UNIVERSITY OF OKLAHOMA
GRADUATE COLLEGE

KIDNEY OCT 3D IMAGES CLASSIFICATION USING MACHINE LEARNING

A THESIS
SUBMITTED TO THE GRADUATE FACULTY
in partial fulfillment of the requirements for the
degree of
Master of Science

By

SINARO LY
Norman, Oklahoma
2023

KIDNEY OCT 3D IMAGES CLASSIFICATION USING MACHINE LEARNING

A THESIS APPROVED FOR THE
SCHOOL OF COMPUTER SCIENCE

BY THE COMMITTEE CONSISTING OF

Dr.Chongle Pan, Chair

Dr.Dimitrios Diochnos

Dr. Qi Cheng

Acknowledgments

I would like to thank Dr Pan who gave me a lot of guidance for my research. I would also like to thank Dr Badré who also helped me a lot during my research by answering a lot of questions I had when I began implementing my program. I would like tom thank Parker Brandt for I also wanted to thank the Pan Lab and Dr Tang's lab for helping me during the weekly meetings we hold. Finally, I would also like to thank my committee members for their supervision for this Master Thesis.

Table of Contents

chapterAcknowledgmentsiv

List Of Figures **vi**

Abstract **vii**

1 Introduction **1**

2 Literature Review **3**
2.1 Percutaneous nephrostomy review 3
2.2 3D imaging OCT review 4
2.3 3D CNN review 4

3 Methods **7**
3.1 Data presentation 7
3.2 Data pre-processing 9
3.2.1 Image cropping 9
3.2.2 Data augmentation with image rotation 13
3.3 Cross-validation presentation and implementation 14
3.4 Models implementation 17
3.4.1 3D CNN model 17
3.4.2 2D ResNet50 angular cuts model 22

4 Results and discussion **23**
4.1 3D CNN model results 23
4.2 2D ResNet50 angular cuts model results 26
4.3 Execution time performance 27

5 Conclusion **28**

List Of Figures

3.1 Representation of 3D image 8
3.2 Planes of 3D image 8
3.3 Representation of expected cropped 3D image 9
3.4 Starting points of uncropped images 10
3.5 Process to find the starting index 11
3.6 Image that has been too cropped 12
3.7 Image correctly cropped 12
3.8 Performing a 40 degrees rotation 14
3.9 Cross-validation illustration 15
3.10 The different layers of the 3D CNN model 21
3.11 New 2D cuts model illustration 22

4.1 First results of cross-validation 24
4.2 2D and 3D model comparison (Wang et al. (2021)) 25
4.3 2D, 3D and 2D 10 angular cuts models comparison 26
4.4 2D, 3D and 2D 10 and 100 angular cuts models comparison with p-values relative to 2D model 27

Abstract

The goal of this research is to improve percutaneous nephrostomy success rate. To do so, OCT imaging and image classification will be used to recognize which type of kidney tissue the needle is going through.

3D Optical coherence tomography (OCT) is a non-invasive imaging test that uses light waves to take cross-section pictures images.

A classification program for 3D OCT images by using a CNN model will be implemented. The images to classify are kidney images that have 3 different classes: Pelvis, Medulla and Cortex.

To do so, a data preprocessing was needed. The data preprocessing went through two big steps: cropping the 3D images to have smaller image volume and rotating the images to get a data enrichment.

After that data preprocessing, the next step is to build a model that can achieve a better accuracy than 2D models that were used previously.

After implementation and running the model through the dataset, using 3D images yield better accuracy than 2D basic cuts.

Chapter 1

Introduction

Image classification is a big topic in the industry, it has multiple and wide applications such as autonomous driving, medical diagnosis, object detection, AI development... (Sansoni et al. (2009))

Getting a high accuracy is crucial in this context. With the increasing computation capacity, unlocking new possibilities. Thus new models are being developed and tested (Sansoni et al. (2009)).

Percutaneous nephrostomy (PCN) is a medical procedure used to treat kidney conditions by inserting a needle into the kidney. However, that process is difficult and involves a lot of injury and complication risks. Indeed, there is a 18% failure rate. Thus, different imaging techniques were tested out to decrease that rate but as the different tissue types all look very similar and indistinguishable with human eyes (Efesoy et al. (2018)).

A potential solution would be to use Optical coherence tomography (OCT) to get an image of the subsurface tissue by inserting a probe with a depth of several millimeters. This technique presents several advantages (Fercher et al. (2003)):

- Better resolution.
- Non invasive medical imaging method.
- Real time images can be obtained.

However, the difficulty of recognizing the different types of tissue with naked eyes still remain. Machine learning can address that issue by training a model to spot the subtle differences and recognize the different tissue types (Schmidt-Erfurth et al. (2018)).

By combining that OCT method and a program that would recognize the tissue type, that would greatly improve PCN by reducing the risks.

We currently have 3D kidney images provided by the Biomedical Engineering department at OU with different 3 associated classes corresponding to the 3 different kidney tissue types for supervised learning. A classification program with 2D cuts have already been implemented with that dataset with an average accuracy of 85.63% (Wang et al. (2021)).

Goal: improve the accuracy rate over other models by developing a model for 3D images, especially for the current dataset we have in hand.

This study is divided into different steps: prepare the data, implement 3D models and make some comparison with other models.

The hypothesis is the fact that taking the whole 3D image instead of 2D cuts will improve the classification accuracy.

Chapter 2

Literature Review

2.1 Percutaneous nephrostomy review

The first step for percutaneous nephrolithotomy (PCNL) surgery and other therapeutic intervention requiring trans-urethral access of surgical to the urological system is difficult. Despite being a common urological procedure, it is still technically challenging to insert the PCN needle correctly in the right place without unnecessarily damaging the kidney tissue. During PCN, a needle penetrates the cortex and medulla of the kidney to reach the renal pelvis. Conventional imaging modalities have been used in PCN puncture. Ultrasound technique, as a commonly used medical diagnostic imaging method, has been utilized in PCN surgery for decades (Efesoy et al. (2018)).

