Appendix A

Resumé

V tejto práci sa zameriavame na automatizovanú segmentáciu bedrových implantátov na rádiologických snímkach s dôrazom na kvantifikáciu a využitie predikčnej neistoty. Hlavným cieľom bolo nielen dosiahnuť vysokú presnosť segmentácie, ale zároveň poskytovať klinicky interpretovateľné mapy neistoty, ktoré upozornia na oblasti, kde model nie je istý svojím rozhodnutím. Praktická motivácia spočíva v zefektívnení klinických postupov: rýchla a spoľahlivá segmentácia šetrí čas rádiológov a ortopédov a znižuje riziko prehliadnutia drobných chýb či komplikovaných štruktúr. Aby sme tento cieľ dosiahli, položili sme si päť výskumných otázok:

- RQ1: Ako optimálne upraviť a natrénovať architektúru siete pre presnú segmentáciu implantátov?
- RQ2: Ktorá epistemická metrika neistoty (rozptyl, štandardná odchylka, entropia, IQR, 95%-rozsah) najlepšie koreluje s chybou segmentácie po pixeloch pri výpadku v Monte-Carlo?
- RQ3: Ako vplýva pridanie penalizačného člena založeného na entropii na presnosť segmentácie (Dice) a úroveň prediktívnej neistoty v porovnaní so

základným modelom bez tejto penalizácie?

- RQ4: Dokáže odstránenie najneistejších 10% pixelov (s najvyššou entropiou) zlepšiť priemerné Dice skóre a poskytnúť spoľahlivejší odhad oblasti implantátu?

A.0.1 Analýza

Prvou časťou analýzy je povaha segmentačnej úlohy. Na rozdiel od klasifikácie, kde model prideľuje jednej snímke jedinú triedu, segmentácia vyžaduje priradenie triedy (implantát vs. pozadie) ku každému pixelu. To znamená, že chyby na hraniciach implantátu alebo v dôsledku artefaktov môžu zásadne poškodiť klinickú použiteľnosť výsledku. V medicínskom prostredí je napriek vysokému Dice skóre často kritické poznať aj to, kde model „váha“ – pretože práve tieto miesta si vyžadujú zvýšenú pozornosť ošetrujúceho lekára. Z tohto dôvodu sme vo svojej práci pridali k U-Netu mechanizmus kvantifikácie epistemickej neistoty. Epistemická neistota odráža „vedomostné medzery“ modelu – teda oblasti, kde sa sieť ešte nenaučila dostatočne spoľahlivo rozlíšiť implantát od pozadia. Jedným zo spôsobov, ako ju aproximovať, je Monte Carlo dropout [22], ktorý v testovaciej fáze opakovane vykoná aktívny dropout a z výsledných pravdepodobností zostrojí mapu pixel-wise entropie. Táto entropia potom označuje pixely, na ktoré sa model neodváži rozhodnúť s istotou, a zároveň koreluje s miestami, kde segmentácia najčastejšie zlyháva. Z tohto dôvodu sme vo svojej práci pridali k U-Netu mechanizmus kvantifikácie epistemickej neistoty. Epistemická neistota odráža „vedomostné medzery“ modelu – teda oblasti, kde sa sieť ešte nenaučila dostatočne spoľahlivo rozlíšiť implantát od pozadia. Jedným zo spôsobov, ako ju aproximovať, je Monte Carlo dropout [22], ktorý v testovaciej fáze opakovane vykoná aktívny dropout a z výsledných pravdepodobností zostrojí mapu pixel-wise entropie. Táto entropia

potom označuje pixely, na ktoré sa model neodváži rozhodnúť s istotou, a zároveň koreluje s miestami, kde segmentácia najčastejšie zlyháva. Z tohto dôvodu sme vo svojej práci pridali k U-Netu mechanizmus kvantifikácie epistemickej neistoty. Epistemická neistota odráža „vedomostné medzery“ modelu – teda oblasti, kde sa sieť ešte nenaučila dostatočne spoľahlivo rozlíšiť implantát od pozadia. Jedným zo spôsobov, ako ju aproximovať, je Monte Carlo dropout [22], ktorý v testovaciej fáze opakovane vykoná aktívny dropout a z výsledných pravdepodobností zostrojí mapu pixel-wise entropie. Táto entropia potom označuje pixely, na ktoré sa model neodváži rozhodnúť s istotou, a zároveň koreluje s miestami, kde segmentácia najčastejšie zlyháva.

