International Master's Thesis

Computer aided renal calculi detection using Convolutional Neural Networks

Antai Llaquet Bayo
Technology

Studies from the Department of Technology at Örebro University
örebro 2016

Computer aided renal calculi detection using Convolutional Neural Networks

Studies from the Department of Technology
at Örebro University

Antai Llaquet Bayo

Computer aided renal calculi detection using Convolutional Neural Networks

Supervisor: Martin Längkvist

Examiners: Amy Loutfi
Andreas Persson

Title: Computer aided renal calculi detection using Convolutional Neural Networks

ISSN

Abstract

In this thesis a novel approach is developed to detect urethral stones based on a computer-aided process. The input data is a CT scan from the patient, which is a high-resolution 3D grayscale image. The algorithm developed extracts the regions that might be stones, based on the intensity values of the pixels in the CT scan. This process includes a binarizing process of the image, finding the connected components of the resulting binary image and calculating the centroid of each of the components selected. The regions that are suspected to be stones are used as input of a CNN, a modified version of an ANN, so they can be classified as stone or non-stone. The parameters of the CNN have been chosen based on an exhaustive hyperparameter search with different configurations to select the one that gives the best performance. The results have been satisfactory, obtaining an accuracy of 98,3%, a sensitivity of 99,5% and a F1 score of 98,3%.

Acknowledgements

Thanks to Martin Längkvist for the supervision of the project, the time spent on it and his advice during the development of the thesis.

I appreciate the help from Mats Linden for his effort, time and explanations about the stones and the information that the CT scans contain as well as his advise to develop the project.

Contents

List of Figures

List of Algorithms

List of Tables

Chapter 1
Introduction

The main purpose of the project is to develop an automatic computer aided detector of urethral stones. A kidney stone (renal calculus or nephrolith) is a solid piece of material formed in the kidney by minerals contained in the urine. If kidney stones grow sufficiently large, they may block the ureter. This can cause pain, nausea, vomiting, fever, blood in the urine, pus in the urine and painful urination [16]. The lifetime risk of of being diagnosed with kidney stones is between 10% and 25%, depending on the country. The annual new cases a year is around 0,5% [7].

Risk factors of getting a kidney stone are related with genetics, being overweight, taking some types of foods and medication and not drinking enough fluids. Kidney stones are classified by its mineral composition and position. The diagnosis is based on symptoms, urine tests, blood tests, X-ray images and CT scans (computed tomography) [4]. The diagnosis is made by a specialist radiologist. The advantages of building an automatic urethral stone system are that it would be faster and cheaper than the actual process used, in which a specialist has to identify the existence of the urethral stone and its position.

Treatment of urethral stones depends on their size, composition and position. Small stones are expelled without any treatment except pain medication and drinking lots of fluids. Bigger stones are treated by Shock wave lithotripsy (used to break the stone in smaller pieces), ureteroscopy (used to remove the stone or break it using a laser), and percutaneous nephrolithotomy (removing the stone by using a small surgery).

In this work, the data used to detect if a patient has a stone and its position is only a CT scan (computed tomography) of the patient. A computed tomography uses the combination of multiple x-ray images taken from different angles to produce cross-sectional images of the patient (slices). The result

is a 3D image of a specific area. The information contained in the CT scan is a grayscale 3D image, in which the value of every pixel is directly related to the type of material that occupies that position. Since the kidney stones are made of a specific set of substances, the value of the pixels occupied by a kidney stone can take a specific range. However, there are different kind of components that are made of this specific set of materials in a human body. Bones and other type of materials concentrations have values of pixels inside the same range as kidney stones.

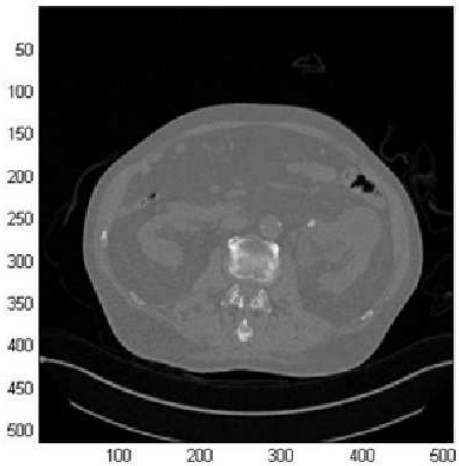


Figure 1.1: Example of slice of a CT scan.

This is a project in collaboration with the hospital of Örebro. The computer-aided algorithm is programmed using Matlab. The system is based in a Convolutional Neural Network, which is a modified version of a Artificial Neural Network. There are several works that use extracted features to classify kidney stones, therefore, a comparison will be done.

1.1 Objectives

The main objectives of this thesis are to detect the presence or not presence of a urethral stone in a patient and its position based on a CT scan. The algorithm used to detect the existence of a urethral stone includes the detection of its position at the same time, as a result of the process used.

One of the challenges with computer-aided urethral stones detection is to differentiate between real stones and objects that might look similar to a real stone. The accuracy of the system is an important variable for the general performance of the system. However, due to the application of the system, the sensitivity in stones is much more important and critical, since a false positive can be checked easily in the images, but a false negative can have disastrous consequences. For this reason, the main objective of the system is to have a high accuracy and sensitivity when classifying stones.

Another challenge with automatic kidney stone detection that is addressed in this thesis is how to deal with the enormous quantity of information that a CT scan contains. A single CT scan of the abdomen contains around one hundred million pixels. So, the amount of variables in the input of the system is huge, which means that the parameters used in the algorithm to process the data will be huge too, since the input data is raw data. It is necessary that a learning system has a number of training examples proportional to the amount of the parameters learned to perform well and have a high accuracy rate.

1.2 List of abbreviations

- ANN: Artificial Neural Network
- CC: Connected Components
- CNN: Convolutional Neural Network
- CT scan: Computed Tomography
- DHV: Difference Histogram Variation
- MSER: Maximally Stable Extremal Region
- ROC: Receiver Operating Characteristics Curve
- SVM: Support Vector Machine
- TV: Total Variation

1.3 Contributions

This thesis has a set of contributions to the area of computer-aided detection of kidney stones from CT scans:

- Use an unsupervised feature learning algorithm, namely the CNN, to perform urethral stone classification from raw CT scans.

- Modify the CNN to work on 3D volumes instead of 2D images.
- Perform a comprehensive parameter search that gives an in-sight on the importance of selecting the hyperparameters of the proposed method.

1.4 Outline

The rest of this thesis is organised as follows:

Chapter 2 gives an overview of related work that use CT scans as input data to detect kidney stones on applications of Convolutional Neural Networks that use CT scan images as input.

Chapter 3 describes the system and subsystems used in this work. The chosen hyperparameters are discussed in this chapter too.

Chapter 4 contains an evaluation of the performance of the system built and a comparison with a regular ANN trained by features extracted from CT scan images and the pixels directly.

Chapter 5 summarizes the objectives of the system and suggests future work related to this project

Chapter 2
Related work

There are some works which use automatic detection systems to detect the existence of a kidney stone based on CT scans. In [14] the authors developed a total variation (TV) flow method to reduce image noise within the kidneys while maintaining the characteristic appearance of renal calculi. Maximally stable extremal region (MSER) features are calculated to identify calculi candidates. Finally, the authors compute texture and shape features that are imported to support vector machines for calculus classification. The texture features include mean and standard deviation of intensity values at a segmented candidate, local binary pattern, and histogram of oriented gradients. The shape features include the volume of a candidate, two aspect ratios of height/length and width/length of the candidate, and the distance of the current candidate to the kidney boundary. Shape and texture features are fed into the SVM classifier. The method was validated on a dataset of 192 patients. The results are show a false positive rate of 8 per patient and a sensitiveness of 69%.

Another work which uses extracted features to detect kidney stones is [12]. The shape's features include disperseness, convex hull depth, and lobulation. The internal textures features include edge density, skewness, difference histogram variation (DHV), and the graylevel co-occurrence matrix moment. For evaluation of the diagnostic accuracy of the shape and texture features, an artificial neural network (ANN) is trained and receiver operating characteristics curve (ROC) analyses are performed. Fifty-nine ureter stones and fifty-three vascular calcifications on precontrast CT images of the patients are evaluated. The Az value is 0.85 for the shape parameters and 0.88 for the texture parameters.

In the previous works the maximum accuracy is 88%. Moreover, the classifiers developed used extracted features from the CT scans, but not the pixels

directly. The main disadvantage of using extracted features from the images and not the pixels themselves is that part of the information contained in the images is lost and it is not used [22]. Moreover, the performance of the classifiers based on extracted features is heavily dependant on the choice of the measures. These features are chosen arbitrarily and require expert knowledge to decide which will be used. To improve the performance of the classifiers based on features new features have to be chosen, which takes time and effort. One of the contributions of this thesis compared with the previously mentioned works is to use the pixels as input of the system but not extracted statistical measures from them.

One challenge with using the pixels as input is that the amount of pixels in a CT scan is around one hundred million. Most classifiers use an equal amount of parameters to the amount of input variables. If the amount of parameters in the classifier are comparable to the amount of input variables, and all the pixels in the CT scan are used, the classifier will get really huge, and as a consequence, it takes long time and it is difficult to train.

One type of algorithm used for automatic feature learning in images is a modified artificial neural network, which is called Convolutional Neural Network (CNN). The main advantage of this type of artificial neural network is that the number of parameters is not directly related to the number of input variables and can be much lower [11] [10]. More details of this algorithm are given in chapter 3.

There are some recent works that use CNN with CT scans as input in applied medicine, but not in kidney stone detection. In [18] they present a probabilistic approach for pancreas segmentation in abdominal computed tomography (CT scans), using multi-level deep convolutional networks. First a set of regions are selected as input to the CNN based a Random forest classifier. Then these regions are used to train the CNN using a set of bounding boxes around each region at different scales. The evaluation is based on CT images of 82 patients. They obtain an accuracy of 83.6%.

In [13] an automatic method based on convolutional neural networks is presented to segment lesions in livers from CT images. Firstly, Gaussian smoothing filter is used to reduce noise. Secondly, images are normalized and then, the CNN is trained. Experimental evaluation is performed on 30 CT images. Precision and recall achieved are 82.67% and 84.34% respectively.

In this thesis, it is going to be used a CNN with a three dimension input image. Three dimension CNNs have been used in different works when

classifying, but the input is not always a 3D image. In [6], the authors use a 3D input CNN to classify actions made by humans. There are two spatial dimensions and one time dimension. The CNN has three convolutional layers and two subsampling layers. The actions are classified in three labels. They achieved a precision of 71 %. In [9] they use a similar technique when classifying videos using a dataset of 1 million videos belonging to 487 classes. Each input consists of several frames in a row, so the input is a 3D matrix with two spatial dimensions and one time dimension. They use different configurations of CNN and the maximum accuracy obtained is 63.9%.

In other works, the input of the CNN has a three dimension image, but the third dimension is not deep. In [21] they introduce a model based on a combination of convolutional and recursive neural networks (CNN and RNN) for learning features and classifying RGB-D images (third dimension has size equal to four). The CNN layer learns low-level translationally invariant features which are then given as inputs to a RNN in order to compose higher order features. There are 51 different classes of objects. They get an accuracy of 86,8 %.

Finally, there are works in which the CNN has a 3D deep input image, like in this thesis. In [8] they present a 3D Convolutional Neural Network developed for the segmentation of brain lesions. The developed system segments pathology after processing a 3D patch at multiple scales. The results show a Dice score of 64%. Regardless its size, the network is capable of processing a 3D brain volume in 3 minutes. In [15] the aim is to classify objects using as input a 3D occupancy grid map in a supervised 3D Convolutional Neural Network. They use two convolutional layers and one pooling layer, getting a F1 score of 73%.