Fluoroscopy and computed tomography (CT) are also used for PCN guidance (Zegel et al. (1981)). However, due to the limited spatial resolution, these standard imaging methods have been proven to be inadequate for accurately locating the needle tip position: the failure rate of PCN needle placement is 18%, especially in non-dilated systems or for urolithiasis. Failure of inserting the needle into the targeted location in the kidney through a suitable route might result in rupture of renal blood vessels by needle penetrations which can cause bleeding or even more severe complications (Pabon-Ramos et al. (2016)). Moreover, fluoroscopy has no soft tissue contrast so the different types of tissue cannot be identified, such as blood vessels, which are important to avoid during the needle insertion process. Temporary bleeding after PCN placement

occurs in about 95% of cases. Retro-peritoneal hematomas have been found in 13% of the time (Pabon-Ramos et al. (2016)). Furthermore, injuries related to the PCN can lead to infectious complications such as fever or sepsis, thoracic complications like pneumothorax and hydrothorax, and other complications.

2.2 3D imaging OCT review

Optical coherence tomography (OCT) is a non-invasive biomedical imaging process allowing to get image subsurface tissue by inserting a probe with a depth of several millimeters into the kidney. After getting and processing the coherent infrared lights backscattered from the reference arm and sample arm, OCT can provide 3D images with high axial resolution (about 10μm), which is 10 to 100 times higher than conventional medical imaging methods such as CT and MRI for example (Li et al. (2009)). Thanks to the high speed of laser scanning and data processing, the 3D images of the detected sample formed by numerous cross-sectional images can be obtained in real time.

In the case of this study with kidney image scans, there are three different tissue types to classify: cortex, medulla and pelvis.

2.3 3D CNN review

Convolutional neural networks is very popular for image classification tasks, especially in the medical field for computer-aided diagnosis to detect disease and cancer. 3D CNN use filters (also called kernels) to extract features from local patches of the input data (Kamnitsas et al. (2017)).

A filter is a small, learnable matrix used for convolutional operations.

Filters serve several important purposes:

- Feature Extraction: Filters are primarily used to extract features from input data. In CNNs, filters convolve over the input data, capturing local patterns or features. These patterns can include edges, textures, shapes, or more complex structures, depending on the depth and training of the network.

- Dimension Reduction: As filters convolve over the input, they reduce the spatial dimensions of the data. For example, in image processing, a filter might reduce a large image into smaller feature maps that highlight specific aspects of the input. This reduction in dimensionality is important for subsequent layers in the network and helps in focusing on relevant information.

- Hierarchical Feature Learning: Filters are stacked in multiple layers of a CNN, creating a hierarchy of features. Lower layers capture simple, low-level features, like edges and corners, while deeper layers learn more abstract and complex features by combining information from lower-level features. This hierarchical feature learning is essential for the network's ability to understand complex data.

- Learnable Parameters: Filters are learnable parameters of the neural network. During training, the network adjusts the values within these filters using optimization algorithms like gradient descent. This process allows the network to adapt and discover the most informative features for a given task.

- Spatial Locality: Filters capture spatial locality, meaning they focus on local regions of the input. This property enables the network to recognize patterns regardless of their position within the input data. In images, for instance, a filter can identify the same feature (e.g., an edge) regardless of where it appears.

3D CNN is also pretty similar to 2D CNN as they include convolutional layers, activation functions (like ReLU), pooling layers (such as max pooling), and fully connected layers.

The activation layer is used to introduce non linearity to the model by applying specific functions. In the case of ReLu, it takes x and output the max between x and 0.

A pooling layer is used to reduce the dimension of the input. For example, with the max pooling, max of the input will be processed.

However, 3D CNN can be computationally more taxing than 2D CNN due to the increase of dimensions.

Chapter 3

Methods

3.1 Data presentation

The dataset is a collection of 3D images taken from pig kidneys provided by Dr Tang's biomedical lab. Dr Qinggong Tang is an Assistant Professor of Biomedical Engineering specialized in optical imaging techniques. The dataset contains 3 different classes with 30 images for each class, corresponding to cortex, medulla and pelvis. There are 10 folders of 3D images corresponding to 10 different kidneys and each folder contains those 3 different classes. The goal is to determine which class a 3D image belongs to. To do so, different learning models will be used.

The 3D images are in form of files in TIFF format. Each image contains a 3D cylinder and is around 11 Mb. Those 3D images contain a lot of empty space and there is just a few of them. Thus, to address these issues, some data pre-processing and data augmentation will be performed.

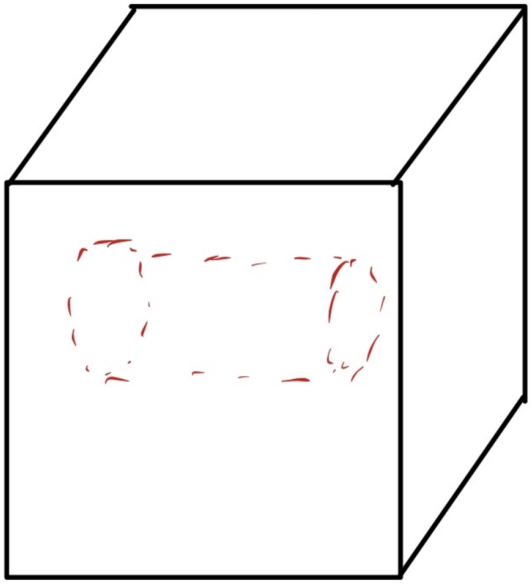

Figure 3.1: Representation of 3D image

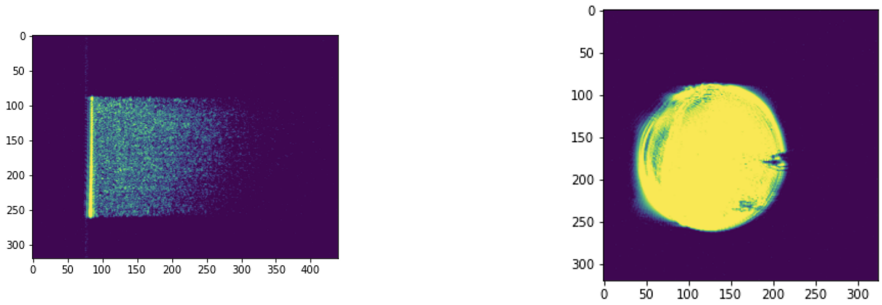

Figure 3.2: Planes of 3D image

To do so, some cropping has been be performed on those big images to have smaller images without losing any information, allowing the program to run much faster. Python with Tensorflow will be used for this project. After the 3D image is read, it is represented as a 3D numpy array with each cell representing the intensity at the (x, y, z) coordinates, these indexes represent position on the different planes as it is shown on figure 3.2.

