

UCLA

UCLA Electronic Theses and Dissertations

Title

Cognitive AI to Improve Deep Learning Segmentation and Quantitative Analysis of Renal CT

Permalink

<https://escholarship.org/uc/item/8wv9z0d3>

ISBN

9798186340850

Author

Melendez-Corres, Gabriel

Publication Date

2026-08-11

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Cognitive AI to Improve Deep Learning Segmentation and Quantitative Analysis of Renal
CT

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Physics & Biology in Medicine

by

Gabriel Melendez Corres

ABSTRACT OF THE DISSERTATION

Cognitive AI to Improve Deep Learning Segmentation and Quantitative Analysis of Renal
CT

by

Gabriel Melendez Corres

Doctor of Philosophy in Physics & Biology in Medicine

University of California, Los Angeles, 2026

Matthew Sherman Brown, Chair

Segmentation in medical imaging is an important step to identify and describe organs and diseases such as tumors. Segmentation allows for the quantitative analysis of these anatomical structures to develop biomarkers that can be utilized in the diagnosis, progression, and prognosis of diseases and conditions. In volumetric imaging data, such as those acquired through the use of Computed Tomography (CT) scanners, the process of segmentation involves the manual delineation of voxels composing the structures of interest through the series of images that form the volume. This manual process is time-consuming, limiting high-throughput production of segmentations for clinical usage. To solve this issue, Deep Learning (DL) segmentation algorithms have been developed, leveraging Convolutional Neural Networks (CNN) to learn voxel-level features for the automated segmentation of medical images. While DL tools have seen advancements and success in recent years, clinical usage of these tools has been limited. DL tools like CNNs operate as a "black box", containing hidden layers where learned features are not readily visible to the end user. The CNN can provide an output, but not an explanation for said output. Additionally, these models are prone to mistakes that defy basic anatomical knowledge and reasoning. These issues limit models' trustworthiness which in turn limit their implementation into clinical settings.

In this dissertation we propose the use of Cognitive AI tools in the form of Neuro-

Symbolic (NeSy) AI to enable anatomical reasoning and ensure anatomically consistent outputs. NeSy methods allow for the combination of DL, neural network methods which are capable of learning and updating from data, with symbolic methods that allow for the establishment of logic-based rules and relationships to provide anatomical constraints and model explainability. We developed these tools in the SimpleMind (SM) software environment, an agent-based framework where we developed agents to enable an end-to-end segmentation and quantitative analysis pipeline within an anatomical Knowledge Graph (KG). This includes image processing tools, reasoning agents, feature extraction, predictive modelling and hyperparameter optimization. Developed tools are general and task-agnostic; however, in this dissertation we present the use case of kidney and lesion segmentation in renal CT, nephrometry feature extraction and feature-based predictive modelling and optimization.

To demonstrate the effectiveness of NeSy reasoning methods, a CNN model, nnUNet, trained for kidney segmentation on the KiTS23 dataset was integrated into the SM reasoning framework. Kidney segmentation candidates were evaluated and selected using SM reasoning methods Fuzzy Membership Reasoning (FMR), Adaptive Reasoning (AR), and Largest Connected Component Analysis Per Side (LCCAPS), and performance was compared to the commonly used Largest Connected Component Analysis (LCCA). Performance was assessed via False Positive Rate (FPR) and False Negative Rate (FNR) across 5 validation folds. Candidate-level acceptance and rejection metrics were also introduced to evaluate selective performance. Robustness was further examined under added noise and added synthetic candidates. Reasoning-based approaches (FMR, AR, and LCCAPS) reduced false positives and anatomically invalid outputs more effectively than LCCA, while preserving acceptance of valid segmentation candidates. Results persisted under noise and added synthetic candidates, avoiding acceptance of implausible results. Incorporating reasoning into DL segmentation supports selective acceptance of anatomically valid outputs and rejection of invalid ones, increasing reliability and autonomy in medical imaging workflows. This selective segmentation framework provides a practical path toward trusted, semi-autonomous clinical deployment.

We extended the segmentation KG to automatically extract features used in nephrometry scoring for the assessment of renal lesion complexity. On a test set of 61 scans, we compared the automated features to human features obtained through manual visual assessments. Agreement among humans and automated methods was within the level of agreement reported in the literature while a variability analysis demonstrated automated features are within the human variability, with lesion radius and endophycity being significantly different than random nephrometry scoring.

Lastly, within the SM environment, we incorporated a Machine Learning (ML)-based predictive modelling agent encompassing feature selection, pre-processing, resampling, model selection and training. The agent was utilized to construct prediction pipelines for renal lesion malignancy, necrosis, surgical procedure, and subtype using the extracted nephrometry and radiomic features as inputs, with clinical outcomes data from KiTS23. Optimization of the pipeline was done using a Genetic Algorithm (GA) within SM’s Knowledge Graph Learning and Optimization (KNoLO) module to enable end-to-end optimization of the predictive modelling pipeline. The GA demonstrated optimization of the custom fitness function for each clinical outcome. However, generalization outside of training and validation remains to be seen. Additionally, we provide a diversity analysis which evaluates the search space of the GA and limitations to be addressed for future improvements.

The dissertation of Gabriel Melendez Corres is approved.

Michael F. McNitt-Gray

Grace Hyun Kim

John M. Hoffman

William Hsu

Matthew S. Brown, Committee Chair

University of California, Los Angeles

2026

To my family and friends

Contents

Abstract	ii
List of Figures	xi
List of Tables	xvi
List of Listings	xvii
Acknowledgements	xviii
Vita	xxi
1 Introduction	1
1.1 Specific Aims	1
1.1.1 Specific Aim 1: Integration of kidney and lesion segmentation deep learning models supported by cognitive AI agents.	1
1.1.2 Specific Aim 2: Development of machine reasoning and quantitation agents for automated nephrometry scoring on CT.	2
1.1.3 Specific Aim 3: Development of a general-purpose predictive modelling agent, applied to renal lesion pathology and surgical outcomes, and supported by automatic parameter tuning.	3
1.2 Deep Learning for Biomedical Image Segmentation	3
1.2.1 Image Segmentation/Annotation	3
1.2.2 An Introduction to Deep Learning	5
1.2.3 Convolutional Neural Networks	8
1.2.4 Applications of Deep Learning in Imaging	10
1.3 Cognitive AI using Neuro-Symbolic methods	15
1.3.1 Fuzzy Logic and Membership Functions	17
1.3.2 Recent Neuro-Symbolic (NeSy) AI Works in Medical Imaging	18
1.4 Quantitative Image Analysis & Imaging Biomarkers in CT	22
1.5 Contributions and Dissertation Organization	26
2 The SimpleMind Infrastructure for Cognitive AI in Medical Imaging	29
2.1 Introduction to The SimpleMind Software Environment	29
2.1.1 Agents in SimpleMind	30
2.2 Biomedical Image Segmentation using nnUNet	31
2.2.1 Integration of nnUNet in SimpleMind	33

2.3	General Computer Vision and Utility Agents in SimpleMind	33
2.3.1	Image Processing Agents	34
2.3.2	Mask Processing Agents	36
2.4	Integration of reasoning in SimpleMind	40
2.4.1	Mask Logical Operations	40
2.4.2	Fuzzy Membership Reasoning	41
2.4.3	Adaptive Reasoning	43
2.4.4	Candidate Selection	45
2.4.5	Summary	46
3	Kidney Segmentation with Reasoning for Reducing CNN Segmentation Errors	47
3.1	Introduction	47
3.1.1	Selective Segmentation with Rejection Option	48
3.2	Methods	50
3.2.1	Dataset	50
3.2.2	Kidney Segmentation Using SimpleMind	51
3.2.3	Analysis	54
3.3	Results	57
3.3.1	Noise experiment: Kidney Segmentation Detection Analysis	57
3.3.2	Noise experiment: Kidney Candidates Acceptance & Rejection Analysis	60
3.3.3	Synthetic Candidate Experiment: Kidney Candidates Acceptance & Rejection Analysis	62
3.4	Discussion	67
3.5	Conclusion	70
4	Deployment of the SimpleMind Kidney Segmentation Model for Research and Clinical Workflows	71
4.1	Introduction: Considerations for Model Deployment	71
4.2	Methods	74
4.2.1	Infrastructure	74
4.2.2	Research Deployment	75
4.2.3	Clinical Deployment	78
4.2.4	Clinical Deployment Evaluation	78
4.3	Results	79
4.3.1	Research Deployment	79
4.3.2	Clinical Deployment	79
4.4	Discussion	81
4.5	Conclusion	82
5	Automated Nephrometry Scoring Using SimpleMind	83
5.1	Introduction	83
5.1.1	Nephron Sparing Procedures	83
5.1.2	Nephrometry Scoring	84
5.1.3	Reproducibility of RENAL Nephrometry Scoring	86

5.1.4	Automated Nephrometry	88
5.2	Methods	89
5.2.1	Dataset	89
5.2.2	Kidney and Lesion Segmentation	90
5.2.3	Biological Landmark Segmentation	90
5.2.4	Lesion Diameter, Nearness, Endophycity, and Polar Line Location . .	92
5.2.5	Experimental Testing	97
5.2.6	Statistical Analysis	97
5.3	Results	98
5.3.1	Automated Segmentation Accuracy	98
5.3.2	Nephrometry Component Agreement and Equivalence Analysis of AU- TOSEG_CAD and CAD	100
5.3.3	Agreement of Automated Methods and Human Raters	101
5.3.4	Bland-Altman Analysis of Automated Methods vs Human Median . .	106
5.3.5	Within Variability Analysis	107
5.4	Discussion	111
5.5	Conclusion	115
6	The SimpleMind Infrastructure for Predictive Modelling and Hyperpa- rameter Tuning	117
6.1	Introduction	117
6.2	Using Predictive Models with SimpleMind	119
6.2.1	Agent Configuration	119
6.2.2	Predictive Modelling Agent Operation	120
6.3	Hyperparameter Optimization Using a Genetic Algorithm	123
6.3.1	Genetic Algorithms	123
6.3.2	The SimpleMind GA Implementation	124
6.3.3	Non-Functional Chromosomes	127
6.3.4	Phenotypic Clones and Junk DNA	130
6.4	Summary	133
7	Kidney Lesion Pathology & Outcome Prediction Models Using Nephrom- etry and Radiomics	134
7.1	Introduction	134
7.2	Methods	135
7.2.1	Dataset Preparation	135
7.2.2	Nephrometry Feature Extraction	137
7.2.3	Radiomic Feature Extraction	138
7.2.4	Predictive Modeling	138
7.2.5	Genetic Algorithm Optimization	140
7.2.6	Statistical Analysis	141
7.3	Results	142
7.3.1	Malignancy Prediction	142
7.3.2	Surgical Procedure Prediction	143
7.3.3	Tumor Necrosis Prediction	144

7.3.4	Tumor Histologic Subtype	145
7.3.5	Population Diversity Analysis	146
7.4	Discussion	148
7.5	Conclusion	152
Appendices		154
A	The SimpleMind Knowledge Graphs for Kidney Segmentation	155
B	Methodology for rotation of kidney polar lines from Chapter 5	159
B.0.1	Image Geometry and Coordinate Systems	159
B.0.2	Organ Long Axis via Principal Component Analysis (PCA)	160
B.0.3	Coronal-Plane Tilt Angle Relative to Midline	160
B.0.4	Rotate the Polar Lines About the Coronal Axis	161
C	Chapter 7 Uninstantiated YAML configurations for KNoLO	163

List of Figures

1.1	Illustration of image segmentation. Structures are delineated within the original image and represented in a binary mask, where foreground pixels or voxels belonging to the segmented objects are assigned a value of 1 and background pixels or voxels are assigned 0.	5
1.2	Example of image and annotation/segmentation in renal CT from the publicly available KiTS23 dataset [44].	5
1.3	Artificial neuron example.	6
1.4	Example of activation functions used in Artificial Neural Networks (ANN)s. .	7
1.5	Basic fully connected artificial neural network with one hidden layer.	8
1.6	Illustration of a convolution operation. A filter (kernel) is applied to a local receptive field in the input image through element-wise multiplication ($\odot$), which are then summed ($\sum$) and then passed through an activation function (ϕ) to produce feature a_{ij} . Sliding the same filter across the image generates a complete feature map.	9
1.7	General schematic of a U-Net architecture. Each encoder block contains two 3×3 convolutions followed by ReLU activations and 2×2 max pooling. The decoder uses 2×2 transposed convolutions for up-sampling, concatenates encoder features through skip connections, and refines them with two 3×3 convolutions. A final 1×1 convolution with sigmoid activation produces a single-channel binary segmentation mask.	13
1.8	Venn-diagram interpretation of the Dice coefficient. The Dice score measures overlap between the predicted region P and the ground-truth region G . The overlap corresponds to true positives (TP), while the non-overlapping regions correspond to false negatives (FN) and false positives (FP).	14
1.9	Knowledge-graph style representation of First-Order Logic (FOL) using an anatomical example. Constants are represented as entities, while predicates encode labeled relations between them. The resulting grounded predicates can be interpreted as factual statements under a given interpretation.	16
1.10	Representation of <i>LeftOf</i> First-order Logic (FOL) relationship as the function $LeftOf(D_k, s)$. Distance of a kidney (D_k) relative to the spine midline (s). Truth values are obtained at decision boundary $D_k = 0$ when $D_k < 0$	17
1.11	Examples of common fuzzy membership functions: (a) triangular, (b) trapezoidal, (c) Gaussian, (d) generalized bell, and (e) sigmoidal. Each function is bounded within the interval $[0, 1]$	19
1.12	Illustration of a single feature (x_1) with clear separation between two classes. In this simplified setting, a single threshold can provide effective discrimination. 24	

1.13	Illustrative decision boundaries for several machine learning models in a two-feature (x_1, x_2) example. Logistic regression produces a linear decision boundary, decision trees partition the feature space through axis-aligned splits, KNN forms local neighborhood-based decision regions, and SVM seeks a separating hyperplane with maximum margin defined by support vectors. These examples illustrate how ML methods can model multi-dimensional and non-linear relationships to discriminate between classes.	25
2.1	Representation of promise-based execution in the SimpleMind environment. Agents communicate indirectly through the BlackBoard, which functions as a shared communication space and short-term memory while active. Each agent reads the promise required for its execution, performs its assigned task, and then updates the BlackBoard with a newly generated output. These updated promises can then be used by downstream agents, enabling the staged execution of further agents.	31
2.2	(A) Example of the HU thresholding agent being used to segment the bones in a CT scan. (B) 3D rendering of the bone segmentation in (A).	34
2.3	(A) Input image with bounding box for cropping. (B) Cropped image.	35
2.4	Illustration of basic morphological operations on a binary mask. Erosion removes foreground pixels around object boundaries, shrinking the mask. Dilation expands the foreground region by adding pixels around the boundary. Opening, defined as erosion followed by dilation, smooths boundaries and removes small foreground specks. Closing, defined as dilation followed by erosion, fills small holes and gaps within the mask. All operations are performed with respect to a structuring element, shown here as a 3×3 cross-shaped neighborhood.	37
2.5	Spatial offset output subregion bounding box for defining the spine region from the abdomen centroid.	38
2.6	Illustration of the convex hull operation. Given a set of boundary points, the convex hull forms the smallest convex region that fully encloses all points. . .	39
2.7	(A) Segmentation output from a DL model. (B) Identification of candidates using the connected components agent with removal of components under a voxel threshold.	39
2.8	Venn-style illustration of logical mask operations between two input masks, A and B . Shaded regions represent voxels or pixels evaluated as <i>TRUE</i> in the output mask for each operation: logical <i>and</i> ($A \wedge B$), logical <i>or</i> ($A \vee B$), logical difference ($A \wedge \neg B$), and exclusive <i>or</i> ($A \oplus B$).	41
2.9	Examples of piecewise linear fuzzy membership functions used to compute feature-specific sub confidences $conf_{ij}(x_{ij})$ for candidate j . Left: a two-threshold confidence function that increases from 0 to 1 between x_0 and x_1 . Right: a four-threshold trapezoidal confidence function that increases from 0 to 1 between x_0 and x_1 , remains at 1 between x_1 and x_2 , and decreases back to 0 between x_2 and x_3	42

2.10	Adaptive reasoning sub-confidence function for feature i , shown as a normalized Gaussian-shaped confidence curve centered at the learned mean $\bar{x}_i$. The spread of the curve is governed by the standard deviation σ_i , with larger σ_i producing a broader confidence region and smaller σ_i producing a narrower peak. . . .	44
3.1	Image examples of DL segmentation errors highlighted in red circles. (a) An extraneous component segmented outside of the kidneys. (b) Inaccurate segmentation of a kidney. Visualization done using Slicer [28].	49
3.2	Example image of quality degradation from added noise ($\sigma = 0.04$). Visualization done using Slicer [28].	50
3.3	Examples of kidney masks with added synthetic candidates. Original kidney masks are shown in purple while synthetic candidates are shown in green. From left to right, top to bottom: Original kidney masks, 1 synthetic candidate added, 2 synthetic candidates added and 3 synthetic candidates added. Visualization performed using Slicer [28].	51
3.4	Illustration of the distinction between false positives from Convolutional Neural Network (CNN) candidate generation and false acceptances during candidate selection. The CNN output mask defines the candidate sample set, producing true positives, false positives, and false negatives, but no true negative candidates. After candidate selection, negative candidates may be accepted or rejected, allowing false acceptance rate to be computed as $FAR = FA/(FA+TR)$	56
3.5	Bar charts of average kidney detection performance comparison across noise levels on validation data.	58
3.6	Bar charts of kidney detection performance comparison across noise levels on test data.	59
3.7	Bar charts of average candidate analysis performance comparison across noise levels on validation data.	62
3.8	Bar charts of candidate analysis performance comparison across noise levels on test data.	63
3.9	Bar charts of average candidate analysis performance comparison across number of added synthetic candidates on validation data.	65
3.10	Bar charts of average candidate analysis performance comparison across number of added synthetic candidates on test data.	66
3.11	Example image demonstrating the AR method removing erroneous candidates generated by the CNN. Red arrows highlight the two types of errors mentioned in Section 3.1 and Figure 3.1: Inaccurate segmentation and an extraneous component.	68
4.1	Kidney Segmentation Directed Acyclic Graph (DAG) for research deployment	77
4.2	Boxplots comparing annotation times for manual vs AI-assisted.	80
4.3	Boxplots of percent change in volume measurement after editing for right and left kidneys.	81
5.1	Example image illustrating the kidney polar lines	85
5.2	Identification of the affected kidney via contact with the lesion (red).	91

5.3	Body fat segmentation using HU thresholding from -195 to -45 HU.	92
5.4	Extraction of the convex hull of the affected kidney and subsequent erosion.	93
5.5	Segmentation of the renal sinus fat using a boolean logical <i>AND</i> operation between the body fat and the eroded convex hull of the affected kidney.	94
5.6	Final segmentation representing the Urinary Collection System (UCS) of the affected kidney using the convex hull of the renal sinus fat.	95
5.7	Initial polar line segmentation using the UCS boundaries in z.	96
5.8	Final polar line segmentation rotated to be oriented with kidney axis.	96
5.9	Boxplots of kidney and lesion Dice Similarity Coefficient (DSC).	99
5.10	Scatter plots of RENAL components comparison of AUTOSEG_CAD vs CAD with identity line (dashed) and best fit line (blue solid line). From left to right, top to bottom: Diameter, Endophycity, Polar line Overlap and Nearness.	101
5.11	Bland-Altman analysis of RENAL components comparison of AUTOSEG_CAD vs CAD. From left to right, top to bottom: Diameter, Endophycity, Polar line Overlap and Nearness.	102
5.12	Example case with zero difference in total Radius, Exophytic, Nearness, Anterior/posterior, and Location relative to the polar lines (RENAL) score compared to human median. RENAL total = 9.	103
5.13	Example case with largest negative difference (-3) in total RENAL score compared to human median. CAD RENAL total = 6, human median RENAL total = 9.	104
5.14	Total RENAL score and RENAL component heatmaps of quadratic weighted kappa agreement across all pairs of raters.	105
5.15	Pairwise Quadratic weighted Kappa agreement analysis stratified by RENAL_R.	105
5.16	Bland-Altman analysis of RENAL score and RENAL components comparison between CAD and human median. From top to bottom, left to right: Total score, RENAL R, RENAL E, RENAL N, and RENAL L.	107
5.17	Bland-Altman analysis of RENAL score and RENAL components comparison between AUTOSEG_CAD and human median. From top to bottom, left to right: Total score, RENAL R, RENAL E, RENAL N, and RENAL L.	108
5.18	Leave-one-out within-variance coverage analysis. From top to bottom: CAD, AUTOSEG_CAD and RND_CAD.	110
5.19	Within-variance coverage analysis for CAD methods.	110
6.1	JSON configuration examples.	121
6.2	Example bit-string chromosome encoding model selection and penalty representing the <i>apt_lists</i> from Listing 6.3.	126
6.3	Representation of the SimpleMind Genetic Algorithm cycle showing fitness evaluation, tournament selection, shuffle, mutation, crossover, and repopulation.	128
7.1	Class distribution of selected KiTS23 clinical outcomes and diagnoses.	136
7.2	GA evolution for malignancy prediction chromosomes.	143
7.3	ROC curve analysis for malignancy prediction. Baseline, Random Chromosome, and Best Chromosome.	144
7.4	GA evolution for surgical procedure prediction chromosomes.	145

7.5	ROC curve analysis for surgical procedure prediction. Baseline, Random Chromosome, Best Chromosome.	146
7.6	GA evolution for tumor necrosis prediction chromosomes.	147
7.7	ROC curve analysis for tumor necrosis prediction. Baseline, Random Chromosome, Best Chromosome.	148
7.8	GA evolution for tumor subtype prediction chromosomes.	149
7.9	ROC curve analysis for tumor histologic subtype prediction. Baseline, Random Chromosome, Best Chromosome.	150
7.10	Line plots of fraction of non-functional chromosomes across generations.	151
7.11	Line plots of fraction of phenotypic clones across generations.	152
A.1	SimpleMind Knowledge graph for segmentation of the abdomen, bone and spine as described in 3.2.2.	156
A.2	SimpleMind Knowledge graph for segmentation of the Kidneys Total Kidney Volume (TKV) using nnUNet as described in 3.2.2.	157
A.3	SimpleMind Knowledge graph of the kidney segmentation refinement methods as described in 3.2.2.	158

List of Tables

3.1	Mean and standard deviation of kidney detection performance comparison of methods for validation data.	57
3.2	Kidney detection performance comparison across noise levels for test data. .	59
3.3	Pairwise Statistical Comparison Results for FPR and FNR at 0% and 4% Noise	60
3.4	Mean candidate acceptance and rejection performance comparison of methods.	61
3.5	Candidate acceptance and rejection performance comparison of methods on test data.	62
3.6	Pairwise Statistical Comparison Results for True Acceptance Rate (TAR) and False Rejection Rate (FRR) at 0% and 4% Noise	64
3.7	Mean candidate acceptance and rejection performance comparison of methods across number of synthetic candidates.	64
3.8	Candidate acceptance and rejection performance comparison of methods across number of synthetic candidates on test data.	65
3.9	Pairwise Statistical Comparison Results Across All Candidate Selection Metrics and Number of Synthetic Candidates	67
5.1	RENAL nephrometry score criteria.	86
5.2	Dice similarity coefficient (DSC) summary statistics for kidney and lesion segmentations.	99
5.3	McNemar’s test results comparing within-variability CAD proportion ($\hat{p}_{cad}$) to RND_CAD ($\hat{p}_{rnd}$). BH-adjusted p-values. $n = 60$	111
5.4	McNemar’s test results comparing within-variability AUTOSEG_CAD proportion ($\hat{p}_{autoseg_cad}$) to RND_CAD ($\hat{p}_{rnd}$). BH-adjusted p-values. $n = 59$. .	112
7.1	Per-generation summary statistics for non-functional chromosomes (fitness=0 fraction of population) across outcomes.	147
7.2	Per-generation summary statistics for phenotypic clones (fraction of population) across outcomes.	152

Listings

4.1	Example HTCondor submit file.	76
6.1	Example YAML configuration for the predictive modelling agent.	120
6.2	Example YAML configuration for the ML pipeline.	122
6.3	Example YAML configuration for KNoLO optimization.	125
6.4	Example YAML configuration for the GA optimizer.	127
6.5	Example YAML configuration for KNoLO optimization of a Logistic Regression Model.	129
6.6	Example YAML configuration for KNoLO optimization of two models with distinct hyperparameters.	131
C.1	Uninstantiated YAML configuration for KNoLO optimization: Preprocessing.	164
C.2	Uninstantiated YAML configuration for KNoLO optimization: Feature Selection.	165
C.3	Uninstantiated YAML configuration for KNoLO optimization: Resampling.	166
C.4	Uninstantiated YAML configuration for KNoLO optimization: Model Selection.	167

Acknowledgements

I owe the completion of this dissertation to my mentors and colleagues from the Physics & Biology (PBM) program and from the Center for Computer Vision & Imaging Biomarkers (CVIB) at the University of California, Los Angeles (UCLA).

First, I thank the chair of my committee, Dr. Matthew S. Brown. Since first meeting him during my interview for graduate school he has always expressed a welcoming and kind demeanor with a pragmatic approach to research that resonated with me. This has remained the case throughout my years as a PhD student with him always being approachable to any questions or concerns I may have, as well as being direct and honest in his feedback while always being kind. He has pushed me towards quality research with the goal of meaningful clinical impact, rather than chasing marginal performance increases for leaderboards, thus giving me an outlook of what research should really be about. His mentorship style and approach have inspired me and it is what I seek to emulate in future leadership roles.

Next, Dr. Michael F. McNitt-Gray. I am not alone in saying that the PBM program could not have a better director. Since before I applied to the program he provided support from meeting with me to orient me on Medical Physics, to providing advice during a moment of concern during my first internship. This continued on during my years as a PhD student with a willingness to always talk about any concerns, with his calmness and wisdom grounding me in moments of distress. In research as well, Mike would not hesitate to ask difficult questions, not as tests, but to push us to be better scientists and produce meaningful quality work.

I want to next thank Dr. Grace Hyun Kim. Before entering grad school, statistics was not an area I was concerned about when it came to research. I naively thought that larger performance numbers are inherently better. Thanks to Grace, I now recognize the utmost importance of statistics in research: results and performance are nuanced and should be rigorously tested, especially given the potential impact our work can have on real patients. Thanks to her, I find myself to have become a better scientist, better able to scrutinize results from other works and my own.

I wish to thank Dr. John M. Hoffman. Despite joining later in my PhD years, he immediately became an important part in my research. He developed the fundamental framework on which the work in this dissertation relies, enabling me and other graduate students to accelerate our work through rapid tool development. In addition, his oratory skills have served as inspiration for me for future presentations. His conversational but professional style allows for quick understanding of otherwise complex concepts while stimulating meaningful discussion.

To complete the acknowledgments of my committee members, I give thanks to Dr. William Hsu. Will has been a great collaborator to CVIB as a whole. Having first met him during my first year as a graduate student, I found his feedback and research experience to be very valuable and always presented in a welcoming demeanor. He was able to provide a much-needed outsider perspective on this work, being able to highlight areas where clarity was needed or limitations that ultimately pushed me to produce innovative work within a saturated and fast-moving field.

Next, I wish to thank the CVIB staff. I give my thanks to Kalyani Vyapary, Pangu Teng, Liza Shrestha, Richard Pais, KP Wong, and Mitch Murphy for building and maintaining the infrastructure needed to enable my research, for helping me in research by providing insights into the clinical workflows, and for taking my projects into production and clinic. I also give my thanks to the CVIB imaging analyst team for providing me with the necessary data collection to make this dissertation possible. I give my thanks to the administrative staff: Judy Ho, Eloisa Rodriguez-Mena, and Mercedes Valladarez for helping me become established in CVIB, for organizing my meetings with faculty, and conference logistics.

Lastly, I would like to extend my thanks to past and present postdocs and graduate students of CVIB. My first and only postdoctoral mentor, Dr. Youngwon Choi, I give my thanks for teaching me the fundamentals of AI when I had no experience in the field. Her cheerful personality and abundant knowledge made the learning experience and work environment wonderful. An experience I hope to provide when I find myself in a mentorship

position. Wasil Wahi, my gateway into CVIB, I give my thanks for teaching me the inner workings of the group and in-house software. For always replying to my incessant questions during my first years and being an all-around wonderful collaborator and friend. I give my thanks to Siyuan Wei, a savior in our less stellar moments of the group, a wonderful researcher and person with an admirable work ethic and unmatched politeness. Morgan Daly, our group's source of sanity, I give my thanks for her kindness ever since I joined the group, for keeping me in check and holding me accountable for my misbehavior, and for sharing a mutual love of cats. I give my thanks to Spencer Welland and Dallas Tada. Although they were not part of my cohort, their timing in joining the group allowed them to become the colleagues I had missed during my first year. We developed as scientists and experienced Los Angeles together. Dr. Andie Gong and Tanmay Hukkeri, although briefly members of the group, in their short stay they quickly became fundamental members, in and outside of research, that I will always remember. I give thanks to Jin Kim, a husband and father, an admirable person for his work ethic, passion for science, and commitment to his family. Tommaso Frigerio, the youngest member of the group, I give thanks for being a source of youthful spirit within our group, and for always providing insightful conversations. I thank Zixuan Zhao for being a great collaborator with inspiring quick understanding and intelligence, and for being able to provide great break room conversations. Lastly, our most recent student member, I would like to thank Minji Kim for keeping me company during late work nights, for being a wonderful source of conversation, calmness, and inspiring kindness. To all of them I am forever grateful for making my graduate school experience as wonderful as it was, for supporting me at my highest and lowest, and for being more than just colleagues, but brothers and sisters.

Vita

Education

University of California, Los Angeles 2023

M.Sc. in Physics & Biology in Medicine

University of Puerto Rico, Humacao 2020

B.Sc. in General Biology; Applied Physics minor equivalent

Research Experience

Center for Computer Vision & Imaging Biomarkers, UCLA 2021–2026

Graduate Student Researcher

O’Hern Group, Yale University 2019

Research Intern

Theoretical Physics Lab, University of Puerto Rico, Humacao 2018–2020

Undergraduate Research Assistant

Certification

American Board of Radiology 2022

Medical Physics Part 1 (ABR)

Awards

University of California, Los Angeles 2020

Eugene V. Cota Robles Fellowship

University of Puerto Rico, Humacao 2018

Maximizing Access to Research Careers Undergraduate Student Training in Academic Research (MARC U-STAR) training grant

University of Puerto Rico, Humacao 2018

Puerto Rico-Louis Stokes Alliance for Minority Participation (PR-LSAMP) grant

Publications

Peer-Reviewed Journal Articles

1. Wei, S., Shrestha, L., Melendez-Corres, G. & Brown, M. S. Anatomically Guided Deep Learning System for Right Internal Jugular Line (RIJL) Segmentation and Tip Localization in Chest X-Ray. *Life* **15**, 201 (2025).
2. Esteban, E. P. & Meléndez, G. A compartment model for total body irradiation. *Informatics in Medicine Unlocked* **20**, 100359 (2020).

Submitted Articles

3. Melendez-Corres, G., Vyapary, K. V., Pais, R., Wong, K., Lopez, M. M., Diaz, E., Kamini, S., Raman, S. S. & Brown, M. S. Interpretable Automated RENAL Nephrometry Scoring: A Multi-Reader Variability Analysis. Submitted; under review. 2026.
4. Melendez-Corres, G., Wahi-Anwar, M. W., Kim, J., Tada, D. K., Hoffman, J. M., Genender, J. & Brown, M. S. Selective Segmentation with Rejection Option: A Machine Reasoning Approach and Evaluation Metrics. Submitted; under review.

Chapter 1

Introduction

1.1 Specific Aims

The goal of this dissertation is to develop general, agent-based cognitive AI tools for use in biomedical image segmentation, quantitative image analysis, biomarker development, and predictive modelling. They will be developed in the SimpleMind (SM) software environment [18]. The tools will be applied to kidney and lesion segmentation in CT with quantitative nephrometry scoring of renal lesions.

1.1.1 Specific Aim 1: Integration of kidney and lesion segmentation deep learning models supported by cognitive AI agents.

In SimpleMind we will integrate the Deep Learning (DL) nnUNet [51] as the neural-network portion of our Neuro-Symbolic (NeSy), cognitive AI model. Within an anatomical Knowledge Graph (KG), this model will be supported by symbolic agents capable of integrating reasoning

through first-order logical operations, fuzzy operations and probabilistic confidence measuring operations. Our hypothesis for this aim is that cognitive AI supported DL improves performance over the base DL method alone.

1.1.2 Specific Aim 2: Development of machine reasoning and quantitation agents for automated nephrometry scoring on CT.

Measurements and scoring systems are often used clinically to assess various conditions. As opposed to using only Machine Learning-based predictions, which provide outputs with minimal explainability, these scoring systems typically provide an intuitive and interpretable description of the target condition to aid in clinical decision making. However, these measurements are often performed manually which can be time-consuming and prone to inter-observer variability. We will study renal nephrometry scoring that involves extracting lesion and kidney features to assess lesion complexity and overall surgical feasibility.

A cognitive AI based Knowledge Graph allows us to readily automate these scoring systems while maintaining the same level of interpretability. We will incorporate an extensive general-purpose feature-calculation library of agents that can be integrated into processing pipelines within SM. Our focus will be on agents for the extraction of features for nephrometry scoring systems. However, these agents will be constructed to be general-purpose and applicable to other tasks. Validation of the agents will be done by comparison with human measurements. The hypothesis for this aim is that the automated scoring tools produce measurements within the variability of humans.

1.1.3 Specific Aim 3: Development of a general-purpose predictive modelling agent, applied to renal lesion pathology and surgical outcomes, and supported by automatic parameter tuning.

The main utility of clinical measurements, biomarkers and scores is that they provide information that can be used to predict conditions or treatment effects and thus inform the clinical decision-making process. Imaging measurements have the added advantage of being minimally invasive. Nephrometry scoring specifically has been used to predict renal pathology and surgical outcomes. Thus, we propose incorporating a general-purpose predictive modelling agent within SM. The agent will be designed for flexibility in methods for preprocessing, resampling, feature selection, model selection and model training. The agent will also be compatible with the SM automatic parameter tuning tool known as Knowledge Graph Learning and Optimization (KNoLO), to optimize the hyperparameters for predictive performance. The application of this agent will be in using nephrometry and radiomic features for prediction of renal lesion pathology and surgical outcomes.

1.2 Deep Learning for Biomedical Image Segmentation

1.2.1 Image Segmentation/Annotation

Image segmentation refers to the process of delineating the shape of a structure or Region of Interest (ROI) within an image which creates a labeled image, known as an image mask.

The pixels/voxels that comprise the ROI (also referred to as the foreground) are labeled with a non-zero value, usually 1 in binary masks, while the rest of the image pixels/voxels in the background are labeled with 0 [35]. Figure 1.1 shows an example of the segmentation of 3 objects in a 2D image. When done manually by humans, these object delineations are typically referred to as annotations or contours. This process is applied to many tasks from simple cropping of images, detection of objects for counting, or to describe an object’s characteristics from an image such as size, shape and texture in more complex quantitative analysis tasks; see Section 1.4. In the biomedical field, manual annotations are needed for contouring targets such as organs or lesions. These annotations can have a multitude of uses such as the development of biomarkers for diagnosis or prognosis of a disease to radiation dose estimates for clinical targets and organs at risk [63, 70]. Manual annotations such as these are typically done in 3D images such as those acquired from CT or MRI scanners. Figure 1.2 shows an example of annotation/segmentation in a renal CT image. These 3D images represent the body in x, y and z dimensions where typically x and y represent a 2D transverse slice of the body, right to left and anterior to posterior, respectively. The z dimension traverses multiple transverse slices in the cranial-caudal direction. Thus, to manually annotate a CT or MRI scan, the user has to scroll through many images (often numbering in the hundreds or more, depending on the resolution and field of view) and manually delineate the ROI on each slice it appears. Depending on the task and annotation target, this can become a time-consuming process which can slow down clinical workflows. Additionally, in modalities such as x-ray radiographs and ultrasound, the target structure in these modalities may have reduced contrast, resolution, or increased noise which adds difficulty to manual delineation of the ROI and increases inter-observer variability.

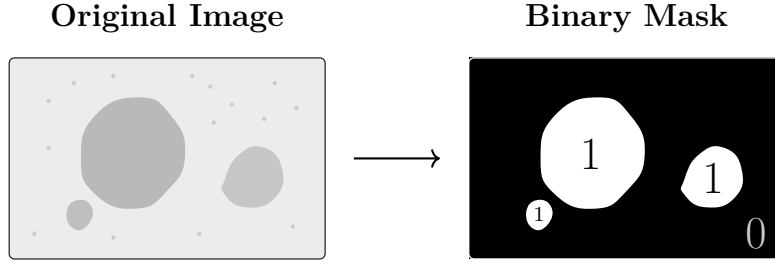


Figure 1.1: Illustration of image segmentation. Structures are delineated within the original image and represented in a binary mask, where foreground pixels or voxels belonging to the segmented objects are assigned a value of 1 and background pixels or voxels are assigned 0.

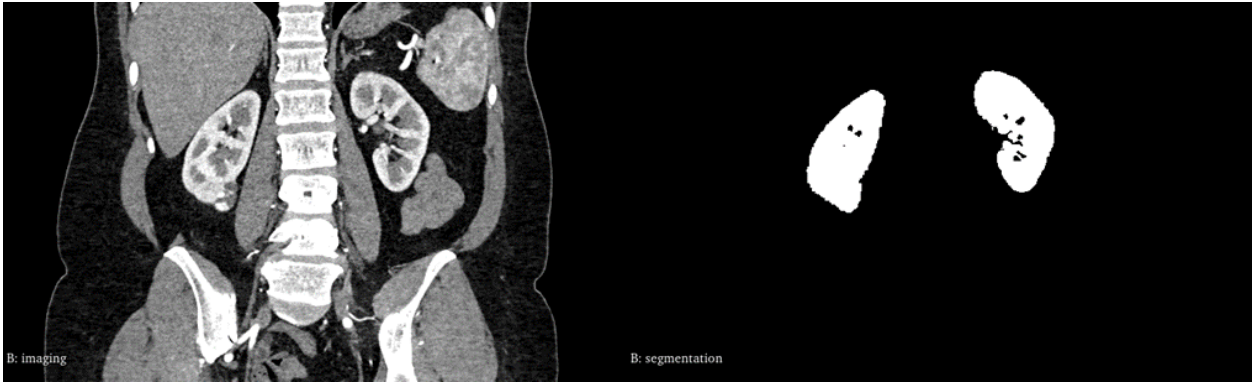


Figure 1.2: Example of image and annotation/segmentation in renal CT from the publicly available KiTS23 dataset [44].

1.2.2 An Introduction to Deep Learning

The limitations of manual annotations of ROIs have led to research into automation of this task. Many algorithms have been developed to accomplish this task; however, of relevance to this dissertation is the development of DL models. DL refers to the sub-field of AI that is based on multi-layered Artificial Neural Networks (ANN). ANNs, as the name suggests, intend to imitate how biological neural networks operate [37]. In Biology, the basic unit of the neural network, the neuron, receives an input signal either from an external stimulus or from another neuron. Upon receiving this signal, the neuron processes this signal and outputs a new signal in response depending on the strength and type of the input signal

which can be excitatory or inhibitory. This output signal can then be delivered to other neurons or activate a response to the initial stimulus [6]. Biological neural networks are capable of undergoing changes where connections and responses may weaken or strengthen in response to the environment, thus influencing the learning process [58].

Artificial neurons, also known as nodes, like biological neurons, receive input signals such as an image, feature vector values, or other node outputs. These inputs are then processed in the node through an activation function, which will produce a new output depending on the initial input signals and the internal weights and bias of the neuron [37]. Figure 1.3 shows an example of an artificial neuron.

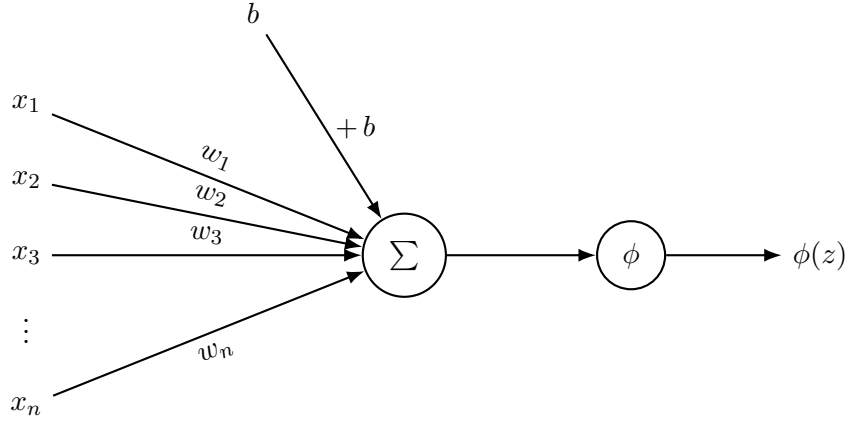


Figure 1.3: Artificial neuron example.

The artificial neuron receives inputs $x_1...x_n$, with connection weights $w_1...w_n$ and bias b . These components are processed by the neuron into the weighted sum,

$$z = \sum_{i=1}^n w_i x_i + b. \quad (1.1)$$

The output of the neuron is then determined by inputting the weighted sum into the activation function ϕ producing $\phi(z)$ [48]. The activation function ϕ is a task-dependent

choice. For binary classification a Sigmoid function, where outputs are constrained to $(0, 1)$, would be used, and for regression tasks a Linear activation would be used given that its outputs are $(-\infty, \infty)$. A commonly used activation function is the Rectified Linear Unit (ReLU) function which constrains outputs to $(0, \infty)$, typically used in neuron to neuron connections, ensuring positive responses for any given input [48]. Figure 1.4 shows example visuals of the activation functions described. Like in the biological example, the internal weights and bias of the artificial neuron are the components capable of being modified such that the network connections and output responses are able to strengthen or weaken depending on the task. The modification of the weights and biases involves a training/learning process guided by a loss function. Depending on the task, the loss function computes the error between the correct output and the network's output, from which the weights and biases are then modified to minimize the loss function.

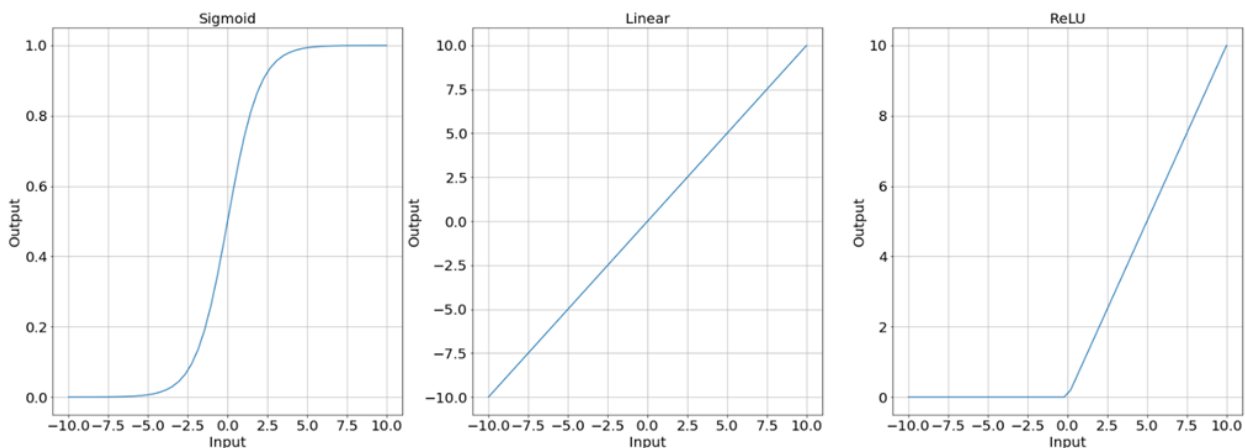


Figure 1.4: Example of activation functions used in ANNs.

Connecting the outputs of multiple layered neurons forms the ANN, Figure 1.5 shows a basic fully connected network, known as such given that all neurons from each layer all have connections to all neurons in the following layer. The general structure is composed of

an input layer of neurons, a set of hidden layers and an output layer. The hidden layers are called such since these layers learn and process the features from the given input, however, this information is not directly visible or outputted as is the case with either the input or output of the ANN. Due to this characteristic of the hidden layers, ANNs are referred to as a "black box" [37], with limited explainability of the model. The model can provide an output, but it does not provide a "why" for the output.

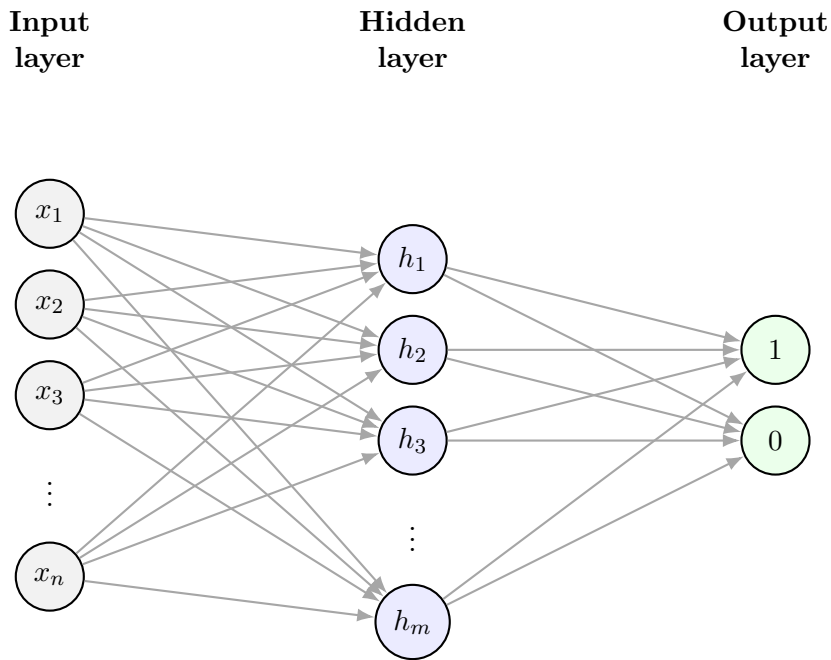


Figure 1.5: Basic fully connected artificial neural network with one hidden layer.

1.2.3 Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a variant of ANNs intended to mimic how biological vision operates. In Biological vision systems, the entirety of the visual input is not processed by all neurons at once. Rather, signals are passed through selective filters that create different receptive fields which activate different neurons. Information from multiple receptive fields

is combined to produce the final output for the given visual signal [68, 115]. In a CNN, the receptive field is created by the passing of a matrix filter (also known as a kernel) throughout the input matrix in a convolution operation to produce a feature map. This filter contains the learnable weights of the network, thus the neuron output is determined by the weighted sum of inputs inside the receptive field using the weights from the filter and then passed through an activation function [121, 76]. Figure 1.6 shows an example of the convolution operation using a 3×3 filter. Convolutional layers are followed by pooling operations, where the feature map produced by the convolutional layer is down-sampled typically by max pooling, average pooling, or a strided convolution to aggregate and summarize the feature map for the following convolutional layer. This computation pattern has the effect of enabling hierarchical learning where different levels of features are learned by the CNN by increasing the relative size of the receptive field relative to the input [66, 121, 76].

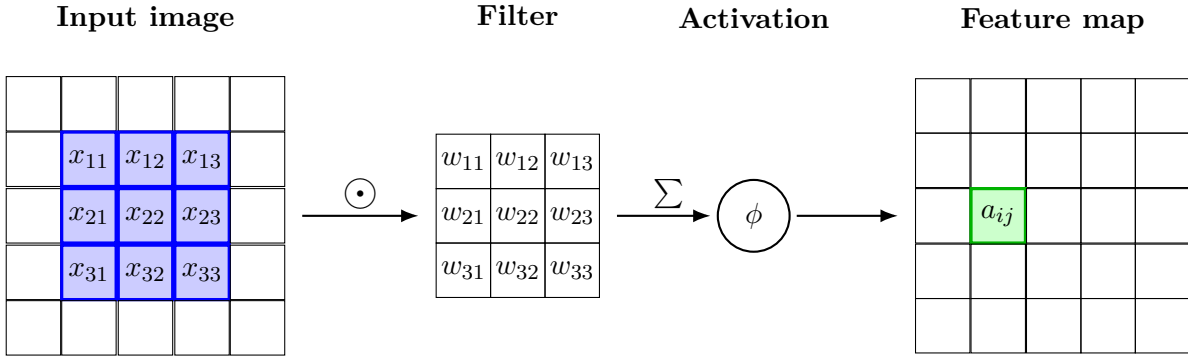


Figure 1.6: Illustration of a convolution operation. A filter (kernel) is applied to a local receptive field in the input image through element-wise multiplication ($\odot$), which are then summed (Σ) and then passed through an activation function (ϕ) to produce feature a_{ij} . Sliding the same filter across the image generates a complete feature map.

1.2.4 Applications of Deep Learning in Imaging

The initial development of CNNs can be attributed to Fukushima [32] with the development of the Neocognitron. While their work does not mention convolutional layers directly, they developed an ANN, inspired by early models of the visual nervous system, where neurons are layered in a hierarchical learning structure and inputs are transformed through receptive fields. This early model was capable of self-learning recognition of drawn digits, correctly identifying digits even when distortion was applied to the images. The next milestone in CNN development came with the development of LeNet by LeCun et al. [65]. In their work, they made direct use of 3 convolutional layers with 5×5 receptive fields, as well as two pooling layers (one after the first two convolutions) through averaging to down-sample the feature maps. After the last convolutional layer, they applied fully connected layers with a softmax activation, using the information learned from the feature maps to output a single prediction value for a hand-drawn digit image input. Despite these advancements in network architecture design, progression and adoption of CNNs was limited by the computational hardware limitations available at the time. The advancement of computational hardware such as the Graphical Processing Unit (GPU), capable of parallel processing, enabled deeper networks with faster training times and improved performance. This allowed for the development of new architectures for image classification. One such architecture was Alex-Net developed in 2012 [59]. The Alex-Net architecture is composed of 5 convolutional layers with 3 max pooling layers, followed by 3 fully connected layers with a final softmax activation as in LeNet. The Alex-Net architecture achieved state of the art performance on an image classification task of 1000 labels.