Chapter 3
Method

In this section, the method of the thesis is going to be overviewed. The algorithm used to detect urethral stones and its position is based on a convolutional neural network and a set of pre-processing subsystems. The main purpose of the pre-processing subsystems is to reduce the amount of data that the convolutional neural network has to use as input, which is the subsystem used for classification. The background related to the systems previous to the CNN is going to be overviewed, as well as the basic theory behind the use of CNN's. The candidates are classified using a Softmax classifier, which is the last layer of the CNN.

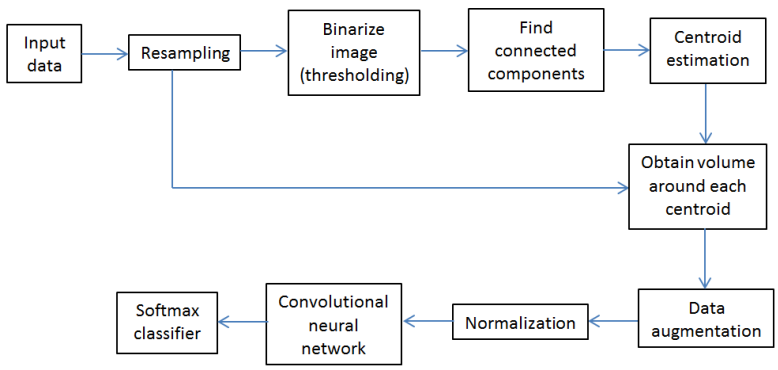


Figure 3.1: Scheme of the system.

In 3.1 there is a scheme of the system used. The CT scans are the input of the system. The first step of the system is to resample the data, then the images are binarized in order to calculate the centroid of each connected component

and, finally, the regions around the objects that can be stones are used as input in the CNN, which extracts some features used to classify the candidates.

3.1 Input data

The input data of the system are a set of CT scans. A CT scan can be described as a 3D grayscale image of a patient. A 3D grayscale image is an application of three natural numbers (position of each pixel) to a natural number that represents the intensity of the gray in the image:

$I \in \mathbb{N}^3$ represents the intensity c in pixel position i, j, k by $I_{ijk} = c$ where $c = [-1024\ 3072]$.

The full CT scan is used as input of the system, but only specific volumes around the objects that might be stones are classified by the CNN. These volumes are chosen by the subsystems previous to the CNN.

When training classifiers, it is important to have enough data, so the neural network has enough examples to learn from. The amount of available data is quite low and there are few examples of stones. As the number of examples increases, the accuracy of the network is higher. If the training set is small compared to the amount of parameters in the network, the model will not generalize well to new unseen examples.

Data augmentation

In this work, the amount of data is not big enough compared to the number of parameters in the neural network. In every patient there are thousands of examples of objects that are not stones, while there is only one example of stone. If the neural network is trained using all the data, it will learn to classify every object as a no stone, since the accuracy will be very high due to the class-imbalance. It is important to have a close number of examples of every class, so the neural network can find the features that define every class of objects. Due to the limited number of CT scans from patients, if the stones of every patient and the same amount of objects that are not stones are used, the network does not have enough examples to learn and the accuracy drops down. However, there is not sufficiently available data.

To solve this issue, the stones available have been copied several times to train the network. Nevertheless, the data has been copied with some differences to obtain more examples of stones than the existing. The process that has been used to obtain more examples is based on rotating the images and translating

them slightly. The centroid of each pixel has been moved one pixel to each of the 26 neighbour pixels. Each of these copies has been rotated 90 degrees in two consecutive directions, obtaining 24 copies of each copy. Doing this, there can be obtained 648 copies of stones from every original, obtaining a big amount of examples of stones. The objects that are not stones do not need to be copied, since there are thousands of examples in every patient. In Figure 3.2 it can be seen an example of three copies of a single stone in a single slice.

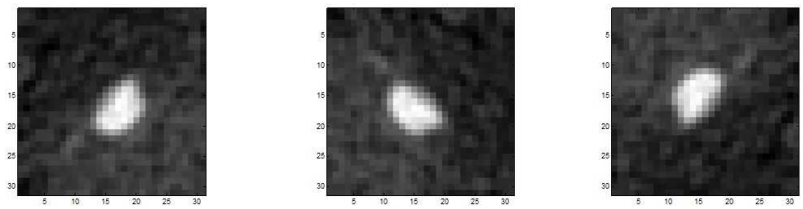


Figure 3.2: Example of three slices copied from a single one.

3.2 Resampling

The pixels in the CT scans, used as the input of the system are not equidistant. Neighbour pixels in one slice are separated 0.82 millimetres, but slices are separated 1 mm. The first step is to resample the data, so the new slices are separated 0.82 millimetres as well. This process is important for the following subsystems, as the subsystem which calculates the centroid and to obtain the volume of the components in the CT scans. The volume of the components will be used to prune some of them, so only the most likely objects to be stones are going to be classified by the CNN.

The resampling has been done by selecting each of the pixels that occupy the same position on each of the slices and creating an array out of them. So, there have been created as many arrays as positions on each slice. Then, each array has been resampled with the same proportion and the resulting arrays have been put all together to get a 3D image with equidistant pixels in the three dimensions.

3.3 Binarizing image

The input of the system is a CT scan, which is a grayscale 3D image. Most of the pixels contained in a CT scan are not useful to obtain the existence and position of the stones. Moreover, a great amount of them are not even inside the body. In the CT scan, the value of the pixels is related with the type of material that is occupying that position. The kidney stones are composed by a set of minerals that have a range of values in the CT scan. In this way, only the zones around pixels with intensity values inside this range are classified as stone or not stone by the convolutional neural network, instead of all volume in the CT scan. This reduces significantly the amount of input variables in the CNN.

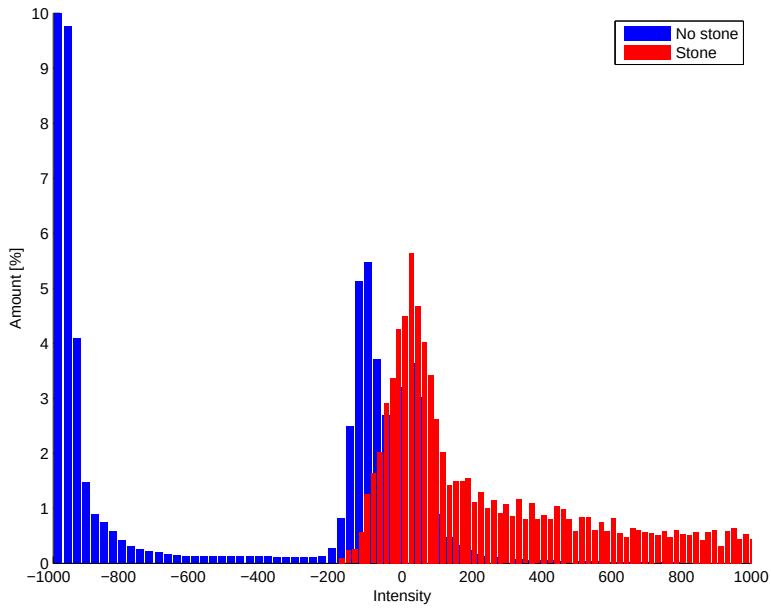


Figure 3.3: Amount of pixels with a given intensity that belong to the full input data (blue) and 7x7x7 volumes around the stones (red).

The figure 3.3 contains the histogram of the amount of intensity values in a CT scan and the amount of intensity values of pixels in a volume of 7x7x7 pixels around each stone. As it can be seen, the pixels around the centroids of

the stones take a specific range and distribution, completely different to the distribution of pixels around all the CT scan.

A set of operations are computed to detect the volumes inside the CT scan that are likely to be stones. The first operation is to binarize the image, in which pixels that have a value in the range of the values taken by the minerals in the urethral stones take white colour (1) and pixels that are not in the range of the minerals take black colour (0). The thresholding value used has been proposed by an expert radiologist and the result of this operation is a binary 3D image. A binary image is an image in which each of the pixels can be either black or white $\mathsf{c} = \{0, 1\}$.

As can be seen from Figure 3.4, some histograms with the values that have objects that are stones and objects that are not stones have been built in order to check that the chosen threshold is related to the acquired data. The range of the intensity values in the pixels that belong to stones is shorter than the range of intensity values in pixels that do not belong to stones. All in all, the majority of pixels that belong to objects that are not stones belong to the same range and there are few that are outside the range. Actually, there are so few that they cannot be seen in the histogram.

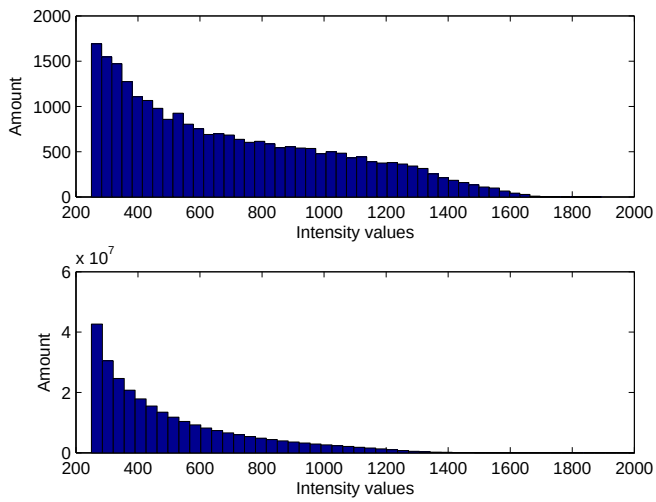


Figure 3.4: Histogram of pixels intensity that belong to stones (up) and histogram of pixels intensity that belong to non-stones (down).

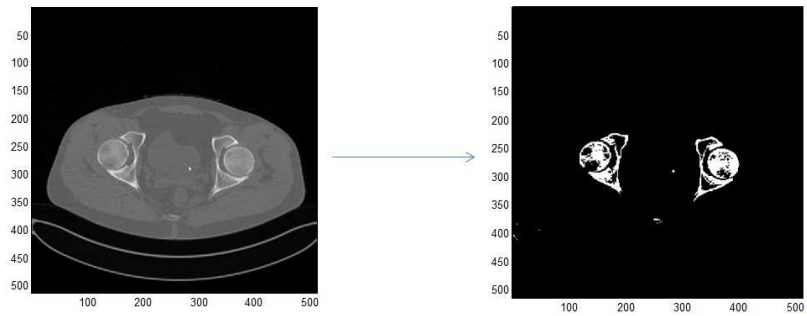


Figure 3.5: Binarizing process. In the left a slice of a CT scan before binarizing, in the right the same slice converted to a binary image.

In figure 3.5 there is an example of slice of a CT scan before and after binarizing. The white pixels in the binary image belong mostly to bones, stones and calcifications.

3.4 Connected components

Next step is to separate each of the connected components that have a white colour in the binary image got after the binarizing process. The objective is to separate the possible objects that can be stones from each other and from the full image. In this way, the amount of pixels that are going to be classified by the CNN is going to decrease. While the connected components are identified, the ones that have a volume bigger and smaller than a given threshold are not taking into account, since they are bones or noise. The threshold has been set after building a histogram out of the objects that are known as stones and the ones that are not stones. The position of each of the stones is known (it is part of the information got by the hospital), so, the components that are placed in those positions are stones, while the objects that occupy other regions are not stones.

Connected-component labelling is used in computer vision applications to detect connected regions in binary digital images. In other words, these type of algorithms find a set of neighbour pixels that share the same value and then label them as the same region. The objective of finding the connected components in this thesis is to find the regions that might be stones and separate them from the 3D image.

In this work the process is made in 3D, so the result of the process are a set of volumes in which pixels are neighbours of others and share the same value. There are different algorithms used to find the connected components in an image. The flood fill algorithm has been used in this project, in which two pixels are neighbours if one of them is one of the 26 3D pixels around the other.