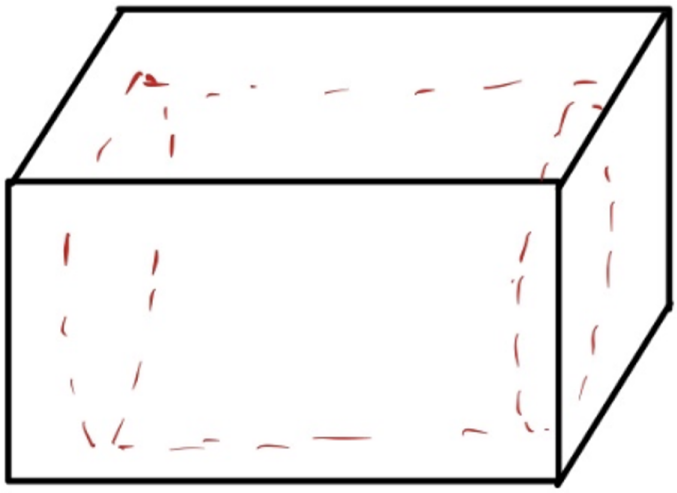

Figure 3.3: Representation of expected cropped 3D image

3.2 Data pre-processing

The data pre-processing is divided into 2 big steps:

- Cropping each image to get images that doesn't contain useless information, using up resources pointlessly.
- Perform a data augmentation on those cropped images by rotating the cylinder by different angles.

3.2.1 Image cropping

The 3D images are stored on the supercomputer as TIFF files and need to be read on the program. To do so, the library Scikit-image will be used to get Numpy 3D arrays after reading the image file. After getting the Numpy array, the cropping will be perfomed automatically on it and then it will be reconverted into a smaller TIFF file to be easily reusable.

To crop the image, the size of the 3D numpy array will be reduced, leaving out the parts that are not useful to keep only the actual kidney image. The cylinder's dimensions are known, thus, it will be faster and more precise to use that information: where the cylinder starts from different planes to be able to perform the cropping by using the cylinder's diameter and the depth: That starting points are the borders of the cylinder shown on figure 3.4

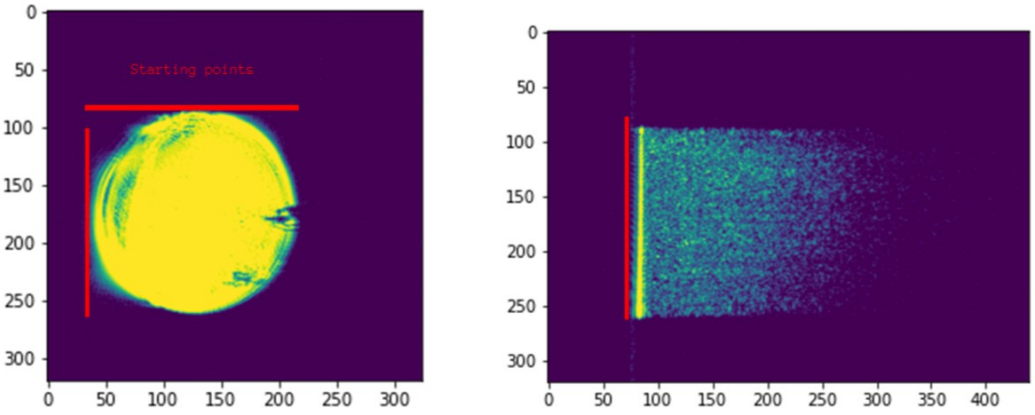

Figure 3.4: Starting points of uncropped images

As the images are in form of a Numpy array, cropping the image would correspond to extracting a subarray of the Numpy array by using the array indexes. After getting the cropped image, the data enrichment with the image rotation would be performed as well, as the cropped image would be already loaded, it would save some time.

What's left to do is to find the correct index to know from where the cropping should be done. To do so, multiple ideas, came up to find the starting index, the main idea was to use the sum of the intensity on a plane: a big percentage of the intensity is in the cylinder. By taking advantage of that property, the starting index can be deduced.

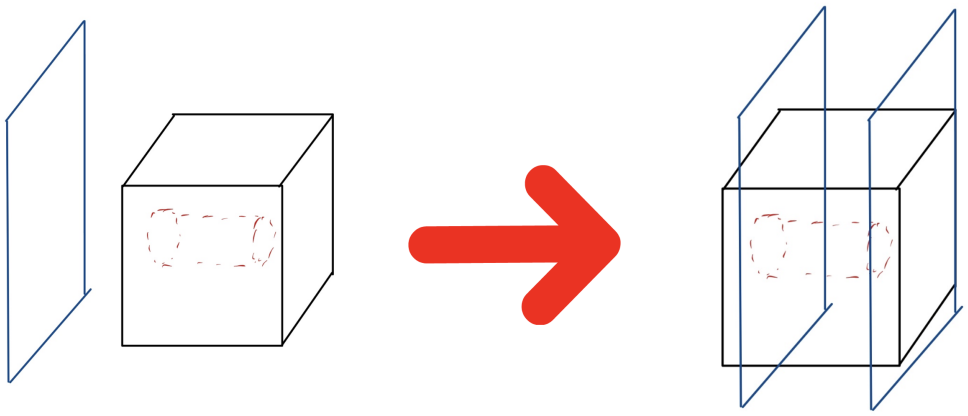

Figure 3.5: Process to find the starting index

The idea is to move the left blue plane at each iteration until it hits the cylinder. That blue plane will cut out the part of the cube before it, creating a new cropped 3D Numpy array. To know when to stop, the ratio between the sum of the intensity of the image that has been cropped after that plane and the full image will be processed. If the ratio reaches a determined threshold, it means that the starting point is here and the starting index is now known. The same is done to the other dimensions to crop entirely the image.

However, after processing with that method, the cut went a bit too far and the chosen diameter was too small (Figure 3.6), leading to some information loss.

