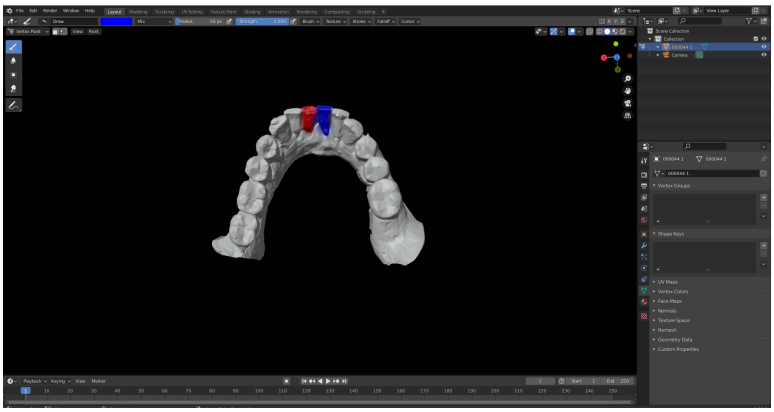
sekcie ako Layout, Sculpting, UV Editing, Animation a Scripting. V tejto bakalárskej práci využívam len dve pracovné sekcie: Layout a Scripting.

### Anotačný nástroj vytvorený v Blender–i

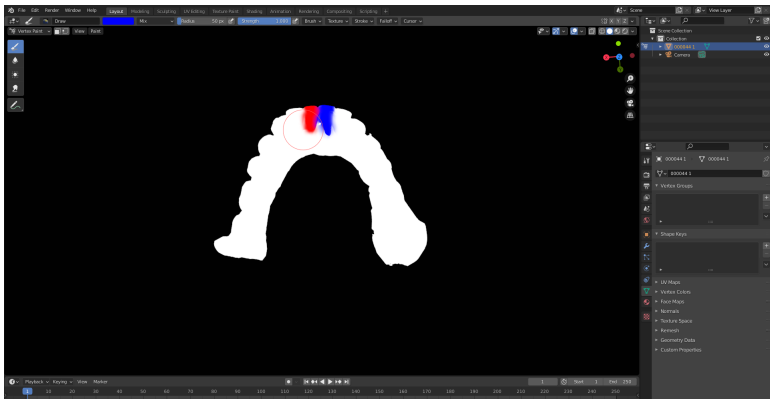
Pre rendrovanie masiek potrebujem dva objekty: kameru a model zubov. Keďže nevyužívam textúru, ale vertex farby, netreba mať svetlo ako objekt.

Každý modelu, nastavujem centroid na stred scény pomocou príkazu `Geometry to origin`. V `Output properties` nastavil som veľkosť rendrovacieho obrázka na $500 \times 500$. V `Camera properties` je nutné zadať `Clip end` na `100000 m`. Pomocou tlačidla pre osi bol nastavený pohľad na zuby podľa obrázku 5.2. Následne som použil príkaz `Align Active Camera to View`. Tento príkaz nastaví kameru na scénu na aktuálny pohľad . Toto je nastavenie v `object mode`.

Každý model rozdeľujem na ľavú a pravú časť. Pre ľavú časť som si zvolil čisto červenú farbu `#ff0000` a pre pravú stranu čisto modrú farbu `#0000ff`. Každý zub má identifikátor. Ak začnem anotovať s rezákmi odpredu, tak ich podľa obrázku 2.2 v teoretickej časti budem nazývať podľa triedy `A` ako `A_left` a `A_right`. Zvolím si `Vertex mode`, kde následne zuby vymaľujem podľa obrázka 5.2.



Obr. 5.2: Vertex obrázok.



Obr. 5.3: Vertex obrázok s kolmým svetlom bez tieňa.

V rámci pracovných sekcii `layout` vymeníme za `scripting`. Do textového editora pridáme script z `blender_script_teeth.py` V globálnych parametroch skriptu je potrebné

zadať meno modelu, cestu, kde sa má uložiť render a typ zubu. Taktiež sa dá upraviť aj veľkosť uhla a krok. V obrázku 5.2 je typ zubu `A`. Následne stačí skript spustiť v blenderi. Pre veľkosť uhla -35 bude skript otáčať model po osy x, y a z od -35 do 35 stupňov. Ak sa mu nastaví krok 5, spolu vznikne okolo 5488 súborov masiek a pohľadov o približnej veľkosti 844 MB. To je okolo 2744 pohľadov, čo je extrémne veľa pohľadov anotovaných za pomerne krátky čas 40 minút. Analogicky použijem tento spôsob aj pri tvorení landmarkových masiek, ale na miesto vyfarbenia zubu vytvorím krúžok v danom landmarkovom strede.

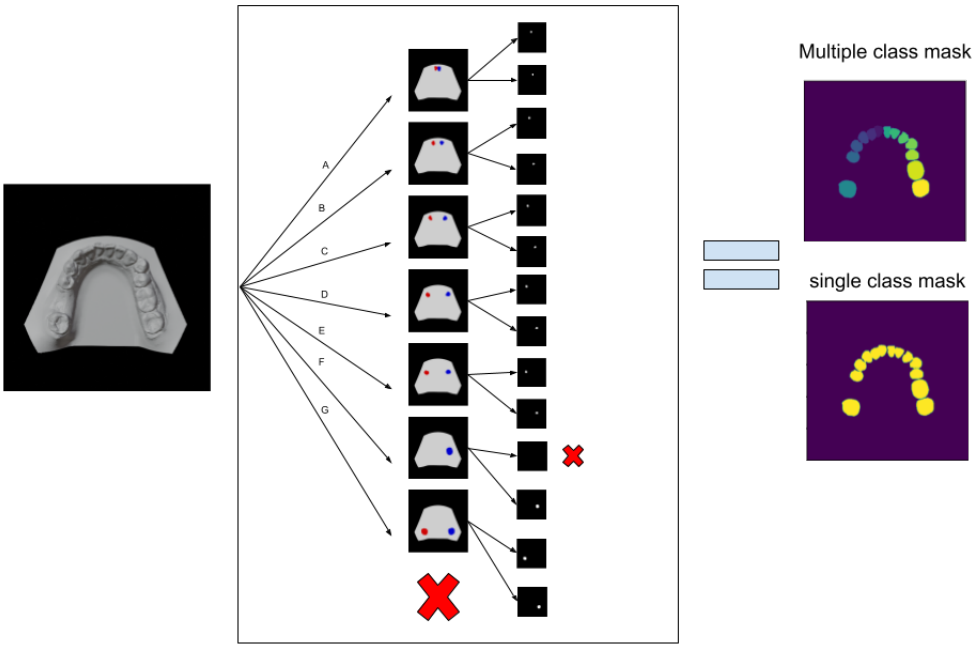
### Možné problémy pri farbení

Pri farbení zubu si treba dať pozor na viacero vecí. Zuby medzi sebou majú spojitú plochu, ktorá nepatrí ani jednému zubu. Je to buď ďasno alebo pozostatok odliatku. Taktiež maľovanie vo `Vertex mode` neukáže nedokonalosti masky pri nevyfarbených miestach ako sú hrany zubov či dokonca fliačky bielej v strede zubov, ukážka v obrázku 5.4.

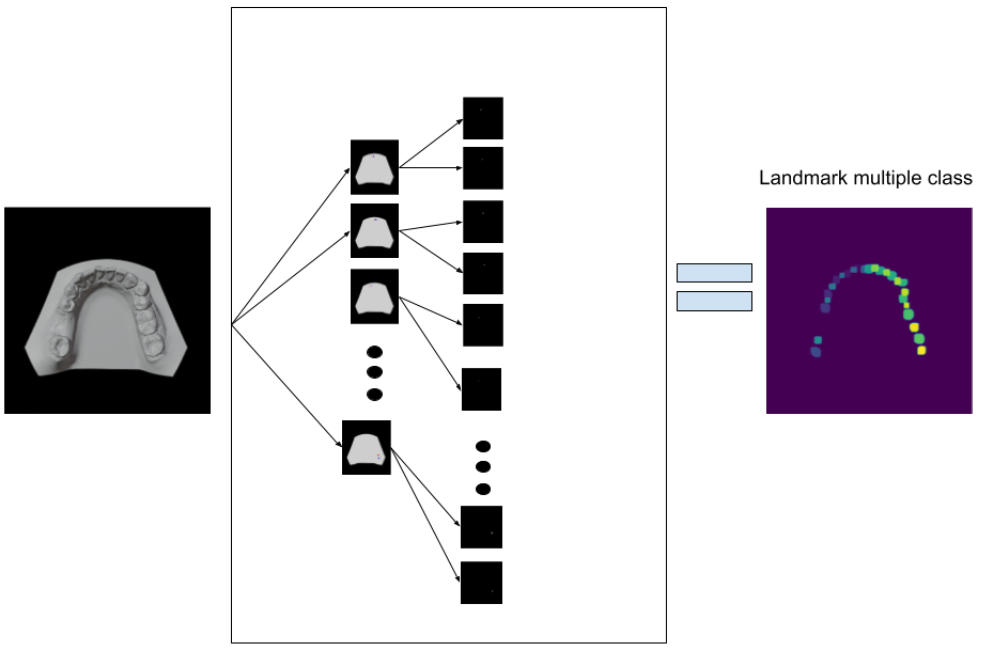
Obr. 5.4: Zelená farba znázorňuje časť zubu, ktorá ani jednému zubu nepatrí.

### Multi-view rendering pipeline

Postup pri anotácii farieb opakujem na každý typ zubov bez zmeny lokácie daného 3D modelu. Tento proces je náročný na čas, keďže aplikujem iba 2 farby na ľavú a pravú časť. Nedá sa aplikovať každá farba na každý zub kvôli nedostatku farieb na typy zubov. Po vygenerovaní každého vertex obrázku s kolmým svetlom bez tieňa, sa pomocou skriptov vyrežú binárne masky do príslušnej triedy. Výhoda tohto procesu je, že vznikne na každý pohľad každá maska oddelene v príslušnom priečinku pomenovaných po daných triedach. Znázornenie procesu je v obrázkoch 5.5 a v 5.6.