The flood fill algorithm consists of two loops, one inside the other. The outer loop selects a random pixel of the desired colour which has not been labelled so far and is labelled as a new region. The inner loop checks if the neighbours of the starting pixel have the same intensity value (colour). In case they have the same value, they are labelled as the same region. The neighbour pixels of any pixel that is included in the region and are not labelled in any region are checked and included in the same region. The loop continues until all the neighbour pixels of any pixel that has been labelled have been visited. When there are not any pixels in the image with the desired value, the algorithm is over. The output of the algorithm are a set of regions,this is a set of points with the same neighbouring value.

Algorithm 1 Flood-fill algorithm

Require: Binary image

```
for all unlabelled pixels. do
    Put pixel in the queue.
    Label pixel with a new label (L).
    for all Pixels in the queue (P). do
        for all unlabelled neighbour pixels of P (N). do
            if P=N then
                Label N with label L.
                Put N in the queue.
            end if
        end for
    end for
end for
Return labelled connected components.
```

The process of taking into account only components in a range of volume has been done because the amount of objects can be reduced considerably. The threshold used has a big range compared to the maximum and minimum obtained in the stones, since the principal objective is to do not misclassify any stone. The maximum size of a stone are 2609 pixels, while the minimum

size are 4 pixels. So, any object that has more than one pixel and less than 10000 pixels is considered for classification and the objects that are not in this range are not classified, since we consider that they cannot be a stone. This thresholds have been chosen because they represent 4 times less than the minimum obtained value and around 4 times more than the maximum obtained value. Therefore, it is really difficult that there exists a stone which is outside this range of volume, but at the same time, there are a lot of objects that are not stones that are not going to take into account (there are a lot of objects with one pixel) and the bigger objects, which require more computation, are not going to be included neither.

In figure 3.6 there can be seen the histogram of the size of the stones. Most of the stones are small, and only some of them grow bigger. A really small amount grow really big, but since there is the possibility, bigger objects have to be considered for classification.

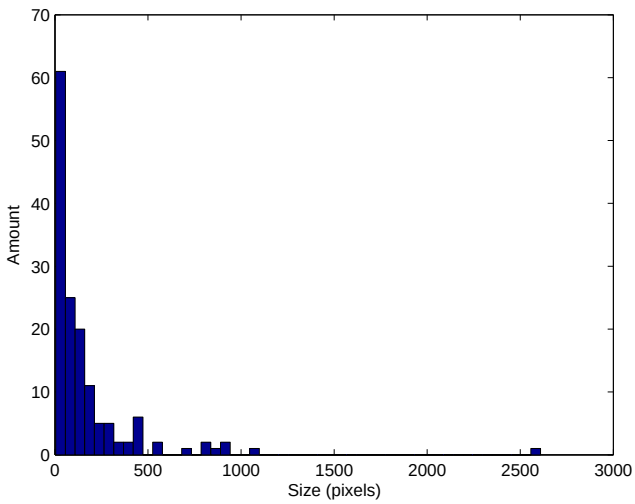


Figure 3.6: Histogram of stones size in pixels.

In figure 3.7 there can be seen the histogram of the size of objects that belong to objects that are not stones before and after imposing a maximum and a minimum size. Most of the objects have a size really small and there are few that are really big. There are so few in comparison, that they cannot be seen in the histogram. After imposing the range in which objects will be selected to classify, around a 30% of the amount of objects that are not stones

are not considered, obtaining objects that have sizes more similar to the size of the stones and discarding those that have sizes too big or too small.

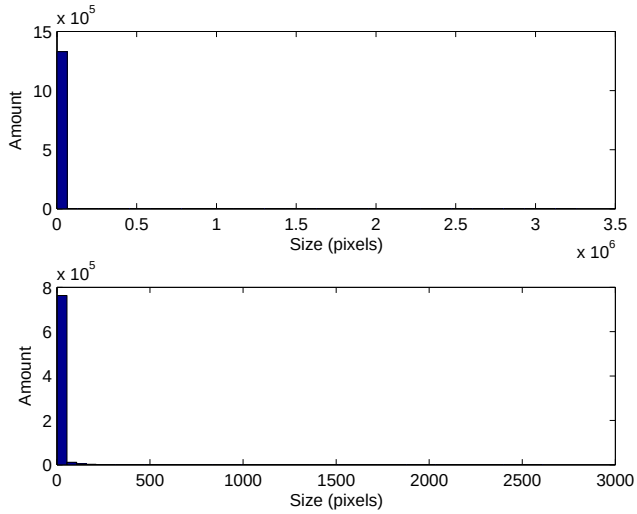


Figure 3.7: Histogram of size of objects that are not stones in pixels (up) and histogram of size of objects that are not stones inside the size range used (down).

3.5 Centroid estimation

Third step consists on calculating the centroid of each of the connected components identified (the ones that have a volume between the thresholds used). The centroid is calculated so the regions of interest (volume around the centroid of the component) can be identified as stones or no stones. Moreover, if one of these objects is classified as stone, the position of the stone is going to be known directly (the centroid), and additional calculus is not going to be needed to find the position of the stone.

The centroid of a volume is the average position of all the points in the volume. It can be calculated using the coordinates of all the points in the region by:

$C = \sum_{(i=1)}^{K} \frac{x_i}{K}$, where C is the centroid of the volume, K is the number of pixels in the volume and x_i is the position of a given point in a global coordinates frame.

3.6 Normalization

The pixels in the CT scans have intensity values between -1024 and 3072. The regions that are selected as possible stones are normalized with values between 0 and 1, so the classifier has a better accuracy. To do it, the maximum and minimum in every region is calculated and the values are scaled according to these values:

$$\mathrm{Normalized}_{value} = \frac{value - value_{min}}{value_{max} - valuemin}$$

3.7 Convolutional Neural Network

The Convolutional Neural Network is the main subsystem of the system and it is the one that requires more computational power and memory. It is a modified version of ANNs that extracts features from images by learning which are the interesting ones and it uses them to classify the different images. The main difference between an ordinary neural network and a Convolutional Neural Network is that the neurons in a layer are only connected to a specific amount and region of neurons in the previous layer, instead to be connected to all the neurons in the previous layer. The Convolutional Neural Network includes a loss function or error function, used to learn the desired parameters (weights and biases) in the structure, as well as a Softmax classifier.

3.7.1 Background

Convolutional Neural Networks are biologically-inspired algorithms of Machine Learning. The visual cortex contains a complex distribution of cells. The cells are sensitive to some small regions of the visual field, which all together cover the entire visual field. The cells act as local filters over the input space to exploit the spatial correlation in images.

There are two basic types of cells: Simpler cells respond to specific patterns in a small receptive field, while complex cells respond to larger fields and are locally invariant to the position of the pattern. The power of the animal processing system is the main reason to emulate its behaviour.

3.7.2 Advantages

Most classification techniques and algorithms do not use the spatial distribution of the input data and they relate all the data in the same way. They do not relate pixels that are far or close in a different way. In the present work, in which the input data is coming from images and there is a relation

between close pixels, taking into account proximity between pixels is an advantage. Convolutional Neural Networks take advantage of the spatial structure of the images. This makes Convolutional Neural Networks faster to train than ANN's, since the spatial relation is already included in the structure of the neural network: It does not need to be learned.

Another important advantage of convolutional neural networks is that they have much fewer parameters to train than an Artificial Neural Network for big input data, since the amount of parameters depend mostly on the amount and size of the filters used and not in the amount of input data.

Some of the existent literature extracts features from the input and builds a classification algorithm based on these features. These features are normally quite arbitrary and difficult to justify (even if they can work in some applications). Convolutional Neural Network let the classification algorithm to learn about the interesting features for every application by training them.

Convolutional Neural Networks are usually used to detect features in images. They have some characteristics that make them especially useful for this purpose. The first one is that CNN are translational invariant: this means that the result of a given local distribution of pixels is the same in any global position of the image. The second important characteristic is compositionality: this means that the different detected features can be composed together to form high-level features.

3.7.3 Structure

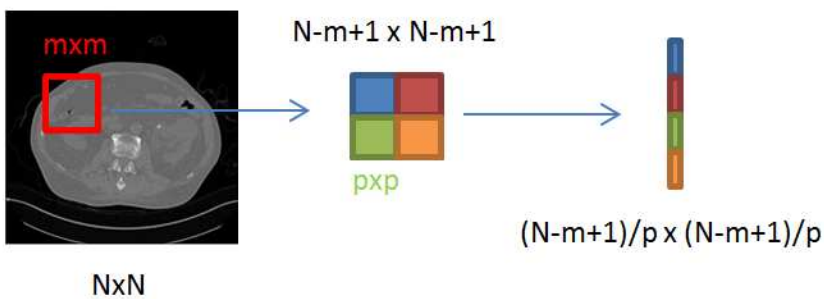


Figure 3.8: Example of convolution layer and pooling layer in two dimensions with an input of size N, one filter of size m and a pooling layer of size p.

An artificial neural network is a model inspired by biological neural networks used to approximate functions. They are made of neurons that have learnable weights and biases. Each neuron in the network receives some inputs, multiplies them by a weight followed by an optional non-linear transformation (activation function). The neurons are structured in layers and they can be classified as input nodes (input variables), hidden nodes and output nodes. In an ordinary neural network, all the nodes or neurons in one layer are connected to all the nodes in the following layer, while in a CNN each neuron is only connected to a small set of neurons in the previous layer.

A Convolutional Neural Network is based on a set of layers of three types: convolutional layers, pooling layers and fully connected layers (ordinary neural network). The first two types of layer alternate as many times as it is desired in the convolutional neural network and the last layer is a fully connected layer. Convolutional layers and fully connected layers perform different operations depending on the values of the weights and biases, while a pooling layer executes always the same function, so no parameters have to be learned. In figure 3.8 it can be seen the size result of applying a convolutional layer and a pooling layer to the input data in two dimensions.

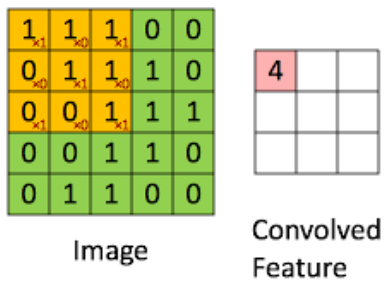


Figure 3.9: Example of convolution [3].

The convolutional layer is based on a set of filters that are used to convolve the input. The filters use to be small compared with the size of the input, but they are extended through all the input. The filter is slide along the input by computing the product between the values in the filter and the input. As a result of the convolution, a high value is obtained when a given distribution is found in a particular position of the input, according to the values of the filter. The network will learn filters that have a distribution that match with the features to be detected. This architecture takes into account the local

distribution of the data, since each neuron is connected to a specific amount of inputs which occupy close positions.

One of the characteristics of the convolutional layers are that the dependencies between input variables are supposed to be local, since only close input data is used at the same time. The operation is translation invariant too, since given a set of values in a region of the input the output of the convolution does not depend on the global location of the input values.

The convolutional layer incorporates usually a non-linear function after adding the bias, which usually consists on hiperbolic tangent function, sigmoid function or RELU function [17]. The purpose of applying a non-linear function at the output of the convolutional layer is that the output of the convolutional layer can adapt and can approximate better non-linear functions.

There are some hyperparameters that control the size of the output of a convolutional layer and the amount of parameters that need to be learned. The first hyperparameter is the size of the filter or size of the receptive field of the neuron, (F). The second designing decision is to set the dimension of the filters used (D). This value use to be two in 2D images, but it will be three since we are using 3D images. The third hyperparameter is the number of filters (N) used or number of neurons that are connected to the same region of the input. The forth hyperparameter is the stride (S). The stride is the displacement between the inputs of two neural networks that are the closest they can be. In other words, it is the displacement of the filters while multiplying its values by the inputs. When ordinary convolution is performed, the stride is one, since the filter is displaced one unity while performing convolution.

