https://www.overleaf.com/project/5f81b795d430cc0001f50400

BABEŞ BOLYAI UNIVERSITY, CLUJ NAPOCA, ROMÂNIA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

AI assistant for a prostate medicine student

– ITSG report –

Team members
Bocsa Mirela Alexandra, Software Engineering, 258, bmis2033@scs.ubbcluj.ro
Roman Adrian, Software Engineering, 258, rais2087@scs.ubbcluj.ro
Vaida Andrei Lucian, Software Engineering, 258, vais2096@scs.ubbcluj.ro
Vrincean Alexandru, Software Engineering, 258, vais2100@scs.ubbcluj.ro

2020-2021

Abstract

Prostate cancer is the most common cancer among men and is the second leading cause of cancer death in men after lung cancer.In the last few years, In the medical system, the amount of data that needs to be analyzed has increased dramatically, which means that much more effort is needed from doctors and experts. Therefore, an intelligent algorithm capable of detecting the cancer prostate is needed.

In order to solve this problem, we proposed two UNET models that will learn to detect the possible cancer zone from a medical image. Both models follow the classical UNET architecture, what is different are the number of features per layer.

The dataset contains about 900 slices from 30 patients. Because of this small number of input data and due the fact that the images are very dark, with a lot of noise and just a small part of them represents the prostate and even a smaller one describes the cancer area, the results are quite disappointing. The UNets didn't manage to learn and identify, neither the prostate, nor the cancer area.

Even if the results are not very satisfying and the U-Net models are not yet able to detect the prostate cancer zone very accurately, they still prove that a U-Net can be used for prostate cancer segmentation. With more training time and more competent hardware, if the number of epochs and the batch size is increased, as well as the original size of the input images is kept, and a better preprocessing of the input data, the result may improve and the models will be able to accurately detect the prostate cancer zone.

Contents

List of Tables

List of Figures

Team contribution:

Bocsa Mirela:

- Intelligent algorithm research, dataset acquisition and dataset processing
- Prepare the input for the neural network, implement the 2 UNET models, train the algorithm, play with the parameters
- Analyze result
- Update documentation

Roman Adrian:

- Server and client implementation
- Dataset researching
- Update documentation

Vaida Andrei Lucian:

- Application basecode, client-server architecture and implementation
- Dataset researching and acquisition, training attempts
- Update documentation and video presentation

Vrincean Alexandru:

- Server and client implementation
- Continuous feedback of teamwork

Chapter 1

Introduction

1.1 What? Why? How?

Prostate cancer is the most common cancer among men and is the second leading cause of cancer death in men after lung cancer. The condition usually develops very slowly and mainly affects older men. About 6 out of 10 cases are diagnosed in men over 65, the disease being rarely detected before 40 years. The average age at diagnosis is approximately 66 years. The choice of appropriate treatment depends on several factors, including the stage of the disease, the patient's age, possible previous treatment options and last but not least the patient's preferences.[2]

In the medical system, the people responsible for interpreting medical data are only medical experts. But, in the last few years, the amount of data that needs to be analyzed has increased dramatically, which means that much more effort is needed from doctors. Furthermore, the interpretation of medical data performed by a medical expert is due the level of expertise of the person that reads the data, to its subjectivity and the complexity of the image.

Therefore, we propose an application that will allow a student to upload a medical image of a prostate organ, visualize it and, with the help of an intelligent algorithm, tell him if that prostate is suspected of cancer or not. The application will display a new image with a frame surrounding the location of the possible tumor.

Machine learning algorithms come with a big help in medical data analysing since they are capable of interpreting very big images and providing an accurate and efficient diagnosis. Moreover, there will be no subjectivity involved and, unlike humans, the algorithm will not get tired, on the contrary, the more data received, the more accurate the result will be.[10]

1.2 Paper structure and original contribution(s)

The research presented in this paper advances the theory, design, and implementation of several particular models.

The main contribution of this report is to present an intelligent algorithm for solving the problem of prostate cancer detection from a medical image.

The second contribution of this report consists of building an intuitive, easy-to-use and user friendly software application. Our aim is to build an application that will help a medical student to identify and monitor cancer cells from a prostate medical image. This is done in a user friendly-manner, but with all the support of the accuracy of an intelligent algorithm.

The third contribution of this thesis consists of the new results obtained by our intelligent algorithm comparing them with the results obtained by other similar algorithms.

The present work contains xyz bibliographical references and is structured in five chapters as follows.

The second chapter/section describes the problem definition and the representation of our data.

The chapter 3 details the related work and state-of-the-art results of the methods utilised until now in order to solve the given problem.

The chapter 4 details our approach and offers more details about the algorithm we use to address the problem.

The chapter 5 introduces our application and its main functionalities from the user perspective and also describes the tools and technologies we used for building the application.

The chapter 6 explains the experimental methodology and the numerical results obtained with our approach and the state of art approaches. Our focus in this chapter is on the interpretation and the statistical validation of the results.

In the end, in the chapter 7 we summarize our conclusions and future work and also try to analyze the strengths and weaknesses of our application with the focus on what we can improve both in the algorithm and the application.

Chapter 2

Scientific Problem

2.1 Problem definition

The goal of this project is to help medicine students to study prostate cancer by providing them an application in which they can load an image an receive a result with the zone of the possible prostate tumor. Students can use the application to apply the theoretical notions learned in the courses.

In recent years, prostate cancer has become more common such that it is estimated that one in six men will develop prostate cancer in their lifetime. Although the prostate cancer is usually not the direct cause of death, the differentiation of clinically significant prostate cancer lesions from those low-grade lesions is critical. The state-of-the-art diagnostic method uses multiparameteric MR imaging with transrectal ultrasound-guided biopsy. This method is the most accurate imaging method for prostate cancer detection, but, it requires the expertise of experienced radiologists, and as such, there is inconsistency across readers of varying experience. In order to reduce the rate of occurrence of errors due to inattention or the medical judgment, subjectivity and level of expertise, an intelligent algorithm that can detect the prostate cancer is needed.[8]

The main advantage of an artificial intelligent algorithm is its ability to analyse and employ an enormous quantity of data, much more efficiently than possible for humans through classical statistical analyses. Moreover, the more data received, the more accurate the result will be. There will be no more subjectivity involved and less high level of prostate cancer expertise will be required.[10]

On the other hand, processing medical image is a challenge even for an intelligent algorithm. Most of the prostate image may be difficult to read an interpret, which means that large sets of data with labeled images are required for the algorithm to work properly. Machines are still having a hard time understanding medical images and using certain unclear images can be misleading for the AI, that can provide erroneous or implausible detections.

