FH University of Applied Sciences TECHNIKUM WIEN

MASTER THESIS

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Engineering at the University of Applied Sciences Technikum Wien - Degree Program Data Science

Mask R-CNN: Kidney Segmentation on CT Images with Deep Learning

By: Odo Luo, BSc

Student Number: 2010854016

Supervisor: DI Dr. techn. Matthias Blaickner

Wien, September 30, 2022

Declaration

“As author and creator of this work to hand, I confirm with my signature knowledge of the relevant copyright regulations governed by higher education acts (see Urheberrechtsgesetz /Austrian copyright law as amended as well as the Statute on Studies Act Provisions / Examination Regulations of the UAS Technikum Wien as amended).

I hereby declare that I completed the present work independently and that any ideas, whether written by others or by myself, have been fully sourced and referenced. I am aware of any consequences I may face on the part of the degree program director if there should be evidence of missing autonomy and independence or evidence of any intent to fraudulently achieve a pass mark for this work (see Statute on Studies Act Provisions / Examination Regulations of the UAS Technikum Wien as amended).

I further declare that up to this date I have not published the work to hand nor have I presented it to another examination board in the same or similar form. I affirm that the version submitted matches the version in the upload tool.“

Wien, September 30, 2022 Signature

Kurzfassung

Eine der größten medizinischen Durchbrüche war die Entwicklung der Computertomographie, welche ein dreidimensionales Bild des menschlichen Körpers liefert. Weitere Forschungen konzentrieren sich nicht nur auf die Bildgebung, sondern auch auf die Bildverarbeitung. In den letzten Jahren wurde die künstliche Intelligenz für die Bildverarbeitung eingeführt, indem künstliche Neural Networks zur Klassifizierung, Segmentierung und Erkennung von Objekten wie z.B. Organen, in Bildern eingesetzt wurden. Im Jahre 2021 gelang “nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation” [12] mit der Einführung des nnU-Net Networks, das andere Algorithmen auf dem Gebiet der biomedizinischen Segmentierung übertraf, ein wichtiger Durchbruch. Andere Deep-Learning Networks könnten jedoch genauere Vorhersagen treffen. Für Convolutional Neural Networks gibt es zwei verschiedene Arten von Frameworks: (a) einstufige Algorithmen und (b) zweistufige Algorithmen. Zweistufige Algorithmen schneiden bei der Segmentierung mehrerer Objekte besser ab, zum Beispiel bei der Segmentierung beider Nieren des menschlichen Körpers. Deshalb hat “Mask R-CNN” [9] einen hohen Forschungswert [27]. In diesem Zusammenhang analysiert diese Arbeit die Leistung des “Mask R-CNN” Networks und konzentriert sich auf die Beantwortung folgender Forschungsfragen:

1. Ist es möglich ein Mask R-CNN für die automatische Segmentierung auf heterogene CT-Daten zu trainieren, um klinischen Anforderungen gerecht zu werden?

2. Welchen Grad der Übereinstimmung kann mit der manuellen Experten-Segmentierung erreicht werden und wie hoch ist die Laufzeit?

Zur Beantwortung der Fragen wurde ein Mask R-CNN auf öffentlich zugängigen CT - Bildern trainiert. Das System stellte sich unter Verwendung von zweidimensionalen Operationen als nicht geeignet dar. Jedoch zeigte das Experiment die Limitationen des Systems, auf dessen Basis das System deutlich verbessert werden kann. Die Ergebnisse zeigen wie in der Forschung weiter vorzugehen ist.

Schlagworte: Mask R-CNN, Niere, Organ Segmentierung, CT Bilder

Abstract

One of the major medical breakthroughs in medicine and physics was the development of Computed Tomography scans, which deliver a three-dimensional image of the human body. Further research focused not only on medical imaging but also image processing. In recent years, Artificial Intelligence was introduced for image processing purposes by applying Artificial Neural Networks (ANN) to classify, segment or detect objects, such as organs, within images. In 2021 “nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation” [12] made a major breakthrough by introducing the nnU-Net Neural Network which outperformed other algorithms in the field of biomedical image segmentation. Nevertheless, other Deep Learning Networks could provide even more accurate predictions. For Convolutional Neural Networks there are two different types of frameworks: (a) one-stage algorithms; (b) two-stage algorithms. Two-stage algorithms perform better on multi-object segmentation, for example for segmenting both kidneys of the human body. Therefore “Mask R-CNN” [9] has great research value [27].

In this regard, this thesis analyzes the performance “Mask R-CNN” and focuses on answering following research questions:

1. Is it feasible to train a Mask R-CNN for automated kidney segmentation on heterogeneous CT-Data to meet clinical requirements?

2. What degree of congruence can be achieved with manual expert segmentation and what is the runtime of the classification algorithm?

To answer the questions, a Mask R-CNN network was trained on publicly available Computed Tomography (CT) images. The framework proved not to be feasible using a two-dimensional operation. However, the experiment displays the limitation of the framework on which the framework can be improved, clearly. These results indicate how to proceed in further research.

Keywords: Mask R-CNN, Kidney, Organ Segmentation, CT Image

Contents

1 Introduction

To what extent is computer science able to improve medical decision making? Although many technical revolutionary inventions have already been discovered in the past century, the technical advance into the medical field has not stopped yet. One of the major breakthroughs in this field was the discovery of the X-Ray in 1895. First usages of X-Ray's involved the creation of two dimensional images of the human body. These images were able to help to treat different conditions as broken bones. Based on this, CT scans, which deliver a three-dimensional image instead of a two dimensional image, were developed.

Further research focused not only on medical imaging but also image processing. In recent years, Artificial Intelligence (AI) was introduced for image processing purposes by applying Artificial Neural Network (ANN) to classify, segment or detect objects within images. Especially the segmentation e.g., of organs on images, is relevant for the medical field. It is needed for various applications as in detection of lung nodules, tumor analysis, embolism detection or radiotherapeutic treatment planning [28]. The manual contouring of organs is labor intensive and time consuming. Automating segmentation would minimize the needed labor and speed up the process [18]. These segmentation algorithms are then implemented in application as QDOSE [22], a dosimetry software for radio therapy. Nevertheless, there are well known challenges. Unlike natural image segmentation, organ segmentation face challenges like low contrast of organ boundaries or motion artifacts. Furthermore, there is little to no tolerance for mistake, as the slightest error can lead to irreversible damage on a human body. As a consequence, a fully automated segmentation is difficult [18]. Therefore, multiple studies [16, 25] research different methods. In 2021 "nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation" [12] made a major breakthrough by introducing the nnU-Net Neural Network which outperformed other algorithms in the field of biomedical image segmentation. Compared to the traditional manual segmentation of organs, the automated process is more efficient regarding time consumption. Nevertheless, the main goal of both approaches is to minimize the delineation of the segmentation. Although the nnU-Net [12] was able to present remarkable results, other Deep Learning Networks could provide even more accurate predictions. For Convolutional Neural Network (CNN) there are two different types of frameworks: (a) one-stage algorithms; (b) two-stage algorithms. Two-stage algorithms perform better on multi-object segmentation, for example for segmenting both kidneys of the human body. Therefore "Mask R-CNN" [9] has great research value [27].

In this regard, this thesis paper analyzes the performance “Mask R-CNN” and focuses on answering following research questions:

1. Is it feasible to train a Mask R-CNN for automated kidney segmentation on heterogeneous CT-Data to meet clinical requirements?
2. What degree of congruence can be achieved with manual expert segmentation and what is the runtime of the classification algorithm?

Chapter 2 describes the general approach of this thesis. Chapter 3 describes the “Mask R-CNN” [9] framework and its development. Chapter 4 describes the given data, experiment set up and the results with its conclusion. Chapter 5 concludes this thesis, sets its limitations and refers to further research.

2 Methodology

For this thesis a Mask R-CNN network was trained on publicly available CT images. As Mask R-CNN uses a two-stage algorithm and is suitable for multi-object segmentation [27], the experiment focuses on kidneys. Humans mostly have two kidneys and are the only organ within the abdominal region to be so. Therefore, they pose a suitable challenge. The performance of the trained algorithm was measured with following metrics:

- *Accuracy (AC)* expresses the ratio of right predictions to the sum of all prediction. It is defined as:
$$AC = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$
where TP stands for True Positive, TN for True Negative, FP for False Positive and FN for False Negative.
- *Sørensen–Dice Coefficient (DSC)* [24, 3], also known as F1-Score, is defined as:
$$DSC = \frac{2\,|X \cap Y|}{|X| + |Y|} \tag{2}$$
where X is the segmentation mask and Y the ground truth.
- *Jaccard Index (J)* [13] is defined as:
$$J = \frac{|X \cap Y|}{|X \cup Y|} \tag{3}$$
where X is the segmentation mask and Y the ground truth. It expresses the similarity of the two sets.

- *Relative Volume Difference (RVD)* [19, 10] is defined as:

$$RVD = \frac{|X| - |Y|}{|Y|} \tag{4}$$

  where X is the segmentation mask and Y the ground truth.

The metrics were chosen as they are the most common metrics in segmentation task. As a programming language for this project, Matlab was chosen for two reasons: (a) if successfully it shall be implemented into QDOSE [22] in which Matlab was used; (b) Matlab already has a fully implemented Mask R-CNN network. For analytics and visualization, Python is used. The scripts for the experiment can found in the attachments. Considering this thesis evaluates only Mask R-CNN with solely targeting the kidneys and only works with a limited amount of data, this work is regarded as qualitative research. It aims only to give an indication, if Mask R-CNN may be practical for medical image segmentation. The outcome will show if further research is feasible or not.

3 Mask R-CNN

Mask R-CNN is a framework for instance segmentation. Instance segmentation differs from image classification and semantic segmentation.

Classification, detection and semantic segmentation are image processing tasks that have been developed in recent years [9][26][11]:

- *Classification* is defined as the categorizing of an image to a certain category.
- *Object Detection* refers to locating and classifying multiple objects within an image. The location is displayed by a squared bounding box.
- *Semantic Segmentation* is a process where each pixel of an image is classified towards a certain category. This results in a pixel-based mask for each category.

Although image classification is able to determine the existence of an object within an image, it cannot locate it. Object Detection Models are not only able to detect multiple objects at once, but can also locate them. Nevertheless, Object Detection Models cannot determine the shape of an object. Therefore, Semantic Segmentation was introduced, classifying each pixel and hence defining the shape for each category. However, Semantic Segmentation treats all pixels of a category as one object and cannot distinguish between different instances. Instance Segmentation provides this ability to discern multiple instances of a category.

3.1 (Fast/er) R-CNN

Mask R-CNN ability for Instance Segmentation is the end result of the development through three different networks:

- *R-CNN* [6] uses three modules for Object Detection. First, it uses Selective Search to generate a list of Region of Interests (RoI). Second, each proposal is processed through a CNN for feature extraction. The third and last step is the classification and localization of the RoI.
- *Fast R-CNN* [5] processes the entire image through a CNN to generate a feature map. Each RoI is rendered from the feature map and transformed into a uniform size using RoI Pooling. The RoI is then fed to a fully connected network in order to be classified and localized by a SoftMax layer at the end.
- *Faster R-CNN* [23] switches the selective search algorithm with a Region Proposal Network and thus significantly improves the prediction time.

R-CNN [6] provides the basic architecture for classification and localization. Faster R-CNN [5] improves it in multiple ways. It is using the entire image instead of each RoI for the Convolutional Neural Network and thus increases the training speed. Additionally, the proposals are combined to one batch, which increases the speed even further. Furthermore, the RoI Pooling puts the proposals into the same size and thus enabling the usage of a fully connected network to increase the accuracy. Faster R-CNN [23] improves the quality and speed by implementing a Region Proposal Network. This network enables the framework to render proposals faster.

3.2 Mask R-CNN

Mask R-CNN [9] extends Faster R-CNN in two ways (see Figure 1). First, it changes RoI Pooling with RoI Align, which uses bi-linear interpolation to calculate the exact input values based on the four nearest points in each bin. This improves the average precision by approximately three points. Second, Mask R-CNN adds an additional branch to predict a segmentation mask. The additional branch consists of a small fully connected network and predicts a mask for each class. It is only a small computational overhead but enables the network to classify each proposal in a pixel-to-pixel manner.

Like its predecessors, the Mask R-CNN [9] framework feeds the entire image to a CNN and uses a Region Proposal Network to extract proposals from the feature map. Proposals from one feature map are then put together into one batch. Each proposal is cropped to the same size by RoI Align and then fed to a fully connected network. The proposal is then classified by predicting a discrete probability for each k+1 probability, with the additional class being the background. Mask R-CNN takes a threshold between zero and one as hyper-parameter. If the

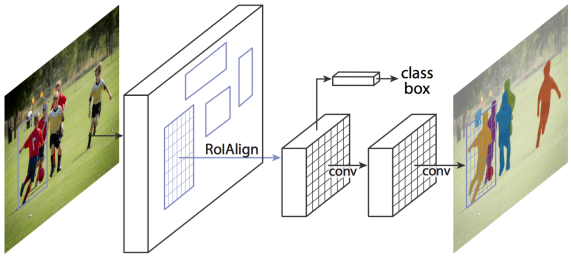


Figure 1: Mask R-CNN framework

probabilities fall under the threshold, the proposal is classified as background. Else, the object is localized while simultaneously the segmentation mask is predicted.

The loss function L is a summation of three different factors:

$$L = L_{cs} + L_{box} + L_{mask} \tag{5}$$

- L_{cs} refers to the classification loss determined by the log loss function $L_{cs}(p, u) = -log(p_u)$. Mask R-CNN calculates a discrete probability $p = (p_0, ..., p_K)$, over K+1 classes, with the additional class being the background and the true class u.