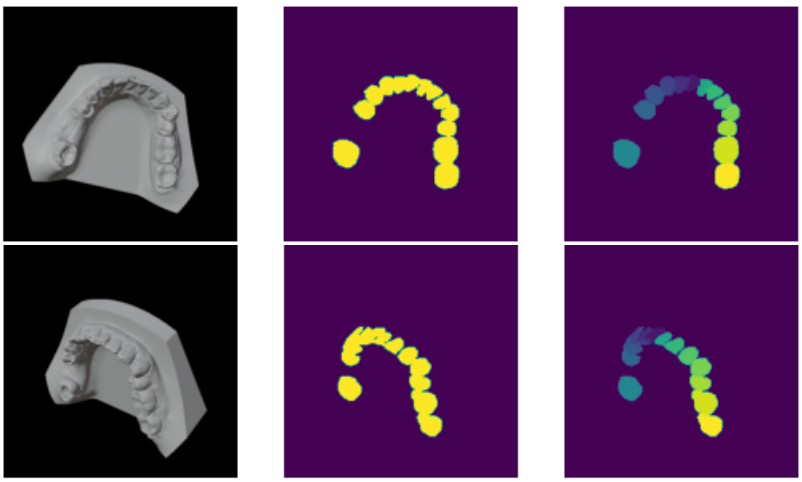

Obr. 5.5: Znázornenie tvorby masky pri zuboch.



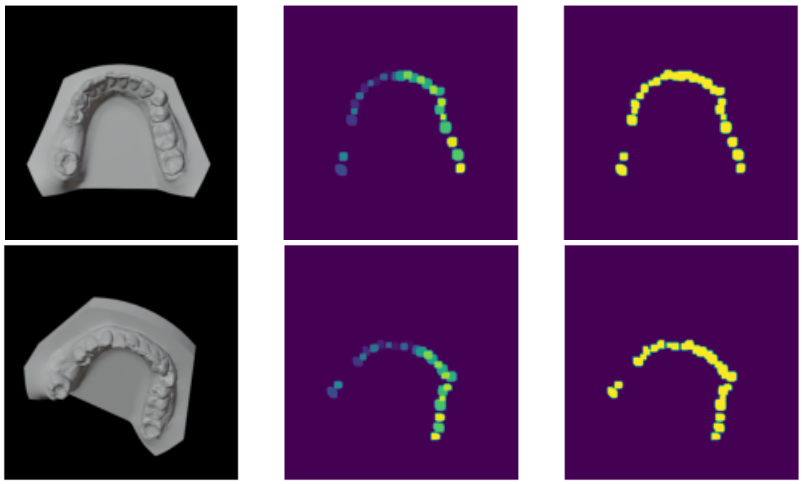

Obr. 5.6: Znázornenie tvorby masky pri landmarkoch na zuboch.

### Masky

Multi-view rendering pipeline nám dovoľuje vytvoriť extrémne veľký a presný dataset. Tento dataset má o dosť menej chybných anotácii, ak nejaké vôbec má oproti iným datasetom. Znázornenie pohľadov na dané masky sú v obrázkoch 5.7 a v 5.8. Spolu pre dané datasety bolo vygenerovných 157 GB vstupov a výstupov, z toho pre anotáciu zubov bolo použitých 13 modelov a anotáciu landmarkov 6 modelov. Každý model má 2744 pohľadov, pričom pri binárnych maskách pri zuboch vzniklo okolo 38416 oddelených binárnych masiek a pri landmarkoch vzniklo 71344 oddelených masiek. Kvôli pamäťovej náročnosti datasetu som neanotoval viac zubov, aj keby mi to táto metóda dovoľovala.



Obr. 5.7: Znázornenie $N$ pohľadov otáčania modelu s maskami pre zuby.



Obr. 5.8: Znázornenie $N$ pohľadov otáčania modelu s maskami pre Landmarky.

## 5.3 Trénovanie neurónových sietí na datasete

V tejto časti, vysvetlím konkrétnu detailnú implementáciu sietí a zároveň priebeh trénovania, tak aby čitateľ mal základný prehľad ako fungujú dané neurónové siete na datasete.

### Mask-RCNN

Mask-RCNN je prvá neurónová sieť, ktorú som implementoval v tejto práci. V rámci pamäťovej náročnosti je použitá kostra Resnet-50, kvôli nedostatočnej veľkosti pamäte grafickej karty pre Resnet-101. Využíva Region proposal network (RPN) na predikciu ohraničenia kandidátov pomocou bounding-boxov. Je vylepšená oproti Fast-RCNN pridaním paralerného klasifikátora na predikciu binárnych masiek.

#### Defíncica datasetu

Dataset je definovaný na základe triedy `torch.utils.data.Dataset`. Táto trieda ma metódy `__len__` a `__getitem__` pričom `__len__` znázorňuje veľkosť datasetu a `__getitem__` vráti dvojicu definovanú ako `Image` a `Target`. `Image` musí byť definovaný ako vstup PIL obrázku z frameworku PILLOW[1]. `Target` očakáva nasledovné triedy a to:

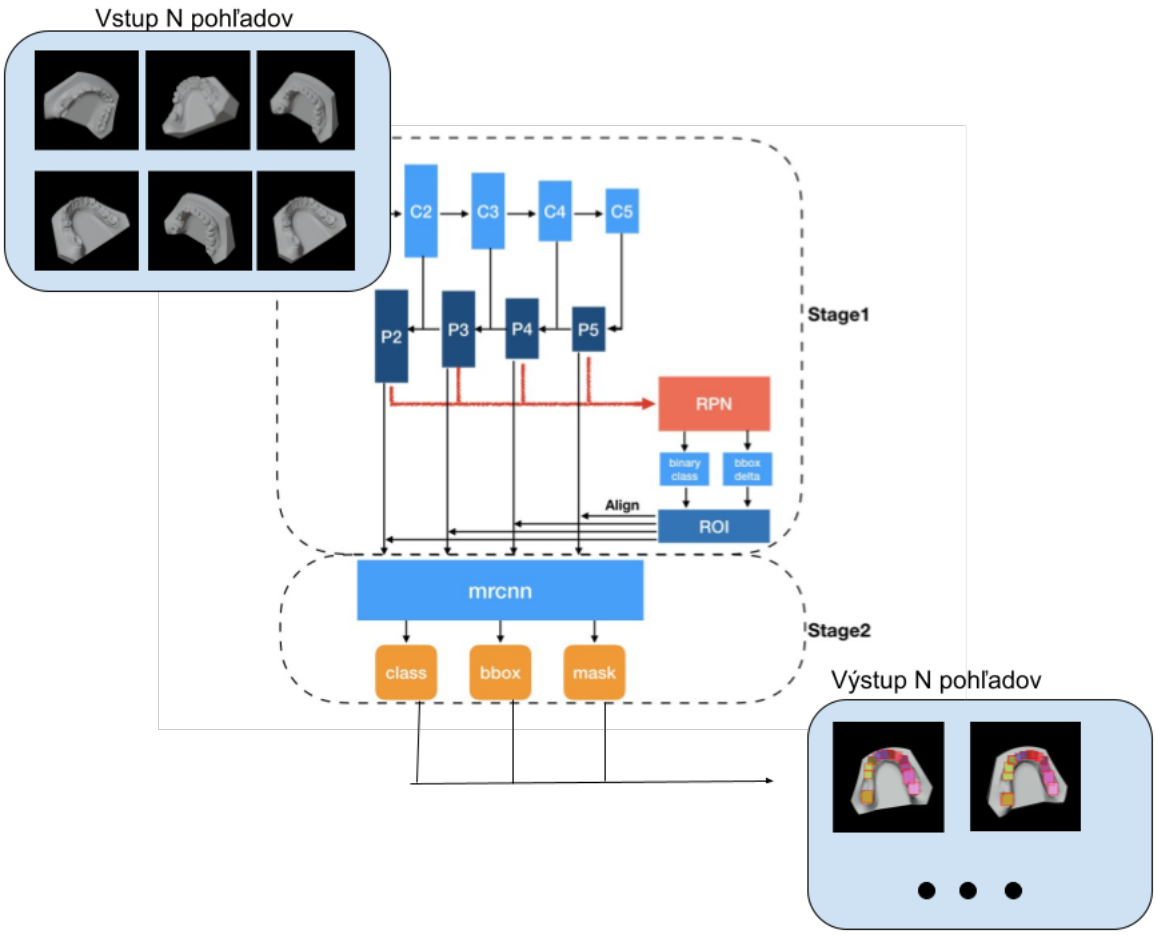
- `boxes (FloatTensor[N, 4])`
- `labels (Int64Tensor[N])`
- `masks (UInt8Tensor[N, H, W])`
- `area (Tensor[N])`

Tvorba masiek spočíva v pridaní každej binárnej masky do jedného objektu numpy–array, z ktorého sa následne stane tensor. Boxy sa dávaju vo formáte `[xmin,ymin,xmax,ymax]`. Jednotlivé súradnice pre daný objekt sú tým pádom počítané z binárnych masiek pomocou `np.min` a `np.max`. Obsah bounding boxu zadaného súradnicami `[xmin,ymin,xmax,ymax]` je `area`.Tensor tried, ktoré sa nachádzajú na obrázku je `labels`. Dataset bol rozdelený na trénovací a testovací časť a to ku pomeru 7 ku 3.