Normally, the convolution of a filter is followed by the application of a bias, which modifies the value obtained while doing convolution by adding a constant. The number of parameters to be learnt (P) can be calculated as the amount of filters multiplied by the size of the filter raised to the dimensionality of the filters plus one:

$$\mathsf{P} = \mathsf{N}(\mathsf{S}^{\mathsf{D}} + 1)$$

The number of outputs in a convolutional layer depends on the number of filters (N), the size of the filters (F), the size of the input (W), the dimensionality (D) and the stride (S). Each filter operates over all the input data, giving a set of outputs (V). The size of the output can be calculated as:

$$\mathsf{V} = \mathsf{N}((\tfrac{\mathsf{W}-\mathsf{F}}{\mathsf{S}} + 1)^{\mathsf{D}})$$

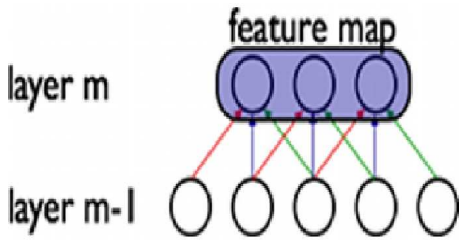


Figure 3.10: Example of convolution layer in one dimension with one filter of size three [1].

The pooling layer is a non-linear subsampling layer. The main purpose of this layer is to reduce the amount of parameters in the system and grouping more important data in fewer variables. The pooling process is applied independently to the outputs of each filter, so pooling does not relate output data coming from the convolution executed by different filters. Pooling partitions the data obtained in the convolutional layer in a set of regions (overlapping or non-overlapping) and gets a single value from each of them. Typical values pooled are mean or maximum.

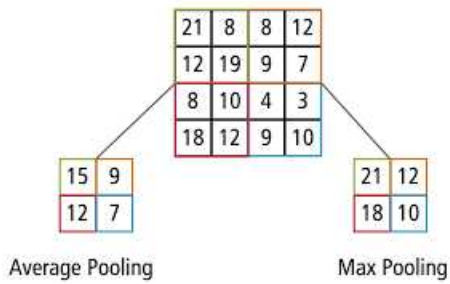


Figure 3.11: Example of pooling [2].

The input to the pooling layer is the output from de convolutional layer, so a set of N blocks of data coming from the N filters in the convolutional layer. Each block has $(\frac{W-F}{S}+1)^D$ variables. These blocks of data are divided in regions of the same size (F_2). The hyperparameters included in the pooling layer are the size of the regions (F_2) and the stride, the displacement between

one region and the next one (S_2). If the regions are non-overlapping, the stride and the size of the regions have the same value. If the regions are overlapping, the stride is smaller than the size of the regions.

The output of the pooling layer is a set of N blocks of data that contain $\frac{\frac{V}{N}-F_2}{S_2}+1$ values. The pooling layer does not add any parameter to the neural network, since it performs a fixed operation that does not depend on any parameter. The mean or the maximum are typical values obtained from every region of data.

The last layer is a Softmax classifier, this is an ordinary artificial neural network, which has as an input the output of the last pooling layer and as an output a prediction estimation for each class (in this work classification between stone/no stone).

Mathematical model

Even if the CNN designed in this work uses images and filters of 3D, this subsection will be explained as if they would have 2D for simplicity. Suppose that we have a NxN input to a convolutional layer and we use mxm weight filter of sparse equal to one. The output of the convolutional layer will be of size $(N-m+1)\times(N-m+1)$ for every filter. In order to compute the output of the layer l it is needed to sum up the contribuitons weighted by the filter components from the previous layer:

$$x_{ij}^{l} = \sum_{c=0}^{m-1}\sum_{d=0}^{m-1} w_{cd} y_{(i+c)(j+d)}^{l-1} + I_l$$

where l is the layer, y^{l-1} is the output of the previous layer, I is the bias of the filter, w_{cd} is all of the weights in the filter and x_{ij} is the result of applying the filter to the input data.

Next, the non-linear operation is applied:

$$y_{ij}^{l} = \sigma(x_{ij}^{l})$$

where y_{ij} is the output of the convolutional layer and σ is the activation function used.

The pooling layer, if we suppose that is non-overlapping, just partitions the data into squares and calculates a value from each of them.

In the fully connected layer, the forward propagation algorithm is quite simple. The first step is to compute the input to each neuron:

$$x_i^l = \sum_j w_{ji}^{l-1} y_j^{l-1} + I_i^l$$

Then the output of the neuron can be calculated:

$$y_i^l = \sigma(x_i^l)$$

The backward propagation process can be found in chapter 6

3.7.4 Training

In supervised learning, a set of examples are given to the artificial neural network in order to approximate the function that matches the input with the desired output. To approximate all the parameters that are used in the neural network, a cost function is used to minimize the error in the output. Gradient descent is used to implement a back-propagation algorithm to adjust the value of the parameters in the network (the weights).

3.7.5 Computational considerations

The bottleneck when building CNN architectures is the memory. There are several major sources of memory to worry about:

- The parameters used by the filters (the weights).
- The activations (outputs of every layer) in every iteration of the training process and their gradients while implementing backpropagation.
- Input data to train the network.

3.7.6 Choosing hyperparameters

The CNN have more hyperparameters to tune while training than an ANN. The first important hyperparameters are the number of filters and their size. Computing the activations of a single convolutional layer is much more expensive than in a traditional neural network. Assume a layer $(l-1)$ contains K_{l-1} feature maps and the size of each feature map is M. The layer (l) has K_l filters of size m. Then, when computing the feature map at a layer l costs a memory of $(M-m)^D m^D K_l$. The objective is to find a level of granuality which create abstractions at the proper scale without using too many parameters.

Another important hyperparameter when designing CNN is the pooling shape. If the size of the pooling regions are small, the amount of information that is thrown away is small, so the reduction of parameters is small too. If the pooling regions are two big, the reduction of parameters is big too, but the amount of information lost can be critical.

Chapter 4
Experimental results

In this section the experimental results of the CNN are going to be over viewed. The first step will be to select the best combination of parameters in the CNN in order to get the best configuration. Then, the results of the classification are going to be analysed and they will be compared with different types of ANNs which use previously extracted features from the CT scans and others that use the pixels directly.

4.1 Design of the CNN

A volume around each centroid of objects that are classified is obtained as input for the CNN. Therefore, for every CT scan, there are several examples of connected components that are used to train the network, most of them are not stones and the stones are replicated with small differences to obtain more examples as explained in section 3.1. The centroid of every object that might be a stone is calculated from the binarized image. However, the input for the CNN is the original resampled image, since it contains much more information (organs can be seen on it and the distribution of intensity on the pixels can be different around the shape).

There are several hyperparameters that have to be chosen for the CNN. The first hyperparameter that has to be chosen is the amount of convolutional layers and pooling layers, the general structure of the CNN. The comparison is going to be made taking into account that the amount of input features in the softmax classifier should be the same. Hence, if more layers are used, they will have filter's and pooling's size smaller than if less layers of each type are used. The advantage of using several convolutional and pooling layers alternated in front of using one layer of each is that the amount of parameters in the neural network will be fewer. The problem of using several convolutional and pooling

layers in front of using one of each type is that the amount of memory that needs to be used while backpropagation is bigger. Since the main bottleneck of the application is the amount of memory used by the training process, it has been chosen one layer of each type.

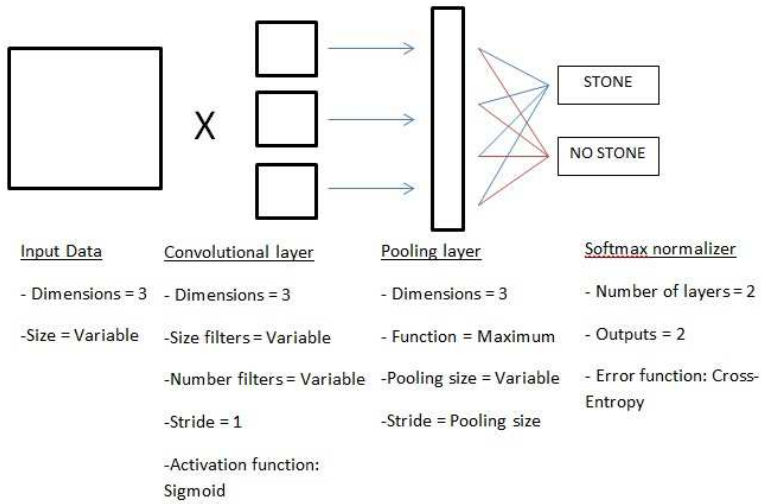


Figure 4.1: CNN design parameters.

The second decision that has to be taken is the activation function that is going to be used in the convolutional layer. The activation function is the source of the neural networks power. The selection of the activation function has a huge impact on the network performance. In [20] they give a quantitative comparision of different commonly used activation functions over ten different datasets. The results show that the sigmoid function outperforms the other activation functions when regarding error, while the linear activation functions are much faster because its derivative is easily calculated. The sigmoid function is defined as:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

It takes a real number and it saturates it into a range between 0 and 1. It has a direct relation with the firing in a neuron: from not firing at all to a fully-saturated firing. However, it has a major drawback: When the output of the function is close to zero or one, the gradient is almost zero. During back propagation, the derivative of the activation function is multiplied by the gradient of the error of the output. In case the derivative of the activation

function is close to zero, it will set the multiplication of both terms to a really small number, and almost no signal will flow through the network. All in all, it has been the activation function chosen, since the training time is not a major issue in the system design and the gradient is only close to zero when the firing or no firing of the neuron has been learnt after several examples (The initialization of the parameters is random around 0).

The third parameter that is chosen is the type of pooling that will be used by the pooling layer. The purpose of the pooling layers is to achieve spatial invariance while reducing the resolution of the feature maps. In [19] they compare different pooling operations on fixed architectures. Their results show that maximum pooling operation significantly outperforms subsampling operations for capturing invariances in image-like data. Their empirical results are based on several datasets. The maximum pooling layer takes the maximum from a $n \times n$ patch and selects and pools the maximum:

$a_j = max_{nxn}(a_i)$

where a_j is the ouput of the patch and a_i are the values contained in a nxn patch.

The forth parameter that must be chosen is which function will be used to calculate the error of the classification and compute the gradient while executing backpropagation. In [5] they investigate the error criteria that optimize training in artificial networks. They compare squared error and cross-entropy criteria. Their results show that when initializing the weights of the network randomly, the cross-entropy error criteria performs better. In practice, cross-entropy converges faster and gives better results in terms of classification error rates. The cross-entropy function is estimated as:

$h = -\sum_{i=1}^{N} \frac{1}{N} log(x_i)$

where x_i is the probability associated to the correct output of the network.

The fifth and sixth hyperparameters that have been chosen is the stride in the convolutional and pooling layer. A higher stride in the convolutional layer than one supposes that the filters are not convolving each cube that could be convolved, so the position where the filter would have a higher value could be missed and the position where a feature can be identified too. So, the stride in the convolutional layer has been set to one.

As the stride in the pooling layer decrease, some of the values got by the convolutional layer are used more than once. This configuration can have sense if the mean pooling is used, since when using mean-pooling in non-overlapping

regions, each value is used once and the mean could change radically if a small displacement in the region would be performed. However, since the operation performed by the pooling layer is to pool only the maximum of the region, it is a spatial invariant operation, and, any value that is a maximum in all the regions will be pooled. So, the most important detected features will be pooled anyway and it does not have any sense to pool twice the same features.

Another important hyperparameter to be chosen is which number of layers will have the fully-connected network. If the data is linear separable, it is not necessary to use a hidden layer. This assumption cannot be done, since the features extracted from the convolutional and pooling layers are unknown when the design of the network has been done. However, if one or more hidden layers are introduced in the fully-connected network, the number of parameters in the network increases and the computational cost and memory used when performing backpropagation. The main purpose of the method used is to extract the most useful features using the convolutional neural network. This is the main reason that justifies to use a fully-connected network without hidden layers: Concentrate the available resources in the convolutional and pooling layer.