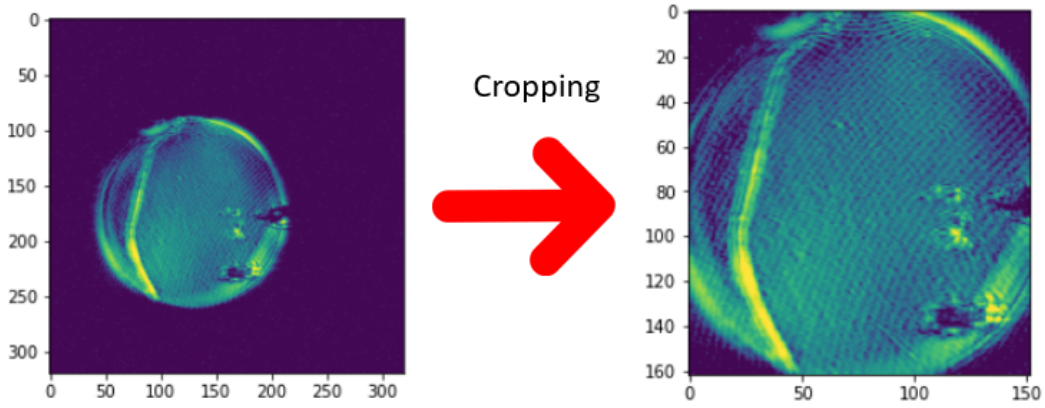

Figure 3.6: Image that has been too cropped

To solve the issue, the starting index has been decreased, to have a bit more margin around the cylinder and the diameter has been increased (Figure 3.7).

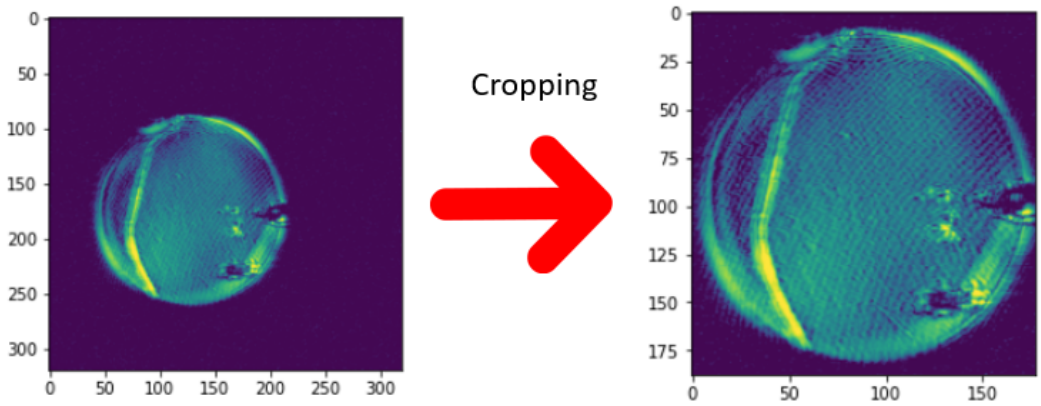

Figure 3.7: Image correctly cropped

Algorithm 1 Image Cropping

```
function ImageCropping(image, depth, diameter, threshold)
    ratio ← 1
    full_sum ← sum(image)
    cropping_indexes ← []
    for each image_dimension do
        cropping_index ← 0
        while ratio > threshold do
            cropped_image ← image[image_dimension, cropping_index]
            new_sum ← sum(cropped_image)
            ratio ← new_sum/full_sum
            cropping_index ← cropping_index + 1
        end while
        cropping_indexes.append(cropping_index)
    end for
    result ← image[cropping_indexes[1] + diameter,
                   cropping_indexes[2] + depth,
                   cropping_indexes[3] + diameter]
    return result
end function
```

With that function implemented, the cropping can be automated and applied to all images. The image original dimensions were 325x325x450. After cropping, the image dimensions are now 175x200x250.

After getting a satisfying result, rotations can be performed on the cylinder. With Parker Brandt's collaboration who was working on the rotation algorithm at the same time, it was possible to get satisfying results pretty early on. Parker Brandt is another Master Student who was part of the lab at that time for his Master Program.

3.2.2 Data augmentation with image rotation

Rotating the 3D image could be done pretty easily as a 3D image is represented by a 3D numerical Numpy Array. That Numpy Array just needs to be multiplied by a rotation matrix to get a 3D rotated image. A function has been implemented to take as input the Numpy Array directly after it has been cropped and the number of rotations

needed. Depending on the number of rotations needed, the rotation matrix will change to adapt to that number of rotations. A loop is executed to rotate the Numpy Array and save that to a TIFF file that is significantly smaller than the original image (4MB).

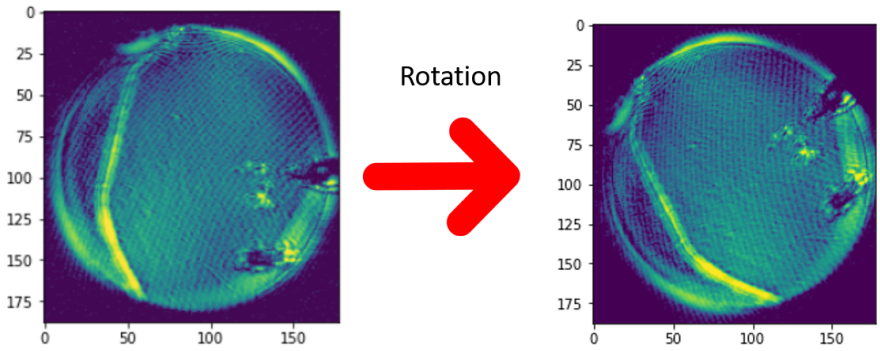

Figure 3.8: Performing a 40 degrees rotation

The function used would take the number of rotations wanted as an argument and the rotation angle would depend on that number of rotations as it would be used to divide by 360 degrees.

3.3 Cross-validation presentation and implementation

A 10 fold cross-validation will be performed to find good hyper-parameters for the model that will be used. In this case, it is the number of epochs needed to get satisfying results.