The following sections will describe the input and the output set.

2.2 Input

The input for the intelligent algorithm will be a set of medical images containing from 26 to 30 slices with the prostate MRI of a specific patient. These images may or may not represent a prostate with possible cancer zones. Each slice has 512x512 pixels.

2.3 Output

The output that the intelligent algorithm will provide will be a new medical image with the segmentation of the prostate and the possible prostate cancer zones.

Chapter 3

State of the art/Related work

To solve problems similar to ours, the literature describes the way deep convolution neural networks are used. The success of the most used approaches is based on the fact that lately there have been developed more powerful graphics processing units and the big number of annotated images that appeared. Helpful was also the fact that there were developed many classification [7], object detection [11] and even segmentation [9] algorithms.

The success of segmenting an image is determined by the edge detection in different contexts. This is why a lot of work consisted in finding the edge's features.[13] Fortunately, it is believed that deep learning can effectively learn edge features.

As described above, many researchers have utilized deep learning in medical image analysis. With the advance of deep learning, these methods obtained outstanding performances. Considering these results, the medical imaging research community started manifesting interest for deep learning-based methods used to detect cancer. In the following sections, two approaches used for prostate cancer detection will be described, with the focus on the results they obtained.

3.1 U-Net for Prostate Segmentation

This approach uses a U-Net which, trained end-to-end, can segment the prostate on MR images accurately and fast. The neural network architecture they used is a classical U-Net, but with some additional improvements to combat overfitting and all the challenges the prostate images came with.

3.1.1 Algorithm

The proposed U-Net neural network contains three parts. The first five stages consist of a compression path which extract features from the data and reduce the resolution by an appropriate stride. From

top to the fourth stage, the number of feature channels is doubled at each stage. In the first stage, the number of feature channels is 64, which means that the last stage will have 512 feature channels. In each stage, a 3x3 convolutions and one 1x1 convolution are performed, followed by one 2x2 max pooling operation for down sampling. On the contrary, the later four stages consist of an expansive path which upsample the features maps and halve the number of feature channels until its original size is reached. These stages have same operations like the stage within compression path except the max pooling operation. On the part of supervised layers, each supervised layer consists of a upsampling layer and a deconvolution layer. The upsampling layer upsamples the features map and then via deconvolution layer obtain the segment result. During training, these supervised layers control the process of training according to the difference between segmentation result and ground truth.

A standard stochastic gradient descent with Dice coefficient were used as a loss function. [14]

3.1.2 DataSet

The model was trained on a dataset containing 1324 images that came form 81 patients. The images was acquired using a Philips 3T MRI scanner and were grouped in volumes consisting of 26 slices, each slice having 512x512 pixels. Because of this small number of data, in order to increase the robustness, reduce overfitting and enlarge the training dataset, the data augmentation strategy was employed by applying translations, rotations and zooms.[14]

The ground truth and result of segmentation were binary images with the possible location of the cancer zone. [14]

3.1.3 Results

In order to evaluate the model, 4 patients with 64 images were randomly chosen before training to take part of the evaluation dataset. To evaluate the algorithm the Dice coefficient was used. The obtained results are described in Table 3.1. These results prove that, comparing with a FCN or a simple U-Net, the proposed approach obtained better score, showing that the proposed method is a viable method for improving neural network's performances for medical images segmentation.[14]

Method	**meanDSC**	**medianDSC**	**maximumDSC**
Approach method	0.885	0.945	0.985
U-Net	0.865	0.940	0.969
FCN	0.759	0.832	0.918

Table 3.1: Comparison between the proposed approach and other methods [14]

3.2 Prostate Cancer Detection using Deep Convolutional Neural Networks

This approach proposed an automated pipeline for two levels of prostate cancer classifcation: slice level and patient level. For slice-level classifcation, a stack of individually trained modifed ResNet CNNs are used. In order to convert slice-level classifcation results into patient level, this paper came with first-order statistical features extractor, a decision tree-based feature selector and a Random Forest classifer. The obtained results seems to be more robust compared to similar studies that proposed CNNs for prostate cancer detection. [12]

3.2.1 Algorithm

The base architecture for this research is a ResNet architecture. Each Residual block consists of convolutional layers and identity shortcut connection that skips those layers and their outcomes are added at the end. In order to boost the performance of the architecture, a fully pre-activated residual network is implemented by applying batch normalization and ReLU activation layers before the convolution layers. Instead of using the classical 2-layer deep ResNet block, the method this paper proposed uses a 3-layer deep bottleneck building block because of the fact it will reduce the training time without sacrificing the performance. The ResNet neural network is a 41 layers deep network. The architecture is composed of 2D convolutional layers with a 7 x 7 filter and a 3 x 3 Max pooling layer and residual blocks. Since the input images were small and the tumorous regions were even smaller, additional ResNet blocks were needed. Te first ResNet block is 3-layer bottleneck blocks with 2D CNN layers with (64, 64, 256) filter size. It will be stacked 4 times. Te second ResNet Block is 3-layer bottleneck blocks with 2D CNN layers with (128, 128, 512) filter size. This one will be stacked 9 times. At the end of the Res Blocks a 2 x 2 Average pooling, a dropout layer and 2D fully connected layer are applied. The optimizer this method used was a Stochastic Gradient Decent one with Binary Cross Entropy as the loss function.[12] The parameters used to train the neural network are describe in Table 3.3. The output of each ResNet will consist of two probabilities associated with each class (PROSTATE CANCER and NON PROSTATE CANCER) for each DWI slice.

3.2.2 DataSet

The data was acquired using using a Philips Achieva 3T from 427 patients. From these 427 patients, 175 patients had clinically significant prostate cancer and 252 patients did not. From all this images, a total of 5,832 2D slices were obtained coming in DWI format. In order to use DWI images as input

Parameter	Value
Initial learning rate	0.001
LR reduce factor	10
Batch size	8
Dropout rate	0.9
Weight decay	0.000001
Momentum	0.9

Table 3.2: ResNet Parameters [12]

for the neural network, each slice was resized to 144 x 144 pixels and center cropped with 66 x 66 pixels such that the prostate was covered. The CNNs were modified to feed DWI data with 6 channels instead of images with 3 channels. In order to test the algorithm on a different data, all the DWI images were separated into three different sets, the training set with 271 patients consisting of 3,692 slices, the validation set with 48 patients containing 654 slices and the test set with 108 patients with 1,486 slices. Before using this images as input for the neural network, they were all normalized.[12]

3.2.3 Results

In order to evaluate the performance of the proposed method, the AUC (area under the curve) and ROC (receiver operating characteristic curve) curve were used. The ROC curve plots true positive rates and false positive rates with different thresholds, while an AUC described the performance of the algorithm with a single number. Regarding the slice-level performance, the best AUC score was obtained by the first CNN, but the other 4 also come with great results. When talking about patient-level performance, the CNN worked also very well.[12] All of these results are illustrated in Table 3.2.