The previously described networks are fundamental in the development and advancements of CNNs in imaging. However, they are classification models where an image is given as input and the output is a single integer representing a specific class or label. These networks do not address the task of image segmentation described in Section 1.2.1. The task of image segmentation can be considered a variant of classification. Rather than classifying the image as a whole, individual pixels are classified as being from one class or another. This method is known as semantic segmentation [73]. The first major breakthrough in CNNs for semantic segmentation arrived with the development of the Fully Convolutional Network (FCN) in 2015 by Long et al. [73]. In the previously described networks, pooling operations down-sample feature maps which are eventually given as inputs to fully connected layers that output a single value. The FCN, however, as the name suggests, has no fully connected layers, but rather it is composed fully of convolutional layers. In their work, rather than developing a uniquely novel architecture, they modified pre-existing classification CNNs by replacing fully connected layers with convolutional layers. This converts the feature vectors into class score heatmaps representing the pixel-wise neuron activations for a given input image. The final output layer is composed of a transposed convolution layer that functions as a deconvolution which up-samples the heatmap into the original image size, producing the segmentation output.

In the field of medical imaging, the primary CNN work of interest is the U-Net architecture by Ronneberger et al. [95] in 2015. They extended the concept of the FCN for semantic segmentation into medical imaging segmentation. Figure 1.7 shows a general schematic of a U-Net architecture. This architecture initializes through an encoder path, where, as in the previous architectures, convolution and pooling layers are applied subsequently extracting

features at various resolutions, with the number of filters increasing as resolution decreases. The encoder ends at the bottleneck where the minimum resolution and maximum number of filters are reached. After the bottleneck, the decoder path begins. Within the decoder path, the feature maps are subsequently up-sampled, followed by additional convolution blocks to continue the feature extraction and learning process. The convolution blocks use skip connections to concatenate features learned in the encoder unto the decoder path. The final segmentation output is then obtained through 1×1 convolution followed by a sigmoid activation function for binary segmentation or softmax for multi-class segmentation. The U-Net architecture demonstrated its effectiveness by improving on state of the art performance by significant margins in three cell microscopy segmentation tasks.

The development of the U-Net architecture was extended to volumetric data with the development of 3D U-Net [126], where the two-dimensional convolution and pooling operations from the original U-Net were replaced with their respective three-dimensional versions. This allowed them to apply the U-Net architecture to segment *Xenopus* kidney embryos from volumetric microscopy data using sparse annotations and fully-automated segmentation. Direct application into volumetric patient data was achieved with the V-Net architecture, used to segment prostate volumes in three-dimensional MRI images [83]. While not a direct conversion of U-Net into three dimensions, V-Net follows the core principles of U-Net, being a FCN using encoder and decoder paths connected through skip connections. However, the primary novelty of interest is their use of the Dice Similarity Coefficient (DSC) as a loss function. DSC is defined as:

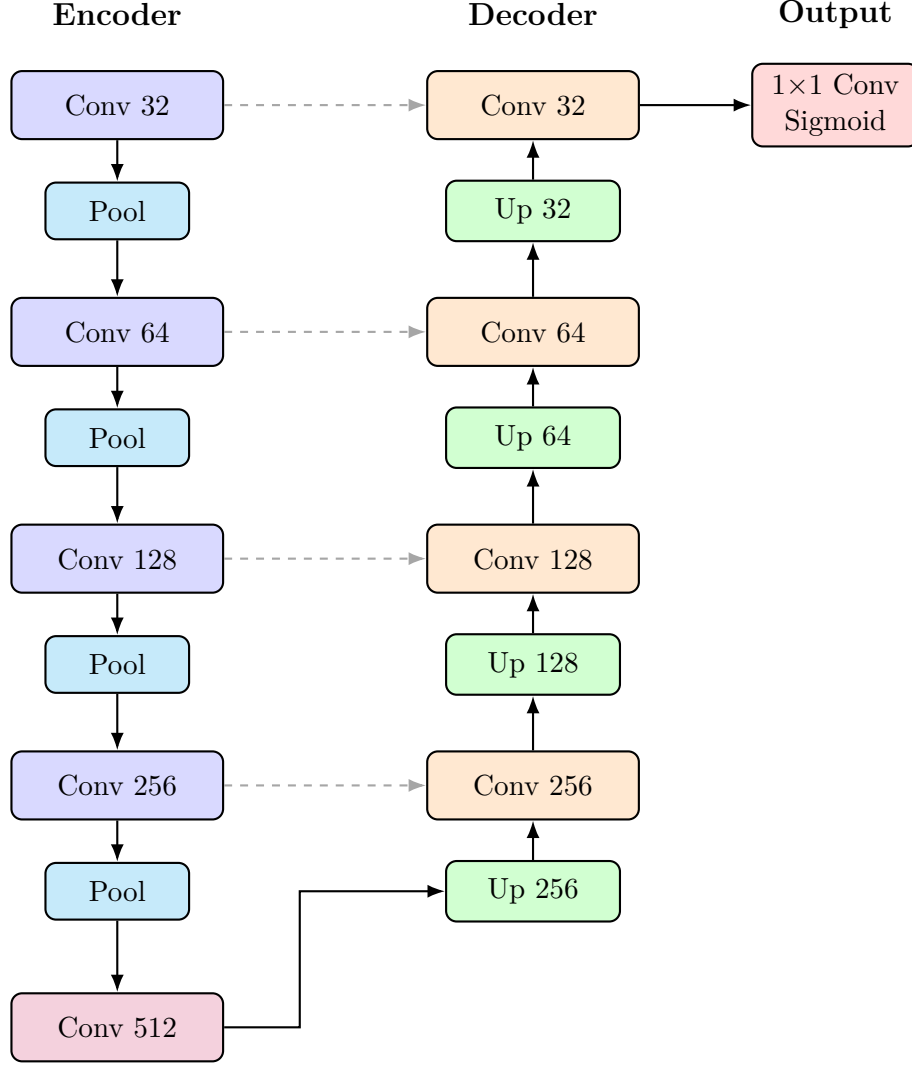


Figure 1.7: General schematic of a U-Net architecture. Each encoder block contains two 3×3 convolutions followed by ReLU activations and 2×2 max pooling. The decoder uses 2×2 transposed convolutions for up-sampling, concatenates encoder features through skip connections, and refines them with two 3×3 convolutions. A final 1×1 convolution with sigmoid activation produces a single-channel binary segmentation mask.

$$\text{DSC} = \frac{2 \sum_{i=1}^N p_i g_i}{\sum_{i=1}^N p_i + \sum_{i=1}^N g_i}, \quad (1.2)$$

where p_i represents the prediction voxels and g_i represents the truth voxels. Thus the numerator represents twice the number of elements where the prediction and truth intersect, while the denominator represents the union of the prediction and truth. This produces a

metric for quantifying the degree of overlap between two binary image masks, normalized between 0 and 1, where 0 indicates no overlap, and 1 indicates perfect overlap [124] (see Figure 1.8).

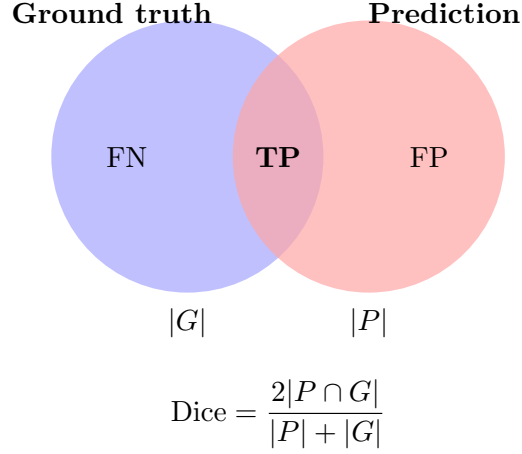


Figure 1.8: Venn-diagram interpretation of the Dice coefficient. The Dice score measures overlap between the predicted region P and the ground-truth region G . The overlap corresponds to true positives (TP), while the non-overlapping regions correspond to false negatives (FN) and false positives (FP).

For minimization during the training process, the DSC can be written as a loss function as:

$$\mathcal{L}_{\text{DSC}} = 1 - \frac{2 \sum_{i=1}^N p_i g_i + \epsilon}{\sum_{i=1}^N p_i + \sum_{i=1}^N g_i + \epsilon}, \quad (1.3)$$

where ϵ represents a smoothing factor to prevent division by zero. The architectural innovations and use of DSC as a loss function enabled V-Net to place among the top performers of the PROMISE 2012 prostate segmentation challenge [72]. Their use of the DSC loss function led to the popularization of overlap-based metrics for training and evaluation of medical image segmentation models. DSC based losses and variants are currently the standard for many modern segmentation models.

The architectural and network training developments of CNNs described in this section have had a major impact in the space of medical imaging. Classification tasks have evolved into precise, automated segmentations of anatomical structures and diseases in two-dimensional and three-dimensional image data. In recent years, transformer-based methods have made a transition from natural language processing into medical image analysis, becoming increasingly prominent as a Vision Transformer (ViT) [41]. However, recent work has shown that CNNs can match ViTs at scale under comparable training budgets in large-scale image classification [108]. CNN-based segmentation models therefore remain a well-justified focus, because of their continued practical relevance and thus they form the basis for the methods explored in this dissertation.

1.3 Cognitive AI using Neuro-Symbolic methods

Cognition can refer to the capability of processing knowledge to achieve reasoning. In the field of Deep Learning, an approach to create cognitive AI systems is through NeSy methods. NeSy methods are a hybrid approach to Deep Learning, derived from two components: neuro, referring to neural networks, and symbolic, which represent knowledge in the form of rules, relationships and known priors [87]. As mentioned in 1.2, neural networks by themselves operate with a "black box" approach that limits their explainability and ability to apply commonsense reasoning. Symbolic methods by themselves lack the ability to adjust and learn from data which impacts their generalization. Thus, implementing NeSy methods enables these two approaches to complement each other. The neural network learns from data while symbolic methods constrain outputs within commonsense reasoning and

provide explainability of the model’s outputs [87]. Symbolic methods incorporate logic to achieve a form of deductive reasoning. For example, relationships between objects can be established using propositional logic. This establishes connections using logical connectors or operators that can be conjunctive, disjunctive or negation (*and*, *or*, *not*) [102]. Extending from propositional logic, First-order Logic (FOL), also known as Predicate Calculus, is used to address limitations in natural language representation that come from only using propositional logic. FOL symbolic methods allow the construction of a KG to represent connections or relationships between structures with more information by using predicates, or rather, alternative functions not limited to propositional logic operators. Figure 1.9 shows an example of a simple KG using FOL applied to anatomy. Using the 3 defined constants: Left Kidney (k_L), Right Kidney (k_R), and Spine (s), we can utilize FOL to establish truth relationships among them. The left kidney is to left of the spine ($LeftOf(k_L, s) = True$) and the right kidney is to the right of the spine ($RightOf(k_R, s) = True$).

FOL Relations in a Knowledge-Graph Style Representation

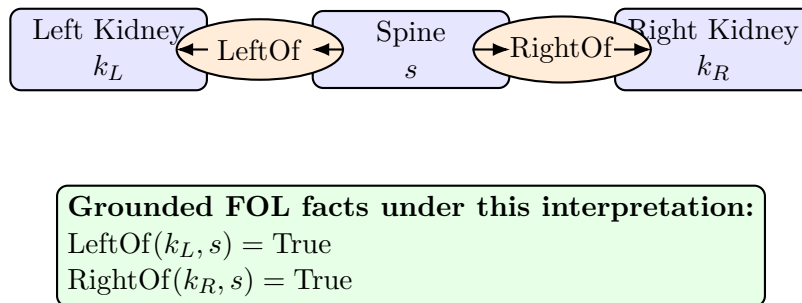


Figure 1.9: Knowledge-graph style representation of First-Order Logic (FOL) using an anatomical example. Constants are represented as entities, while predicates encode labeled relations between them. The resulting grounded predicates can be interpreted as factual statements under a given interpretation.

1.3.1 Fuzzy Logic and Membership Functions

Relationships utilizing propositional logic and FOL are accurate as long as a predefined boundary/threshold exists. In our kidney example, we can consider $LeftOf(D_k, s)$ in Figure 1.10, where D_k represents the distance of a kidney from the spine. There is a single defined boundary at $D_k = 0$ which allows us to establish whether a given D_k will output *True* or *False*. In practice, anatomical relationships can be ambiguous, defined within gradual ranges as opposed to having a single threshold.

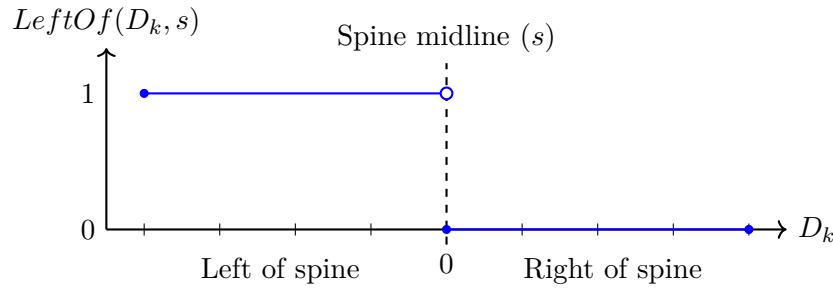


Figure 1.10: Representation of $LeftOf$ FOL relationship as the function $LeftOf(D_k, s)$. Distance of a kidney (D_k) relative to the spine midline (s). Truth values are obtained at decision boundary $D_k = 0$ when $D_k < 0$.

To model these ambiguous relationships we can use fuzzy operators. While regular logical operators provide a binary answer (*False*, *True*) from a predefined boundary, fuzzy operators provide a degree of confidence to handle more gradual and ambiguous relationships. Fuzzy operators can be interpreted as measuring the confidence that a proposition is true [102]. Of interest to this dissertation is the use of fuzzy membership functions. As the name suggests, fuzzy memberships are functions that model the degree of membership that a given input has towards a set. For example, a fuzzy membership function can model the degree to which a given length measurement can be considered "short" or "tall". Concepts such as "short" and "tall" tend to be relative, imprecise statements, considered subjective. However, a fuzzy

membership function can model this ambiguity, to output a value in $[0, 1]$, as a measure of how "tall" a given input can be considered, depending on user definitions [10]. Continuing with our applied example of the kidney and spine relation, we can consider the statement: *the right kidney is to the right of the spine*. In this example, any object measured to be located to the right of the spine ($D_k > 0$) will yield *True* as to being the right kidney. However, multiple objects can be located to the right of the spine, thus this statement, by itself, does not function to accurately discriminate the correct right kidney from other potential objects in the vicinity. To improve this, we can instead apply anatomical knowledge of the right kidney where, while located to the right of the spine, its location is, in most cases, within a range of distances from the spine. Thus, a fuzzy membership function can be constructed to receive an input distance from the spine and output a membership or grade of consistency of the right kidney. Figure 1.11 shows examples of different forms of membership functions $\mu(x)$. As seen in the Figure, membership functions can be created in various forms, to represent the membership output within the interval $[0, 1]$. Note, however, this interval does not represent a probability, but rather an empirically derived measurement of the degree a given input satisfies its belonging (membership) to a set [10,11].

1.3.2 Recent NeSy AI Works in Medical Imaging

NeSy methods have found usage in a multitude of applications in recent years. However, their use in medical imaging has been limited. A recent survey of NeSy methods by Wang et al. [116] reviewed 70+ works of NeSy applications. The applications of NeSy methods were found to be typically in: scientific discovery, programming systems, question answering,

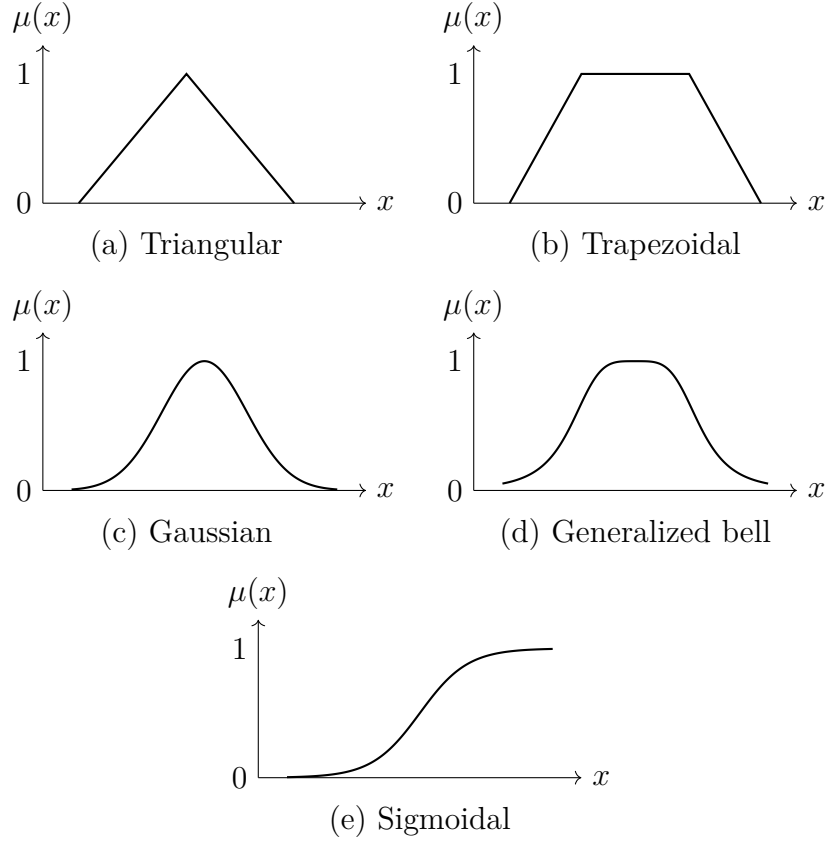


Figure 1.11: Examples of common fuzzy membership functions: (a) triangular, (b) trapezoidal, (c) Gaussian, (d) generalized bell, and (e) sigmoidal. Each function is bounded within the interval $[0, 1]$.

vision-language analysis and reasoning, robotics and control, visual scene understanding, mathematical reasoning and argumentation. While not a comprehensive review of NeSy application, nonetheless, despite the diverse set of topics, no direct application of NeSy AI into medical imaging is found within the works surveyed.

Medical applications of NeSy methods are seen in diagnosis applications to improve diagnostic accuracy and improve result explainability. A medical application of NeSy AI is seen in the work of Lu et al. [75]. They applied differentiable versions of logical operators with learnable parameters within these to construct a Logical Neural Network (LNN). Using the Pima Indians Diabetes dataset [8], they constructed two primary LNN models to predict a

diabetes diagnosis based on metabolic and physiological factors, as well as family history and hereditary background. Compared to common Machine Learning approaches such as Logistic Regression and Random Forest, their LNN models achieved superior predictive performance for diabetes diagnosis with added model explainability to support the diagnostic decision.

In medical imaging specifically, recent NeSy AI applications can be found in the work of Gao et al. [33], where they incorporated a CNN with symbolic reasoning in the form of a Concept Rule Learner (CRL) capable of learning boolean logical rules from visual concepts to improve explainability in medical image classifications. The NeSy aspects of the CRL appear in its use of differentiable conjunction and disjunction layers which enables optimization of these logical operations. The CRL model was applied to image classification of skin diseases and white blood cells. Compared to similar concept-based methods, they achieved top performance in accuracy of concepts while maintaining comparable results on diagnosis metrics.

A KG-like approach was implemented by Sengupta and Rekik [100] with the development of FireGNN. Their model consists of a Graph Neural Network (GNN), a variant of ANN’s that operate on graph-based inputs and outputs where connection strength between constants (also referred to as nodes) become trainable parameters. They mention that, similar to ANNs, GNNs operate as black boxes, despite the resemblance to interpretable FOL KGs. Thus, to improve GNN performance, they incorporated a trainable symbolic reasoning in the form of Fuzzy Interpretable Rule Embeddings (FIRE), achieving improved classification performance on all backbones on 2D patch images from the MedMNIST dataset [122].

In an application on volumetric imaging data, He et al. [42] developed a NeSy framework, known as NeuroSymAD, to mimic a clinician’s reasoning when analyzing MRI images for

Alzheimer’s disease diagnosis. Their model is composed of a neural perception module which provides an initial probability of Alzheimer’s disease in the input scan. The initial probability is passed to a symbolic reasoning module for refinement, using domain knowledge and patient information with learned rules. The learned rules are provided by a Large Language Model (LLM) through an automated knowledge acquisition module. In this module, the LLM generates rules by extracting information from clinical guidelines, research and educational sources. This module is also capable of providing a human-readable interpretation of the diagnostic results. The NeuroSymAD model outperformed baseline models in Alzheimer’s disease diagnosis, as well as improve performance of neural network models used as a backbone for the neural perception module.

As seen in the previously described works, medical imaging applications of NeSy methods are focused around diagnosis, with limited applications in segmentation. An example of a segmentation application can be found in the work of Abdullah et al. [2]. They developed the NeSy AI model called DeepInfusion for the segmentation of intracranial aneurysms on two-dimensional Digital Subtraction Angiography (DSA) images. The DeepInfusion model aims to improve segmentation performance through the incorporation of expert domain knowledge. However, the implementation of NeSy methods does not apply the typical logic-based rules as in other works; rather, hand-crafted features of interest from domain knowledge are incorporated within the segmentation CNN, hence the use of the term "infusion". Nonetheless, this approach demonstrated segmentation performance improvements for various CNN backbones, both on their in-house and publicly available DSA imaging datasets.

A recent extension of NeSy methods into volumetric biomedical image segmentation is found in the work of Bergamin et al. [9]. They aimed to incorporate common medical

knowledge by applying a Logic Tensor Network (LTN) into the ANN architecture SwinUNETR for automated segmentation of the brain hippocampus in MRI. Similar to the previous discussed works, the LTN employs differentiable logical operations that are optimizable via training. Using this framework, they employed FOL spatial constraints such as anterior and posterior connection, anterior not containing posterior and vice versa, and similar volumes between anterior and posterior. Evaluated on the MRI dataset from the Medical Segmentation Decathlon [3], they demonstrated improved segmentation performance from the baseline SwinUNETR with a reduction in constraint violations suggesting the generation of segmentation outputs better align with expected anatomy.

These works demonstrate the potential capabilities of NeSy methods for improving not only model performance, but model explainability. A model that can provide accurate outputs with interpretations providing an explanation for said outputs is a model that can be considered trustworthy, and thus suitable for clinical implementation. Nonetheless, NeSy AI is an emerging field with limited medical applications, particularly in medical image segmentation.

1.4 Quantitative Image Analysis & Imaging Biomarkers in CT

As mentioned in Section 1.2.1, segmentation of anatomical structures in medical imaging are of utmost utility to extract quantitative features and potentially inform the clinical decision-making process. These features are also known as imaging biomarkers. Biomarkers

are defined as: *"A characteristic that is objectively measured and evaluated as an indicator of normal biological processes, pathogenic processes, or pharmacologic responses to a therapeutic intervention"* [12,17]. Thus, biomarkers can originate from any biological measurement that can be indicative of a diagnosis, progression or prognosis. For example, when assessing renal function, the glomerular filtration rate is measured by an inulin injection and clearance from the kidneys as a biomarker [101]. However, bodily punctures involving injections of foreign substances, tissue extraction such as in the case of biopsies, or other methods of quantifying biological processes are considered invasive procedures leading to increased risks and distress for patients. Thus, with the increased use of anatomical imaging techniques such as CT imaging, imaging biomarkers offer a minimally invasive alternative for diagnosis, prognosis and treatment response assessment.

CT imaging is of particular interest for quantitative image analysis using segmentations since it provides three-dimensional information to describe shape based features, and also the intensity values of the voxels forming the images have clinically meaningful significance. Voxel values in CT imaging are known as Hounsfield Units (HU), which measure the x-ray attenuation of a volume element normalized to the x-ray attenuation of water and scaled by 1000. Thus, the HU for a voxel of water ideally becomes 0, while structures less dense than water are < 0 , with air at ≈ -1000 and denser than water being > 0 , with dense structures such as bone at $\approx +1000$ or more in the case of added iodine contrast [16].

The attenuation-based voxel values of CT imaging allow for the development of biomarkers based on intensity-based features. Previous works have leveraged the utility of contrast-enhancement and peak attenuation values to discriminate among different renal lesions [22] and evaluate quantitative features under the effects of CT dose reduction [23]. Voxel intensity

comparisons, however, are typically readily observable features by humans. However, to extract features beyond visual inspection of the image, we can turn to Radiomics, which can objectively quantify features that would otherwise not be measurable or features that are usually described in subjective terms [78]. An example is describing the texture of a structure in an image. A human observer might be capable of describing the structure as "smooth" or "rough", however, these terms are not readily quantifiable for use as a biomarker. Radiomics can quantify a structure's texture through methods that quantify the relationships, patterns and fluctuations among voxel neighborhoods [78].

Relationships between biomarkers and outcomes can be, in ideal cases, defined by discrete thresholds that can discriminate between classes, with negligible to no overlap between feature values (see Figure 1.12).

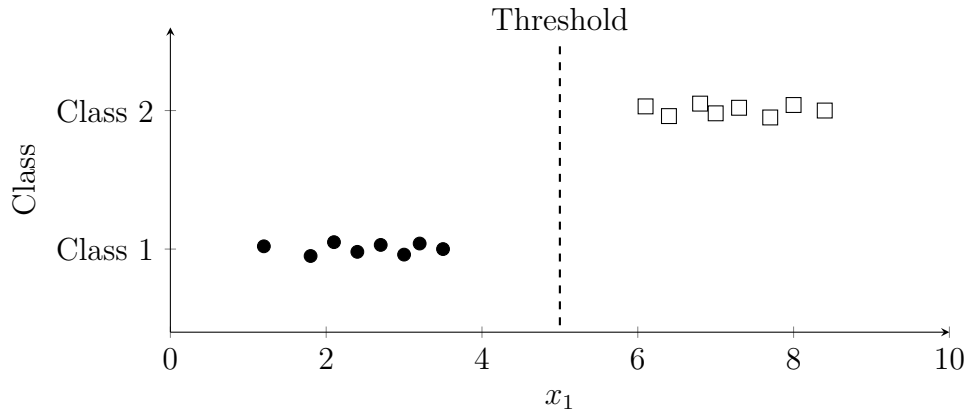


Figure 1.12: Illustration of a single feature (x_1) with clear separation between two classes. In this simplified setting, a single threshold can provide effective discrimination.

However, in real world datasets, feature and outcome relationships are seldom discretely defined as separable classes. Additionally, the extension to radiomics can extract thousands of features, increasing the dimensionality of the dataset and producing non-linear relationships between features and outcomes that cannot be modeled with standard mathematical functions.

To model these multi-dimensional, non-linear relationships, feature-based Machine Learning (ML) algorithms have been developed capable of being trained on feature input data to output a prediction of an outcome or class of interest. In Section 1.2, we already discussed a subset of ML methods, Deep Learning. The key difference is that Deep Learning methods such as ANNs are capable of learning which features to extract during the training process, while in traditional ML methods, the feature set is provided. Figure 1.13 illustrates a subset of examples of feature-based ML methods, namely: Logistic Regression, Decision Tree, K-Nearest Neighbor (KNN), Support Vector Machine (SVM).

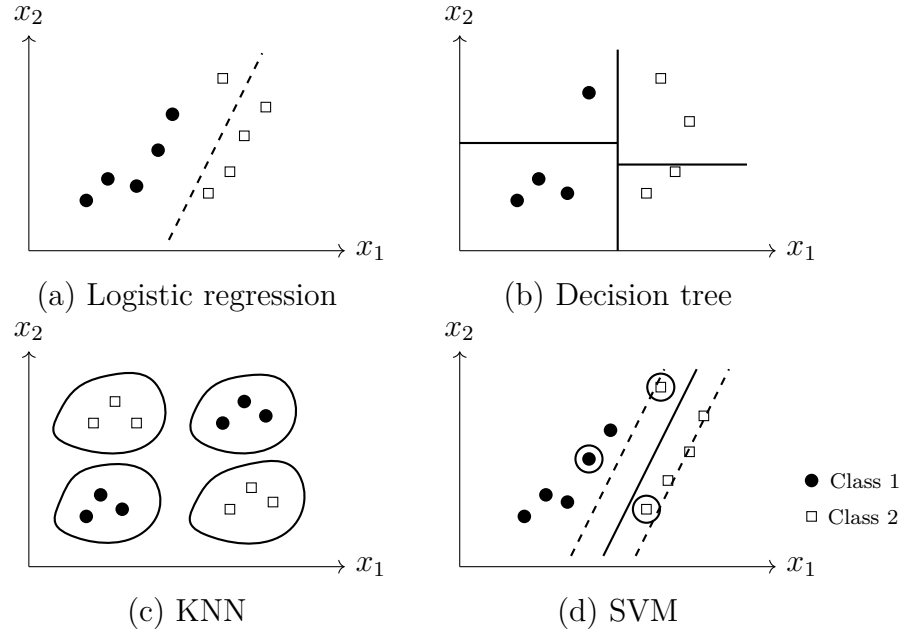


Figure 1.13: Illustrative decision boundaries for several machine learning models in a two-feature (x_1 , x_2) example. Logistic regression produces a linear decision boundary, decision trees partition the feature space through axis-aligned splits, KNN forms local neighborhood-based decision regions, and SVM seeks a separating hyperplane with maximum margin defined by support vectors. These examples illustrate how ML methods can model multi-dimensional and non-linear relationships to discriminate between classes.

The variety of steps and tools involved in quantitative image analysis is substantial. Multi-step pipelines are constructed from feature extraction to predictive modelling, with

each step having their own strengths, weaknesses, and suitable hyperparameters for any given task. Intended to solve multi-dimensional problems, the construction of these pipelines can become a multi-dimensional optimization task in itself. Thus, in this dissertation we will explore not only the construction of these pipelines, but the optimization as well.

1.5 Contributions and Dissertation Organization

As stated in the Specific Aims, the goal of this dissertation is to develop a cognitive AI framework for biomedical image segmentation, quantitative image analysis, biomarker development, and predictive modelling. While applied to the task of kidney segmentation and nephrometry, the framework is designed to be task-agnostic, where users and developers can adapt any agent or tool to their own task. The following is an organization of this dissertation providing the main contributions from each chapter.

Chapter 2 introduces the SimpleMind Software Environment as the base framework in which cognitive AI agents and tools will be developed. We demonstrate the library of task-agnostic, integrated neural network, computer vision, and reasoning agents for biomedical imaging. The agents will be used for Aims 1 and 2 of this dissertation.

Chapter 3 applies the neural network and reasoning agents to demonstrate how cognitive AI can improve segmentation performance by rejecting errors in kidney segmentation, thus improving AI trustworthiness for production and clinical implementations. We introduce the concept of **Selective Segmentation with a Rejection Option**, wherein segmentation outputs are evaluated for anatomical consistency to enable automatic acceptance or rejection and routing for further human review, thus moving towards autonomous clinical workflows.

Additionally, we provide a novel analysis of segmentation performance to assess the lack of segmentation error sensitivity in standard overlap-based evaluation metrics. This chapter provides formal testing and analysis for Aim 1.

Chapter 4 demonstrates the application of the cognitive AI SM kidney segmentation model for research and clinical deployment. We provide the considerations and infrastructure development required for the successful deployment of a segmentation model in research hospital environments.

Chapter 5 expands on the usage of the agents developed in Chapter 2 to extract features for the development of an automated nephrometry scoring system. We provide a thorough methodology for validation of automated nephrometry to identify if the system can effectively act as another human rater, which covers the validation portion of Aim 2.

Chapter 6 introduces the SM infrastructure for predictive modelling and hyperparameter tuning for Aim 3. We demonstrate the general usage for training and prediction, the internal operations through a proactive design where newer modelling methods can be used without requiring direct backend updates, and integration with SM’s KNoLO to enable automated hyperparameter tuning for a novel usage of a Genetic Algorithm method for optimizing predictive model pipeline end-to-end.

Chapter 7 completes Aim 3 by demonstrating application of the SM predictive modelling and automated hyperparameter tuning for the prediction of renal lesion pathology and surgical outcomes. We expand on nephrometry features to include radiomic features. In contrast to previous works, where only a Logistic Regression step is used to predict outcomes from features, we include and optimize pre-processing, feature selection, data resampling and, model selection and training hyperparameters. Additionally, we provide a diversity analysis

of the genetic algorithm optimization to identify bottlenecks in model convergence.

Chapter 2

The SimpleMind Infrastructure for Cognitive AI in Medical Imaging

2.1 Introduction to The SimpleMind Software Environment

SimpleMind (SM) is a cognitive AI software environment where users and developers can apply human knowledge and reasoning to guide and/or post-process deep-learning algorithms to produce reliable results and correct mistakes that are inconsistent with common-sense reasoning. SM will form the basis for the development of our methodology throughout this dissertation.

The contents of this chapter appeared in papers [81] and [82]

2.1.1 Agents in SimpleMind

A core concept that forms the functional units of SM is that of an agent. Software agents are derived from a programming paradigm known as Agent Oriented Programming (AOP), initially conceptualized by Shoham in 1993 [103]. As defined by Shoham: *"An agent is an entity whose state is viewed as consisting of mental components such as beliefs, capabilities, choices, and commitments"* [103]. The definition, however, can be considered subjective, and, by the author themselves, the determination of what can be considered an agent is left to the programmer. Nonetheless, Shoham's work outlines 2 behaviors for agents in an AOP system: (1) Reading messages and updating its mental state, (2) Executing an action and updating its mental state. These two behaviors form the basis of agent operation in SM.

The underlying AOP of SM is built on the Core Framework [62]. Using Core, SM agents function by communicating via what is referred to as the BlackBoard. Using the BlackBoard, agents send and read messages, to and from other agents. Agents execute a specific action written in their program and subsequently update the BlackBoard with their current state. This enables a rudimentary form of short-term memory while the BlackBoard is active, which enables any agent to access the information from any other agents in the form of Promises (inputs required for the agent to take action). Based on the Promises required, the Core controller can automatically resolve each agent's dependencies and thus execute agents in sequence or parallel depending on whether their required promise is available on the BlackBoard. Figure 2.1 illustrates an example of agent communication through the BlackBoard for a segmentation and reasoning refinement task. The SimpleMind environment consists of a library of Core agents that each perform specific tasks such as image processing,

mask processing, reasoning, and segmentation using integrated deep-learning algorithms [18]. These capabilities have allowed deep-learning segmentation to be implemented clinically for identification of chest x-ray tube placement [15, 119] and segmentation of the kidneys for organ donor volume estimations with SM [114]. In this chapter, we will describe the general functionality of the agents that were applied in the segmentation of the kidneys and nephrometry feature extraction.

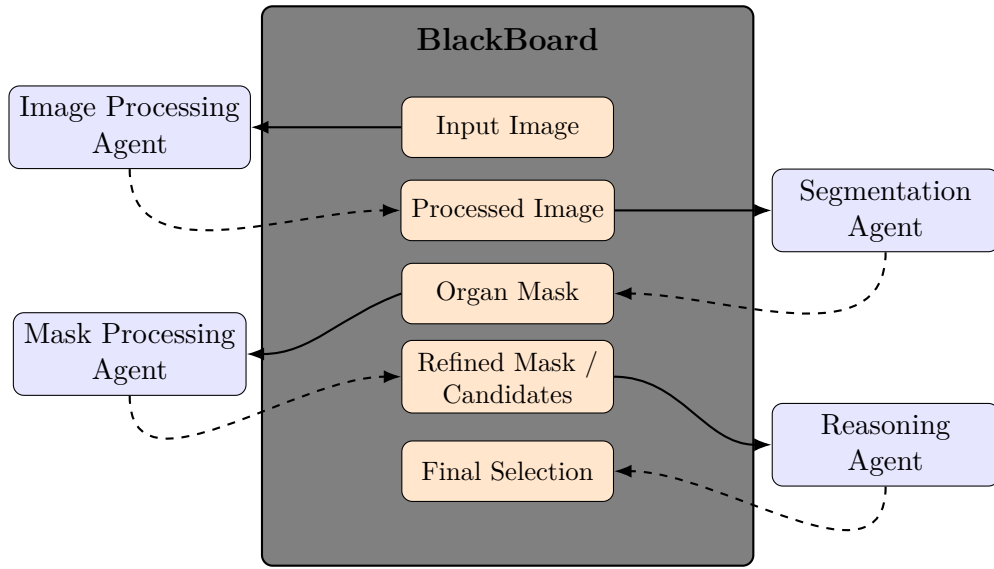


Figure 2.1: Representation of promise-based execution in the SimpleMind environment. Agents communicate indirectly through the BlackBoard, which functions as a shared communication space and short-term memory while active. Each agent reads the promise required for its execution, performs its assigned task, and then updates the BlackBoard with a newly generated output. These updated promises can then be used by downstream agents, enabling the staged execution of further agents.

2.2 Biomedical Image Segmentation using nnUNet

Developed by Isensee et al [51], nnUNet is a biomedical image segmentation method based on the original Unet architecture [95]. Using the Unet architecture as the base, nnUNet aims to

self-configure pre-processing, training and post-processing hyper-parameters based on training data and heuristics. The method initiates by extracting a data fingerprint which consists of the following features: distribution of image spacings, median shape, intensity distribution, and modality. This fingerprint is then used to configure rule-based parameters: image and annotation resampling strategies, target spacing, and intensity normalization. Additional rule-based parameters are configured based on user constraints (GPU memory limits and selection of either a low-resolution model, or a full-resolution model) and the fingerprint: patch size, batch size, network topology, and the cascade trigger for multi-network models (low-resolution to full-resolution). With the rule-based parameters determined, image data is pre-processed for training. Network training in nnUNet uses a set of fixed parameters: a Unet architecture template with instance normalization, leaky-ReLU and Deep Supervision; Stochastic Gradient Descent optimizer with Nesterov momentum = 0.99; Dice coefficient plus cross-entropy loss function; Linear Polynomial learning rate scheduler initialized at 0.01; data augmentation consisting of rotations, scaling, gaussian noise/blur, random brightness, random contrast, low resolution simulations, gamma correction, and mirroring; foreground oversampling of 33%; 5-fold cross-validation training for a length of 1000 epochs each, with 250 training mini-batches and 50 validation mini-batches per epoch. The inference procedure is also included in the fixed-parameters using a method consisting of a sliding window approach with patches overlapping by half their size and gaussian center weighting. Lastly, nnUNet configures empirical parameters through trial and error: ensemble selection where nnUNet determines which combination of nnUNet models produces the best performance on the validation data; post-processing consisting of largest connected component analysis (LCCA), where nnUNet applies this method iterating through the different foreground classes contained

in the data.

2.2.1 Integration of nnUNet in SimpleMind

The nnUNet library described is integrated into SM as an agent. The agent uses a forked version of the base nnUNet, found in <https://gitlab.com/gaby97/simplemind-nnUNet>, modified to enable inference per image rather than the nnUNet default of using a folder as an input. The agent is built upon two main scripts, an nnunet runner and the agent script itself is called in the Knowledge Graph (KG). The nnunet runner simplifies the usage of nnUNet by automating the required directory setup operations and commands for fingerprint, preprocessing, training and inference. With this implementation, a user constructs a KG using the nnUNet agent and only needs to provide the input images to either train or run inference. The implementation also allows for the input images to be received as Promises from the BlackBoard so any desired preprocessing tools can be applied before the nnUNet agent.

2.3 General Computer Vision and Utility Agents in SimpleMind

Deep Learning (DL) tools such as the nnUNet method provide powerful methods for organ segmentation. However, they typically require large amounts of data that may not always be available. As such, a DL approach may be considered unnecessarily excessive in terms of computational and data usage for a given task. Also, segmentations from the DL model,

might contain errors or require refinement via post-processing. To address this, SM includes a library of general computer vision and utility agents to generate segmentations and support DL models through the KG.

2.3.1 Image Processing Agents

While nnUNet is capable of pre-processing images adaptively, users can also apply custom pre-processing for input to the nnUNet model, their own custom DL model, or some form of feature extraction. The following agent tools can be used to pre-process image data; used for renal segmentation and feature extraction in CT described in later chapters.

HU Thresholding

SM contains an agent for simple image segmentation using intensity thresholding. The user provides a CT image as input and an upper and lower HU limit to segment the image voxels within a desired range. Figure 2.2 shows an example of the HU thresholding to segment the bones in a CT scan.



Figure 2.2: (A) Example of the HU thresholding agent being used to segment the bones in a CT scan. (B) 3D rendering of the bone segmentation in (A).

Image Cropping

SM can be used to crop images and masks using a bounding box mask image as an input. The target image will then be cropped to the bounding box dimensions while preserving the image’s physical information. Figure 2.3 shows an example of CT image being cropped to a bounding box surrounding the left kidney.

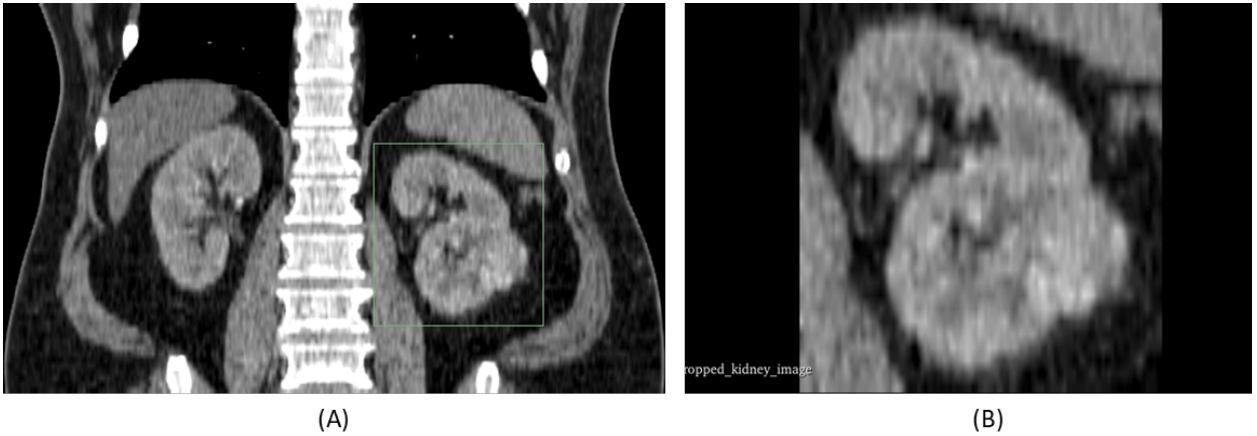


Figure 2.3: (A) Input image with bounding box for cropping. (B) Cropped image.

Resampling

DL methods and image processing techniques often operate using the image intensity values. However, medical images do not just contain intensity information, but also physical information to determine the real size and spatial location of organs and structures. Additionally, images in a dataset are often reconstructed with differing spacing, thus a voxel in one image does not represent the same physical size as a voxel in another image. To preserve this information during processing, we can resample the image arrays to a common spacing (often isotropic such as 1 mm in x, y, z). SM enables this through an agent that leverages the SimpleITK library [74] to resample images and masks via interpolation.

2.3.2 Mask Processing Agents

As mentioned in the beginning of this section, segmentation outputs such as those generated from DL models, or through other means like HU thresholding, can be further processed to refine the output mask or to produce a new segmentation. Here we describe the mask processing agents available in SM relevant to kidney segmentation in CT.

Morphology

Morphology is composed of a set of operations known as neighborhood operations, in reference to the fact that they use more than one pixel at a time for their calculations [5]. These operations are done using a structuring element of a specified size and shape to determine the neighborhood of pixels/voxels. While morphology encompasses a variety of operations, here we focus on those available in SM, erosion and dilation, and derived from these, opening and closing. The erosion operation, as the name implies, "erodes" the mask, by using the structuring element to remove positive pixels/voxels around the boundary of the mask. Dilation is the opposite of erosion, where positive pixels/voxels are added onto the mask, increasing its size. Opening performs an erosion followed by a dilation using the same structuring element. This has the effect of smoothing mask boundaries and removing specks of positive pixels/voxels (effectively noise) surrounding the mask. Inversely, closing performs a dilation followed by an erosion using the same structuring element. As the name implies, this has the effect of "closing", filling holes (negative pixels/voxels) contained within the mask. The SM morphology agent enables these four operations with a cross, rectangle, and ellipse as structuring elements. This agent leverages the OpenCV library [14] for all operations.

Figure 2.4 illustrates all available operations using a cross structuring element.

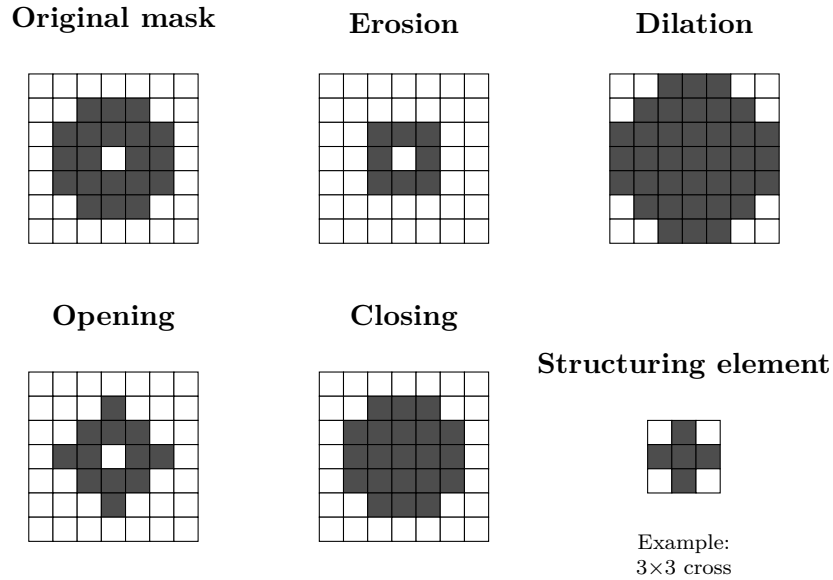


Figure 2.4: Illustration of basic morphological operations on a binary mask. Erosion removes foreground pixels around object boundaries, shrinking the mask. Dilation expands the foreground region by adding pixels around the boundary. Opening, defined as erosion followed by dilation, smooths boundaries and removes small foreground specks. Closing, defined as dilation followed by erosion, fills small holes and gaps within the mask. All operations are performed with respect to a structuring element, shown here as a 3×3 cross-shaped neighborhood.

Spatial Offset

The spatial offset agent in SM allows users to define a specific region within the image relative to a reference mask. From the reference mask, the centroid is computed and a subregion is generated by creating a bounding box using provided x, y, z upper and lower distances from the centroid. Figure 2.5 shows an example of a subregion generated to locate the region of the image where the spine is located using the abdomen as a reference.



Figure 2.5: Spatial offset output subregion bounding box for defining the spine region from the abdomen centroid.

Convex Hull

Given a set of points in the image, a convex hull algorithm finds the smallest convex boundary that encompasses all foreground (positive) pixels/voxels [36]. Figure 2.6 illustrates a convex hull operation. The SM convex hull agent can operate on a 2D or 3D input mask. The 2D method is based on the convex hull method from [47] using the OpenCV library [14]. This method loops through each axial slice of the image, creating a convex hull on each slice with foreground voxels using Sklansky’s algorithm [107]. The 3D convex hull method uses the SciPy library [113] to generate a convex hull based on Delaunay triangulation [26].

Connected Components

A core principle in the SM environment is the concept of candidates to which reasoning is applied (Sections 2.4.2, 2.4.3, 2.4.4). A candidate in SM is any individual connected

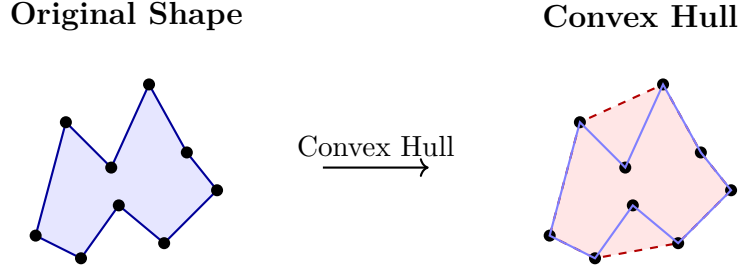


Figure 2.6: Illustration of the convex hull operation. Given a set of boundary points, the convex hull forms the smallest convex region that fully encloses all points.

component in the image mask, identified with the connect components agent using the cc3d library [105]. With this agent, segmentation outputs can be divided into multiple candidates for further evaluation and refinement. The agent also includes a voxel-count threshold where components under the user specified voxel threshold are rejected. Figure 2.7 shows an example of the connected components agent identifying candidates from a DL model output.

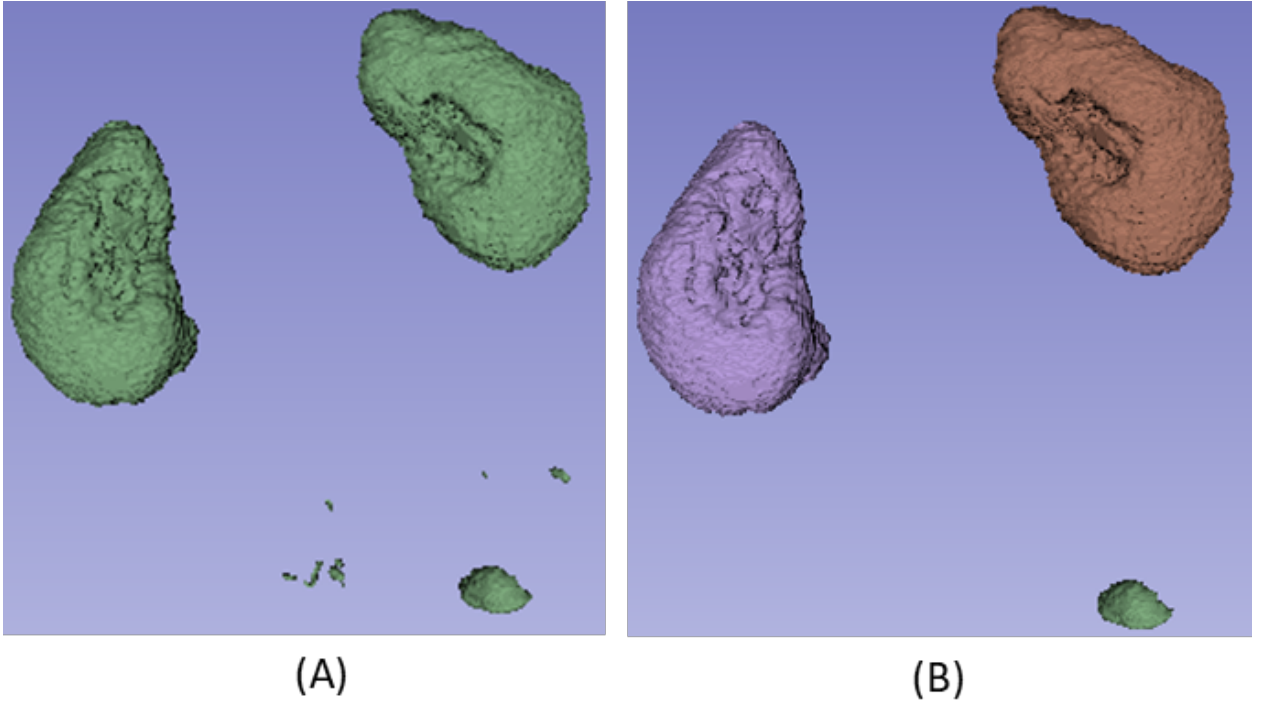


Figure 2.7: (A) Segmentation output from a DL model. (B) Identification of candidates using the connected components agent with removal of components under a voxel threshold.