#### Implementácia

Implementácia siete je založená na oficiálnom kóde Mask-RCNN[2] vo frameworku Pytorch. Ako optimalizátor siete bol zvolený SGD optimalizátor s momentom 0.9, learning ratom 0,005 a s penalizáciou weight decay 0,0005 . Optimalizátor implementuje stochastický gradient descent. Batch-size je 2 pretože viac daná grafická karta nezvládne. Veľkosť obrázkov pre zuby je o veľkosti $416 \times 416$. Pre landmarky je to veľkosť $256 \times 256$.

[1] https://pillow.readthedocs.io/en/stable/
[2] https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html




Obr. 5.9: Znázornenie procesu pri použití reálnych vstupov a reálneho výstupu. Architektúra prebratá z [25].

### YOLO v3

Yolo v3 v rámci trénovania je sieť, ktorá z daných troch sietí je na druhom mieste v rámci dokončenia rýchlosti dokončenia epochy. Táto sieť má 2 veľké problémy: v porovnaní na trénovacom datasete YOLO je schopný klasifikovať až po 50–tich epochách, i keď je batch_size nastavený na 16. Druhý problémom tejto siete je, že loss funkcia pri landmarkoch na datasete nevie vôbec nadobudnúť hodnotu menšiu ako 2,5. Dôvodov môže byť viacero, ale osobný názor je, že landmark masky pre YOLO v3 sú príliš malé, i keď autori tvrdia, že podporuje malé objekty. Optimalizátor pre túto sieť bol zvolený Adam. Learning rate bola zvolená hodnota 0,00001 , penalizácia weight decay 0,0001. V rámci mierky boli použité 3 mierky a to $\frac{1}{32}$, $\frac{1}{16}$ a $\frac{1}{8}$.

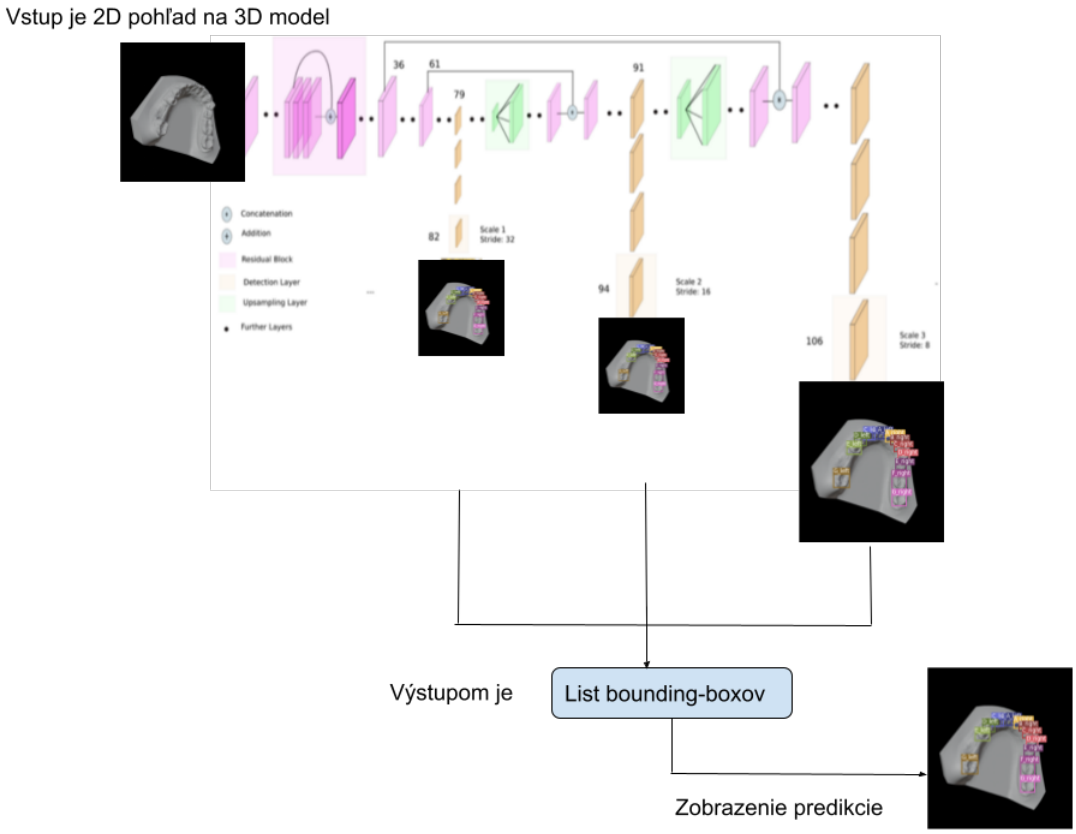
#### Definícia datasetu

Dataset je tak isto ako v Mask RCNN definovaný na `torch.utils.data.Dataset`. Rozdiel je však vo výstupoch do siete. V `targets` sa podľa indexu rozdeľujú tri mierky, kde do každej mierky sa pridajú aktuálne súradnice pre bounding boxy v danej mierke. Formát bounding boxov je iný oproti Mask-RCNN. Obsahuje 5 hodnôt a to `[x_center, y_center, width, height, label]`.

#### Implementácia

Implementácia siete Yolo v3 je založená na oficiálnom papieri [18] a na pytorch verzii [3]. Kostru siete tvorí darknet-53, ktorého architekúra je definovaná ako premenná `config` v súbore `model.py` Sieť je tým pádom ľahko configurovateľná pre ďalšie pokusy, čo nie je cieľom tejto práce.

[3] https://github.com/aladdinpersson/Machine-Learning-Collection/tree/master/ML/Pytorch/object_detection/YOLOv3




Obr. 5.10: Sémantické znázornenie s reálnymi vstupmi a výstupmi v Yolo v3, architektúra je zobrazená podrobne v obrázku 3.13. Obrázok upravený podľa potreby z [11].

### U-NET

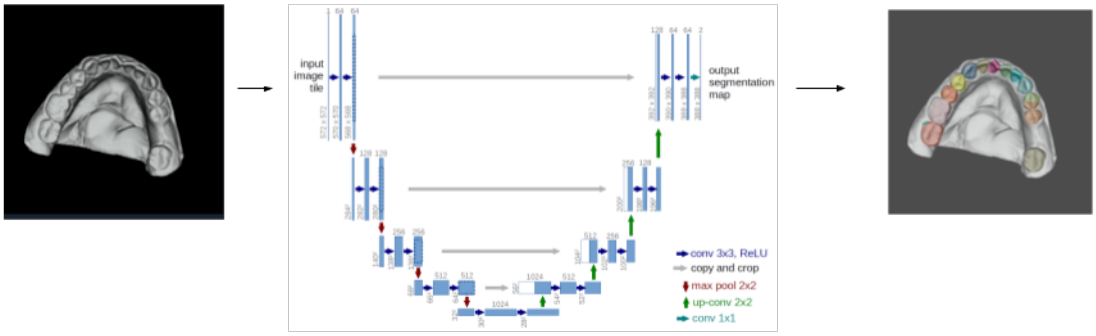
Unet na základe architektúry, ktorá je dramaticky menšia oproti ostatným architektúram v tejto práci, je najrýchlejšie natrénovateľná. Na testovacom datasete pri trénovaní som dosiahol úspešnosť 91 %. V rámci manuálneho testovania je aj najpresnejšia pre landmarky a masky zubov.

### Definícia datasetu

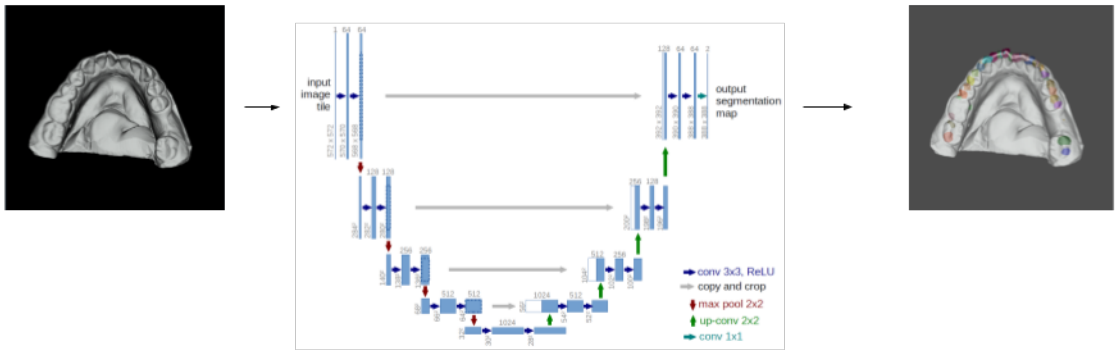
Dataset pre U-net je definovaný ako dvojica `'image': torch.from_numpy(img)` , a dvojica `'mask': torch.from_numpy(mask)` V nastavenia globálnych parametrov je možné zmeniť veľkosť obrázku na základe pomer. Maska je však oproti iným architektúram U-netu tvorená tvarom len $H \times W$, kde rôzne triedy sú predstavené ako iné číslo.

### Implementácia

Ako optimalizátor bol zvolený Adam. Learning rate je definovaný ako 0.0001, Weight decay je definovaný na 1e-8, čo je najmenšia hodnota oproti ostatným neurónovým sieťam. V implementácii je generizovaný prechod dole a prechod hore pretože tento krok sa veľmi často opakuje, operácie smerom hore sú bilineárne.




Obr. 5.11: Znázornenie architektúry s reálnymi vstupmi a s reálnymi výstupmi pre zuby.




Obr. 5.12: Znázornenie architektúry s reálnymi vstupmi a s reálnymi výstupmi pre landmarky.