Slice-level CNN	AUC	Confdence Interval
CNN1	0.87	0.84-0.90
CNN2	0.87	0.83-0.90
CNN3	0.86	0.83-0.89
CNN4	0.85	0.82-0.88
CNN5	0.85	0.82-0.88

Patient-level CNN	AUC	Confdence Interval
CNN	0.84	0.76-0.91

Table 3.3: ResNet Results [12]

The obtained results seems to have a superior and a more robust performance compared to similar

studies that proposed CNNs for prostate cancer detection.

Chapter 4

Investigated approach

4.1 U-Net architecture

The approach we used for detecting the prostate cancer is a convolutional network architecture. More precisely, it is a UNet with 3 sections: the contraction, the bottleneck and the expansion. All the convolutional layers were build in a 2D format to fit the input dataset format.

4.1.1 U-Net Model 1

The contraction section is made of 4 contraction stages. The first stage will start with 64 filters. This number doubles after each block so that the neural network can learn the complex structures effectively, which means that the second stage will have 128 features, the third one will go with 256 and the last one will end up with 512 filters. For each stage, two convolutions with 3x3 kernel size are performed, one 1x1 convolution for the output and a 2x2 max pooling is attached for down sampling. Also, after some convolutions, dropout is also performed. Each 3x3 convolution uses RELU as activation function and the 1x1 convolution will use SIGMOID as activation function.

The bottleneck layer mediates between the contraction layer and the expansion layer. This will have 512 features channels and only two 3x3 convolutions and the 1x1 convolution are performed, with no max pooling.

The expansion layer consists of several expansion stages. Each stage of this part includes two kinds of operations. The first one is upsampling which makes the size of feature map increase gradually until it reaches the size of the original input image and the second one is deconvolution (Deconv2D). This operation is used to halve the number of feature channels, so that the number of convolution kernels will be halved after each stage to maintain symmetry. However, because some image information will be lost after every convolution, the features extracted from early stages are get appended by features

of the corresponding expansion layer. This action would ensure that the features that are learned while contracting the image will be used to reconstruct it. The number of expansion and contraction stages is the same (see Figure 4.1).

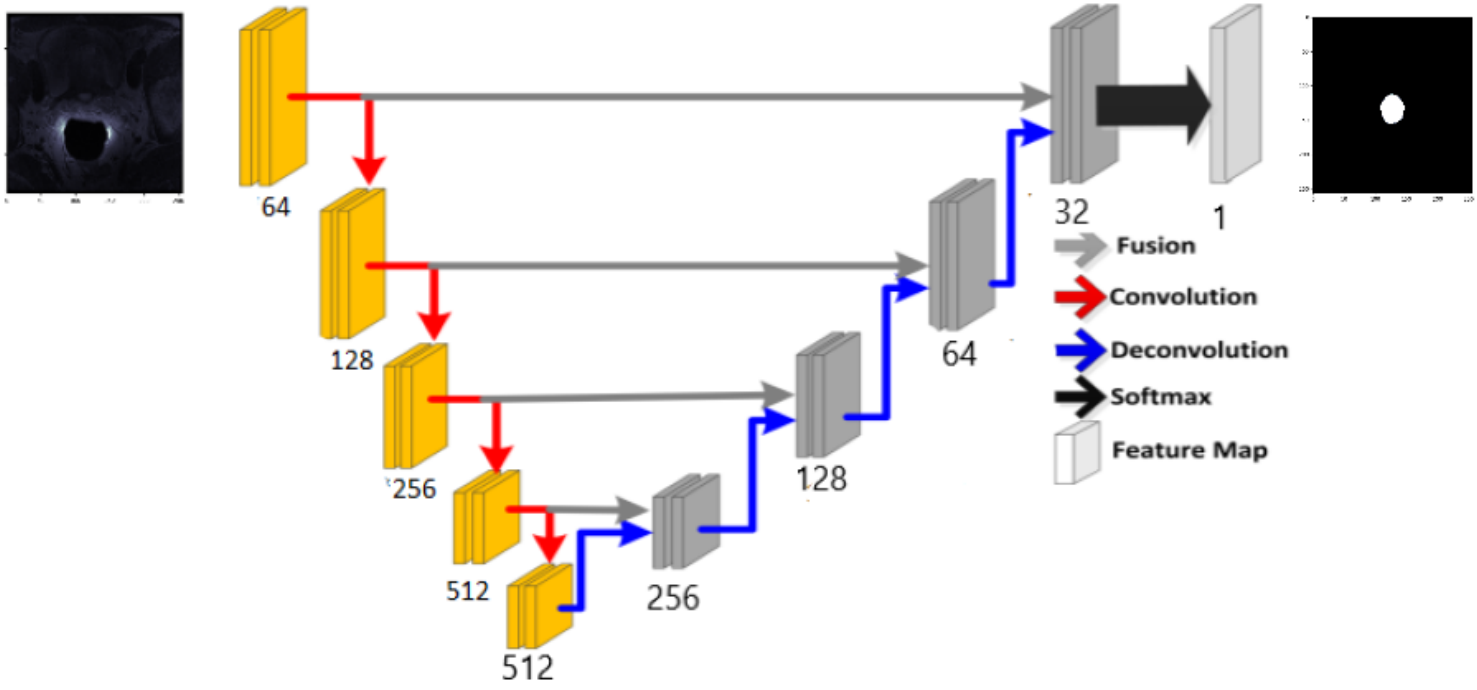

Figure 4.1: U-Net(Model 1) Architecture

4.1.2 U-Net Model 2

The second U-Net architecture is similar with the previous one, but the number of filters at each stage is reduced. The first stage of the contraction section will start with 16 filters. This number is also doubled after each block so that the neural network can learn the complex structures effectively, which means that the second stage will have 32 features, the third one will go with 64 and the last one will end up with 128 filters. Two convolutions with 3x3 kernel size, one 1x1 convolution for the output, a 2x2 max pooling and a dropout are performed for each stage. The activation function used for this model is RELU, excepting for the last layer which will use SIGMOID.