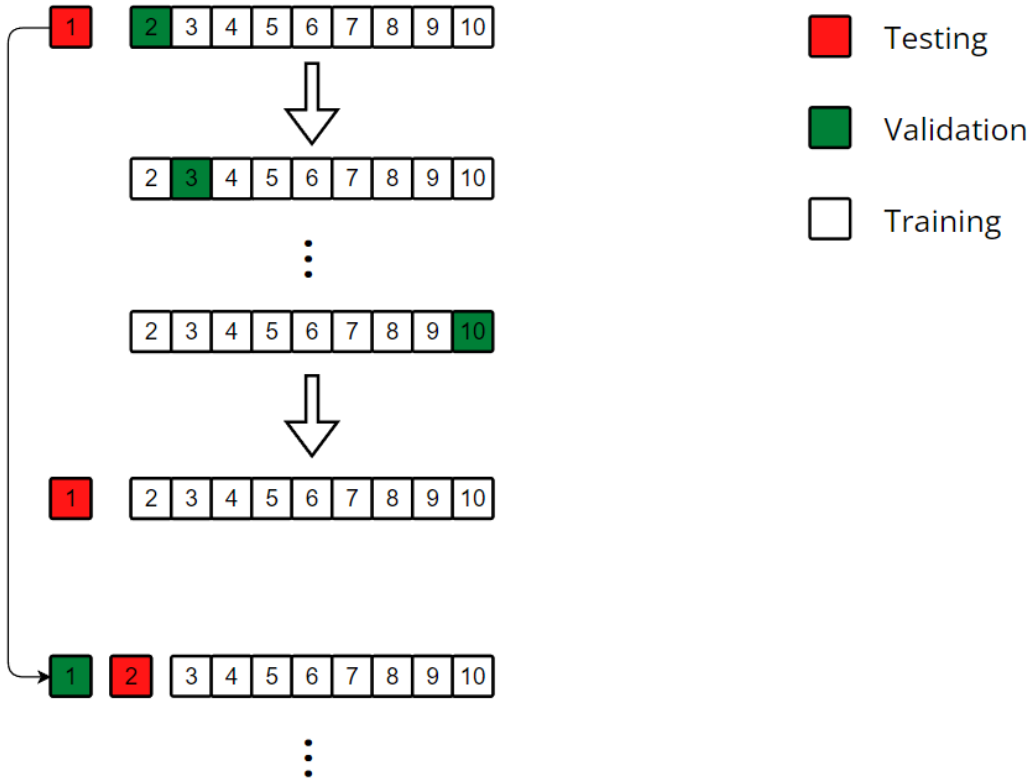

Figure 3.9: Cross-validation illustration

A cross-validation is divided into 2 big steps: the validation process to find fitting hyper-parameters and the final evaluation. The idea is to have 2 for loops nested together. The outer loop is made to separate the testing folder from the others that will be used in the inner loop and do the evaluation after the inner loop, using that test folder. The inner loop will go through validation process with a different folder as validation set at each iteration to determine the best epoch to stop for the evaluation with the test folder.

Algorithm 2 Cross Validation

```
1: function CrossValidation(test_index)
2:     Initialize best_epoch_fold ← []
3:     Initialize val_scores ← []
4:     for validation_index ← 0 to 9 do
5:         if validation_index = test_index then
6:             Continue
7:         end if
8:         validation_data ← get_data(validation_raw[0], validation_raw[1])
9:         training_data ← get_data(training_raw[0], training_raw[1])
10:        validation_kidney_number ← get_kidney_number(validation_raw, data_path)
11:        test_kidney_number ← get_kidney_number(test_raw[0][0], data_path)
12:        Clear Session
13:        model ← get_model()
14:        model.fit(training_data, validation_data = validation_data, epochs =
    epochs, callbacks = callbacks_list)
15:        model.load_weights(filepath)
16:        best_epoch_fold.append(get_best_epoch(csv_file1))
17:        val_scores.append(model.evaluate(validation_data))
18:    end for
19:    Return {'best_avg_epoch': int(mean(array(best_epoch_fold))),
    'best_avg_accuracy': mean(array(val_scores)), 'std_accuracy':
    std(array(val_scores))}
20: end function
```

The performance is always recorded and logged into files so nothing is lost, especially with the supercomputer which would crash quite often. A load function was also implemented to resume where the program left, using those log files as well. These log files record the epoch, the training accuracy, the loss, the validation accuracy and the validation loss. A log file is created for each different validation and test folder.

By using that log file, a function will read the log files for the same test folder, for each file find the epoch with high accuracy and low deviation around. The best epoch returned will be the average of the best epoch of each file.

After getting the best epoch, that number will be used as a parameter for the training and final testing with the test folder. The result will also be saved in a file for results analysis and comparison with other models.

3.4 Models implementation

3.4.1 3D CNN model

CNNs are used in machine learning and data mining that tries to mimic human brain with layers of interconnected nodes and are widely used for image classification which corresponds to the goal here. The implemented 3D CNN is organized into several types of layers:

- **Convolutional Layers:**
    - Convolutional layers are fundamental to CNNs. They apply learnable filters to the input data, allowing the network to learn features and patterns in the data.

- Convolutional layers use convolution operations to detect local patterns like edges, textures, and simple features.
- These layers can have multiple filters, each learning to recognize different features.
- Convolutional layers are often followed by activation functions (e.g., ReLU), in the case of this model, ReLU has been used.

- **Pooling Layers:**
  - Pooling layers are used to reduce the spatial dimensions of the feature maps produced by convolutional layers.
  - Max-pooling is used for this model to the feature map's size.
  - Pooling helps to make the network translation-invariant, reduce computational complexity, and control overfitting.

- **Fully Connected Layers:**
  - Fully connected layers are similar to layers in traditional feedforward neural networks.
  - These layers connect every neuron to every neuron in the previous and next layers, effectively flattening the feature maps.
  - FC layers are typically placed at the end of the network to make final predictions or classifications.

- **Normalization Layers:**
  - Normalization layers, like Batch Normalization, are used to standardize the inputs to a layer. This helps stabilize training, improve convergence, and make the network more robust.

- Normalization is applied to this model.

- **Dropout Layers:**

  - Dropout layers randomly deactivate a fraction of neurons during training, which helps to prevent overfitting.
  - Dropout is a regularization technique, and dropout layers are typically added after fully connected layers.
  - Dropout is a regularization technique, and dropout layers are typically added after fully connected layers.
  - In the case of this model,

Activation functions define the output of the neuron given the weighted sum of its inputs. Common activation functions include the sigmoid function, Rectified Linear Unit(ReLU). In this case, ReLU has been used.

The model used is a 3D CNN with different layers of filters, average pooling and batch normalization with Adam optimizer.