- L_{box} is defined over tuple $u, v = (v_x, v_y, v_w, v_h)$ with v being a vector of bounding boxes for each class and the prediction $t = (t_x^u, t_y^u, t_w^u, t_h^u)$ with u being the true class again. If u is the background the L_{box} is 0, otherwise

$$L_{box}(t^u, v) = \sum_{i\epsilon\{x,y,w,h\}} smooth_{L_1}(t_i^u - v_i) \tag{6}$$

  in which

$$smooth_{L_1}(s = t_i^u - v_i) = \begin{cases} 0.5s^2 & if|s| < 1 \\ |s| - 0.5 & otherwise, \end{cases} \tag{7}$$

  is a L_1 loss.

- L_{cs} is similar to L_{box} defined as average cross entropy loss. The predicted mask is compared to the ground truth.

Although Mask R-CNN predicts bounding boxes and mask for each class, only the losses of the true class from the ground truth contributes to the loss function. This decouples the mask prediction from the classification task. As a result, the mask branch can generate a mask for each class without competition among each other. Each ROI acts as a sample and all ROI from an image are used as one mini batch. Mask R-CNN then uses Stochastic Gradient Descent [15] to optimize the model. For details see *Fast R-CNN* [5] and “Mask R-CNN” [9].

4 Application

As described in chapter 2, the experiment is conducted with different CT images. These CT images are provided by Sharok Kimiaei (SK). SK is a developer of the QDOSE Software [22]. The QDOSE Software is a dosimetry software used for individual radiotherapy treatment planning. As part of the software, CT images are loaded and the organs are automatically segmented. Used images are gathered by SK from public sources and used as training and validation data.

4.1 Data Description

In total there are 210 different CT images and 210 label masks. Each image consists of approximately 216 slices with an average size of 512:512. For the experiment the traditional 75 to 25 train-test split was used, resulting in 158 CT images for training and 52 CT images for validation. The training data consists of 13127 slices containing a kidney for training, while the test images contain 2700 slices with a kidney. The values of the training and test data are not significantly different, which can be seen in Figure 2. Label masks are encoded with three different values:

- 0 for background
- 1 for kidney
- 2 for tumor

For the purpose of this thesis, tumors are ignored and treated as background with value zero. Both, CT images and label masks are provided in an NIfTI [1, 17] file format.

4.2 Experiment

The network was trained with following environment:

OS Microsoft Windows 10 Education Version 10.0.19044

CPU Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz

RAM 16,0 GB

GPU NVIDIA GeForce GTX 1080 Ti

Before training, the training images described in chapter 4.1 were normalized to reach a value between zero and one. In total, the network is trained over ten epochs. Each epoch consisted of 13127 steps over approximately 6 hours, resulting into a total training time of 60 hours. The Matlab Mask R-CNN implementation [7, 4] uses a ResNet50 [14] backbone and allows to set

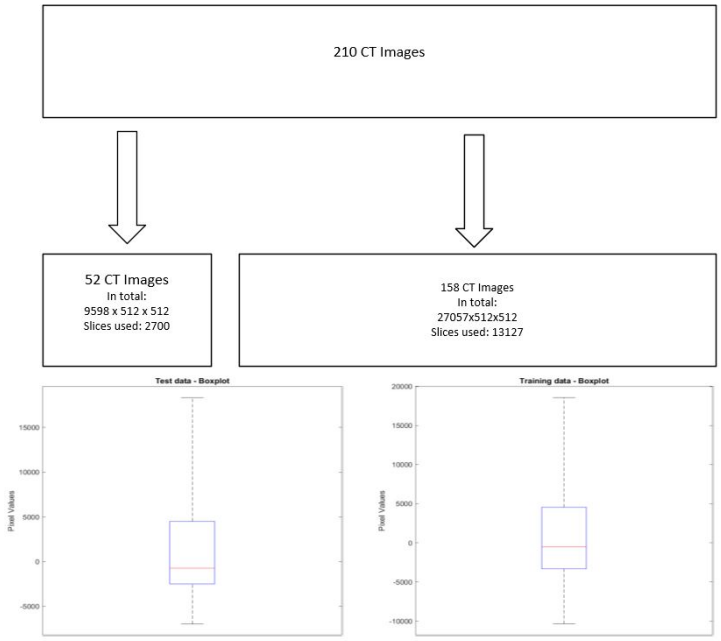


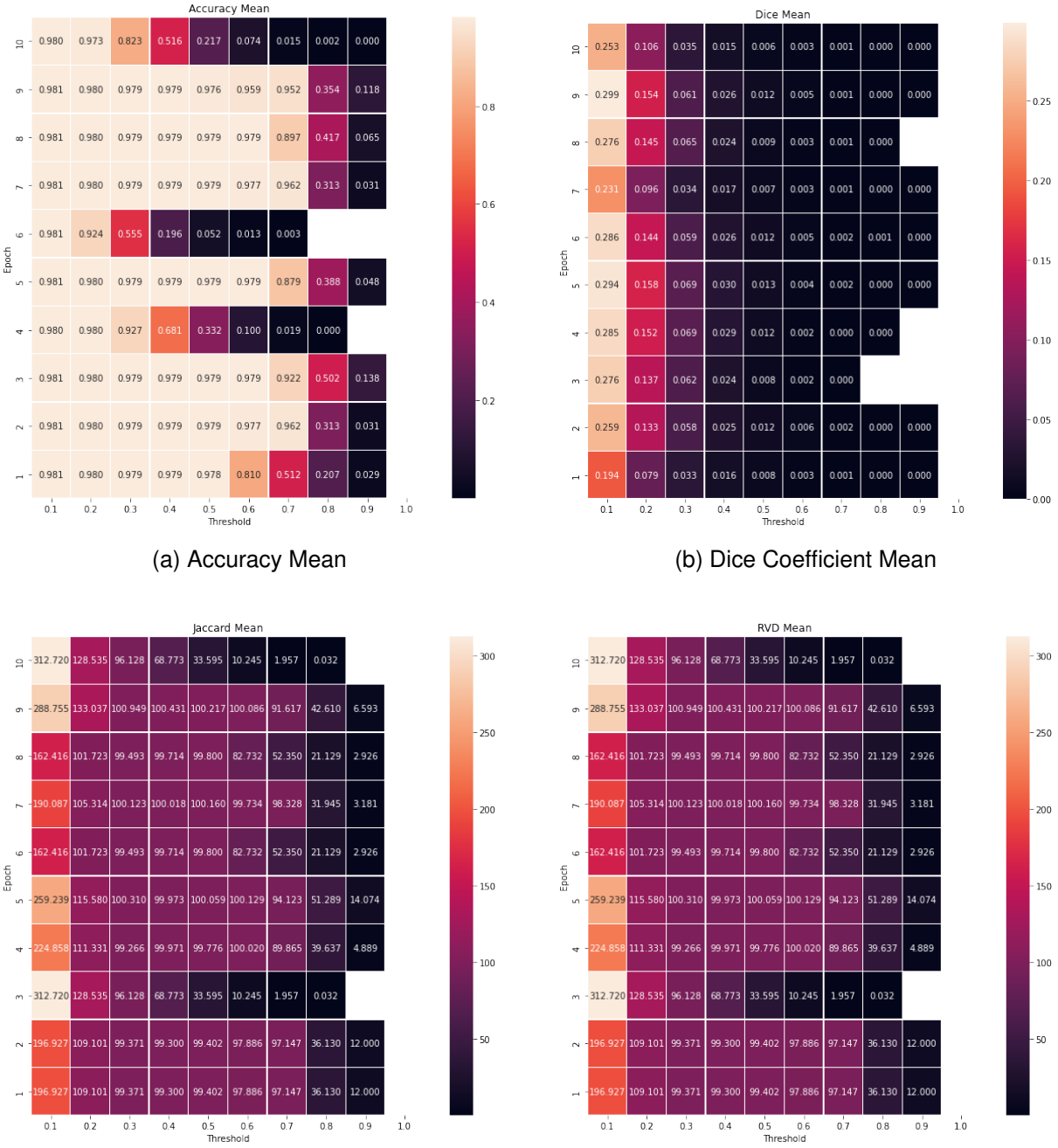
Figure 2: Data description - *Test data:* Min: -6986, 1st: -1001, Median: -744, 3rd: -110, Max: 18326, Var: 348110; *Training data:* Min: -10240, 1st: -1003, Median: -540, 3rd: -109, Max: 18558, Var: 305240

up a threshold as described in chapter 3.2. With the threshold it is possible to increase and decrease the amount of false positive/negative. For the experiment, each of the ten networks was validated against the test data with ten different thresholds ranging from 0.1 to 1.0 with an interval of 0.1. As Mask R-CNN adjust the mini batch size automatically as described in chapter 3.2, no batch size is given. Results can be seen in chapter 4.3. Used metrics are described in chapter 2.

4.3 Results and Discussion

Each network was tested with ten different thresholds against 52 CT images using the metrics described in chapter 4.2. For the sake of simplicity, the results from the 52 CT images are taken as a mean in figure 3.

The results displayed in figure 3 indicate greater results on a lower threshold, which deteriorate as the threshold increases. The performance drops significantly when the threshold reaches 0.5. Noticeably, the DSC performs worse than the accuracy. This indicates a large unbalance between the amount of background and kidney pixels. An example can be seen in figure 4, where the green prediction mask is more than twice the size of the pink ground truth is more than double in size. Additionally, the kidney is also small, compared to the rest of the background.



(a) Accuracy Mean

(b) Dice Coefficient Mean

(c) Jaccard Index Mean

(d) RVD Mean

Figure 3: Mean Average, Dice Coefficient, Jaccard Index and RVD for each epoch from one to ten and threshold 0.1 to 1.0

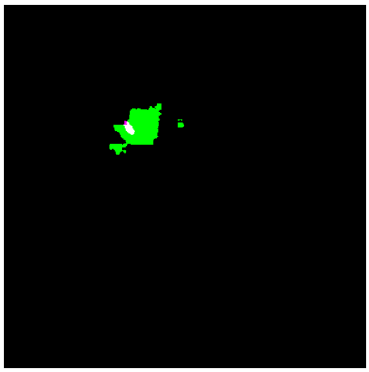


Figure 4: Prediction over ground truth label - Pink: kidney label; Green: prediction mask

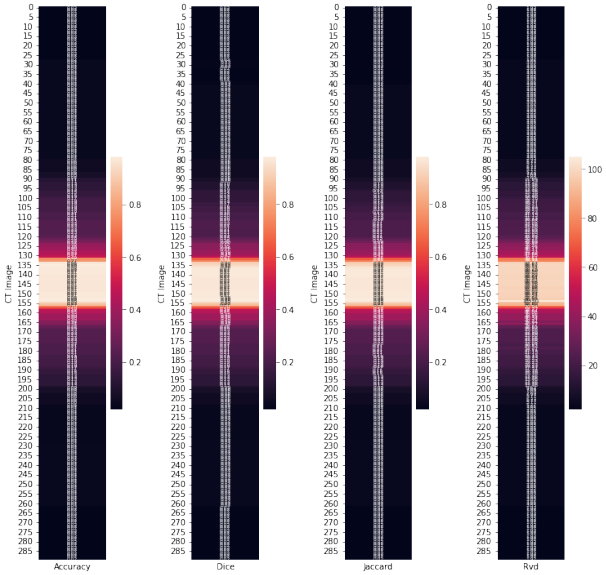


Figure 5: Epoch: 5, Threshold: 0.5 - Mean Accuracy, Jaccard Index, Dice Coefficient and RVD from top to bottom

For further analyzation, results using the network of epoch five and an 0.5 threshold were grouped by CT image and taken as a mean, in order to display the performance of the model for the different parts of the kidney (see figure 5).

The volume distribution can be seen in figure 6. Details are attached at the end of the thesis. J, DSC and RVD in figure 6 show low performance while AC being the only positive indicator. J and DSC display low congruence with the expert segmentation while RVD indicate large differences between prediction and actual segmentation. The visualization of figure 5 clearly shows a better performance in the middle part of the kidney, where it has a more regular and balanced form than at the top or bottom, where unbalanced data is to be expected. This leads to the conclusion, that Mask R-CNN is able to segment objects of a class if the form and size does not change but fails if otherwise. In regards to the research questions, it can be concluded that it is not feasible to train Mask R-CNN for automated kidney segmentation because of the low congruence with expert segmentation. With a segmentation time of approximately