### Priebeh trénovania

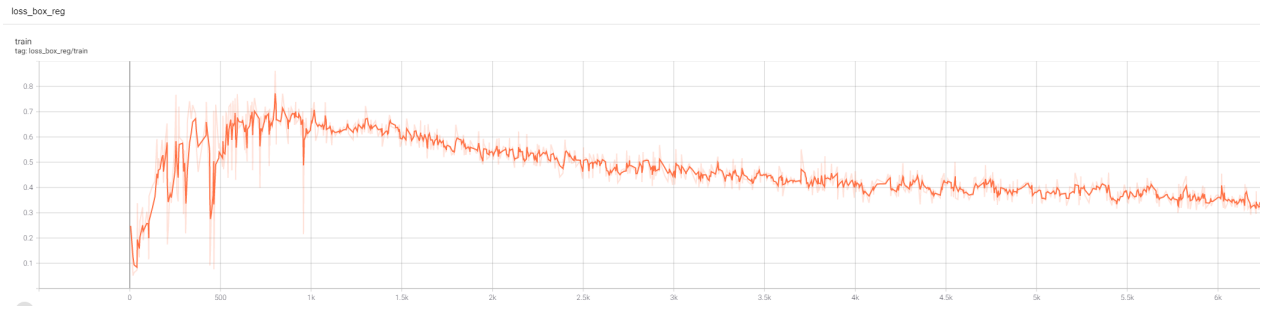
V rámci trénovania U-Net a MASK-RCNN boli zachytené grafy z tensorboardu. Pre sieť Yolo sa mi nepodarilo spojazdniť zapisovanie pri trénovaní do tensorboardu.

Dá sa povedať, že v rámci trénovania táto sieť je najpomalšia, pretože jedna epocha trvá minimálne 1 hodinu, či už pre landmarky alebo pre zuby. Architektúra a spôsob predikovania tejto siete však dovoľuje mať veľmi dobrú úspešnosť na testovacom datasete už po prvej epoche.


Obr. 5.13: Priebeh trénovania na sieti U-net landmark.

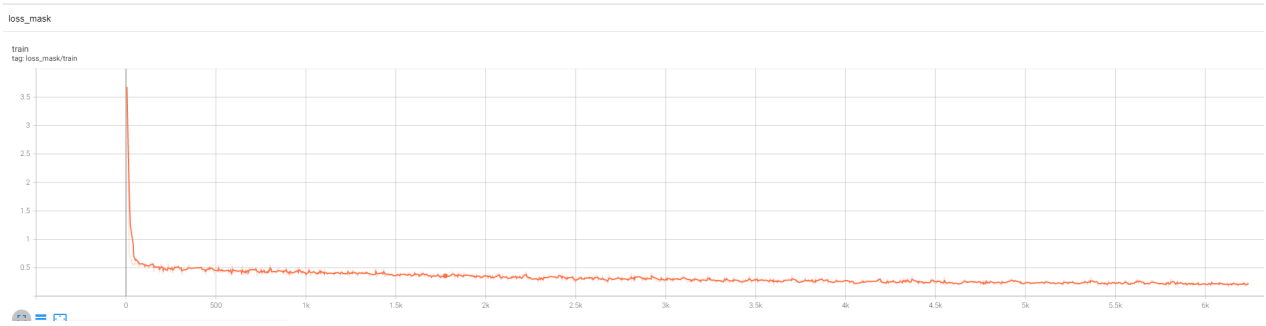

Obr. 5.14: Priebeh trénovania na sieti U-net teeth.

Ako môžme vidieť siete, sú viac úspešné pri trénovaní segmentácie len zubov oproti trénovaniu landmarkom. Mask-RCNN oproti U-netu je pomalý ,pretože po 6000 krokoch sotva prekročil úspešnosť 0.4.




Obr. 5.15: Priebeh trénovania na sieti Mask-RCNN podľa box regresie na landmarky

Záverom tohto trénovania je to, že najrýchlešie a najspolahlivejšie sa trénuje U-NET ako taký, oproti Mask-RCNN a YOLO pričom musím zdôrazniť, YOLO sa mi ani nepodaril natrénovať detekciu landmarkov.




Obr. 5.16: Priebeh trénovania na sieti Mask-RCNN podľa maskovej segmentácie na landmarky.

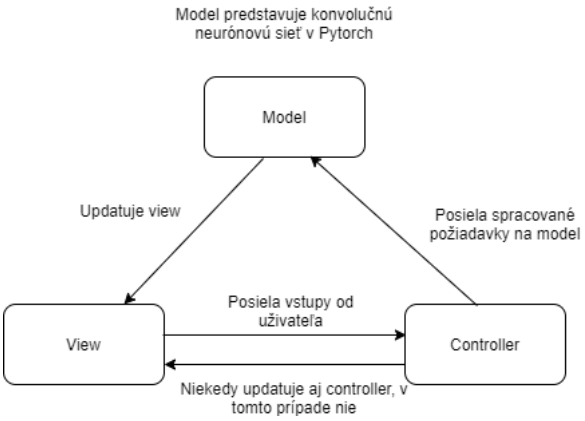
## 5.4 Aplikácie na analýzu 3D modelov čeľuste

V tejto sekcii budem rozoberať implementáciu a architektúru aplikácie na analýzu 3D modelov čeľuste.

### Architektúra

Pri architektúre som využil architektúru Model-view-controller. Je to základná architektúra pre aplikácie a pre potreby testovania bude dostatočná. V rámci architektúry som musel vybrať knižnicu PyQt, kvôli natrénovaným modelom v Pythone.

PyQT sa stará o celú logiku Model-view-controlleru. Má controller, ktorý čaká na vstupy od uživateľa. Model čo je v našom prípade model neurónovej siete a má view kde v nekonečnej smyčke vygresluje dialog.



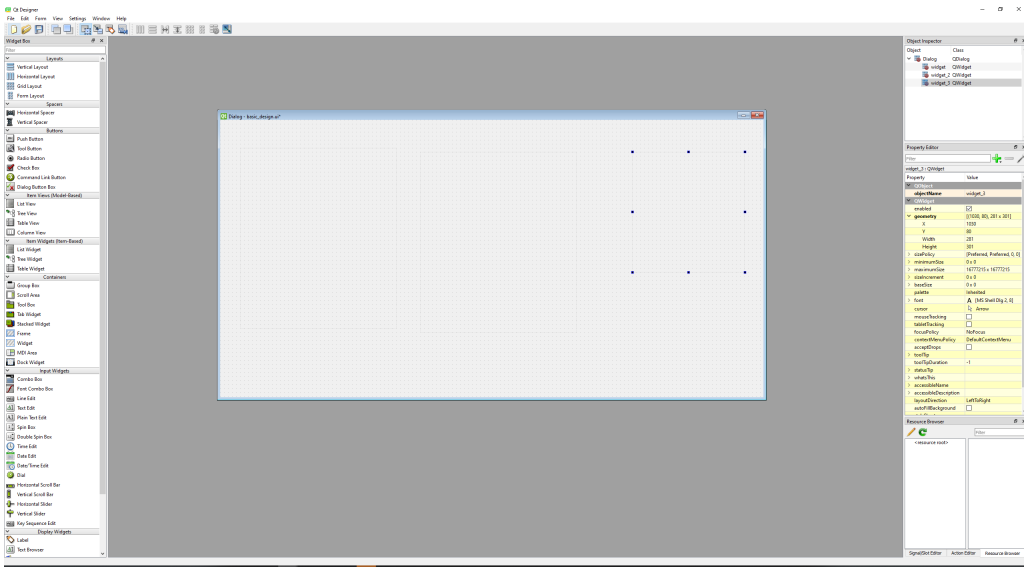

Obr. 5.17: Architektúra MVC.

PyQt aplikácie majú podporu v každom operačnom systéme. Ku ich najjednoduchšiemu vytvoreniu sa používa aplikácia Qt Designer[4]. Qt Desinger je návrhárska aplikácia pre vytvorenie grafického užívateľského rozhrania v skratke GUI. V rámci Qt Designeru vytvoríme len jeden dialóg s troma widgetami. Widget je oblasť, ktorá slúži na rendrovanie obrázkov alebo na interkatívne spracovanie požiadavkov užívateľa. Jeden widget bude pre interaktívnu aplikáciu VTK, do ktorej budeme nahrávať 3D .stl model a následne s ním interagovať. Druhý Widget bude slúžiť ako výsledok klasifikácie daného vstupu z VTK widgetu. Tretí widget nám bude zobrazovať len najideálnejší pohľad pre zuby.

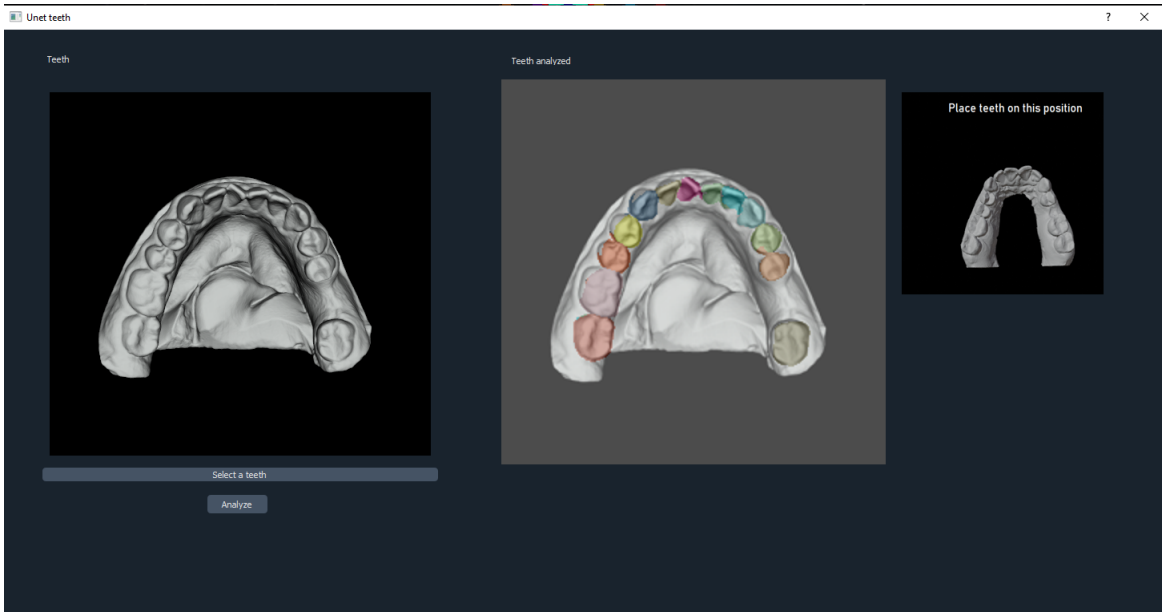
V rámci architektúry nemôžeme načítať do pamäte RAM a do pamäte grafickej karty naraz všetky modely konvolučných neurónových sietí. Zvolil som preto jednoduchšie riešenie a vytvoril 5 inštancií danej aplikácie pre každý jeden model (model YOLO v3 pre landmarky je nepoužitelný).