2.4 Integration of reasoning in SimpleMind

DL methods by themselves are described as operating as a "black box", leading to segmentation outputs without explanation or reasoning. This lack of reasoning also leads to DL methods producing segmentation errors that lack common-sense, negatively impacting the model's trustworthiness and clinical implementation. Thus, in SM, we incorporate Neuro-Symbolic (NeSy) methods to supplement DL segmentation by applying anatomical reasoning. SM achieves reasoning using various methods such as logical operators applied to image masks, Fuzzy Membership Reasoning (FMR) based on linear piece-wise functions from extracted features, and Adaptive Reasoning (AR) which computes confidence scores by fitting a Probability Density Function (PDF) on extracted features, the latter two being capable of evaluating individual image candidates.

2.4.1 Mask Logical Operations

Logical operations are fundamental to Neuro-Symbolic AI. SM's mask logic agent leverages NumPy's *and*($\wedge$), *or*($\vee$), *not*($\neg$) and *xor*($\oplus$) to perform boolean logic operations between masks in a single tool [38]. These operations are applied element-wise on the array, generating a single array as output where the criterion is met. Given array A as our first input and array B as our second input, the logical *and* will generate an array with *TRUE* where both A and B are *TRUE*. Logical *or* will generate an array with *TRUE* where either A or B are *TRUE*. The logical *not* operation will generate an array with *TRUE* where A is *TRUE*, but *FALSE* where B is *TRUE* ($A \wedge \neg B$). Lastly, the *xor* operation, also known as exclusive *or*, will generate *TRUE* where $A \neq B$. Figure 2.8 shows example VENN diagrams representations of

the mask logical operations described.

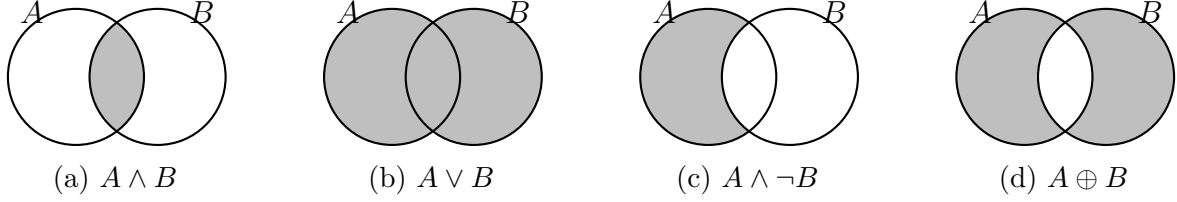


Figure 2.8: Venn-style illustration of logical mask operations between two input masks, A and B . Shaded regions represent voxels or pixels evaluated as *TRUE* in the output mask for each operation: logical *and* ($A \wedge B$), logical *or* ($A \vee B$), logical difference ($A \wedge \neg B$), and exclusive *or* ($A \oplus B$).

2.4.2 Fuzzy Membership Reasoning

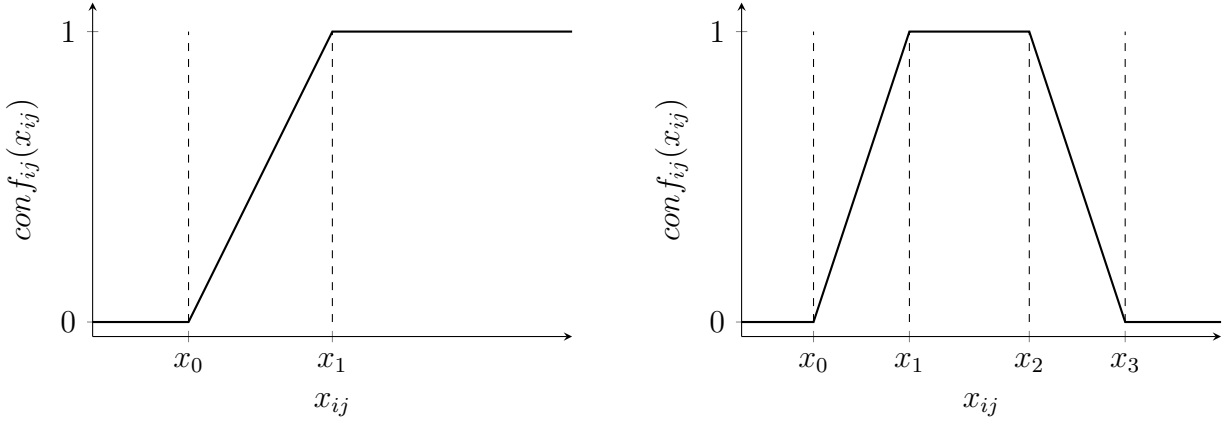
Given candidate j , FMR computes sub confidences $conf_{ij}(x_{ij})$ for every feature i based on linear piecewise functions with various user-defined thresholds for a feature value on candidate j , x_{ij} . In the case of two thresholds (x_0, x_1) the sub confidence function takes the form of:

$$conf_{ij}(x_{ij}) = \begin{cases} 0 & \text{if } x_{ij} \leq x_0 \\ \frac{x_{ij}-x_0}{x_1-x_0} & \text{if } x_0 < x_{ij} \leq x_1 \\ 1 & \text{if } x_{ij} > x_1 \end{cases} \quad (2.1)$$

With 4 thresholds (x_0, x_1, x_2, x_3) , the confidence function takes the trapezoidal form:

$$\text{conf}_{ij}(x_{ij}) = \begin{cases} 0 & \text{if } x_{ij} \leq x_0 \\ \frac{x_{ij}-x_0}{x_1-x_0} & \text{if } x_0 < x_{ij} \leq x_1 \\ 1 & \text{if } x_1 < x_{ij} \leq x_2 \\ 1 - \frac{x_{ij}-x_2}{x_3-x_2} & \text{if } x_2 < x_{ij} \leq x_3 \\ 0 & \text{if } x_{ij} > x_3 \end{cases} . \quad (2.2)$$

Both FMR confidence functions are illustrated in Figure 2.9.



(a) Two-threshold fuzzy confidence function.

(b) Four-threshold fuzzy confidence function.

Figure 2.9: Examples of piecewise linear fuzzy membership functions used to compute feature-specific sub confidences $\text{conf}_{ij}(x_{ij})$ for candidate j . Left: a two-threshold confidence function that increases from 0 to 1 between x_0 and x_1 . Right: a four-threshold trapezoidal confidence function that increases from 0 to 1 between x_0 and x_1 , remains at 1 between x_1 and x_2 , and decreases back to 0 between x_2 and x_3 .

The final confidence for candidate j , conf_j , is then computed via the arithmetic mean of the sub-confidences of every computed feature:

$$\text{conf}_j = \frac{1}{n} \sum_{i=1}^n \text{conf}_{ij} . \quad (2.3)$$

However, users can also designate certain features as essential. If an essential feature returns a confidence score of zero, $conf_j$ is then also set to zero, regardless of the confidence scores for the other features. This is useful for features that the target organ must have.

2.4.3 Adaptive Reasoning

The FMR method requires user-defined thresholds for every feature. This adds a degree of manual finetuning and limits the features a user can reasonably estimate thresholds for. To address this, SM can enable reasoning via a data-driven approach by fitting a PDF to every feature and automatically compute a confidence score using a weighted geometric mean. The method is available through an Adaptive Reasoning agent and works as follows: Given a set of training cases, for every case k , and every feature i , extract the case feature x_{ik} . Next, we compute the mean across all cases, $\bar{x}_i$ for feature i ,

$$\bar{x}_i = \frac{1}{n} \sum_{k=1}^n x_{ik}, \quad (2.4)$$

and the variance σ_i^2 ,

$$\sigma_i^2 = \frac{1}{n} \sum_{j=1}^n (x_{ik} - \bar{x}_i)^2. \quad (2.5)$$

The weight w_i for feature i can then be computed using the inverse variance:

$$w_i = \frac{1}{\sigma_i^2}. \quad (2.6)$$

The normalized weights W_i are then computed by dividing by the sum total of every

weight w_i :

$$W_i = \frac{w_i}{\sum_{i=1}^n w_i}. \quad (2.7)$$

With the means, variances and weights computed, the agent uses these values to evaluate new candidates against the learned distribution of the data. Thus, for a given candidate j , the sub-confidence given a feature value of x_{ij} , $conf_{ij}$, is computed as the density at point x_{ij} from a normal distribution, normalized by dividing by the max possible density of the distribution. This simplifies to:

$$conf_{ij}(x_{ij}, \bar{x}_i, \sigma_i^2) = e^{-\frac{(x_{ij} - \bar{x}_i)^2}{2\sigma_i^2}}, \quad (2.8)$$

to form a Gaussian membership function as shown in Figure 2.10.

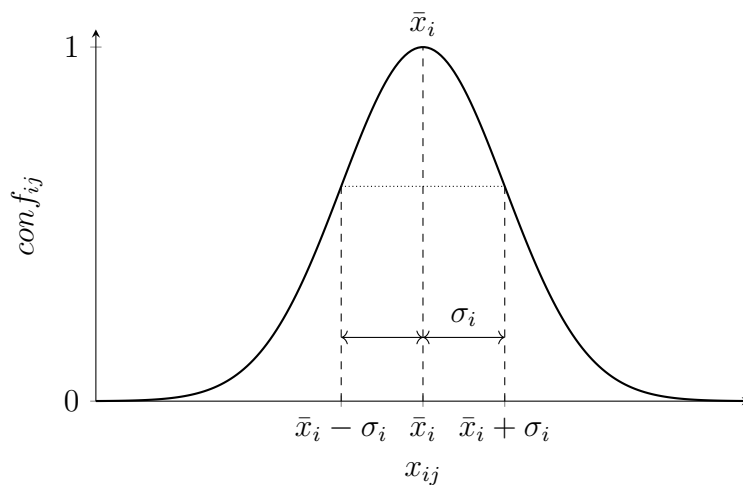


Figure 2.10: Adaptive reasoning sub-confidence function for feature i , shown as a normalized Gaussian-shaped confidence curve centered at the learned mean $\bar{x}_i$. The spread of the curve is governed by the standard deviation σ_i , with larger σ_i producing a broader confidence region and smaller σ_i producing a narrower peak.

The final confidence for candidate j is then computed using the weighted geometric mean

from all sub-confidences:

$$conf_j = \left(\prod_{i=1}^n (conf_{ij})^{W_i} \right)^{\frac{1}{\sum_{i=1}^n W_i}} . \quad (2.9)$$

In addition to the confidence scores, the agent also computes a z-score for every feature computed for candidate j . While this value does not contribute directly to the final confidence score, it provides a high level explanation to the user as to why a candidate received a score by providing the number of standard deviations of the computed feature from the learned feature mean:

$$Z_{ij} = \frac{x_{ij} - \bar{x}_i}{\sqrt{\sigma_i^2}} . \quad (2.10)$$

The Adaptive Reasoning method offers a novel approach to fuzzy membership functions in NeSy methods by enabling automatic construction of the Gaussian membership function. This eliminates the manual estimation of feature ranges to describe ambiguous features where anatomical information/relationships may not be immediately available. The weighting term also allows for the automatic categorization of feature importance, modulating feature impact on confidence. These characteristics allow the reasoning to scale with both added patient data and added feature types.

2.4.4 Candidate Selection

While these reasoning agents measure and compute confidences for the different candidates, they merely inform rather than actually select or eliminate candidates. This information is passed, in the form of a dictionary that includes each candidate’s confidence score, to a

candidate selector agent. Within this agent, a user can set a confidence threshold to determine which candidates are retained. Settings can be adjusted by manually inputting a confidence threshold or selecting candidates based on the relative max of the group. If more than one candidate is selected and a single one is desired, the user can also opt for selecting the largest candidate among the ones that meet the confidence threshold.

2.4.5 Summary

In this chapter we covered the infrastructure of the SimpleMind Software environment. SM allows for the integration of DL models, computer vision methods, and NeSy reasoning tools in the form of agents. The agent-based environment allows us to support DL segmentation models with cognitive AI to evaluate and select segmentation output candidates for anatomical consistency. Thus, ensuring outputs remain within anatomical constraints for trustworthy and explainable AI.

Chapter 3

Kidney Segmentation with Reasoning for Reducing CNN Segmentation Errors

3.1 Introduction

DL algorithms, including Convolutional Neural Network (CNN) and transformer-based architectures, have achieved state-of-the-art performance in medical image segmentation. Many works like nnUNet [51] and TotalSegmentator [118] demonstrate excellent Dice Similarity Coefficient (DSC) performance in various challenge sets and in-house datasets. However, despite strong average metrics, these models frequently produce false positive regions: either small extraneous components or, infrequently, grossly inaccurate segmentations; errors that

The contents of this chapter appeared in paper [81]

can be considered, by human standards, to be lacking basic common-sense reasoning and anatomy knowledge. Even if these errors occur on small percentage of cases, while not having significant impact on typical DL performance metrics like DSC, such errors undermine automation: every case must still be reviewed by an imaging analyst to detect and correct mistakes. In real-world clinical workflows, these issues hinder the implementation of fully automated, DL methods. Thus, current systems operate in a mode of semi-automation, rather than true autonomy which limits their clinical implementation.

3.1.1 Selective Segmentation with Rejection Option

Our goal is to move towards autonomous segmentation, where model outputs can be trusted without human checking of every case. To achieve this, we introduce the concept of Selective Segmentation with a Rejection Option. In this paradigm, the system evaluates each segmentation candidate and decides whether to accept or reject it. Accepted regions are passed forward without analyst review, while rejected cases are routed for manual inspection. In practice, this has the potential to reduce the number of cases requiring human attention, by ensuring that anatomically implausible results are not accepted.

A rejection mechanism must handle two primary error types seen in Figure 3.1: (a) extraneous components, where regions are segmented outside the organ of interest, and (b) inaccurate segmentations, where the target organ of interest is poorly segmented. Both error types can compromise downstream volumetric analysis and clinical decision support, and both are overlooked by conventional performance metrics.

In this experiment we make two primary contributions:

1. We propose a method for Selective Segmentation with Rejection Option, which can evaluate DL segmentation output candidates to autonomously accept anatomically-consistent candidates and reject extraneous or inaccurate candidates, thereby supporting fully autonomous segmentation in clinical workflows.
2. We introduce evaluation metrics tailored for selective segmentation, which better capture the systems effectiveness in reducing segmentation errors, beyond traditional metrics such as DSC.

These contributions address key limitations of existing DL segmentation systems and provide a framework for more dependable and efficient kidney segmentation.

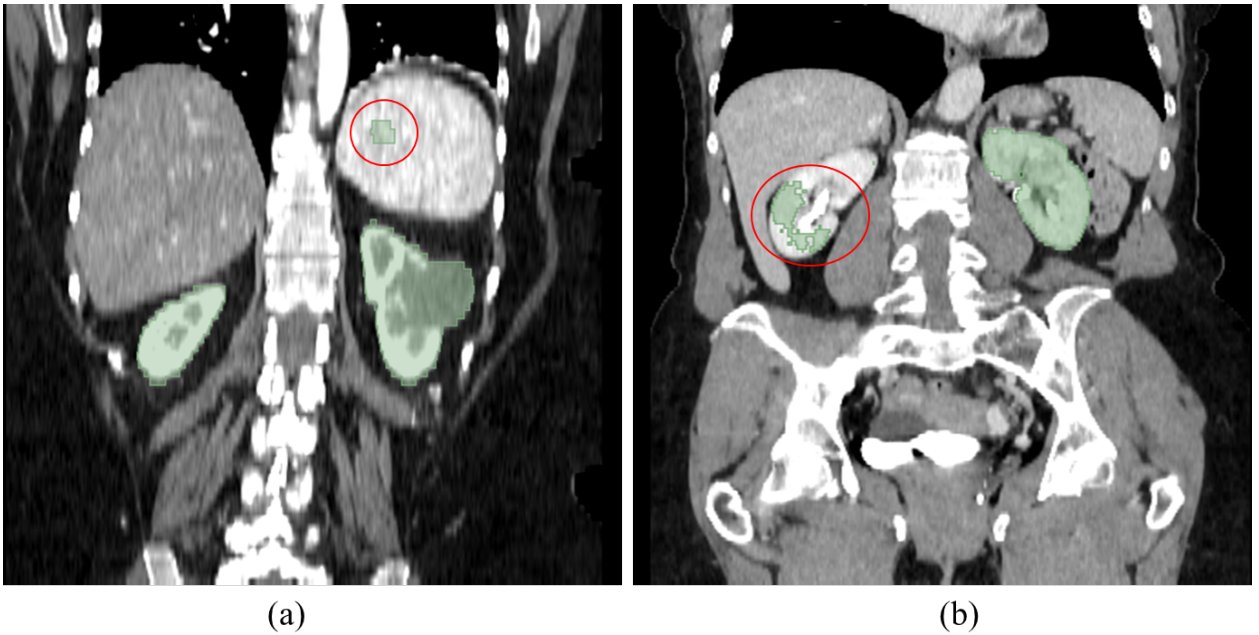


Figure 3.1: Image examples of DL segmentation errors highlighted in red circles. (a) An extraneous component segmented outside of the kidneys. (b) Inaccurate segmentation of a kidney. Visualization done using Slicer [28].

3.2 Methods

3.2.1 Dataset

We used the publicly available KiTS23 data [44], containing 489 contrast-enhanced CT images of kidneys, lesions and cysts with their corresponding masks. The Dataset was split into 80% for training with 5-fold cross-validation, and 20% as a hold-out test set. To generate challenging scenarios for selective segmentation, we artificially added Gaussian noise ($\sigma = 0.04$ or 4%) to the images as seen in Figure 3.2. These degraded images are used to evaluate the robustness of the DL segmentation algorithm and to induce false positives that can be detected and corrected via reasoning.



Figure 3.2: Example image of quality degradation from added noise ($\sigma = 0.04$). Visualization done using Slicer [28].

For our second experiment, we generated synthetic candidates from the original reference masks to test the SM reasoning algorithms against varying numbers of candidates. 3 new masks were generated for every image containing 1, 3, and 5 synthetic candidates added to

the reference mask (see figure 3.3). Candidates were generated by randomly copying one of the kidneys, randomly scaling each image dimension between 0.30 to 2, and shifting the location of its centroid to a random voxel within -70 to 300 HU. The new candidate was then checked for contact with previous candidates and the reference mask, if candidate contact was detected, the candidate was regenerated. This process was repeated until the desired number of candidates were added to the image without any contact between them.

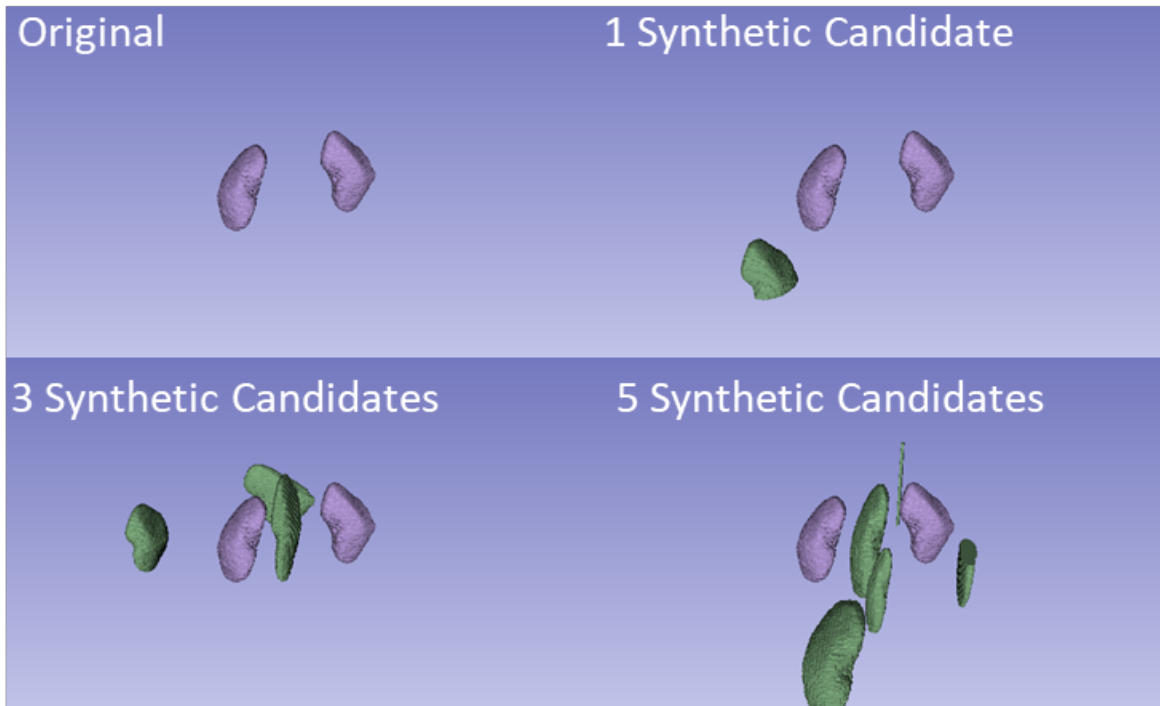


Figure 3.3: Examples of kidney masks with added synthetic candidates. Original kidney masks are shown in purple while synthetic candidates are shown in green. From left to right, top to bottom: Original kidney masks, 1 synthetic candidate added, 2 synthetic candidates added and 3 synthetic candidates added. Visualization performed using Slicer [28].

3.2.2 Kidney Segmentation Using SimpleMind

We expand on the previous work from Melendez-Corres et al [80] to incorporate nnUNet as the CNN-based kidney segmentation agent and additionally segment any lesions present in

the kidneys to form the Total Kidney Volume (TKV). The following subsections describe step-by-step the methods used. However, rather than a linear sequence of ad-hoc scripts, the different agents are incorporated into a single KG where SM's internal scheduling can automatically resolve each agent's dependencies and launch them as needed, whether in sequence or in parallel.

Abdomen, bone, and spine segmentation

Once the CT image is loaded into SM, the abdomen can be located via a thresholding agent set to threshold between -400 to 2000 HU. Various candidates can be generated for the abdomen, thus these are passed through FMR and candidate selection agents to select the best abdomen candidate based on total volume (two threshold linear confidence function with $x_0 = 1.0 \times 10^3 mm^3$ and $x_1 = 1.0 \times 10^7 mm^3$). Next, bone voxels are segmented via the thresholding agent, including voxels between 200 – 2000 HU. Voxels located outside the abdomen are removed via a logical *and* operation between the initial bone mask and previously identified abdomen mask. Simultaneously, a spine region is defined using a spatial offset agent to create a bounding box in the posterior region of the abdomen: with the centroid of the abdomen as a reference point, the region is generated at -50 to 50 mm in the x coordinate and -100 to 20 mm in the y coordinate (RAS coordinate system). Another *and* operation is then performed between the spine region and the bone mask to obtain the spine. Similar to the abdomen, many candidates can be generated so these also go through FMR. The confidence function for the spine candidates is determined via two features: x-offset relative to abdomen centroid (thresholds: $x_0 = -60$ mm, $x_1 = -50$ mm, $x_2 = 50$ mm and $x_3 = 60$ mm) and posterior distance relative to the abdomen centroid (thresholds: $x_0 = -20$

mm, $x_1 = 100$ mm). The largest of the candidates with the highest confidence score is then selected as the final spine mask. An illustration of the KG can be seen in Appendix A, Figure A.1.

Kidney and lesion segmentation using nnUNet

As described in 2.2.1 SM incorporates an nnUNet [51] agent. It was trained on the KiTS23 data to segment the kidneys and their respective lesions and cysts. The default fingerprint settings for the 3D architecture were used with no modifications. After the nnUNet inference, agents were used to separate the different labels of the masks and a logical *or* operation was used to combine the kidneys, lesions, and any cysts into a single TKV mask. An illustration of the KG can be seen in Appendix A, Figure A.2.

Kidney segmentation refinement using reasoning

Kidney candidates produced by nnUNet were evaluated and selected using 4 different methods within a single SM KG. We compared SM’s FMR, Adaptive Reasoning (AR), Largest Connected Component Analysis (LCCA), and Largest Connected Component Analysis Per Side (LCCAPS). Using the spine’s centroid as the reference landmark, the right and left kidneys can be identified using FMR by computing distances to the right and left of the spine, posterior distance to the spine, and volume. Thus, for each kidney, the FMR confidence functions consist of thresholds for volume: $x_0 = 5.0 \times 10^4 mm^3$, $x_1 = 1.0 \times 10^5 mm^3$, $x_2 = 2.5 \times 10^5 mm^3$, and $x_3 = 5.0 \times 10^5 mm^3$; right/left distance (marked as essential): $x_0 = 10$ mm, $x_1 = 40$ mm, $x_2 = 80$ mm and $x_3 = 200$ mm; posterior distance: $x_0 = -50$ mm, $x_1 = 10$ mm, $x_2 = 20$ mm and $x_3 = 30$ mm. The best candidate for each kidney is then selected as

the largest among those meeting the confidence threshold of 0.5. AR and LCCAPS also use FMR, however, only the right/left distance functions are used to split candidates into right and left. In LCCAPS, the candidate selection for each kidney selects the largest candidate with a confidence threshold greater than or equal to 0.01, i.e, the single largest candidate on each side with a non-zero confidence score. In the case of AR, rather than selecting a single candidate per side, all non-zero confidence candidates for each side are preserved. Then, for each side, the candidates are reevaluated with the AR agent, using means, variances and weights computed from the training data, against the learned distribution of the following features: volume, right/left distance to the spine and posterior distance to the spine. The final candidate for right/left kidney is selected as the largest candidate meeting the confidence threshold of 0.5. For LCCA, we used a simple FMR with just the volume feature with thresholds set $x_0 = 0 \text{ mm}^3$, $x_1 = 1 \text{ mm}^3$, thus any candidate with any existing volume will score a confidence of 1. The candidate selection agent is then set to use a confidence threshold of 0 and select the single largest among these. An illustration of the KG can be seen in Appendix A, Figure A.3.

3.2.3 Analysis

The comparisons of the methods were done via 5-fold cross validation using the folds generated by nnUNet. While the task generates segmentation masks of the kidneys, our goal is to evaluate how the different methods can correctly identify the kidneys with minimal anatomical errors, thus we evaluate the methods as a detection task, where a true positive is defined as a connected component/candidate with a $\text{DSC} \geq 0.80$ with any of the reference truth

kidneys. Due to no presence of negative samples, a true negative count of performance is not possible, thus we compute False Positive Rate (FPR), False Negative Rate (FNR), and a rejection rate; the FPR being computed as FP/total candidates for a FPR computed across all the candidates produced. The rejection rate refers to the rate of cases where the method did not provide an output, thus it is computed via unanswered cases/total cases. Statistically significant differences in FPR and FNR between CNN, LCCA, LCCAPS, FMR, and AR were assessed via a global Pearson’s Chi-square test with α at 0.05. Pairwise comparisons were done using two-proportion Z-test with α at 0.05 with p -values adjusted using the Benjamini-Hochberg (BH) correction.

In addition, we also analyzed how the individual segmentation refinement methods are able to refine the base CNN segmentation. For every candidate generated by the CNN, we evaluate whether the candidate was correctly accepted ($DSC \geq 0.80$) or rejected ($DSC < 0.80$). Thus, analogous to the detection task, we compute the following: True Acceptance Rate (TAR), True Rejection Rate (TRR), False Acceptance Rate (FAR), and False Rejection Rate (FRR). These metrics are defined as:

$$TAR = \frac{TrueAcceptance}{TrueAcceptance + FalseRejection}, \quad (3.1)$$

$$FAR = \frac{FalseAcceptance}{FalseAcceptance + TrueRejection}, \quad (3.2)$$

$$TRR = \frac{TrueRejection}{TrueRejection + FalseAcceptance} = 1 - FAR, \quad (3.3)$$

and

$$FRR = \frac{FalseRejection}{TrueAcceptance + FalseRejection} = 1 - TAR. \quad (3.4)$$

statistically significant differences in TAR, FAR, TRR and FRR between LCCA, LCCAPS, FMR, and AR were also assessed via a global Pearson’s Chi-square test with α at 0.05 and pairwise comparisons using two-proportion Z-test with α at 0.05 and p -values adjusted using the BH correction. An illustration of the difference between the FPR and FAR metrics is shown in Figure 3.4. This analysis was also applied to the synthetic candidates experiments. Although, given the synthetic candidate masks were derived from the original masks, no DSC thresholding was necessary.

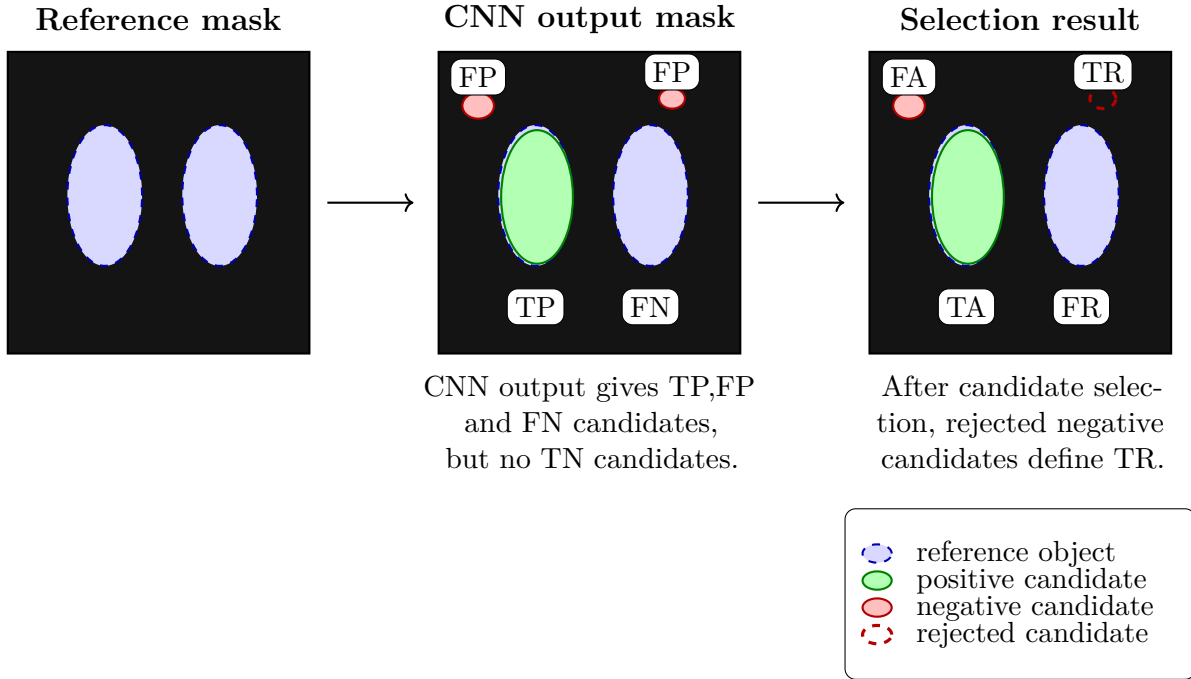


Figure 3.4: Illustration of the distinction between false positives from CNN candidate generation and false acceptances during candidate selection. The CNN output mask defines the candidate sample set, producing true positives, false positives, and false negatives, but no true negative candidates. After candidate selection, negative candidates may be accepted or rejected, allowing false acceptance rate to be computed as $FAR = FA/(FA + TR)$.

3.3 Results

3.3.1 Noise experiment: Kidney Segmentation Detection Analysis

Table 3.1 summarizes kidney segmentation detection performance for the different methods. The baseline CNN segmentation produced a mean FPR of 0.33 across folds, even at zero added noise. All segmentation refinement methods effectively reduce the FPR, with AR achieving the lowest FPR overall, albeit, with higher FNR compared to FMR and LCCAPS. LCCA also achieves a low FPR, but at the cost of a higher FNR, indicating rejection of valid candidates.

At 4% added noise, DL performance degrades as expected, increasing both FPR and FNR. All methods maintain their relative trends: AR consistently achieves the lowest FPR, but increased rejection rates. Figure 3.5 illustrates the performance metrics across methods and noise levels.

Table 3.1: Mean and standard deviation of kidney detection performance comparison of methods for validation data.

Noise %	Method	FPR $\pm \sigma$	FNR $\pm \sigma$	Rejection Rate. $\pm \sigma$
0%	CNN	0.33 ± 0.04	0.01 ± 0.01	0.00 ± 0.00
	LCCA	0.02 ± 0.02	0.50 ± 0.01	0.00 ± 0.00
	LCCAPS	0.02 ± 0.01	0.01 ± 0.01	0.00 ± 0.00
	FMR	0.01 ± 0.02	0.10 ± 0.02	0.01 ± 0.01
	AR	0.01 ± 0.01	0.12 ± 0.04	0.08 ± 0.03
4%	CNN	0.81 ± 0.07	0.16 ± 0.05	0.01 ± 0.01
	LCCA	0.13 ± 0.05	0.56 ± 0.03	0.02 ± 0.01
	LCCAPS	0.14 ± 0.03	0.15 ± 0.04	0.02 ± 0.01
	FMR	0.12 ± 0.03	0.17 ± 0.03	0.06 ± 0.03
	AR	0.10 ± 0.03	0.23 ± 0.05	0.12 ± 0.01

For the hold-out test set performance, Table 3.2 shows the FPR, FNR and rejection rate

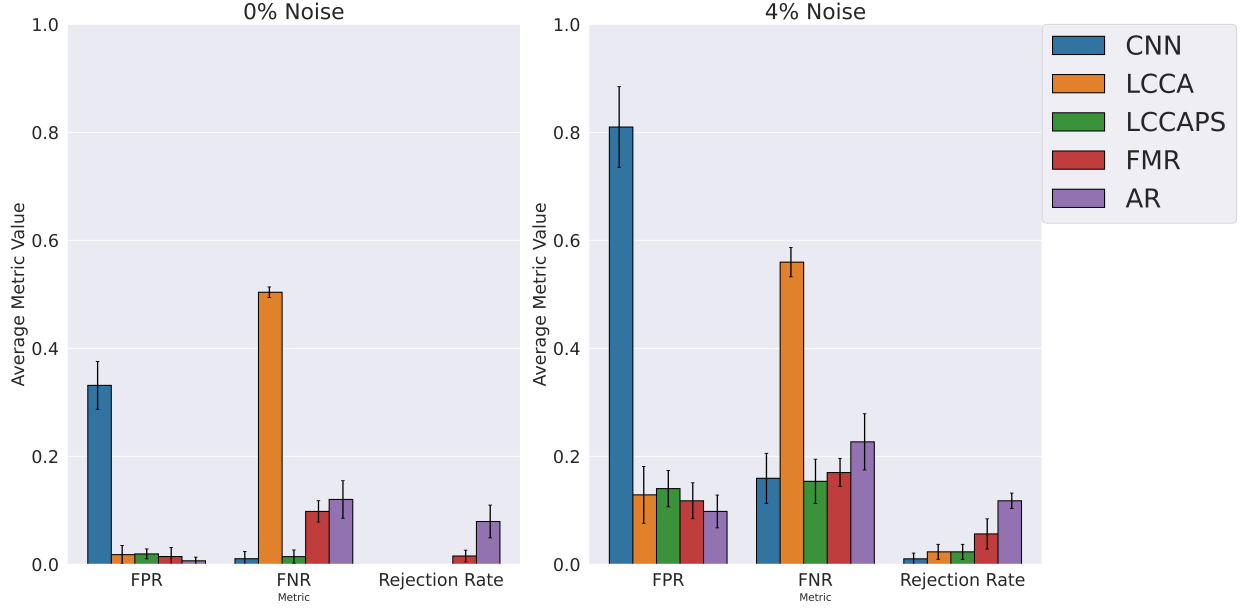


Figure 3.5: Bar charts of average kidney detection performance comparison across noise levels on validation data.

on the test data for all methods (CNN, LCCA, LCCAPS, FMR, and AR) and Figure 3.6 barcharts provide a visual comparison of the detection metrics between the methods across noise levels. Test data results followed the trends observed in the cross-validation results, with FPR being the highest using the base CNN and the highest FNR observed using the LCCA method. The global Pearson’s Chi-square test returned a significant difference between the methods for FPR at 0% noise ($p < 0.001$), at 4% noise ($p < 0.001$) and FNR, for 0% noise and 4% noise ($p < 0.001$), FNR at 0% ($p < 0.001$) and 4% noise ($p < 0.001$).

Pairwise comparisons show statistically significant differences in FPR between the CNN and all the segmentation refinement methods (BH-corrected $p < 0.001$) across both noise levels. No statistically significant differences in FPR were found when comparing the segmentation refinement methods. Statistically significant differences in FNR were found when comparing: CNN vs FMR (BH-corrected $p < 0.001$), CNN vs AR (BH-corrected $p < 0.001$), CNN vs

Table 3.2: Kidney detection performance comparison across noise levels for test data.

Noise %	Method	FPR	FNR	Rejection Rate
0%	CNN	0.447	0.016	0.000
	LCCA	0.041	0.510	0.000
	LCCAPS	0.026	0.021	0.000
	FMR	0.017	0.096	0.020
	AR	0.013	0.134	0.071
4%	CNN	0.558	0.172	0.000
	LCCA	0.186	0.584	0.010
	LCCAPS	0.150	0.163	0.010
	FMR	0.120	0.181	0.020
	AR	0.115	0.247	0.102

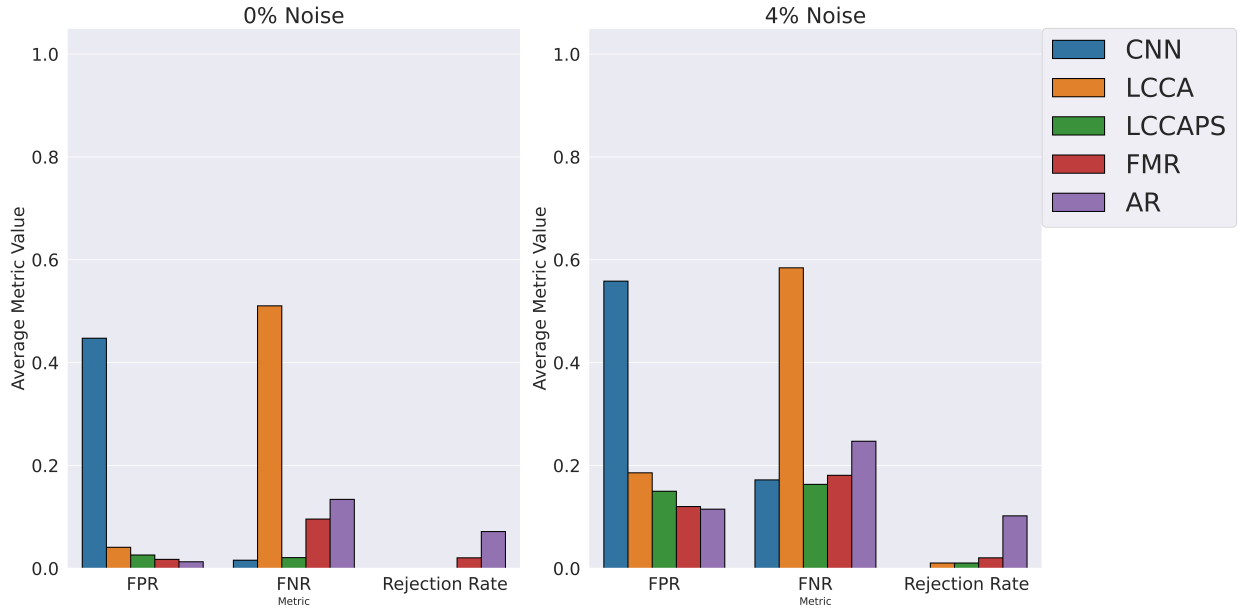


Figure 3.6: Bar charts of kidney detection performance comparison across noise levels on test data.

LCCA (BH-corrected $p < 0.001$), LCCA vs AR (BH-corrected $p < 0.001$), LCCA vs FMR (BH-corrected $p < 0.001$), LCCAPS vs AR (BH-corrected $p < 0.001$), LCCA vs LCCAPS (BH-corrected $p < 0.001$) at 0% noise, and LCCAPS vs FMR (BH-corrected $p = 0.002$). At 4% noise we found statistically significant differences in FNR when comparing: CNN vs

LCCA (BH-corrected $p < 0.001$), LCCA vs AR (BH-corrected $p < 0.001$), LCCA vs FMR (BH-corrected $p < 0.001$), and LCCA vs LCCAPS (BH-corrected $p < 0.001$). Raw p -values and BH p -values for all pairwise comparisons can be seen in Table 3.3.

Table 3.3: Pairwise Statistical Comparison Results for FPR and FNR at 0% and 4% Noise

Comparison	0% Noise		4% Noise	
	Raw p	BH p	Raw p	BH p
FPR				
CNN vs LCCAPS	< 0.001	< 0.001	< 0.001	< 0.001
CNN vs FMR	< 0.001	< 0.001	< 0.001	< 0.001
CNN vs AR	< 0.001	< 0.001	< 0.001	< 0.001
CNN vs LCCA	< 0.001	< 0.001	< 0.001	< 0.001
LCCA vs AR	0.150	0.300	0.122	0.232
LCCA vs FMR	0.242	0.403	0.139	0.232
LCCAPS vs AR	0.381	0.545	0.353	0.486
LCCA vs LCCAPS	0.487	0.609	0.437	0.486
LCCAPS vs FMR	0.576	0.640	0.409	0.486
FMR vs AR	0.732	0.732	0.887	0.887
FNR				
CNN vs LCCAPS	0.703	0.703	0.820	0.820
CNN vs FMR	< 0.001	< 0.001	0.818	0.820
CNN vs AR	< 0.001	< 0.001	0.076	0.127
CNN vs LCCA	< 0.001	< 0.001	< 0.001	< 0.001
LCCA vs AR	< 0.001	< 0.001	< 0.001	< 0.001
LCCA vs FMR	< 0.001	< 0.001	< 0.001	< 0.001
LCCAPS vs AR	< 0.001	< 0.001	0.047	0.094
LCCA vs LCCAPS	< 0.001	< 0.001	< 0.001	< 0.001
LCCAPS vs FMR	0.002	0.002	0.649	0.811
FMR vs AR	0.249	0.277	0.124	0.177

3.3.2 Noise experiment: Kidney Candidates Acceptance & Rejection Analysis

Table 3.4 summarizes the performance of candidate-level acceptance and rejection metrics on the cross-validation data. These metrics reflect the system’s ability to selectively accept

valid candidates and reject false positives, aligning directly with the goal of autonomous segmentation. In essence, how well each method is correcting the base CNN output.

At 0% noise, all methods provide FARs below 0.05, with AR providing the lowest FAR, albeit, with higher FRR compared to the other methods. LCCA achieves the lowest FAR at 4% noise, but at the cost of a higher FRR, meaning it frequently rejects valid candidates that would be considered useful compared to the other methods. Figure 3.7 shows the candidate evaluation metrics across methods and noise levels.

Table 3.4: Mean candidate acceptance and rejection performance comparison of methods.

Noise %	Method	FAR $\pm \sigma$	FRR $\pm \sigma$	TAR $\pm \sigma$	TRR $\pm \sigma$
0%	LCCA	0.02 $\pm$ 0.02	0.50 $\pm$ 0.01	0.98 $\pm$ 0.02	0.50 $\pm$ 0.01
	LCCAPS	0.04 $\pm$ 0.03	0.00 $\pm$ 0.01	0.96 $\pm$ 0.03	1.00 $\pm$ 0.01
	FMR	0.03 $\pm$ 0.04	0.10 $\pm$ 0.01	0.97 $\pm$ 0.04	0.90 $\pm$ 0.01
	AR	0.01 $\pm$ 0.01	0.18 $\pm$ 0.02	0.99 $\pm$ 0.01	0.82 $\pm$ 0.02
4%	LCCA	0.02 $\pm$ 0.00	0.48 $\pm$ 0.01	0.98 $\pm$ 0.00	0.52 $\pm$ 0.01
	LCCAPS	0.04 $\pm$ 0.01	0.01 $\pm$ 0.00	0.96 $\pm$ 0.01	0.99 $\pm$ 0.00
	FMR	0.03 $\pm$ 0.01	0.06 $\pm$ 0.02	0.97 $\pm$ 0.01	0.94 $\pm$ 0.02
	AR	0.02 $\pm$ 0.01	0.18 $\pm$ 0.03	0.98 $\pm$ 0.01	0.82 $\pm$ 0.03

Test set performance followed the same trends as seen in the cross-validation. Test data FAR, FRR, TAR, and TRR across both noise levels can be seen in Table 3.5. Figure 3.8 provides barcharts to compare the metrics across the methods. Global Pearson’s Chi-Squared returned statistically significant differences in TAR and FRR ($p < 0.001$) between the methods, but no statistically significant differences in TRR and FAR were found ($p = 0.27$). Given this, we performed our pairwise test comparisons between methods for TAR and FRR. Statistically significant differences in TAR and FRR were found for all pairs across both noise levels. Table 3.6 shows the raw p -values and BH p -values for every pairwise comparison.

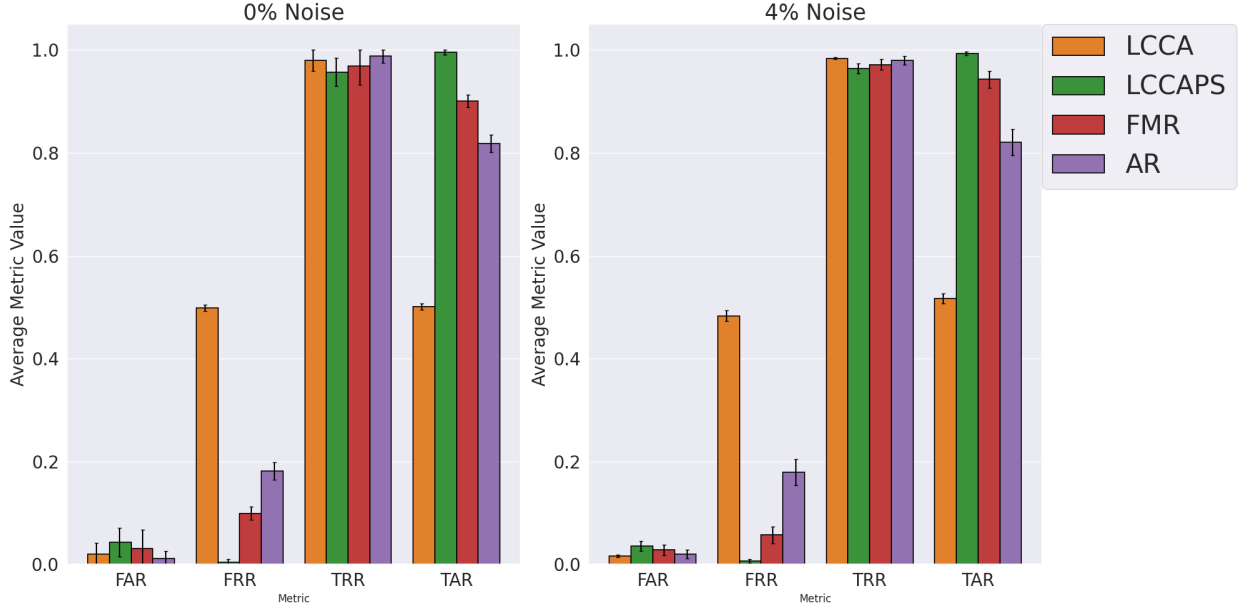


Figure 3.7: Bar charts of average candidate analysis performance comparison across noise levels on validation data.

Table 3.5: Candidate acceptance and rejection performance comparison of methods on test data.

Noise %	Method	FAR	FRR	TAR	TRR
0%	LCCA	0.026	0.503	0.497	0.974
	LCCAPS	0.033	0.005	0.995	0.967
	FMR	0.020	0.101	0.899	0.980
	AR	0.013	0.180	0.820	0.987
4%	LCCA	0.090	0.503	0.497	0.910
	LCCAPS	0.139	0.000	1.000	0.861
	FMR	0.104	0.031	0.969	0.896
	AR	0.085	0.176	0.824	0.915

3.3.3 Synthetic Candidate Experiment: Kidney Candidates Acceptance & Rejection Analysis

Candidate acceptance and rejection performance on validation data for the synthetic candidate experiment is summarized in Table 3.7. This experiment evaluates performance as increasing

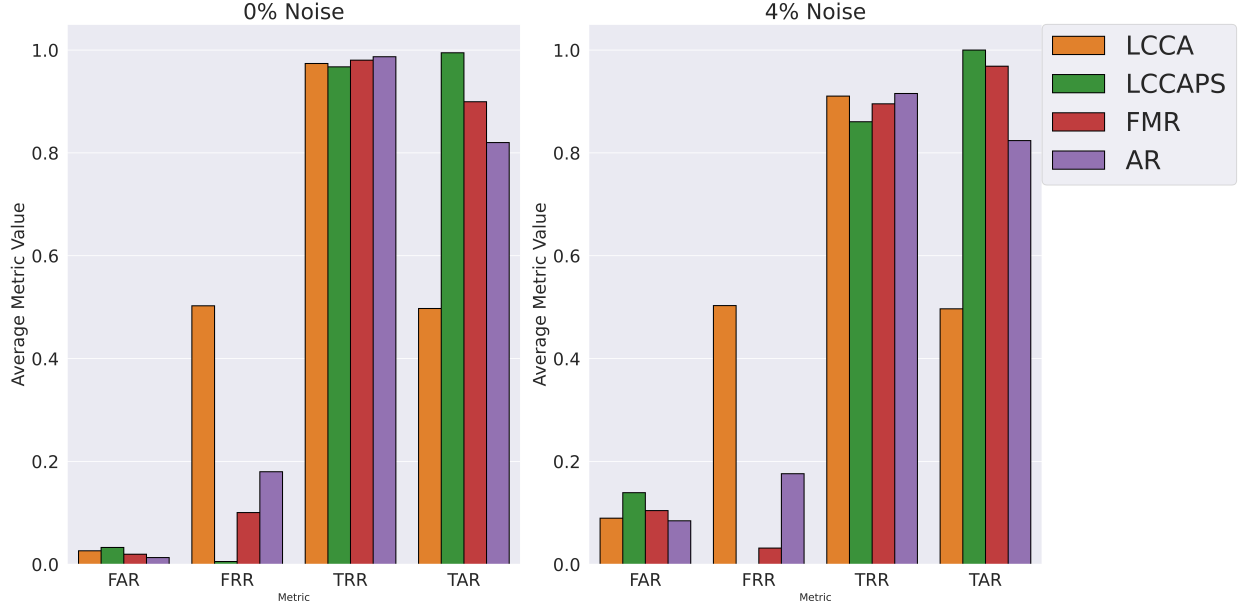


Figure 3.8: Bar charts of candidate analysis performance comparison across noise levels on test data.

numbers of false synthetic candidates are added to the candidate pool. Across the validation data, AR had the lowest FAR for all synthetic candidate levels, with values of 0.015, 0.009, and 0.018 for 1, 3, and 5 synthetic candidates, respectively. AR also had the highest TRR across all synthetic candidate levels, with values of 0.985, 0.991, and 0.982.