A.0.2 Metodológia

V experimentálnej časti sme najprv testovali rôzne architektúry segmentačných sietí: klasický U-Net, jeho rozšírené verzie U-Net++ a EfficientNet-U-Net, ako aj kombináciu CNN a Vision Transformer v TransUNet. Predspracovanie snímok zahŕňalo konzistentnú normalizáciu intenzít a zmenu rozlíšenia na 256×256 pixelov cez vybrané orezávanie a adaptívne škálovanie. Epistematická neistota bola kvantifikovaná pomocou Monte Carlo dropout – model vykonal viaceré (N = 30) priechody s aktivovaným dropoutom a z entropie zistených pravdepodobností sme zostrojili mapy neistoty. Na overenie agregácie predikcií sme porovnali strednú hodnotu (mean) a medián (median) pravdepodobností a ukázali sme, že mediánne agregovanie síce mierne znižuje odchýlku prístupu, no pri danom rozdelení dát neznamenalo významnú výhodu oproti priemeru. Augmentácia počas tréningu (otočenia a zrkadlenie) nepreukázala štatisticky významné zlepšenie Dice skóre. Ďalšie etapy zahŕňali analýzu, kde sme priebežne skúmali konvergenciu Dice skóre pri rastúcom počte MC vzoriek a spoľahlivosť predikčnej chyby voči entropii; následne sme odstránili 10% pixelov s najvyššou entropiou a ukázali významné zlepšenie

priemerného Dice skóre z 0.929 na 0.948. Nakoniec sme modifikovali trénovací proces pridaním penalizačného člena úmerného entropii do stratovej funkcie čo umožnilo redukovať predikčnú neistotu za cenu mierneho úbytku presnosti pri väčších penalizáciách; optimálna hodnota $\lambda = 0.05$ priniesla vyvážený výsledok Dice skóre s výrazne nižšou entropiou.

A.0.3 Výsledky a diskusia

Segmentačné sieťe v našich experimentoch dosahovali val-Dice medzi 0,79 (EffUNet/U-Net++) a 0,90 (baseline U-Net), pričom TransUNet zaostával kvôli obmedzenému množstvu tréningových dát a potrebe širších architektúr. Zavedenie epistematickej neistoty poskytlo cenný nástroj pre klinickú prax: model sám vyznačil oblasti potenciálne nespoľahlivej segmentácie, čo môže lekárom urýchliť prácu a znížiť riziko prehliadnutia komplikovaných štruktúr. Na to sme prišli aj vďaka experimentu s odstránením 10% najneistejších pixelov. Na začiatku sme však porovnali jednotlivé metriky, ktoré by mali byť použité pri získavaní hodnoty epistematickej neistoty. Potvrdilo sa, že entropia je spoľahlivejšia metrika pre vizuálnu a kvantitatívnu kontrolu. Taký výsledok sme aj pôvodne očakávali, keďže ide o najbežnejšie používanú metriku pri Monte Carlo dropout. Na záver sme natrénovali nový posledný model, ktorý zahŕňal penalizačný člen v stratovej funkcií. Ten pomohol tlmiť chybné predikcie v problematických oblastiach bez výrazného zhoršenia celkovej presnosti pri primeranej voľbe váhy λ.

Naše výsledky dopĺňajú súčasný stav výskumu v oblasti 'kvantifikácie neistoty v umelej inteligencií' pre medicínske snímky a potvrdzujú, že integrácia neistoty do segmentačného pipeline je realizovateľná, užitočná a perspektívna. Limitujúcim faktorom zostáva veľkosť a rozmanitosť datasetu, preto ďalší vývoj by mal smerovať k rozšíreniu tejto množiny, obohateniu o rozličné typy implantátov a k

zefektívneniu architektúr pre vyššiu generalizáciu.