K aplikácii som pridal štylistický tmavý design na základe väčšej obľúbenosti medzi programátormi. Výsledná aplikácia je znázornená v obrázku 5.19.

[4]https://doc.qt.io/qt-5/qtdesigner-manual.html



Obr. 5.18: Šablona na vytvorenie PyQt aplikácie pomocou Qt Designera.




Obr. 5.19: PyQt5 aplikácia verzia U-net teeth.

# Kapitola 6

# Testovanie a výsledky

## 6.1 Výsledky na anotovanom dataset

| Výsledky sietí na anotovanom datasete | | |
|---|---|---|
| Názov sieťe | Zuby | Landmark |
| U-net | 91,126% | 83,091% |
| Yolov3 | 85% | X |
| Mask-RCNN Resnet-50 | 87,321% | 74,645% |

Tabuľka 6.1: Výsledky úspešnosti na testovacom datasete.

Ako môžeme vidieť na tabuľke, celkovo detekcia zubov je úspešnejšia ako detekcia landmarkov. Je to zapríčinené tým, že landmarky boli generované ako fľaky spreja pri anotácii zubov. Z toho vyplíva, že jednotlivé masky nemali rovnaký tvar a veľkosť pre jednotlivé landmarky. Tento problém som sa snažil riešiť pomocou funkcie Gaussian_Blur, ale bez efektu. Pravdepodobne treba landmarky detekovať pomocou heatmap metódy. Najúspešnejšia sieť je U-Net. U-net ako jediná nemá predikciu bounding-boxov, čo má asi vplyv na detekciu.

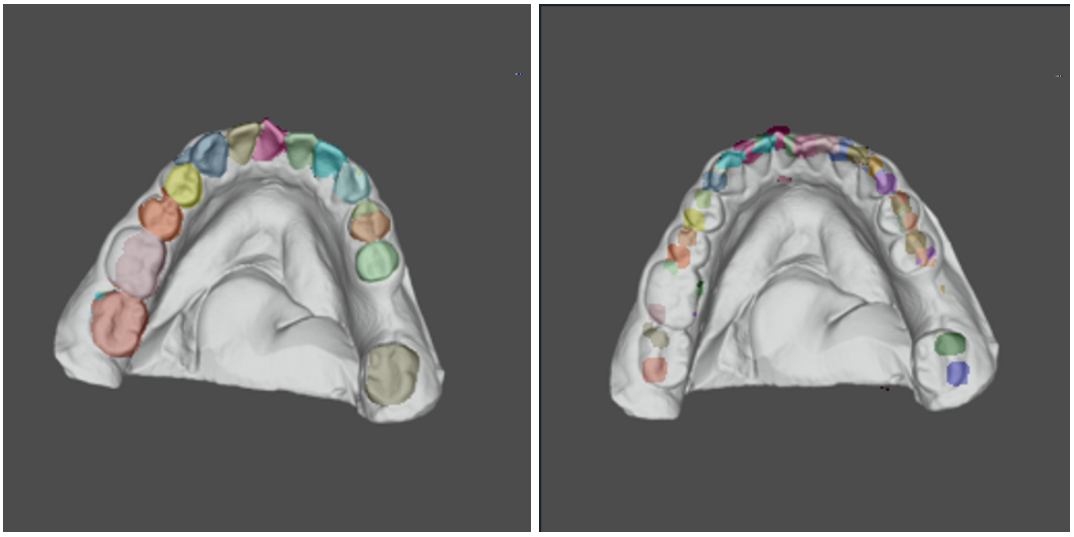
## 6.2 Testovanie pomocou implementovanej aplikácie

Základným testom je overenie správnosti modelu, ktorý bol použitý v datasete pri trénovaní. Ak chceme začať pri detekcii a segmentácii zubov, tak U-net a Mask-RCNN s Yolo v3 zvládajú klasifikovať zuby, ale spravia chyby. Je použitý základný pohľad, kde model nemá žiadnu rotáciu.

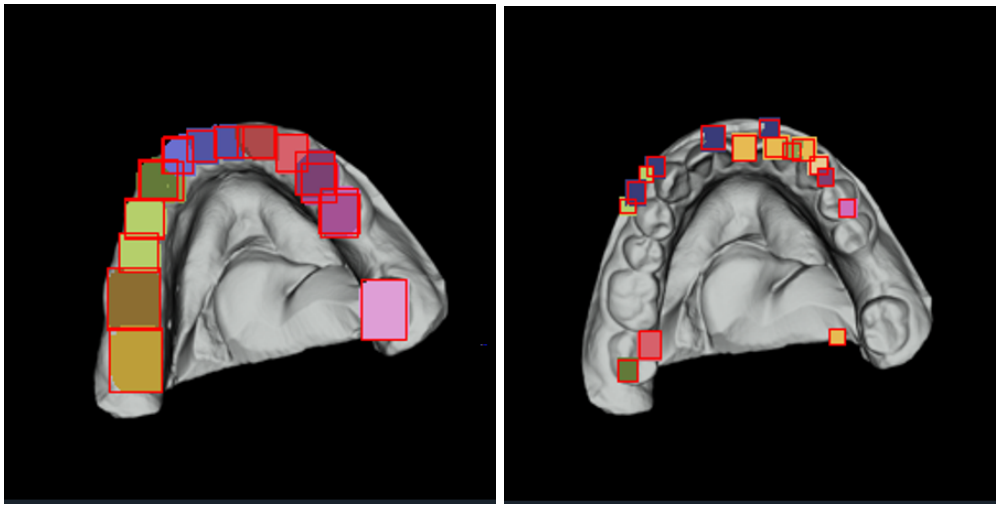
U-net sieť základné pohľady celkom zvláda ako aj pre zuby tak aj pre landmarky. Je vidno pochybenie v strede zubu triedy `D_right`, ale nie je to nič závažné. Pre landmarky sa na niektorých zuboch predikujú až 3-krát. Špecificky `C_right` a `D_right`.

Mask-RCNN taktiež zvládne základné pohľady pri zuboch avšak textúra modelov vo VTK pravdepodobne neumožňuje správnu detekciu landmarkov. Z toho vyplýva Mask-RCNN nie je vhodný na landmarky generované týmto spôsobom.

Yolo pri predikcii zubov sa viackrát pomýlil. Detekoval dvakrát `D_left` zuby a zub kategórie `A_left` je predikovaný v pravej časti zubov.

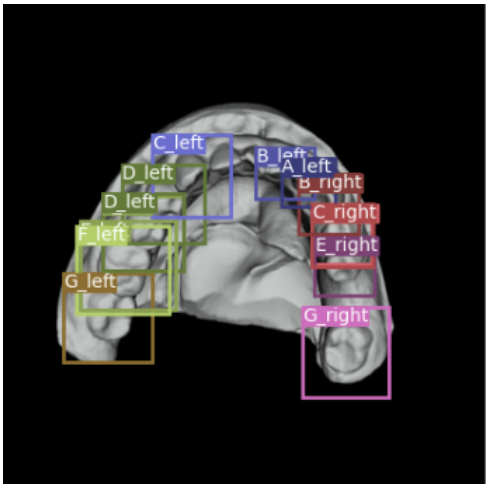


Obr. 6.1: U-net predikcia zubov a landmarkov.

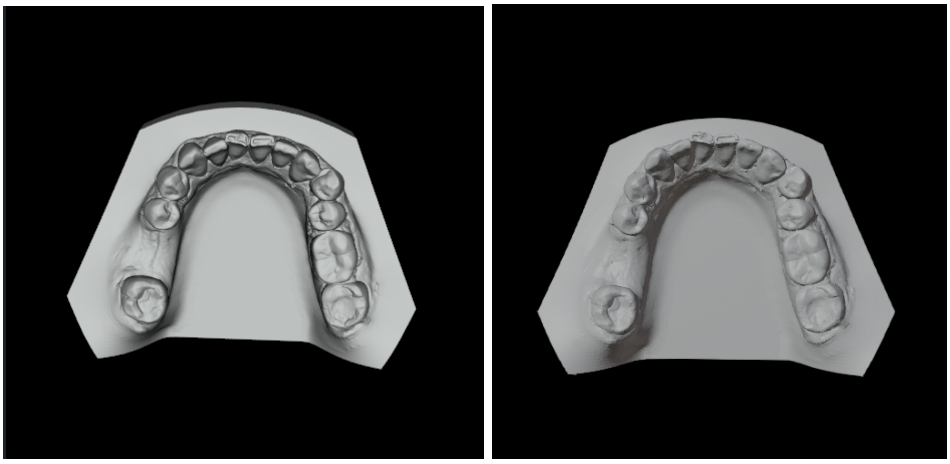


Obr. 6.2: Mask-RCNN predickia zubov a landmarkov.