The bottleneck layer will have 256 features channels and only two 3x3 convolutions and the 1x1 convolution are performed, with no max pooling.

The expansion layer consists of 4 expansion stages. Each stage of this part includes upsampling and deconvolution(Conv2DTranspose). The number of filters starts from 128 for the most bottom layer and decreases until it reaches the same size as the first stage from the contraction stage, ending up with 16 filters. (see Figure 4.2).

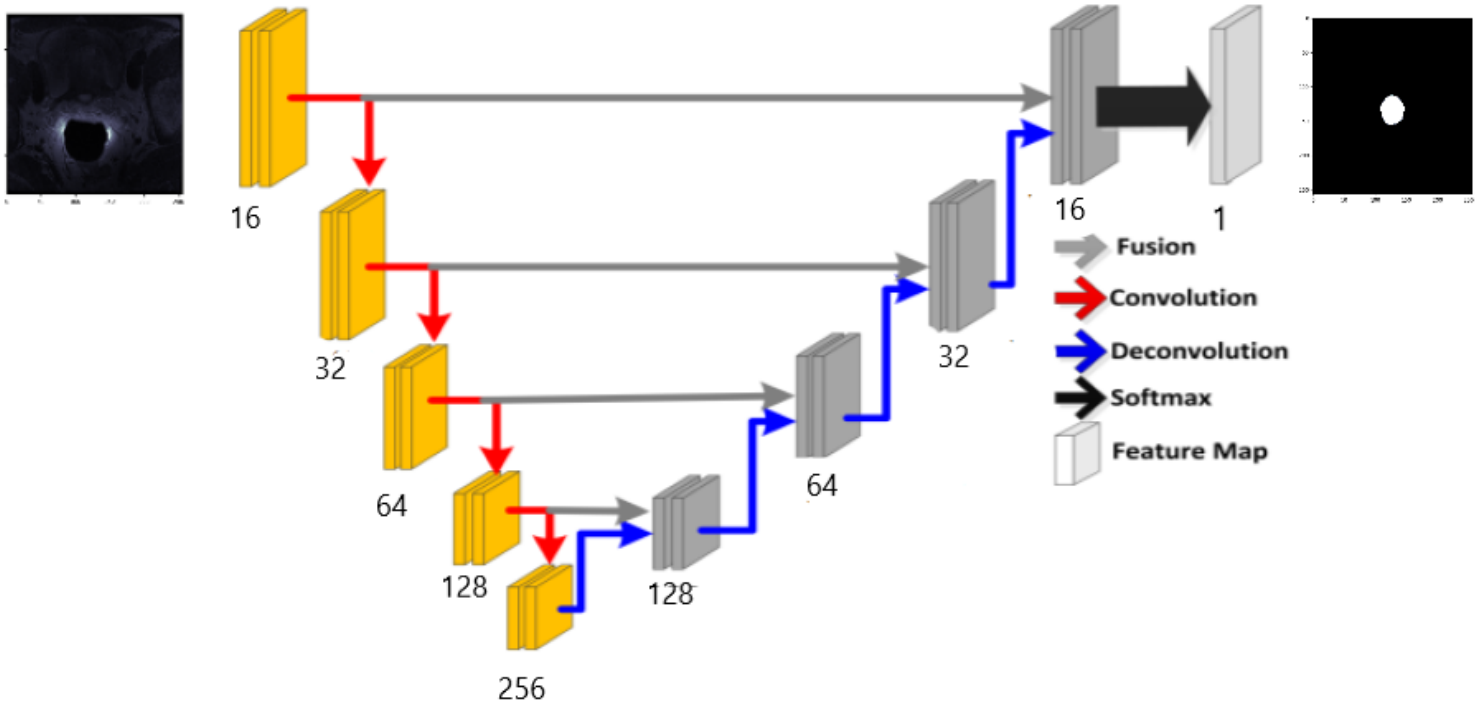

Figure 4.2: U-Net(Model 2) Architecture

4.2 Model evaluation

During training, the supervised layers control the process of training according to the difference between segmentation result and ground truth. In order to measure this difference, we used DICE coefficient, which is essentially a measure of overlap between two samples. The values of the DICE coefficient ranges from 0 to 1 where a Dice coefficient of 1 denotes perfect and complete overlap.

$$DICE = \frac{2\times|A\cap B|}{|A|+|B|}$$

Where [6]:

- |A ∩ B| = the common elements between sets A and B
- |A| = the number of elements in set A
- |B| = the number of elements in set B

Because we have to apply the DICE coefficient on segmentation masks, in order to evaluate the |A ∩ B| an element-wise multiplication between the prediction and target mask was applied followed by the sum of the resulting matrix. The loss function that will be used by the neural network will be $\mathbf{1 - DICE}$.

4.3 Implementation details

For implementing the UNet we used Keras with Python. As layers we used Conv2D, Deconv2D and Conv2DTranspose, as well as MaxPooling2D and Dropout. The neural network will receive the following parameters:

- Optimizer: Adam
- Learning rate: 0.001
- Momentum: 0.9
- Dropout: 0.3
- Batch size:2
- Number of epochs: 250

4.4 Dataset

The dataset had very limited number of images, only 30 patients with 2 examinations which means a total of 60 volumes. Each volume has from 26 to 30 slices, so the final dataset contains 890 slices. Because of this small number of data, the model resulted suffering from overfitting. To increase the robustness and reduce overfitting, we adopted the strategy of data augmentation to enlarge the training dataset. The augmentation transformations include translation, rotation and zoom. Before feeding the model with the input dataset, all the images were normalized and the pixel values were reduced to [0-1] interval.

In order to train and evaluate the model, we chose 75% of the total images for training and the rest for evaluation. We also excluded 1 patient from the dataset in order to use it for testing the model.