For valid candidate acceptance, FMR and AR had more consistent TAR values across increasing numbers of synthetic candidates compared with the connected-component methods. FMR had validation TARs of 0.814, 0.722, and 0.650 for 1, 3, and 5 synthetic candidates, respectively, while AR had corresponding TARs of 0.817, 0.811, and 0.797. In comparison, LCCA had TARs of 0.330, 0.195, and 0.115, while LCCAPS had TARs of 0.802, 0.556, and 0.407. Figure 3.9 shows the candidate evaluation metrics across methods and number of synthetic candidates.

Test set results are summarized in Table 3.8. Similar to the validation data, AR had

Table 3.6: Pairwise Statistical Comparison Results for TAR and FRR at 0% and 4% Noise

Comparison	0% Noise		4% Noise	
	Raw p	BH p	Raw p	BH p
TAR				
LCCA vs LCCAPS	< 0.001	< 0.001	< 0.001	< 0.001
LCCA vs FMR	< 0.001	< 0.001	< 0.001	< 0.001
LCCA vs AR	< 0.001	< 0.001	< 0.001	< 0.001
LCCAPS vs FMR	< 0.001	< 0.001	0.024	0.024
LCCAPS vs AR	< 0.001	< 0.001	< 0.001	< 0.001
FMR vs AR	0.026	0.026	< 0.001	< 0.001
FRR				
LCCA vs LCCAPS	< 0.001	< 0.001	< 0.001	< 0.001
LCCA vs FMR	< 0.001	< 0.001	< 0.001	< 0.001
LCCA vs AR	< 0.001	< 0.001	< 0.001	< 0.001
LCCAPS vs FMR	< 0.001	< 0.001	0.024	0.024
LCCAPS vs AR	< 0.001	< 0.001	< 0.001	< 0.001
FMR vs AR	0.026	0.026	< 0.001	< 0.001

Table 3.7: Mean candidate acceptance and rejection performance comparison of methods across number of synthetic candidates.

No. Cands	Method	FAR $\pm \sigma$	FRR $\pm \sigma$	TAR $\pm \sigma$	TRR $\pm \sigma$
1	LCCA	0.347 ± 0.051	0.670 ± 0.028	0.330 ± 0.028	0.653 ± 0.051
	LCCAPS	0.398 ± 0.049	0.198 ± 0.028	0.802 ± 0.028	0.602 ± 0.049
	FMR	0.176 ± 0.025	0.186 ± 0.034	0.814 ± 0.034	0.824 ± 0.025
	AR	0.015 ± 0.014	0.183 ± 0.027	0.817 ± 0.027	0.985 ± 0.014
3	LCCA	0.204 ± 0.022	0.805 ± 0.034	0.195 ± 0.034	0.796 ± 0.022
	LCCAPS	0.297 ± 0.021	0.444 ± 0.028	0.556 ± 0.028	0.703 ± 0.021
	FMR	0.133 ± 0.008	0.278 ± 0.019	0.722 ± 0.019	0.867 ± 0.008
	AR	0.009 ± 0.006	0.189 ± 0.019	0.811 ± 0.019	0.991 ± 0.006
5	LCCA	0.154 ± 0.007	0.885 ± 0.019	0.115 ± 0.019	0.846 ± 0.007
	LCCAPS	0.239 ± 0.017	0.593 ± 0.044	0.407 ± 0.044	0.761 ± 0.017
	FMR	0.124 ± 0.009	0.350 ± 0.028	0.650 ± 0.028	0.876 ± 0.009
	AR	0.018 ± 0.004	0.203 ± 0.022	0.797 ± 0.022	0.982 ± 0.004

the lowest FAR across all synthetic candidate levels, with values of 0.010, 0.010, and 0.018 for 1, 3, and 5 synthetic candidates, respectively. AR also had the highest TRR across all synthetic candidate levels, with values of 0.990, 0.990, and 0.982. FMR had test TARs of 0.839, 0.688, and 0.625 across 1, 3, and 5 synthetic candidates, while AR had corresponding

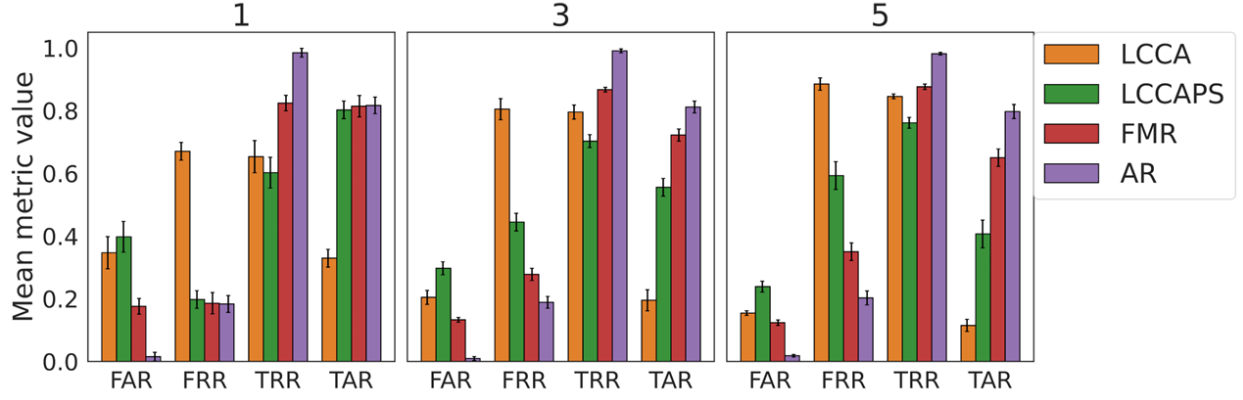


Figure 3.9: Bar charts of average candidate analysis performance comparison across number of added synthetic candidates on validation data.

TARs of 0.812, 0.812, and 0.786. The connected-component methods showed lower TARs as the number of synthetic candidates increased, with LCCA decreasing from 0.365 to 0.156 and LCCAPS decreasing from 0.839 to 0.385. Figure 3.10 provides barcharts comparing the test-set metrics across methods.

Table 3.8: Candidate acceptance and rejection performance comparison of methods across number of synthetic candidates on test data.

No. Cands	Method	FAR	FRR	TAR	TRR
1	LCCA	0.286	0.635	0.365	0.714
	LCCAPS	0.337	0.161	0.839	0.663
	FMR	0.133	0.161	0.839	0.867
	AR	0.010	0.188	0.812	0.990
3	LCCA	0.204	0.802	0.198	0.796
	LCCAPS	0.296	0.438	0.562	0.704
	FMR	0.163	0.312	0.688	0.837
	AR	0.010	0.188	0.812	0.990
5	LCCA	0.139	0.844	0.156	0.861
	LCCAPS	0.249	0.615	0.385	0.751
	FMR	0.133	0.375	0.625	0.867
	AR	0.018	0.214	0.786	0.982

Global Pearson’s Chi-Squared tests found statistically significant differences between

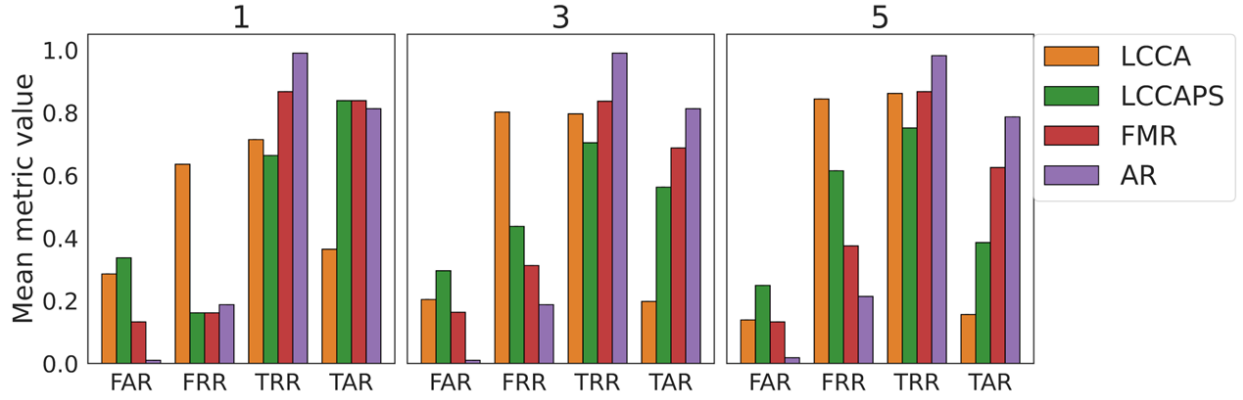


Figure 3.10: Bar charts of average candidate analysis performance comparison across number of added synthetic candidates on test data.

methods for all metrics: TAR, FRR, FAR, and TRR ($p < 0.001$). Therefore, pairwise comparisons were performed for all four metrics. Table 3.9 shows the raw and BH-adjusted p -values for each pairwise comparison.

For TAR, LCCA differed significantly from all other methods at each synthetic candidate level. At 1 synthetic candidate, no significant differences were observed between AR, FMR, and LCCAPS. At 3 and 5 synthetic candidates, all pairwise TAR comparisons were statistically significant after BH correction.

For FAR, AR differed significantly from all other methods across all synthetic candidate levels. FMR and LCCAPS also differed significantly across all candidate levels. The comparison between LCCA and LCCAPS was not significant at 1 synthetic candidate but was significant at 3 and 5 synthetic candidates. The comparison between LCCA and FMR was significant at 1 synthetic candidate but not significant at 3 or 5 synthetic candidates.

Table 3.9: Pairwise Statistical Comparison Results Across All Candidate Selection Metrics and Number of Synthetic Candidates

Comparison	1 Candidate		3 Candidates		5 Candidates	
	Raw p	BH p	Raw p	BH p	Raw p	BH p
TAR						
LCCA vs LCCAPS	< 0.001	< 0.001	< 0.001	< 0.001	< 0.001	< 0.001
LCCA vs FMR	< 0.001	< 0.001	< 0.001	< 0.001	< 0.001	< 0.001
LCCA vs AR	< 0.001	< 0.001	< 0.001	< 0.001	< 0.001	< 0.001
LCCAPS vs FMR	1.000	1.000	0.011	0.011	< 0.001	< 0.001
LCCAPS vs AR	0.501	0.602	< 0.001	< 0.001	< 0.001	< 0.001
FMR vs AR	0.501	0.602	0.005	0.006	< 0.001	< 0.001
TRR						
LCCA vs LCCAPS	0.440	0.440	0.010	0.012	< 0.001	< 0.001
LCCA vs FMR	0.008	0.010	0.201	0.201	0.780	0.780
LCCA vs AR	< 0.001	< 0.001	< 0.001	< 0.001	< 0.001	< 0.001
LCCAPS vs FMR	< 0.001	0.001	< 0.001	< 0.001	< 0.001	< 0.001
LCCAPS vs AR	< 0.001	< 0.001	< 0.001	< 0.001	< 0.001	< 0.001
FMR vs AR	< 0.001	0.001	< 0.001	< 0.001	< 0.001	< 0.001
FAR						
LCCA vs LCCAPS	0.440	0.440	0.010	0.012	< 0.001	< 0.001
LCCA vs FMR	0.008	0.010	0.201	0.201	0.780	0.780
LCCA vs AR	< 0.001	< 0.001	< 0.001	< 0.001	< 0.001	< 0.001
LCCAPS vs FMR	< 0.001	0.001	< 0.001	< 0.001	< 0.001	< 0.001
LCCAPS vs AR	< 0.001	< 0.001	< 0.001	< 0.001	< 0.001	< 0.001
FMR vs AR	< 0.001	0.001	< 0.001	< 0.001	< 0.001	< 0.001
FRR						
LCCA vs LCCAPS	< 0.001	< 0.001	< 0.001	< 0.001	< 0.001	< 0.001
LCCA vs FMR	< 0.001	< 0.001	< 0.001	< 0.001	< 0.001	< 0.001
LCCA vs AR	< 0.001	< 0.001	< 0.001	< 0.001	< 0.001	< 0.001
LCCAPS vs FMR	1.000	1.000	0.011	0.011	< 0.001	< 0.001
LCCAPS vs AR	0.501	0.602	< 0.001	< 0.001	< 0.001	< 0.001
FMR vs AR	0.501	0.602	0.005	0.006	< 0.001	< 0.001

3.4 Discussion

This study introduced selective segmentation with a rejection option as a framework for advancing beyond semi-automation toward autonomous segmentation. Current DL kidney segmentation models achieve high Dice scores, yet still produce small extraneous false

positives or occasional gross inaccuracies that require human analysts to review every case. Our results demonstrate that candidate evaluation and reasoning methods implemented in the SM framework can substantially reduce false positives, thereby improving the reliability of DL-based segmentation and enabling the possibility of autonomous workflows. Figure 3.11 shows an example of the AR method removing the errors described in section 3.1 and Figure 3.1.

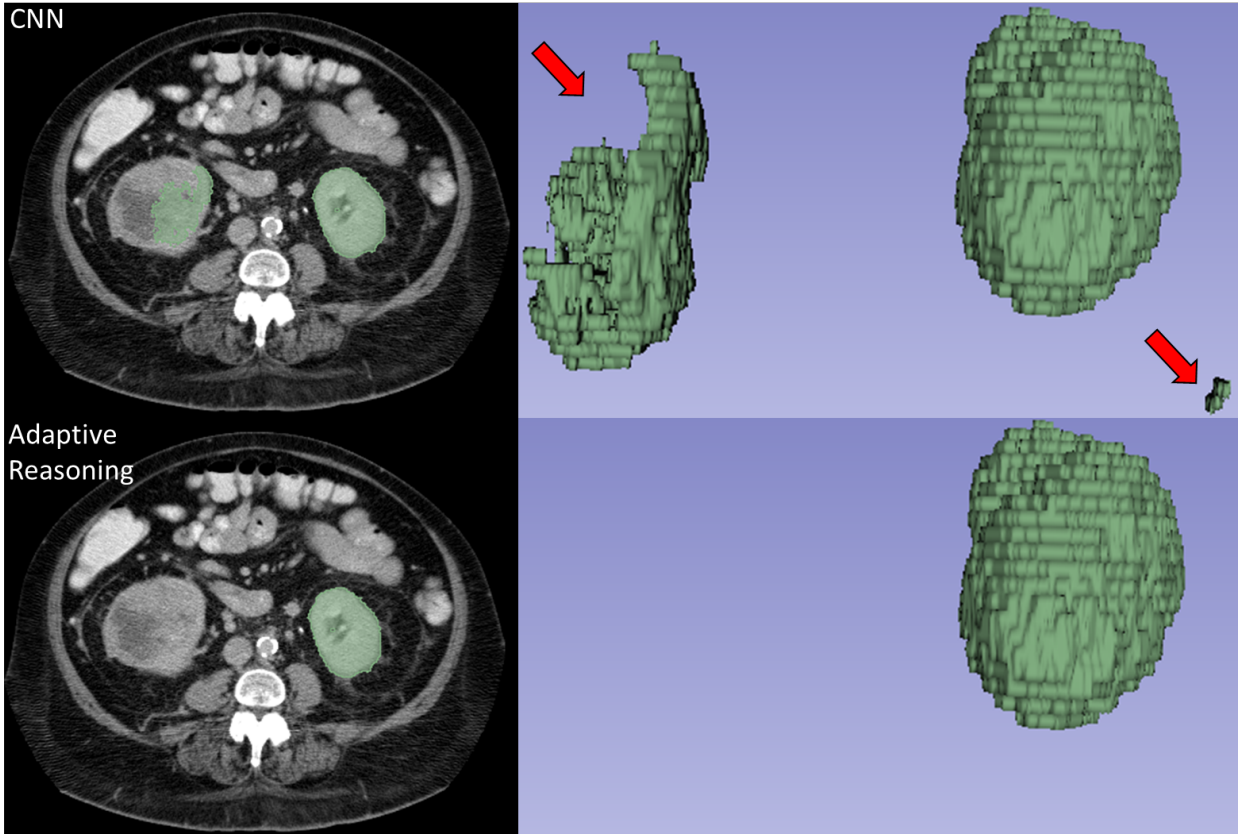


Figure 3.11: Example image demonstrating the AR method removing erroneous candidates generated by the CNN. Red arrows highlight the two types of errors mentioned in Section 3.1 and Figure 3.1: Inaccurate segmentation and an extraneous component.

Among the segmentation refinement approaches, the widely used LCCA significantly reduced false positives produced by the CNN, but at the expense of missing valid kidneys, since it is constrained to selecting the single largest candidate. The addition of simple

reasoning enhancements such as LCCAPS, which selects the largest candidate on each side of the spine, mitigate this limitation and provide a more balanced performance. The reasoning-based methods, FMR and AR, while performing comparable to LCCAPS in this task, can potentially improve performance by virtue of being more dynamic and tunable; capable of evaluating candidates against anatomical priors such as size, location, and spatial relationships and potentially any additional features a user desires. AR offers the additional advantage of automatically fitting features to training data, eliminating the need for manual threshold setting and making it more scalable in terms of image data and computable features. In contrast, LCCA and LCCAPS are not a tunable or trainable methods and have no varying confidence scores that can be used as thresholds, only simple selection of the largest candidate or binary checks against anatomical priors in the case of LCCAPS. The reasoning-based methods can compute continuous confidence functions, therefore, a user can adjust their confidence function and candidate selection thresholds to control, to an extent, for their desired FNR and rejection rates. The lack of finetuning the confidence threshold, can explain the higher FNR and FRR of the FMR and AR methods. A confidence threshold of 0.5 is potentially too strict, inadvertently eliminating adequate kidney candidates. Ultimately, however, the decision of the most optimal method will be task-dependent, with more complex, multi-organ tasks requiring the more dynamic reasoning methods.

Nonetheless, the synthetic candidate experiments demonstrate the potential superior performance of FMR and AR in a more complex scenario where the kidneys need to be identified against multiple candidates. The difference in performance from FMR and AR compared to the connect-components methods increased as more candidates were added. Thus, in more complex cases where the based segmentation method consistently produces multiple

segmentation candidates (such as a DL method trained with limited data, thresholding-based methods, or multi-organ segmentation), FMR and AR provide superior discrimination for segmentation acceptance/rejection.

Finally, the evaluation metrics introduced here provide a more appropriate measure of usefulness for autonomous segmentation than traditional overlap scores like DSC. These metrics better reflect the practical goal: to accept valid segmentations without human intervention while rejecting erroneous ones that would otherwise require correction.

3.5 Conclusion

We presented a method for selective segmentation with a rejection option, implemented in the SimpleMind framework, and applied it to kidney segmentation in CT. Our results show that reasoning-based candidate evaluation can effectively reduce false positives that are inconsistent with common-sense anatomical reasoning.

This work demonstrates that simple anatomical features are sufficient to enable selective acceptance and rejection of candidates, opening a pathway toward fully autonomous segmentation systems that self-check their outputs and only route uncertain cases for human review. Future work will explore varying combinations of FMR, AR, and other methods, with more added features, to improve discrimination capabilities, and fine-tuning confidence thresholds. We will also explore complementary methods for reducing false negatives in such cases where a base DL method fails to locate the target organ or only partially segments it.

Chapter 4

Deployment of the SimpleMind Kidney Segmentation Model for Research and Clinical Workflows

4.1 Introduction: Considerations for Model Deployment

In the field of Artificial Intelligence, new models are constantly being developed and reporting ever-increasing performance improvements over the previous. Segmentation models in particular of interest given that segmentations are routinely done by groups in research settings for retrospective and prospective studies, and in clinical settings for evaluations and

The contents of this chapter appeared in paper [114]

treatment planning. However, when it comes to practical implementation, few models are able to progress past the research and development stage. Deploying a model to be used by non-developers, in a high-throughput system can be a non-trivial task. Thus, in this chapter we explore the considerations to be taken into account when implementing an AI model into a clinical setting. Specifically, we present the use case for clinical imaging analysts teams that process scans under clinician guidance for research and clinical usage. The considerations are the following: Resource management, scheduling, error handling, consistent performance, inter-system communication, ease of usage. Exploring each in turn:

1. **Resource Management:** Research groups often operate within shared computing clusters with limited hardware, storage and computation. Deployed models should not only operate within the computational limitations of the cluster, but also in a way that does not impede resource access for other users. Thus, models should be designed for efficiency, using the least resources possible during runtime, while remaining useful for the target task.
2. **Scheduling:** As part of resource management, schedulers are often used to reduce conflicts with other shared resources. Models should be designed to be able to run through schedulers. Schedules should be editable so jobs can be set to run during periods of low loads, but also dynamic in-case an interactive run is needed (e.g. for debugging).
3. **Error Handling:** Easily identifying error sources is vital for any developer, and even more so for non-developers. Since model deployment involves multiple processing steps, if an error occurs, the system should allow end-users to identify the step where the

error occurred and whether it was caused by user error, or requires technical revision.

4. **Consistent Performance:** Many models are hyper-tuned to a specific test set or challenge set that has been well curated and therefore do not generalize well. Thus, when deployed to handle real, uncured data, the model should be up-kept to maintain consistent performance within what it was designed to do. If performance degrades, the design of the system should be such that the model can easily be retrained when new data becomes available.
5. **Inter-system communication:** Medical image data files come in multiple possible formats. These include, but are not limited to: different file types (e.g. dcm and .nii), different image array orientations such as (x, y, z) vs (z, y, x), and different coordinate systems such as Left, Posterior, Superior (LPS) and Right, Anterior, Superior (RAS). Many tools may be used when deploying a model, so it is important that all these tools are working with the same expected input and output formats to enable smooth communication between the different systems. This may require modifications of the data, deployment tools, or the model if these assume different formatting.
6. **Ease of Usage:** Clinical implementations are directed towards non-developer users such as clinicians and imaging analysts. When implementing a model, or any computational tool, consideration must be given to how easy it is to use, where the design of the model should not require programming knowledge to be used. This can be achieved by providing intermediary front-end software with user-interfaces. In this manner, human users and models can communicate without the human requiring direct interaction with the model.

In this chapter, we will describe the tools and methods used to implement an AI kidney segmentation model to be used by clinical imaging analyst teams in research and clinical deployment. Our hypothesis being, AI assisted segmentations provide significant reductions in annotation times.

4.2 Methods

4.2.1 Infrastructure

Imaging Database

Our deployment uses our in-house application QIWS. QIWS is an imaging database where images and masks are stored, and can be imported or exported. The application provides a user-interface where imaging analysts can visualize scans, as well as use tools for contouring and labeling masks. QIWS communicates with other tools such as AI segmentation models to provide scans for segmentation and receive output segmentation masks. Thus, QIWS serves as an intermediary between AI models and human analysts.

Job and Resource Scheduler

For job distribution with resource management and scheduling, we use HTCondor [27]. HTCondor is a distributed-batch computing system popular in high-throughput computing applications ranging from physics to computer animations. Using HTCondor we can monitor resource usage and run jobs based on specified required resources. Listing 4.1 shows an example condor submit file used for inference. As seen by the submit file, users can control

for different types of environments, executable files, system requirements, different types of memory such as shared memory, GPU VRAM and CPU RAM, number of CPUs and GPUs to use, and disk space. Batch jobs can also be limited to spawn a certain number simultaneously and can be removed if they exceed a timeout time.

Job Management and Oversight

While HTCondor can manage jobs and resources, and monitor job progression for oversight, it still requires a user to manually initiate the jobs. A model deployment pipeline may contain various steps that need to be performed in order. Thus, a job manager is needed that can automatically resolve dependencies and launch jobs as needed, in a set schedule, without manual user input. For this we use the Apache Airflow Platform [31]. Using airflow, we can specify workflows using a Directed Acyclic Graph (DAG), where nodes represent specific tasks organized by each other’s dependencies. This allows us to deploy, manage, and monitor the different jobs/steps needed to run our model.

4.2.2 Research Deployment

4-phase contrast-enhanced CT scans are stored in QIWS including: Pre-contrast, Corticomedullary, Nephrographic, and Excretory. The goal of the deployed workflow is to automatically find and download these images, generate kidney and lesion masks using the kidney segmentation model and push back the masks to the corresponding image series in QIWS. The workflow consists of the following steps:

1. **SQL query and generate transfer list:** A SQL query is generated to search for image series of 4 phases that have not yet been contoured/annotated in the QIWS database.

```

universe = docker
docker_image = registry.cvib.ucla.edu/kidney-segmentation:prod
docker_pull_policy = always

executable = inference.sh
should_transfer_files = YES
transfer_input_files = inference.sh, inference.py
requirements = (OpSys == "LINUX" && Arch == "X86_64" && GPUMEM >=
30000)

+DevShmSize = 10737418240

when_to_transfer_output = ON_EXIT
output = log/inference.$(cluster).$(process).out
error = log/inference.$(cluster).$(process).err
log = log/inference.$(cluster).$(process).log

request_cpus = 4
request_gpus = 1

request_memory = 48GB
request_disk = 2GB

max_materialize = 10
periodic_remove = (JobStatus == 2) && (CurrentTime -
EnteredCurrentStatus > (2*3600) )

arguments = "$(MYARGS)"
queue

```

Listing 4.1: Example HTCondor submit file.

From the query, we generate a transfer list containing the image data information for the series that meet our criteria.

2. **Image Download:** Using the transfer list, images are downloaded and stored locally.
3. **Model Inference:** We then apply the kidney segmentation model to perform inference and generate masks for the kidneys and lesions on the available scans. The CNN instance of this model is trained on a combination of in-house imaging data and the publicly available KitS23 data [44]. FMR reasoning described in Section 3.2.2 is applied

to the segmentation results.

4. **Mask Push:** Generated masks are pushed back to their corresponding image series in QIWS under a CAD label, for review and editing by an image analyst.
5. **Generate Summary:** After the inference workflow is completed, a summary file is generated containing the image series information for scans that were processed. This summary is constructed based on the original transfer list with information appended as to whether the scan was successfully processed end-to-end, and if not, what step of the workflow failed to proceed with its corresponding error message.

This workflow is managed and monitored by an airflow DAG as seen in Figure 4.1. The DAG is scheduled to run on a weekly schedule during Sunday off-hours to minimize conflicts with other active jobs. Each individual step in the workflow is submitted as an HTCondor job to automatically allocate available resources. The workflow summary file is automatically emailed to relevant project personnel for review.

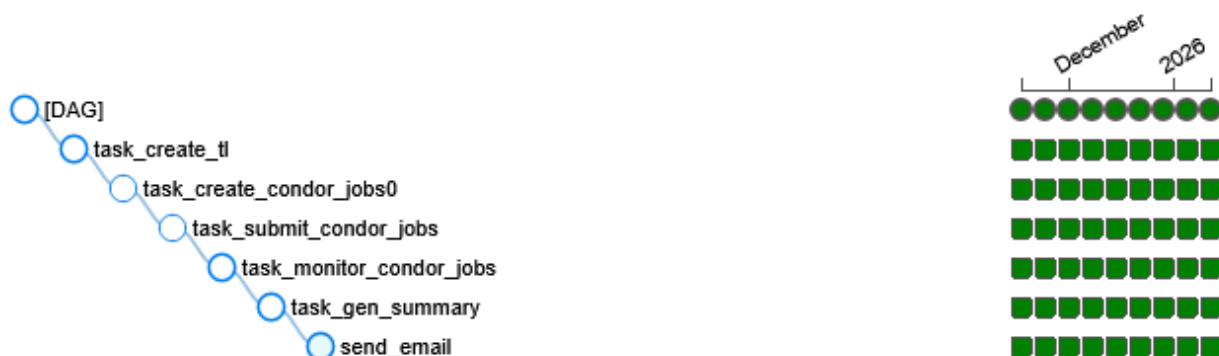


Figure 4.1: Kidney Segmentation DAG for research deployment

4.2.3 Clinical Deployment

The Quantitative 3D (Q3D) lab at UCLA processes patient scans for 3D volumetrics, often through manual annotations. This includes kidney volume measurements to evaluate kidney donors for transplant patients. The Q3D lab typically receives 12 to 15 requests for kidney volumetrics on a weekly basis, with the processing time of each case averaging 25 to 30 minutes [114]. As such, an automated kidney segmentation algorithm could be highly beneficial.

The clinical deployment of the kidney segmentation model involves communication with the hospital’s Picture Archiving and Communication System (PACS). The initial trigger comes from a specific hospital Current Procedural Terminology (CPT) image code indicating a specific order for the Q3D lab. This, in turn, initiates a query to transfer the CT scan from a Visage PACS to an internal Orthanc Dicom Server [52]. When a CT scan is transferred to Orthanc, jobs are then triggered for labeling, segmentation, and lastly exported as a DICOM-seg object back into the Visage PACS. Within the Visage PACS, the DICOM-seg object containing the kidney segmentation can be reviewed and edited as needed for volumetry calculations.

4.2.4 Clinical Deployment Evaluation

We randomly selected 22 CT scans requested for kidney volumetrics in transplant donor assessments from Q3D. We compared annotation times between manual vs AI-assisted. We also compared kidney volumes for segmentations before editing DICOM-seg in Visage, and after editing DICOM-seg in Visage. Wilcoxon Signed Rank test [97] was used to determine

statistical differences in annotation times. The Root Mean Square Error (RMSE) was calculated between the volume measurements to evaluate the extent of editing necessary to achieve clinically acceptable results.

4.3 Results

4.3.1 Research Deployment

Since its initial deployment, the kidney segmentation model has processed 5,803 research cases. Per marking, this is comprised of 5,559 left kidneys, 5,494 right kidneys, and 1665 lesions.

4.3.2 Clinical Deployment

The mean/median time for manual annotations was $11.5/11.0 \pm 3.7mins$ with a minimum time of 7.0 mins and a max time of 19 mins. In contrast, the mean/median time for AI-assisted annotations was $3.9/3.0 \pm 1.0mins$ with a minimum time of 1.5 min and a max of 5.0 mins. Figure 4.2 shows boxplots of the distributions of time in minutes per case between manual annotations and AI-assisted annotations. AI-assisted annotations reduced the time to complete a case by 67.6%, with a highest reduction of 89.5% and a lowest reduction of 37.5%. The decrease in annotation time was found to be statistically significant ($p < 0.05$) by the Wilcoxon Signed Rank test. Volume measurements between the original AI segmentation and the analysts edited version were compared to assess the amount of change needed to complete a case. The RMSE for the kidney volume was $3.65mL$ and $4.59mL$ for the right

and left kidney, respectively. Figure 4.3 shows boxplots of the distribution of percent changes in volume going from the AI segmentation to analysts edited segmentation for the right and left kidneys. The mean percent change for the right kidney was $-1.98 \pm 0.75\%$ with a max absolute change of -3.79% and a minimum change of -0.84% . The left kidney volume had a mean percent change of $-2.48 \pm 1.05\%$ with a max of -4.92% and a minimum of -0.75% .

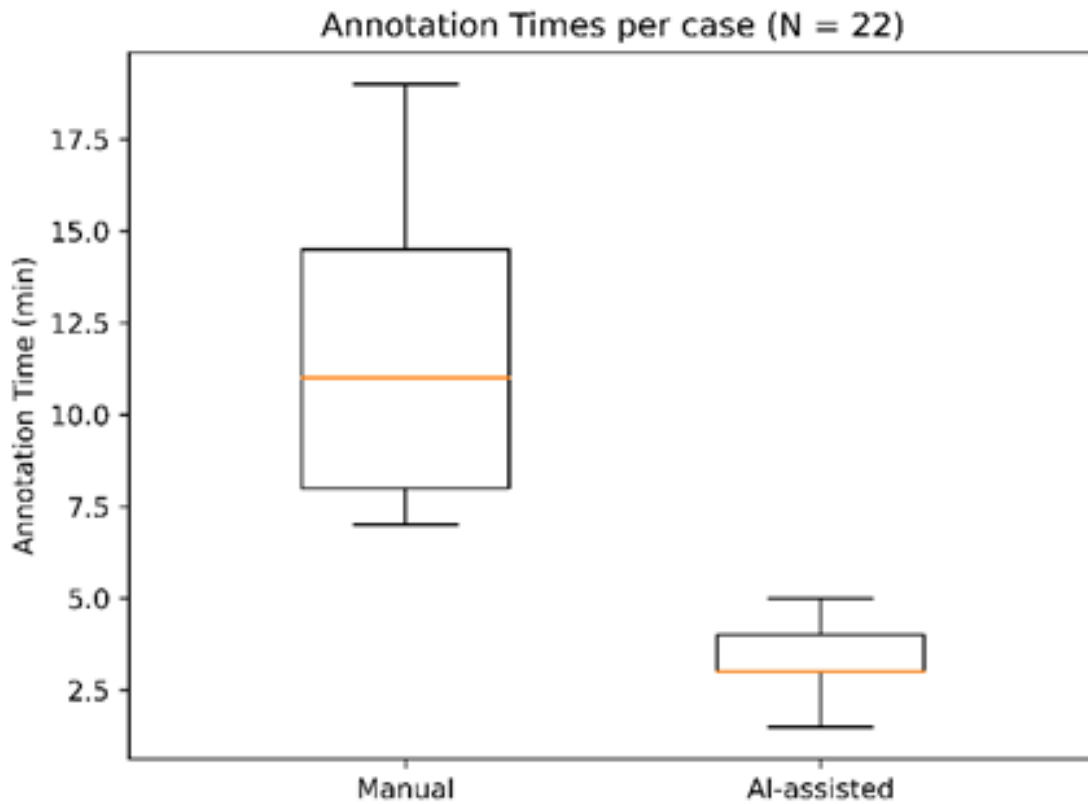


Figure 4.2: Boxplots comparing annotation times for manual vs AI-assisted.

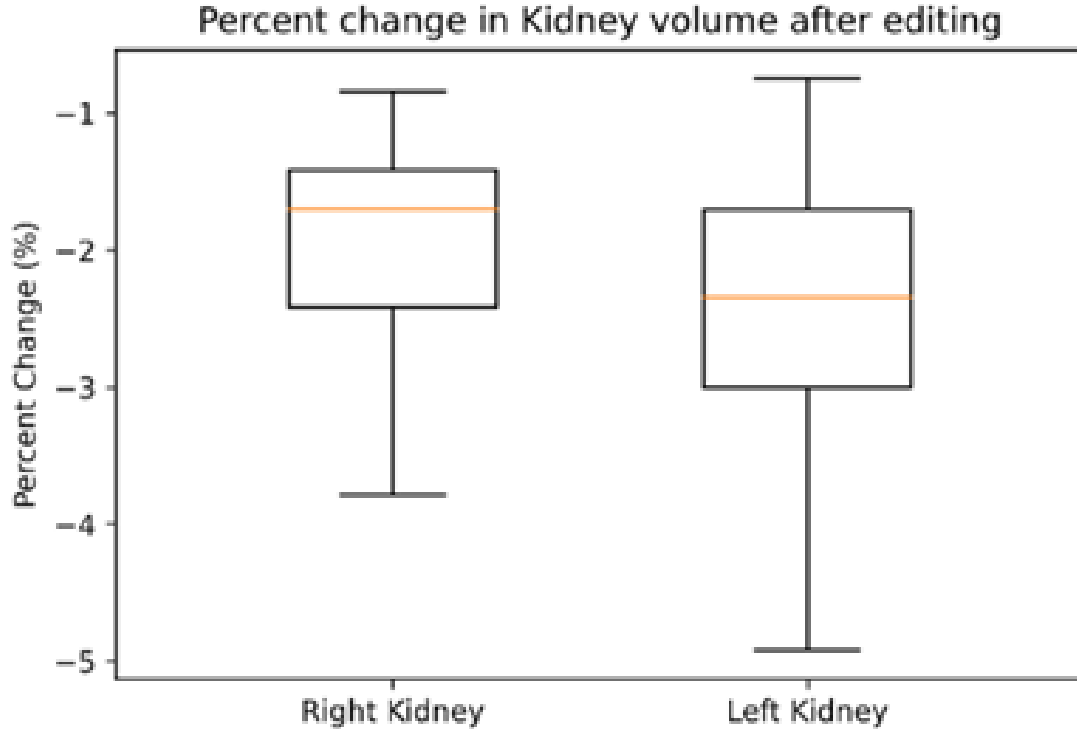


Figure 4.3: Boxplots of percent change in volume measurement after editing for right and left kidneys.

4.4 Discussion

The clinical deployment demonstrates the utility of the kidney segmentation model, with significant reduction in annotation times without impacting the standard Q3D workflow. Volume comparisons showed that the model tended to overestimate the kidney volumes, however, for practical purposes, the volume differences tended to be minor and the time saving benefits outweigh the editing needed to correct the volume. This study, however, is limited by the small sample size of $N = 22$. Thus, a more rigorous study of model robustness would require more samples. Nonetheless, given that the volume of cases is 12 to 15 per week, our current sample size can be considered representative of a weekly workload in Q3D.

4.5 Conclusion

We successfully deployed the SM kidney segmentation model into the research and clinical workflows. The workflow was able to run automatically to search for, process, and export cases. It yielded significant improvements in clinical annotations times with minimal editing required.

Chapter 5

Automated Nephrometry Scoring

Using SimpleMind

5.1 Introduction

5.1.1 Nephron Sparing Procedures

Partial Nephrectomy is considered the gold standard for treatment of small renal lesions. As opposed to Radical Nephrectomy, where the entire affected kidney is excised with the lesion, Partial Nephrectomy is a nephron-sparing procedure. It aims to excise as much of the lesion as possible while minimizing damage to surrounding renal tissue. This technique has been found to be just as effective as Radical Nephrectomy for lesion treatment, while preserving kidney function [89]. However, in other cases, patients with small renal lesions

The contents of this chapter appeared in paper [82]

may not be suitable candidates for nephrectomy due to age or other comorbidities that make surgeries high risk. For these cases, nephron sparing ablation procedures have been found to be as effective as partial nephrectomy while reducing additional risks that come with surgery [53]. The increased usage of medical imaging has led to an increase of incidental discovery of small renal lesions in their early stages; potential candidates for nephron sparing procedures. However, given the increased complexity of the procedure, size alone is insufficient to determine if the lesion is a suitable candidate. The lesion shape and position must also be considered. If the lesion is not properly assessed, this can lead to failures, often in the form of conversions from a planned Partial Nephrectomy to a Radical Nephrectomy. Petro et al. [92] found conversion rates of 5%, while Wang et al. [117] found a 4.2% conversion rate. In other instances, it has been found that 15 to 30% of lesions post-nephrectomy are benign [109]. Inaccurate assessments can lead to over-treatment, where a benign lesion may be surgically excised unnecessarily. This can expose the patient to harm such as decreased kidney function, surgical risks, as well as psychological and financial distress [109]. Given the risks of nephron sparing surgical failures and over-treatments, there is a clinical need for evaluating lesion complexity to determine the proper procedure.

5.1.2 Nephrometry Scoring

To assess renal lesion complexity, Nephrometry scoring systems have emerged. The first scoring system, and most widely used, is the **Radius, Exophytic, Nearness, Anterior/posterior, and Location relative to the polar lines (RENAL)** nephrometry score by Kutikov and Uzzo [61]. It evaluates lesion complexity using four measurements: **Radius**, measures

the longest diameter of the lesion, **Exophytic/endophytic**, measure how much of the lesion is embedded inside the kidney, **Nearness**, distance of the lesion to the Urinary Collection System (UCS), **Anterior/posterior**, whether the lesion is located anterior or posterior, and **Location**, which refers to the location of the lesion relative to a set of imaginary polar lines above and below the UCS with a midline in between, see Figure 5.1 [61]. Table 5.1 shows how a renal lesion is scored using RENAL. The complexity score ranges from a minimum of 4 to a maximum of 12 using the **R**, **E**, **N**, and **L** criteria, while **A** only provides a categorical description on its anterior (A), posterior (P), or indeterminate (X) position. Score sum ranges are further categorized into low (4 to 6), moderate (7 to 9), and high complexity (10 to 12).

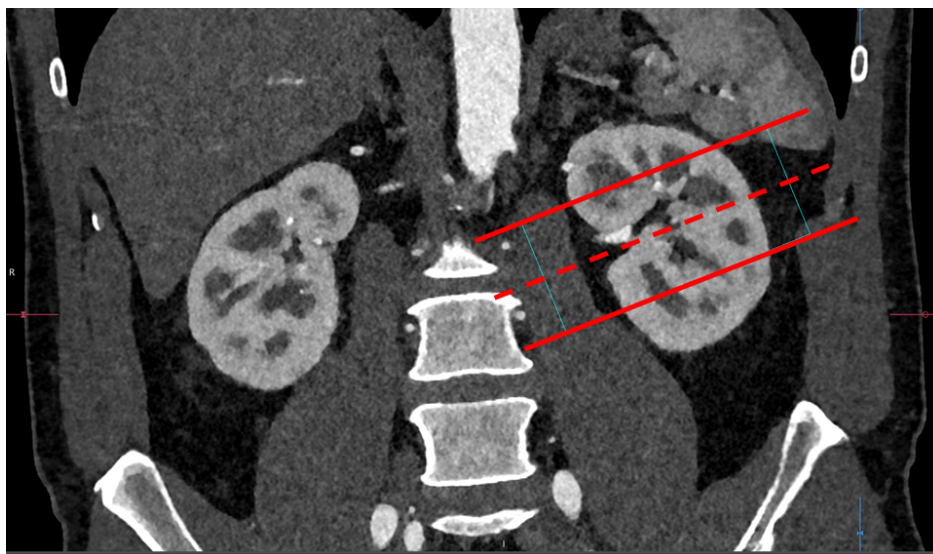


Figure 5.1: Example image illustrating the kidney polar lines

This system has been validated independently by multiple studies, demonstrating significant differences in various outcomes such as ischemia time, blood loss, kidney function, and surgical complications in partial nephrectomies [56, 40, 93, 77]. This also extends to ablation procedures, where the RENAL nephrometry score has been found to be associated with

Table 5.1: RENAL nephrometry score criteria.

Component	What it measures	1	2	3
R (Radius)	Lesion diameter (cm)	≤ 4	> 4 to ≤ 7	> 7
E (Exophytic/Endophytic)	Percent of lesion protruding from renal parenchyma	$\geq 50\%$ exophytic	$< 50\%$ exophytic	Entirely endophytic
N (Nearness)	Min distance (mm) to collecting system or renal sinus	≥ 7	> 4 to < 7	≤ 4
A (Ant/Post)	Anterior vs posterior on axial imaging (descriptor only)		A / P / X	
L (Location)	Relative to polar lines (central/interpolar vs polar)	Entirely above/below polar line	Crosses polar line	$> 50\%$ between polar lines or crosses midline

ablation failure [98]. However, RENAL nephrometry category differences and correlations with outcomes are variable, with studies finding no significant differences in outcomes such as operative time, transfusion rate, complication rates, and pre/post operative creatinine clearance (indicator of kidney function) [40], as well as blood loss, complications, length of stay and positive surgical margin [77]. This can be further complicated by different Partial Nephrectomy procedures with Hua et al. [50] finding the RENAL nephrometry score to be more accurate for predicting outcomes after Robot Assisted Partial Nephrectomy vs Laparoscopic Partial Nephrectomy.

5.1.3 Reproducibility of RENAL Nephrometry Scoring

Given that nephrometry scoring is manual, reproducibility of scores between readers is also a concern. Evidence on the reproducibility of RENAL is mixed with studies demonstrating the scoring to be reproducible among different readers [4, 86, 112], while others demonstrating

reduced inter-rater agreement [85, 57, 49]. Arora et al. [4] evaluated agreement of the RENAL nephrometry scoring between 2 radiologists on a cohort of 42 patients. They found a minimum agreement of 85.71% with a Kendall's Tau [55] of 0.88 on RENAL_L, and a max agreement of 97.61% with a Kendall's Tau of 0.98 on RENAL_R. Similarly, Montag et al. [86], using a cohort of 149, evaluated agreement between two urology residents. Agreement was evaluated using concordance rates and Kendall's Tau β coefficient [54]. They found concordance rates for the RENAL components ranging from a minimum of 0.86 for RENAL_N and 0.96 for RENAL_R. Kendall's Tau β coefficients ranged from 0.85 for RENAL_L and 0.90 for RENAL_R. Lastly, in a cohort of 46 patients, Vilaseca et al. [112] evaluated RENAL components inter-observer agreement between two radiologists. They used exact agreement proportions ranging from 0.80 in RENAL_E, to 0.98 in RENAL_R. Kappa agreements among the raters corresponded to the exact agreement proportions ranging from 0.65 in RENAL_E to 0.96 in RENAL_R.

In a cohort of 120 cases, Monn et al. [85] evaluated RENAL scoring agreement using Cohen's Kappa Coefficient [20] across 3 pairs of readers: 2 urologists, 2 radiologists, and a collaborator pair composed of 1 urologist and 1 radiologist. The RENAL_R score demonstrated strong agreement across urologists, radiologists, and collaborators with Kappa values 0.860, 0.852, and 0.841, respectively. The RENAL_E, RENAL_N, and RENAL_L components showed fair-to-moderate agreement, with RENAL_E ranging from 0.357 to 0.442, RENAL_N from 0.419 to 0.486, and RENAL_L from 0.356 to 0.440. Similarly, Khandwala et al. [57] evaluated agreement between two radiologists in a 49 patient cohort also using Cohen's Kappa. The highest Kappa agreement was found in RENAL_R with 0.72, followed by RENAL_L with 0.85, RENAL_N with 0.44, and RENAL_E with 0.33. Hew et al. [49] evaluated RENAL

agreement between 3 urologists on a 134 patient cohort using Fleiss' Kappa [30]. They obtained minimum agreement of 0.23 for RENAL_N, 0.43 for RENAL_E, 0.47 for RENAL_L and a maximum agreement of 0.73 for RENAL_R. These works demonstrate that RENAL nephrometry scoring remains susceptible to inter-observer variability, necessitating more robust and reproducible methods. Additionally, the scoring process can be time-consuming and often requiring training by experts. Thus, there is a need to automate the scoring process for high throughput without compromising measurement accuracy and improving reproducibility.

5.1.4 Automated Nephrometry

Efforts have been made to use segmentation models and geometric algorithms to fully automate nephrometry scoring. Stemming from the KiTS19 challenge [43, 46], Heller et al. [47] used AI models and geometric algorithms to segment kidneys and their respective lesions, and then extract nephrometry features used in the RENAL scoring system. Their automated system achieved comparable scores to human readers with a Lin's concordance correlation coefficient of $p = 0.59$. This work was continued by Abdallah et al. [1], where in addition to using the categorical scores from measurements generated by the automated method, they also used the continuous RENAL features to construct prediction models using Logistic Regression. Both their automated categorical and continuous prediction models achieved superior predictive performance for the prediction of malignancy, tumor stage, tumor grade, tumor necrosis, and surgery type. Wood et al. [120], expanded this methodology to automate other nephrometry scoring systems such as: Preoperative Aspects and Dimensions

Used for an Anatomical (PADUA) [29], Centrality Index (CI) [106], and Contact Surface Area (CSA) [69]. Similar to [1], they found the automated systems to yield superior performance, over human readers, when used to predict tumor stage, tumor grade, indolence, necrosis, surgical approach, and high grade complications.

These works demonstrate the effectiveness and potential utility of automating nephrometry scoring. Given that nephrometry scoring relies on various spatial relationships between lesions and organs, these methods can be readily integrated into the SimpleMind Software Environment. As described in Chapter 2, SM contains agents capable of general computer vision tasks and agents capable of extracting the various features and spatial relationships among organs. Our goal is to construct an Anatomical Knowledge Graph within SM to automatically extract features used in the RENAL scoring system.

5.2 Methods

5.2.1 Dataset

We used the publicly available C4KC-KITS dataset from The Cancer Imaging Archive [45] containing 210 4-phase contrast enhanced CT scans with corresponding kidney and lesion masks. We filtered the dataset to only include cases containing a Corticomedullary phase scan for a total of 209 cases. Of the 209 cases, 61 cases meeting the criteria of containing a single lesion were randomly selected to be used as test set to compare the automated nephrometry scoring method against human raters. The remaining 148 cases were used as training/validation to train an SM kidney and lesion segmentation model.

5.2.2 Kidney and Lesion Segmentation

We evaluated two versions of the automated nephrometry scoring: (1) a version using the manual kidney and lesion annotations available from C4KC-KITS to which we will refer to as CAD. (2) a version using automated segmentation through SM to which we will refer to as AUTOSEG_CAD. The AUTOSEG_CAD was trained to segment the kidneys and lesions on 148 scans reserved for training and validation mentioned in 5.2.1. For training and inference, we used the FMR branch from the KG described in Section 3.2.2.

5.2.3 Biological Landmark Segmentation

As mentioned in the introduction 5.1, computation of nephrometry features requires measurements between kidney landmarks, mainly that of the UCS, from which the polar lines are derived. So we must first segment this region of the kidney.