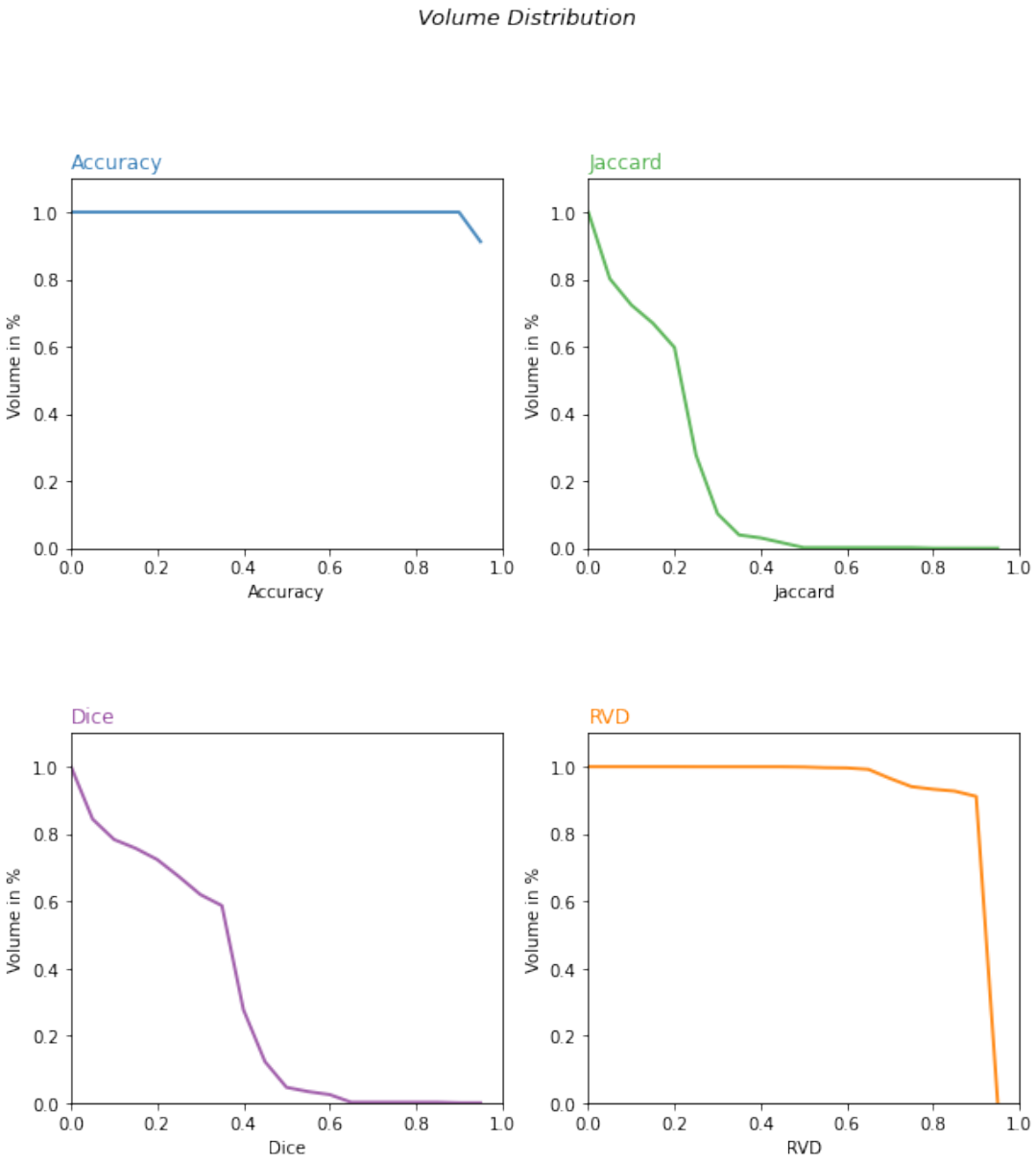


Figure 6: Volume Distribution for Accuracy, Jaccard Index, Dice Coefficient and RVD - epoch 5, threshold 0.5, RVD 10^{-2}

0.1254 seconds per slice or 6.4464 seconds per CT image, the model is fast enough for practical usage, but based on the other metrics, it is not advised. Nevertheless, this conclusion is based on the given data and the usage of a 2D Mask R-CNN implementation of Matlab [7, 4]. Although papers as "*Mask-RCNN and U-net Ensembled for Nuclei Segmentation*" [29] and “Kidney segmentation for quantitative analysis applying MaskRCNN architecture” [21] display the shortcomings of Mask R-CNN with organic images and describe tasks such as kidney segmentation as challenging, different papers suggest better results or improvements. As an example, in “Automated kidney segmentation by mask R-CNN in T2-weighted magnetic resonance imaging” [8] the framework reaches a DSC of 90,5% and a J of 82,8 %, while other papers [2, 27] reach metrics significantly above 90% by developing the framework even further. One of the suggested developments was the change from a two dimensional ROI Align to a three dimensional ROI Align algorithm. Another paper, “Automated and robust organ segmentation for 3D-based internal dose calculation” [20], uses a similar approach to reach a better performance. Even though the aggregated results indicate low performance, they also show high AC with low DSC. This is typical for unbalance data as described above. This supports the theory for the need of further research by improving the current framework.

5 Conclusion

This thesis is an additional contribution that shows the limitations of Mask R-CNN on a two dimensional plane. Although the framework is proven to be efficient with natural images, segmentation on biological images is still a challenge. Mask R-CNN is sufficient if the object has a consistent shape on a two dimensional plane, but not if it varies. As the later circumstance is common with organs, it is not feasible to use Mask R-CNN with a two-dimensional ROI Align for organ segmentation. As described in chapter 4.3, other papers support this statement. Nevertheless, the papers also indicate potential improvement by further developing the current Mask R-CNN model. Furthermore, this thesis only uses only a limited amount of data and is bound to the implementation of Matlab. This has to be considered when evaluating the impact of this study.

Bibliography

[1] Brainder. *The NIFTI file format*. 2012. URL: https://brainder.org/2012/09/23/the-nifti-file-format/.

[2] Cong Chen et al. “Kidney and Tumor Segmentation Using Modified 3D Mask RCNN”. In: *Submissions to the 2019 Kidney Tumor Segmentation Challenge: KiTS19*. University of Minnesota Libraries Publishing. DOI: 10.24926/548719.061.

[3] Lee R. Dice. “Measures of the Amount of Ecologic Association Between Species”. In: *Ecology* 26.3 (1945), pp. 297–302. ISSN: 00129658. DOI: 10.2307/1932409.

[4] *Getting Started with Mask R-CNN for Instance Segmentation - MATLAB & Simulink - MathWorks Deutschland*. 25-Aug-22. URL: https://de.mathworks.com/help/vision/ug/getting-started-with-mask-r-cnn-for-instance-segmentation.html.

[5] Ross Girshick. *Fast R-CNN*. URL: https://arxiv.org/pdf/1504.08083.

[6] Ross Girshick et al. *Rich feature hierarchies for accurate object detection and semantic segmentation*. URL: https://arxiv.org/pdf/1311.2524.

[7] GitHub. *GitHub - matlab-deep-learning/mask-rcnn: Mask-RCNN training and prediction in MATLAB for Instance Segmentation*. 25-Aug-22. URL: https://github.com/matlab-deep-learning/mask-rcnn.

[8] Manu Goyal et al. “Automated kidney segmentation by mask R-CNN in T2-weighted magnetic resonance imaging”. In: *Medical Imaging 2022: Computer-Aided Diagnosis*. Ed. by Karen Drukker. SPIE / International Society for Optical Engineering, 2022, p. 134. ISBN: 9781510649415. DOI: 10.1117/12.2612449. URL: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/12033/2612449/Automated-kidney-segmentation-by-mask-R-CNN-in-T2-weighted/10.1117/12.2612449.full.

[9] Kaiming He et al. “Mask R-CNN”. In: *ICCV 2017*. Proceedings (IEEE International Conference on Computer Vision. Online). Los Alamitos, Washington, and Tokyo: CPS and IEEE Computer Society, op. 2017, pp. 2980–2988. ISBN: 978-1-5386-1032-9. DOI: 10.1109/ICCV.2017.322.

[10] Tobias Heimann et al. “Comparison and Evaluation of Methods for Liver Segmentation From CT Datasets”. In: *IEEE Transactions on Medical Imaging* 28.8 (2009), pp. 1251–1265. DOI: 10.1109/TMI.2009.2013851.

[11] *Image Classification vs Semantic Segmentation vs Instance Segmentation | by Nirmala Murali | Medium*. 8.05.2022. URL: https://nirmalamurali.medium.com/image-classification-vs-semantic-segmentation-vs-instance-segmentation-625c33a08d50.

[12] Fabian Isensee et al. “nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation”. In: *Nature Methods* 18.2 (2021), pp. 203–211. ISSN: 1548-7105. DOI: 10.1038/s41592-020-01008-z. URL: https://www.nature.com/articles/s41592-020-01008-z.

[13] Paul Jaccard. “THE DISTRIBUTION OF THE FLORA IN THE ALPINE ZONE.1”. In: *New Phytologist* 11.2 (1912), pp. 37–50. ISSN: 0028-646X. DOI: 10.1111/j.1469-8137.1912.tb05611.x.

[14] K. He et al. “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. ISBN: 1063-6919. DOI: 10.1109/CVPR.2016.90.

[15] J. Kiefer and J. Wolfowitz. “Stochastic Estimation of the Maximum of a Regression Function”. In: *The Annals of Mathematical Statistics* 23.3 (1952), pp. 462–466. ISSN: 0003-4851. DOI: 10.1214/aoms/1177729392.

[16] Hojin Kim et al. “Abdominal multi-organ auto-segmentation using 3D-patch-based deep convolutional neural network”. In: *Scientific Reports* 10.1 (2020), p. 6204. ISSN: 2045-2322. DOI: 10.1038/s41598-020-63285-0. URL: https://www.nature.com/articles/s41598-020-63285-0.

[17] Henry Knipe and Candace Moore. “NIfTI (file format)”. In: *Radiopaedia.org*. Radiopaedia.org, 2005. DOI: 10.53347/rID-72562.

[18] Yingzi Liu et al. “CT-based multi-organ segmentation using a 3D self-attention U-net network for pancreatic radiotherapy”. In: *Medical physics* 47.9 (2020), pp. 4316–4324. DOI: 10.1002/mp.14386.

[19] Ying-Hwey Nai et al. “Comparison of metrics for the evaluation of medical segmentations using prostate MRI dataset”. In: *Computers in biology and medicine* 134 (2021), p. 104497. DOI: 10.1016/j.compbiomed.2021.104497.

[20] Mahmood Nazari et al. “Automated and robust organ segmentation for 3D-based internal dose calculation”. In: *EJNMMI research* 11.1 (2021), p. 53. ISSN: 2191-219X. DOI: 10.1186/s13550-021-00796-5.

[21] Mathilde Overgaard Lauersen et al. “Kidney segmentation for quantitative analysis applying MaskRCNN architecture”. In: *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*. Piscataway, New Jersey: IEEE, 2021, pp. 1–6. ISBN: 978-1-7281-9048-8. DOI: 10.1109/SSCI50451.2021.9660052.

[22] *QDOSE | Personalised Dosimetry in Molecular Radiotherapy*. 23-Aug-22. URL: https://www.quantitativedose.com/#features.

[23] Shaoqing Ren et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. URL: https://arxiv.org/pdf/1506.01497.

[24] Yutaka Sasaki. “The truth of the F-measure”. In: *Teach Tutor Mater* (2007).

[25] Oliver Schoppe et al. “Deep learning-enabled multi-organ segmentation in whole-body mouse scans”. In: *Nature Communications* 11.1 (2020), p. 5626. ISSN: 2041-1723. DOI: 10.1038/s41467-020-19449-7. URL: https://www.nature.com/articles/s41467-020-19449-7.

[26] Pulkit Sharma. “Image Classification vs. Object Detection vs. Image Segmentation”. In: *Analytics Vidhya* (21.08.2019). URL: https://medium.com/analytics-vidhya/image-classification-vs-object-detection-vs-image-segmentation-f36db85fe81.

[27] Jian-Hua Shu et al. “An Improved Mask R-CNN Model for Multiorgan Segmentation”. In: *Mathematical Problems in Engineering* 2020 (2020), pp. 1–11. ISSN: 1024-123X. DOI: 10.1155/2020/8351725.

[28] Bram van Ginneken, Cornelia M. Schaefer-Prokop, and Mathias Prokop. “Computer-aided diagnosis: how to move from the laboratory to the clinic”. In: *Radiology* 261.3 (2011), pp. 719–732. DOI: 10.1148/radiol.11091710.

[29] Aarno Oskar Vuola, Saad Ullah Akram, and Juho Kannala. *Mask-RCNN and U-net Ensembled for Nuclei Segmentation*. URL: https://arxiv.org/pdf/1901.10170.

List of Figures

List of Abbreviations

AC	Accuracy
AI	Artificial Intelligence
ANN	Artificial Neural Network
CT	Computed Tomography
CNN	Convolutional Neural Network
DSC	Sørensen–Dice Coefficient
fc	fully connected
IoU	Intersection over Union
J	Jaccard Index
RoI	Region of Interests
RVD	Relative Volume Difference
SVM	Support Vector Machine
SK	Sharok Kimiaei