Below is an example of an input image and the corresponding prostate mask as well as the input and the corresponding cancer zone:

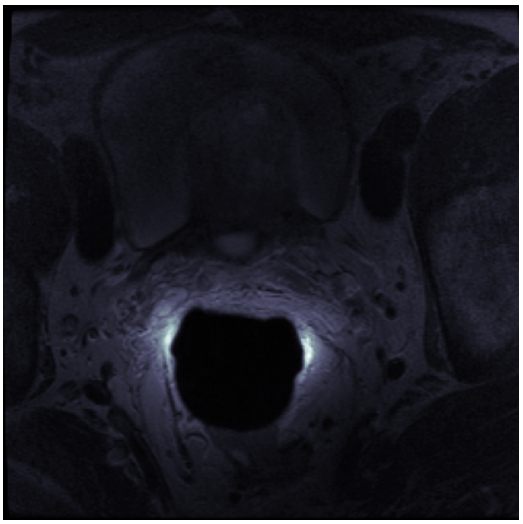

Figure 4.3: Input

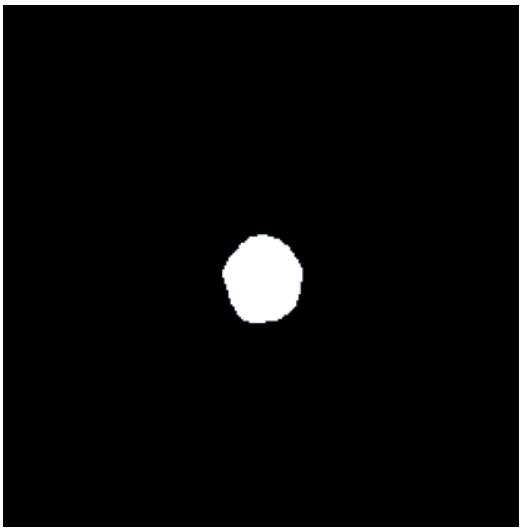

Figure 4.4: Mask (whole prostate)

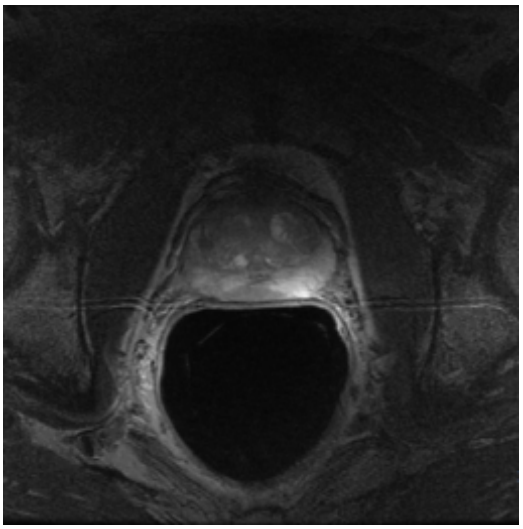

Figure 4.5: Input

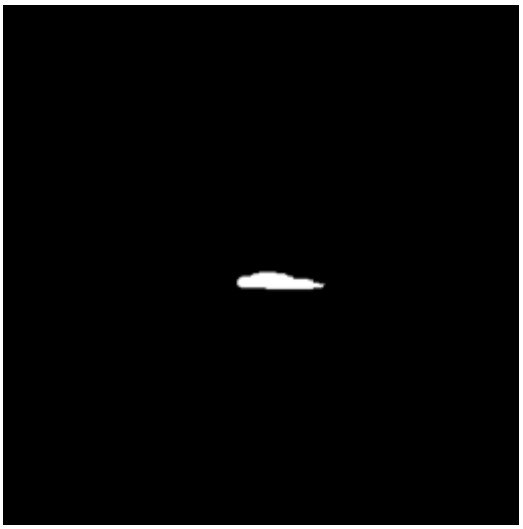

Figure 4.6: Mask (cancer zone)

Chapter 5

Application Requirements

5.1 Application main functionalities

The main functionalities of the application we developed are the following:

- The system should allow the user to upload a medical image that is a representation of the prostate organ
- The system should display the medical image the user has uploaded
- The system should allow the user to check if the medical image has a prostate cancer zone
- The system should allow the user to visualize the prostate cancer zone
- The system save the images that the user uploaded
- The system should allow the user to visualize again the images that he uploaded

5.2 Tools and Technologies

In order to provide to students an easy way to access the application, we developed a web application based on a Python server and a React web client.

The Python server: The server side was written in Python using Keras for the neural network, a generator for applying different transformations on the input images and pydicom and numpy libraries for preparing the input dataset.

The React client was developed using TypeScript and the new Hooks technology. React is a declarative and component-based JavaScript library for efficiently building interactive user interfaces. It uses encapsulated components that manage their own state, then compose them to make complex UIs.[4]

TypeScript is an open-source language which builds on JavaScript, by adding static type definitions. This allow the developers to better understand the code, makes it more readable and thus it reduces errors. Hooks are a new addition in React 16.8. They let you use state and other React features without writing a class[5]. Hooks technology allowed us to develop lighter and more readable components by using hooks like *useState()*, so we could update our components without taking care to update the sensitive state of the components as was done before. Furthermore, we didn't used the classic stateful class component, but the stateless functional components, which are plain JavaScript functions that returns JSX and are easier to develop. To create requests to the server we used Axios, a promise based HTTP client.

Even if our client application is, for now, a simple image uploader and downloader, we created a scalable architecture by creating a generic HTTP service which is used by a custom service for our images, which is used by our UI component.

In order to display medical images (like DICOM and NIFTI), we used the Med3Web library, a high performance web tool for advanced visualization medical volumetric (both in 2d and 3d modes). The latest version can be used with WebGL-enabled desktop browsers (Chrome, Firefox, Opera) and allow limited usage with mobile browsers (Android Chrome). Version for Safari (macOS, iOS) is planned for future.[1] Because medical images can be very dark, one of the advantages of this library is that the user is allowed to change the brightness of uploaded images in order to see them better. Even if they are not mandatory for our application, other features like drawing over the image or calculating distances between two points are available.

Chapter 6

Application (numerical validation)

6.1 Methodology

In order to evaluate our method we used the DICE coefficient, which measures the overlap between the input and the mask. We also output the accuracy, but the model didn't take it into consideration as part of the loss calculation. We trained our model 2 times, one for the prostate segmentation and the other one for the cancer zone. The aim is to predict the location of the prostate and the cancer zone and in order to display these 2 segmentations over the original image. In this way, a student can better analyze and study the prostate and the possible tumor.