The remaining hyperparameters that need to be chosen are the number of filters, the size of the filters, the size of the pooling regions and the size of the images that are going to be use around the centroid of each of the connected components that are going to be classified. These parameters are quite arbitrary and difficult to set, since the images used are 3D images and there is not much existing research in this field. To obtain the four parameters there have been done a set of simulations in which the parameters take different values in order to compare the different results and chose a combination that performs as good as possible. The performance measures are the accuracy of the network, the sensitivity when classifying stones and the time used to train the networks. These performance measures have been chosen for different reasons. The accuracy of the network has been chosen because it is the main goal in a classifier. The sensitivity in the stones has been chosen because it is a critical result in the work that has been developed, since it is much more critical to classify a stone as a no stone than to classify an object that is not a stone as a stone. The training time has been used to compare combinations of parameters that have similar performance but with different number of parameters and difficulty to train.

The set of trainings has been done with data coming from 147 patients out of the total amount of patients, since it was the available amount of data when the set of trainings was developed. All the patients have one stone. As it has been explained before in 3.1, the amount of data is not large enough to train

the network, since for every patient there is only one example of stone. So, the available stones have been copied several times by turning them around and translating them, as described in section 3.1.

The results of the set of trainings can be seen in chapter 6. The time used to train the networks is not a critical variable and it is not very different from one configuration to another (The maximum difference between two configurations is ten times more the time needed to train the network). The configurations selected as best are those that succeed well in accuracy and in sensitivity among stones. Therefore, the configurations chosen are those in which the total accuracy is higher than 0.97 and, at the same time, the sensitivity among stones is higher than 0.98. These configurations will be compared next with a bigger dataset as input to select only one of them.

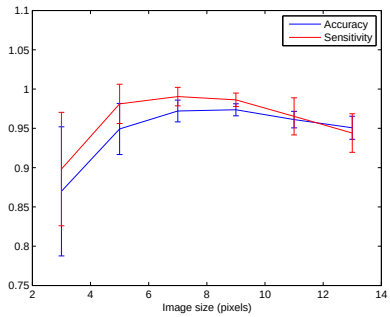
In figure 4.2a it can be seen the mean and standard deviation of the accuracy and the sensitivity of the training set that has been developed with different values for the image size. The mean of the accuracy and sensitivity is higher when using an image size of seven or nine pixels, but the standard deviation is minimum when using an image size of nine pixels. If the image size is smaller, the information in the image is not enough to predict with enough accuracy if the object is a stone or not and it has not enough sensitivity. If the image size is bigger than nine pixels, the mean accuracy and the sensitivity are a little smaller while the standard deviation grows, since there are many pixels in the image, some of them do not belong to the object itself, but the surroundings.

In figure 4.2b it can be seen the mean and standard deviation of the accuracy and sensitivity of the training set that has been developed with different values for the number of filters used. The mean of the accuracy and sensitivity are quite constant for all the values, but the standard deviation grows as the number of filters grows. Therefore, a smaller number of filters ensures a better performance.

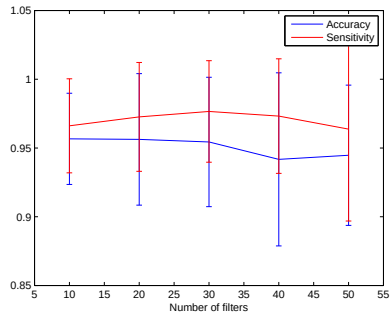
In figure 4.2c it is shown the mean and standard deviation of the accuracy and sensitivity of the training set that has been developed with different values for the filter size. The mean of the accuracy and sensitivity is higher when using a big enough filter size and the standard deviation drops down as the filter size grows. Hence, there is a minimum in the filter size so the CNN performs with high accuracy and sensitivity.

In figure 4.2d it is shown the mean and standard deviation of the accuracy and sensitivity of the training set that has been developed with different values for the pooling size. The mean of the accuracy and sensitivity are bigger

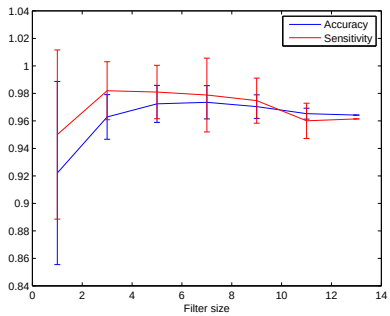
when the pooling size is really small, since all the features are pooled and no information is lost, but it grows again when the pooling size is big, since only the most important features are pooled. The standard deviation tends to be smaller as the pooling size is bigger, due to the selection and use of the best features.



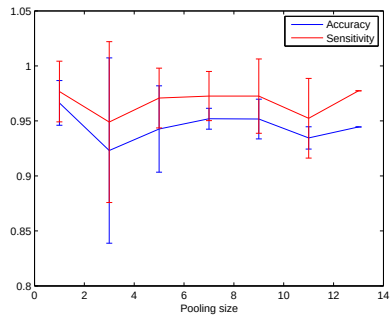
(a) Accuracy (mean ± standard deviation) vs image size.



(b) Accuracy (mean ± standard deviation) vs number of filters.



(c) Accuracy (mean ± standard deviation) vs filter size.



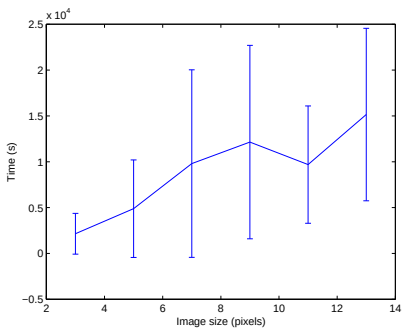
(d) Accuracy (mean ± standard deviation) vs pooling size.

Figure 4.2: Performance of different configurations.

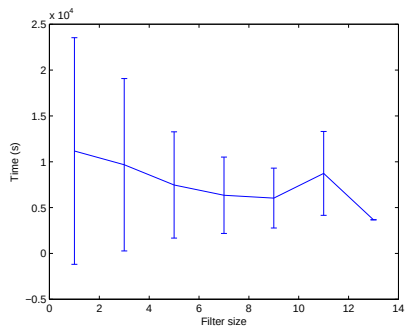
In figure 4.3a it can be seen the mean and standard deviation of the time of the training set that has been developed with different values for the image size. The time is smaller and has low deviation when using small images, since the filters are small too. As the image size grows, the mean and the

standard deviation of the time used to train the CNN grows, since some of the simulations use bigger filters and there are more parameters to learn.

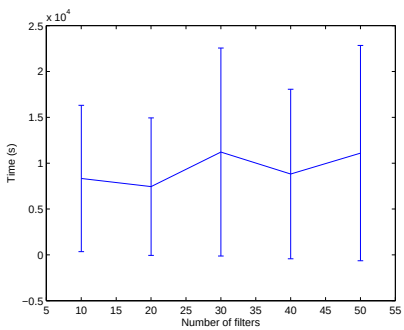
In figure 4.3b it shown the mean and standard deviation of the time of the training set that has been developed with different values for the filter size. The time is smaller when using a filter size bigger, since the number of parameters to learn in the softmax normalizer are few. The time is bigger when the filter size is smaller, since there are a lot of parameters to learn in the softmax normalizer.The standard deviation drops down when using bigger filter size because the amount of possible pooling size are less, so the amount of simulations has been less, and its values closer.



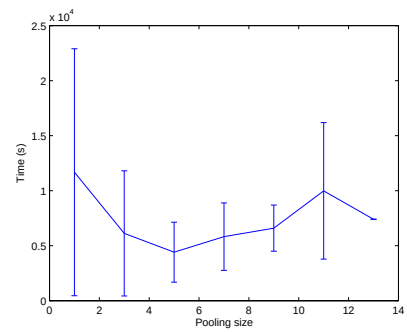
(a) Training time (mean ± standard deviation) vs image size.



(b) Training time (mean ± standard deviation) vs filter size.



(c) Training time (mean ± standard deviation) vs number of filters.



(d) Training time (mean ± standard deviation) vs pooling size.

Figure 4.3: Training time of different configurations.

In figure 4.3c it can be seen the mean and standard deviation of the time of the training set that has been developed with different values for the number of filters used. The standard deviation of the time grows as the amount of filters used is bigger, but the mean of the time is quite constant.

In figure 4.3d it is shown the mean and standard deviation of the time of the training set that has been developed with different values for the pooling size. It shows us that the mean and standard deviation of the training time is smaller when using a medium pooling size since the amount of features in the softmax classifier is not too big, but neither too small.

4.2 Performance

In this section, the best configurations of the CNN found in section 4.1 will be trained with a bigger dataset in order to obtain their performance and select the best. The different configurations will be compared based on their accuracy, sensitivity and training time, but the best configuration will be selected based on the F1 score. The F1 score is a performance measure that combines both, the sensitivity and the precision. It is used because the sensitivity is a critical parameter in this work, but a big amount of false positives is bad since it is difficult to known which ones are real positives and it has to be checked by another algorithm or manually. The best configuration will be selected to effectuate the validation and comparison with other methods.

In summary, the configurations that have been found to have a good performance in section 4.1 are compared in this section to select only one of them and compare it to other methods.

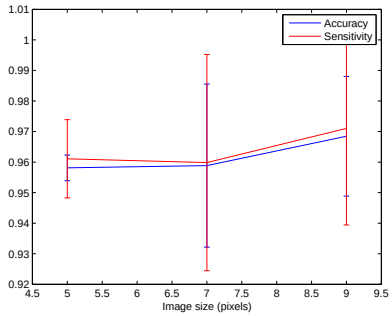
4.2.1 Experimental setup

The dataset consists of 600 CT scans of patients, 460 of them have a urethral stone and 140 do not have a urethral stone but 70 of them have a kidney stone. 140 patients of each type (with a urethral stone and without urethral stone) have been selected for validation, while the rest have been used to train and test the system. The 80% of the available examples used to train and test have been used to train the network, while the 20% have been used to test it.

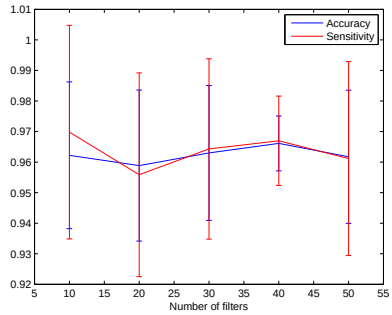
As it has been explained previously 3.1, the amount of examples of stones is not big enough, so they have been copied with slightly modifications (rotations and translations) to have a bigger amount of examples to train the convolutional neural network.

4.2.2 Comparison

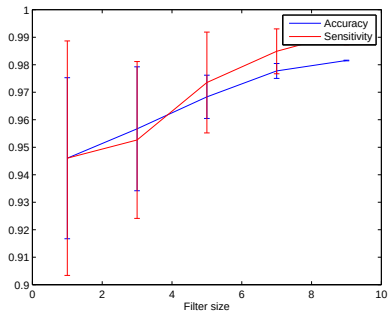
The results obtained can be found in chapter 6. In figures 4.4 and 4.5, it can be found the summary of the results obtained.



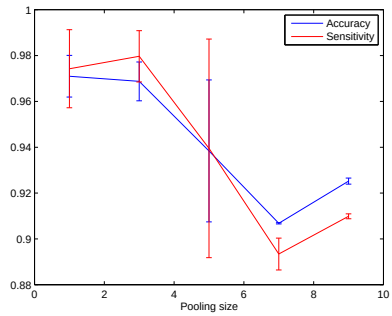
(a) Accuracy (mean ± standard deviation) vs image size.



(b) Accuracy (mean ± standard deviation) vs number of filters.



(c) Accuracy (mean ± standard deviation) vs filter size.



(d) Accuracy (mean ± standard deviation) vs pooling size.

Figure 4.4: Performance of best configurations.