Z prvotného testu môžeme predikovať, že Mask-RCNN a U-net z vizuálnej stránky sú najpresnejšie. Je to ovplyvnené viacerými faktormi. Dataset pre landmarky nie je generovaný pomocou Gaussovských heatmap, preto dané landmarky majú rôznu veľkosť a šírku. Tieto testy sú ovlyvnené aj textúrou VTK knižnice oproti Blender textúry modelu. Môžeme pozorovať rozdiel v Obr. č. 6.4 vo farbe ako aj rozdiel v tieni zubov. Pravdepodobne Mask-RCNN a YOLO v3 má problém určiť bounding-boxy oproti U-Net, ktorá automaticky aplikuje segmentáciu.



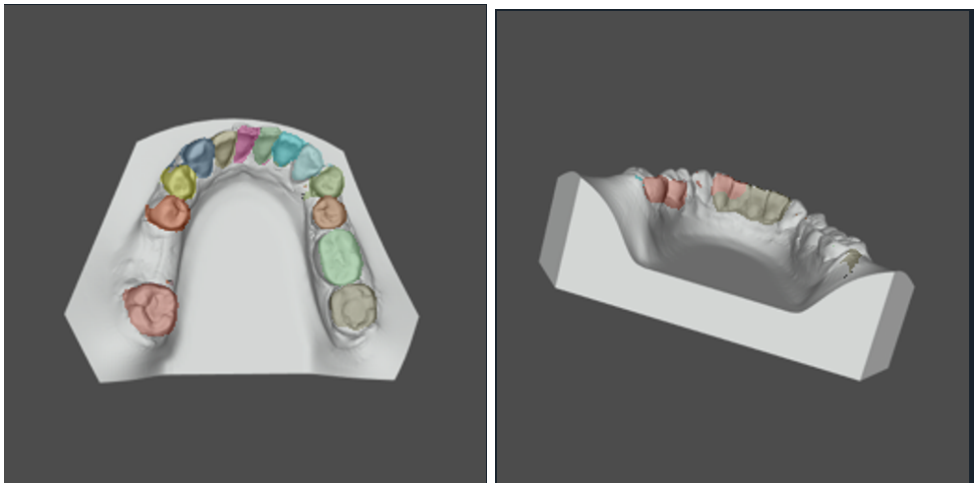

Obr. 6.3: Yolo detekcia zubov.



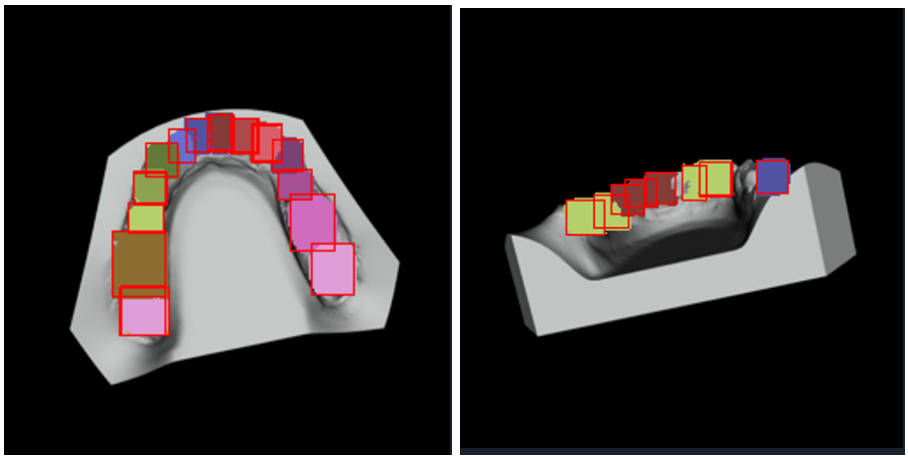
Obr. 6.4: Vľavo model VTK vs model vyrendrovaný v Blenderi.

## 6.3 Experimenty

Keďže najviac sú úspešné siete pre detekciu a segmentáciu zubov Mask-RCNN a U-Net, tak som ich zaradil do experimentálnej časti. Prvý test sa skladá z natočenia modelu zhora a zboku. Siete pri pohľade zboku mali problém správne klasifikovať triedu, nielen zuby.



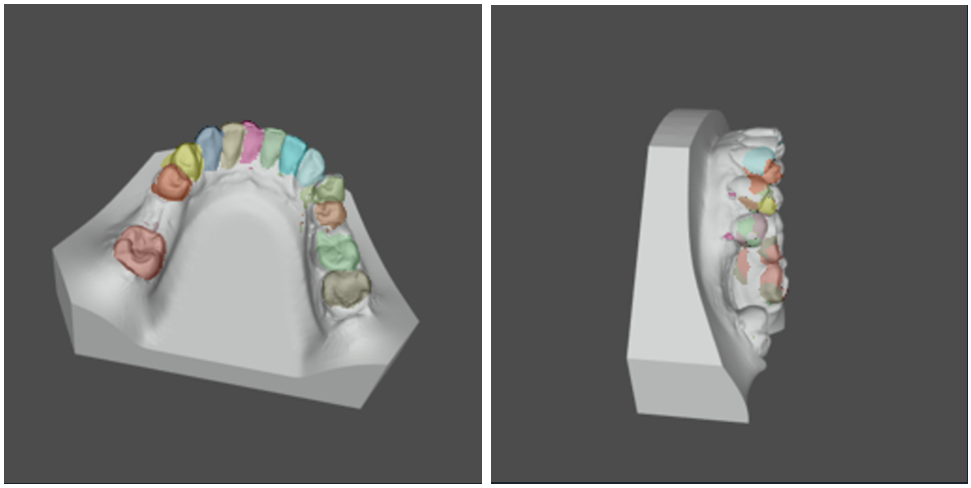
Obr. 6.5: Pohľad z hora a z boku z južnej strany.



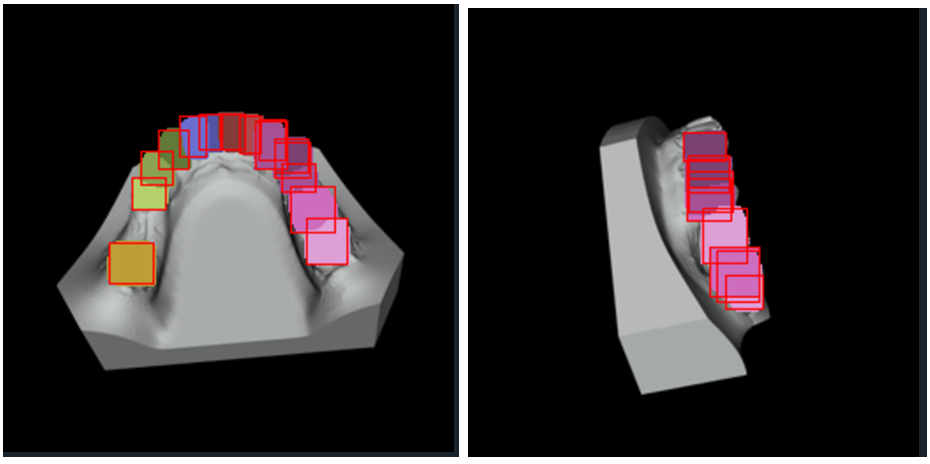
Obr. 6.6: Pohľad zhora a zboku z južnej strany.

V ľavej časti obrázkov môžeme vidieť, že základné pohľady zhora zvládajú pomerne dobre. Na Obr. č 6.6 je zreteľne vidieť, že sieť Mask-RCNN detekovala zub `G_left` na ľavej aj pravej časti, ktorý je znázornený ružovou farbou.Pohľady z boku z juhu Mask-RCNN sú úspešnejšie ako U-net, pretože Mask-RCNN klasifikovalo každý zub, ale nepriradilo im správne triedy. U-net označil oveľa menej zubov.

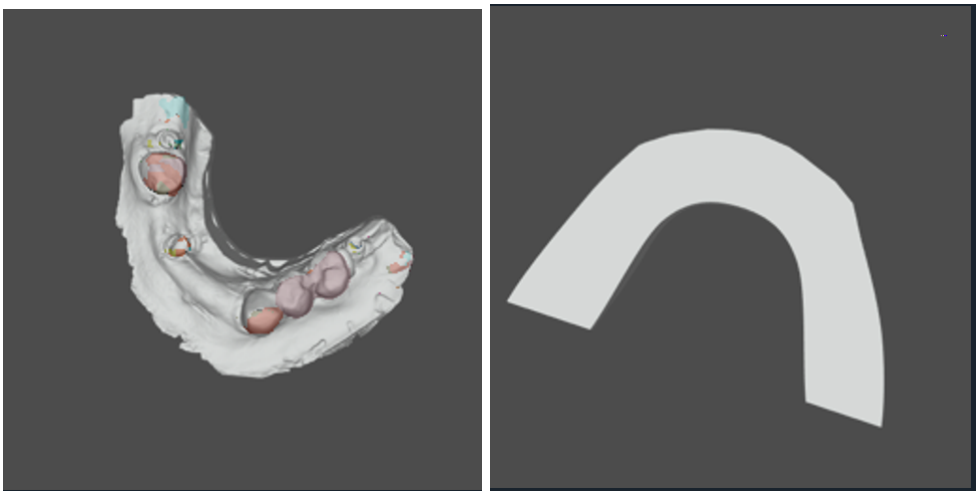
Zase ide o extrémny prípad, ktorý sa v datasete nenachádzal, ale môžeme vidieť, že Mask-RCNN a aj U-Net klasifikovalo aspoň väčšinu zubov z bočného pohľadu s nesprávnymi triedami.