The dataset used for training corresponds to real prostate images collected from 30 patients. These images are medical images and come in a DICOM format representing the axial, T2-weighted magnetic resonance imaging of the patient prostate. This dataset was a great challenge for our model since all the images were very dark, with lack of clear boundary specifically at the apex and base and a huge variation of shape and texture between the images from different patients. We also had to resize the images at half of their dimension which made it even more difficult for the neural network to learn and to be able to predict the prostate/cancer zone. The methods described in the Related Work 3 section used different datasets than our model. For instance, in [14], the dataset contained over 1200 images, but, most important, the quality of the these images was better. They were lighter and there was not so much noise, which helped the model to learn faster and more accurate. The second approach[12] presented as related work, had even more slices in the dataset. There were about of 5,832 2D slices, way more than 890 that we used.

6.2 Data

6.2.1 Data acquisition

The training and testing data were taken from an online database provided by the ***The Cancer Imaging Archive (TCIA) Public Access***. All of the imaging studies were acquired at 3 Tesla magnet strength. Due to the scanner hardware upgrade in the middle of the study, 6 of the patients had baseline and repeat study performed on a GE Signa HDxt platform, while the remaining 7 patients were scanned on a GE Discovery MR750w. Transrectal coil within an air-filled balloon was used in all imaging studies. mpMRI protocol included T2-weighted, Diffusion Weighted (DW) and Dynamic Contrast Enhanced (DCE) sequences. The imaging data is accompanied by the following types of derived data: manual segmentations of the total prostate gland, peripheral zone of the prostate gland, suspected tumor and normal regions (where applicable). Segmentations were done by a radiologist with the expertise in prostate MRI volume measurements (for axial T2w images and ADC images) and mean ADC (for ADC images) corresponding to the segmented regions. Type of cancer: confirmed or suspected prostate cancer.[3]

6.2.2 Data storage

The data used for training and evaluating the model is stored locally, on the development server, in a folder called *data* with 2 subfolders, *input* and *mask*. Additionally, when using a data generator, the preprocessed images are also stored on the local development server, in a npy format such as, the next time the model will be trained there will be no need to preprocess and load the dataset again. In this situation, the already npy saved data will be used.

For the data the student will want to analyze, he/she needs to press the *Select image(s) to analyze* button and to select a Nifti image or one or more DICOM slices. The application will upload the image(s) to the server. The server will analyze the file(s) and will return a ZIP archive containing a DCM file, representing the prediction, for each image provided by the student. In the current stage of development, these images are stored only until the next request from a user, so they are not persisted. As future work we want to add the possibility for the student to create a personal account through which it can be saved all his/her uploaded images and their predictions.

6.2.3 Data visualisation

During training, we wanted to visualize the input data, but more over, the predicted output of the model we built. In order to be able to do that, we used **Pyplot** form ***Matplotlib*** library which works

very well with numpy arrays that we used for defining the input for our neural network. The predicted output will also be a numpy array. Therefore, by using **Pyplot** we managed to display and analyze the neural network prediction.

On the client side, the student have two possibilities to visualize medical images, by pressing the *Open* button: loading from computer and loading from an URL. It can be selected a single file or multiple files, if the image is a DICOM with multiple slices. Before displaying the selection, the application shows a small window with the selected image (or the middle one if the student selected multiple slices) and prompts the user to confirm and adjust the luminosity of the image; more exactly the lower bound of the dark pixels and the upper bound of the white pixels. This is a very helpful feature of the Med3Web library since some medical images are very dark and has to be brighter so that they can be easily seen by human. After loading the images into the web client, the user can change the plane (slice) view (thus, the library automatically detects the appropriate display method) and interactively visualize each slice of the image by dragging a slider. Furthermore, in the top-left corner of the image are displayed some information about the image, like it's resolution, and, by selecting the *Get voxel intensity* tool, each click on the image will display the coordinates of that point, in 3D. The application allows a user to create an account and login with that account to save his image(s), including predictions on the server. He can then see them displayed in a list organized by dates.

6.3 Results

As mentioned before, the dataset used for training and testing the model consists of 890 2D slices: 645 for training, 215 for validation and 30 (1 patient) for testing. We built 2 UNets, with the same hyper-parameters that we previously described, but with different architecture. For both the models, we used generator for building the input data, which means that the input will also be the same. Both models were describe in 4.

In the following sections, we will describe the results obtained by this two models, comparing them with the results obtained by other approaches that solved the same problem of prostate cancer segmentation.

6.3.1 Model 1 results

The first model we used is a 2D Unet, having **[64, 128, 256, 512, 512, 512, 256, 128, 64, 32]** number of features per convolutional layer and the input/output image size of 256X256.

After 100 epoch, the results are quite disappointing. The DICE coefficient reached a value of only

0.0023. Even if we let our model run for 250 epoch, the results are the same, the DICE coefficient didn't reach any improvements. Moreover, due the huge the number of features per convolutional layer, the time required by the UNet to complete an epoch was about **5 mins**.

The results are illustrated in Fig 6.1 and Fig 6.2.

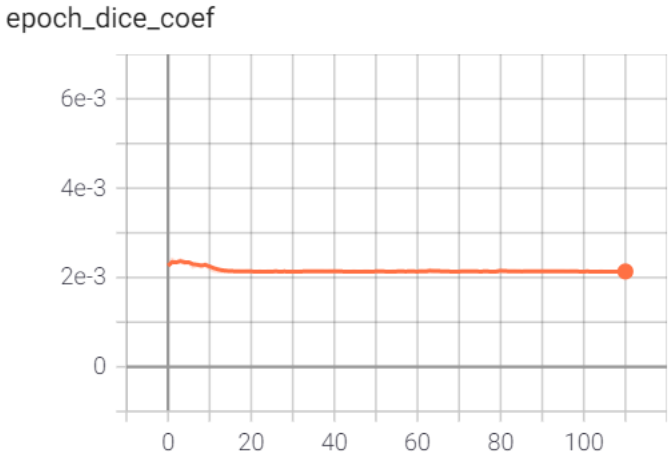

Figure 6.1: Epoch DICE coefficient (x: number of epochs, y: DICE coefficient value)

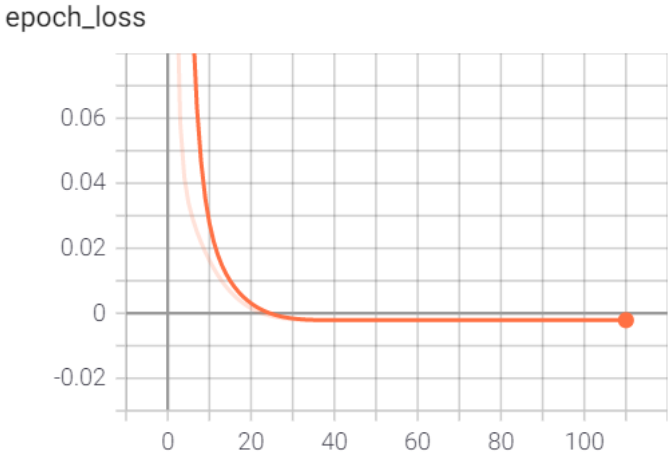

Figure 6.2: Epoch loss (x: number of epochs, y: loss value)

6.3.2 Model 2 results

The second model we used is also a 2D Unet, but, with **[16, 32, 64, 128, 256, 256, 128, 64, 32, 16]** number of features per convolutional layer and the same input/output image size of 256X256.