In figure 4.4a it can be seen the mean and standard deviation of the accuracy and sensitivity of the training set that has been developed with different values for the image size. The mean of the accuracy and sensitivity are higher when using a bigger image size, since there is more available information about the object. The standard deviation is lower when using a smaller input image since there is less information about the objects and the most important

features, in the middle of the connected components, are used in any configuration.

In figure 4.4b it is shown the mean and standard deviation of the accuracy and sensitivity of the training set that has been developed with different values for the number of filters used. The mean and standard deviation of the accuracy and sensitivity are quite constant but they have the best mean and standard deviation when using forty filters.

In figure 4.4c it can be seen the mean and standard deviation of the accuracy and sensitivity of the training set that has been developed with different values for the filter size. The mean of the accuracy is higher and the standard deviation lower when using a bigger filter because more data is used to detect features, therefore, the features used are based on more information.

In figure 4.4d it is shown the mean and standard deviation of the accuracy and sensitivity of the training set that has been developed with different values for the pooling size. The mean of the accuracy and sensitivity is higher when using a smaller pooling size, since all the features are used. The standard deviation is higher when using a medium pooling size and it drops down when using a big pooling size, hence all the important features are pooled and the performance is quite constant and does not depend in all the other parameters.

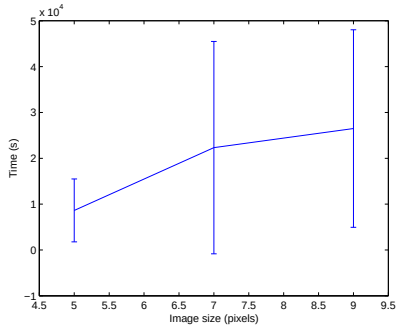
In figure 4.5a it can be seen the mean and standard deviation of the time of the training set that has been developed with different values for the image size. The time is higher when using a bigger image size, since there is more data to compute. The standard deviation is higher when using bigger images, since the values for the other parameters have a wider range.

In figure 4.5b it is shown the mean and standard deviation of the time of the training set that has been developed with different values for the filter size. The time is smaller when using a filter size bigger, since the number of parameters to learn in the softmax normalizer are few.

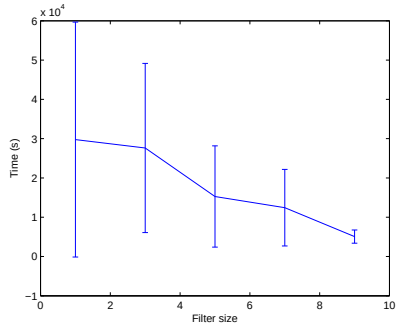
In figure 4.5c it can be seen the mean and standard deviation of the time of the training set that has been developed with different values for the number of filters used. The results show that the mean and standard deviation of the training time grow as the number of filters is higher, which is quite related with the amount of parameters that the CNN has to learn.

In figure 4.5d it is shown the mean and standard deviation of the time of the training set that has been developed with different values for the pooling

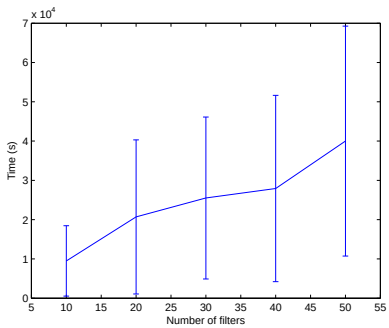
size. The time is smaller when using a smaller pooling size, since there are less parameters that have to be learn by the CNN because there are less features that are used in the classification.



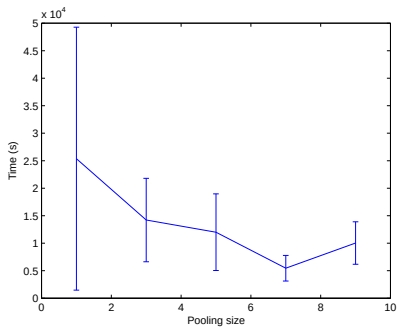
(a) Training time (mean ± standard deviation) vs image size.



(b) Training time (mean ± standard deviation) vs filter size.



(c) Training time (mean ± standard deviation) vs number of filters.



(d) Training time (mean ± standard deviation) vs pooling size.

Figure 4.5: Training time of best configurations.

In table 4.1 it can be seen the F1 score of the different configurations that have been trained. Based on the F1 score, the best configuration uses images of size equal to 9 pixels, 10 filters of size 7 and the pooling size is 1.

Table 4.1: Comparison based on the F1 score.

Image size (pixels)	Filter size	Pooling size	Number of filters	F1 score
5	1	1	10	0,953
5	3	1	10	0,960
5	3	1	20	0,952
5	3	1	30	0,959
5	3	1	40	0,952
5	3	1	50	0,957
5	5	1	10	0,964
5	5	1	20	0,958
5	5	1	30	0,963
5	5	1	40	0,963
7	1	1	10	0,975
7	1	1	20	0,974
7	1	1	40	0,966
7	1	1	50	0,972
7	1	7	10	0,905
7	1	7	20	0,906
7	1	7	30	0,906
7	3	1	10	0,977
7	3	1	20	0,975
7	3	1	30	0,968
7	3	1	40	0,977
7	3	1	50	0,975
7	3	5	10	0,911
7	3	5	20	0,912
7	3	5	50	0,913
7	5	1	10	0,976
7	5	1	20	0,975
7	5	1	30	0,971
7	5	1	40	0,968
7	5	1	50	0,978
7	5	3	10	0,959
7	5	3	40	0,958
7	7	1	10	0,978
7	7	1	20	0,979
7	7	1	30	0,978
7	7	1	40	0,979
7	7	1	50	0,973
9	1	1	10	0,981
9	1	1	20	0,983
9	1	9	10	0,922
9	1	9	20	0,924
9	1	9	30	0,925
9	3	1	10	0,978
9	3	1	20	0,965
9	3	1	30	0,972
9	5	1	20	0,958
9	5	1	30	0,980
9	5	5	10	0,976
9	5	5	30	0,977
9	7	1	10	0,983
9	7	1	30	0,978
9	7	3	10	0,977
9	7	3	20	0,976
9	7	3	30	0,975
9	9	1	10	0,982
9	9	1	20	0,982

4.2.3 Data augmentation effect

In this subsection, the optimal configuration of the CNN has been trained with the original data, without the data augmentation process. The objective of this test is to check if the data augmentation process is useful and how much does it change the results. Since there is only one example of stone in every CT scan and the amount of classes of each type (stone/no stone) must be balanced, there have been used 320 examples of stones and 320 examples of objects that are not stones but they might be.

The results of the training process with no data augmentation show that the classifier has an accuracy of 76%, a sensitivity of 75%, a F1 score of 76% and the training time has been 376 seconds. In comparison to the CNN trained with the data augmentation process, the training time is much smaller, since the training set is much smaller. Nevertheless, the performance of the classifier is worse, which means that the CNN didn't have enough examples of each class without the data augmentation process.

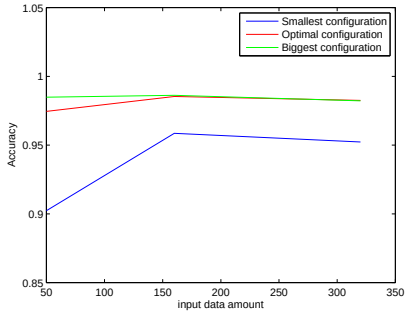
4.2.4 Training set size effect

In this section, it is going to be studied the consequences of changing the training set size for different sizes of configuration of the CNN. The training set has been modified by using 50, 160 and 320 CT scans as input data. The configurations in which the effect is studied are a small configuration, with few parameters to learn, the optimal configuration and a big configuration, with many parameters. The small configuration uses images of five pixels size, ten filters of size one and pooling size equal to one. The big configuration uses images of nine pixels size, twenty filters of size nine and pooling size equal to one.

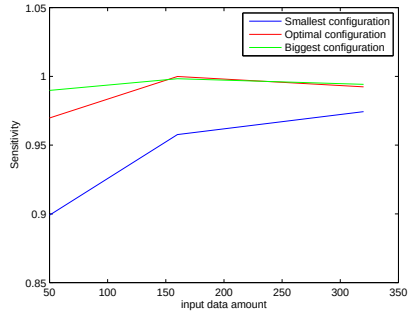
In figure 4.6 it is shown the relation between the accuracy, sensitivity, F1 score and training time of the CNN depending on the size of the CNN and the size of the training set.

Figures 4.6a, 4.6b and 4.6c show the accuracy, sensitivity and F1 score of the three different configurations, changing the training set size. The maximum accuracy, sensitivity and F1 score is reached when 160 samples are used to train the classifier, obtaining approximately the same results when 320 samples are used. The big and the optimal configuration have the same accuracy, sensitivity and F1 score when the training set is big enough, but the big configuration behaves better if the training set is not big enough. This means that the amount of samples is enough to train the CNN, but the smaller configurations do not have enough parameters to perform so well, even if the input is small.

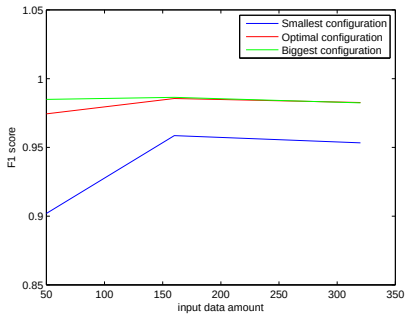
In figure 4.6a it can be seen the time used to train the CNN of the three different configurations, changing the training set size. The time necessary to train the CNN is directly related to the training set size and the amount of parameters in the CNN: the smaller configurations take less time to train for any size of input data and as the training set size is increased, the time necessary to train the CNN is bigger.



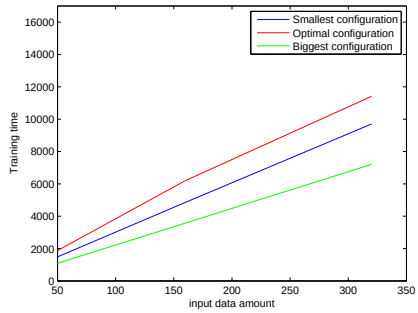
(a) Accuracy vs training set size.



(b) Sensitivity vs training set size.



(c) F1 score vs training set size.



(d) Training time vs training set size.

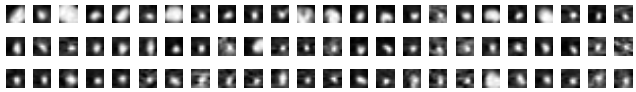
Figure 4.6: Performance vs training set size.

4.2.5 False positives

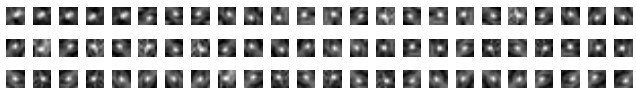
In this subsection, it is going to be overview the relation between the false positives and the real stones. It is going to be studied graphically, to check if there are stones that look like the false positives that have been obtained. Since the image is a 3d image, the piece of slice that passes through the centroid of the stone and the false positives is going to be shown. Therefore there is only available information about two dimensions of the stone.

Figure 4.6 shows different examples of stones (up) and examples of false positives obtained by the classifier. As it can be checked, there are some examples of stones that look like most of the false positives, having a pixel intensity

distribution close to each other. Therefore, it will be difficult to prune the false positives if no anatomical information is used.



(e) Examples of stones.



(f) Examples of false positives.

Figure 4.6: Examples of stones and false positives.

4.3 Comparison with other methods

To compare the system designed with the state of the art and other systems that use the same data to succeed when performing stones classification, a comparison has been done. This comparison consists on training an ANN with the features extracted in [14] and [12] and with a ANN trained with the pixels intensity as input data. When detecting stones directly from the pixels, the data introduced in the ANN are the intensity values of the cubes that around the center of objects that are stones and others that are not, with the same amount of each class. The size used to train the ANNs are 5, 7 and 9 pixels.