An optimizer is used to compile the model after all the layers have been determined. It is used for different purposes:

- Parameter Optimization: Adjust the parameters (weights and biases) of the 3D CNN model during the training process to keep the best parameters.
- Convergence to a Minimum: Optimizers help the model converge to a local minimum of the loss function.
- Stability and Generalization: An effective optimizer can help ensure that the training process is stable and that the model generalizes well to unseen data.

In this case, Adam and SGD were used and some comparison is made between those two models.

The layers and parameters are refined through tests by running the model and checking preliminary results and changing parameters one by one. After getting results that seem satisfying, the cross-validation could run to benchmark the model.

After implementing a model that seemed to get good results and ran a cross-validation, results are logged into files. With those raw data, a function plots them and some analysis is done to evaluate the reliability of the model.

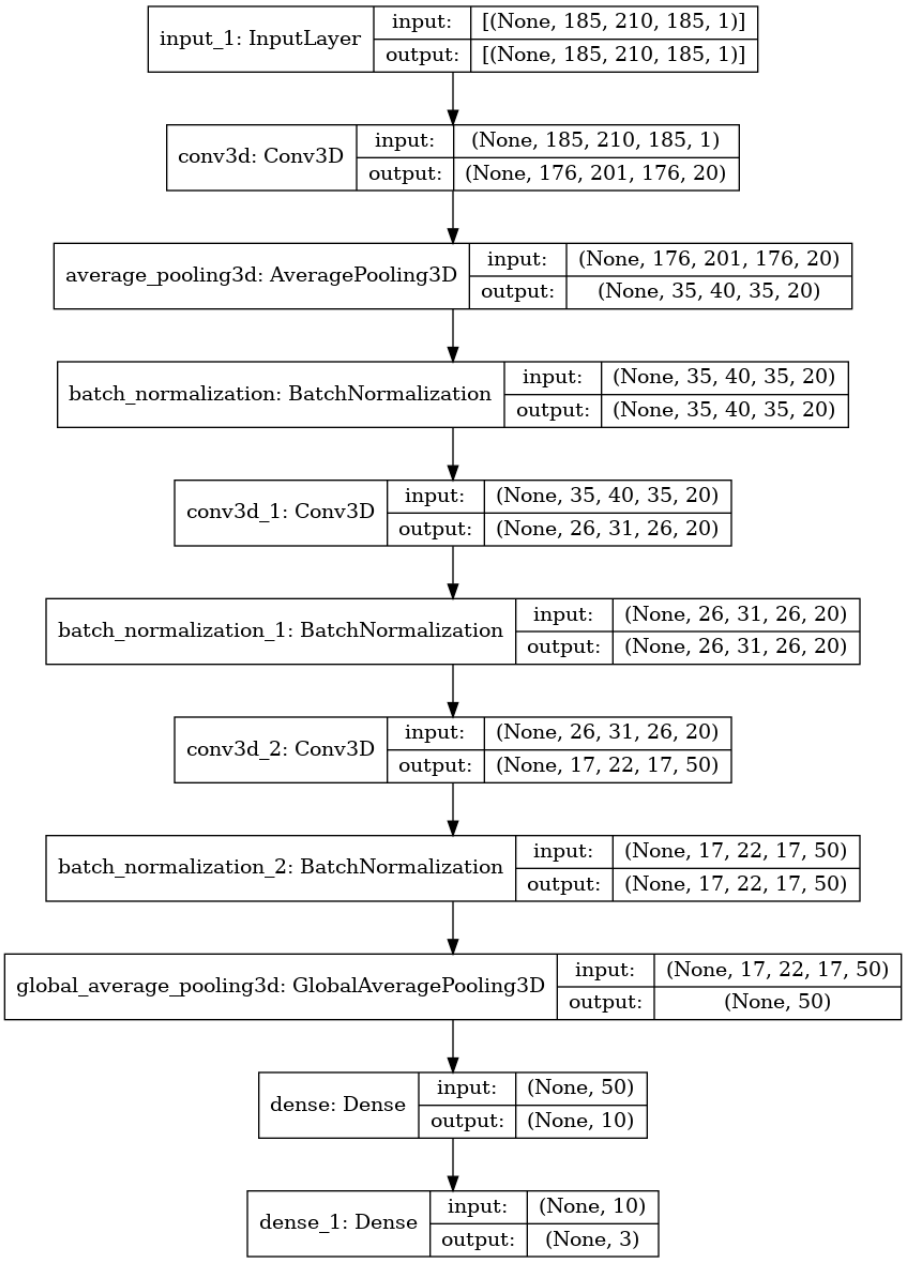

Figure 3.10: The different layers of the 3D CNN model

3.4.2 2D ResNet50 angular cuts model

Preliminary results have been obtained with a 2D ResNet model: the same 3D images were used but with cuts at the same angle were taken from the original 3D image for training and predictions.

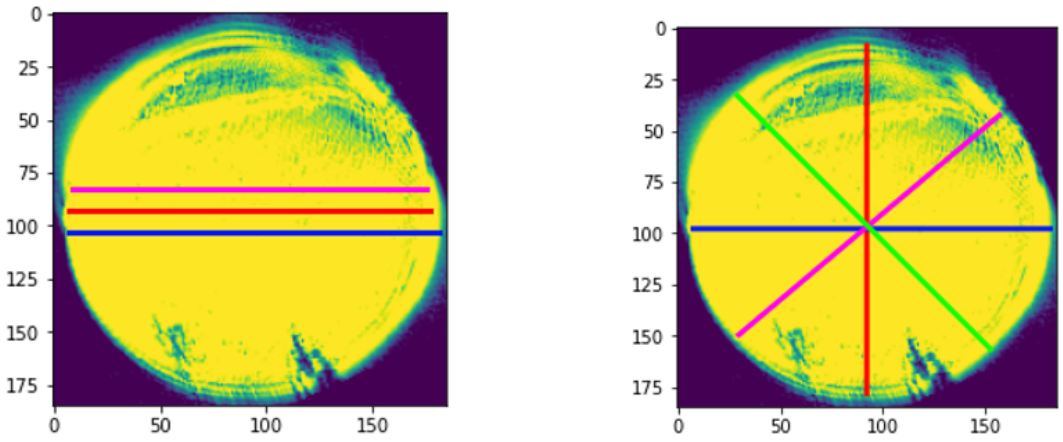

Figure 3.11: New 2D cuts model illustration

The next step would be implementing a 2D model but with different angles (on the right of figure 3.11). That could result in an accuracy improvement over the old 2D model. The number of cuts would also be customizable for comparison and analysis purpose.