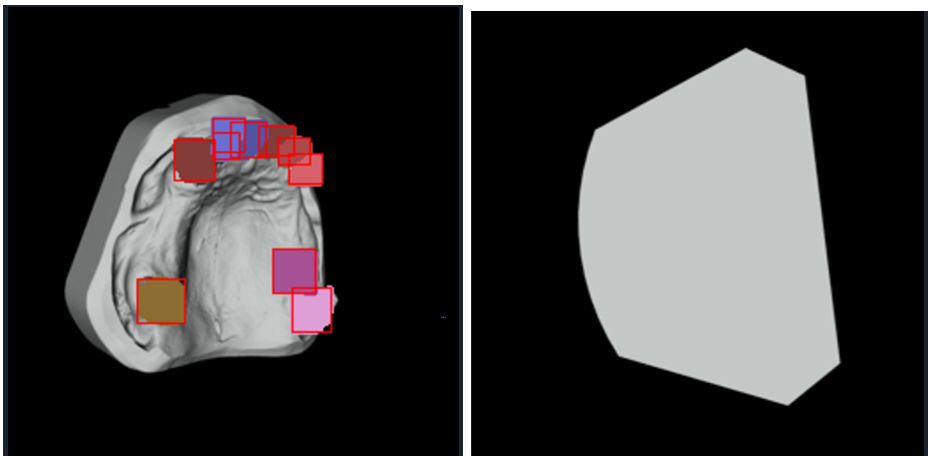
Obr. 6.7: Pohľad zhora a zboku zo západnej strany.



Obr. 6.8: Pohľad zhora a zboku zo západnej strany.

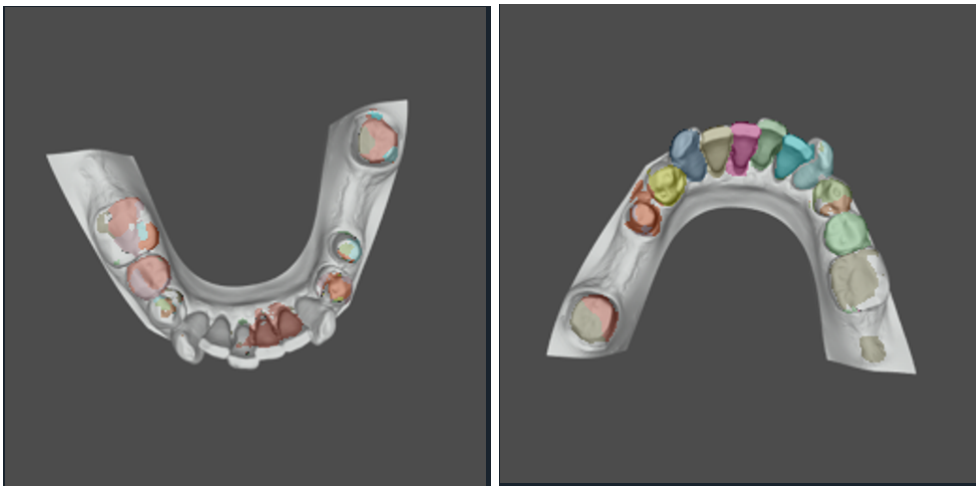


Obr. 6.9: Zlý model vs. model bez zubov.

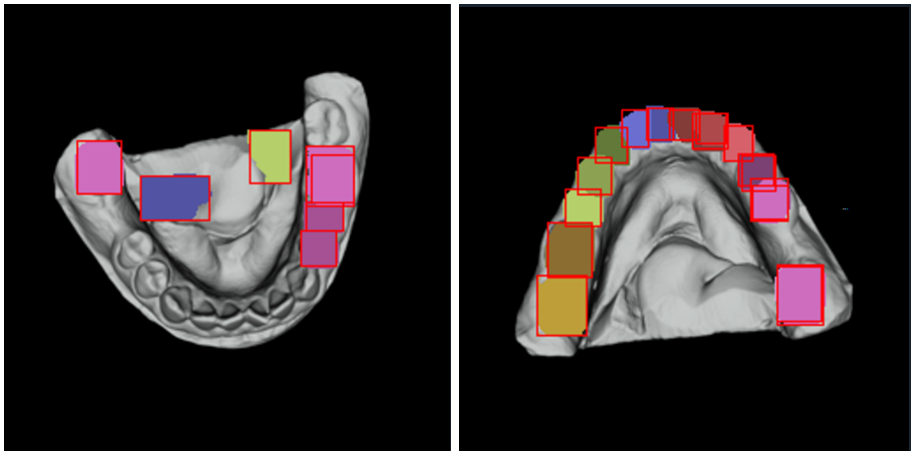


Obr. 6.10: Zlý model vs. model bez zubov.

Obrázky 6.9 a 6.10 nám poukazujú výsledky testu na segmentáciu a detekciu zubov v zlých modeloch, ako aj detekcie a segmentácie žiadneho zubu, čo sa obidvom sieťam úspešne podarilo. Klasifikovali každý zub opäť s nesprávnymi triedami.



Obr. 6.11: Zrkadlové pohľady.



Obr. 6.12: Zrkadlové pohľady.

Posledný test sa týkal obráteného pohľadu na zuby. Ako môžme vidieť, neurónové siete sú neúspešné oproti klasifikácií zo zrkadlových pohľadov.

### Výsledky

Dané poznatky z testov ukazujú, že metóda Multi-view bude potrebovať minimálne polovicu zrkadlových obrázkov. Tým vzniká predpoklad, že dané modely sa lepšie natrénujú na všeobecnú detekciu typu zubov.

# Kapitola 7

# Záver

Cieľom tejto bakalárskej práce bolo použiť a natrénovať 2D neurónové siete na 3D modely pomocou rôznych pohľadov na model. V priebehu práce sa vyskytol problém s neanotovaným datasetom a časovej neefektive anotovania 3D datasetu. Tento problém bol v bakalárskej práci vyriešený pomocou vlastného anotačného skriptu pre program Blender. Vďaka anotačnému skriptu bolo vyrenderovaných 157 GB vstupov a binárnych masiek pre zuby a landmarky, čo viedlo k efektívnemu trénovaniu modelov.

Výsledkom tejto práce sú aplikácie s modelmi sietí U-net, Yolo, a Mask-RCNN. Implementoval som aplikácie, ktoré umožňujú nahrať 3D .stl model zubov, rôzne tieto modely priblížiť a otočiť. V aplikácii som použil modely neurónových sietí na klasifikáciu pomocou objektovej detekcie a obrázkovej segmentáciu. Zistil som tak polohu zubov, ako aj ich landmarkov. V rámci testovania sa zistilo, že multi-view metóda funguje spoľahlivo, iba s textúrou modelov, na ktorých boli vybrané siete natrénované. Predikciu tiež ovplyvňuje priblíženie modela.

Z osobného hľadiska som sa naučil základy neurónových sietí, ako aj tvorbu veľkých datasetov potrebných pre trénovanie. Táto práca mi umožnila mať dobrý prehľad v hlbokom učení so základnými architektúrami, tiež aj s programovaním neurónových sietí vo frameworku `Pytorch`. Táto práca ukázala problematiku datasetov v medicínskom priemysle, ako aj problematiku s reálnymi dátami.

V budúcnosti sa dá pokračovať na tejto bakalárskej práci na základe vlastných textúrnych povrchov pre 3D modely, čím by sa zistilo aké povrchy sú najefektívnejšie na obrázkovú segmentáciu a objektovú detekciu pre viac-pohľadovú metódu.

# Literatúra


[1] *Max-pooling*. Dostupné z: https://computersciencewiki.org/index.php/Max-pooling_/_Pooling.

[2] Amidi, S. *Convolutional neural networks*. Dostupné z: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks.

[3] Bouafif, O., Khomutenko, B. a Daoudi, M. Hybrid Approach for 3D Head Reconstruction: Using Neural Networks and Visual Geometry. *CoRR*. 2021, abs/2104.13710. Dostupné z: https://arxiv.org/abs/2104.13710.

[4] Chen, T.-H. *What is "PADDINGïn convolutional neural network?* Machine Learning Notes, Nov 2017. Dostupné z: https://medium.com/machine-learning-algorithms/what-is-padding-in-convolutional-neural-network-c120077469cc.

[5] Ching, T., Himmelstein, D. S., Beaulieu Jones, B. K., Kalinin, A. A., Do, B. T. et al. Opportunities and obstacles for deep learning in biology and medicine. *Journal of the Royal Society Interface*. The Royal Society. 2018, zv. 15, č. 141. ISSN 17425689. Dostupné z: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC5938574/pdf/.

[6] Chollet, F. *Deep learning with Python*. 1. Shelter Island, NY: Manning, 2018. ISBN 978-1-61729-443-3.

[7] Dikici, E., Nguyen, X. V., Bigelow, M. a Prevedello, L. M. *Augmented Networks for Faster Brain Metastases Detection in T1-Weighted Contrast-Enhanced 3D MRI*. 2021.

[8] Fumo, D. *A Gentle Introduction To Neural Networks Series - Part 1*. Towards Data Science, Oct 2017. Dostupné z: https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc.

[9] Goodfellow, I. *Deep learning*. Cambridge, MA: MIT press, 2016. Adaptive computation and machine learning series. ISBN 978-0-262-03561-3.

[10] Hamdi, A., Aboeleneen, A. a Shaban, K. B. MARL: Multimodal Attentional Representation Learning for Disease Prediction. *CoRR*. 2021, abs/2105.00310. Dostupné z: https://arxiv.org/abs/2105.00310.

[11] He, K., Gkioxari, G., Dollár, P. a Girshick, R. B. Mask R-CNN. *CoRR*. 2017, abs/1703.06870. Dostupné z: http://arxiv.org/abs/1703.06870.

[12] He, K., Gkioxari, G., Dollár, P. a Girshick, R. *Mask R-CNN*. 2018.

[13] Lo, S. a Yin, Y. An Interaction-based Convolutional Neural Network (ICNN) Towards Better Understanding of COVID-19 X-ray Images. *CoRR*. 2021, abs/2106.06911. Dostupné z: https://arxiv.org/abs/2106.06911.