Appendix B

Technical Documentation

This appendix describes the content of the code and its individual files used in this work.

`requirements.txt`

This file contains all the libraries and versions needed to run our segmentation-uncertainty pipeline. For the project to function properly, they need to be installed all at once using command: pip install -r requirements.txt. After running the command above, you have all the necessary modules set up, so that the entire pipeline — from image loading to training, inference with uncertainty quantification to visualization of the results — works without any further problems.

`segmentation-models.py`

This module was used to compare four types of architectures and evaluate which one we will use for segmentation tasks. Specifically, we trained 5 epochs of the classic UNet, UNet++, EffNet and TransNet and compared their Dice score and loss function.

The python file, that uses PyTorch contains implementation of segmentation models, we used in comparison., auxiliary classes for training control and calculation of loss functions. Specifically, it contains:

- DoubleConv – the basic building block of U-Net
- UNet – standard convolutional U-Net for segmentation
- TransUnet - TransUNet hybrid architecture, which combines a classic U-Net encoder-decoder with an intermediate Vision-Transformer layer
- _CBR - Conv–BatchNorm–ReLU, a basic four-line convolution block followed by normalization and activation. It serves as a building block of the EffNet decoder.
- EfficientNetUNet - Classic U-Net, where the encoder replaces the pre-trained EfficientNet-B0.
- EarlyStopping - a class that terminates the training, when it stops getting better
- DiceLoss – calculates the Dice loss between the prediction and the target mask.
- CombinedLoss - Combines DiceLoss and BCEWithLogitsLoss according to given weights to simultaneously train shape accuracy (Dice) and classical probabilistic consistency (BCE).
- SegmentationDataset - Basic PyTorch Dataset for loading samples from directories.
- and Joint classes: Extension of SegmentationDataset by simple augmentation by rotations and horizontal flips.

After installing the required dependencies listed in the requirements.txt file and loading the dataset, the code can be executed without any further settings.

`masks_generating.py`

This script is used to train and then automatically generate binary implant masks for a new batch of X-ray images. It uses a modified U-Net architecture with residual blocks (inspired by ResNet) in Keras/TensorFlow, and after training the model, it applies the prediction to all images in the specified folder and saves the resulting masks to the output directory.

As for the helper functions, we use residual_block(x, filters) - It creates two 3x3 convolutional layers with squeeze-excite mechanism, which are connected to the skip-connection using input and output mapping. It is used to build the deeper parts of the U-Net. And improved_unet(input_shape) - it builds the entire U-Net model with four levels of residual blocks in both the encoder and decoder. At the end, it returns a Keras Model(inputs, outputs), ready for compilation.

Again after installing the required dependencies listed in the requirements.txt file and loading the dataset, the code can be executed without any further settings.

`segmentation-hyperparameters.py`

This script is used to automatically find optimal hyperparameters for a selected segmentation architecture - UNet - using the Optuna library. It uses the model definition, data loader, and loss function from segmentatin-models.py and, based on a predefined spatial distribution of candidate values, explores combinations that achieve the best validation Dice score.

At the beginning of the script you will find the definition of the range of hyperparameters that you want to explore with tuning. Typically this is the objective(trial)

function. There we set the ranges for lr, batch size, dropout, weight of dice loss in loss function. After running the script, the Optuna study returns the results, which we then used as settings in the segmentation model.

Again, the code is executable after loading dependencies from requirements.txt and loading data.

`segmentation.py`

This script provides a complete end-to-end pipeline for training and evaluating a segmentation model (U-Net) with and without training augmentation. We count here with a number of pre-prepared classes and functions from segmentation-models.py, set up data loaders, perform dataset partitioning, train both versions of the network, save the best states of the model and then evaluate them on the test set.