Each 3D image would be processed to a fixed number n of 2D images corresponding to the n angular cuts made. Those 2D images will then be trained with the same ResNet50 model. For the predictions, a prediction is made for each cut of the 3D image and a max pooling is applied to get a single prediction corresponding to the 3D image class prediction: The most occurring prediction within those n 2D cuts will be kept for the 3D image.

Chapter 4

Results and discussion

4.1 3D CNN model results

This graph below represents an iteration of cross-validation, with kidney 1 as test, 2 as validation and 3, 4, 5, 6, 7, 8, 9, 10 as training. Here, the validation accuracy is very unstable. Thus, some changes needed to be made in the layers. After trying out different ideas, a stable model eventually came out and could be used afterwards.

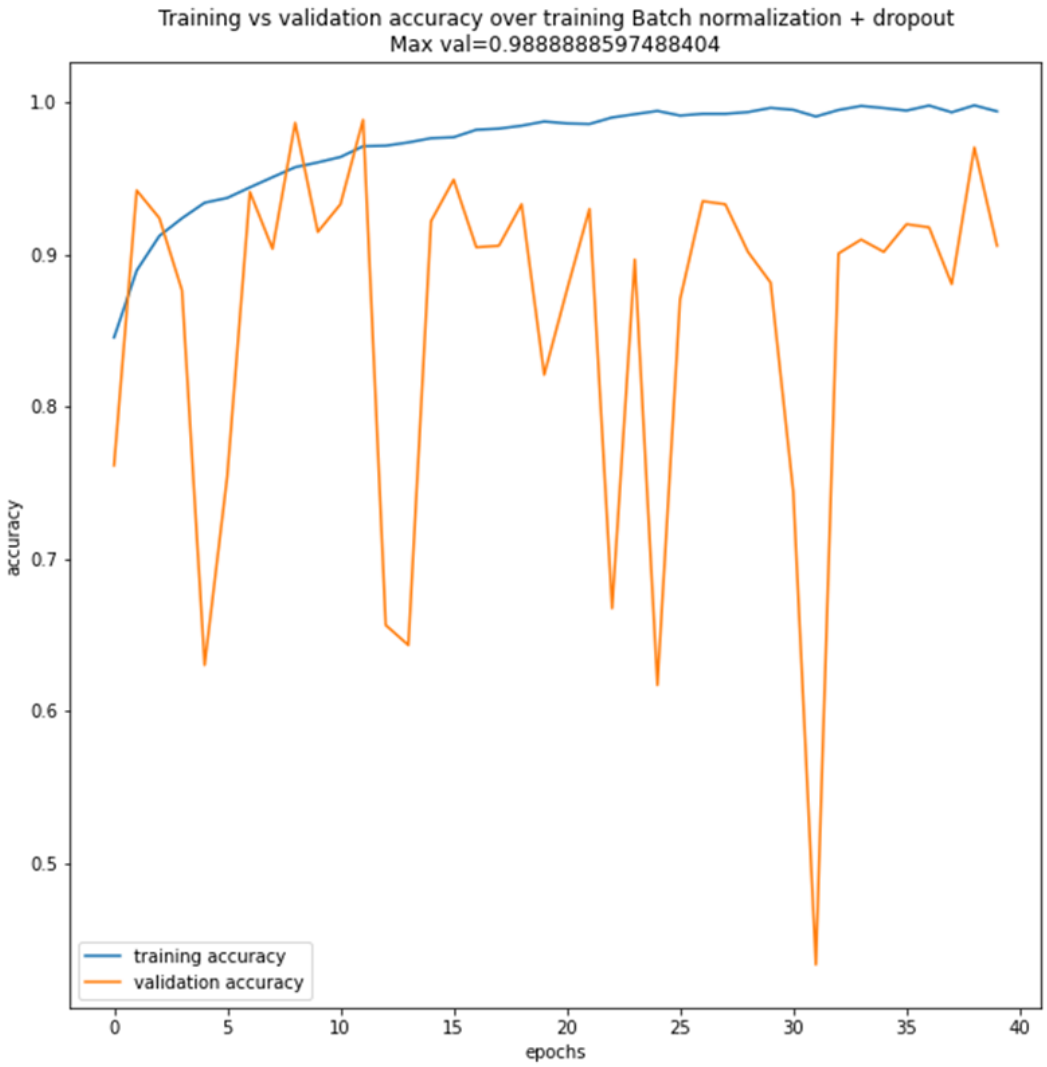

Figure 4.1: First results of cross-validation

Here, the validation accuracy is more stable. Multiple factors can explain that stability gap:

Testing folds	2D model	3D CNN model SGD
	Accuracy	Accuracy
Kidney 1	79.00%	86.72%
Kidney 2	90.00%	93.33%
Kidney 3	76.80%	81.23%
Kidney 4	95.90%	94.32%
Kidney 5	95.60%	99.49%
Kidney 6	68.90%	87.56%
Kidney 7	86.40%	88.49%
Kidney 8	92.30%	90.67%
Kidney 9	94.80%	93.73%
Kidney 10	76.60%	90.18%
Average	85.63%	90.57%
Standard deviation	0.096407757	0.050203891

Figure 4.2: 2D and 3D model comparison (Wang et al. (2021))

On average, the 3D model is performing better than the 2D model. There are multiple possible reasons for that outcome:

- The model is very sensitive to learning rate and Adam adapts the learning rate for each parameter, which can be beneficial in some cases. However, if the learning rate is not appropriately tuned, Adam can converge too quickly or overshoot the optimal solution. SGD uses a fixed learning rate, which is more adapted for this application.

- Momentum in the context of optimization is a technique used to accelerate the convergence of gradient-based optimization algorithms like stochastic gradient descent (SGD) and to help these algorithms navigate through flat or noisy regions of the optimization landscape. Adam includes a momentum-like term and adaptive learning rates, which can act as a form of implicit regularization. That momentum can also cause some instability in the validation accuracy.