The number of epochs was also 250, but the results are very different comparing with the previous model. With this UNet, the DICE coefficient started from a value of **0.22** and after 10 epochs reached a value of **0.78**. The problem is that, even after the 250 epoch, it remains almost the same. With each

new epoch completed this value changed just a little bit, remaining between 0.78 and 0.80, while the [14] obtained a maximum DICE value of **0.985**. The model used smaller number of features per layer, which led to shorter time required for completing an epoch. This time, only **about 50 seconds** were need to complete an epoch. Even if the DICE=0.80, the output segmentation is still mostly black. The results are illustrated in Fig 6.3 and Fig 6.4.

Figure 6.3: Epoch DICE coefficient (x: number of epochs, y: DICE coefficient value)

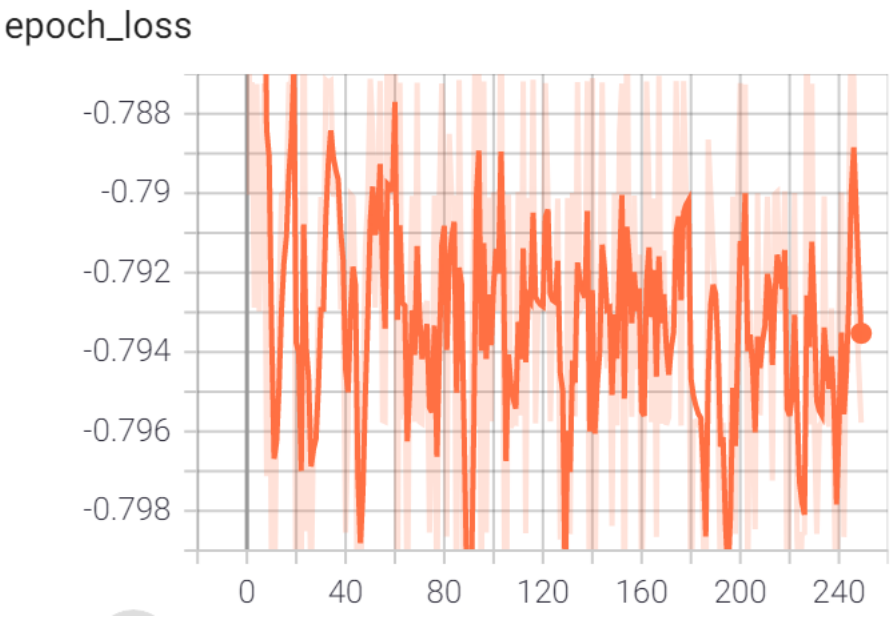

Figure 6.4: Epoch loss (x: number of epochs, y: loss value)

Some small improvements we tried to implement were to use **binary-crossentropy** as loss function and **crop** the input images in order to eliminate the unnecessary information from each slice and increase the number of pixels of prostate zone.

The results we obtained with these changes and by using Model 2 are different than the previous ones. The dice coefficient started from a value of **0.0037** and the maximum value it reached was **0.05**. The loss value was **0.161** after the first epoch and in the end (after 20 epochs) it increased up to **0.0221**. These results are illustrated in the logs below(Fig 6.5):

```
Epoch 1/20
 - 83s - loss: 0.1610 - dice_coef: 0.0037 - binary_accuracy: 0.9828 - val_loss: 0.0986 - val_dice_coef: 0.0060 - val_binary_accuracy: 0.9872
Epoch 2/20
 - 81s - loss: 0.0463 - dice_coef: 0.0025 - binary_accuracy: 0.9958 - val_loss: 0.0964 - val_dice_coef: 0.0044 - val_binary_accuracy: 0.9872
Epoch 3/20
 - 82s - loss: 0.0336 - dice_coef: 0.0071 - binary_accuracy: 0.9958 - val_loss: 0.0734 - val_dice_coef: 0.0021 - val_binary_accuracy: 0.9872
Epoch 4/20
 - 82s - loss: 0.0310 - dice_coef: 0.0036 - binary_accuracy: 0.9958 - val_loss: 0.0732 - val_dice_coef: 0.0023 - val_binary_accuracy: 0.9872
Epoch 5/20
 - 82s - loss: 0.0299 - dice_coef: 0.0090 - binary_accuracy: 0.9958 - val_loss: 0.0879 - val_dice_coef: 0.0048 - val_binary_accuracy: 0.9872
Epoch 6/20
 - 82s - loss: 0.0276 - dice_coef: 0.0045 - binary_accuracy: 0.9958 - val_loss: 0.0998 - val_dice_coef: 0.0114 - val_binary_accuracy: 0.9872
Epoch 7/20
 - 82s - loss: 0.0279 - dice_coef: 0.0119 - binary_accuracy: 0.9958 - val_loss: 0.1032 - val_dice_coef: 0.0038 - val_binary_accuracy: 0.9872
Epoch 8/20
 - 81s - loss: 0.0254 - dice_coef: 0.0058 - binary_accuracy: 0.9958 - val_loss: 0.0920 - val_dice_coef: 0.0033 - val_binary_accuracy: 0.9872
Epoch 9/20
 - 81s - loss: 0.0255 - dice_coef: 0.0043 - binary_accuracy: 0.9958 - val_loss: 0.0954 - val_dice_coef: 0.0049 - val_binary_accuracy: 0.9872
Epoch 10/20
 - 82s - loss: 0.0245 - dice_coef: 0.0072 - binary_accuracy: 0.9958 - val_loss: 0.0711 - val_dice_coef: 0.0033 - val_binary_accuracy: 0.9872
Epoch 11/20
 - 81s - loss: 0.0272 - dice_coef: 0.0551 - binary_accuracy: 0.9958 - val_loss: 0.0734 - val_dice_coef: 0.0028 - val_binary_accuracy: 0.9872
Epoch 12/20
 - 82s - loss: 0.0225 - dice_coef: 0.0057 - binary_accuracy: 0.9958 - val_loss: 0.0773 - val_dice_coef: 0.0035 - val_binary_accuracy: 0.9872
Epoch 13/20
 - 81s - loss: 0.0229 - dice_coef: 0.0247 - binary_accuracy: 0.9958 - val_loss: 0.1672 - val_dice_coef: 0.0099 - val_binary_accuracy: 0.9872
Epoch 14/20
 - 82s - loss: 0.0239 - dice_coef: 0.0165 - binary_accuracy: 0.9958 - val_loss: 0.0967 - val_dice_coef: 0.0031 - val_binary_accuracy: 0.9872
Epoch 15/20
 - 81s - loss: 0.0268 - dice_coef: 0.0115 - binary_accuracy: 0.9958 - val_loss: 0.0843 - val_dice_coef: 0.0030 - val_binary_accuracy: 0.9872
Epoch 16/20
 - 80s - loss: 0.0233 - dice_coef: 0.0055 - binary_accuracy: 0.9958 - val_loss: 0.0790 - val_dice_coef: 0.0039 - val_binary_accuracy: 0.9872
Epoch 17/20
 - 81s - loss: 0.0222 - dice_coef: 0.0062 - binary_accuracy: 0.9958 - val_loss: 0.0964 - val_dice_coef: 0.0038 - val_binary_accuracy: 0.9872
Epoch 18/20
 - 81s - loss: 0.0220 - dice_coef: 0.0059 - binary_accuracy: 0.9958 - val_loss: 0.0805 - val_dice_coef: 0.0037 - val_binary_accuracy: 0.9872
Epoch 19/20
 - 80s - loss: 0.0217 - dice_coef: 0.0066 - binary_accuracy: 0.9958 - val_loss: 0.0973 - val_dice_coef: 0.0038 - val_binary_accuracy: 0.9872
Epoch 20/20
 - 80s - loss: 0.0221 - dice_coef: 0.0065 - binary_accuracy: 0.9958 - val_loss: 0.0880 - val_dice_coef: 0.0039 - val_binary_accuracy: 0.9872
```