Identification of the Affected Kidney

Images and corresponding masks are loaded into the SM KG. After separating the kidney and lesion masks, we identify potential lesion and kidney candidates. The best lesion candidate is identified as the largest candidate in contact with the kidneys. Only a single lesion is selected even if multiple are present in the scan. The affected kidney is then identified by contact with the best lesion candidate (see Figure 5.2). Contact between two regions A and B is defined by single voxel dilation of B and computing the overlap between A and dilated B. If there is overlap, contact between A and B is *True*.

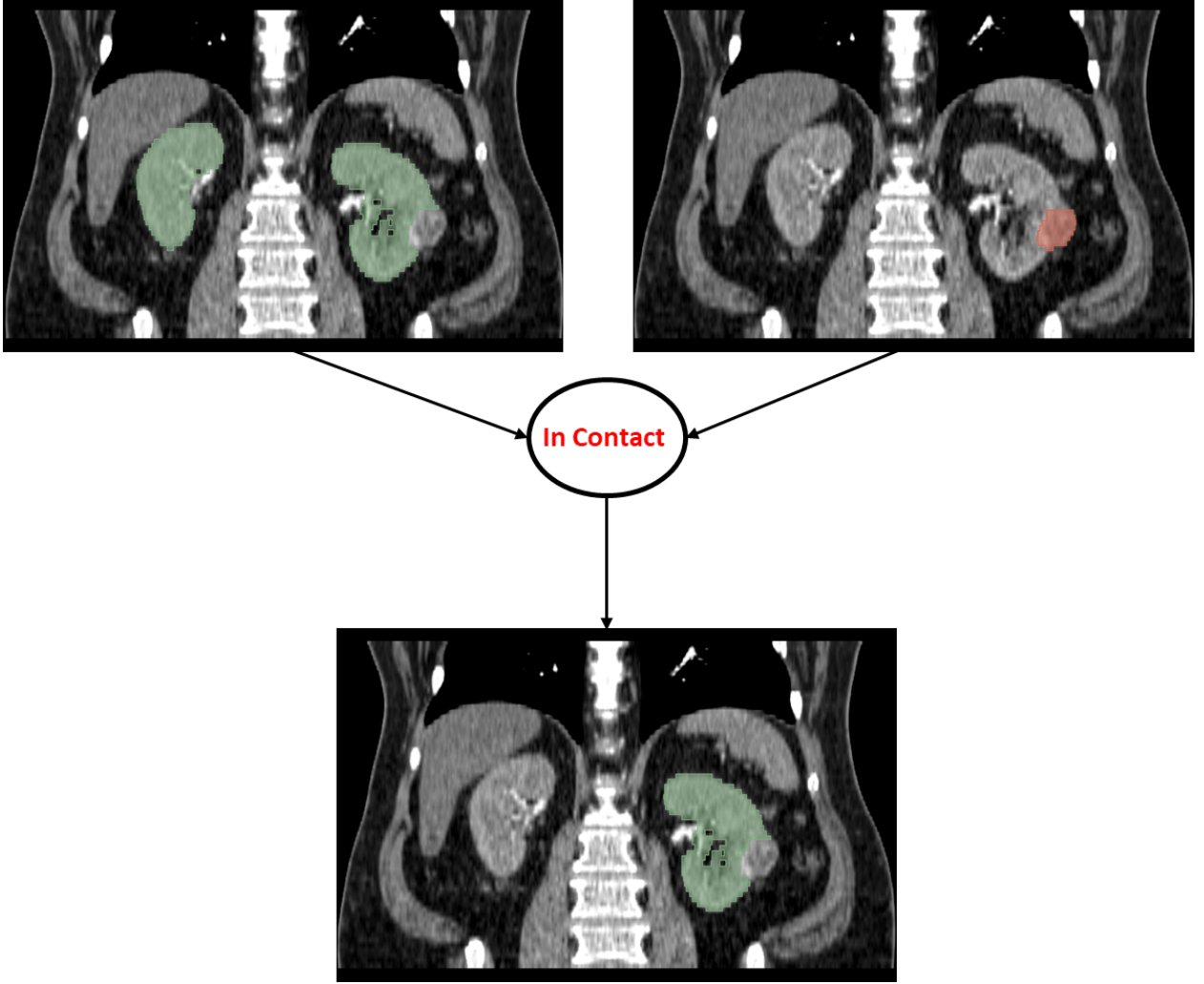


Figure 5.2: Identification of the affected kidney via contact with the lesion (red).

Urinary Collection System and Polar Line Segmentation

The UCS is located using the methodology from [47]. Agents first segment body fat via HU thresholding from -195 to -45 HU (Figure 5.3).

From the affected kidney mask, we extract the convex hull to fill in gaps in the kidney mask. To identify the renal sinus fat, we erode this convex hull, using a circular structuring element 20 mm in diameter, to exclude fat voxels surrounding the outer kidney cortex (Figure 5.4).

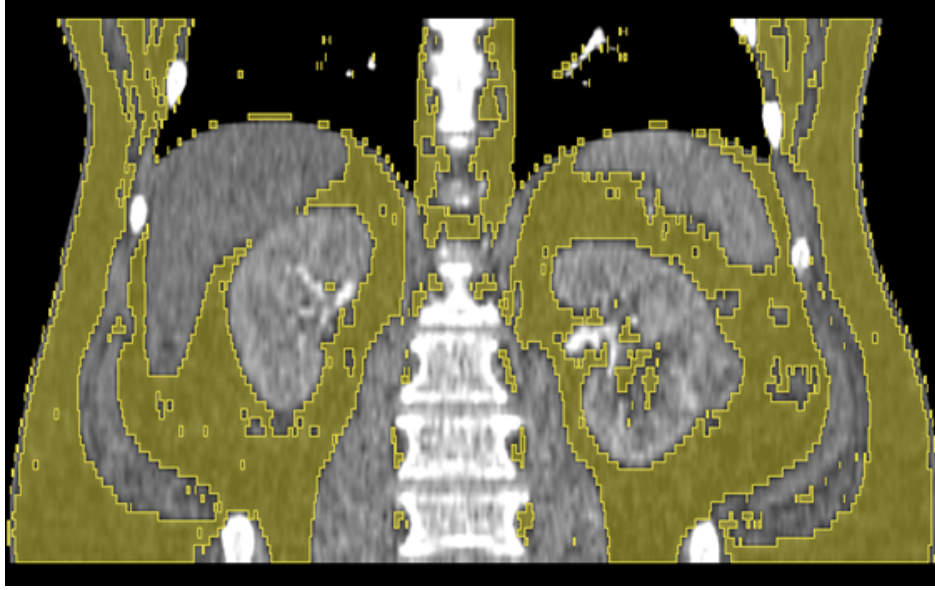


Figure 5.3: Body fat segmentation using HU thresholding from -195 to -45 HU.

Next, the renal sinus fat is isolated by using a boolean logical *AND* operation between the body fat voxel mask and the eroded convex hull of the affected kidney (Figure 5.5).

The UCS is then obtained by taking the convex hull of the renal sinus fat (Figure 5.6).

Using the UCS, the kidney polar lines are extracted by generating a volume perpendicular to the z-axis, between the upper and lower z-slices that form the UCS (see Figure 5.7).

The polar lines are rotated to follow the angle of the kidney axis relative to the z-axis (Figure 5.8). A complete description of the polar line rotation methodology can be found in Appendix B.

5.2.4 Lesion Diameter, Nearness, Endophycity, and Polar Line Location

Features computed from the RENAL scoring system [61] are automated via the SM KG. Lesion diameter is computed by obtaining the largest 3D euclidean distance between the

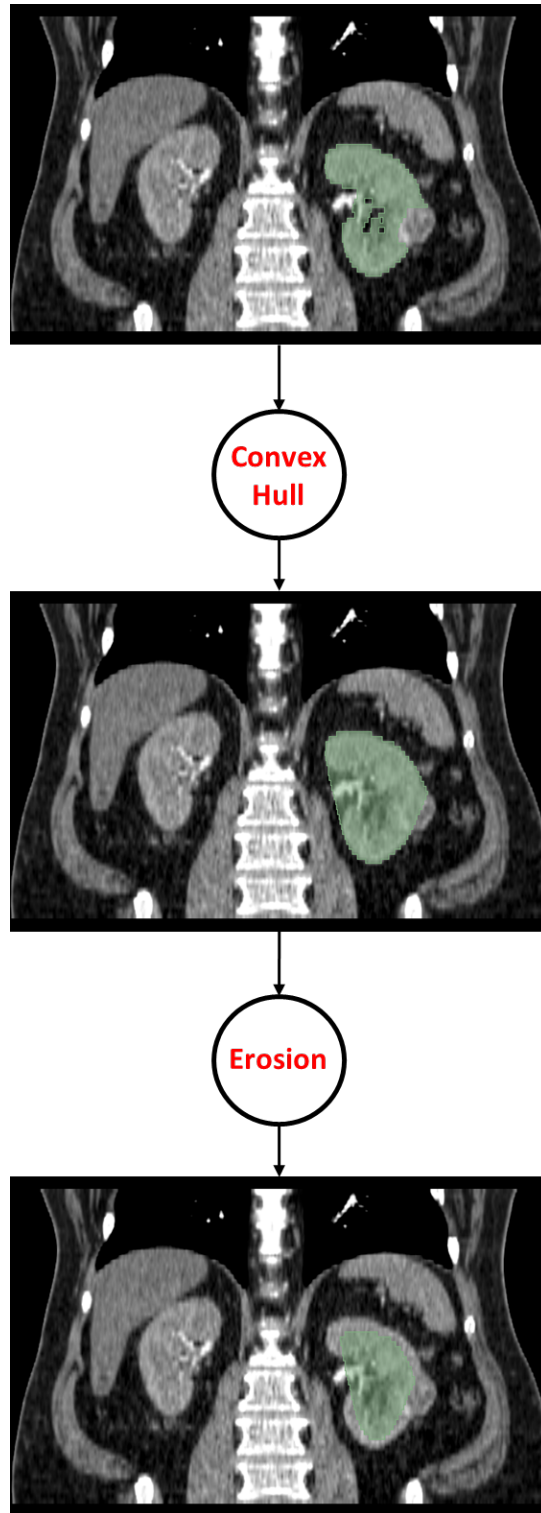


Figure 5.4: Extraction of the convex hull of the affected kidney and subsequent erosion.

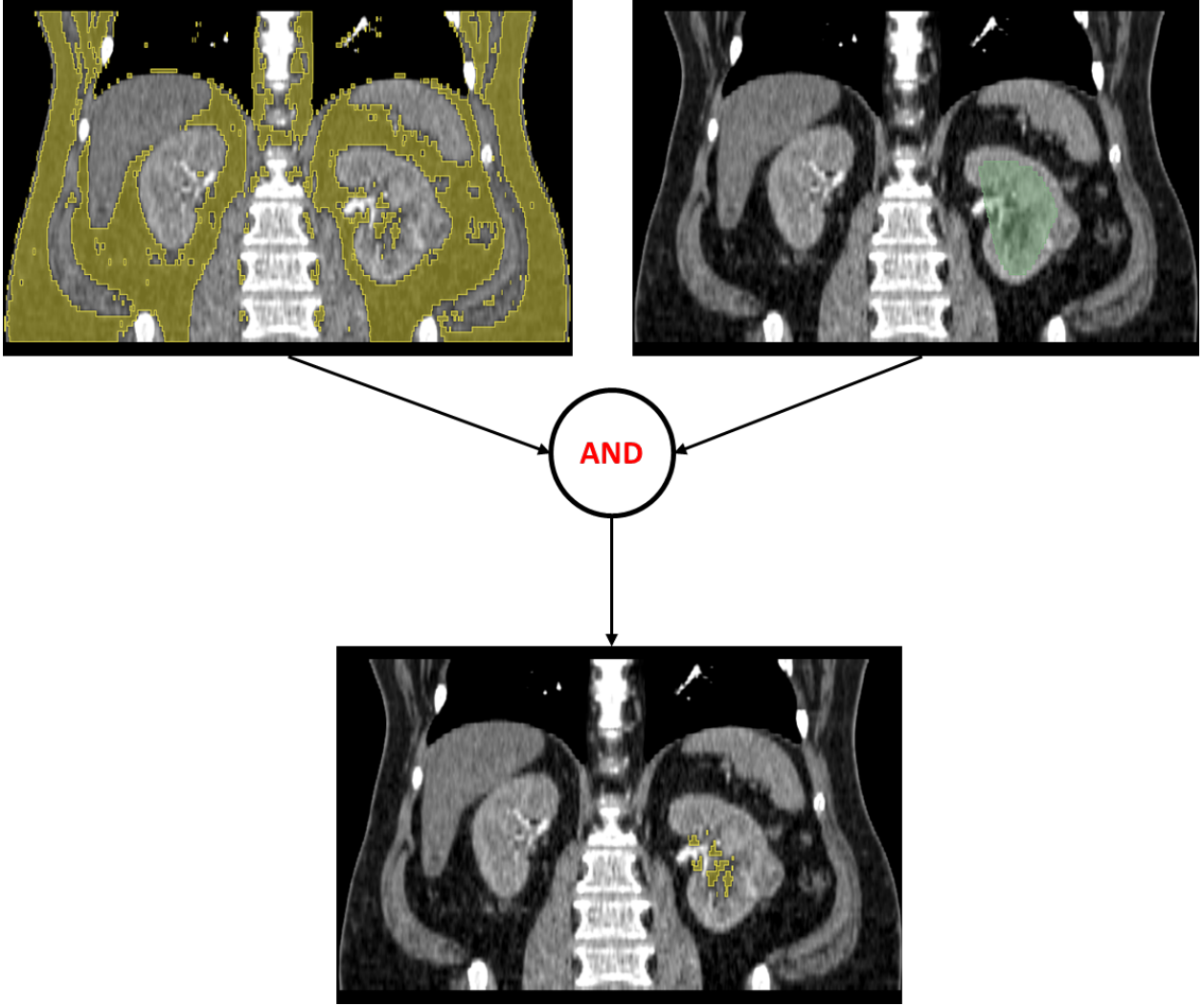


Figure 5.5: Segmentation of the renal sinus fat using a boolean logical *AND* operation between the body fat and the eroded convex hull of the affected kidney.

lesion mask voxels using the pyradiomics library [110]. Nearness, referring to the distance of the lesion to the UCS is computed via the minimum edge to edge distance between the lesion and the UCS. To compute endophycity, we use the convex hull of the affected kidney to fill in the gap left by the lesion. The endophycity of the lesion is then obtained by computing the fraction of the lesion mask that overlaps with the kidney's convex hull mask. Lastly, Polar Line location is computed as the fraction of the lesion mask that overlaps with the polar

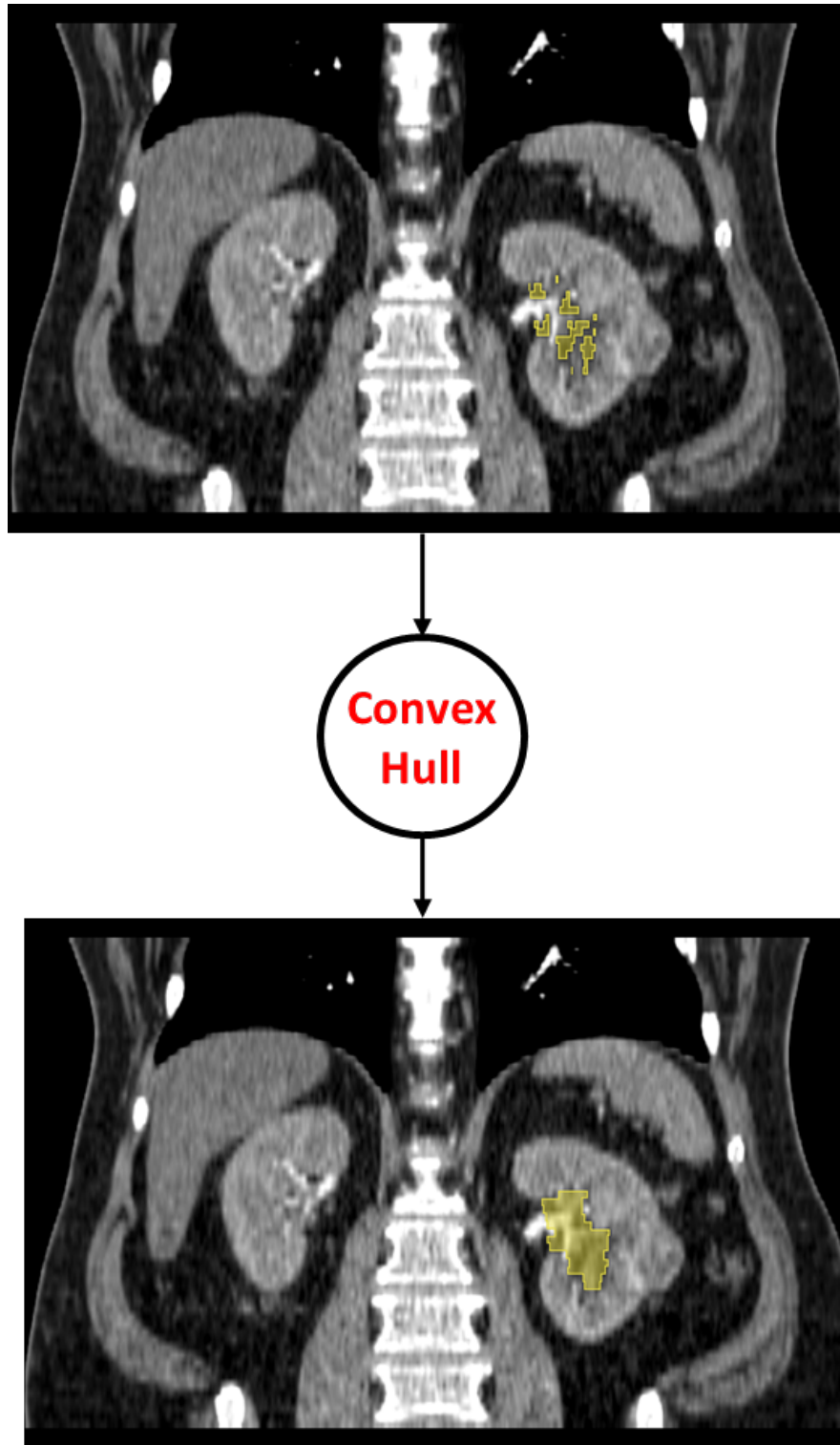


Figure 5.6: Final segmentation representing the UCS of the affected kidney using the convex hull of the renal sinus fat.

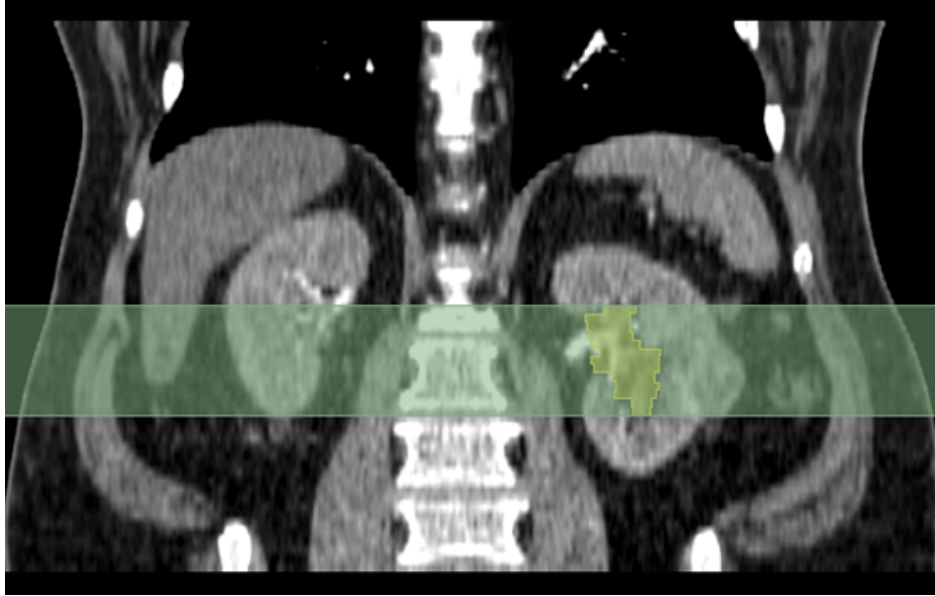


Figure 5.7: Initial polar line segmentation using the UCS boundaries in z .

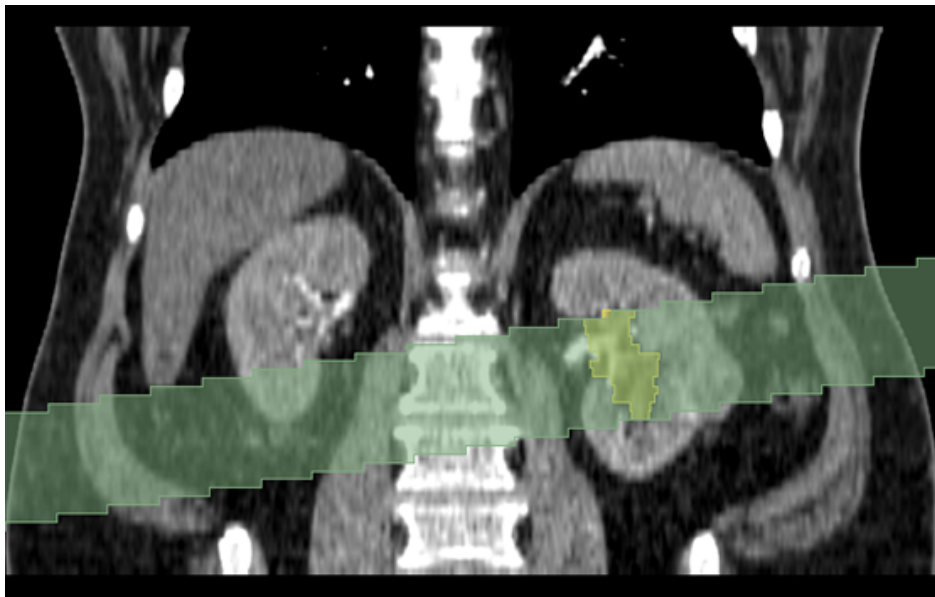


Figure 5.8: Final polar line segmentation rotated to be oriented with kidney axis.

lines.

5.2.5 Experimental Testing

To test our automated nephrometry scoring system, we compared CAD and AUTOSEG_CAD with RENAL ordinal scores done by 3 imaging analysts as independent raters: MML, ESD and SKA. They were trained to perform the RENAL scoring analysis by a Research and Technical operations manager, KVV, and an abdominal radiologist, SSR, with 7 and 25+ years of experience in renal analysis, respectively.

5.2.6 Statistical Analysis

Accuracy of the automated kidney and lesion segmentation was assessed using the DSC [124]. Agreement of continuous measurements per component between AUTOSEG_CAD and CAD nephrometry scoring was assessed using the Intra-Class Correlation Coefficient (ICC) [104], the Concordance Correlation Coefficient (CCC) [71], Pearson’s R [90] and Bland-Altman analysis [13]. Equivalence between AUTOSEG_CAD and CAD nephrometry measurements was assessed using Two one-sided T-tests (TOST) [99] with bounds set at $\pm 15mm$ for lesion diameter, $\pm 1.5mm$ for Nearness, and $\pm 10\%$ for lesion endophycity and polar line overlap. CAD and AUTOSEG_CAD agreement was assessed using the quadratic weighted kappa agreement for ordinal variables [21] and Bland Altman analysis comparing the automated methods against the human median. Given the potential for variance between human raters, we assessed whether the automated methods scores were within the ranges of the human raters through a Leave-One Out (LOO) analysis, where each rater (human or CAD) was iteratively compared to the remaining raters. The within variability proportion $\hat{p}$ of k cases that fit within the $min - max$, $median \pm 1$ and $median \pm 2$ was obtained with Binomial

Proportion Confidence Intervals computed using the Clopper-Pearson’s exact method [19].

Depending on the range of scores given for a nephrometry component, and given the limited scoring range [1,3], randomly assigning a nephrometry component value has minimum probability of 1/3 to be within the *min* – *max* variability of humans if all humans agree on a single value. Conversely, if the *min* – *max* of humans covers the full score range, random assignment has a probability of 1 to be within the human range. Similarly, when computing within variability coverage of *median* $\pm$ 1, if the median is either 1 or 3, random assignment has a 2/3 probability of being within that range. This probability becomes 1 if the median is 2. For these reasons, to determine if the automated nephrometry scoring methods behave differently than random assignment, the within variability $\hat{p}$ of the automated methods was compared to that of a random nephrometry scoring system (RND_CAD), which randomly assigns an ordinal in the interval [1, 3] for each nephrometry component. We performed McNemar’s test [79] to determine significant differences in within variability $\hat{p}$ between the automated methods and the random nephrometry scoring. Significance was established with Benjamini-Hochberg [7] corrected p-value, $p_{BH} \leq 0.05$.

5.3 Results

5.3.1 Automated Segmentation Accuracy

Summary statistics for the AUTOSEG_CAD kidney and lesion segmentation DSC are shown in Table 5.2. AUTOSEG_CAD achieved a mean kidney segmentation DSC of 0.971 ± 0.0154 with a median of 0.974 and a minimum and maximum of 0.922 and 0.988, respectively. For

lesion segmentation, it achieved a mean DSC of 0.859 ± 0.161 , a median of 0.918 with a minimum of 0 and a maximum DSC of 0.980. Figure 5.9 shows boxplots of the distribution of DSC scores for kidney and lesion segmentation.

Table 5.2: Dice similarity coefficient (DSC) summary statistics for kidney and lesion segmentations.

Label	Count	Mean	Std	Median	Min	Max
Kidney	61	0.971	0.0154	0.974	0.922	0.988
Lesion	61	0.859	0.161	0.918	0.000	0.980

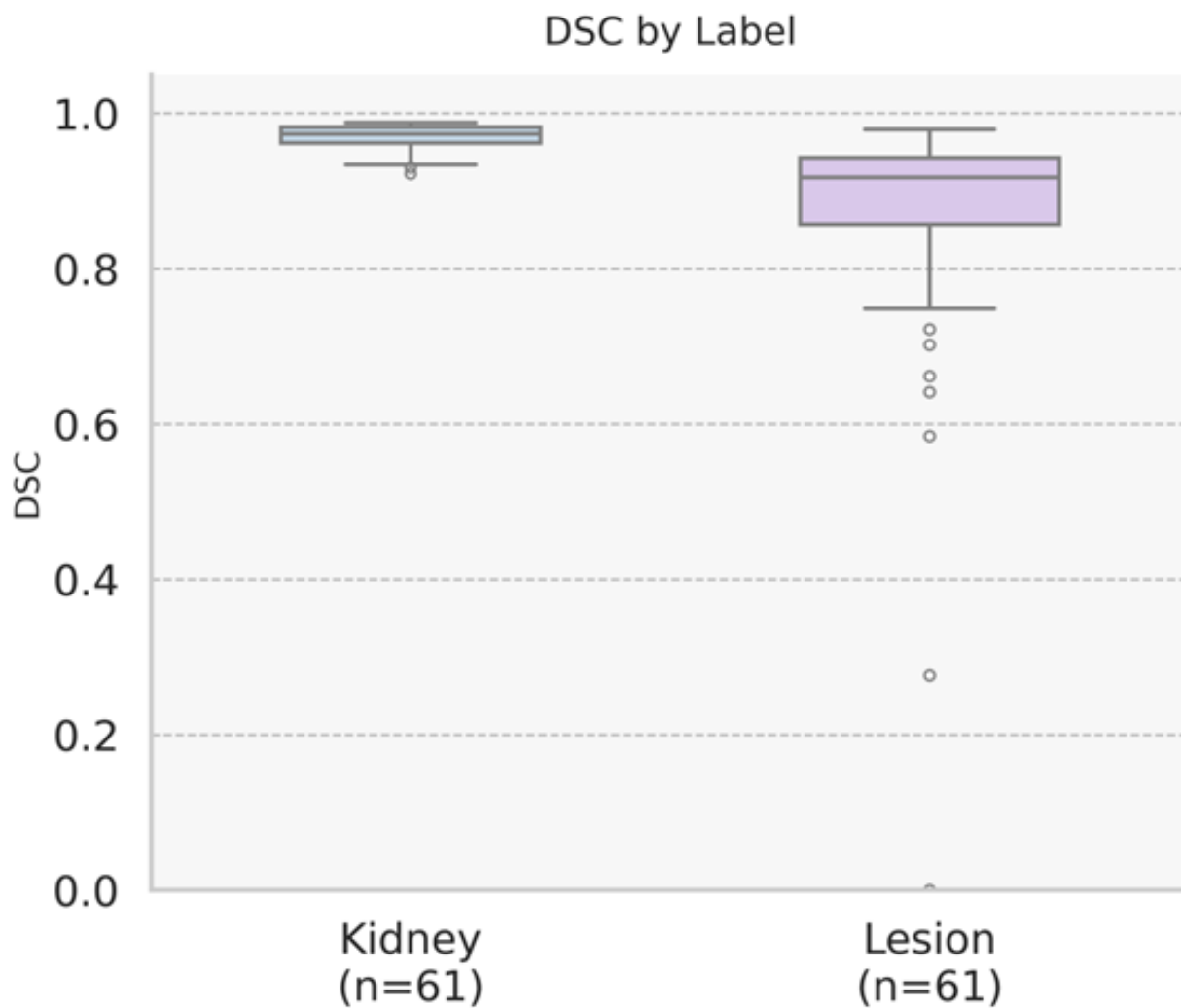


Figure 5.9: Boxplots of kidney and lesion DSC.

5.3.2 Nephrometry Component Agreement and Equivalence Analysis of AUTOSEG_CAD and CAD

The AUTOSEG_CAD method failed to generate a lesion segmentation for one case. Both automated methods failed to generate a UCS segmentation on one separate case. Thus, $n = 60$ for diameter and endophycity measurements, while $n = 59$ for polar line and nearness measurements. Agreement, reliability and correlation analysis between AUTOSEG_CAD and CAD produced values > 0.90 : Lesion Diameter ($ICC = 0.965$, $CCC = 0.964$, $r = 0.969$), Lesion Endophycity ($ICC = 0.906$, $CCC = 0.904$, $r = 0.916$), Lesion Polar Line Overlap ($ICC = 0.922$, $CCC = 0.921$, $r = 0.923$) and Nearness ($ICC = 0.930$, $CCC = 0.929$, $r = 0.930$).

Bland-Altman analysis of the lesion diameter produced a mean difference/bias of $-2.82 \pm 8.03mm$ with Limits of Agreement (LoA) at $[-18.57, 12.92]$. Endophycity produced Bland-Altman plots with bias of -0.04 ± 0.12 , LoA of $[-0.27, 0.19]$. Polar line overlap obtained a bias of -0.02 ± 0.15 with LoA at $[-0.31, 0.26]$. Lastly, nearness obtained a bias of $0.18 \pm 2.15mm$ with LoA of $[-4.03, 4.39]$. Figure 5.11 shows the Bland-Altman analysis for AUTOSEG_CAD vs CAD for all RENAL component measurements.

TOST equivalence test produced significant results for equivalence between AUTOSEG_CAD and CAD across all RENAL components: Diameter (mm) $CI_{90\%} = [-4.56, -1.09]$, Endophycity $CI_{90\%} = [-0.066, -0.015]$, Polar Line overlap $CI_{90\%} = [-0.055, 0.009]$, and Nearness (mm) $CI_{90\%} = [-0.29, 0.65]$.

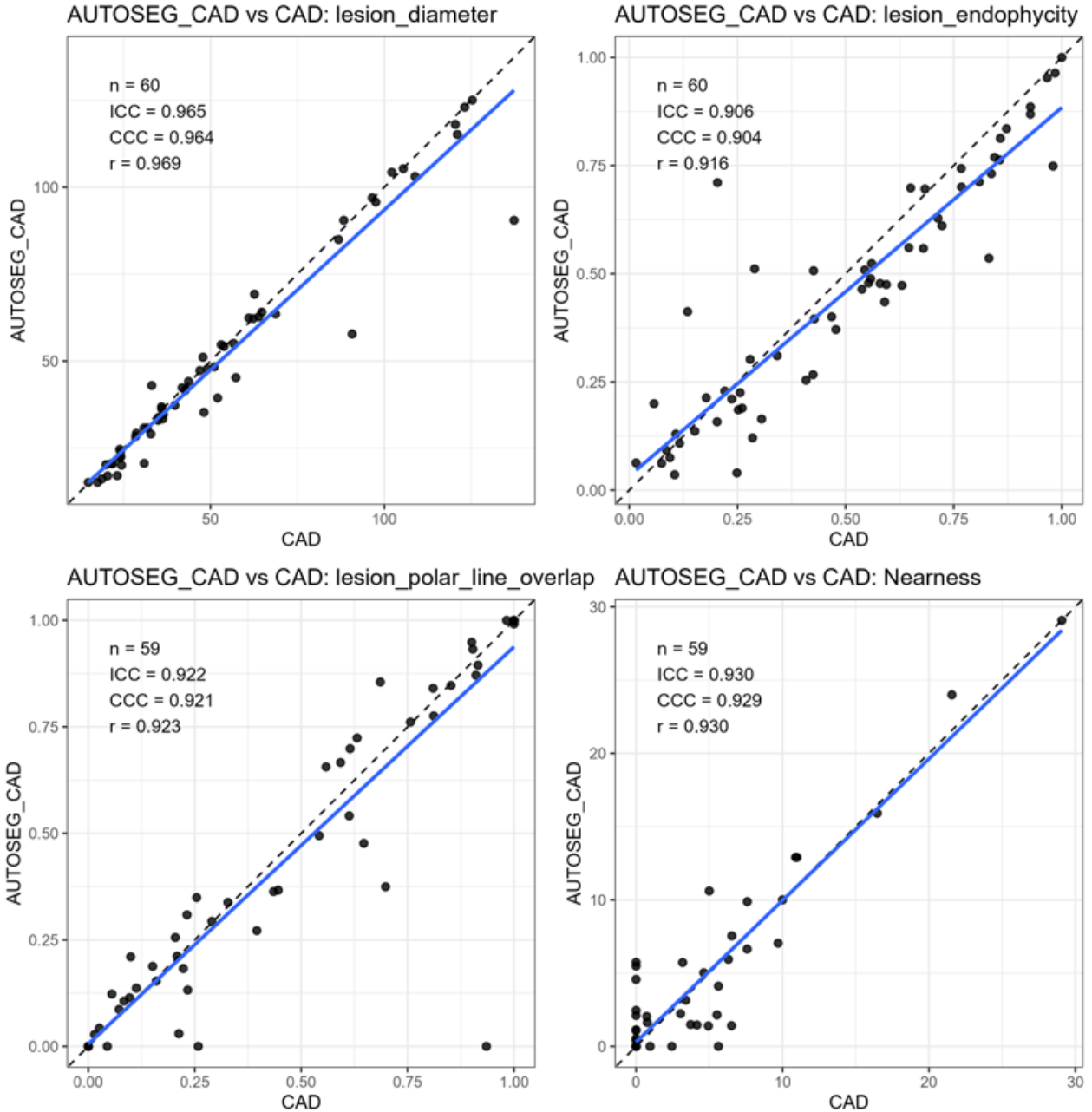


Figure 5.10: Scatter plots of RENAL components comparison of AUTOSEG_CAD vs CAD with identity line (dashed) and best fit line (blue solid line). From left to right, top to bottom: Diameter, Endophycity, Polar line Overlap and Nearness.

5.3.3 Agreement of Automated Methods and Human Raters

Weighted kappa agreement between automated methods

AUTOSEG_CAD and CAD achieved weighted kappa agreements values of 0.94, 0.71, 0.82 and 0.86 for total renal score, RENAL_R (diameter), RENAL_E (Endophycity), RENAL_N

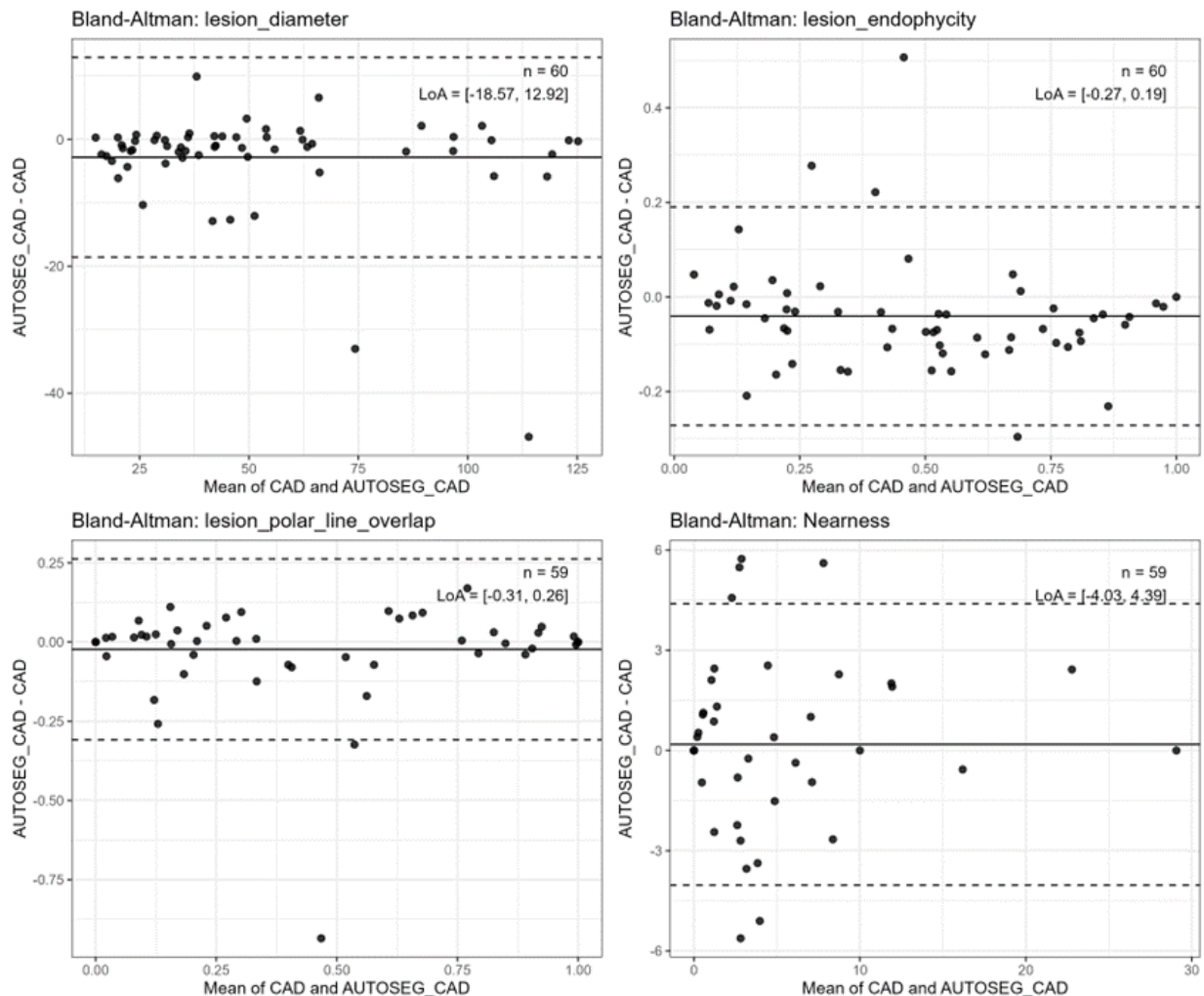


Figure 5.11: Bland-Altman analysis of RENAL components comparison of AUTOSEG_CAD vs CAD. From left to right, top to bottom: Diameter, Endophycity, Polar line Overlap and Nearness.

(Nearness) and RENAL_L (Location), respectively.

Weighted kappa agreement between automated methods and human median

Weighted kappa agreement with the human median RENAL total score was 0.56 for both CAD and AUTOSEG_CAD. In RENAL_R, CAD and AUTOSEG_CAD achieved an agreement with the human median of 0.93 and 0.90, respectively. For RENAL_E, CAD agreement

was 0.49, with AUTOSEG_CAD agreement at 0.48. Agreement with the human median RENAL_N was 0.23 and 0.19 for CAD and AUTOSEG_CAD, respectively. Lastly, for RENAL_L, the agreement of CAD was 0.28 and AUTOSEG_CAD at 0.35. Figures 5.12 and 5.13 shows representative CAD outputs of cases where the CAD RENAL total was the same as the human median RENAL total, and where CAD had the largest difference from the human median, respectively.

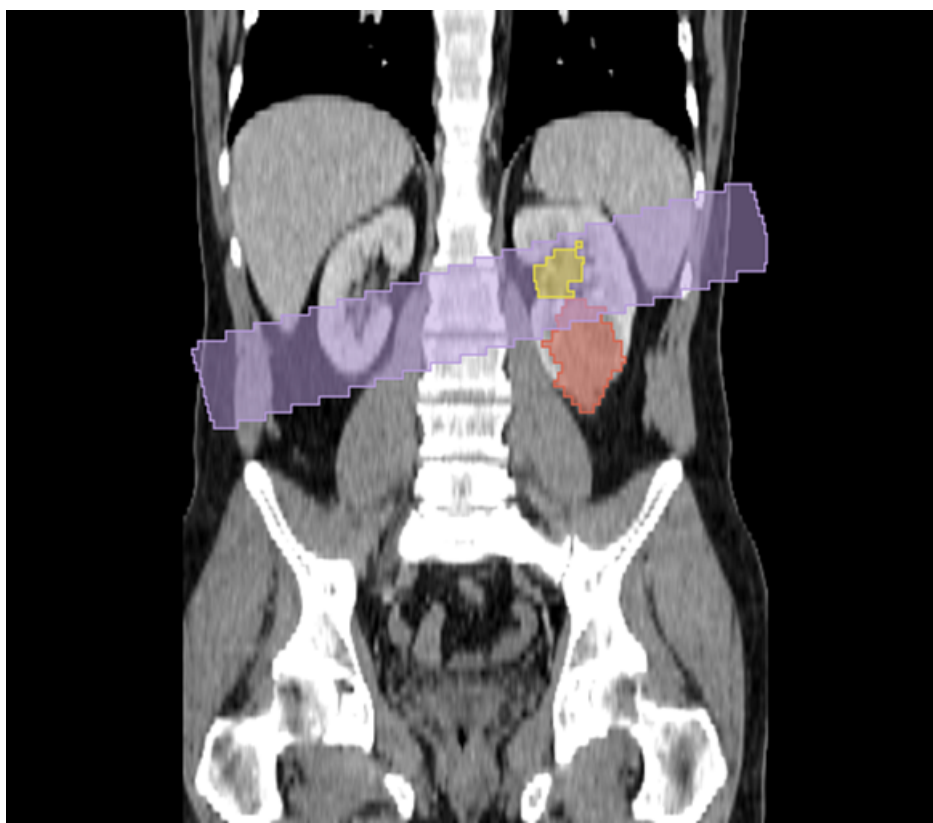


Figure 5.12: Example case with zero difference in total RENAL score compared to human median. RENAL total = 9.

Weighted kappa agreement across all raters

Comparing CAD methods against each individual human rater, weighted kappa agreement for the total renal score across all pairs ranged from 0.48 between both automated methods

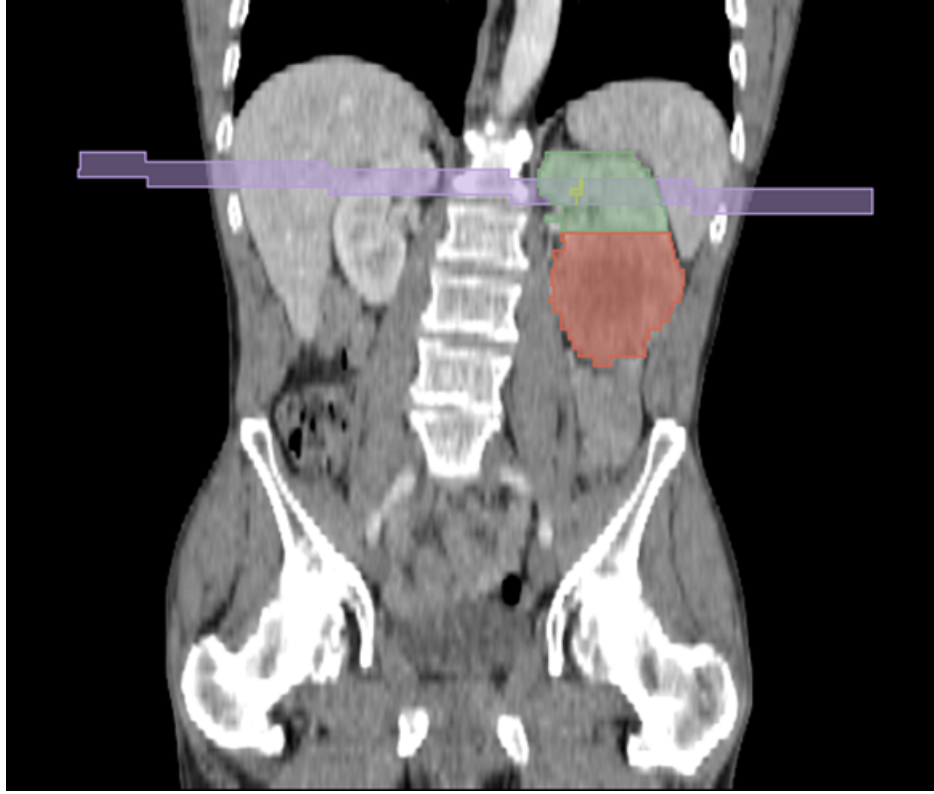


Figure 5.13: Example case with largest negative difference (-3) in total RENAL score compared to human median. CAD RENAL total = 6, human median RENAL total = 9.

and MML, with the highest agreement found between SKA and MML with 0.76. RENAL_R component agreement across all pairs ranged from 0.87 to 0.95. In the RENAL_E component, minimum kappa agreement was 0.32 found between raters ESD and MML, and between AUTOSEG_CAD and SKA. Max RENAL_E kappa agreement was found between human raters MML and SKA at 0.52. Highest agreement in RENAL_N was 0.49 between raters SKA and MML, while the lowest agreement was 0.13 between AUTOSEG_CAD and SKA. Lastly, in RENAL_L, the highest agreement was found between raters SKA and MML with 0.54 and lowest at 0.13 between CAD and SKA. Figure 5.14 shows heatmaps of the quadratic weighted kappa agreement scores for RENAL and its components across all pairs of raters.

Stratifying the weighted kappa analysis by RENAL_R tier demonstrates decreasing kappa

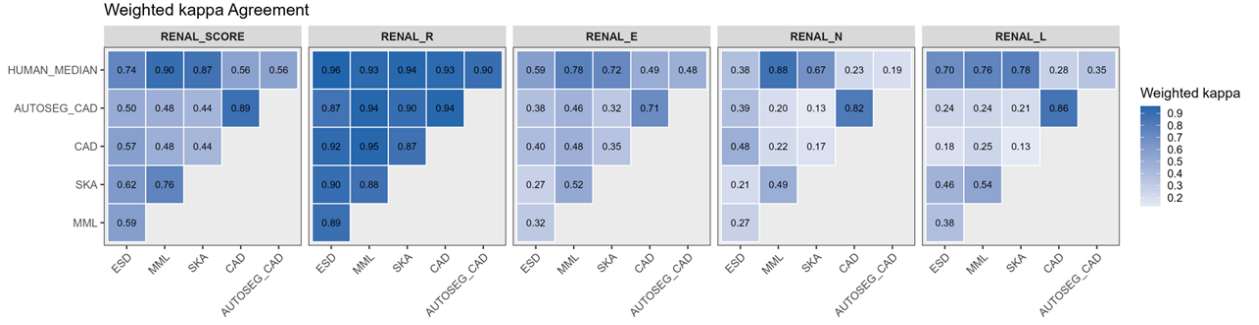


Figure 5.14: Total RENAL score and RENAL component heatmaps of quadratic weighted kappa agreement across all pairs of raters.

agreement values with increasing RENAL_R across automated methods and human raters.

Figure 5.15 shows heatmaps of the pair-wise quadratic weighted kappa agreement analysis stratified by RENAL_R.

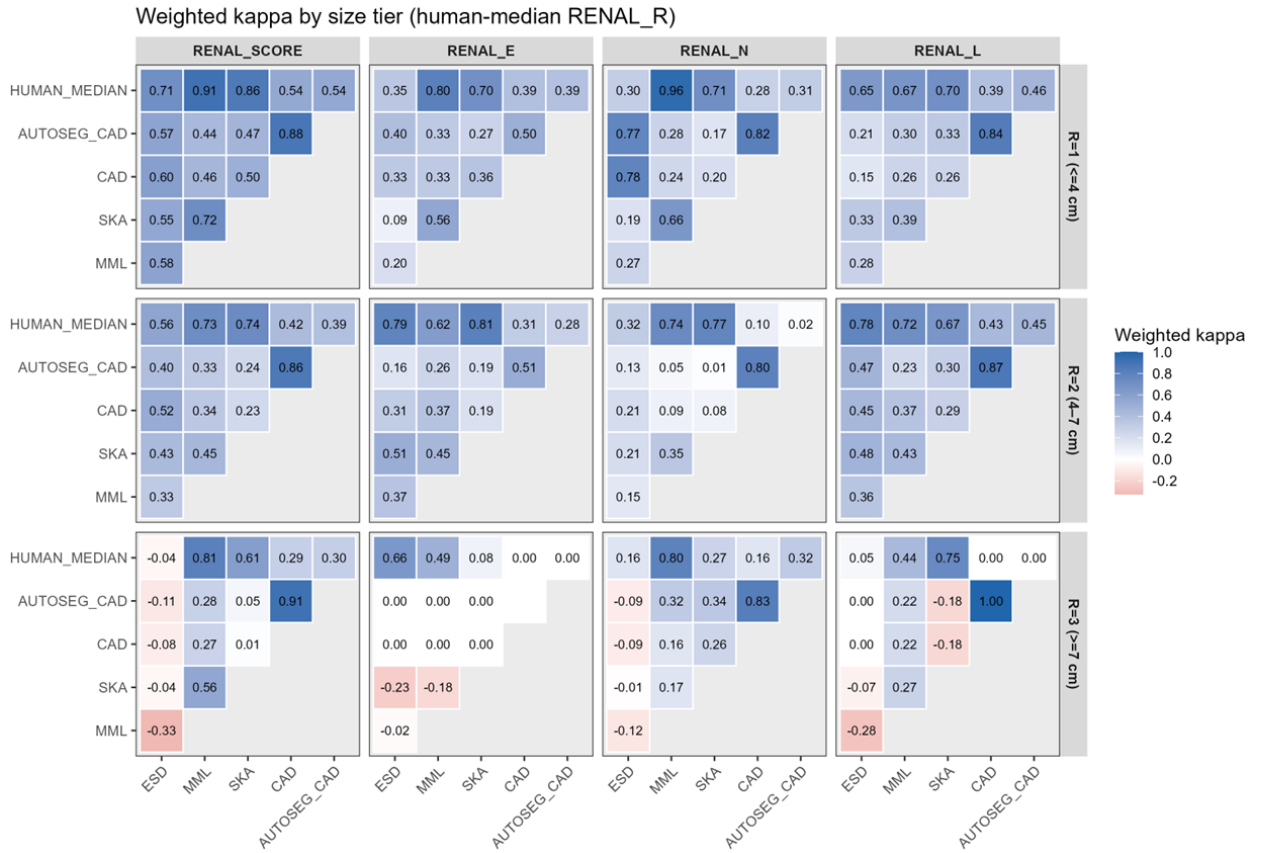


Figure 5.15: Pairwise Quadratic weighted Kappa agreement analysis stratified by RENAL_R.