A Anhang - Detailed results

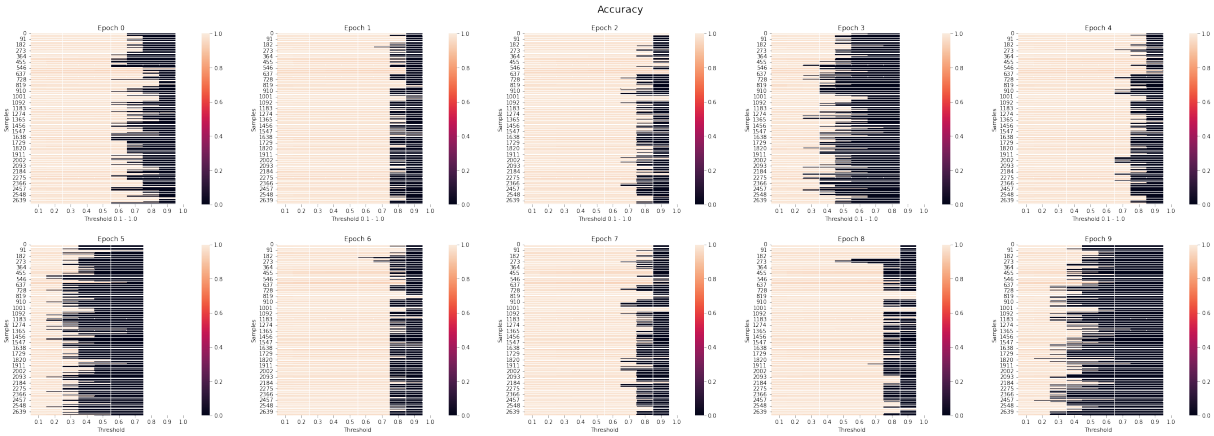


Figure 7: Accuracy per epoch, threshold and sample

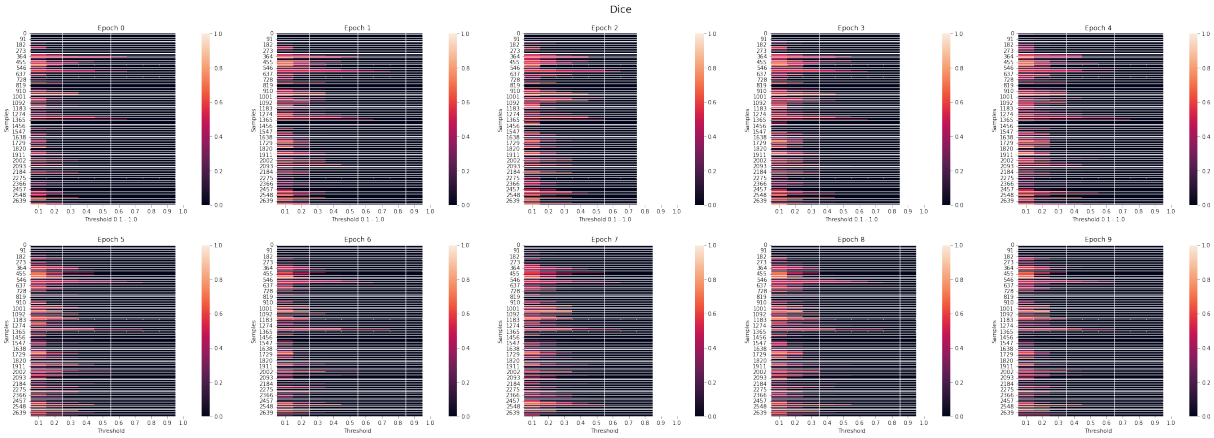


Figure 8: Dice Coefficient per epoch, threshold and sample

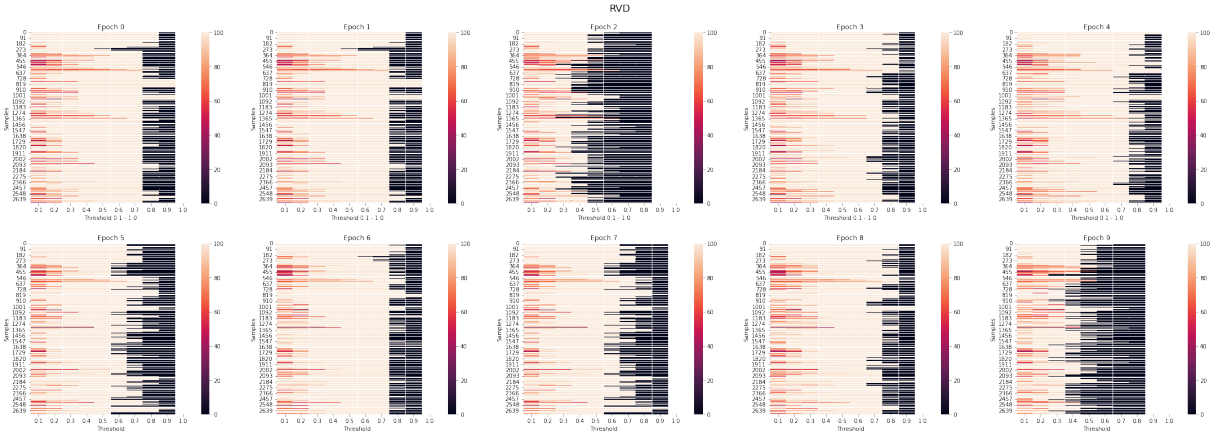


Figure 9: RVD per epoch, threshold and sample

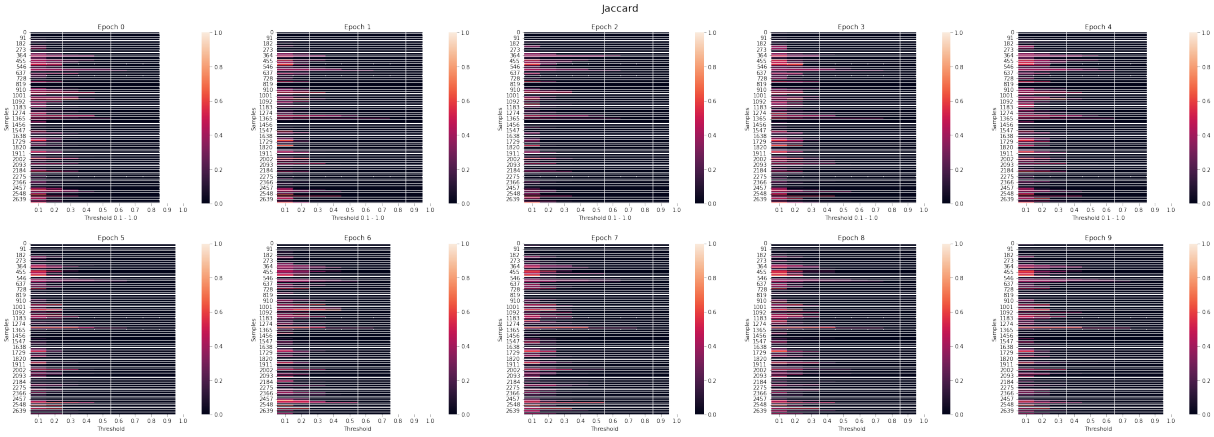


Figure 10: Jaccard Index per epoch, threshold and sample

B Anhang - Code

All Code and data can also be found on https://github.com/Codo155/maskRcnn_Msc

B.1 Matlab Code

Matlab Code

```
function [xCoordinate,yCoordinate,boxWidth,boxHeight]= getCoordinates(i)
    boxWidth=0;
    boxHeight=0;
    xCoordinate=0;
    yCoordinate=0;
    [rows, columns] = size(i);
    parfor row = 1 : rows
        % Make measurements of all lines of 1's.
        props = regionprops(i(row,: ), 'Area');
        % Extract all the lengths into a vector and then put into a cell.
        R{row} = [props.Area];
        R{row}= sum(R{row})
    end
    sumR = cellfun(@sum,R);
    boxWidth=max(sumR);
    if boxWidth ~= 0
        lineMaxWidth = find(sumR==boxWidth);

        for  counter =1 : length(lineMaxWidth)
            rowOfIntrest=i(lineMaxWidth(counter),:);
            y(counter)= find(rowOfIntrest~=0,1,'first');
        end
        yCoordinate=min(y);
        xCoordinate= find(sumR~=0,1,'first');
        boxHeight= find(sumR~=0,1,'last')- xCoordinate;
    end
end
```

Code 1: Code/getCoordinates.m

```
tic
ctDirectory = uigetdir("",'Please select the folder with the CT images.');

```

```
labelDirectory = uigetdir("",'Please select the folder with the labels.');


rgbImagesFolder = uigetdir("",'Please select the destination folder for the RGB
    images');
checkpointPath = uigetdir("",'Please select the destination folder for
    checkpoints');

ctFiles = dir(strcat(ctDirectory,'\*.nii.gz'));

counter=int16(1);
masks={};
for i=1:length (ctFiles)
    try
        labelFile = strcat(labelDirectory,'\',ctFiles(i).name);
        labelImage = load_nii(labelFile).img;

        ctFile=strcat(ctDirectory,'\',ctFiles(i).name);
        ctImage = load_nii(ctFile).img;
    catch exception
        continue;
    end


    labelImageIndex=[];

    [xSlices,ySlices,zSlices] = size(labelImage);

    for z = 1:zSlices
        boxes=[];
        imageCategory={};
        mask=[];
        left = logical(labelImage(1:(xSlices/2),:,z)==1);
        right = logical(labelImage((1+(xSlices/2):xSlices),:,z)==1);

        %% left side
        [xCoordinate,yCoordinate,boxWidth,boxHeight] = getCoordinates(left);
        if boxWidth  ~= 0 && boxHeight ~= 0
            mask(1:(xSlices/2),:,1)=left;
            mask((1+(xSlices/2):xSlices),:,1)=0;
            boxes(1,:)=[xCoordinate yCoordinate boxWidth boxHeight];
        end
        %% right side
        [xCoordinate,yCoordinate,boxWidth,boxHeight] = getCoordinates(right);
        if boxWidth  ~= 0 && boxHeight ~= 0
            [x,y,zMaskSize]=size(mask);
            if(zMaskSize==0 || x==0)
                mask((1+(xSlices/2):xSlices),:,1)=right;
                mask(1:(xSlices/2),:,1)=0;
            else
```

```
                mask(1:(xSlices/2),:,2)=0;
                mask((1+(xSlices/2):xSlices),:,2)=right;
            end
            [xBoxSize,~]=size(boxes);
            if(xBoxSize==0)
                boxes(1,:)=  [ xCoordinate yCoordinate boxWidth boxHeight];
            else
                boxes(2,:)= [ xCoordinate yCoordinate boxWidth boxHeight];
            end
                end
        %%
        masks(counter,1)={logical(mask==1)};
        %% label with bounding boxes
        [x,~]=size(boxes);
        if(x==1)
            box{counter,1}=boxes;
            imageCategory{1}='kidney';
            box{counter,2}=imageCategory;
            counter=counter+1;
            labelImageIndex= [labelImageIndex,z];
        elseif(x==2)
            box{counter,1}=boxes;
            imageCategory=cell(2,1);
            imageCategory{1}='kidney';
            imageCategory{2}='kidney';
            box{counter,2}=imageCategory;
            counter=counter+1;
            labelImageIndex= [labelImageIndex,z];
        end
    end

    for z=1:length(labelImageIndex)
        img=ctImage(:,:,labelImageIndex(z));
        img=uint8(img*255);
        rgb= cat(3,img,img,img);
        outputBaseName=
        strcat(ctFiles(i).name,"_",int2str(labelImageIndex(z)),".png");
        fullDestinationFileName = fullfile(rgbImagesFolder, outputBaseName);
        imwrite(rgb, fullDestinationFileName,'png');
    end
end
ctds = imageDatastore(rgbImagesFolder);

table = cell2table(box,'VariableNames',{'Boxes','Labels'});
lockds= boxLabelDatastore(table);
lables=arrayDatastore(masks,"ReadSize",1,"OutputType","same");
ds= combine(ctds,lockds,lables);


trainClassNames = {'kidney'};
```

```
numClasses = length(trainClassNames);
imageSizeTrain = [512 512 3];
net = maskrcnn("resnet50-coco",trainClassNames,InputSize=imageSizeTrain);

options = trainingOptions("sgdm", ...
    InitialLearnRate=0.002, ...
    LearnRateSchedule="piecewise", ...
    LearnRateDropPeriod=1, ...
    LearnRateDropFactor=0.95 , ...
    Plot="none", ...
    Momentum=0.9, ...
    MaxEpochs=10, ...
    MiniBatchSize=1, ...
    BatchNormalizationStatistics="moving", ...
    ResetInputNormalization=false, ...
    ExecutionEnvironment="gpu", ...
    CheckpointPath=checkpointPath,...
    CheckpointFrequency=1,...
    CheckpointFrequencyUnit='epoch',...
    VerboseFrequency=1000);

modelDateTime = string(datetime("now",Format="yyyy-MM-dd-HH-mm-ss"));
netPath = strcat(checkpointPath,'\',"trainedMaskRCNN-"+modelDateTime+".mat");
infoPath= strcat(checkpointPath,'\',"info-"+modelDateTime+".mat");
doTraining = true;
if doTraining
    [net,info] = trainMaskRCNN(ds,net,options,FreezeSubNetwork="backbone");
    modelDateTime = string(datetime("now",Format="yyyy-MM-dd-HH-mm-ss"));
    save(netPath,"net");
    save(infoPath,"info");
end
toc
```

Code 2: Code/maskRcnn.m

```
directory = uigetdir;
tic
% load files into variable files and get its count
files = dir (strcat(directory,'\*.nii.gz'));
files_count= length (files);

% generating arrays for each dimension
dim_x = ones(1,files_count);
dim_y = ones(1,files_count);
dim_z = ones(1,files_count);

% go through each file

local_min=zeros(files_count);
local_max=zeros(files_count);
```

```
parfor i=1:files_count
    file = strcat(directory,'\',files(i).name);
    image = load_nii(file).img;

    % Set size of images
    s=size(image);
    dim_x(i)= s(1);
    dim_y(i)= s(2);
    dim_z(i)= s(3);
    % Search for global min and max
    %%
    local_max(i)= max(image,[],'all');
    local_min(i)= min(image,[],'all');

end
global_max = max(local_max);
global_min = min(local_min);
global_max=global_max(1);
global_min=global_min(1);

parfor i=1:files_count
    file = strcat(directory,'\',files(i).name);
    image = load_nii(file);
    image.img = mat2gray(image.img,[global_min,global_max]);
    %Todo: Insert destination
    save_nii(image, strcat( Todo:Insert Destination ,files(i).name));
end

toc
disp(mean(dim_x))
disp(mean(dim_y))
disp(mean(dim_z))
```

Code 3: Code/normalization.m

```
ctDirectory = uigetdir("",'Please select the folder with the CT images.');

labelDirectory = uigetdir("",'Please select the folder with the labels.');
ctFiles = dir(strcat(ctDirectory,'\*.nii.gz'));
counter=int16(1);
masks={};
netWorkDirectory= uigetdir("",'Please select the folder with the networks.');


modelNames = dir(strcat(netWorkDirectory,'\*.mat'));

speed_results=[];
for modelIndex=1:length (modelNames)

    modelFile = strcat(netWorkDirectory,'\',modelNames(modelIndex).name);
```

```
    iteration= split(modelFile,"_");
    iteration= char(iteration(4));
    net=load(modelFile);
    net=net.net;

    for i=1:length (ctFiles)
        try
            labelFile = strcat(labelDirectory,'\',ctFiles(i).name);
            labelImage = logical(load_nii(labelFile).img==1);

            ctFile=strcat(ctDirectory,'\',ctFiles(i).name);
            ctImage = gpuArray(load_nii(ctFile).img);

        catch
            continue;
        end

        labelImageIndex =find( sum(labelImage,[1,2])>0);

        if(length(labelImageIndex)<=0)
            continue;
        end
        speed_ctFile=[];
        for z=1:length(labelImageIndex)
            index=labelImageIndex(z);
            img=ctImage(:,:,index);
            img=uint8(img*255);
            rgb= cat(3,img,img,img);
                treshold= 0.5;
            try
                tmp_cunction= @() segmentObjects(net,rgb,Threshold=treshold);
                t = timeit(tmp_cunction);
            catch
                disp('exception');
            end
            speed_ctFile(z)=t;

        end
        speed_results(i,counter)=mean(speed_ctFile);
        ctFile=[];
    end
    counter = counter +1;
end
```