[14] Moons, B., Bankman, D. a Verhelst, M. *Embedded Deep Learning: Algorithms, Architectures and Circuits for Always-on Neural Network Processing*. Cham: Springer International Publishing, 2018. ISBN 9783319992228.

[15] Moons, B., Bankman, D. a Verhelst, M. *Embedded Deep Learning: Algorithms, Architectures and Circuits for Always-on Neural Network Processing*. Cham: Springer International Publishing, 2018. ISBN 9783319992228.

[16] Paulsen, R. R., Juhl, K. A., Haspang, T. M., Hansen, T., Ganz, M. et al. Multi-view Consensus CNN for 3D Facial Landmark Placement. *Lecture Notes in Computer Science*. Springer International Publishing. 2019, s. 706–719. ISSN 1611-3349. Dostupné z: http://dx.doi.org/10.1007/978-3-030-20887-5_44.

[17] Redmon, J., Divvala, S. K., Girshick, R. B. a Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. *CoRR*. 2015, abs/1506.02640. Dostupné z: http://arxiv.org/abs/1506.02640.

[18] Redmon, J. a Farhadi, A. YOLOv3: An Incremental Improvement. *CoRR*. 2018, abs/1804.02767. Dostupné z: http://arxiv.org/abs/1804.02767.

[19] Ren, S., He, K., Girshick, R. a Sun, J. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2016.

[20] Ronneberger, O., Fischer, P. a Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. *CoRR*. 2015, abs/1505.04597. Dostupné z: http://arxiv.org/abs/1505.04597.

[21] Shapiro Linda, G. *Computer vision*. Vyd. 1. New Jersey: Prentice-Hall, 2001. ISBN 0-13-030796-3.

[22] Tripathi, A., Rakkunedeth, A., Panicker, M. R., Zhang, J., Boora, N. et al. *Domain Specific Transporter Framework to Detect Fractures in Ultrasound*. 2021.

[23] Watson, S. *Teeth Names: Shape and Function of Four Types of Teeth*. Healthline Media, May 2018. Dostupné z: https://www.healthline.com/health/teeth-names#diagram.

[24] Zajac, J. C., Abbate, O., Oh, A. K., Mantilla Rivas, E., Aivaz, M. et al. Dental Topics for Plastic Surgeons, Part One: Normal Anatomy, Growth and Development. *Journal of Craniofacial Surgery*. 2020, zv. 31, č. 4. ISSN 1049-2275. Dostupné z: https://journals.lww.com/jcraniofacialsurgery/Fulltext/2020/06000/Dental_Topics_for_Plastic_Surgeons,_Part_One_.71.aspx.

[25] Zhang, X. *Simple Understanding of Mask RCNN*. Medium, Apr 2018. Dostupné z: https://alittlepain833.medium.com/simple-understanding-of-mask-rcnn-134b5b330e95.

## Zoznam príloh

Na pamäťovom médiu je kód so štruktúrou v priečinku pytorch_google_colab. Na spojazdenie celého projektu je napísaný postup v `readme.md`. Taktiež je súčasťou vzorový dataset zubov a landmark. Šablona na vytváranie datasetu je uložená v `template.blend`

# Príloha A

# Obsah zdrojového kódu

Tu bude vypísany zdrojový obsah kódu.

```
|-- bachelor_thesis\
||-- App\
|||-- src\
||||--landmark\
|||||--mask_rcnn\
||||||--__init__.py
||||||--main.py
|||||--unet\
||||||--__init__.py
||||||--main.py
||||--teeth\
|||||--mask_rcnn\
||||||--__init__.py
||||||--main.py
|||||--unet\
||||||--__init__.py
||||||--main.py
|||||--yolo\
||||||--__init__.py
||||||--main.py
||||--app.png
||||--uiOfApp.ui
||-- BlenderProjekt\
|||--__init__.py
|||--template.blend
||-- DatasetUtils\
|||--blender_image_to_mask_landmark.py
|||--blender_image_to_mask_teeth.py
|||--blender_script_landmark.py
|||--blender_script_teeth.py
|||--check_landmark_plot.py
|||--check_teeth_plot.py
|||--gauss.py
```

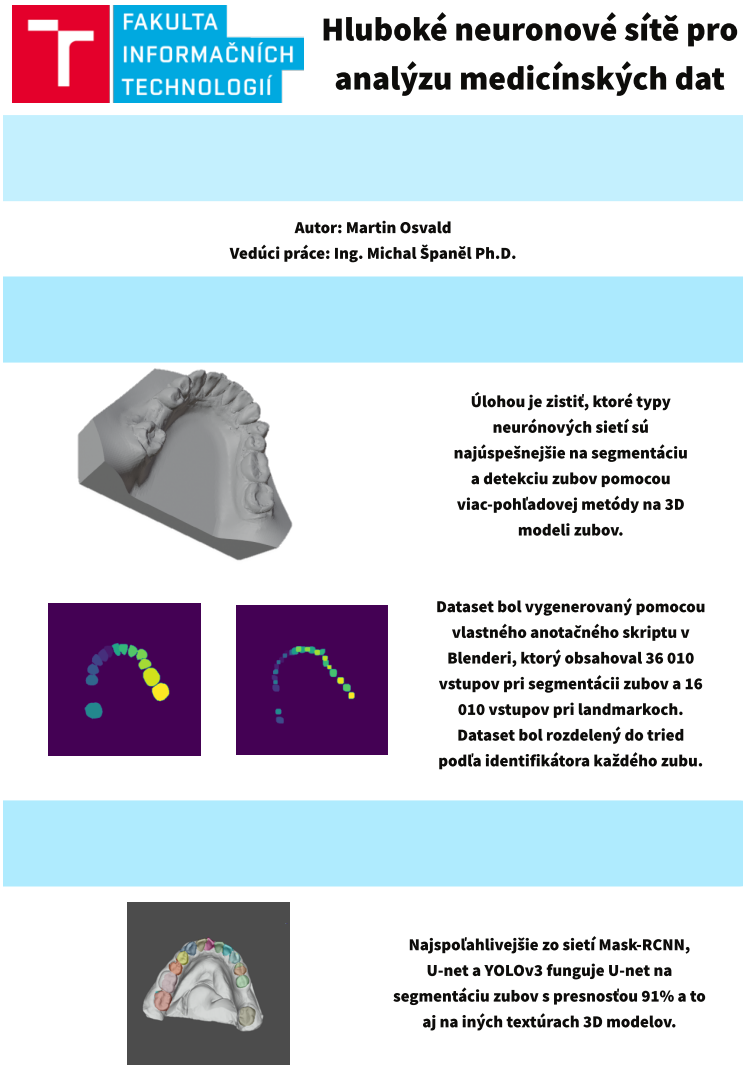
```
||-- images\
|||--landmark\
||||--mask-rcnn\
|||||--__init__.py
||||--yolo\
|||||--__init__.py
||||--unet\
|||||--__init__.py
|||--teeth
||||--mask-rcnn\
|||||--__init__.py
||||--yolo\
|||||--__init__.py
||||--unet\
|||||--__init__.py
||-- models_checkpoint
|||--landmark\
||||--mask-rcnn\
|||||--__init__.py
||||--yolo\
|||||--__init__.py
||||--unet\
|||||--__init__.py
|||--teeth
||||--mask-rcnn\
|||||--__init__.py
||||--yolo\
|||||--__init__.py
||||--unet\
|||||--_init__.py
||-- selected_checkpoints\
|||--landmark\
||||--mask-rcnn\
|||||--_init__.py
||||--yolo\
|||||--__init__.py
||||--unet\
|||||--_init__.py
|||--teeth
||||--mask-rcnn\
|||||--__init__.py
||||--yolo\
|||||--__init__.py
||||--unet\
|||||--__init__.py
||-- Tensorboard
|||--landmark\
||||--mask-rcnn\
```

```
|||||--__init__.py
||||--yolo\
|||||--_init__.py
||||--unet\
|||||--__init__.py
|||--teeth
||||--mask-rcnn\
|||||--_init__.py
||||--yolo\
|||||--__init__.py
||||--unet\
|||||--__init__.py
||-- Trainning
|||--lanmark\
||||--mask-rcnn\
|||||--_init__.py
|||||--coco_eval.py
|||||--coco_utils.py
|||||--engine.py
|||||--mask-rcnn-training.py
|||||--transforms.py
|||||--utils.py
||||--yolo\
|||||--config.py
|||||--dataset.py
|||||--loss.py
|||||--model.py
|||||--train.py
|||||--utils.py
||||--unet\
|||||-LovazsSoftmax\
||||||-pytorch\
|||||||--_init__.py
|||||||-lovasz_losses.py
|||||--unet\
||||||--_init__.py
||||||--unet\
|||||||--_init__.py
|||||||--model.py
|||||||--unet_base.py
||||||--utils\
|||||||--colors.py
||||||--dataset.py
||||||--inference.py
||||||--inference_color.py
||||||--losses.py
||||||--train.py
|||--teeth\
```

```
||||--mask-rcnn\
|||||--_init__.py
|||||--coco_eval.py
|||||--coco_utils.py
|||||--engine.py
|||||--mask-rcnn-training.py
|||||--transforms.py
|||||--utils.py
||||--yolo\
|||||--config.py
|||||--dataset.py
|||||--loss.py
|||||--model.py
|||||--train.py
|||||--utils.py
||||--unet\
|||||-LovazsSoftmax\
||||||-pytorch\
|||||||--_init__.py
|||||||-lovasz_losses.py
|||||--unet\
||||||--_init__.py
||||||--unet\
|||||||--_init__.py
|||||||--model.py
|||||||--unet_base.py
||||||--utils\
|||||||--colors.py
||||||--dataset.py
||||||--inference.py
||||||--inference_color.py
||||||--losses.py
||||||--train.py
|||||--_init__.py
||-- evniroment.yml
config_global.py
```

# Príloha B

# Plagát




Obr. B.1: Plagát ku bakalárskej práci.