At the beginning, we set SAVE_DIR - the directory where the continuous results of our segmentations will be saved and the paths to the data: DATA_DIR (x-ray images) and MASK_DIR (which must have already been generated)

After installing the dependencies from requirements.txt and preparing the dataset, just run the code and the entire training + validation + testing will take place automatically without any further intervention.

`unceratainty-epistematic.ipynb`

This Jupyter Notebook is built as a workbook, in which all tasks related to the estimation and use of epistemic uncertainty in hip implant segmentation are implemented.

At the beginning there is an import block - the PyTorch, torchvision, NumPy, Matplotlib, OpenCV libraries and other auxiliary modules are loaded. After

this, our model classes and functions are defined: implementation of model architectures (primarily UNetWithDropout), loss functions (DiceLoss and CombinedLoss), classes for data loading (SegmentationDataset), utilities for early stopping and especially the $\mathrm{mc}_{d}ropout_{p}redictfunction, whichprovidesrepeatedMC - dropouttoobtainpixel-wiseprobabilitiesanduncertaintymetrics(variance, standarddeviation, e$

Other cells are organized according to individual research tasks:

- Task 1: Demonstration of multiple uncertainty metrics on a single test. One example is selected from the test loader, MC-dropout is performed, the original, ground-truth mask, binarized prediction and all prepared uncertainty maps are graphically displayed.

- Task 2: The effect of the number of MC samples N = 0,10,20,30,40,50 on the average Dice score and prediction stability. First, the Dice convergence curve with a narrowed Y-axis is calculated and plotted, then the entropy for the selected image and different values of N is visualized.

- Task 3: Verification of the histogram overlay of uncertainty for correct and incorrect pixels. The distribution of entropy (and other metrics) for both groups is shown in distinct plots after the algorithm eliminates the pixels where the model correctly predicted the implant and the pixels where it was incorrect.

- Task 4: Construction of a slice-wise 2D heatmap: pixels divided by epistemic entropy values into bins and the calculated error rate in each bin-slice pair. The result is a density matrix with an average error rate vs. uncertainty curve.

- Task 5: Scatter plot of structural uncertainty (SV) versus 1 – Dice of each structure (volume). A scatter plot with a regression line shows whether high

error rate correlates with higher uncertainty.

- Task 6: Comparison of mean vs. median aggregation in MC-dropout. Entropy maps and a bar graph of average Dice scores over multiple iterations are plotted for both methods, supplemented by a paired t-test.
- Task 7: Evaluation of the removal of the top 10% of the most uncertain pixels. Dice is calculated on the full image and after excluding highly uncertain pixels, a graphical demo of this mask and a statistical comparison are displayed.
- Task 8: Testing the impact of training data augmentation on epistemic uncertainty and Dice. Two models – with and without horizontal mirroring and rotations – are compared, and their results (Dice, entropy) are evaluated by a paired test and visualization of predictions and entropy differences.
- Task 9: Qualitative 4-panel paper-style figure: input RTG with baseline and GT contours, entropy map, a priori MC-dropout prediction with contours. This overall layout is repeated for several random examples.
- Task 10: Modified training with entropy penalty. In the training loop, K dropout approaches are averaged, $\lambda \times$ mean entropy is added to the loss function, and the evolution of tren-loss, val-loss, and val-Dice is followed over epochs for different λ. Finally, it is evaluated which weight provided the best compromise.

Appendix C

Contents of Included Media

Registrationnumberofthethesisintheinformationsystem: FIIT-182905-110796

Content of the digital part of the work (ZIP):

File	Content
requirements.txt	Python dependencies list
segmentation-models.py	Segmentation model implementations file
masks_generating.py	Mask generating training script
segmentation-hyperparameters.py	Hyperparameter tuning with Optuna
segmentation.py	Full training and evaluation pipeline
unceratainty-epistematic.ipynb	Uncertainty analysis notebook
DP_Hlavinova_Petra.pdf	Final document
DP_Hlavinova_Petra_appendices.pdf	Appendices

Uploaded folder name: DP_PetraHlavinova.zip
The folder can also be found on github in the repository:
https://github.com/petrahlavinova/DP_Final.git