Code 4: Code/speedTest.m

```
tic
ctDirectory = uigetdir("",'Please select the folder with the CT images.');

labelDirectory = uigetdir("",'Please select the folder with the labels.');
```

```
5
6  netWorkDirectory= uigetdir("",'Please select the folder with the networks.');
7
8
9  modelNames = dir(strcat(netWorkDirectory,'\*.mat'));
10
11
12 for modelIndex=1:length (modelNames)
13     modelFile = strcat(netWorkDirectory,'\',modelNames(modelIndex).name);
14     iteration= split(modelFile,"_");
15     iteration= char(iteration(4));
16     net=load(modelFile);
17     net=net.net;
18         ctFiles = dir(strcat(ctDirectory,'\*.nii.gz'));
19         counter=int16(1);
20         masks={};
21         dice_results=[];
22         rvd_results=[];
23         jaccard_results=[];
24         accuracy_results=[];
25         time_results=[];
26         for i=1:length (ctFiles)
27             try
28                 labelFile = strcat(labelDirectory,'\',ctFiles(i).name);
29                 labelImage = logical(load_nii(labelFile).img==1);
30
31                 ctFile=strcat(ctDirectory,'\',ctFiles(i).name);
32                 ctImage = gpuArray(load_nii(ctFile).img);
33
34             catch
35                 continue;
36             end
37
38             labelImageIndex =find( sum(labelImage,[1,2])>0);
39             % spy(labelImage(:,:,z)) to check
40
41             if(length(labelImageIndex)<=0)
42                 continue;
43             end
44
45             for z=1:length(labelImageIndex)
46                 index=labelImageIndex(z);
47                 img=ctImage(:,:,index);
48                 img=uint8(img*255);
49                 rgb= cat(3,img,img,img);
50
51                 for t=5:5
52                     dice_res=NaN;
53                     rvd=NaN;
54                     jaccard_res=NaN;
```

```
55                      accuracy=NaN;
56
57                      treshold= 0.1*t;
58                      try
59                          [masks,labels,~] =
                                segmentObjects(net,rgb,Threshold=treshold);
60                          timeFunction =
                                @()segmentObjects(net,rgb,Threshold=treshold);
61                          time_result=timeit(timeFunction);
62                      catch
63                          disp('exception');
64                          continue;
65                      end
66                      mask_result= any(masks,3);
67                      groundTruth=labelImage(:,:,index);
68                      if all(size(mask_result)>0)
69
70                          dice_res = dice(mask_result,groundTruth);
71                          difference = abs((mask_result-groundTruth));
72                          rvd = 100*(sum(difference,"all")/sum(groundTruth,"all"));
73                          if(rvd > 1000)
74                              disp("RVD over 100 at ctFile: " + ctFiles(i).name +
                                    ...
75                                  "at slice: "+ ...
76                                  string(index) +" with threshold:
                                        "+string(treshold) );
77                          end
78                          jaccard_res = jaccard(mask_result,groundTruth);
79                          accuracy = sum(mask_result ==
                                groundTruth,'all')/numel(groundTruth);
80
81                      end
82                      if ~isnan(dice_res) && all(size(labels) >0)
83                          dice_results(counter,i)=dice_res;
84                      end
85                      if ~isnan(rvd) && all(size(labels) >0)
86                          rvd_results(counter,i)=rvd;
87                      end
88
89                      if ~isnan(jaccard_res) && all(size(labels) >0)
90                          jaccard_results(counter,i)=jaccard_res;
91                      end
92                      if ~isnan(accuracy) && all(size(labels) >0)
93                          accuracy_results(counter,i)=accuracy;
94                      end
95                      time_results(counter,i)=time_result;
96                  end
97                  counter = counter +1;
98              end
99          end
```

```
        dice_results=dice_results;
        rvd_results=rvd_results;
        jaccard_results=jaccard_results;
        accuracy_results=accuracy_results;
end

toc

ncol=size(time_results,2);
for c=1:ncol
    l = length(nonzeros(time_results(:,c)));
    x(1:l,c)= nonzeros(time_results(:,c));
end


idx2keep_columns = sum(abs(x),1)>0 ;
idx2keep_rows    = sum(abs(x),2)>0 ;

time_results = x(idx2keep_rows,idx2keep_columns) ;
```

Code 5: Code/validateAlignKidney.m

```
tic
ctDirectory = uigetdir("",'Please select the folder with the CT images.');

labelDirectory = uigetdir("",'Please select the folder with the labels.');

netWorkDirectory= uigetdir("",'Please select the folder with the networks.');


modelNames = dir(strcat(netWorkDirectory,'\*.mat'));


for modelIndex=1:length (modelNames)

    modelFile = strcat(netWorkDirectory,'\',modelNames(modelIndex).name);
    iteration= split(modelFile,"_");
    iteration= char(iteration(4));
    net=load(modelFile);
    net=net.net;

    [dice_results,rvd_results,jaccard_results,accuracy_results] =
        validationFunction(net,labelDirectory,ctDirectory);
    dice_Name= strcat(netWorkDirectory,"\dice_",iteration,".xls");
    rvd_Name= strcat(netWorkDirectory,"\rvd_",iteration,".xls");
    jaccard_Name= strcat(netWorkDirectory,"\jaccard_",iteration,".xls");
    accuracy_Name= strcat(netWorkDirectory,"\accuracy_",iteration,".xls");

    writematrix(dice_results,dice_Name);
    writematrix(rvd_results,rvd_Name);
```

```
    writematrix(jaccard_results,jaccard_Name) ;
    writematrix(accuracy_results,accuracy_Name) ;

end

toc
```

Code 6: Code/validation.m

```
    function [dice_results,rvd_results,jaccard_results,accuracy_results]=
        validationFunction(net,labelDirectory,ctDirectory)
        ctFiles = dir(strcat(ctDirectory,'\*.nii.gz'));
        counter=int16(1);
        masks={};
        dice_results=[];
        rvd_results=[];
        jaccard_results=[];
        accuracy_results=[];

        for i=1:length (ctFiles)
            try
                labelFile = strcat(labelDirectory,'\',ctFiles(i).name);
                labelImage = logical(load_nii(labelFile).img==1);

                ctFile=strcat(ctDirectory,'\',ctFiles(i).name);
                ctImage = gpuArray(load_nii(ctFile).img);

            catch
                continue;
            end

            labelImageIndex =find( sum(labelImage,[1,2])>0);
            % spy(labelImage(:,:,z)) to check

            if(length(labelImageIndex)<=0)
                continue;
            end

            for z=1:length(labelImageIndex)
                index=labelImageIndex(z);
                img=ctImage(:,:,index);
                img=uint8(img*255);
                rgb= cat(3,img,img,img);

                for t=1:10
                    dice_res=NaN;
                    rvd=NaN;
                    jaccard_res=NaN;
                    accuracy=NaN;

```

```
                    treshold= 0.1*t;
                    try
                        [masks,labels,~] =
                            segmentObjects(net,rgb,Threshold=treshold);
                    catch
                        disp('exception');
                        continue;
                    end
                    mask_result= any(masks,3);
                    groundTruth=labelImage(:,:,index);
                    if all(size(mask_result)>0)

                        dice_res = dice(mask_result,groundTruth);
                        difference = abs((mask_result-groundTruth));
                        rvd = 100*(sum(difference,"all")/sum(groundTruth,"all"));
                        if(rvd > 1000)
                            disp("RVD over 100 at ctFile: " + ctFiles(i).name +
                                ...
                                "at slice: "+ ...
                                string(index) +" with threshold:
                                    "+string(treshold) );
                        end
                        jaccard_res = jaccard(mask_result,groundTruth);
                        accuracy = sum(mask_result ==
                            groundTruth,'all')/numel(groundTruth);

                    end
                    if ~isnan(dice_res) && all(size(labels) >0)
                        dice_results(counter,t)=dice_res;
                    end
                    if ~isnan(rvd) && all(size(labels) >0)
                        rvd_results(counter,t)=rvd;
                    end

                    if ~isnan(jaccard_res) && all(size(labels) >0)
                        jaccard_results(counter,t)=jaccard_res;
                    end
                    if ~isnan(accuracy) && all(size(labels) >0)
                        accuracy_results(counter,t)=accuracy;
                    end
                end
                counter = counter +1;
            end
        end
        dice_results=dice_results;
        rvd_results=rvd_results;
        jaccard_results=jaccard_results;
        accuracy_results=accuracy_results;
    end
```

Code 7: Code/validationFunction.m

```
tic
ctDirectory = uigetdir("",'Please select the folder with the CT images.');

labelDirectory = uigetdir("",'Please select the folder with the labels.');

netWorkDirectory= uigetdir("",'Please select the folder with the networks.');


modelNames = dir(strcat(netWorkDirectory,'\*.mat'));


for modelIndex=1:length (modelNames)
    modelFile = strcat(netWorkDirectory,'\',modelNames(modelIndex).name);
    iteration= split(modelFile,"_");
    iteration= char(iteration(4));
    net=load(modelFile);
    net=net.net;
        ctFiles = dir(strcat(ctDirectory,'\*.nii.gz'));
        counter=int16(1);
        masks={};
        dice_results=zeros(20,1);
        rvd_results=zeros(20,1);
        jaccard_results=zeros(20,1);
        accuracy_results=zeros(20,1);

        for i=1:length (ctFiles)
            try
                labelFile = strcat(labelDirectory,'\',ctFiles(i).name);
                labelImage = logical(load_nii(labelFile).img==1);

                ctFile=strcat(ctDirectory,'\',ctFiles(i).name);
                ctImage = gpuArray(load_nii(ctFile).img);

            catch
                continue;
            end

            labelImageIndex =find( sum(labelImage,[1,2])>0);
            % spy(labelImage(:,:,z)) to check

            if(length(labelImageIndex)<=0)
                continue;
            end
            for z=1:length(labelImageIndex)
                index=labelImageIndex(z);
                img=ctImage(:,:,index);
```

```
47                  img=uint8(img*255);
48                  rgb= cat(3,img,img,img);
49
50                  for t=5:5
51                      dice_res=NaN;
52                      rvd=NaN;
53                      jaccard_res=NaN;
54                      accuracy=NaN;
55
56                      treshold= 0.1*t;
57                      try
58                          [masks,labels,~] =
                                segmentObjects(net,rgb,Threshold=treshold);
59                      catch
60                          disp('exception');
61                          continue;
62                      end
63                      mask_result= any(masks,3);
64                      groundTruth=labelImage(:,:,index);
65                      kidneyPixel=sum(groundTruth==1,'all');
66                      if all(size(mask_result)>0)
67
68                          dice_res = dice(mask_result,groundTruth);
69                          difference = abs((mask_result-groundTruth));
70                          rvd = 100*(sum(difference,"all")/sum(groundTruth,"all"));
71                          rd_res = jaccard(mask_result,groundTruth);
72                          jaccard_res = jaccard(mask_result,groundTruth);
73                          accuracy = sum(mask_result ==
                                groundTruth,'all')/numel(groundTruth);
74
75                      end
76                      pixelCount = numel(mask_result);
77                      if ~isnan(dice_res) && all(size(labels) >0)
78                          for counter= 1:20
79                              ma= (counter*5)/100;
80                              mi= ((counter-1)*5)/100;
81                              if dice_res<=ma && dice_res>mi
82                                  dice_results(counter,1)=
83                                  dice_results(counter,1)+kidneyPixel;
84                              end
85                          end
86                      end
87
88                      if ~isnan(rvd) && all(size(labels) >0)
89                          for counter= 1:20
90                              ma= (counter*5.5);
91                              mi= ((counter-1)*5.5);
92                              if rvd<=ma && rvd>mi
93                                  rvd_results(counter,1)=
94                                  rvd_results(counter,1)+kidneyPixel;
```

```
                        end
                    end
                end

                if ~isnan(jaccard_res) && all(size(labels) >0)
                    for counter= 1:20
                        ma= (counter*5)/100;
                        mi= ((counter-1)*5)/100;
                        if jaccard_res<=ma && jaccard_res>mi
                            jaccard_results(counter,1)=
                            jaccard_results(counter,1)+kidneyPixel;
                        end
                    end
                end
                if ~isnan(accuracy) && all(size(labels) >0)
                    for counter= 1:20
                        ma= (counter*5)/100;
                        mi= ((counter-1)*5)/100;
                        if accuracy<=ma && accuracy>mi
                            accuracy_results(counter,1)=
                            accuracy_results(counter,1)+kidneyPixel;
                        end
                   end
                end
            end
            counter = counter +1;
        end
    end
    dice_sum=sum(dice_results(:,1));
    rvd_sum=sum(rvd_results(:,1));
    jaccard_sum=sum(jaccard_results(:,1));
    accuracy_sum=sum(accuracy_results(:,1));

    dice_length=length(dice_results(:,1));
    rvd_length=length(dice_results(:,1));
    jaccard_length=length(dice_results(:,1));
    accuracy_length=length(dice_results(:,1));

    for l=1:dice_length
        dice_results(l,2) = sum(dice_results(l:dice_length,1))/dice_sum;
    end
    for l=1:rvd_length
        rvd_results(l,2) = sum(rvd_results(l:dice_length,1))/rvd_sum;
    end
    for l=1:jaccard_length
        jaccard_results(l,2) =
            sum(jaccard_results(l:dice_length,1))/jaccard_sum;
    end
    for l=1:accuracy_length
        accuracy_results(l,2) =
```

```
                    sum(accuracy_results(1:dice_length,1))/accuracy_sum;
        end
end

toc
% %
writematrix(accuracy_results,'G:\checkpoint\06\aligDist\accuracy_results.csv') ;
writematrix(jaccard_results,'G:\checkpoint\06\aligDist\jaccard_results.csv') ;
writematrix(rvd_results,'G:\checkpoint\06\aligDist\rvd_results.csv') ;
writematrix(dice_results,'G:\checkpoint\06\aligDist\dice_results.csv') ;
% writematrix(time_results,'G:\checkpoint\06\aligDist\time.csv') ;
```