5.3.4 Bland-Altman Analysis of Automated Methods vs Human Median

CAD vs Human Median

Bland-Altman analysis for CAD and human for total RENAL score and all components can be seen in Figure 5.16. Total RENAL score: bias = 0.68 ± 1.48 , LoA = $[-2.21, 3.58]$ with 10% of cases having a difference > 2 points. RENAL R: bias = -0.03 ± 0.26 , LoA = $[-0.54, 0.47]$ with 0% of cases having a difference > 1 point. RENAL E: bias = -0.30 ± 0.62 , LoA = $[-1.51, 0.91]$ with 0% of cases having a difference > 1 point. RENAL N: bias = 0.80 ± 0.94 , LoA = $[-1.03, 2.63]$ with 28.3% of cases having a difference > 1 point. RENAL L: bias = 0.22 ± 0.88 , LoA = $[-1.52, 1.95]$ with 8.3% of cases having a difference > 1 point.

AUTOSEG_CAD vs Human Median

Bland-Altman analysis for AUTOSEG_CAD and human for total RENAL score and all components can be seen in Figure 5.17. Total RENAL score: bias = 0.42 ± 1.51 , LoA = $[-2.54, 3.39]$ with 10.2% of cases having a difference > 2 points. RENAL R: bias = -0.07 ± 0.31 , LoA = $[-0.68, 0.55]$ with 0% of cases having a difference > 1 point. RENAL E: bias = -0.39 ± 0.59 , LoA = $[-1.54, 0.76]$ with 0% of cases having a difference > 1 point. RENAL N: bias = 0.78 ± 0.98 , LoA = $[-1.15, 2.71]$ with 30.5% of cases having a difference > 1 point. RENAL L: bias = 0.08 ± 0.86 , LoA = $[-1.59, 1.76]$ with 6.8% of cases having a difference > 1 point.

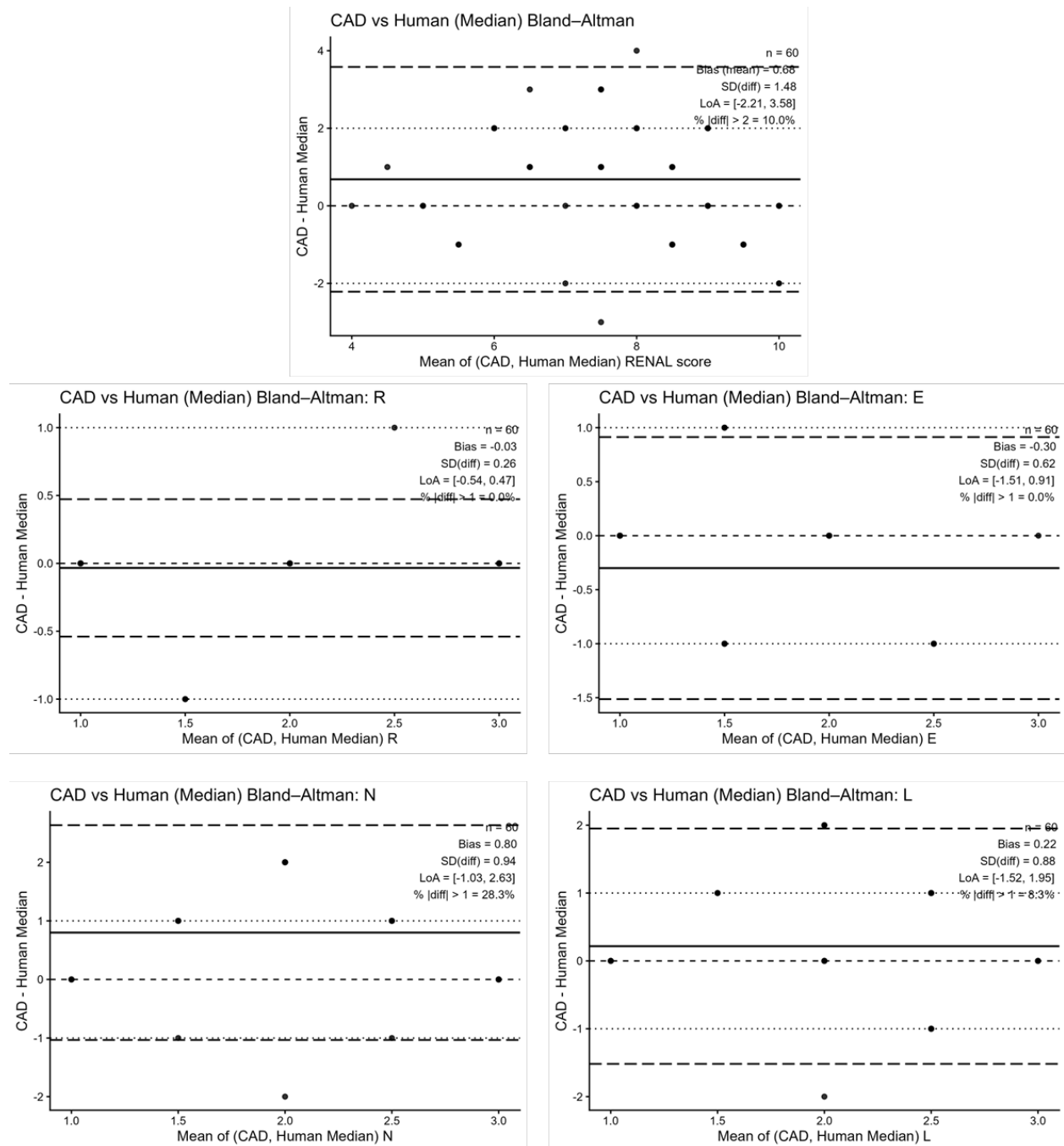


Figure 5.16: Bland-Altman analysis of RENAL score and RENAL components comparison between CAD and human median. From top to bottom, left to right: Total score, RENAL R, RENAL E, RENAL N, and RENAL L.

5.3.5 Within Variability Analysis

Figure 5.19 shows the within-variance coverage analysis for each CAD method. Figure 5.18 shows the LOO analysis done separately for CAD, AUTOSEG_CAD and RND_CAD

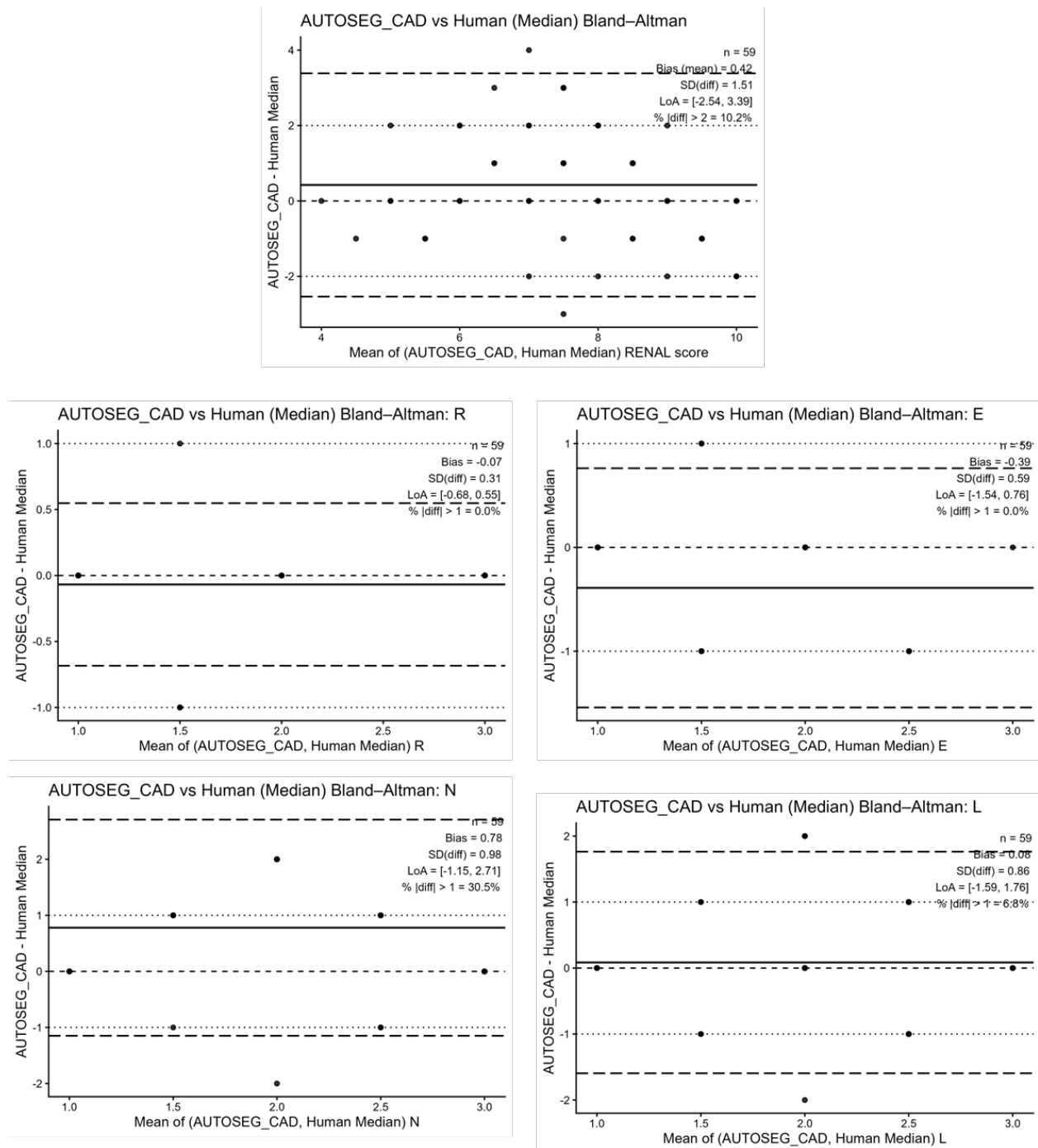


Figure 5.17: Bland-Altman analysis of RENAL score and RENAL components comparison between AUTOSEG_CAD and human median. From top to bottom, left to right: Total score, RENAL R, RENAL E, RENAL N, and RENAL L.

against human raters. On total RENAL score, the proportion of CAD RENAL scores within the human range was 0.583 ($CI_{95\%}[0.449, 0.709]$), 0.6 ($CI_{95\%}[0.465, 0.724]$) were within the $median \pm 1$, and 0.9 ($CI_{95\%}[0.794, 0.962]$) were within the $median \pm 2$. For AUTOSEG_CAD, the proportion of RENAL scores within the human range was 0.610 ($CI_{95\%}[0.474, 0.735]$), 0.64 ($CI_{95\%}[0.509, 0.764]$) were within the $median \pm 1$, and 0.898 ($CI_{95\%}[0.792, 0.962]$) were within the $median \pm 2$. On the RENAL_R component, proportion of CAD within human range and $median \pm 1$ was 1.0 ($CI_{95\%}[0.940, 1]$). For AUTOSEG_CAD, the proportion of cases within human range 0.983 ($CI_{95\%}[0.909, 1]$), while the proportion within $median \pm 1$ was 1.0 ($CI_{95\%}[0.939, 1]$). In the RENAL_E component, the proportion of CAD cases within human range was 0.883 ($CI_{95\%}[0.774, 0.952]$) and 1.0 ($CI_{95\%}[0.940, 1]$) for within $median \pm 1$. AUTOSEG_CAD proportions were 0.847 ($CI_{95\%}[0.73, 0.928]$) within human range and 1.0 ($CI_{95\%}[0.939, 1]$) within $median \pm 1$. On the RENAL_N component, proportion of CAD within human range and $median \pm 1$ was 0.867 ($CI_{95\%}[0.754, 0.941]$) and 0.717 ($CI_{95\%}[0.586, 0.826]$), respectively. For AUTOSEG_CAD, the proportion of cases within human range was 0.814 ($CI_{95\%}[0.691, 0.903]$), while the proportion within $median \pm 1$ was 0.695 ($CI_{95\%}[0.561, 0.808]$). Lastly, in the RENAL_L component, the proportion of CAD cases within human range was 0.7 ($CI_{95\%}[0.568, 0.812]$) and 0.917 ($CI_{95\%}[0.816, 0.972]$) for within $median \pm 1$. AUTOSEG_CAD proportions were 0.695 ($CI_{95\%}[0.561, 0.808]$) within human range and 0.932 ($CI_{95\%}[0.835, 0.981]$) within $median \pm 1$. All RENAL components score proportions within $median \pm 2$ were 1.0 for CAD ($CI_{95\%}[0.940, 1]$) and AUTOSEG_CAD ($CI_{95\%}[0.939, 1]$), given that the ordinal range of component scores is $[1, 3]$.

For both CAD and AUTOSEG_CAD, McNemar's test showed significant differences from RND_CAD in within variability proportions in total RENAL score within $median \pm 2$,

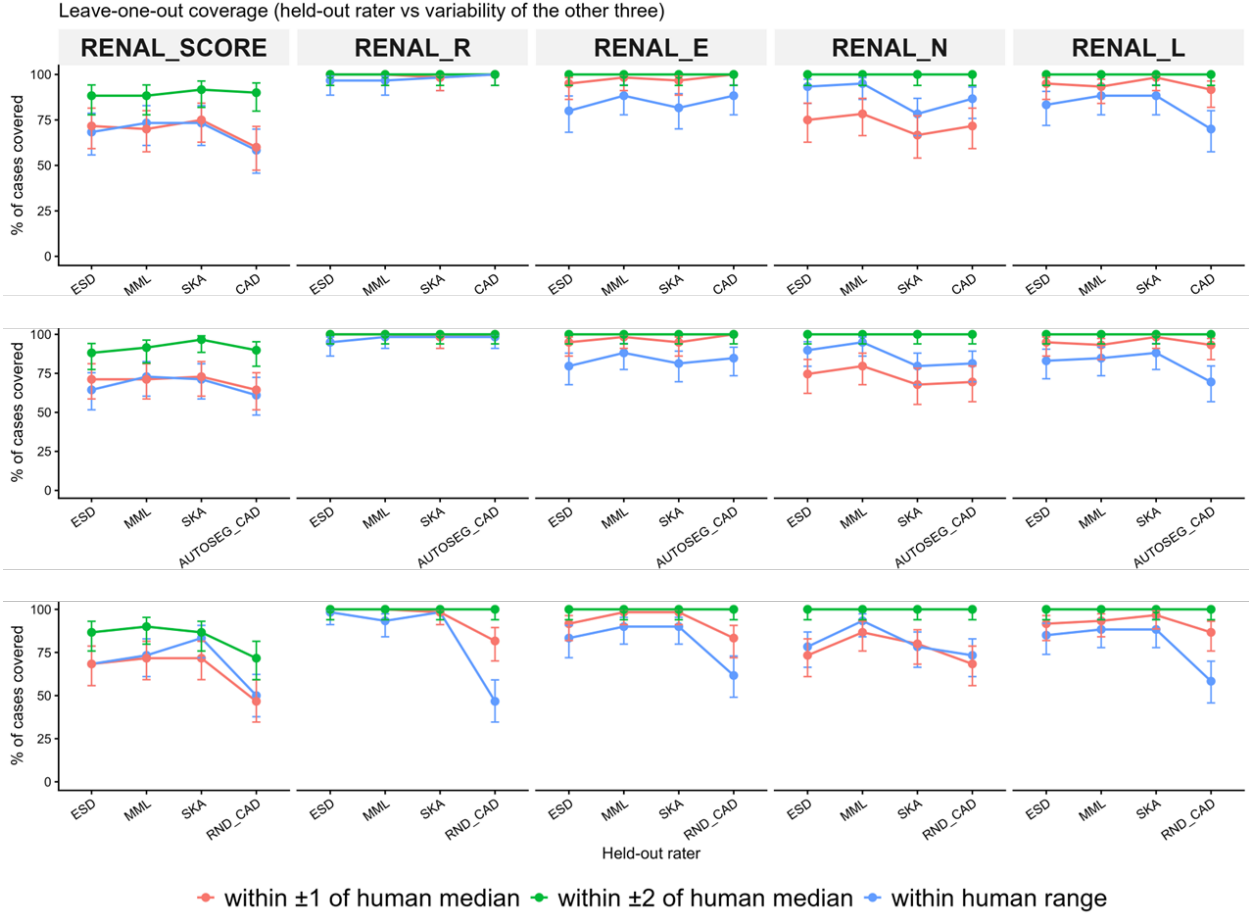


Figure 5.18: Leave-one-out within-variance coverage analysis. From top to bottom: CAD, AUTOSEG_CAD and RND_CAD.

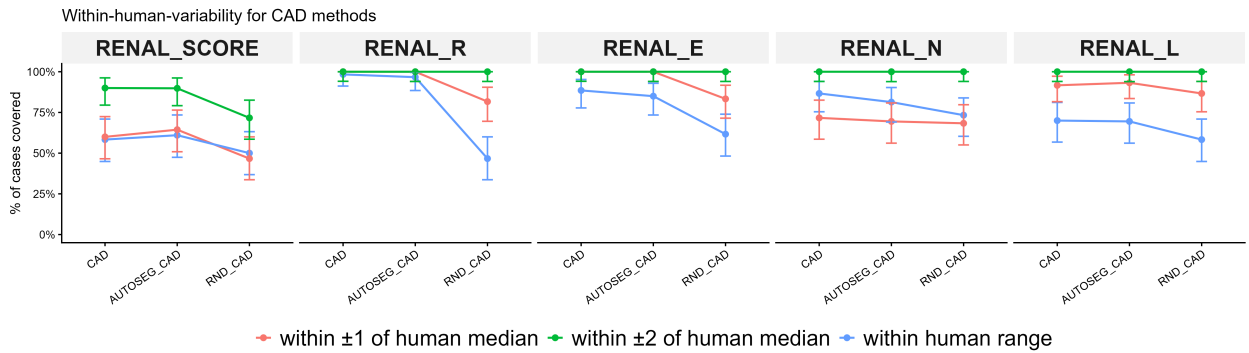


Figure 5.19: Within-variance coverage analysis for CAD methods.

and in Renal_R and Renal_E within human range and within *median* ± 1 . No significant differences were found in RENAL_N and RENAL_L for either CAD method compared to RND_CAD. Table 5.3 and Table 5.4 shows the McNemar’s test results for CAD and AUTOSEG_CAD, respectively.

Table 5.3: McNemar’s test results comparing within-variability CAD proportion ($\hat{p}_{cad}$) to RND_CAD ($\hat{p}_{rnd}$). BH-adjusted p-values. $n = 60$

Target	Metric	$\hat{p}_{cad}$	$\hat{p}_{rnd}$	Δ	p_{BH}	Sig.
RENAL_SCORE	within human range	0.583	0.500	0.083	0.427	No
	within median ± 1	0.600	0.467	0.133	0.212	No
	within median ± 2	0.900	0.717	0.183	0.0256	Yes
RENAL_R	within human range	1.000	0.467	0.533	2.84e-07	Yes
	within median ± 1	1.000	0.817	0.183	0.00401	Yes
	within median ± 2	1.000	1.000	0.000	–	–
RENAL_E	within human range	0.883	0.617	0.267	0.00401	Yes
	within median ± 1	1.000	0.833	0.167	0.00480	Yes
	within median ± 2	1.000	1.000	0.000	–	–
RENAL_N	within human range	0.867	0.733	0.133	0.0650	No
	within median ± 1	0.717	0.683	0.033	0.702	No
	within median ± 2	1.000	1.000	0.000	–	–
RENAL_L	within human range	0.700	0.583	0.117	0.212	No
	within median ± 1	0.917	0.867	0.050	0.446	No
	within median ± 2	1.000	1.000	0.000	–	–

5.4 Discussion

The SimpleMind KG for kidney and lesion segmentation was able to produce accurate segmentations on the majority of cases with only two cases with lesion segmentation below 0.50 DSC. The utility and accuracy of the automated segmentation is further demonstrated with the RENAL nephrometry measurements comparisons between AUTOSEG_CAD, which utilizes the automated segmentations, against CAD, which utilizes the manual annotations

Table 5.4: McNemar’s test results comparing within-variability AUTOSEG_CAD proportion ($\hat{p}_{autoseg_cad}$) to RND_CAD ($\hat{p}_{rnd}$). BH-adjusted p-values. $n = 59$

Target	Metric	$\hat{p}_{autoseg_cad}$	$\hat{p}_{rnd}$	Δ	p_{BH}	Sig.
RENAL_SCORE	within human range	0.610	0.508	0.102	0.353	No
	within median ± 1	0.644	0.475	0.169	0.158	No
	within median ± 2	0.898	0.712	0.186	0.00558	Yes
RENAL_R	within human range	0.983	0.458	0.525	2.84e-07	Yes
	within median ± 1	1.000	0.814	0.186	0.00401	Yes
	within median ± 2	1.000	1.000	0.000	–	–
RENAL_E	within human range	0.847	0.610	0.237	0.00480	Yes
	within median ± 1	1.000	0.831	0.169	0.00480	Yes
	within median ± 2	1.000	1.000	0.000	–	–
RENAL_N	within human range	0.814	0.746	0.0678	0.353	No
	within median ± 1	0.695	0.695	0.0000	1.000	No
	within median ± 2	1.000	1.000	0.0000	–	–
RENAL_L	within human range	0.695	0.576	0.119	0.212	No
	within median ± 1	0.932	0.864	0.0678	0.341	No
	within median ± 2	1.000	1.000	0.0000	–	–

obtained from the C4KC-KITS dataset. AUTOSEG_CAD and CAD produced values > 0.90 across all agreement, reliability and correlation metrics applied, and across all RENAL components. Bland-Altman analysis demonstrated a bias of $-2.82mm$ or $0.282cm$ for the lesion diameter measurements, and, based on the LoA, 95% of differences are within $-1.9cm$ and $1.3cm$. While the upper bound is within our clinical threshold of $1.5cm$, the lower bound lies outside by $0.4cm$. Endophycity and Polar Line overlap demonstrated LoA outside our established bounds of ± 0.10 , however, they are within ± 0.50 , where the RENAL scoring system differentiates between a score of 2 and 3 (see Table 5.1). A bias of $0.18mm$ was obtained for the Nearness measurement, however, the upper and lower LoA lie outside our clinical target of $\pm 1.5mm$ with $-4.03mm$ and $4.39mm$, respectively. Despite the LoA lying outside our established limits, testing for equivalence using these same limits produced

significant results based on the TOST for all components, indicating that both groups are equivalent.

Based on Kappa agreement benchmarks from Landis & Koch [64], lowest Kappa agreement between automated methods and human medians, seen on AUTOSEG_CAD RENAL_N is considered slight ($0 - 0.20$). Kappa agreements considered fair ($0.21 - 0.40$) are seen for both CAD and AUTOSEG_CAD on RENAL_L, and RENAL_N for CAD. RENAL_E and total RENAL score agreements are considered moderate ($0.41 - 0.60$) for both CAD and AUTOSEG_CAD. RENAL_R for both methods demonstrated the highest Kappa agreements within what is considered the "almost perfect" range ($0.81 - 1.00$). This range of Kappa agreement is also seen when comparing CAD and AUTOSEG_CAD on all components, except for RENAL_E where the Kappa agreement produced 0.71, considered "substantial agreement" ($0.61 - 0.80$). Agreement results between automated methods and the human median indicate superior performance on RENAL_R and RENAL_N, which is confirmed by Bland-Altman analysis. Bland-Altman analysis of both automated methods against the human median showed that no differences larger than 1 were found for RENAL_R and RENAL_E.

Kappa agreement between automated methods and human raters, and between each pair of human raters, produced "almost perfect" agreement for all pairs in the RENAL_R component. On total RENAL score and remaining components, the highest Kappa agreement between automated methods and human raters was 0.57 on Total RENAL score and 0.48 on RENAL_N with rater ESD, while the lowest agreement was 0.13 for AUTOSEG_CAD and SKA on RENAL_N, and CAD with SKA on RENAL_L. Highest among human raters (excluding RENAL_R) was found on the total RENAL score with 0.76 between SKA

and MML, while the lowest was 0.21 between SKA and ESD in RENAL_N. While this demonstrates that human raters tend to have higher agreements with each other compared to with the automated methods, individual pairwise comparisons indicate rater dependency. As an example, in RENAL_N, human rater ESD had higher agreements with CAD and AUTOSEG_CAD (0.39 and 0.48), compared to human raters SKA and MML (0.21 and 0.27). Additionally, excluding total RENAL score and RENAL_R, the highest agreement among human raters was 0.54 on RENAL_L, considered "moderate", indicating a lack of agreement among humans. Kappa agreement analysis stratified by RENAL_R suggests that raters (both human and automated) have less agreement for larger lesions with Kappa values reducing to 0.0, as well as demonstrating disagreements among raters with negative Kappa values when lesions are 7cm or more in diameter. Based on previous works described in section 5.1.3, the agreement levels obtained among humans, and between automated and humans are within the ranges observed in previous works [85, 57, 49].

Given the discrepancies among human raters and the lack of a real truth measurement for the nephrometry components, the LOO analysis allows us to assess if the automated methods can be considered as behaving as another human rater. Qualitative assessment from Figure 5.18 demonstrates that CAD and AUTOSEG_CAD exhibit coverage within overlapping confidence intervals of the humans, with RENAL_L showing the least amount of overlap; as opposed to RND_CAD, where overlap with human raters is seen only on RENAL_N. Direct comparison of variance coverage of CAD and AUTOSEG_CAD with RND_CAD, also demonstrates higher coverage for CAD and AUTOSEG_CAD, however, with overlapping confidence intervals. Significant differences from RND_CAD were found in RENAL_R and RENAL_E, but not in RENAL_N and RENAL_L, for both CAD and AUTOSEG_CAD.

From the analysis of the results, we found this lack of significant differences from random in RENAL_N and RENAL_L, and lower CAD and AUTOSEG_CAD Kappa agreements being more prevalent on RENAL_N and RENAL_L. These are nephrometry components dependent on the UCS. The automated segmentation of the UCS relies on an intensity based thresholding. Dependency on image intensities can impact the consistency on the segmentation since HU intensity values are dependent on scanner protocols and, in the case of the data utilized, contrast uptake. Thus, accuracy of the UCS segmentation is expected to vary among different scans. The lack of a defined UCS boundary may explain the lack of performance of the automated methods for measuring the lesion nearness, and defining the polar lines. This is further supported by qualitative assessment of outputs as seen in Figure 5.13 where the volume of the lesion invaded portions of the kidney where the UCS is located. This caused the automated method to produce an inaccurate UCS segmentation, overestimating the lesion distance to the UCS and underestimating the extent of the polar lines, thus underestimating the overall score compared to the humans. The lack of agreement among human raters in RENAL_E (min: 0.27, max: 0.52), may be due to the lack of a defined healthy kidney boundary, where the boundary of the kidney is traced as it would be if the lesion were not present. Thus, different raters may operate using different boundaries causing variation in the endophytic/exophytic measurement.

5.5 Conclusion

We have implemented an automated nephrometry scoring system based on previous works [47, 1, 120] in the SimpleMind Software Environment with updates to polar line rotation to

account for the axis of the kidney. Rather than ad-hoc scripts, the nephrometry scoring system is implemented in a human-interpretable Knowledge Graph within SimpleMind for general usage and reproducibility. We provided a methodology for comparing the automated systems with human raters and random nephrometry scoring to validate current and future automated nephrometry scoring systems. Automated scoring methods demonstrate agreement levels from "slight" to "almost perfect" agreement with human raters. Similarly, agreement among human raters ranged from "fair" to "almost perfect" as seen in previous works. We found that automated scoring methods are consistent with reader variability for RENAL_R and RENAL_E. While RENAL_N and RENAL_L within variability $\hat{p}$ were numerically larger from automated methods compared to random scoring, these were not found to be statistically significant potentially due to limited sample sizes.

Future works will focus on improving current systems for robust UCS segmentations and healthy kidney boundaries in an effort to improve automation performance, as well as human inter-rater agreements. Validation analysis will also be expanded to include continuous measurements of the nephrometry components and increased sample sizes for increased statistical power. Additional validation can include assessment of the nephrometry features' predictive power for pathology and surgical outcomes, as well as an analysis of feature importance for prediction. Using the SimpleMind Knowledge graph and available agents, the automated nephrometry system can be extended to include other features from additional nephrometry scoring methods.

Chapter 6

The SimpleMind Infrastructure for Predictive Modelling and Hyperparameter Tuning

6.1 Introduction

In Chapter 2 we covered the SM infrastructure for biomedical image segmentation. However, the flexibility of the agent-based system allows for the integration of predictive modelling, such as Logistic Regression, from features extracted from images or any other user provided source. Many steps are involved when developing Machine Learning (ML) based prediction models, which vary depending on the task. We will focus on the following pipeline: (1) Feature Preprocessing, (2) Feature Selection, (3) Data Resampling, and (4) Model Selection and Training.

Feature Preprocessing

Preprocessing involves methods to prepare the data for training a model. Here we focus on data preparation through normalization. Extracted features can vary in scale. Features with larger ranges can diminish the impact of significant features with smaller ranges. This is evident when working with features from radiomics or nephrometry since the scale of the individual feature values may vary by orders of magnitude.

Feature Selection

Feature selection involves choosing those that are most discriminative for the task. Non-informative, low-variance features and redundant/correlated features can be removed when unsupervised selection methods are used. Supervised methods use the outcome data to determine which features best correlate with the outcome. Feature selection has the benefit of reducing the dimensionality of the data, thus reducing the amount of computation and storage, with the goal of improving the model's performance by removing unnecessary noise.

Data Resampling

Real world data is often imbalanced which can lead to inadequate model performance or robustness due to insufficient representative samples for training. To mitigate this, data can be resampled either through under-sampling or over-sampling the majority or minority classes, respectively, in the data set during training.

Model Selection and Training

A diverse array of ML models are available with different strengths and weaknesses. Careful consideration must be given when choosing a model since it will be dependent on the data, outcomes, preprocessing and feature selection, given that not all models are compatible with every preprocessing and selection method. Thus, model synergy with upstream methods must be considered. Training hyperparameters as well need to be tuned to achieve the best performance for the given task.

6.2 Using Predictive Models with SimpleMind

6.2.1 Agent Configuration

Predictive modelling is integrated into SM through a single agent incorporating the steps listed in Section 6.1. The yaml configuration for this agent can be seen in Listing 6.1. It uses a set of features and outcomes as .json files, formatted as nested dictionaries of feature/outcome names containing a dictionary of the index of the sample with their value for that feature/outcome (see Figure 6.1a for the formatting of features and Figure 6.1b for the formatting of outcomes). A specific outcome can be chosen by specifying the key name on the *response_variable* attribute.

To specify the steps in the ML pipeline, users provide a .yaml settings file describing the *preprocessing_settings*, *feature_selection_settings*, *resampler_settings*, and *ml_settings* (see Listing 6.2). Within each of these keys, users can select their respective class of preprocessor, feature selector, resampler, and model from the Scikit-Learn [91]

```
ml_model:
  entrypoint: "predictive_modeling-ml_agent"
  runner: local
  attributes:
    features: "./all_features.json"
    outcomes: "./outcomes.json"
    response_variable: "malignant"
    ml_settings_yaml: ./settings.yaml
    learn: True
    resample: True
    n_splits: 10
    n_repeats: 5
    alpha: 1
    beta: 0.05
    lambda_feat: 0.1
    save_pipe: True
    working_dir: ./working_dir
    output_name: auc
    output_type: dictionary
    output_dir: ./output_dir
    file_based: True
```

Listing 6.1: Example YAML configuration for the predictive modelling agent.

and Imbalanced-Learn libraries [67]. Required input arguments for each respective class are listed within the same parent dictionary with their own unique key as specified in Scikit-Learn or Imbalanced-Learn. In the next subsection, we will review the internal handling of the settings file to construct and fit a prediction model pipeline.

6.2.2 Predictive Modelling Agent Operation

The predictive modelling agent is designed with modularity, and future-proofing in mind, as well as simplifying model construction and usage for non-developers. The agent automatically handles searching for the correct class provided in the settings file as well as cleaning input arguments to ensure those belonging to the class are correctly used.

```
{
  "feature_1": {
    "0": 1.2
    "1": 2.3
    "2": 1.7
  }

  "feature_2": {
    "0": 5.5
    "1": 3.2
    "2": 4.4
  }

  "feature_3": {
    "0": 10.0
    "1": 9.5
    "2": 5.2
  }
}
```

(a) Example feature file in .json format.

```
{
  "outcome_1": {
    "0": 0
    "1": 0
    "2": 1
  }

  "outcome_2": {
    "0": 1
    "1": 1
    "2": 1
  }

  "outcome_3": {
    "0": 1
    "1": 0
    "2": 1
  }
}
```

(b) Example outcome file in .json format.

Figure 6.1: JSON configuration examples.

Model Fitting

The predictive modelling agent makes use of the *pipeline* module from the imbalanced-learn library. This allows aggregation of steps (preprocessing, resampling, feature selection and model fitting) into a single object or pipe. While the general design of SM is to have separate agents each with their own specific function, the single object allows for simplification of development and prevents data leakage since this ensures that all steps are fitted with the same provided data, whether fitting with the whole training data or when using cross-validation. This design also allows users to add input arguments to simultaneously tune multiple classes within a single settings file (details in section 6.3). In addition to the steps provided in the settings file, prior to preprocessing, the agent also applies an imputer to fill in missing values

```

seed: 2102025

preprocessing_settings: {preprocessor: StandardScaler,
random_state: 2102025, with_mean: false,
  with_std: true}

feature_selection_settings: {feature_selector:
SelectKBest, random_state: 2102025,
  selection_first: false, score_func:
mutual_info_classif}

resampler_settings: {resampler: RandomOverSampler,
random_state: 2102025, sampling_strategy: all,
  shrinkage: 0.0}

ml_settings: {model: LogisticRegression, random_state:
2102025, penalty: elasticnet,
  C: 100.0, solver: saga, l1_ratio: 0.5, dual: false,
tol: 1.0e-05, fit_intercept: true,
  intercept_scaling: 5.0, class_weight: balanced,
max_iter: 300}

```

Listing 6.2: Example YAML configuration for the ML pipeline.

with the mean and a variance threshold to remove features with zero variance.

Pipe fitting occurs in two ways: (1) Via a repeated stratified K-fold cross-validation where users can provide their desired folds (*n_splits* in Listing 6.1) and repeats (*n_repeats* in Listing 6.1), results from the cross-validation are then provided as inputs to a fitness function to provide performance information (see section 6.3). (2) A single fit using the full dataset. This is intended to fit the pipe for test set inference and general practical usage. This fitting also saves a .json of the selected features under the same format shown in Figure 6.1a.

6.3 Hyperparameter Optimization Using a Genetic Algorithm

Developing a ML model involves the selection and tuning of different methods that must synergize to produce optimal performance. This typically requires user knowledge and experience, along with trial and error, to determine the best combination of methods and hyperparameters for the task. Considering all variations of preprocessing, feature selection, resampling and ML models (each with their own unique hyperparameters), the selection and tuning task can become an arduous and time-consuming process. Done manually, a human user is only capable of a limited amount of combinations within a reasonable time-frame. So we propose automating this method selection and fine-tuning process to develop ML models for renal lesion pathology and surgical outcomes.

6.3.1 Genetic Algorithms

A Genetic Algorithm (GA) is an optimization method that takes inspiration from evolution of biological systems in nature. In nature, species populations undergo changes in response to environmental pressures. These changes originate from mutations in the genetic pool of the population. Individuals with deleterious mutations tend to die off, while those with beneficial mutations, suited for their environment, survive and propagate their genes to the next generation. In optimization tasks, we can find solutions by mimicking this behavior. Different hyperparameters in a pipeline can be encoded into a genetic code, in the form of a binary string, from which a population of individuals with different hyperparameter combinations can be generated. Each unique pipeline (set of hyperparameters) can then be

tested for their fitness and top performers can continue to propagate, mutate and cross-over with other individuals [60]. This methodology allows for GAs to keep top performers but maintain a diverse population that reduces the risk of overfitting by becoming trapped in a local minimum or maximum. Due to data limitations, the best performer might not be the best in generalizing beyond the training data, so maintaining diversity is important. The ability of GAs to mutate and cross-over individuals allows the optimization to escape a local minima or maxima, and explore other potentially more suitable solutions. However, these advantages come at the cost of increased computation since, in essence, GAs operate as a more sophisticated form of trial-and-error, where every single pipeline in the population must be trained and tested for fitness. Depending of the type pipeline, some models may be too complex and slow, making them impractical to train repeatedly. Thus, users might have to limit the number of individuals and generations which can in turn limit the search space.

6.3.2 The SimpleMind GA Implementation

The steps involved in a SimpleMind Genetic Algorithm evolution are summarized in Figure 6.3. SimpleMind’s Genetic Algorithm is implemented in its Knowledge Graph Learning and Optimization (KNoLO) module. In KNoLO, users can provide an uninstantiated KG with hyperparameters that they wish to tune under the *apt_list* format. Listing 6.3 shows an example of the formatting for tuning the ML settings portion of a predictive modelling agent pipeline. In this example, we are tuning the type of model class and the penalty (model regularization to handle overfitting). The *value* key contains the default value, while the *apt_list* is the list of parameter options we wish to try and test.


```

ml_settings:
  model:
    value: LogisticRegression
    apt_list:
      - LinearSVC
      - LogisticRegression
      - RandomForestClassifier
      - HistGradientBoostingClassifier

  penalty:
    value: l2
    apt_list:
      - l1
      - l2

```

Listing 6.3: Example YAML configuration for KNoLO optimization.

KNoLO encodes the *apt_lists* into a binary string to represent each hyperparameter combination. Using the *apt_lists* from the example in Listing 6.3, the model contains four options, this can be represented by a bit-string with a length of two bits, while the penalty can be represented by a single bit, thus, the genetic code, or chromosome, can be represented by a 3-bit string (Figure 6.2) for $2^3 = 8$ possible combinations. In these settings we can adjust the *seed* to determine the initial random population and maintain reproducible results. We have the evolution hyperparameters such as *max_gen*, which indicates the max generations the GA should run for, *overall_pop_size*, which determines the population size per generation, *overall_survivor_pop*, the number of chromosomes that get propagated to the next generation, and *overall_t_size*, which sets the overall tournament size. During runtime, once the fitness for all chromosomes in a generation have been computed, tournaments are held to determine which chromosomes will continue to the next generation. *overall_t_size* determines how many chromosomes of the population compete in a given

tournament from which a single chromosome will be added to the survivor population. The number of tournaments held is determined by *overall_survivor_pop*.

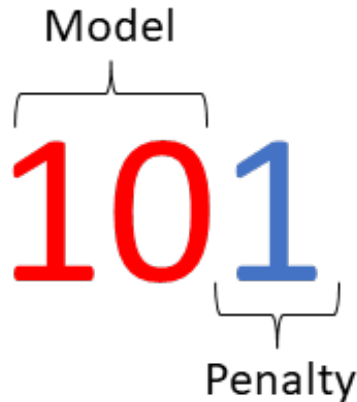


Figure 6.2: Example bit-string chromosome encoding model selection and penalty representing the *apt_lists* from Listing 6.3.

KNoLO also requires a settings file for running the GA optimization as seen in Listing 6.4.

The *operations* key lists the operations to be performed on the population after all fitness values have been computed. These are performed in the order listed. We can see from Listing 6.4 the *operations* dictionary has the *tourn* operation listed first, referring to tournament which we just described. Following the tournament, the remaining survivor population is then shuffled, mutated, crossed over, and repopulated, where random chromosomes are added to the survivor pool to complete the overall population size for the next generation and maintain a level of diversity.

Crossover and mutation rates are determined by *cspb* and *mutpb*, respectively. During crossover, the GA mimics a form of sexual reproduction where two chromosomes are split from two random positions in the bit-string. The sections that were split are then traded or

```

seed: 18122025

##### (2) Basic evolution hyperparameters #####
max_gen: 20

overall_pop_size: 20
overall_survivor_pop: 16
overall_t_size: 10

operations:
  - "tourn"
  - "shuffle"
  - "mutation"
  - "crossover"
  - "repop"

### (B) Crossover and Mutation Rates ###
cxpb: 0.1
mutpb: 0.15
p_mut_per_bit: 0.2

```

Listing 6.4: Example YAML configuration for the GA optimizer.

crossed-over between them forming two new chromosomes. *mutpb* determines the probability of any chromosome to undergo a mutation. If a chromosome passes the probability check then each bit in the chromosome has a probability of being flipped given by *p_mut_per_bit*.

6.3.3 Non-Functional Chromosomes

Optimization through the GA is partially driven by probabilities (given random repopulation, mutation and crossover rates) that determine what hyperparameters will be used in a given chromosome. This can produce certain combinations of hyperparameters that are non-functional, which the GA is not capable of identifying until after the chromosomes are run and their fitness is computed. For example, the KNoLO configuration in Listing 6.5 for

SimpleMind genetic algorithm evolution cycle

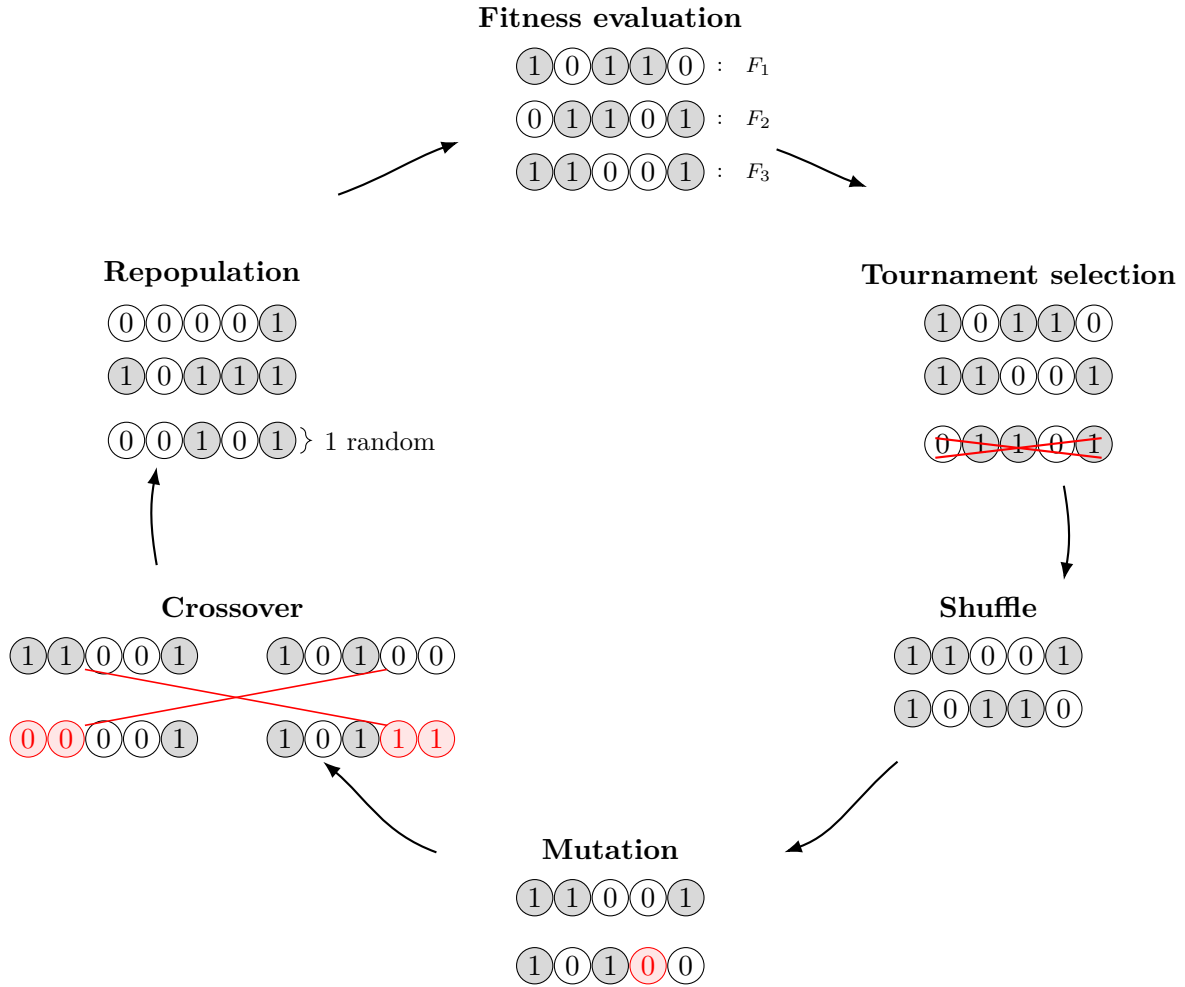


Figure 6.3: Representation of the SimpleMind Genetic Algorithm cycle showing fitness evaluation, tournament selection, shuffle, mutation, crossover, and repopulation.

optimizing a Logistic Regression model, where we are optimizing the *dual*, *penalty*, and *solver* hyperparameters with *apt_list* of length 2, 4, and 4 respectively. A chromosome bit length of 5 or 32 possible hyperparameter combinations. However, *dual = True* only works when *penalty = l2* and *solver = liblinear*. Thus, when considering only the chromosomes where *dual = True*, only 1/16 of chromosomes will be functional. Considering the selection of either *dual = True* or *dual = False*, 17/32 chromosomes will be functional, if we assume that *dual = False* will always produce a functional chromosome. Also, the penalties and solvers as well are not all compatible with each other. Accounting for the penalty and solver incompatibilities, the correct fraction of functional chromosomes, given our example, is $11/32 \approx 34\%$. Given a random population of chromosomes, $\approx 66\%$ will be non-functional chromosomes.

```
ml_settings:
  model: LogisticRegression
  dual:
    value: False
    apt_list: [True, False]

  penalty:
    value: l2
    apt_list: [None, l1, l2,
elasticnet]

  solver:
    value: liblinear
    apt_list: [lbfgs, liblinear, newton-
cg, saga]
```

Listing 6.5: Example YAML configuration for KNoLO optimization of a Logistic Regression Model.

If propagated, these chromosomes occupy the population space, limiting the diversity, and

affecting functional chromosomes through crossover and mutation. In addition, functional chromosomes themselves can potentially become non-functional through a single point mutation. These factors impact the GA search space and potentially trap the optimization process in a local maximum.

Reducing Non-Functional Chromosomes

As described in Section 6.3.2 and seen in Listing 6.4, the SM GA allows users to adjust the population settings that influence the evolution of chromosomes. We can adjust these settings so as to reduce the probability of non-functional chromosomes. If the approximate fraction of expected non-functional chromosomes is known, we can adjust the tournament size, *overall_t_size*, to be $> \frac{1}{fraction_non_functional}$. The expected probability of selecting at least one functional chromosome for the tournament becomes 1. The GA tournaments only propagate the best fitness, all non-functional chromosomes in that tournament will not propagate, reducing the influence in the evolution. However, this requires an estimation of the expected fraction of non-functional chromosomes, either by manually computing the failure combinations or performing preliminary GA runs. This strategy also promotes elitism during the evolution which can cause optimization to converge early.

6.3.4 Phenotypic Clones and Junk DNA

In nature, the genotype of an organism refers to the genetic makeup of the organism while the phenotype refers to the observable expression of the genetic material [88]. Two organisms carrying different genetic materials may express them the same way depending on where the difference occurs and other factors. This can occur through mutations in non-coding strands

of DNA that do not impact the organism’s phenotype. It is known that non-coding DNA offers functions outside of protein synthesis. In the past, this DNA was considered to be junk DNA offering no function [94]. This concept of junk DNA is an issue that can affect the GA optimization when working with predictive models. Consider the following example in Listing 6.6.

```
ml_settings:
  model:
    value: LogisticRegression
    apt_list: [LogisticRegression,
RandomForestClassifier]

  penalty:
    value: l2
    apt_list:[None, l1, l2,
elasticnet]

  max_features:
    value: sqrt
    apt_list:[sqrt, log2, None, 10]
```

Listing 6.6: Example YAML configuration for KNoLO optimization of two models with distinct hyperparameters.

Here we are setting the GA to optimize between the model selection (*LogisticRegression* or *RandomForestClassifier*, represented by a single bit) and two hyperparameters, *penalty* and *max_features*, with 4 options each, which is represented with 2 bits each, for a total chromosome length of 5 bits. The issue lies in that the *penalty* parameter only applies to the *LogisticRegression* model, while the *max_features* parameter only applies to the *RandomForestClassifier* model. Thus, in this example, regardless of the model selected, there will always be 2 bits (40% of the chromosome) that will not affect the chromosome’s performance, in essence, junk DNA. If the chromosome undergoes a mutation in a non-coding

bit, it will produce a different genotype for the chromosome, but an identical phenotypic clone. When the GA propagates and mutates chromosomes throughout generations, this can cause clones to occupy the population, limiting the diversity of the search space, which can trap the optimization. These phenotypic clones can also compound the non-functional chromosome issue by creating additional clones of preexisting non-functional chromosomes, thus again, hindering the search space.