The features extracted in [14] are divided in shape and texture features. The texture features include mean and standard deviation of intensity values,local binary pattern and histogram of oriented patterns. The shape features include the volume of the candidate and two aspect ratios of height/length and width/length of the candidate. In the simulation carried out, only 6 features have been included. These features are mean and standard deviation of intensity values,local binary pattern, the volume of the candidate and two aspect ratios of height/length and width/length of the candidate

The features extracted in [12] are divided in shape features and texture features. The shape features include Disperseness, convex hull depth and lobulation. The texture features are Edge intensity, skewness, difference histogram variation and gray-level co-occurrence. The simulation carried out includes all

of them, except lobulation, since there is not enough information about this feature to use it.

In 4.2 there can be seen the comparison of the different methods developed in this work and the existing methods.

Method	Accuracy	Sensitivity	F1 score	Training time (s)
SVM 7 features [14]*	-	0,69	-	-
ANN 8 features [12]*	0,88	-	-	-
ANN 6 features from [14]	0,985	0,975	0,984	288812
ANN 7 features from [12]	0,773	0,796	0,778	24875
ANN 5 pixels	0,930	0,924	0,930	101
ANN 7 pixels	0,965	0,969	0,965	151
ANN 9 pixels	0,969	0,976	0,969	443
CNN	0,983	0,995	0,983	11421

Table 4.2: Comparison of the performance of different methods (* these methods have not been developed in this thesis.)

The results of the CNN are better than the results obtained in any other method. The Accuracy, sensitivity and F1 score is higher than in all the other methods, except when using a ANN with 6 features as in [14], in which only the accuracy and F1 score are lower but almost the same (comparable), all the other performance measures are worst in the ANN trained with features. Furthermore, the training time when using extracted features is much higher, since the features must be extracted before training the ANN. Moreover, the sensitivity in stones is lower, which is a critical measure.

4.4 Validation

The last step performed consists of the validation of the method developed in this thesis. The validation is carried out by testing the CNN with the configuration chosen (input images of 9 pixels, ten filters of size 7 and pooling size of one), trained previously by the training set. The process consists of taking the CT scans from different patients and check which and how many objects are detected as stones. All the connected components detected on each patient that accomplish the size conditions explained in section 3.4 are introduced in the CNN and they are classified. To detect the components the preprocessing steps explained in chapter 3 have been carried out. On each CT scan there are zero or one stones and thousands of connected components that are not stones, but this unbalance is not important since the CNN has been trained already.

The dataset consists of 140 CT scans from patients that have a urethral stone, 70 CT scans from patients that do not have any stone labelled and 70 CT scans from patients that have a kidney stone but it is not labelled.

Among all the CT scans used for validation, the sensitivity found has been 97,8%, and the accuracy 98,1%. There are an average of 92 false positives in each CT scan. This is an issue because even if the sensitivity is big, the precision is small, and the false positives need to be pruned by another way.

Chapter 5
Conclusion

The results of the project have been satisfactory, since the accuracy and sensitivity in the identification and classification of urethral stones are high.

The preprocessing steps used to reduce the amount of data input during the classification have been proven to be useful as the information contained in a CT scan is big and most of it is not useful to find stones. The development of these subsystems have accomplished the objective of using only the significant data: the pixels around the regions that might be stones based on the pixels intensity.

The data augmentation process has been useful and it has work properly. The amount of input data was not big enough to train the CNN. The processes of rotation and translation of the images have allowed to obtain much many examples of stones to train the CNN learn from them.

Among all the classification methods included in this work, the CNN has been the one that has the best performance, although it has two major drawbacks: the time spent and the necessary memory to train it.

The biggest issue in the results of the project are the significant high false positives that the system obtains, which means that the precision is really low, hence there are many connected components that have intensity values around the values given in the stones.

The use of a CNN in input images of three deep dimensions has been developed and performs successfully, even if it is not commonly used. The major drawback while using it is its increased amount of input data, which increases the amount of parameters that need to be trained in the CNN.

In summary, the algorithm developed is useful to analyse the data coming from the CT scans to find stones on them and identifying its position. The time spent to analyse a CT scan once the CNN has been trained is around seven minutes including the preprocessing steps, which is a saving in time for the expert radiologists.

5.1 Future work

The future work in stones classification based on CT scans should be focused on finding other compositions of CNN that outperform the one designed in this work. This includes changing its architecture, including several convolutional and pooling layers or changing the activation and error functions used, even they have been chosen by its properties. The future work should not be focused only on changing hyperparameters using the same structure because the search process has been deep and complete.

Another field of future research would be to use the position of the regions that are going to be classified, since this is useful information and the urethral stones are situated in a specific region of the body. It could reduce much more the amount of input data. The main problem about using these techniques would be to reference the position to a specific part or organ of the body, for example the spine, and not the position in the CT scan because the start and end positions of the CT scans in reference to the body can vary according to different variables. Registration process can be used to find the transformation to align the spines of many CT scans. The transformation can be applied to the position of the stones, obtaining the positions of the stone in reference to the spine. Then, the position of the stones of all the CT scans used in the training process can be saved and a region around all this positions can be isolated from the CT scan. Only the components inside this region should be selected for classification.

Since the use of CNN in the classification of stones with CT scans as input data have been proven satisfactory, this technique could be used in the identification and classification of other diseases.

Chapter 6
Appendix

6.1 Backpropagation in CNNs

The purpose of back propagation is being able to compute the error in the output to optimize the weights and biases to minimize the error. In order to make the network learn we must compute the derivatives to use gradient descent algorithm. The propagation is done from the output of the network to the input. So the first thing is to propagate the error back in the fully connected network. The derivative of the Error function (E) related to the weights in the fully layer is, applying the chain rule:

$$\frac{dE}{dw_{ij}^{l}} = \frac{dE}{dx_{j}^{l+1}} \frac{dx_{j}^{l+1}}{dw_{ij}^{l}}$$

Note that we only get contributions from the input of the next layer, since the weights are used nowhere else. From the equation of forward propagation ($x_i^l = \sum_j w_{ji}^{l-1} y_j^{l-1}$), we see that the partial with respect to any weight is the activation from its origin neuron, so:

$$\frac{dE}{dw_{ij}^{l}} = y_i^l \frac{dE}{dx_j^{l+1}}$$

We already know all the values of y, so we need to compute the partial with respect to the input x_j. We know that $y_i^l = \sigma(x_i^l) + I_i^l$ so using the chain rule:

$$\frac{dE}{dx_j^l} = \frac{dE}{dy_j^l} \frac{d(\sigma(x_j^l)+I_j^l)}{dx_j^l} = \frac{dE}{dy_j^l} \sigma'(x_j^l)$$

If we are in the output layer, then we know that the partial is just the derivative of the error function:

$$\frac{dE}{dy_i^l} = \frac{dE(y^l)}{dy_j^l}$$

While if we are not in the output layer, the partial is:

$\frac{dE}{dy_i^l} = \sum \frac{dE}{dx_j^{l+1}} \frac{dx_j^{l+1}}{dy_i^l} = \sum \frac{dE}{dx_j^{l+1}} w_{ij}$

As we can see, the error in a particular layer l is the weighted combination of the errors in the next layer (l+1).

In summary, to compute the gradient of the error with respect to the weight we must:

Compute the errors at the output layer:

$\frac{dE}{dy_i^l} = \frac{dE(y^l)}{dy_j^l}$

Compute the partial derivative of the error with respect to the input of the neurons:

$\frac{dE}{dx_j^l} = \frac{dE}{dy_j^l} \sigma'(x_j^l)$

Compute the errors at the previous layer:

$\frac{dE}{dy_i^l} = \sum \frac{dE}{dx_j^{l+1}} w_{ij}$

Compute the gradient of the error:

$\frac{dE}{dw_{ij}^l} = y_i^l \frac{dE}{dx_j^{l+1}}$

The pooling layer does not learn, since there is not any parameter in it. The error is just back propagated to the previous layer.

We have the gradient of the error function (E) at the output of the convolutional neural network, so we can compute the gradient related to the output of the previous layer ($\frac{dE}{dy_{ij}^l}$). We can compute the gradient related to the weights of the filters by applying the chain rule and suming the contributions of all the expressions were the variable is used:

$\frac{dE}{dw_{ab}} = \sum_{i=0}^{N-m} \sum_{j=0}^{N-m} \frac{dE}{dx_{ij}^l} \frac{dx_{ij}^l}{dw_{ab}} = \sum_{i=0}^{N-m} \sum_{j=0}^{N-m} \frac{dE}{dx_{ij}^l} y_{(i+a)(j+b)^{l-1}}$

We know that $\frac{d_x ij^l}{dw_{ab}} = y_{(i+a)(j+b)}^{l-1}$ from forward propagation equations.

In order to compute $\frac{E}{dx_{ij}^l} = y_{(i+a)(j+b)}^{l-1}$ we can use again the chain rule:

$\frac{dE}{dx_{ij}^l} = \frac{dE}{dy_{ij}^l} \frac{dy_{ij}^l}{dx_{ij}^l} = \frac{dE}{dy_{ij}^l} \frac{d\sigma(x_{ij}^l)}{dx_{ij}^l} = \frac{dE}{dy_{ij}^l} \sigma'(x_{ij}^l)$

We already know the error at the output layer, so we just have to multiply it by the derivative of the activation function used. Next step is backpropagate the errors back to the output of the previous layer using chain rule once more:

$\frac{dE}{dy_{ij}^{l-1}} = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} \frac{dE}{dx_{(i-a)(j-b)}^l} \frac{dx_{(i-a)(j-b)}^l}{dy_{ij}^{l-1}} = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} \frac{dE}{dx_{(i-a)(j-b)}^l} w_{ab}$

Looking at the forward propagation equations we know that $\frac{dx^l_{(i-a)(j-b)}}{dy^{l-1}_{ij}} = w_{ab}$. This looks like a convolution but instead of aplying the filter to $x_{(i+a)(j+b)}$we apply it to $x_{(i-a)(j-b)}$. This expression only makes sense for points that are at least m away from the top and left edges. If we do it, it is a simple convolution using w flipped both axes.

6.2 Performance of different CNN configurations

Table 6.1: Performance of different configurations (Part 1).

Performance of every configuration of the CNN (Part 1).

Size of image	Size of filters	Pooling size	Number of filters	Accuracy	Accuracy in stones	Training time (s)
3	1	1	10	0,923	0,934	1670
3	1	1	20	0,896	0,876	3125
3	1	1	30	0,913	0,925	4650
3	1	1	40	0,901	0,904	6193
3	1	1	50	0,905	0,924	7837
3	1	3	10	0,766	0,815	332
3	1	3	20	0,749	0,831	645
3	1	3	30	0,749	0,830	946
3	1	3	40	0,733	0,845	1261
3	1	3	50	0,786	0,729	1575
3	3	1	10	0,945	0,972	261
3	3	1	20	0,946	0,959	520
3	3	1	30	0,946	0,969	778
3	3	1	40	0,945	0,977	1061
3	3	1	50	0,944	0,982	1312
5	1	1	10	0,971	0,984	4280
5	1	1	20	0,965	0,993	8532
5	1	1	30	0,954	0,999	12930
5	1	1	40	0,968	0,989	17200
5	1	1	50	0,953	1,000	21443
5	1	5	10	0,883	0,952	593
5	1	5	20	0,900	0,924	1183
5	1	5	30	0,899	0,917	1777
5	1	5	40	0,879	0,953	2362
5	1	5	50	0,882	0,951	3086
5	3	1	10	0,976	0,998	2025
5	3	1	20	0,972	0,998	4042
5	3	1	30	0,970	0,999	6062
5	3	1	40	0,975	0,991	8078
5	3	1	50	0,973	0,998	10111
5	3	3	10	0,968	0,986	785
5	3	3	20	0,949	0,998	1570
5	3	3	30	0,929	1,000	2351
5	3	3	40	0,950	0,995	3211
5	3	3	50	0,952	0,937	3924
5	5	1	10	0,972	0,997	428
5	5	1	20	0,979	0,991	856
5	5	1	30	0,977	0,992	1282
5	5	1	40	0,974	0,990	1708
5	5	1	50	0,960	0,998	2181
7	1	1	10	0,975	0,982	8723
7	1	1	20	0,971	0,994	17458
7	1	1	30	0,966	0,995	26197
7	1	1	40	0,975	0,982	35664
7	1	1	50	0,972	0,990	45569
7	1	7	10	0,930	1,000	1977
7	1	7	20	0,949	0,951	2510
7	1	7	30	0,948	0,945	3761
7	1	7	40	0,951	0,999	5005
7	1	7	50	0,944	0,999	6255

Table 6.2: Performance of different configurations (Part 2).