Code 8: Code/validateKidneyVolumeHistogram.m

B.2 Python visual analyzation of results

Imports

```
import pandas as pd
import io
from google.colab import files
import os
import seaborn as sns;
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
```

Helper Class

```
class helper():
    @staticmethod
    def filesStartWith(name):
        files=[];
        for file in os.listdir("/content"):
          if file.startswith(name):
              files.append(file);
        return files;
    def plotTen(title,data,min=0,max=1):
      fig, ax =plt.subplots(2,5)
      fig.set_figheight(10)
      fig.set_figwidth(30)
      fig.tight_layout(h_pad=5, w_pad=5)
      counter=range(0,10,1);
      for index in counter:
        if index < 5:
          ax[0,index].set_title('Epoch '+ str(index))
          sns.heatmap(data[index],ax=ax[0,index],vmin=min,vmax=max, linewidths=.001)
          ax[0,index].set_xlabel('Threshold 0.1 - 1.0')
          ax[0,index].set_ylabel('Samples');
        else:
          ax[1,(index-5)].set_title('Epoch '+ str(index))
          sns.heatmap(data[index],ax=ax[1,(index-5)],vmin=min,vmax=max, linewidths=.001)
          ax[1,(index-5)].set_xlabel('Threshold')
          ax[1,(index-5)].set_ylabel('Samples');
      fig.suptitle(title, fontsize=18, y=1.05)
```

```
      plt.show()

    def plotMean(title,data):
      plt.figure(figsize=(10,10))
      ax =  sns.heatmap(data.sort_index(ascending=False), linewidths=.3, annot=True, fmt=".3f
      ax.set_title(title)
      ax.set_xlabel('Threshold')
      ax.set_ylabel('Epoch')
      plt.show()

    def formatDf(data):
      lenghts=[len(data[column][data[column]>0]) for column in data ]
      for column in data:
          half = round((290-lenghts[column])/2)
          cl=[]
          cl[0:half]= [0]*half
          cl[half:len(data[column][data[column]>0])]=data[column][data[column]>0]
          tmp= len(data[column][data[column]>0])+half;
          cl[tmp:290]=[0]* (290-tmp)
          data[column]=cl
      return data;
```

File Upload

```
uploaded = files.upload()
```

Choose Files 49 files

- **accuracy.csv**(text/csv) - 73192 bytes, last modified: 9/19/2022 - 100% done
- **accuracy_13127.csv**(text/csv) - 189046 bytes, last modified: 9/17/2022 - 100% done
- **accuracy_26254.csv**(text/csv) - 235856 bytes, last modified: 9/17/2022 - 100% done
- **accuracy_39381.csv**(text/csv) - 252725 bytes, last modified: 9/17/2022 - 100% done
- **accuracy_52508.csv**(text/csv) - 246402 bytes, last modified: 9/17/2022 - 100% done
- **accuracy_65635.csv**(text/csv) - 396342 bytes, last modified: 9/17/2022 - 100% done
- **accuracy_78762.csv**(text/csv) - 362734 bytes, last modified: 9/17/2022 - 100% done
- **accuracy_91889.csv**(text/csv) - 395397 bytes, last modified: 9/17/2022 - 100% done
- **accuracy_105016.csv**(text/csv) - 392733 bytes, last modified: 9/17/2022 - 100% done
- **accuracy_118143.csv**(text/csv) - 392276 bytes, last modified: 9/17/2022 - 100% done
- **accuracy_131270.csv**(text/csv) - 402740 bytes, last modified: 9/17/2022 - 100% done
- **accuracyResults.csv**(text/csv) - 129 bytes, last modified: 9/27/2022 - 100% done
- **dice.csv**(text/csv) - 73236 bytes, last modified: 9/19/2022 - 100% done
- **dice_13127.csv**(text/csv) - 137855 bytes, last modified: 9/17/2022 - 100% done
- **dice_26254.csv**(text/csv) - 159293 bytes, last modified: 9/17/2022 - 100% done
- **dice_39381.csv**(text/csv) - 155871 bytes, last modified: 9/17/2022 - 100% done
- **dice_52508.csv**(text/csv) - 144657 bytes, last modified: 9/17/2022 - 100% done
- **dice_65635.csv**(text/csv) - 158109 bytes, last modified: 9/17/2022 - 100% done
- **dice_78762.csv**(text/csv) - 144208 bytes, last modified: 9/17/2022 - 100% done
- **dice_91889.csv**(text/csv) - 164487 bytes, last modified: 9/17/2022 - 100% done
- **dice_105016.csv**(text/csv) - 158055 bytes, last modified: 9/17/2022 - 100% done
- **dice_118143.csv**(text/csv) - 153859 bytes, last modified: 9/17/2022 - 100% done
- **dice_131270.csv**(text/csv) - 167918 bytes, last modified: 9/17/2022 - 100% done
- **diceRresults.csv**(text/csv) - 443 bytes, last modified: 9/27/2022 - 100% done
- **jaccard.csv**(text/csv) - 73252 bytes, last modified: 9/19/2022 - 100% done
- **jaccard_13127.csv**(text/csv) - 138773 bytes, last modified: 9/17/2022 - 100% done
- **jaccard_26254.csv**(text/csv) - 160318 bytes, last modified: 9/17/2022 - 100% done
- **jaccard_39381.csv**(text/csv) - 156823 bytes, last modified: 9/17/2022 - 100% done
- **jaccard_52508.csv**(text/csv) - 145322 bytes, last modified: 9/17/2022 - 100% done
- **jaccard_65635.csv**(text/csv) - 159034 bytes, last modified: 9/17/2022 - 100% done
- **jaccard_78762.csv**(text/csv) - 144905 bytes, last modified: 9/17/2022 - 100% done
- **jaccard_91889.csv**(text/csv) - 165458 bytes, last modified: 9/17/2022 - 100% done
- **jaccard_105016.csv**(text/csv) - 158963 bytes, last modified: 9/17/2022 - 100% done
- **jaccard_118143.csv**(text/csv) - 154721 bytes, last modified: 9/17/2022 - 100% done
- **jaccard_131270.csv**(text/csv) - 168888 bytes, last modified: 9/17/2022 - 100% done
- **jaccardResults.csv**(text/csv) - 402 bytes, last modified: 9/27/2022 - 100% done
- **rvd.csv**(text/csv) - 37473 bytes, last modified: 9/19/2022 - 100% done
- **rvd_13127.csv**(text/csv) - 151795 bytes, last modified: 9/17/2022 - 100% done
- **rvd_26254.csv**(text/csv) - 177832 bytes, last modified: 9/17/2022 - 100% done
- **rvd_39381.csv**(text/csv) - 181989 bytes, last modified: 9/17/2022 - 100% done
- **rvd_52508.csv**(text/csv) - 167322 bytes, last modified: 9/17/2022 - 100% done
- **rvd_65635.csv**(text/csv) - 200531 bytes, last modified: 9/17/2022 - 100% done
- **rvd_78762.csv**(text/csv) - 184333 bytes, last modified: 9/17/2022 - 100% done
- **rvd_91889.csv**(text/csv) - 212121 bytes, last modified: 9/17/2022 - 100% done
- **rvd_105016.csv**(text/csv) - 211672 bytes, last modified: 9/17/2022 - 100% done
- **rvd_118143.csv**(text/csv) - 203004 bytes, last modified: 9/17/2022 - 100% done
- **rvd_131270.csv**(text/csv) - 216571 bytes, last modified: 9/17/2022 - 100% done
- **rvdRresults.csv**(text/csv) - 393 bytes, last modified: 9/27/2022 - 100% done
- **time.csv**(text/csv) - 61095 bytes, last modified: 9/19/2022 - 100% done

```
Saving accuracy.csv to accuracy.csv
Saving accuracy_13127.csv to accuracy_13127.csv
Saving accuracy_26254.csv to accuracy_26254.csv
```

```
Saving accuracy_39381.csv to accuracy_39381.csv
Saving accuracy_52508.csv to accuracy_52508.csv
Saving accuracy_65635.csv to accuracy_65635.csv
Saving accuracy_78762.csv to accuracy_78762.csv
Saving accuracy_91889.csv to accuracy_91889.csv
Saving accuracy_105016.csv to accuracy_105016.csv
Saving accuracy_118143.csv to accuracy_118143.csv
Saving accuracy_131270.csv to accuracy_131270.csv
Saving accuracyResults.csv to accuracyResults.csv
Saving dice.csv to dice.csv
Saving dice_13127.csv to dice_13127.csv
Saving dice_26254.csv to dice_26254.csv
Saving dice_39381.csv to dice_39381.csv
Saving dice_52508.csv to dice_52508.csv
Saving dice_65635.csv to dice_65635.csv
Saving dice_78762.csv to dice_78762.csv
Saving dice_91889.csv to dice_91889.csv
Saving dice_105016.csv to dice_105016.csv
Saving dice_118143.csv to dice_118143.csv
Saving dice_131270.csv to dice_131270.csv
Saving diceRresults.csv to diceRresults.csv
Saving jaccard.csv to jaccard.csv
Saving jaccard_13127.csv to jaccard_13127.csv
Saving jaccard_26254.csv to jaccard_26254.csv
Saving jaccard_39381.csv to jaccard_39381.csv
Saving jaccard_52508.csv to jaccard_52508.csv
Saving jaccard_65635.csv to jaccard_65635.csv
Saving jaccard_78762.csv to jaccard_78762.csv
```

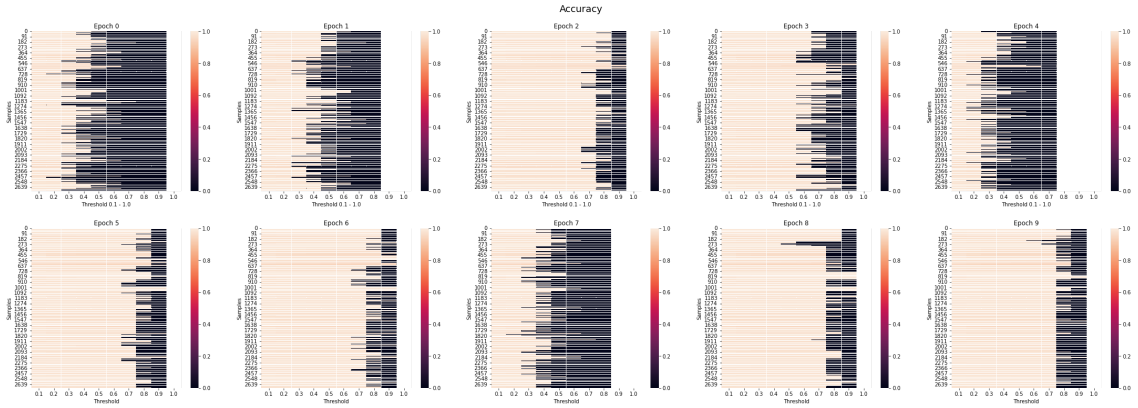
Select Files

```
Saving jaccard_131270.csv to jaccard_131270.csv
```

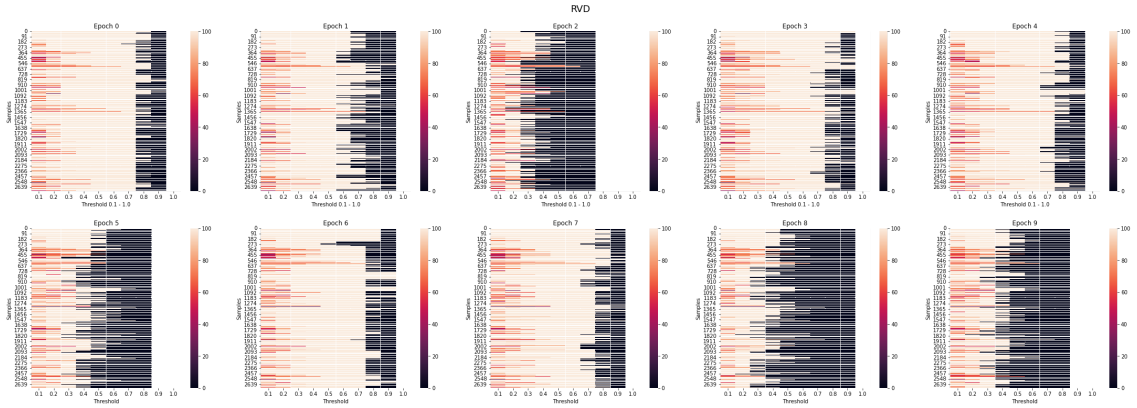
```
col_Names=["0.1", "0.2", "0.3", "0.4","0.5","0.6","0.7","0.8","0.9","1.0"];
files = helper.filesStartWith("accuracy_");
rvd_files= helper.filesStartWith("rvd_");
dice_files= helper.filesStartWith("dice_");
jaccard_files= helper.filesStartWith("jaccard_");
accuracies=[];
rvds=[];
dices=[];
jaccards=[];
for filename in files:
  accuracies.append(pd.read_csv(filename,names=col_Names,sep=';'));
for filename in rvd_files:
  rvds.append(pd.read_csv(filename,names=col_Names,sep=';'));
for filename in dice_files:
  dices.append(pd.read_csv(filename,names=col_Names,sep=';'));
for filename in jaccard_files:
  jaccards.append(pd.read_csv(filename,names=col_Names,sep=';'));
```