To have a better understanding of the reasons, the implementation for the angular 2D cuts model was necessary.

4.2 2D ResNet50 angular cuts model results

Testing folds	2D model	3D CNN model SGD	2D 10 angular cuts model
	Accuracy	Accuracy	Accuracy
Kidney 1	79.00%	86.72%	68.80%
Kidney 2	90.00%	93.33%	91.13%
Kidney 3	76.80%	81.23%	78.98%
Kidney 4	95.90%	94.32%	91.25%
Kidney 5	95.60%	99.49%	94.43%
Kidney 6	68.90%	87.56%	96.23%
Kidney 7	86.40%	88.49%	85.30%
Kidney 8	92.30%	90.67%	88.48%
Kidney 9	94.80%	93.73%	97.67%
Kidney 10	76.60%	90.18%	89.88%
Average	85.63%	90.57%	88.22%
Standard deviation	0.096407757	0.050203891	0.087148599

Figure 4.3: 2D, 3D and 2D 10 angular cuts models comparison

Here, these 10 angular cuts yield better result than the basic 2D cuts model. Furthermore, the results tend to get close to the 3D CNN model without outperforming it. The data augmentation might be the factor for better results, to confirm that supposition, 100 cuts will now be tested to compare the accuracy and the results will give insights too.

Testing folds	2D model	3D CNN model SGD	2D 10 angular cuts model	2D 100 angular cuts model
	Accuracy	Accuracy	Accuracy	Accuracy
Kidney 1	79.00%	86.72%	68.80%	67.80%
Kidney 2	90.00%	93.33%	91.13%	91.11%
Kidney 3	76.80%	81.23%	78.98%	72.22%
Kidney 4	95.90%	94.32%	91.25%	92.25%
Kidney 5	95.60%	99.49%	94.43%	94.43%
Kidney 6	68.90%	87.56%	96.23%	89.73%
Kidney 7	86.40%	88.49%	85.30%	91.30%
Kidney 8	92.30%	90.67%	88.48%	89.98%
Kidney 9	94.80%	93.73%	97.67%	98.67%
Kidney 10	76.60%	90.18%	89.88%	93.88%
Average	85.63%	90.57%	88.22%	88.14%
Standard deviation	0.096407757	0.050203891	0.087148599	0.099564204
p-value with 2D model		0.01377673	0.20714492	0.19762158

Figure 4.4: 2D, 3D and 2D 10 and 100 angular cuts models comparison with p-values relative to 2D model

The 100 2D cuts model doesn't yield better result than the basic 10 model, it even performs worse in this case. Thus, it is a model limitation, meaning that the accuracy can't really go further with the 2D ResNet model.

The p-values relative to the base 2D model have also been processed. A p-value below 0.05 means that the improvement is statistically significant. Here, only the 3D CNN model shows a significant improvement, the different 2D cuts models, yield some improvement but not significant enough statistically, adding more proof to the 2D model's limitations.

4.3 Execution time performance

The 3D CNN model takes around 11 hours to be trained with a cross-validation process and a prediction takes on average 1.234 seconds to be done.

The 2D 10 cuts model takes around 2 hours to be trained with a cross-validation process and a prediction takes on average 1.194 seconds to be done.

The 2D 100 cuts model takes around 5 hours to be trained with a cross-validation process and a prediction takes on average 3.328 seconds to be done.

Chapter 5

Conclusion

For this study, a data preprocessing pipeline for 3D images, a cross-validation, a 3D CNN model that performs better than the 2D model and a 2D angular cuts model have been implemented.

In the process of trying to implement a 3D model for supervised learning to improve accuracy, we got better results than 2D model which seems encouraging.

To do so, we also had to go through data pre-processing and data enrichment. That contributed to better efficiency and performance.

With the better accuracy results with the 3D CNN model and the 2D angular cuts model, the hypothesis of using 3D images for classification to get better accuracy is confirmed. The 3D images hold more information than the 2D images. Furthermore, the 2D model also seem to have limitations regarding to the accuracy it has compared to the 3D model.

What remains to be done is to benchmark the time performance in different situations and environment to check if such a classification program can be used to receive real time images and send predictions in real time as well if this model can really be used for real life situation.

If such a program is usable during a kidney treatment operation, it would greatly decrease the risks of PCN failure and complications for the patients.

Reference List

Efesoy, O., B. Saylam, M. Bozlu, S. Çayan, and E. Akbay, 2018: The results of ultrasound-guided percutaneous nephrostomy tube placement for obstructive uropathy: A single-centre 10-year experience. *Turkish Journal of Urology*, **44 (4)**, 329.

Fercher, A. F., W. Drexler, C. K. Hitzenberger, and T. Lasser, 2003: Optical coherence tomography-principles and applications. *Reports on progress in physics*, **66 (2)**, 239.

Kamnitsas, K., C. Ledig, V. F. Newcombe, J. P. Simpson, A. D. Kane, D. K. Menon, D. Rueckert, and B. Glocker, 2017: Efficient multi-scale 3d cnn with fully connected crf for accurate brain lesion segmentation. *Medical image analysis*, **36**, 61–78.

Li, Q., and Coauthors, 2009: Automated quantification of microstructural dimensions of the human kidney using optical coherence tomography (oct). *Optics express*, **17 (18)**, 16 000–16 016.

Pabon-Ramos, W. M., and Coauthors, 2016: Quality improvement guidelines for percutaneous nephrostomy. *J Vasc Interv Radiol*, **27 (3)**, 410–4.

Sansoni, G., M. Trebeschi, and F. Docchio, 2009: State-of-the-art and applications of 3d imaging sensors in industry, cultural heritage, medicine, and criminal investigation. *Sensors*, **9 (1)**, 568–601.

Schmidt-Erfurth, U., A. Sadeghipour, B. S. Gerendas, S. M. Waldstein, and H. Bogunović, 2018: Artificial intelligence in retina. *Progress in retinal and eye research*, **67**, 1–29.

Wang, C., and Coauthors, 2021: Deep-learning-aided forward optical coherence tomography endoscope for percutaneous nephrostomy guidance. *Biomedical optics express*, **12 (4)**, 2404–2418.

Zegel, H., and Coauthors, 1981: Percutaneous nephrostomy: comparison of sonographic and fluoroscopic guidance. *American Journal of Roentgenology*, **137 (5)**, 925–927.