Mitigating Phenotypic Clones

Similar to the strategy for reducing non-functional chromosomes, knowing the expected fraction of junk DNA can allow adjustment of the GA settings to mitigate the effect of clones. Specifically, we adjust the GA mutation rates shown in Listing 6.4 via the *p_mut_per_bit* parameter, which determines the fraction of bits that will undergo a mutation. By setting this mutation to be $> fraction_junk$ we can attempt to alter at least a single bit that can influence the model's performance. However, we currently do not have a method to reliably determine the fraction of junk DNA in any given chromosome. This would require manual mapping of each hyperparameter to its corresponding class. Additionally, the fraction of junk DNA varies with each chromosome in the population. In the example shown in Listing 6.6 we have two models with a hyperparameter each, containing 4 options each, which generates the same amount of junk DNA for either model. In practice, the amount of model-specific hyperparameters varies. Chromosomes using different models will have different fractions of junk DNA. Users can opt to configure their YAML as shown in Listing 6.6, but this would require limiting the search space, which is not effective for a complete exhaustive search and optimization.

6.4 Summary

This work enables the SimpleMind software environment to build, train and evaluate predictive models through a single agent. The agent encompasses a predictive modelling pipeline consisting of pre-processing, data resampling, feature selection, and model training and inference. This allows for a direct transition from feature extraction into a prediction model within a KG, preserving the KG’s interpretability. The design of the predictive modelling agent is a future-oriented, proactive design, where pipeline functions and methods can be updated without backend development. The agent is also compatible with SM’s KNoLO module enabling end-to-end automatic hyperparameter tuning of the predictive modelling pipeline through a GA.

Chapter 7

Kidney Lesion Pathology & Outcome Prediction Models Using Nephrometry and Radiomics

7.1 Introduction

Renal Cell Carcinoma (RCC) is considered a common type of cancer, the 14th most frequently diagnosed. Around 60% of renal masses are discovered incidentally during unrelated medical imaging procedures [123]. This has led to early detection of small renal lesions, treatable with less invasive, nephron-sparing procedures such as partial nephrectomy, ablation, or even active surveillance, depending on the lesion characteristics and the patient's comorbidities [96]. Radiographic evaluation of renal lesions to inform treatment decisions typically involves the collection of manual measurements and calculation of a summary score reflecting lesion complexity [25]. While these scoring systems, such as the RENAL [61], have seen widespread

usage, they rely on manual measurements that can be time-consuming and prone to inter-observer variability. Additionally, as noted by Abdallah et al. [1], nephrometry scoring systems like RENAL condense measurements into an ordinal variable, which results in information loss. To mitigate these issues, automated nephrometry scoring systems have been developed and have shown comparable or better performance than humans for prediction of lesion pathology and operative outcomes [47, 1, 120]. However, these methods for prediction typically rely on Logistic Regression only as their prediction model. While Logistic Regression can be an effective model, there are a variety of ML models available that may perform better for a given classification task and provided dataset. Additionally, these automated methods, despite having a full segmentation of the kidneys and lesions, limit themselves to using only nephrometry features. Thus, there is potential to expand the feature space with the use of Radiomics. Radiomics allows for extraction of quantitative image features based on image intensity, shapes, and textures [63]. These features, which may not be readily measurable or perceived by humans, may provide additional information that can improve these prediction models.

7.2 Methods

7.2.1 Dataset Preparation

We used the publicly available KiTS23 [44] dataset containing 489 pre-operative, contrast-enhanced CT images with corresponding kidney, cyst, and lesion masks, as well as clinical outcome information. Data was split 60%/40% into training and testing sets, with stratifica-

tion performed to preserve the distribution of the outcome variable in each split. Accordingly, train/test splits were generated for the following outcomes: malignancy, tumor necrosis, tumor histological subtype (binarized as clear cell renal cell carcinoma vs. non-clear cell renal cell carcinoma), and surgical procedure (binarized as radical nephrectomy vs. non-radical nephrectomy). In the full KiTS23 cohort, the outcome distributions were as follows: malignancy, 89.8% malignant and 8.8% benign; tumor necrosis, 22.3% present and 75.1% absent; tumor histological subtype, 63.4% clear cell renal cell carcinoma and 35.4% non-clear cell renal cell carcinoma; and surgical procedure, 36.2% radical nephrectomy and 63.8% non-radical nephrectomy. Figure 7.1 shows the distribution of the outcomes of interest in KiTS23.

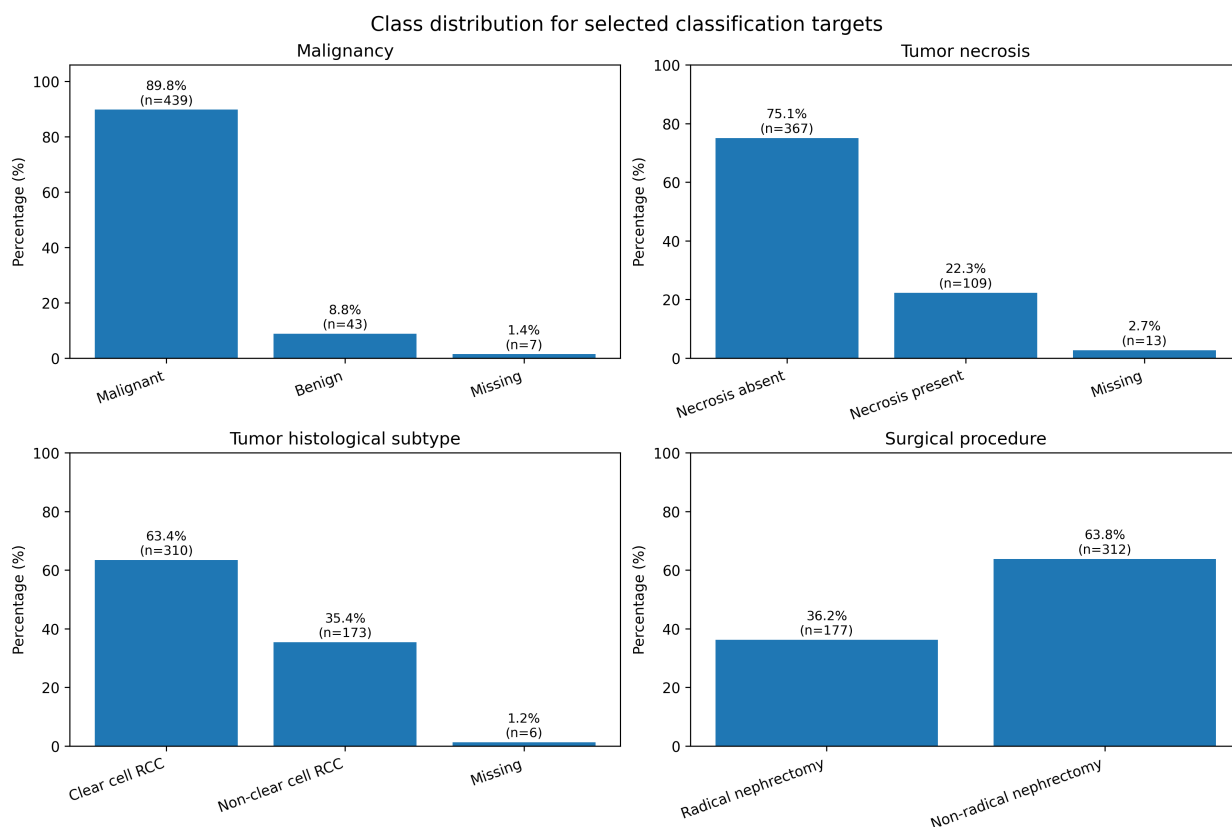


Figure 7.1: Class distribution of selected KiTS23 clinical outcomes and diagnoses.

7.2.2 Nephrometry Feature Extraction

We computed the nephrometry features described in Section 5.2.4. We also extended the SM automated nephrometry KG to compute the following nephrometry features: CSA [69], centrality index, [106] and centrality angle [84].

Contact Surface Area

The tumor CSA [69] refers to the surface area of the lesion that is in contact with the kidney. In the SM KG this is automated by first locating the invading region of the lesion through a logical boolean AND operation between the lesion mask and the affected kidney convex hull. The surface area of that region is then computed through a mesh-based approach from the pyradiomics library [110] and dividing by 2 to obtain the area of a single face of the mesh as in the method from Wood et al [120].

Centrality Index and Centrality Angle

The centrality index [106] measures the ratio of the distance between kidney centroid and the lesion centroid to the lesion radius. In the KG, the centrality index is obtained by extracting the 3D euclidean distance between the kidney centroid and the lesion centroid, and dividing the value by half of the previously computed lesion diameter. The centrality angle measures the angle formed by the tumor relative to the kidney centroid [84]. Typically done manually in 2D coronal slices, we automate and expand this metric to a 3D cone of view. For this feature we compute the 3D euclidean distance from the kidney centroid to the outermost voxel of the lesion invading region D_{kl} . Then, using the lesion radius r , the centrality angle α can be computed as:

$$\alpha = 2 \tan^{-1} \left(\frac{r}{D_{kl}} \right) \quad (7.1)$$

7.2.3 Radiomic Feature Extraction

We followed the guidelines to extract radiomic features from the Image Biomarker Standardization Initiative [125] and the Radiomics "How-to" guide [111]. CT scans were windowed and leveled to 400 and -150 respectively and the corresponding kidney and lesion masks were resegmented to exclude voxels outside the clipped HU intensity range. Using an agent that leverages the pyradiomics library [110], image spacing was resampled to 1 mm isotropic and HU intensities were discretized using a bin width of 25. Lastly, we extracted the full set of original and filter features available in pyradiomics for the affected kidney mask and the lesion mask for a total of 2825 features when combined with the nephrometry features.

7.2.4 Predictive Modeling

We used the predictive modelling approach described in Section 6.2. In this subsection, we will describe the uninstantiated settings file used to explore the options/combinations available to construct our kidney lesion pathology and outcome prediction models. We will focus on the primary classes used on the main steps, the full depth of the classes and input arguments can be found in Appendix C.

Preprocessing

In preprocessing (Appendix Listing C.1), the KNoLO *apt_list* is composed of: no preprocessing, max absolute scaling, standard scaling, normalization, robust scaling, power transformation and quantile transformation.

Feature Selection

In feature selection (Appendix Listing C.2), we cover: no selection, selection using a model, recursive feature selection, selection based on false positive rate, selection based on false discovery rate, selection by best percentile and selection by k best. Order of doing selection before or after resampling is also tuned through an *apt_list*.

Resampling

In resampling (Appendix Listing C.3), the options provided are: no resampling, random over sampling, Synthetic Minority Over-sampling Technique (SMOTE), Borderline SMOTE, random under sampling, edited nearest neighbors, SMOTE using totem links, and SMOTE Edited Nearest Neighbor (SMOTEENN).

Model Selection

For model selection (Appendix Listing C.4), the options include: Linear Support Vector Classifier, Logistic Regression, Random Forest Classifier, Histogram-based Gradient Boosting Classification Tree, Extra Trees Classifier, Support Vector Classifier, Stochastic Gradient Descent Classifier, Ridge Classifier, Linear Discriminant Analysis, and Multi-Layer-Perceptron Classifier. The model selection in this step also determines the estimator and estimator

settings to be used in feature selection for model based methods such as model based selection and Recursive Feature Selection.

7.2.5 Genetic Algorithm Optimization

The set of methods and parameters for preprocessing, feature selection, resampling and, model selection and training described in section 7.2.4 resulted in a GA chromosome of bit length of 126, thus a total of 2^{126} possible method and parameter combinations. We used a custom penalized Area Under Curve (AUC) as the fitness function. This function is composed of multiple terms: The mean validation AUC,

$$\mu_{val} = \left(\frac{1}{n} \sum_{fold=0}^n AUC_val_{fold} \right), \quad (7.2)$$

from which we compute the standard deviation of the validation AUC,

$$\sigma_{val} = \sqrt{\frac{1}{n-1} \sum_{fold=0}^n (AUC_val_{fold} - \mu_{val})^2}. \quad (7.3)$$

To control for overfitting, we compute two more terms: mean model complexity which is given by the mean of the ratio of selected features to total features per fold:

$$\mu_{cplx} = \left(\frac{1}{n} \sum_{fold=0}^n \frac{selected_features_{fold}}{total_features} \right), \quad (7.4)$$

and the training and validation performance gap, which is given by the difference between the mean training AUC and the mean validation AUC:

$$train_val_gap = (\frac{1}{n} \sum_{fold=0}^n AUC_train_{fold}) - \mu_{val}. \quad (7.5)$$

Combining 7.2, 7.3, 7.4, and 7.5, with regularization coefficients, α , β , and λ , our fitness function takes the form of:

$$Fitness = \mu_{val} - \alpha(\sigma_{val}) - \lambda(\mu_{cplx}) - \beta(train_val_gap). \quad (7.6)$$

In this experiment the regularization coefficients were set to $\alpha = 1$, $\beta = 0.05$, and $\lambda = 0.1$.

For the GA optimization, we used an overall population size of 200, a survivor population of 160, and a tournament size of 60 chromosomes. The selection of these population and tournament values was based on the strategy to reduce the selection of non-functional chromosomes described in Section 6.3.3. The bit string representing each chromosome had a probability of undergoing a crossover mutation of 0.1 and an individual mutation probability of 0.15. If the individual chromosome underwent a mutation, each bit had a probability of 0.2 to be flipped. The GA optimization ran for 100 generations maximizing the fitness function (7.6). Each chromosome fitness was computed with a cross-validation of 10 splits repeated 5 times for a total of 50 training and validation scores.

7.2.6 Statistical Analysis

We aim to answer two questions: (1) Is the KNoLO best performing chromosome (highest fitness) better than random selection (indicating positive progression of evolution)? (2) Is the KNoLO best performing chromosome better than a baseline predictive model using default parameters? This can be shown by testing for significant differences in AUC performance. In

hypothesis form:

1. Random chromosome vs best chromosome:

(a) $H_0: AUC_{best} = AUC_{random}$

(b) $H_a: AUC_{best} \neq AUC_{random}$

2. Baseline model vs best chromosome:

(a) $H_0: AUC_{best} = AUC_{baseline}$

(b) $H_a: AUC_{best} \neq AUC_{baseline}$

To test our hypothesis, we used Delong’s test [24] to determine statistically significant differences in AUC between the models on a fixed test set.

7.3 Results

7.3.1 Malignancy Prediction

In malignancy prediction, the best random chromosome had a fitness of 0.584 (mean validation AUC 0.754 ± 0.141), while the overall top performer had a fitness of 0.688 (mean validation AUC 0.822 ± 0.108). Figure 7.2 shows a scatterplot of the evolution of fitness across generations.

Test AUC performance for the baseline was 0.465, 0.582 for the random chromosome, and 0.554 for the best chromosome (Figure 7.3). Test performance between baseline and best chromosome was not statistically different (Delong’s $p = 0.096$). No statistical differences were found between the random chromosome and best chromosome (Delong’s $p = 0.680$).

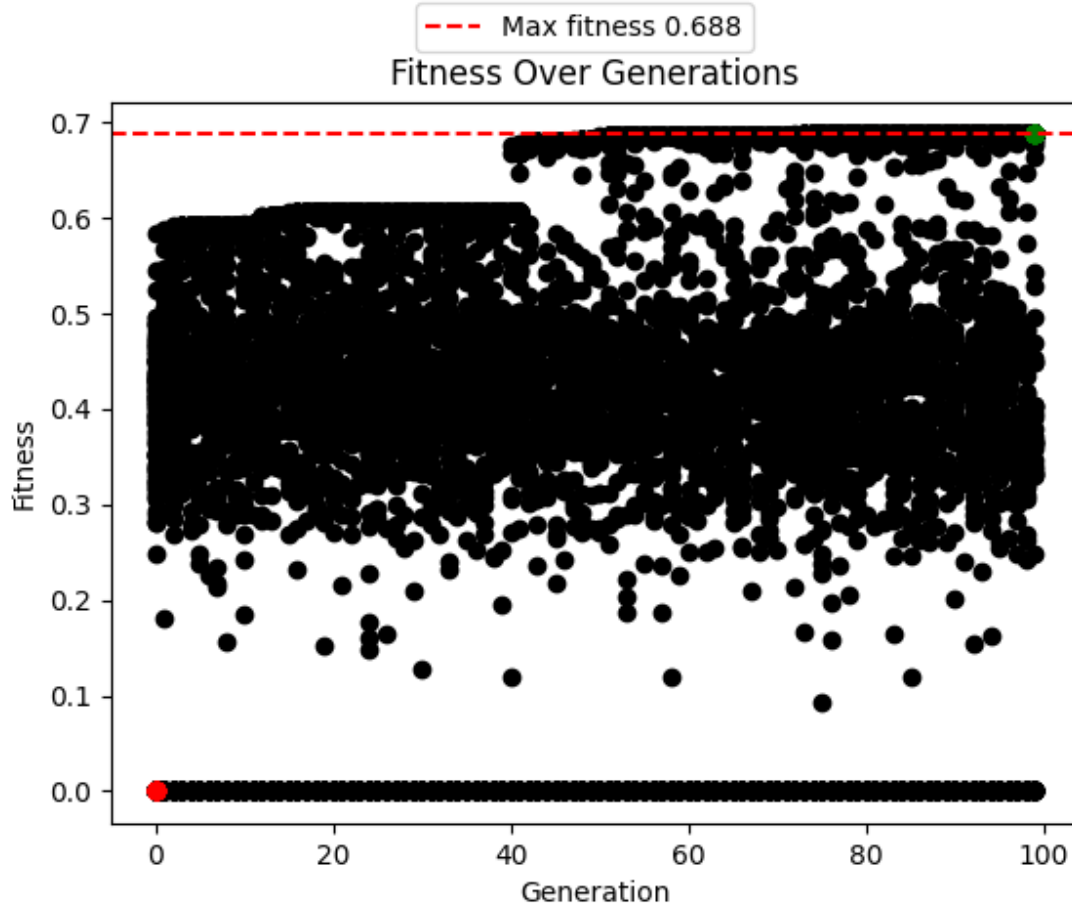


Figure 7.2: GA evolution for malignancy prediction chromosomes.

7.3.2 Surgical Procedure Prediction

For prediction of surgical procedure, the best random chromosome achieved a fitness of 0.837 (mean validation AUC 0.897 ± 0.057), while the overall top performer achieved a fitness of 0.854 (mean validation AUC 0.904 ± 0.05). Figure 7.4 shows a scatterplot of the evolution of fitness across generations.

On the test set, the baseline model achieved an AUC of 0.869, while the random chromosome and best chromosome both achieved an AUC of 0.872 (Figure 7.5). No statistical differences in AUC were found between the baseline and the best chromosome (Delong’s

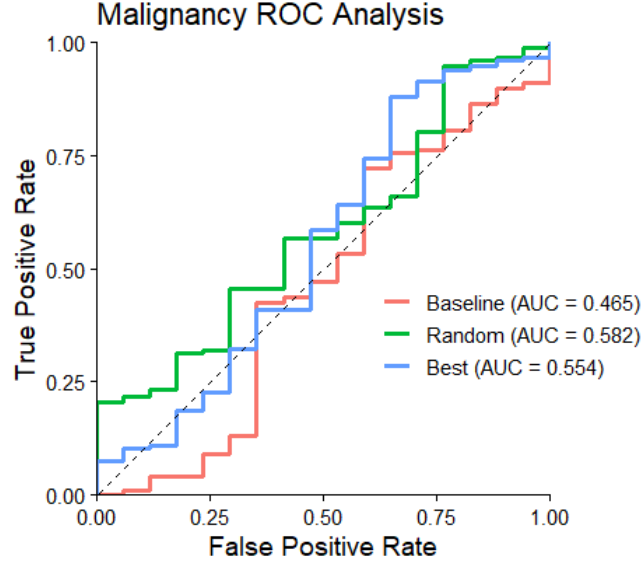


Figure 7.3: ROC curve analysis for malignancy prediction. Baseline, Random Chromosome, and Best Chromosome.

$p = 0.676$) nor between the random chromosome and best chromosome (Delong's $p = 0.955$).

7.3.3 Tumor Necrosis Prediction

For prediction of tumor necrosis, the best random chromosome had a fitness of 0.677 (mean validation AUC 0.782 ± 0.102). The best chromosome achieved a fitness of 0.71 (mean validation AUC 0.803 ± 0.084). Figure 7.6 shows a scatterplot of the evolution of fitness across generations.

On the test set for tumor necrosis prediction, the baseline model achieved an AUC of 0.806, while the random chromosome achieved 0.809 and the best chromosome achieved an AUC of 0.719 (Figure 7.7). Statistical difference in AUC was found between the baseline and the best chromosome (Delong's $p = 0.039$) and between the random chromosome and best chromosome (Delong's $p = 0.009$).

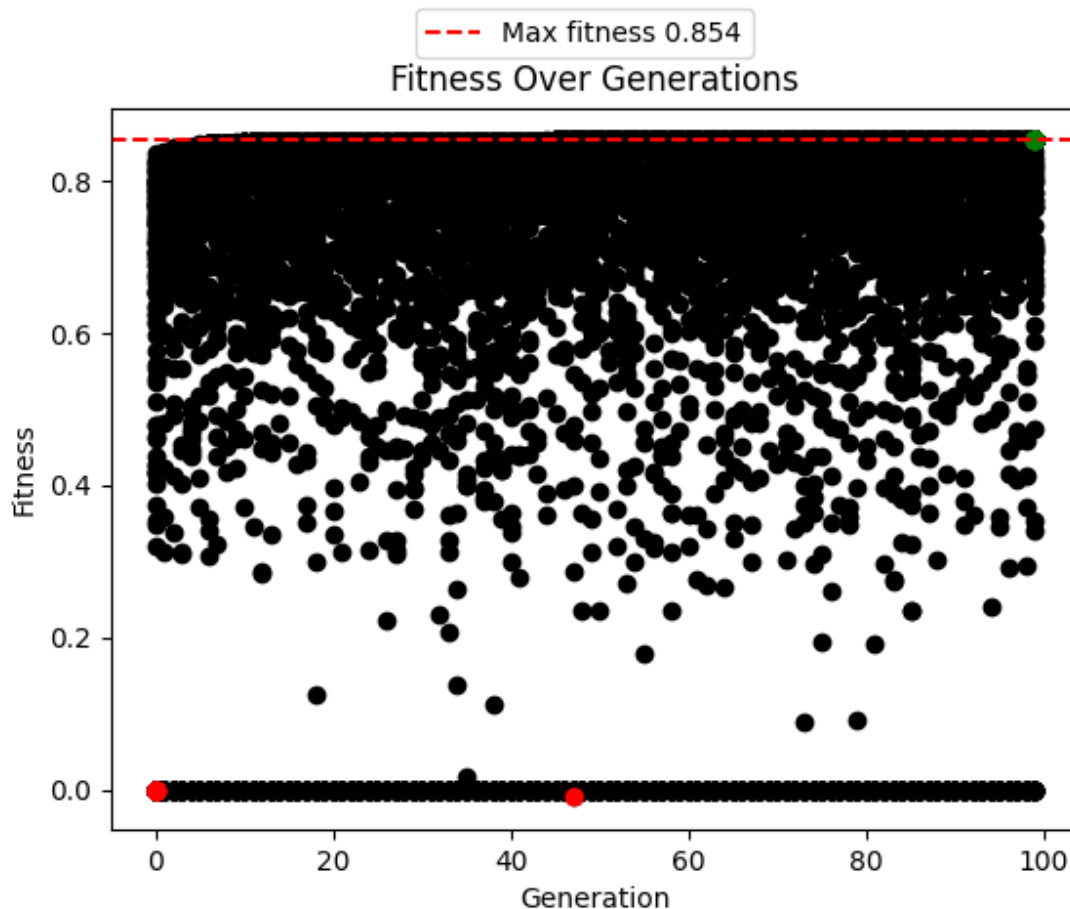


Figure 7.4: GA evolution for surgical procedure prediction chromosomes.

7.3.4 Tumor Histologic Subtype

In the prediction of tumor subtype, the best random chromosome achieved a fitness of 0.625 (mean validation AUC 0.719 ± 0.085). The best chromosome achieved a fitness of 0.674 (mean validation AUC 0.768 ± 0.09). Figure 7.8 shows a scatterplot of the evolution of fitness across generations.

On the test set for tumor subtype prediction, the baseline model achieved an AUC of 0.697, while the random chromosome achieved 0.741 and the best chromosome achieved an AUC of 0.768 (Figure 7.9). Statistical difference in AUC was found between the baseline and

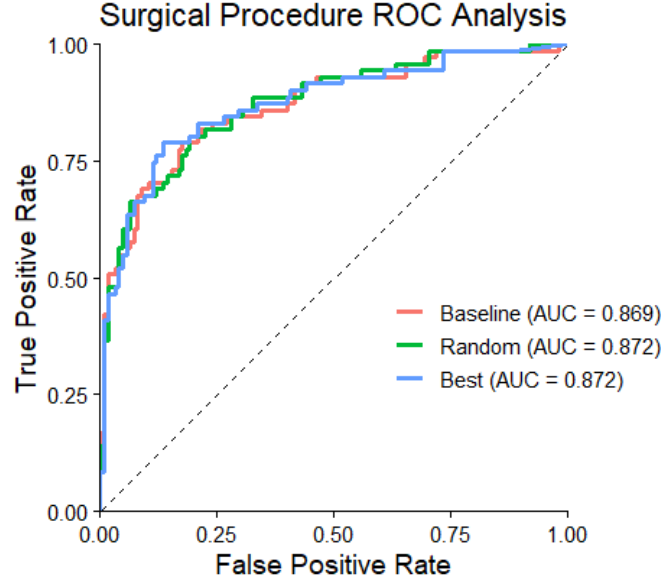


Figure 7.5: ROC curve analysis for surgical procedure prediction. Baseline, Random Chromosome, Best Chromosome.

the best chromosome (Delong's $p = 0.016$), but not between the random chromosome and best chromosome (Delong's $p = 0.367$).

7.3.5 Population Diversity Analysis

We analyzed the chromosome populations to identify the amount of non-functional chromosomes and phenotypic clones. Metrics are normalized by dividing by the total population size of each generation (200). Table 7.1 provides summary statistics of the fraction of non-functional chromosomes per generation for each outcome. The mean fraction of non-functional chromosomes ranged from 0.145 to 0.161. The max fraction for non-functional chromosomes ranged from 0.49 to 0.55. Plots in Figure 7.10 show that this max value was reached on the first generation (where all chromosomes are randomly initialized) and then reduced towards their respective mean from the second generation onward.

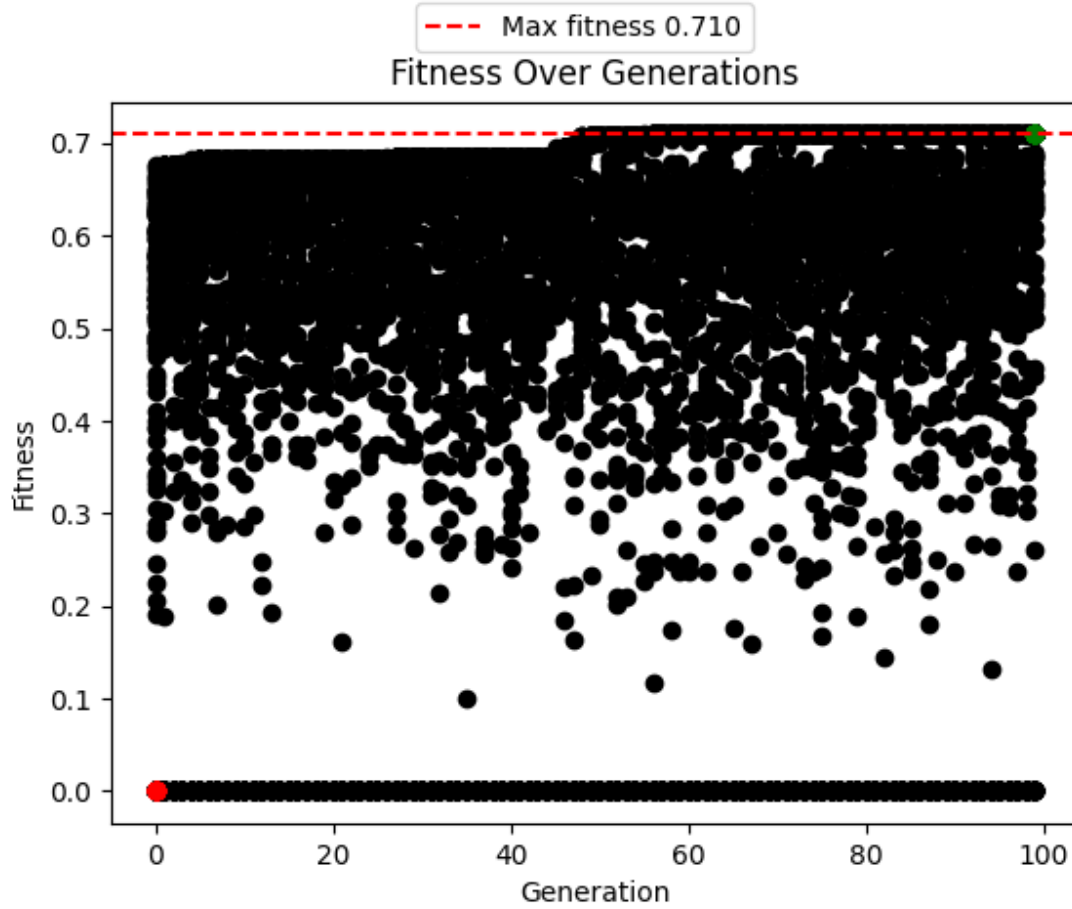


Figure 7.6: GA evolution for tumor necrosis prediction chromosomes.

Table 7.1: Per-generation summary statistics for non-functional chromosomes (fitness=0 fraction of population) across outcomes.

Statistic	Malignant	Surgical procedure	Tumor necrosis	Tumor subtype
Mean	0.15	0.141	0.161	0.145
σ	0.043	0.044	0.040	0.046
Min	0.090	0.070	0.085	0.100
25%	0.130	0.125	0.140	0.125
50%	0.145	0.135	0.160	0.140
75%	0.160	0.150	0.170	0.155
Max	0.505	0.530	0.490	0.550

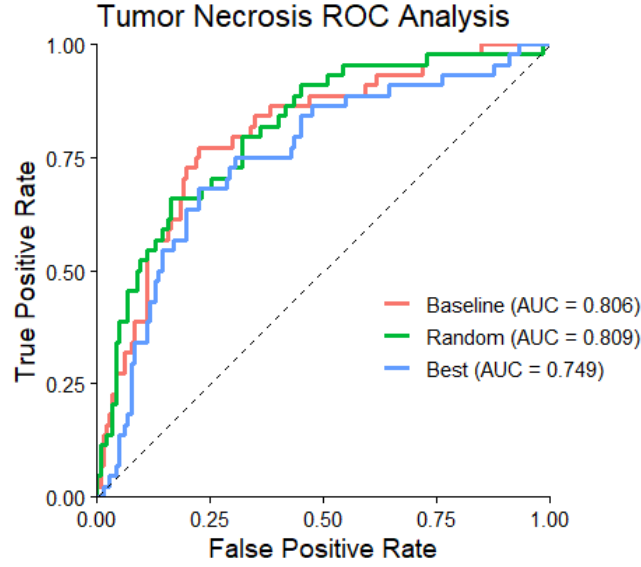


Figure 7.7: ROC curve analysis for tumor necrosis prediction. Baseline, Random Chromosome, Best Chromosome.

7.4 Discussion

The training set fitness changes showed a positive progression from random, to baseline, to GA best. This serves as confirmation that the GA methodology was indeed successful in the optimization of the custom fitness function. However, while model optimization was positive, model generalization was not. Positive progression of the fitness function did not translate into performance enhancements during test time when comparing random vs best nor baseline vs best. The lack of performance improvements from baseline can potentially be due to default settings already being fine-tuned by humans, determined to be the most generalizable and thus were set as the library's defaults. The lack of performance improvements from random to baseline, however, may be indicative of other potential limitations. For example, the limited data and class distribution imbalance can also negatively impact generalization through overfitting on training data. An example is the prediction of lesion malignancy

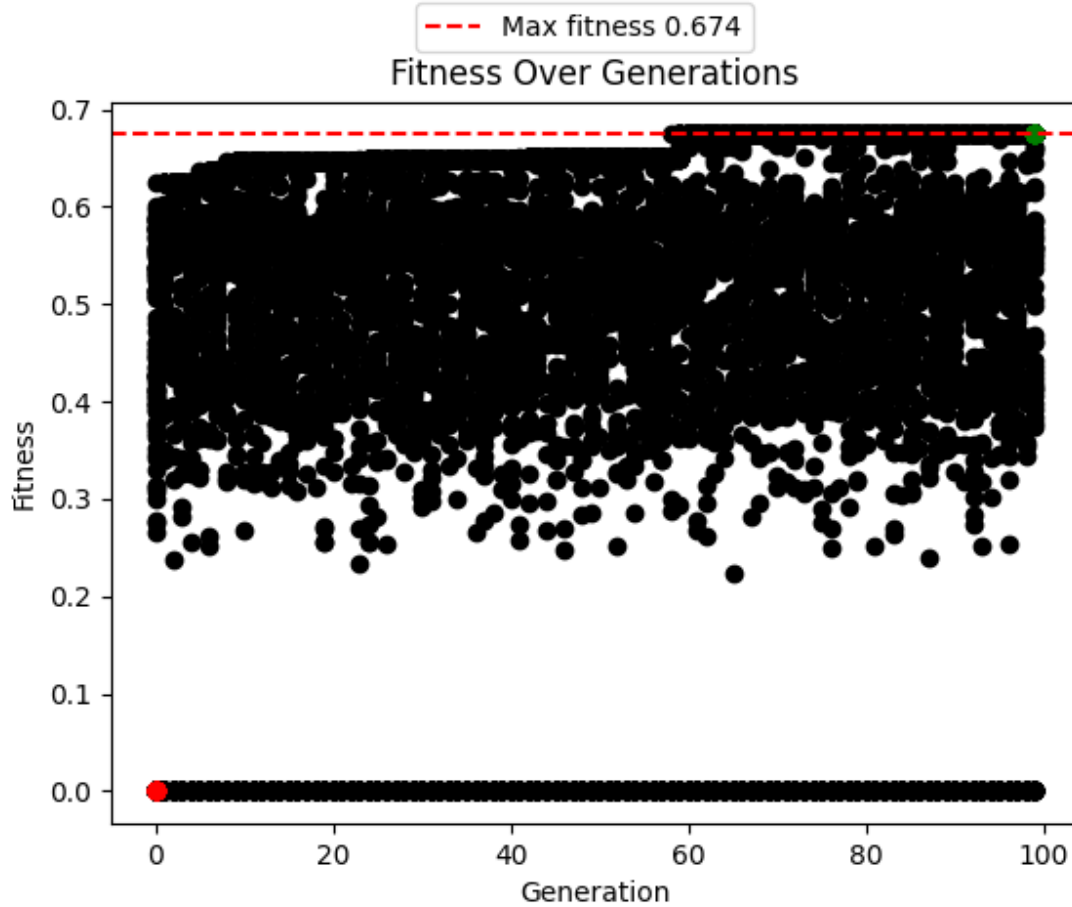


Figure 7.8: GA evolution for tumor subtype prediction chromosomes.

where validation performance of best chromosome was 0.822 while the AUC on the test was 0.554. This is suggestive of overfitting, despite the penalty set in the fitness function. The distribution of the malignancy dataset as well was the most imbalanced of the tested outcomes with 8.8% of cases being benign while 89.8% were malignant.

The search space for this task was generated from a chromosome bit-string length of 126, which produces $2^{126} \approx 8.51 \times 10^{37}$ possible combinations of hyperparameters. Given the size of the search space, using a population size of 200 means, that with our current optimization evolution settings of 100 generations (section 7.2.5), the GA can only test:

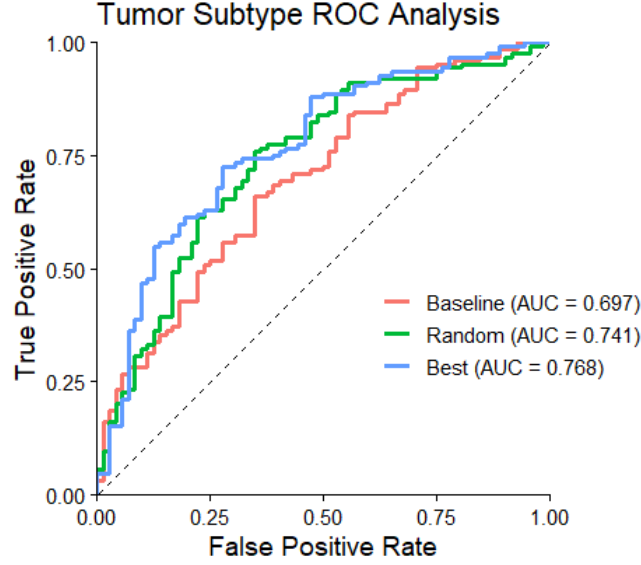


Figure 7.9: ROC curve analysis for tumor histologic subtype prediction. Baseline, Random Chromosome, Best Chromosome.

100 generation $\times$ 200 chromosomes/generation = 20,000 chromosomes, which is a practical search space of $2.35 \times 10^{-32}\%$ of the entire search space. While the GA is not intended to test all combinations, but rather move towards convergence, this limited search space can hinder potential performance improvements.

In Sections 6.3.3 and 6.3.4, we discussed the limitations when performing GA optimization tasks on predictive models. Our diversity analysis in Section 7.3.5 highlights the persistence of: non-functional chromosomes and phenotypic clones. Non-functional chromosomes were reduced using our population settings, going from $\approx 50\%$ of the population, down to $\approx 20\%$ of the population. While having 20% of the population non-functional can negatively impact the optimization, the phenotypic clones, based on the results, were likely the main contributors to stagnation of the evolution. As seen in Table 7.2 and Figure 7.11, $\approx 80\%$ of chromosomes were clones of another chromosome (including non-functional chromosomes). This also hinders the

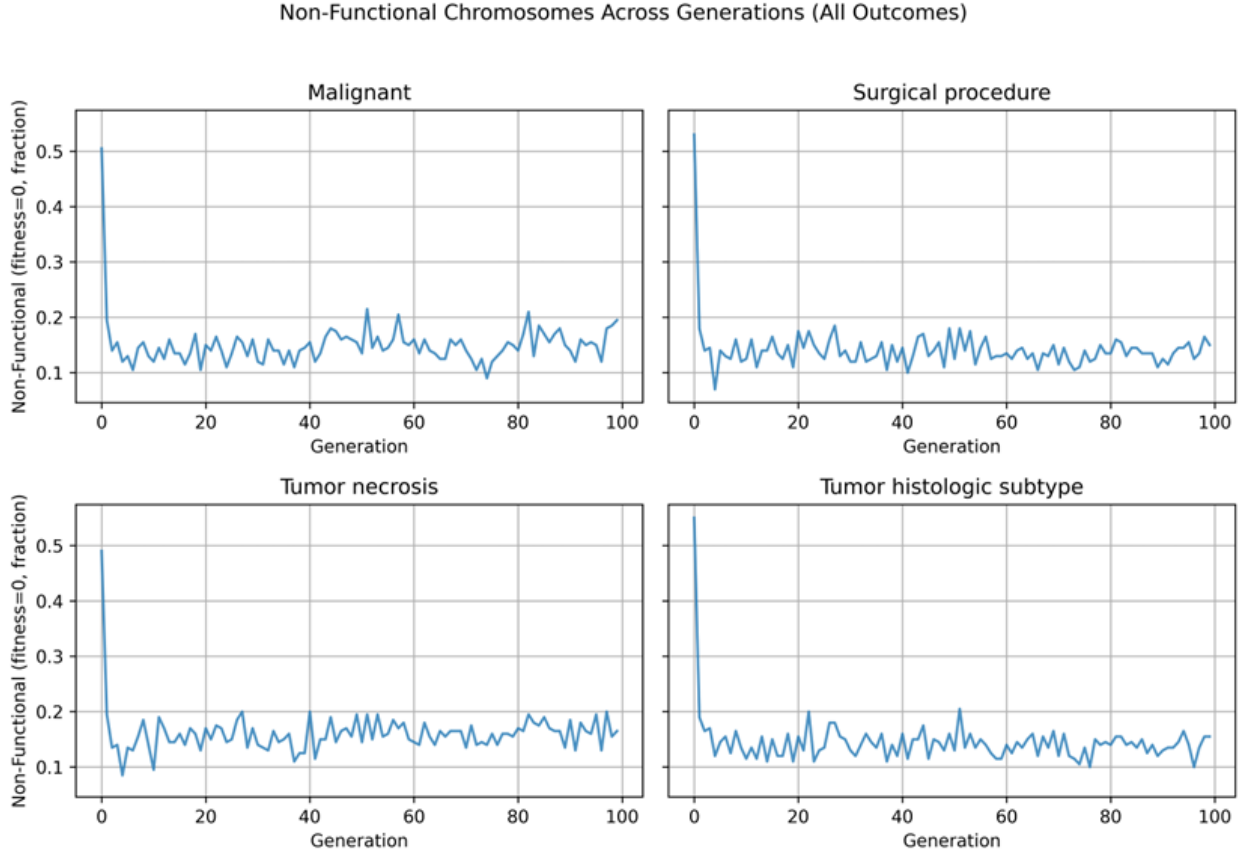


Figure 7.10: Line plots of fraction of non-functional chromosomes across generations.

available search space the GA can effectively explore (i.e. only $\approx 20\%$ of the population per generation were unique chromosomes). Out of a total population size of 200, this means the GA could only practically test 40 chromosomes per generation, 4000 chromosomes in total out of a search space of $\approx 8.51 \times 10^{37}$. While testing any significant fraction of such a large space is not computationally feasible, 4000 chromosomes make up 20% of our original goal of 20,000 chromosomes for the given population settings and generations.

Table 7.2: Per-generation summary statistics for phenotypic clones (fraction of population) across outcomes.

Statistic	Malignant	Surgical procedure	Tumor necrosis	Tumor subtype
mean	0.79	0.789	0.807	0.793
std	0.0441	0.0414	0.0449	0.0401
min	0.5	0.525	0.485	0.545
25%	0.77	0.775	0.795	0.775
50%	0.795	0.793	0.812	0.795
75%	0.815	0.811	0.83	0.815
max	0.89	0.865	0.895	0.885

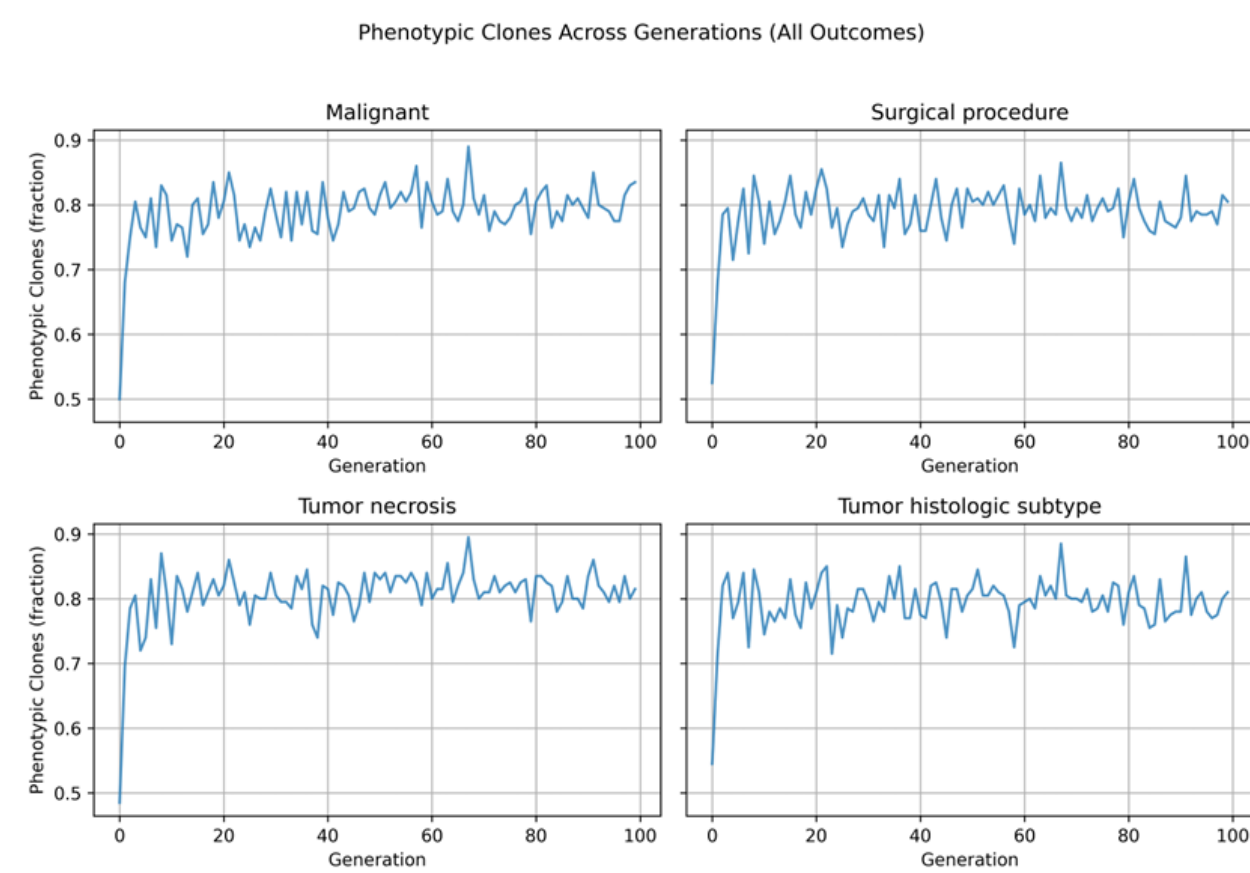


Figure 7.11: Line plots of fraction of phenotypic clones across generations.

7.5 Conclusion

We have developed and applied tools for the construction of predictive models within the SM software environment compatible with SM’s KNoLO module for automatic parameter

tuning via a GA. A custom and modifiable fitness function was implemented to allow the evaluation of chromosome fitness throughout generations. Positive evolution progression was noted indicative of successful fitness function optimization, however, this did not generalize to test data. Potential limitations for test performance could stem from the limited dataset and imbalance of the dataset as seen in Section 7.2.1, extensive search space and phenotypic clones limiting the space. Future work will explore modifications to the fitness function for GA convergence and overfitting prevention. We will also explore the application of KNoLO for predictive models on classical standardized datasets and GA modifications to identify and eliminate phenotypic clones.

Appendices

Appendix A

The SimpleMind Knowledge Graphs for Kidney Segmentation

The following section contains additional figures to illustrate the KGs described in the methods section 3.2.2. Circular nodes represent SM agents while square nodes represent agent outputs.

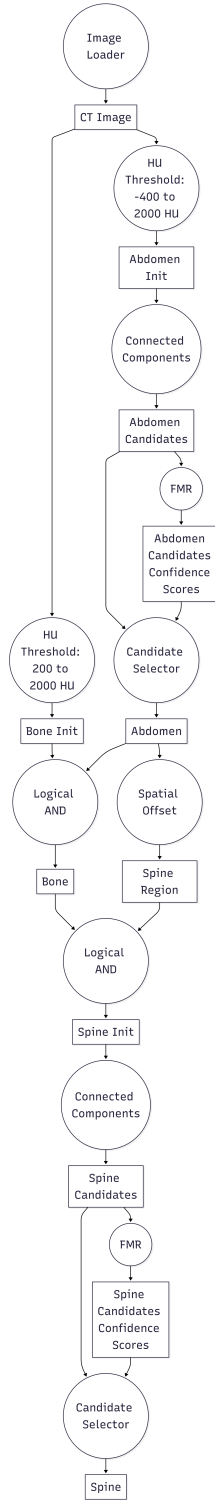


Figure A.1: SimpleMind Knowledge graph for segmentation of the abdomen, bone and spine as described in 3.2.2.

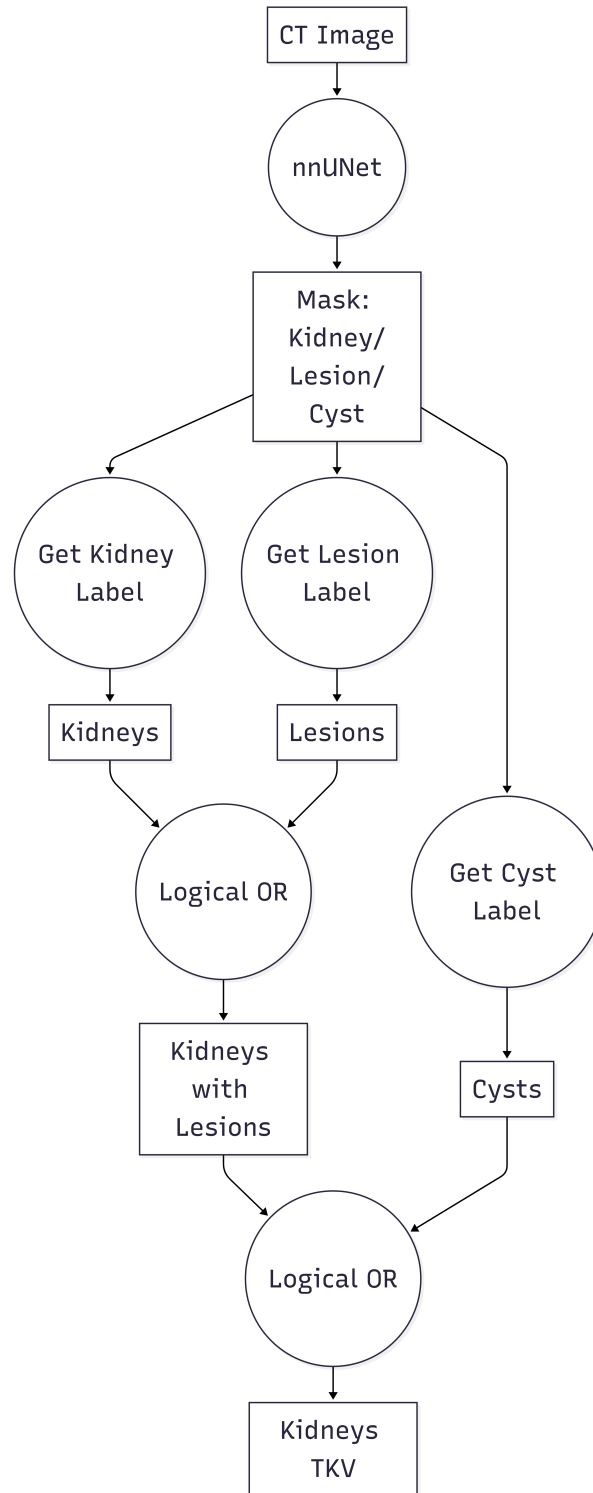


Figure A.2: SimpleMind Knowledge graph for segmentation of the Kidneys TKV using nnUNet as described in 3.2.2.