Plot results for each epoch and threshold

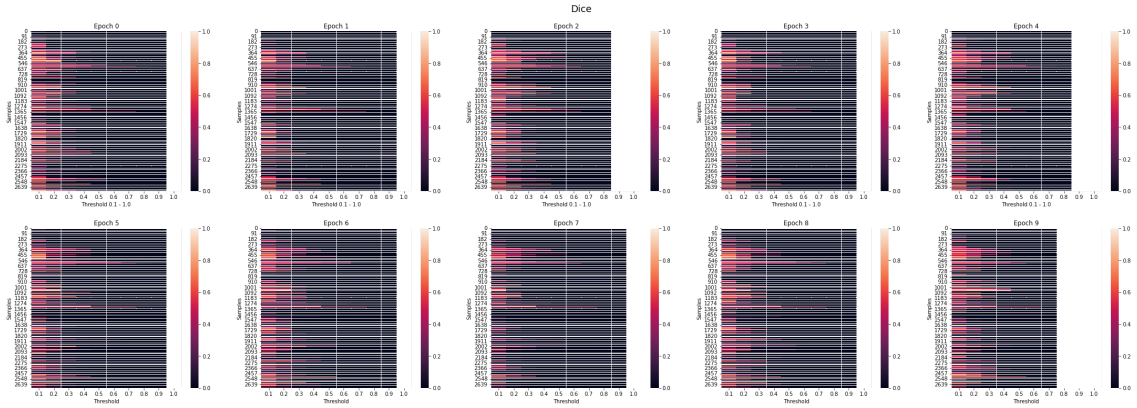
```
helper.plotTen("Accuracy",accuracies)
```




```
helper.plotTen('RVD',rvds,min=0,max=100)
```



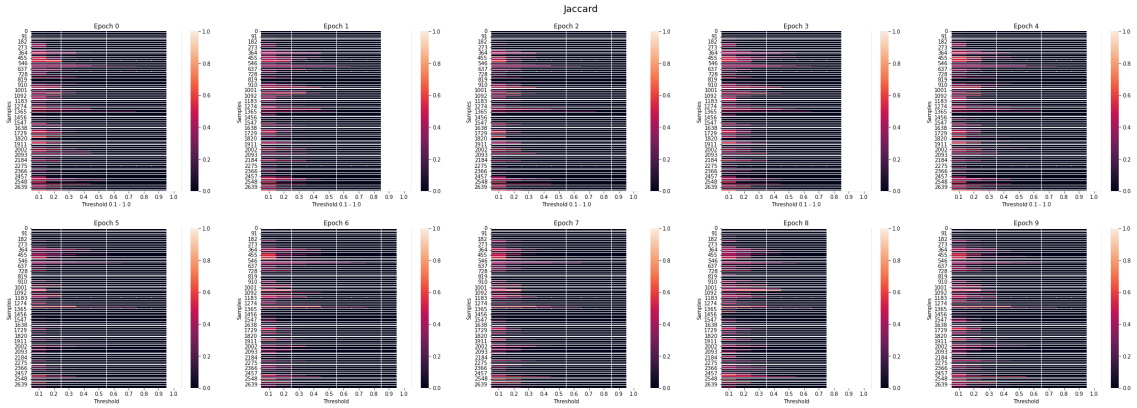

```
helper.plotTen('Dice',dices)
```




```
len(jaccards)
```

```
10
```

```
helper.plotTen('Jaccard',jaccards)
```



Taking the mean for each metric

```
counter=range(0,10,1);
accuracy_mean = pd.DataFrame()
rvd_mean= pd.DataFrame()
dice_mean=pd.DataFrame()
jaccard_mean=pd.DataFrame()
for index in counter:
  accuracy_mean[index]=accuracies[index].mean()
  rvd_mean[index]=rvds[index].mean()
  dice_mean[index]=dices[index].mean()
  jaccard_mean[index]=jaccards[index].mean()
```

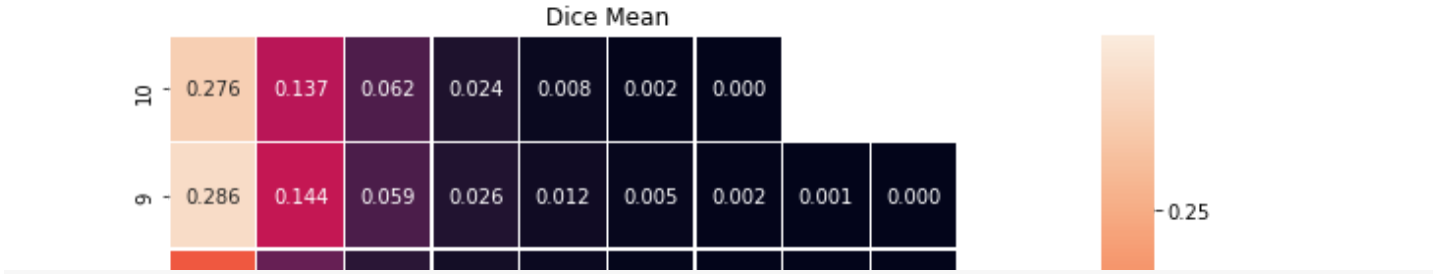
```
counter = range(1,11)
accuracy_mean =accuracy_mean.transpose()
rvd_mean= rvd_mean.transpose()
dice_mean= dice_mean.transpose()
jaccard_mean= jaccard_mean.transpose()

accuracy_mean.index = counter
```

```
rvd_mean.index=counter
dice_mean.index=counter
jaccard_mean.index=counter
```

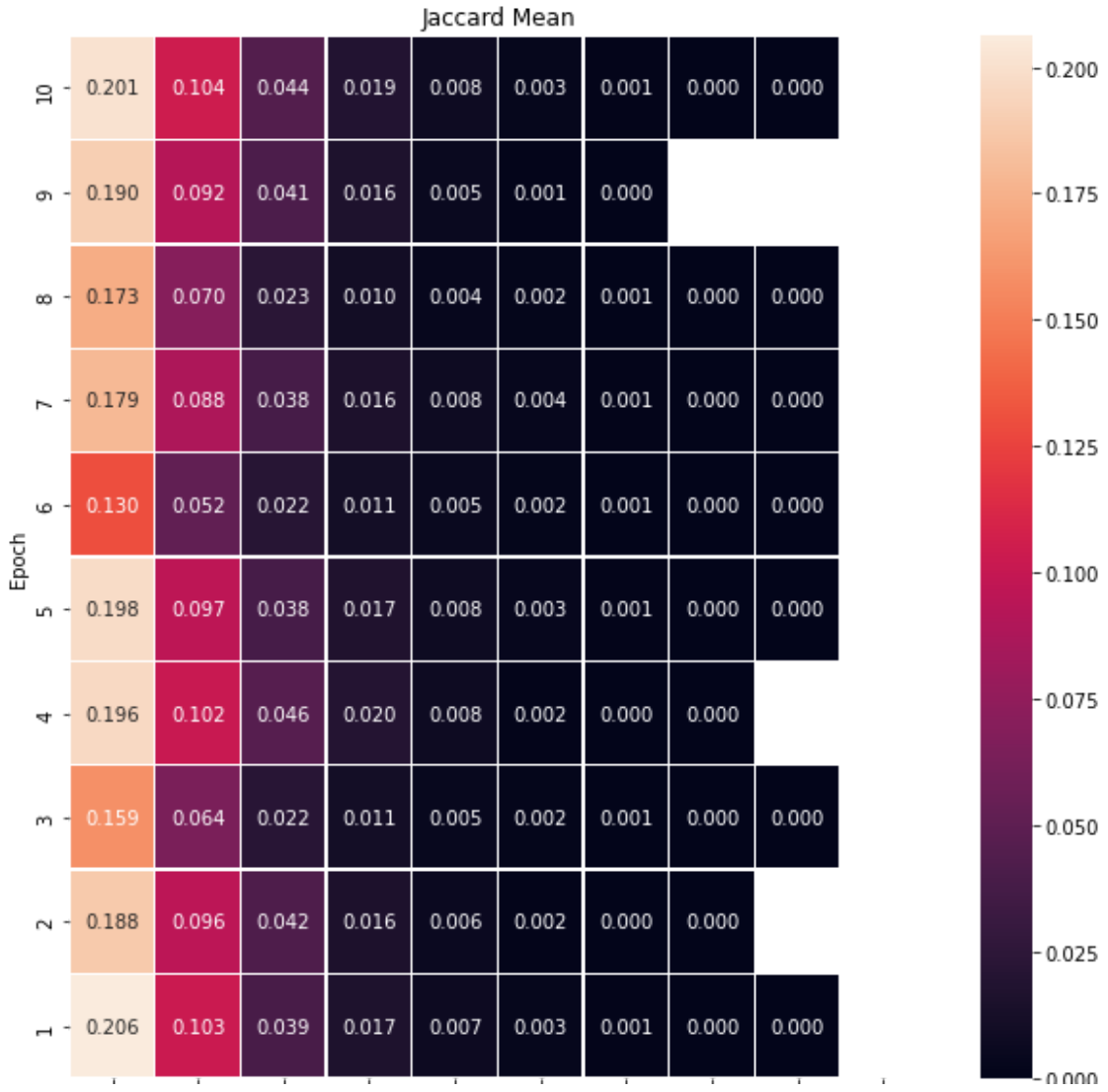
```
helper.plotMean("Accuracy Mean",accuracy_mean)
```


```
helper.plotMean("RVD Mean",rvd_mean)
```


```
helper.plotMean("Dice Mean",dice_mean)
```




```
helper.plotMean("Jaccard Mean",jaccard_mean)
```

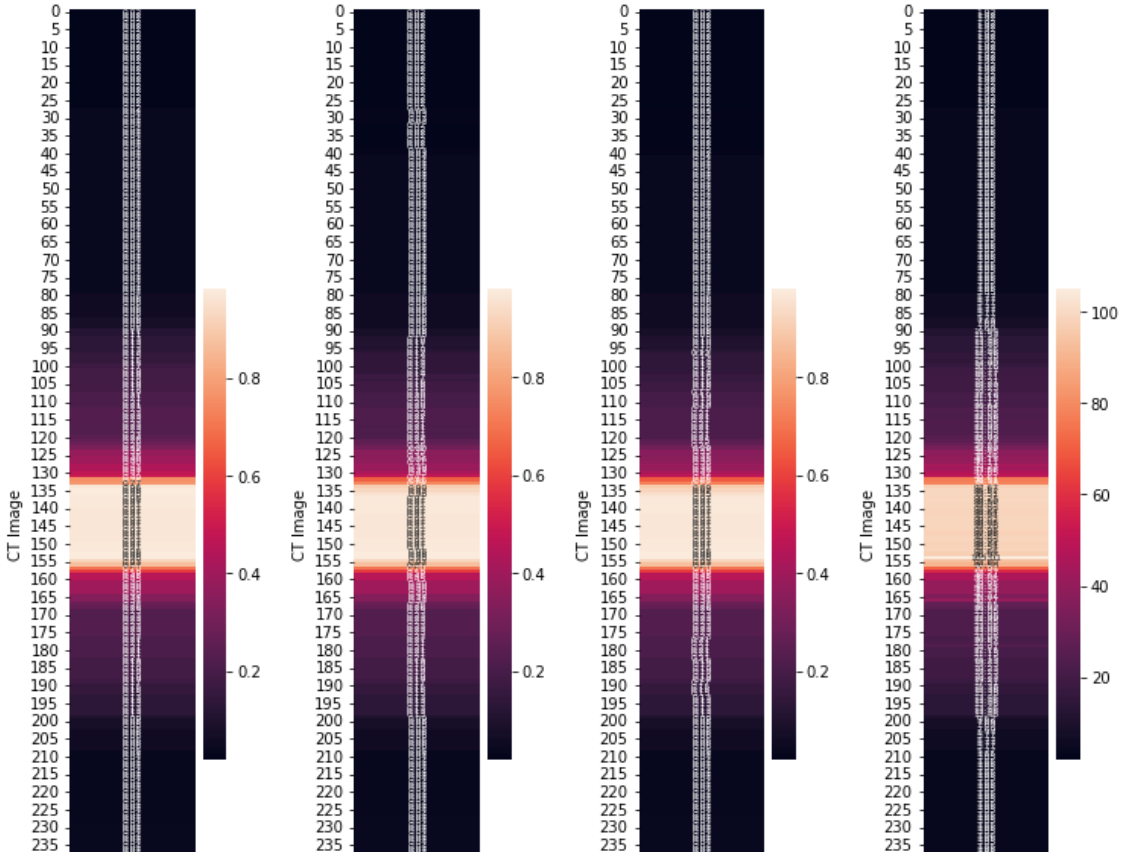


Mean Values over each slice of kidney - top down

```
accuracy=pd.read_csv('accuracy.csv', header=None)
dice=pd.read_csv('dice.csv', header=None)
jaccard=pd.read_csv('jaccard.csv', header=None)
```