Performance of every configuration of the CNN (Part 2).

Size of image	Size of filters	Pooling size	Number of filters	Accuracy	Accuracy in stones	Training time (s)
7	3	1	10	0,979	0,993	5575
7	3	1	20	0,983	0,988	11292
7	3	1	30	0,984	0,995	16724
7	3	1	40	0,978	0,996	22293
7	3	1	50	0,983	0,996	27908
7	3	5	10	0,972	0,998	1850
7	3	5	20	0,970	0,998	3710
7	3	5	30	0,966	0,998	5610
7	3	5	40	0,951	1,000	7386
7	3	5	50	0,985	0,995	9201
7	5	1	10	0,984	0,994	2848
7	5	1	20	0,984	0,988	5689
7	5	1	30	0,983	0,992	8527
7	5	1	40	0,985	0,992	11349
7	5	1	50	0,985	0,986	14211
7	5	3	10	0,976	0,991	1604
7	5	3	20	0,965	0,997	3205
7	5	3	30	0,970	0,996	4802
7	5	3	40	0,978	0,992	6402
7	5	3	50	0,977	0,976	8012
7	7	1	10	0,984	0,992	769
7	7	1	20	0,986	0,992	1577
7	7	1	30	0,980	0,995	2345
7	7	1	40	0,983	0,992	3084
7	7	1	50	0,971	0,996	3846
9	1	1	10	0,979	0,997	15133
9	1	1	20	0,979	0,993	31138
9	1	1	30	0,968	0,988	47047
9	1	3	10	0,965	0,991	7560
9	1	3	20	0,963	0,987	15109
9	1	3	30	0,964	0,979	23402
9	1	9	10	0,966	1,000	4398
9	1	9	20	0,969	1,000	4262
9	1	9	30	0,961	1,000	6559
9	3	1	10	0,983	0,992	12498
9	3	1	20	0,982	0,981	23667
9	3	1	30	0,981	0,989	35741
9	3	7	10	0,959	0,980	4288
9	3	7	20	0,961	0,962	8570
9	3	7	30	0,959	0,989	12859
9	5	1	10	0,981	0,975	6513
9	5	1	20	0,980	0,986	13023
9	5	1	30	0,983	0,989	19591
9	5	5	10	0,975	0,986	2770
9	5	5	20	0,977	0,976	5535
9	5	5	30	0,977	0,984	8316
9	7	1	10	0,975	0,996	4697
9	7	1	20	0,977	0,974	9275
9	7	1	30	0,979	0,981	13832
9	7	3	10	0,977	0,981	3357
9	7	3	20	0,978	0,982	6721
9	7	3	30	0,973	0,990	10076
9	9	1	10	0,979	0,989	1410
9	9	1	20	0,977	0,994	2815
9	9	1	30	0,981	0,979	4222
11	1	1	10	0,949	0,920	23613
11	1	11	10	0,945	0,989	3779
11	3	1	10	0,960	0,938	20675
11	3	3	10	0,964	0,986	13398
11	3	9	10	0,942	0,936	8055
11	5	1	10	0,969	0,982	12458
11	5	7	10	0,957	0,939	5255
11	7	1	10	0,977	0,979	8195
11	7	5	10	0,967	0,984	4639
11	9	1	10	0,968	0,969	7563
11	9	3	10	0,969	0,987	6343
11	11	1	10	0,968	0,974	2306
13	1	1	10	0,941	0,921	36118
13	1	13	10	0,944	0,977	7403
13	3	1	10	0,958	0,964	34635
13	3	11	10	0,924	0,916	16185
13	5	1	10	0,956	0,932	22797
13	5	3	10	0,958	0,960	15090
13	5	9	10	0,921	0,926	9688
13	7	1	10	0,934	0,884	15033
13	7	7	10	0,963	0,963	7693
13	9	1	10	0,962	0,961	11746
13	9	5	10	0,956	0,945	8154
13	11	1	10	0,960	0,943	12642
13	11	3	10	0,968	0,964	11235
13	13	1	10	0,964	0,961	3671

Table 6.3: Performance of best configurations.

Performance of the best configurations of the CNN.

Image size (pixels)	Filter size	Pooling size	Number of filters	Accuracy	Sensitivity	Time(s)
5	1	1	10	0,953	0,970	9156
5	3	1	10	0,960	0,965	4462
5	3	1	20	0,953	0,941	8943
5	3	1	30	0,959	0,959	13353
5	3	1	40	0,952	0,942	18132
5	3	1	50	0,957	0,956	22213
5	5	1	10	0,964	0,970	1005
5	5	1	20	0,958	0,952	2004
5	5	1	30	0,963	0,979	3005
5	5	1	40	0,963	0,977	4013
7	1	1	10	0,974	0,992	18606
7	1	1	20	0,974	0,988	37428
7	1	1	40	0,966	0,962	75796
7	1	1	50	0,972	0,981	95309
7	1	7	10	0,907	0,884	2675
7	1	7	20	0,907	0,898	5323
7	1	7	30	0,907	0,898	8369
7	3	1	10	0,977	0,988	12046
7	3	1	20	0,975	0,976	24077
7	3	1	30	0,968	0,959	36306
7	3	1	40	0,977	0,982	48115
7	3	1	50	0,975	0,974	59901
7	3	5	10	0,911	0,903	4200
7	3	5	20	0,913	0,904	8437
7	3	5	50	0,915	0,895	20907
7	5	1	10	0,976	0,998	6514
7	5	1	20	0,975	0,973	13011
7	5	1	30	0,971	0,964	19822
7	5	1	40	0,968	0,956	26037
7	5	1	50	0,978	0,993	32442
7	5	3	10	0,959	0,971	3899
7	5	3	40	0,958	0,964	15910
7	7	1	10	0,978	0,979	1857
7	7	1	20	0,979	0,985	3703
7	7	1	30	0,978	0,994	5549
7	7	1	40	0,979	0,986	7403
7	7	1	50	0,973	0,968	9230
9	1	1	10	0,981	0,999	34673
9	1	1	20	0,983	0,996	69466
9	1	9	10	0,923	0,909	5475
9	1	9	20	0,926	0,911	9711
9	1	9	30	0,927	0,909	14925
9	3	1	10	0,978	0,980	26919
9	3	1	20	0,966	0,945	53461
9	3	1	30	0,972	0,972	80464
9	5	1	20	0,959	0,931	30203
9	5	1	30	0,980	0,981	44808
9	5	5	10	0,976	0,998	6646
9	5	5	30	0,977	0,997	19827
9	7	1	10	0,983	0,995	11421
9	7	1	30	0,980	0,979	33869
9	7	3	10	0,977	0,990	8278
9	7	3	20	0,975	0,994	17197
9	7	3	30	0,975	0,979	25780
9	9	1	10	0,981	0,996	3389
9	9	1	20	0,982	0,987	6755

References

[1] http://deeplearning.net/tutorial/lenet.html. (Cited on page 22.)

[2] http://www.embedded-vision.com/platinum-members/cadence/embedded-vision-training/documents/pages/neuralnetworksimagereco (Cited on page 22.)

[3] http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/. (Cited on page 20.)

[4] Fredric L Coe, Andrew Evan, and Elaine Worcester. Kidney stone disease. The Journal of clinical investigation, 115(10):2598–2608, 2005. (Cited on page 1.)

[5] Pavel Golik, Patrick Doetsch, and Hermann Ney. Cross-entropy vs. squared error training: a theoretical and experimental comparison. In INTERSPEECH, pages 1756–1760, 2013. (Cited on page 27.)

[6] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. IEEE transactions on pattern analysis and machine intelligence, 35(1):221–231, 2013. (Cited on page 7.)

[7] Nikhil Johri, Bruce Cooper, William Robertson, Simon Choong, David Rickards, and Robert Unwin. An update and practical guide to renal stone management. Nephron Clinical Practice, 116(3):c159–c171, 2010. (Cited on page 1.)

[8] Konstantinos Kamnitsas, Liang Chen, Christian Ledig, Daniel Rueckert, and Ben Glocker. Multi-scale 3d convolutional neural networks for lesion segmentation in brain mri. Ischemic Stroke Lesion Segmentation, page 13, 2015. (Cited on page 7.)

[9] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolu-

tional neural networks. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pages 1725–1732, 2014. (Cited on page 7.)

[10] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, R. E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Handwritten digit recognition with a back-propagation network. In D. S. Touretzky, editor, Advances in Neural Information Processing Systems 2, pages 396–404. Morgan-Kaufmann, 1990. (Cited on page 6.)

[11] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998. (Cited on page 6.)

[12] Hak Jong Lee, Kwang Gi Kim, Sung Il Hwang, Seung Hyup Kim, Seok-Soo Byun, Sang Eun Lee, Seong Kyu Hong, Jeong Yeon Cho, and Chang Gyu Seong. Differentiation of urinary stone and vascular calcifications on non-contrast ct images: An initial experience using computer aided diagnosis. Journal of digital imaging, 23(3):268–276, 2010. (Cited on pages 5, 39, and 40.)

[13] Wen Li, Fucang Jia, and Qingmao Hu. Automatic segmentation of liver tumor in ct images with deep convolutional neural networks. Journal of Computer and Communications, 3(11):146, 2015. (Cited on page 6.)

[14] Jianfei Liu, Shijun Wang, Evrim B Turkbey, Marius George Linguraru, Jianhua Yao, and Ronald M Summers. Computer-aided detection of renal calculi from noncontrast ct images using tv-flow and mser features. Medical physics, 42(1):144–153, 2015. (Cited on pages 5, 39, and 40.)

[15] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, pages 922–928. IEEE, 2015. (Cited on page 7.)

[16] Orson W Moe. Kidney stones: pathophysiology and medical management. The lancet, 367(9507):333–344, 2006. (Cited on page 1.)

[17] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning (ICML-10), pages 807–814, 2010. (Cited on page 21.)

[18] Holger R Roth, Le Lu, Amal Farag, Hoo-Chang Shin, Jiamin Liu, Evrim B Turkbey, and Ronald M Summers. Deeporgan: Multi-level deep convolutional networks for automated pancreas segmentation. In Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015, pages 556–564. Springer, 2015. (Cited on page 6.)

[19] Dominik Scherer, Andreas Müller, and Sven Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In Artificial Neural Networks–ICANN 2010, pages 92–101. Springer, 2010. (Cited on page 27.)

[20] Emad AM Andrews Shenouda. A quantitative comparison of different mlp activation functions in classification. In Advances in Neural Networks-ISNN 2006, pages 849–857. Springer, 2006. (Cited on page 26.)

[21] Richard Socher, Brody Huval, Bharath Bath, Christopher D Manning, and Andrew Y Ng. Convolutional-recursive deep learning for 3d object classification. In Advances in Neural Information Processing Systems, pages 665–673, 2012. (Cited on page 7.)

[22] Kenji Suzuki. Pixel-based machine learning in medical imaging. Journal of Biomedical Imaging, 2012:1, 2012. (Cited on page 6.)