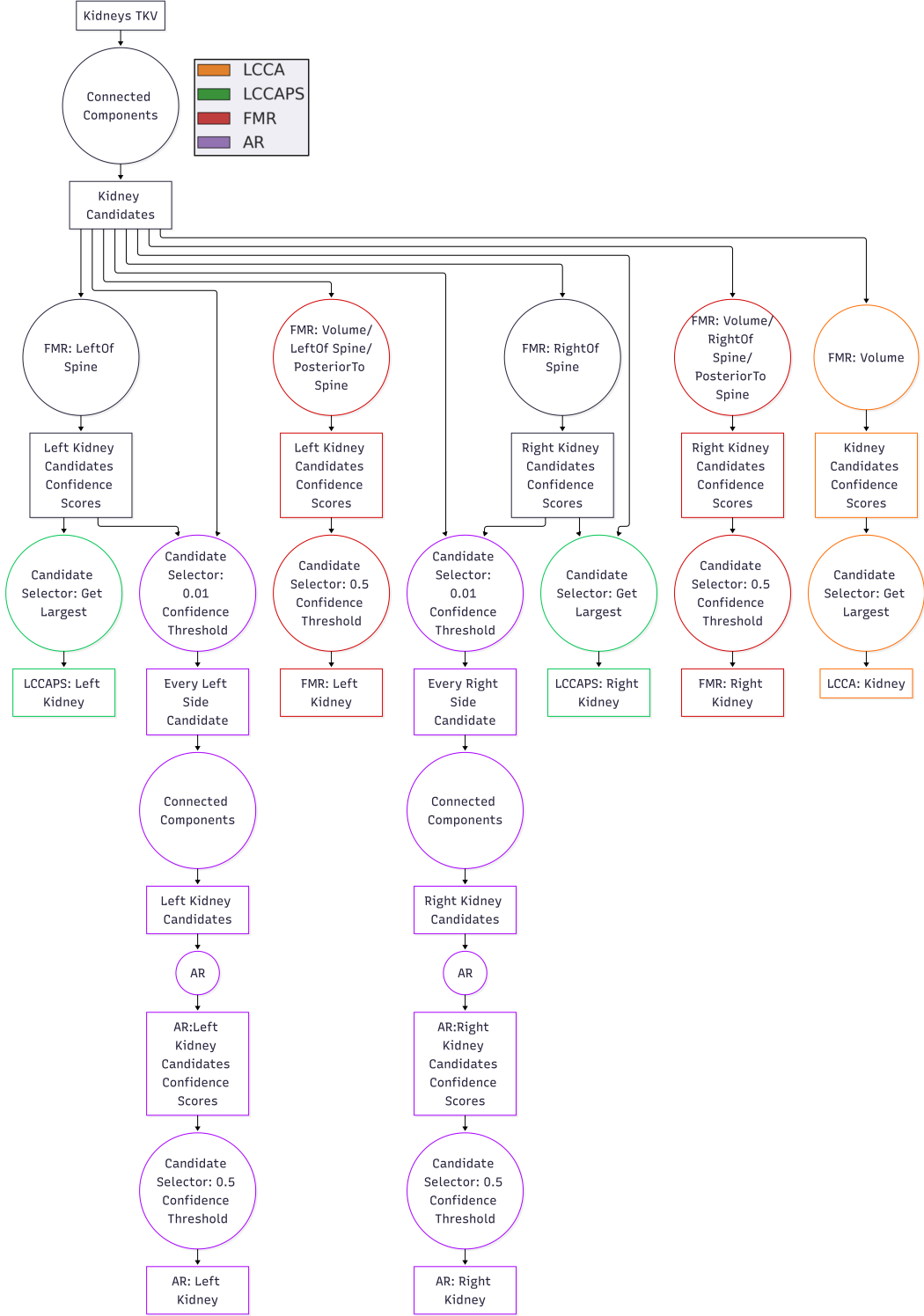


Figure A.3: SimpleMind Knowledge graph of the kidney segmentation refinement methods as described in 3.2.2.

Appendix B

Methodology for rotation of kidney polar lines from Chapter 5

This appendix describes how we (i) estimate an organ’s long axis from a 3D binary mask and (ii) rotate the polar lines so that they are tilted by the same *coronal-plane* (XZ) offset as the organ axis relative to the image midline.

B.0.1 Image Geometry and Coordinate Systems

Let voxel indices be $\mathbf{i} = (i_x, i_y, i_z)^\top$ and physical coordinates (mm) be $\mathbf{p} \in \mathbb{R}^3$. Using the image header, indices map to physical space via

$$\mathbf{p}(\mathbf{i}) = \mathbf{o} + \mathbf{D}(\mathbf{s} \odot \mathbf{i}), \tag{B.1}$$

where $\mathbf{o}$ is the origin, $\mathbf{s} = (s_x, s_y, s_z)^\top$ is voxel spacing, $\odot$ denotes elementwise multiplication, and $\mathbf{D} \in \mathbb{R}^{3 \times 3}$ is the direction cosine matrix. The physical unit vectors corresponding to the

image index axes are the (normalized) columns of $\mathbf{D}$:

$$\hat{\mathbf{e}}_x = \frac{\mathbf{D}_{:,1}}{\|\mathbf{D}_{:,1}\|}, \quad \hat{\mathbf{e}}_y = \frac{\mathbf{D}_{:,2}}{\|\mathbf{D}_{:,2}\|}, \quad \hat{\mathbf{e}}_z = \frac{\mathbf{D}_{:,3}}{\|\mathbf{D}_{:,3}\|}. \quad (\text{B.2})$$

In the coronal view, we treat $\hat{\mathbf{e}}_z$ as the “midline” (vertical) direction and $\hat{\mathbf{e}}_y$ as the normal of the coronal plane.

B.0.2 Organ Long Axis via Principal Component Analysis (PCA)

Let $\{\mathbf{p}_k\}_{k=1}^N$ be the physical coordinates of all foreground voxels in the organ mask. The centroid is given by:

$$\bar{\mathbf{p}} = \frac{1}{N} \sum_{k=1}^N \mathbf{p}_k. \quad (\text{B.3})$$

We center the point cloud and apply PCA (implemented via Singular Value Decomposition) to obtain the dominant direction $\hat{\mathbf{v}}$ (unit length), which we interpret as the organ long axis [34]. Because PCA returns an *axis*, $\hat{\mathbf{v}}$ and $-\hat{\mathbf{v}}$ represent the same line; we therefore apply a sign convention when computing a signed tilt (described below).

B.0.3 Coronal-Plane Tilt Angle Relative to Midline

To imitate the method a human reader would use to create the polar lines, we want the tilt of the organ axis in the coronal (XZ) plane, i.e., we ignore components along $\hat{\mathbf{e}}_y$. For any

vector $\mathbf{u}$, its projection into the plane orthogonal to $\hat{\mathbf{e}}_y$ is

$$\Pi_{XZ}(\mathbf{u}) = \mathbf{u} - (\mathbf{u} \cdot \hat{\mathbf{e}}_y) \hat{\mathbf{e}}_y. \quad (\text{B.4})$$

We form two projected unit vectors:

$$\hat{\mathbf{a}} = \frac{\Pi_{XZ}(\hat{\mathbf{e}}_z)}{\|\Pi_{XZ}(\hat{\mathbf{e}}_z)\|}, \quad \hat{\mathbf{b}} = \frac{\Pi_{XZ}(\hat{\mathbf{v}})}{\|\Pi_{XZ}(\hat{\mathbf{v}})\|}. \quad (\text{B.5})$$

The signed coronal-plane angle $\theta \in (-\pi, \pi]$ from $\hat{\mathbf{a}}$ to $\hat{\mathbf{b}}$ is computed using an $\arctan 2$ expression:

$$\theta = \arctan 2\left(\hat{\mathbf{e}}_y \cdot (\hat{\mathbf{a}} \times \hat{\mathbf{b}}), \hat{\mathbf{a}} \cdot \hat{\mathbf{b}}\right). \quad (\text{B.6})$$

The dot product term controls the magnitude (cosine-like), while the triple product $\hat{\mathbf{e}}_y \cdot (\hat{\mathbf{a}} \times \hat{\mathbf{b}})$ determines the sign according to the right-hand rule about $\hat{\mathbf{e}}_y$.

Axis sign convention. Because PCA may return $\hat{\mathbf{v}}$ or $-\hat{\mathbf{v}}$, we optionally enforce the axis orientation by requiring $\hat{\mathbf{v}} \cdot \hat{\mathbf{e}}_z \geq 0$; if not, we replace $\hat{\mathbf{v}} \leftarrow -\hat{\mathbf{v}}$. This removes sign flips in θ caused by the PCA axis ambiguity and different scan coordinate systems.

B.0.4 Rotate the Polar Lines About the Coronal Axis

We rotate the polar lines by the coronal-plane tilt angle using a rotation about $\hat{\mathbf{e}}_y$. We rotate only about $\hat{\mathbf{e}}_y$ to match the coronal-view tilt definition.

Let $\mathbf{c}$ be the rotation center in physical space (we use the organ centroid $\bar{\mathbf{p}}$). A point $\mathbf{p}$ is

rotated by

$$\mathbf{p}' = \mathbf{c} + \mathbf{R}(\hat{\mathbf{e}}_y, \theta) (\mathbf{p} - \mathbf{c}), \quad (\text{B.7})$$

where $\mathbf{R}(\hat{\mathbf{e}}_y, \theta)$ is constructed using Rodrigues' formula [39]:

$$\mathbf{R}(\hat{\mathbf{e}}_y, \theta) = \mathbf{I} + \sin \theta [\hat{\mathbf{e}}_y]_{\times} + (1 - \cos \theta) [\hat{\mathbf{e}}_y]_{\times}^2. \quad (\text{B.8})$$

We apply this transform to the polar lines mask via resampling on the original image grid using Nearest-neighbor interpolation. However, when the polar lines boundary becomes oblique relative to the voxel grid, staircase artifacts may appear due to discretization of voxels, especially in coronal slices and 3D surface renderings.

Appendix C

Chapter 7 Uninstantiated YAML configurations for KNoLO

The following listings show the complete uninstantiated yaml configurations used to create the search space for optimizing the renal lesion pathology and surgical outcome models using the GA.

```

preprocessing_settings:
  preprocessor: { value: StandardScaler, apt_list: [no_preprocessing,
MaxAbsScaler, MinMaxScaler, Normalizer, StandardScaler, RobustScaler,
PowerTransformer, QuantileTransformer] }

  random_state: *seed
  axis: 0

  clip: { value: false, apt_list: [true, false] }

  with_mean: { value: true, apt_list: [true, false] }
  with_std: { value: true, apt_list: [true, false] }

  with_centering: { value: true, apt_list: [true, false] }
  with_scaling: { value: true, apt_list: [true, false] }

  unit_variance: { value: false, apt_list: [true, false] }

  norm: { value: l2, apt_list: [l1, l2, max] }

  method: yeo-johnson

  standardize: { value: true, apt_list: [true, false] }

  n_quantiles: { value: 100, apt_list: [5, 8, 10] }

  output_distribution: { value: uniform, apt_list: [uniform, normal] }

```

Listing C.1: Uninstantiated YAML configuration for KNoLO optimization: Preprocessing.


```

feature_selection_settings:
  feature_selector: { value: RFE, apt_list: [no_selection, SelectFromModel,
RFE, SelectFpr, SelectFdr, SelectPercentile, SelectKBest] }

  random_state: *seed

  selection_first: { value: true, apt_list: [true, false] }

  score_func: { value: f_classif, apt_list: [f_classif,
mutual_info_classif] }

  n_features_to_select: { value: 10, apt_list: [25, 50, 100, 250, 500] }

  max_features: { value: 10, apt_list: [20, 10, 5, 30, 50, 40, 100, 200,
500] }

  percentile: { value: 10, apt_list: [5, 10, 20, 30, 40, 50] }

  threshold: { value: mean, apt_list: [mean, median, 0.5*mean, 1.5*mean] }

  norm_order: { value: 1, apt_list: [1, 2] }

  step: { value: 10, apt_list: [0.1, 0.2, 0.3] }

  k: { value: 10, apt_list: [50, 100, 200, 300, 500] }

```

Listing C.2: Uninstantiated YAML configuration for KNoLO optimization: Feature Selection.

```

resampler_settings:
  resampler: { value: no_resampling, apt_list: [no_resampling,
RandomOverSampler, SMOTE, BorderlineSMOTE, RandomUnderSampler,
EditedNearestNeighbours, SMOTETomek, SMOTEENN] }
  random_state: *seed

  sampling_strategy: { value: auto, apt_list: [auto, all, minority, not
minority, not majority, 0.1, 0.25, 0.5, 0.75, 1.0] }

  k_neighbors: { value: 5, apt_list: [1, 3, 5, 7, 10, 15] }
  m_neighbors: { value: 10, apt_list: [3, 5, 7, 10, 15] }

  replacement: { value: false, apt_list: [false, true] }

  n_neighbors_enn: { value: 3, apt_list: [1, 3, 5, 7] }

  kind_sel: { value: mode, apt_list: [mode, all] }

```

Listing C.3: Uninstantiated YAML configuration for KNoLO optimization: Resampling.

```

ml_settings:
  model: {value: LogisticRegression, apt_list: [LinearSVC,
LogisticRegression, RandomForestClassifier, HistGradientBoostingClassifier,
ExtraTreesClassifier, SVC, SGDClassifier, RidgeClassifier,
LinearDiscriminantAnalysis, MLPClassifier]}
  random_state: *seed
  penalty: {value: l2, apt_list: [l1, l2, elasticnet, null]}
  l1_ratio: {value: 0.5, apt_list: [0.0, 0.1, 0.3, 0.5, 0.7, 0.9]}
  alpha: {value: 0.0001, apt_list: [0.000001, 0.00001, 0.0001, 0.001, 0.01,
0.1]}
  loss: {value: squared_hinge, apt_list: [hinge, squared_hinge, log_loss,
modified_huber, perceptron]}
  dual: {value: false, apt_list: [true, false]}
  tol: {value: 0.0001, apt_list: [0.001, 0.0001, 0.00001]}
  C: {value: 1.0, apt_list: [0.01, 0.1, 1.0, 10.0, 100.0]}
  fit_intercept: true
  intercept_scaling: {value: 1.0, apt_list: [0.1, 1.0, 10]}
  class_weight: {value: balanced, apt_list: [null, balanced]}
  max_iter: {value: 1000, apt_list: [50, 100, 200, 300, 1000]}
  solver: {value: saga, apt_list: [liblinear, saga, lbfgs, svd, lsqr,
adam]}
  n_estimators: {value: 100, apt_list: [50, 100, 200, 300, 500]}
  criterion: {value: gini, apt_list: [gini, entropy]}
  max_depth: {value: 10, apt_list: [null, 3, 5, 10, 20, 30, 50]}
  min_samples_split: {value: 2, apt_list: [2, 5, 7, 10]}
  min_samples_leaf: {value: 1, apt_list: [1, 2, 4, 6, 10, 20, 30]}
  max_features: {value: sqrt, apt_list: [sqrt, log2, null, 0.5, 1.0, 0.8]}
  learning_rate: {value: 0.1, apt_list: [0.01, 0.05, 0.1, 0.2, optimal,
adaptive]}
  max_leaf_nodes: {value: 31, apt_list: [15, 31, 63]}
  l2_regularization: {value: 0.0, apt_list: [0.0, 0.1, 0.5, 1.0, 5.0]}
  kernel: {value: rbf, apt_list: [linear, rbf, poly]}
  degree: {value: 3, apt_list: [2, 3, 4]}
  gamma: {value: scale, apt_list: [scale, auto]}
  coef0: {value: 0.0, apt_list: [0.0, 0.1, 0.5, 1.0]}
  shrinking: false
  shrinkage: {value: auto, apt_list: [null, auto, 0.1, 0.5, 0.9]}
  eta0: {value: 0.01, apt_list: [0.001, 0.01, 0.1, 1.0]}
  power_t: {value: 0.25, apt_list: [0.1, 0.25, 0.5, 0.75]}
  hidden_layer_sizes: {value: [100], apt_list: [[50], [100], [200], [50,
50], [100, 50], [200, 100]]}
  activation: {value: relu, apt_list: [tanh, relu]}
  early_stopping: true

```

Listing C.4: Uninstantiated YAML configuration for KNoLO optimization: Model Selection.

Bibliography

- [1] Nour Abdallah, Andrew Wood, Tarik Benidir, Nicholas Heller, Fabian Isensee, Resha Tejpaul, Dillon Corrigan, Chalaïrat Suk-ouichai, Griffin Struyk, Keenan Moore, Nitin Venkatesh, Onuralp Ergun, Alex You, Rebecca Campbell, Erick M. Remer, Samuel Haywood, Venkatesh Krishnamurthi, Robert Abouassaly, Steven Campbell, Nikolaos Papanikolopoulos, and Christopher J. Weight. Ai-generated r.e.n.a.l.+ score surpasses human-generated score in predicting renal oncologic outcomes. *Urology*, 180:160167, October 2023.
- [2] Iram Abdullah, Ali Javed, Khalid Mahmood Malik, and Ghaus Malik. Deepinfusion: A dynamic infusion based-neuro-symbolic ai model for segmentation of intracranial aneurysms. *Neurocomputing*, 551:126510, September 2023.
- [3] Michela Antonelli, Annika Reinke, Spyridon Bakas, Keyvan Farahani, Annette Kopp-Schneider, Bennett A. Landman, Geert Litjens, Bjoern Menze, Olaf Ronneberger, Ronald M. Summers, Bram van Ginneken, Michel Bilello, Patrick Bilic, Patrick F. Christ, Richard K. G. Do, Marc J. Gollub, Stephan H. Heckers, Henkjan Huisman, William R. Jarnagin, Maureen K. McHugo, Sandy Napel, Jennifer S. Golia Pernicka, Kawal Rhode, Catalina Tobon-Gomez, Eugene Vorontsov, James A. Meakin, Sebastien

- Ourselin, Manuel Wiesenfarth, Pablo Arbeláez, Byeonguk Bae, Sihong Chen, Laura Daza, Jianjiang Feng, Baochun He, Fabian Isensee, Yuanfeng Ji, Fucang Jia, Ildoo Kim, Klaus Maier-Hein, Dorit Merhof, Akshay Pai, Beomhee Park, Mathias Perslev, Ramin Rezaiifar, Oliver Rippel, Ignacio Sarasua, Wei Shen, Jaemin Son, Christian Wachinger, Liansheng Wang, Yan Wang, Yingda Xia, Daguang Xu, Zhanwei Xu, Yefeng Zheng, Amber L. Simpson, Lena Maier-Hein, and M. Jorge Cardoso. The medical segmentation decathlon. *Nature Communications*, 13(1), July 2022.
- [4] Manali Arora, Lovepreet Singh Randhawa, Devkant Lakhera, Vishal Thakker, K. Aditya Abhishek, Dipesh Singh, Rohit Chaudhary, and Rajiv Kumar Azad. Interobserver reproducibility of renal nephrometry score in comparison to simplified padua renal nephrometry score. *Urology Annals*, 15(4):373382, August 2023.
- [5] Bruce G. Batchelor and Frederick M. Waltz. *Morphological Image Processing*, pages 801–870. Springer London, London, 2012.
- [6] Mark Bear, Barry Connors, and Michael A Paradiso. *Neuroscience: Exploring the brain*. Jones & Bartlett Learning, 2025.
- [7] Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(1):289–300, 1995.
- [8] PeterH Bennett, ThomasA Burch, and Max Miller. Diabetes mellitus in american (pima) indians. *The Lancet*, 298(7716):125128, July 1971.

- [9] Luca Bergamin, Giovanna Maria Dimitri, and Fabio Aioli. Integrating background knowledge in medical semantic segmentation with logic tensor networks, 2025.
- [10] Merrie Bergmann. *An Introduction to Many-Valued and Fuzzy Logic: Semantics, Algebras, and Derivation Systems*. Cambridge University Press, 2008.
- [11] Taner Bilgiç and I. Burhan Türksen. *Measurement of Membership Functions: Theoretical and Empirical Work*, pages 195–227. Springer US, Boston, MA, 2000.
- [12] Biomarkers Definitions Working Group. Biomarkers and surrogate endpoints: preferred definitions and conceptual framework. *Clin. Pharmacol. Ther.*, 69(3):89–95, March 2001.
- [13] J. Martin Bland and Douglas G. Altman. Statistical methods for assessing agreement between two methods of clinical measurement. *The Lancet*, 327(8476):307–310, 1986.
- [14] G. Bradski. The OpenCV Library. *Dr. Dobb’s Journal of Software Tools*, 2000.
- [15] Matthew S. Brown, Koon-Pong Wong, Liza Shrestha, Muhammad Wahi-Anwar, Morgan Daly, George Foster, Fereidoun Abtin, Kathleen L. Ruchalski, Jonathan G. Goldin, and Dieter Enzmann. Automated endotracheal tube placement check using semantically embedded deep neural networks. *Academic Radiology*, 30(3):412420, March 2023.
- [16] Jerrold T Bushberg, J Anthony Seibert, Edwin M Leidholdt, and John M Boone. *The Essential Physics of Medical Imaging*. Wolters Kluwer, 4th edition, 2020.

- [17] Stephane Chauvie, Lorenzo Nicola Mazzoni, and Jim ODoherty. A review on the use of imaging biomarkers in oncology clinical trials: Quality assurance strategies for technical validation. *Tomography*, 9(5):18761902, October 2023.
- [18] Youngwon Choi, M Wasil Wahi-Anwar, and Matthew S Brown. Simplemind: an open-source software environment that adds thinking to deep neural networks. *Plos one*, 18(4):e0283587, 2023.
- [19] C. J. Clopper and Egon S. Pearson. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 26(4):404–413, 1934.
- [20] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.
- [21] Jacob Cohen. Weighted kappa: Nominal scale agreement with provision for scaled disagreement or partial credit. *Psychological Bulletin*, 70(4):213–220, 1968.
- [22] Heidi Coy, Jonathan R. Young, Michael L. Douek, Matthew S. Brown, James Sayre, and Steven S. Raman. Quantitative computer-aided diagnostic algorithm for automated detection of peak lesion attenuation in differentiating clear cell from papillary and chromophobe renal cell carcinoma, oncocytoma, and fat-poor angiomyolipoma on multiphasic multidetector computed tomography. *Abdominal Radiology*, 42(7):19191928, March 2017.
- [23] M Daly, M Wahi-Anwar, G Melendez-Corres, G Kim, S Raman, and M McNitt-Gray. Investigating the effect of reducing radiation dose on quantitative imaging features

extracted from ct renal perfusion protocol. In *Medical Physics*, volume 49, pages E655–E656. WILEY 111 RIVER ST, HOBOKEN 07030-5774, NJ USA, 2022.

- [24] Elizabeth R. DeLong, David M. DeLong, and Daniel L. Clarke-Pearson. Comparing the areas under two or more correlated receiver operating characteristic curves: A nonparametric approach. *Biometrics*, 44(3):837, September 1988.
- [25] Victor Dubeux, José Fernando Cardona Zanier, Carolina Gianella Cobo Chantong, Fabricio Carrerette, Pedro Nicolau Gabrich, and Ronaldo Damião. Nephrometry scoring systems: their importance for the planning of nephron-sparing surgery and the relationships among them. *Radiologia Brasileira*, 55(4):242252, August 2022.
- [26] Yahia S. Elshakhs, Kyriakos M. Deliparaschos, Themistoklis Charalambous, Gabriele Oliva, and Argyrios Zolotas. A comprehensive survey on delaunay triangulation: Applications, algorithms, and implementations over cpus, gpus, and fpgas. *IEEE Access*, 12:1256212585, 2024.
- [27] Richard A. Erickson, Michael N. Fienen, S. Grace McCalla, Emily L. Weiser, Melvin L. Bower, Jonathan M. Knudson, and Greg Thain. Wrangling distributed computing for high-throughput environmental science: An introduction to htcondor. *PLOS Computational Biology*, 14(10):e1006468, October 2018.
- [28] Andriy Fedorov, Reinhard Beichel, Jayashree Kalpathy-Cramer, Julien Finet, Jean-Christophe Fillion-Robin, Sonia Pujol, Christian Bauer, Dominique Jennings, Fiona Fennessy, Milan Sonka, John Buatti, Stephen Aylward, James V. Miller, Steve Pieper,

- and Ron Kikinis. 3d slicer as an image computing platform for the quantitative imaging network. *Magnetic Resonance Imaging*, 30(9):13231341, November 2012.
- [29] Vincenzo Ficarra, Giacomo Novara, Silvia Secco, Veronica Macchi, Andrea Porzionato, Raffaele De Caro, and Walter Artibani. Preoperative aspects and dimensions used for an anatomical (padua) classification of renal tumours in patients who are candidates for nephron-sparing surgery. *European Urology*, 56(5):786793, November 2009.
- [30] Joseph L Fleiss. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382, 1971.
- [31] The Apache Software Foundation. Apache airflow - a platform to programmatically author, schedule, and monitor workflows. <https://github.com/apache/airflow>, 2019. Accessed: 2026-01-16.
- [32] Kunihiro Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193202, April 1980.
- [33] Yibo Gao, Hangqi Zhou, Zheyao Gao, Bomin Wang, Shangqi Gao, Sihan Wang, and Xiahai Zhuang. *Learning Concept-Driven Logical Rules for Interpretable and Generalizable Medical Image Classification*, page 291300. Springer Nature Switzerland, September 2025.
- [34] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, 4 edition, 2013.

- [35] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Prentice Hall, 2 edition, 2002.
- [36] R.L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, 1(4):132133, June 1972.
- [37] Su-Hyun Han, Ko Woon Kim, SangYun Kim, and Young Chul Youn. Artificial neural network: Understanding the basic concepts without mathematics. *Dementia and Neurocognitive Disorders*, 17(3):83, 2018.
- [38] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.
- [39] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 edition, 2003.
- [40] Matthew H. Hayn, Thomas Schwaab, Willie Underwood, and Hyung L. Kim. Renal nephrometry score predicts surgical outcomes of laparoscopic partial nephrectomy. *BJU International*, 108(6):876881, December 2010.

- [41] Kelei He, Chen Gan, Zhuoyuan Li, Islem Rekik, Zihao Yin, Wen Ji, Yang Gao, Qian Wang, Junfeng Zhang, and Dinggang Shen. Transformers in medical image analysis. *Intelligent Medicine*, 3(1):5978, February 2023.
- [42] Yexiao He, Ziyao Wang, Yuning Zhang, Tingting Dan, Tianlong Chen, Guorong Wu, and Ang Li. Neurosymad: A neuro-symbolic framework for interpretable alzheimer’s disease diagnosis, 2025.
- [43] Nicholas Heller, Fabian Isensee, Klaus H. Maier-Hein, Xiaoshuai Hou, Chunmei Xie, Fengyi Li, Yang Nan, Guangrui Mu, Zhiyong Lin, Miofei Han, Guang Yao, Yaozong Gao, Yao Zhang, Yixin Wang, Feng Hou, Jiawei Yang, Guangwei Xiong, Jiang Tian, Cheng Zhong, Jun Ma, Jack Rickman, Joshua Dean, Bethany Stai, Resha Tejpaul, Makinna Oestreich, Paul Blake, Heather Kaluzniak, Shaneabbas Raza, Joel Rosenberg, Keenan Moore, Edward Walczak, Zachary Rengel, Zach Edgerton, Ranveer Vasdev, Matthew Peterson, Sean McSweeney, Sarah Peterson, Arveen Kalapara, Niranjana Sathianathan, Nikolaos Papanikolopoulos, and Christopher Weight. The state of the art in kidney and kidney tumor segmentation in contrast-enhanced ct imaging: Results of the kits19 challenge. *Medical Image Analysis*, 67:101821, 2021.
- [44] Nicholas Heller, Fabian Isensee, Dasha Trofimova, Resha Tejpaul, Zhongchen Zhao, Huai Chen, Lisheng Wang, Alex Golts, Daniel Khapun, Daniel Shats, Yoel Shoshan, Flora Gilboa-Solomon, Yasmeen George, Xi Yang, Jianpeng Zhang, Jing Zhang, Yong Xia, Mengran Wu, Zhiyang Liu, Ed Walczak, Sean McSweeney, Ranveer Vasdev, Chris Hornung, Rafat Solaiman, Jamee Schoepfoerster, Bailey Abernathy, David Wu, Safa Abdulkadir, Ben Byun, Justice Spriggs, Griffin Struyk, Alexandra Austin,

Ben Simpson, Michael Hagstrom, Sierra Virnig, John French, Nitin Venkatesh, Sarah Chan, Keenan Moore, Anna Jacobsen, Susan Austin, Mark Austin, Subodh Regmi, Nikolaos Papanikolopoulos, and Christopher Weight. The kits21 challenge: Automatic segmentation of kidneys, renal tumors, and renal cysts in corticomedullary-phase ct, 2023.

- [45] Nicholas Heller, Niranjan Sathianathen, Arveen Kalapara, Edward Walczak, Keenan Moore, Heather Kaluzniak, Joel Rosenberg, Paul Blake, Zachary Rengel, Makinna Oestreich, Joshua Dean, Michael Tradewell, Aneri Shah, Resha Tejapaul, Zachary Edgerton, Matthew Peterson, Shaneabbas Raza, Subodh Regmi, Nikolaos Papanikolopoulos, and Christopher Weight. C4kc kits challenge kidney tumor segmentation dataset, 2019.
- [46] Nicholas Heller, Niranjan Sathianathen, Arveen Kalapara, Edward Walczak, Keenan Moore, Heather Kaluzniak, Joel Rosenberg, Paul Blake, Zachary Rengel, Makinna Oestreich, Joshua Dean, Michael Tradewell, Aneri Shah, Resha Tejapaul, Zachary Edgerton, Matthew Peterson, Shaneabbas Raza, Subodh Regmi, Nikolaos Papanikolopoulos, and Christopher Weight. The kits19 challenge data: 300 kidney tumor cases with clinical context, ct semantic segmentations, and surgical outcomes, 2019.
- [47] Nicholas Heller, Resha Tejapaul, Fabian Isensee, Tarik Benidir, Martin Hofmann, P. Blake, Zachary Rengal, Keenan Moore, Niranjan Sathianathen, Arveen Adith Kalapara, J. Rosenberg, Sarah Chan, Edward Walczak, Alexander Kutikov, Robert G. Uzzo, Diego Aguilar Palacios, Erick M. Remer, Steven C. Campbell, Nikolaos Papanikolopoulos, and Christopher J. Weight. Computer-generated r.e.n.a.l. nephrometry scores yield

- comparable predictive results to those of human-expert scores in predicting oncologic and perioperative outcomes. *Journal of Urology*, 207(5):11051115, May 2022.
- [48] Leon Herrmann, Moritz Jokeit, Oliver Weeger, and Stefan Kollmannsberger. *Neural Networks*, page 53122. Springer Nature Switzerland, 2025.
- [49] M.N. Hew, B. Baseskioglu, K. Barwari, P.H. Axwijk, C. Can, S. Horenblas, A. Bex, J.J.M.C.H. de la Rosette, and M.P. Laguna Pes. Critical appraisal of the padua classification and assessment of the r.e.n.a.l. nephrometry score in patients undergoing partial nephrectomy. *Journal of Urology*, 186(1):4246, 2011.
- [50] Jintao Hua and Xu Chen. Accuracy of r.e.n.a.l. nephrometry score in predicting perioperative outcomes of minimally invasive partial nephrectomy: impact of different surgical techniques. *Translational Andrology and Urology*, 14(1):124134, January 2025.
- [51] Fabian Isensee, Paul F Jaeger, Simon AA Kohl, Jens Petersen, and Klaus H Maier-Hein. nnu-net: a self-configuring method for deep learning-based biomedical image segmentation. *Nature methods*, 18(2):203–211, 2021.
- [52] Sébastien Jodogne. The orthanc ecosystem for medical imaging. *Journal of Digital Imaging*, 31(3):341352, May 2018.
- [53] K. Katsanos, L. Mailli, M. Krokidis, A. McGrath, T. Sabharwal, and A. Adam. Systematic review and meta-analysis of thermal ablation versus surgical nephrectomy for small renal tumours. *CardioVascular and Interventional Radiology*, 37(2):427437, January 2014.

- [54] Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.
- [55] Maurice G Kendall and Bernard Babington Smith. The problem of m rankings. *The Annals of Mathematical Statistics*, 10(3):275–287, 1939.
- [56] ImranAhmad Khan, Supriya Basu, RanjitK Das, RanjanK Dey, Dawood Khan, and Vishnu Agarwal. Renal nephrometry score: Predicting perioperative outcomes following open partial nephrectomy. *Urology Annals*, 11(2):187, 2019.
- [57] Kumail Khandwala, Dawar B Khan, Zainab Hussain, Nida Sajjad, and Muhammad Ismail Alvi. Interobserver reliability of the renal nephrometry scoring system: Experience from a developing nation. *Cureus*, November 2020.
- [58] Christopher Koch. *The Brain: Neurons, Synapses, and Neural Networks*, page 14. Springer Nature Switzerland, 2024.
- [59] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.
- [60] Max Kuhn and Kjell Johnson. *Applied Predictive Modeling*. Springer New York, 2013.
- [61] Alexander Kutikov and Robert G. Uzzo. The r.e.n.a.l. nephrometry score: A comprehensive standardized system for quantitating renal tumor size, location and depth. *Journal of Urology*, 182(3):844853, September 2009.

- [62] Hoffman Lab. Core: An effort to enable useful multi-agent systems. <https://gitlab.com/hoffman-lab/core>, 2024. Accessed: 2025-07-31.
- [63] Philippe Lambin, Ralph T.H. Leijenaar, Timo M. Deist, Jurgen Peerlings, Evelyn E.C. de Jong, Janita van Timmeren, Sebastian Sanduleanu, Ruben T.H.M. Larue, Aniek J.G. Even, Arthur Jochems, Yvonka van Wijk, Henry Woodruff, Johan van Soest, Tim Lustberg, Erik Roelofs, Wouter van Elmpt, Andre Dekker, Felix M. Mottaghy, Joachim E. Wildberger, and Sean Walsh. Radiomics: the bridge between medical imaging and personalized medicine. *Nature Reviews Clinical Oncology*, 14(12):749762, October 2017.
- [64] J. Richard Landis and Gary G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159, March 1977.
- [65] 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.
- [66] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436444, May 2015.
- [67] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017.
- [68] Peter Lennie. Receptive fields. *Current Biology*, 13(6):R216R219, March 2003.
- [69] Scott Leslie, Inderbir S. Gill, Andre Luis de Castro Abreu, Syed Rahmanuddin, Karanvir S. Gill, Mike Nguyen, Andre K. Berger, Alvin C. Goh, Jie Cai, Vinay A. Duddalwar,

- Monish Aron, and Mihir M. Desai. Renal tumor contact surface area: A novel parameter for predicting complexity and outcomes of partial nephrectomy. *European Urology*, 66(5):884893, November 2014.
- [70] David Lin et al. A systematic review of contouring guidelines in radiation oncology: Analysis of frequency, methodology, and delivery of consensus recommendations. *International Journal of Radiation Oncology, Biology, Physics*, 2020.
- [71] Lawrence I-Kuei Lin. A concordance correlation coefficient to evaluate reproducibility. *Biometrics*, 45(1):255–268, 1989.
- [72] Geert Litjens, Robert Toth, Wendy van de Ven, Caroline Hoeks, Sjoerd Kerkstra, Bram van Ginneken, Graham Vincent, Gwenael Guillard, Neil Birbeck, Jindang Zhang, Robin Strand, Filip Malmberg, Yangming Ou, Christos Davatzikos, Matthias Kirschner, Florian Jung, Jing Yuan, Wu Qiu, Qinquan Gao, Philip Eddie Edwards, Bianca Maan, Ferdinand van der Heijden, Soumya Ghose, Jhimli Mitra, Jason Dowling, Dean Barratt, Henkjan Huisman, and Anant Madabhushi. Evaluation of prostate segmentation algorithms for mri: The promise12 challenge. *Medical Image Analysis*, 18(2):359373, February 2014.
- [73] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.
- [74] Bradley C. Lowekamp, David T. Chen, Luis Ibáñez, and Daniel Blezek. The design of simpleitk. *Frontiers in Neuroinformatics*, 7, 2013.

- [75] Qiuhao Lu, Rui Li, Elham Sagheb, Andrew Wen, Jinlian Wang, Liwei Wang, Jungwei Fan, and Hongfang Liu. Explainable diagnosis prediction through neuro-symbolic integration. *AMIA Joint Summits on Translational Science proceedings. AMIA Joint Summits on Translational Science*, 2025, 04 2025.
- [76] Andreas Maier, Christopher Syben, Tobias Lasser, and Christian Riess. A gentle introduction to deep learning in medical image processing. *Zeitschrift für Medizinische Physik*, 29(2):86101, May 2019.
- [77] André Costa Matos, Marcos F. Dall’Oglio, José Roberto Colombo Jr, Alexandre Crippa, João A. Q. Juveniz, and Felipe Coelho Argolo. Predicting outcomes in partial nephrectomy: is the renal score useful? *International braz j urol*, 43(3):422431, June 2017.
- [78] Marius E Mayerhoefer, Andrzej Materka, Georg Langs, Ida Häggström, Piotr Szczypiński, Peter Gibbs, and Gary Cook. Introduction to radiomics. *J. Nucl. Med.*, 61(4):488–495, April 2020.
- [79] Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947.
- [80] G Melendez-Corres, Y Choi, M Wahi-Anwar, H Coy, S Raman, and M Brown. Machine reasoning for segmentation of the kidneys on ct images: improving cnn performance by incorporating anatomical knowledge in post-processing. In *Medical Imaging 2023: Computer-Aided Diagnosis*, volume 12465, pages 440–444. SPIE, 2023.

- [81] G. Melendez-Corres, M. W. Wahi-Anwar, Jin Kim, Dallas K. Tada, John M. Hoffman, Joshua Genender, and Matthew S. Brown. Selective segmentation with rejection option: A machine reasoning approach and evaluation metrics., 2026. Submitted; under review.
- [82] Gabriel Melendez-Corres, Kalyani Vyapari, Richard Pais, Koon-Pong Wong, Miya Lopez, Esteban Diaz, Srinivas Kamineni, Steven S. Raman, and Matthew S. Brown. Interpretable automated RENAL nephrometry scoring: A multi-reader variability analysis. *Submitted; under review*, 2026. Manuscript in preparation.
- [83] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation, 2016.
- [84] Shunsuke Miyamoto, Keisuke Goto, Ryo Tasaka, Yuki Kohada, Takafumi Fukushima, Kenshiro Takemoto, Takashi Babasaki, Kohei Kobatake, Yohei Sekino, Hiroyuki Kitano, Kenichiro Ikeda, Keisuke Hieda, Tetsutaro Hayashi, and Nobuyuki Hinata. Centrality angle is a novel nephrometry score to predict tumor complexity and perioperative outcomes for partial nephrectomy. *Scientific Reports*, 14(1), February 2024.
- [85] M. Francesca Monn, Paul T. Gellhaus, Timothy A. Masterson, Aashish A. Patel, Mark Tann, Dylan M. Cregar, and Ronald S. Boris. R.e.n.a.l. nephrometry scoring: How well correlated are urologist, radiologist, and collaborator scores? *Journal of Endourology*, 28(8):10061010, August 2014.
- [86] Sylvia Montag, Nikhil Waingankar, Mostafa A. Sadek, Soroush Rais-Bahrami, Louis R. Kavoussi, and Manish A. Vira. Reproducibility and fidelity of the r.e.n.a.l. nephrometry score. *Journal of Endourology*, 25(12):19251928, December 2011.

- [87] Uzma Nawaz, Mufti Anees-ur Rahaman, and Zubair Saeed. A review of neuro-symbolic ai integrating reasoning and learning for advanced cognitive systems. *Intelligent Systems with Applications*, 26:200541, June 2025.
- [88] Virginie Orgogozo, Baptiste Morizot, and Arnaud Martin. The differential view of genotypeâphenotype relationships. *Frontiers in Genetics*, 6, May 2015.
- [89] Ellen OConnor, Brennan Timm, Nathan Lawrentschuk, and Joseph Ischia. Open partial nephrectomy: current review. *Translational Andrology and Urology*, 9(6):31493159, December 2020.
- [90] Karl Pearson. Mathematical contributions to the theory of evolution. III. regression, heredity, and panmixia. *Philosophical Transactions of the Royal Society of London. Series A*, 187:253–318, 1896.
- [91] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [92] Firas G. Petros, Sarp K. Keskin, Kai-Jie Yu, Roger Li, Michael J. Metcalfe, Bryan M. Fellman, Courtney M. Chang, Cindy Gu, Pheroze Tamboli, Surena F. Matin, Jose A. Karam, and Christopher G. Wood. Intraoperative conversion from partial to radical nephrectomy: Incidence, predictive factors, and outcomes. *Urology*, 116:114119, June 2018.

- [93] UD Reddy, R Pillai, RA Parker, J Weston, NA Burgess, ETS Ho, RD Mills, and MA Rochester. Prediction of complications after partial nephrectomy by renal nephrometry score. *The Annals of The Royal College of Surgeons of England*, 96(6):475479, September 2014.
- [94] C. Richard Boland. Non-coding rna: Its not junk. *Digestive Diseases and Sciences*, 62(5):11071109, March 2017.
- [95] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.
- [96] Tracy L. Rose and William Y. Kim. Renal cell carcinoma: A review. *JAMA*, 332(12):1001, September 2024.
- [97] Bernard Rosner, Robert J. Glynn, and MeiLing T. Lee. The wilcoxon signed rank test for paired comparisons of clustered data. *Biometrics*, 62(1):185192, July 2005.
- [98] Grant D. Schmit, R. Houston Thompson, Anil N. Kurup, Adam J. Weisbrod, Stephen A. Boorjian, Rickey E. Carter, Jennifer R. Geske, Matthew R. Callstrom, and Thomas D. Atwell. Usefulness of r.e.n.a.l. nephrometry scoring system for predicting outcomes and complications of percutaneous ablation of 751 renal tumors. *Journal of Urology*, 189(1):3035, January 2013.
- [99] Donald J. Schuirmann. A comparison of the two one-sided tests procedure and the power approach for assessing the equivalence of average bioavailability. *Journal of Pharmacokinetics and Biopharmaceutics*, 15(6):657–680, 1987.

- [100] Prajit Sengupta and Islem Rekik. FireGNN: Neuro-symbolic graph neural networks with trainable fuzzy rules for interpretable medical image classification. In *New Perspectives in Graph Machine Learning*, 2025.
- [101] Hassan Shahbaz, Preeti Rout, and Mohit Gupta. Creatinine clearance. In *StatPearls*. StatPearls Publishing, Treasure Island (FL), January 2026.
- [102] Paulo Shakarian, Chitta Baral, Gerardo I. Simari, Bowen Xi, and Lahari Pokala. *Neuro Symbolic Reasoning and Learning*. Springer Nature Switzerland, 2023.
- [103] Yoav Shoham. Agent-oriented programming. *Artificial Intelligence*, 60(1):5192, March 1993.
- [104] Patrick E. Shrout and Joseph L. Fleiss. Intraclass correlations: Uses in assessing rater reliability. *Psychological Bulletin*, 86(2):420–428, 1979.
- [105] William Silversmith. connected-components-3d 3.22. 0. 2024.
- [106] Matthew N. Simmons, Christina B. Ching, Mary K. Samplaski, Chin Hyong Park, and Inderbir S. Gill. Kidney tumor location measurement using the c index method. *Journal of Urology*, 183(5):17081713, May 2010.
- [107] Jack Sklansky. Finding the convex hull of a simple polygon. *Pattern Recognition Letters*, 1(2):7983, December 1982.
- [108] Samuel L. Smith, Andrew Brock, Leonard Berrada, and Soham De. Convnets match vision transformers at scale, 2023.

- [109] Ericka M. Sohlberg, Thomas J. Metzner, and John T. Leppert. The harms of overdiagnosis and overtreatment in patients with small renal masses: A mini-review. *European Urology Focus*, 5(6):943945, November 2019.
- [110] Joost J.M. van Griethuysen, Andriy Fedorov, Chintan Parmar, Ahmed Hosny, Nicole Aucoin, Vivek Narayan, Regina G.H. Beets-Tan, Jean-Christophe Fillion-Robin, Steve Pieper, and Hugo J.W.L. Aerts. Computational radiomics system to decode the radiographic phenotype. *Cancer Research*, 77(21):e104e107, October 2017.
- [111] Janita E. van Timmeren, Davide Cester, Stephanie Tanadini-Lang, Hatem Alkadhi, and Bettina Baessler. Radiomics in medical imaginghow-to guide and critical reflection. *Insights into Imaging*, 11(1), August 2020.
- [112] Richard Mast Vilaseca, Antonio Carlos Westphalen, Henrique Ferreira Reis, Orlando Salomão Zogbi, Gyl Eanes Silva, Rodolfo Borges dos Reis, and Valdair Francisco Muglia. Reproducibility and interobserver agreement of the r.e.n.a.l. nephrometry score: focus on imaging features. *Radiologia Brasileira*, 50(1):712, February 2017.
- [113] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0

- Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.
- [114] K Vyapari, G Melendez-Corres, P Teng, V Badheka, D Shah, T Kelly, and MS Brown. Integrating advanced ai into the clinical 3d workflow. In *Proc. of SPIE Vol*, volume 13411, pages 1341109–1, 2025.
- [115] Brian A. Wandell and Jonathan Winawer. *Visual processing*, page 360381. Elsevier, 2025.
- [116] Wenguan Wang, Yi Yang, and Fei Wu. Towards data-and knowledge-driven ai: A survey on neuro-symbolic computing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(2):878899, February 2025.
- [117] Yuzhi Wang, Samantha Wilder, Mohit Butaney, Mahmoud Hijazi, David Gandham, Monica Van Til, Benjamin Goldman, Ji Qi, Mahin Mirza, Anna Johnson, Michael Rudoff, David Wenzler, Craig G. Rogers, and Brian R. Lane. Conversion to radical nephrectomy from robotic partial nephrectomy is most commonly due to anatomic and oncologic complexity. *Journal of Urology*, 211(5):669676, May 2024.
- [118] Jakob Wasserthal, Hanns-Christian Breit, Manfred T. Meyer, Maurice Pradella, Daniel Hinck, Alexander W. Sauter, Tobias Heye, Daniel T. Boll, Joshy Cyriac, Shan Yang, Michael Bach, and Martin Segeroth. Totalsegmentator: Robust segmentation of 104 anatomic structures in ct images. *Radiology: Artificial Intelligence*, 5(5), September 2023.

- [119] Siyuan Wei, Liza Shrestha, Gabriel Melendez-Corres, and Matthew S. Brown. Anatomically guided deep learning system for right internal jugular line (rijl) segmentation and tip localization in chest x-ray. *Life*, 15(2):201, January 2025.
- [120] Andrew M. Wood, Nour Abdallah, Nicholas Heller, Tarik Benidir, Fabian Isensee, Resha Tejpaul, Chalaïrat Sukouichai, Caleb Curry, Alex You, Erick Remer, Samuel Haywood, Steven Campbell, Nikolaos Papanikolopoulos, and Christopher Weight. Fully automated versions of clinically validated nephrometry scores demonstrate superior predictive utility versus human scores. *BJU International*, 133(6):690698, February 2024.
- [121] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*, 9(4):611629, June 2018.
- [122] Jiancheng Yang, Rui Shi, Donglai Wei, Zequan Liu, Lin Zhao, Bilian Ke, Hanspeter Pfister, and Bingbing Ni. Medmnist v2 - a large-scale lightweight benchmark for 2d and 3d biomedical image classification. *Scientific Data*, 10(1), January 2023.
- [123] Matthew Young, Francesca Jackson-Spence, Luis Beltran, Elizabeth Day, Christina Suarez, Axel Bex, Thomas Powles, and Bernadett Szabados. Renal cell carcinoma. *The Lancet*, 404(10451):476491, August 2024.
- [124] Kelly H. Zou, Simon K. Warfield, Aditya Bharatha, Clare M.C. Tempany, Michael R. Kaus, Steven J. Haker, William M. Wells, Ferenc A. Jolesz, and Ron Kikinis. Statistical

- validation of image segmentation quality based on a spatial overlap index1. *Academic Radiology*, 11(2):178189, February 2004.
- [125] Alex Zwanenburg, Martin Vallières, Mahmoud A. Abdalah, Hugo J. W. L. Aerts, Vincent Andrearczyk, Aditya Apte, Saeed Ashrafinia, Spyridon Bakas, Roelof J. Beukinga, Ronald Boellaard, Marta Bogowicz, Luca Boldrini, Irène Buvat, Gary J. R. Cook, Christos Davatzikos, Adrien Depeursinge, Marie-Charlotte Desseroit, Nicola Dinapoli, Cuong Viet Dinh, Sebastian Echegaray, Issam El Naqa, Andriy Y. Fedorov, Roberto Gatta, Robert J. Gillies, Vicky Goh, Michael Götz, Matthias Guckenberger, Sung Min Ha, Mathieu Hatt, Fabian Isensee, Philippe Lambin, Stefan Leger, Ralph T.H. Leijenaar, Jacopo Lenkowicz, Fiona Lippert, Are Losnegård, Klaus H. Maier-Hein, Olivier Morin, Henning Müller, Sandy Napel, Christophe Nioche, Fanny Orlhac, Sarthak Pati, Elisabeth A.G. Pfaehler, Arman Rahmim, Arvind U.K. Rao, Jonas Scherer, Muhammad Musib Siddique, Nanna M. Sijtsema, Jairo Socarras Fernandez, Emiliano Spezi, Roel J.H.M. Steenbakkens, Stephanie Tanadini-Lang, Daniela Thorwarth, Esther G.C. Troost, Taman Upadhaya, Vincenzo Valentini, Lisanne V. van Dijk, Joost van Griethuysen, Floris H.P. van Velden, Philip Whybra, Christian Richter, and Steffen Löck. The image biomarker standardization initiative: Standardized quantitative radiomics for high-throughput image-based phenotyping. *Radiology*, 295(2):328338, May 2020.
- [126] Özgün Çiçek, Ahmed Abdulkadir, Soeren S. Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: Learning dense volumetric segmentation from sparse annotation, 2016.