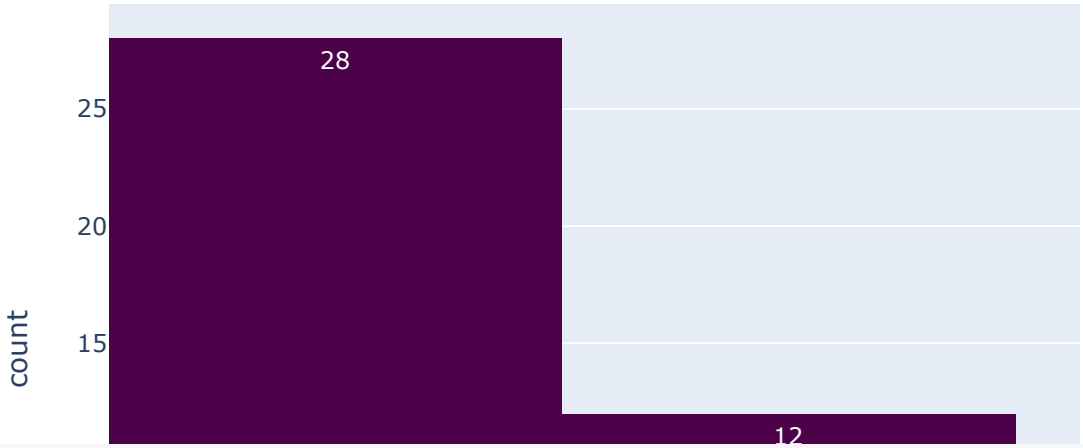
```
rvd=pd.read_csv('rvd.csv', header=None)

accuracy=helper.formatDf(accuracy);
dice=helper.formatDf(dice);
jaccard=helper.formatDf(jaccard);
rvd=helper.formatDf(rvd);
```

```
fig,( ax1,ax2,ax3,ax4) =plt.subplots(ncols=4)
fig.set_figheight(10)
fig.set_figwidth(10)
fig.tight_layout(h_pad=5, w_pad=5)

sns.heatmap( accuracy.mean(axis=1).to_frame(), annot=True, fmt=".2f",annot_kws={"size": 6},ax
ax1.set_title('')
ax1.set_xlabel('Accuracy')
ax1.set_ylabel('CT Image')

sns.heatmap( dice.mean(axis=1).to_frame(), annot=True, fmt=".2f",annot_kws={"size": 6}, ax=ax
ax2.set_title('')
ax2.set_xlabel('Dice')
ax2.set_ylabel('CT Image')

sns.heatmap( jaccard.mean(axis=1).to_frame(), annot=True, fmt=".2f",annot_kws={"size": 6},ax=
ax3.set_title('')
ax3.set_xlabel('Jaccard')
ax3.set_ylabel('CT Image')


sns.heatmap( rvd.mean(axis=1).to_frame(), annot=True, fmt=".2f",annot_kws={"size": 6},ax=ax4,
ax4.set_title('')
ax4.set_xlabel('Rvd')
ax4.set_ylabel('CT Image')

plt.show()
```



Value distribution

```
import plotly.express as px

fig=px.histogram(accuracy.mean(),nbins = 10, color_discrete_sequence=['#4b0249'] ,
                 title='Accuracy Distribution', text_auto=True,  labels={'value':'Accuracy'
fig.update_layout(showlegend=False)
fig.update_layout(
    xaxis = dict(
        tickmode = 'array',
        tickvals = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6,0.7,0.8,0.9,1.0]    )
)

fig.show()
```

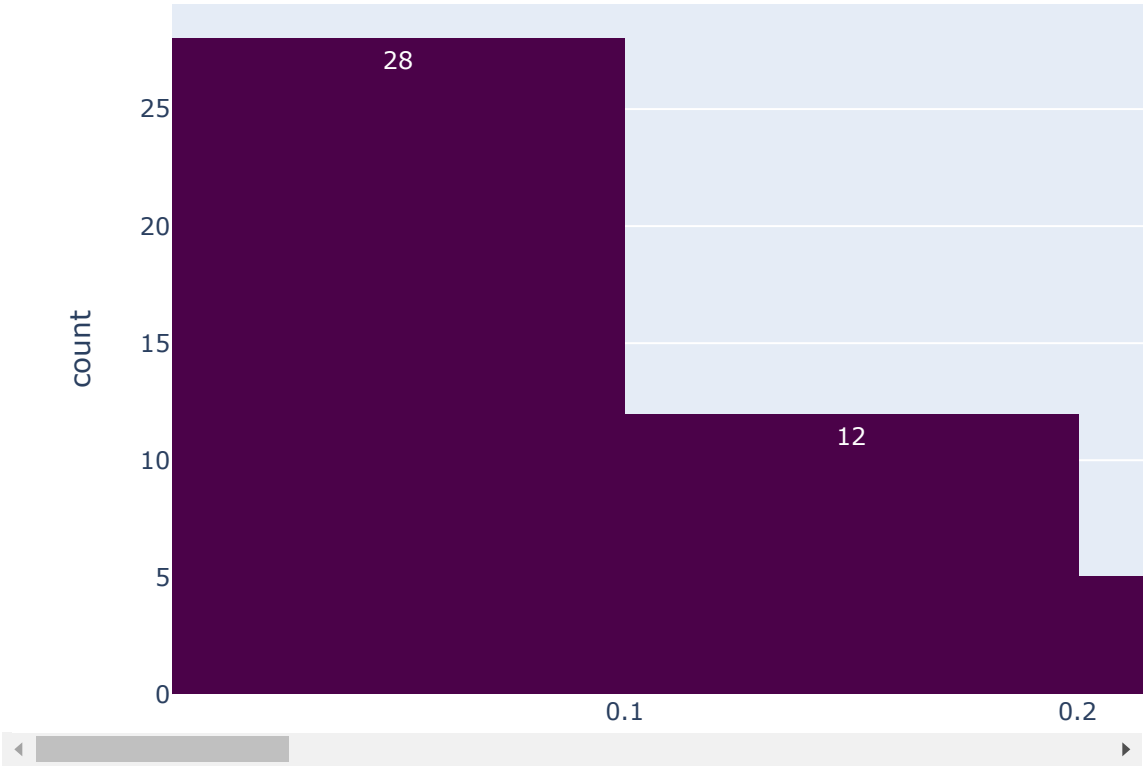



```
fig=px.histogram(dice.mean(),nbins = 10, color_discrete_sequence=['#4b0249'] ,
                  title='Dice Distribution', text_auto=True,  labels={'value':'Dice','count'
fig.update_layout(showlegend=False)
fig.update_layout(
    xaxis = dict(
        tickmode = 'array',
        tickvals = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6,0.7,0.8,0.9,1.0]    )
)

fig.show()
```

Dice Distribution

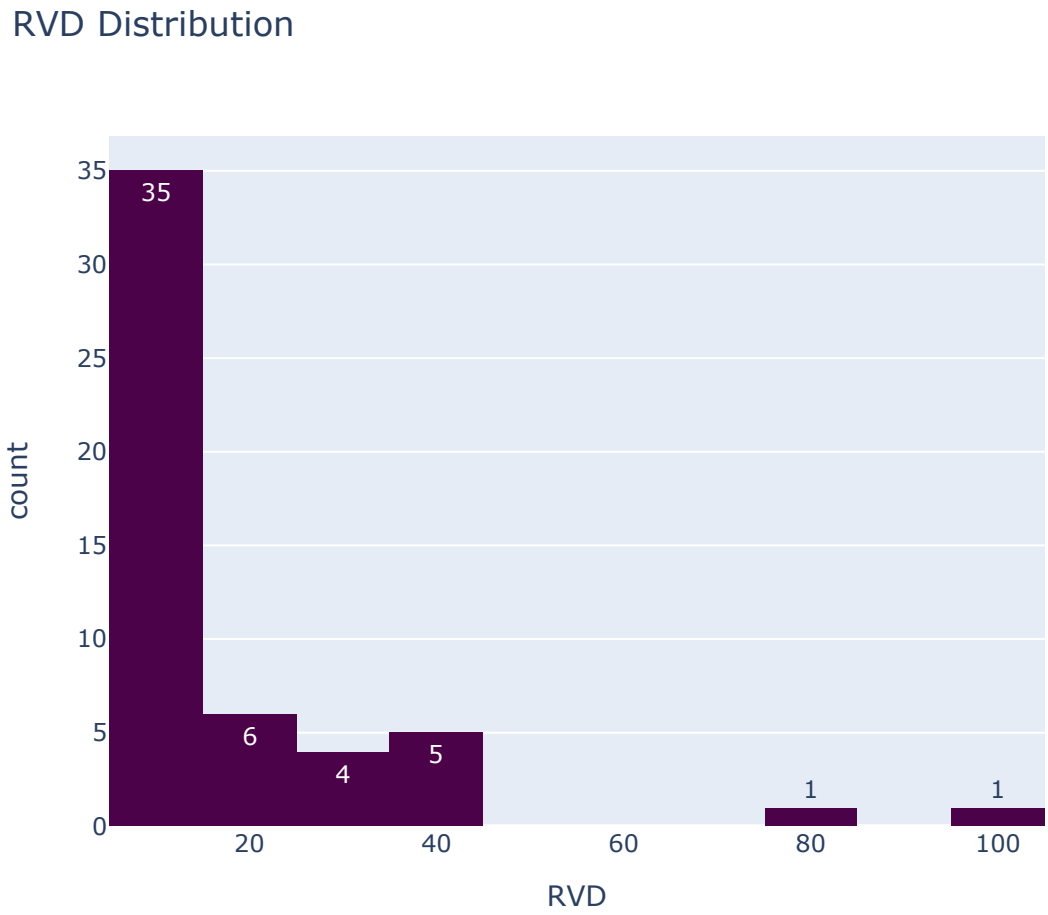
```
fig=px.histogram(jaccard.mean(),nbins = 10, color_discrete_sequence=['#4b0249'] ,
                 title='Jaccard Distribution', text_auto=True,  labels={'value':'Jaccard','
fig.update_layout(showlegend=False)
fig.update_layout(
    xaxis = dict(
        tickmode = 'array',
        tickvals = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6,0.7,0.8,0.9,1.0]    )
)

fig.show()
```

Jaccard Distribution




```
fig=px.histogram(rvd.mean(),nbins = 10, color_discrete_sequence=['#4b0249'] ,
                 title='RVD Distribution', text_auto=True,  labels={'value':'RVD','count':'
fig.update_layout(showlegend=False)
```

```
fig.show()
```

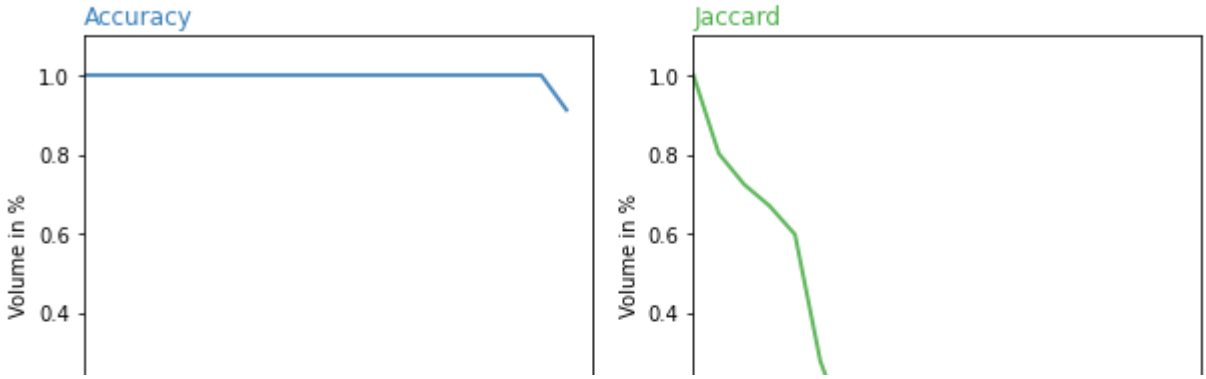



```
accuracyR=pd.read_csv('accuracyResults.csv', header=None)
jaccardR=pd.read_csv('jaccardResults.csv', header=None)
rvdR=pd.read_csv('rvdResults.csv', header=None)
diceR=pd.read_csv('diceResults.csv', header=None)
```

```
r = [*range(20)];
r=[(i+0)*0.05 for i in r]
accuracyR[0]=r
accuracyR.columns=['x','Accuracy']
accuracyR['Jaccard'] = jaccardR[1];
accuracyR['Dice'] = diceR[1];
accuracyR['RVD'] = rvdR[1];
```

```
num=0
palette = plt.get_cmap('Set1')
f = plt.figure()
f.set_figwidth(10)
```

```
f.set_figheight(10)

for column in accuracyR.drop('x', axis=1):
    num+=1

    # Find the right spot on the plot
    plt.subplot(2,2, num)

    # Plot the lineplot
    plt.plot(accuracyR['x'], accuracyR[column], marker='', color=palette(num), linewidth=1.9,
    plt.xlabel(column)
    plt.ylabel("Volume in %")
    # Same limits for every chart
    plt.xlim(0,1.0)
    plt.ylim(0,1.1)

    # Add title
    plt.title(column, loc='left', fontsize=12, fontweight=0, color=palette(num) )

# general title

plt.suptitle("Volume Distribution ", fontsize=13, fontweight=0, color='black', style='italic'
plt.subplots_adjust(hspace = 0.5)



# Show the graph
plt.show()
```



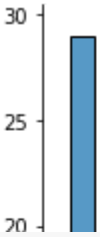
Time analysis

```
time=pd.read_csv('time.csv', header=None)

time.head()
```

	0	1	2	3	4	5	6	
0	0.776024	0.723992	0.730825	0.738769	0.731758	0.729302	0.723156	0.72
1	0.775626	0.722257	0.731860	0.737679	0.728856	0.726785	0.722476	0.72
2	0.742813	0.723129	0.732697	0.736444	0.727294	0.729380	0.722084	0.73
3	0.742685	0.723317	0.735534	0.735868	0.731093	0.733881	0.721323	0.72
4	0.717949	0.726401	0.740903	0.739539	0.726423	0.730096	0.721131	0.72

```
sns.displot(time.mean())
```

```
<seaborn.axisgrid.FacetGrid at 0x7f366d9d9b50>
```




```
time.mean().mean()
```

```
0.12643905205106767
```