Figure 6.5: Logs

6.4 Discussion

As seen before, the results are quite disappointing. The UNets didn't manage to learn and identify, neither the prostate, nor the cancer area. The first reason this could happen may be the data. The input images are very dark, with a lot of noise and just a small part of them represents the prostate and even a smaller one describes the cancer area. The second reason may be the resize that is applied on every image. This operation reduces the quality of the data, making harder for the neural network to learn where the prostate or the cancer zone may be. Other reason that may explain these results could be the small amount of data. The data sets consists of only 890 slices. Moreover, the first and the last slices of every patient are very dark and the prostate area is very small. Due the hardware limitations, the hyper-parameters had to be adjusted accordingly. The batch size = 2, which may also have contributed to these results.

Chapter 7

Conclusion and future work

In this paper, we proposed and compared two U-Net models to accurately segment the prostate MRI. The hyperparameters are the same for both the U-Net, but the architecture is different. By using this 2 models, we were able to compare and analyze how the architecture of a neural network and the number of features can influence the performance, the accuracy and the speed of the learning process.

As we seen before, one of the greatest challenge we had to face was the input. The data set contained only 900 slices which had to be divided for training, validation and testing. Moreover, the images were dark, the prostate area was very small and each slice had to be resized at half of its normal size. All this made the learning process more difficult, which resulted in the U-Net to have problems while learning to detect the cancer zone. To overcome all these problems, we tried to improve the input data set using the data augmentation strategy and applying transformations such as translation, rotation and zoom.

Even if the results are not very satisfying and the U-Net models are not yet able to detect the prostate cancer zone very accurately, they still prove that a U-Net can be used for prostate cancer segmentation. With more training time and more competent hardware, if the number of epochs and the batch size is increased, as well as the original size of the input images is kept, and a better preprocessing of the input data, the result may improve and the models will be able to accurately detect the prostate cancer zone.

With all these improvements, and the intelligent algorithm included in the web application, the resulted software will be very helpful for a student that wants to study the prostate and the possible tumor zones. But, it is very important to keep in mind that, even if the model has learned to detect the prostate cancer zone very well, errors still may occur. Therefore, it is recommended to use AI as a complimentary evaluation and interpretation system, without relying solely on model-based decisions.

Bibliography

[1] Med3Web Documentation. https://github.com/epam/med3web.

[2] Prostate Cancer. https://www.reginamaria.ro/articole-medicale/cancerul-de-prostata. Accessed: 2020-10-14.

[3] QIN-PROSTATE-Repeatability. https://wiki.cancerimagingarchive.net/display/Public/QIN-PROSTATE-Repeatabilit.

[4] React Documentation. https://reactjs.org.

[5] React Hooks Documentation. https://reactjs.org/docs/hooks-intro.html.

[6] Sorensen-Dice coefficient. https://en.wikipedia.org/wiki/S%C3%B8rensen%E2%80%93Dice_coefficient.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[8] Saifeng Liu, Huaixiu Zheng, Yesu Feng, and Wei Li. Prostate cancer diagnosis using deep learning with 3d multiparametric mri. In *Medical imaging 2017: computer-aided diagnosis*, volume 10134, page 1013428. International Society for Optics and Photonics, 2017.

[9] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[10] Muhammad Imran Razzak, Saeeda Naz, and Ahmad Zaib. Deep learning for medical image processing: Overview, challenges and the future. In *Classification in BioApps*, pages 323–350. Springer, 2018.

[11] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[12] Sunghwan Yoo, Isha Gujrathi, Masoom A Haider, and Farzad Khalvati. Prostate cancer detection using deep convolutional neural networks. *Scientific Reports*, 9, 2019.

[13] Lefei Zhang, Qian Zhang, Liangpei Zhang, Dacheng Tao, Xin Huang, and Bo Du. Ensemble manifold regularized sparse low-rank approximation for multiview feature embedding. *Pattern Recognition*, 48(10):3102–3112, 2015.

[14] Qikui Zhu, Bo Du, Pingkun Yan, Baris Turkbey, and Peter Choyke. Deeply-supervised cnn for prostate segmentation. *Scientific Reports*, 2017.