Metric learning for kidney stone classification

Torrell Amado, Alejandro

Abstract– Kidney stone formation problems is a condition whose incidence is increasing overtime. One of the worst problems that come with this condition is that it is common for patients to relapse. However, this can be efficiently stopped by identifying the type of stone and therefore, the causes. Currently this task is performed by small groups of trained experts and therefore, urologist can not give an adequate diagnosis. A solution was proposed by using a classifier using Machine learning techniques but the overall maximum accuracy of 63% was insufficient. In the following article a new approach is proposed for a classifier, as a continuation of the mentioned attempt, using Metric learning techniques like Siamese and Triplet networks. This will result in an overall improvement, from the current Machine learning techniques with a test reaching an outstanding accuracy of 74%. Finally, it is concluded that improving the dataset will be necessary for improving the overall results.

Keywords– Metric learning, Siamese network, Triplet network, Deep learning, ResNet, CNN, Renal calculi, Kidney stone, Computer Vision, Image classifier

1 Introduction

NOWADAYS kidney stone formation is a condition whose incidence rate is increasing overtime. One of the worst complications, which comes with this condition, is that it is very common for patients to relapse and keep forming new kidney stones during their lifetimes. Therefore, finding a way of avoiding patient relapse, with a good diagnosis, is crucial. This diagnosis can be efficiently found by identifying the type of kidney stone and therefore, the causes. Currently, the process of identifying kidney stones is done by small groups of trained human experts. As a result, it is hard for urologist to give an accurate diagnosis.

In this article a solution is presented, which will detail the approach proposed for the resolution of the main objective: being able to classify kidney stones images, in and easy and quick way, using Metric learning techniques.

Traditionally, Machine learning techniques require pre-established visual features, and training of big sets of data samples to obtain an image embedding function, which will help with the sample mapping into an embedding space and therefore, with classifying.

However, the kidney stone dataset, at the disposal of this project, consists of a limited amount of samples and, in addition, the incidence rate of some samples is noticeably rarer than other types, within the same dataset.

For this reasons, the proposed approach is, using two different Metric learning techniques: the Siamese and the Triplet network, since they have shown, in other projects, to be capable of obtaining great results in similar cases: limited amounts of data for each sample or huge amount of classes.

The main goal of this project is, therefore, not only to implement a kidney stone classifier but also, to improve the results obtained from regular Machine learning techniques.

2 State of the art

As mentioned in the introduction, work has already been done in this matter by the CVC. This project was created with the goal of improving the results from myStone: A system for automatic kidney stone classification [1] whose main goal was to offer an alternative for kidney stone classification done by human experts, who share the knowledge and the techniques for this matter which are not common nor widespread. These experts have to be trained for long periods of time with substantial amounts of samples of different types since the basis of the human classification is color and texture feature analysis.

The approach proposed was to use a device for capturing images, in different light conditions, of the kidney stone samples which where then introduced in a classifier which was developed using Computer Vision and Machine Learning techniques. This samples where previously split, resulting in two pieces, which allowed obtaining two images from both halves, one from the exterior and one from the interior.

Since the project being described in this article is considered a continuation of the previous project, the dataset used in both project is exactly the same. This dataset, consist

• E-mail de contacte: alextorrell@hotmail.es
• Menció realitzada: Menció en Computació
• Treball tutoritzat per: Joan Serrat (departament)
• Curs 2017/18

of 454 samples of kidney stones, which will be explained in detail in following sections, and accordingly it suffers from the same problems. Despite the setbacks from the dataset, which will be explained in the following sections, it achieved an overall accuracy value of 63% when the real class was the predicted most probable class and 80% when it was either the first or the second prediction.

The previous results were obtained using a random forest classifier which used the color and texture features from the kidney stone samples obtained using various Computer Vision feature extraction techniques, such as gray level histograms, color histograms, Linear binary pattern and combinations between them.

After extended tests were conducted, it was concluded that the best combination for this particular case was the combination of the features extracted from the gray level histogram, the color histogram for 16 values and the local binary pattern (Accuracy of 62.91%), followed by the combination of the color histogram for 8 values and the linear binary pattern, with similar results (Accuracy of 62.78%).

However, the results were previously improved by the use of a Neural Network. However, there is not a lot of information about this, but it used a ResNet architecture for training the data, and the test results reached a maximum accuracy of 67%.

3 Objectives

After the previous analysis and understanding of the problem, the main objective would be, without any doubt, testing a new set of techniques to improve the previous results. In this case, the new techniques will be Embedding learning techniques like Siamese and Triplet networks.

Therefore, this objectives can be expanded into the following:

- Implementing a Siamese Network architecture
- Implementing a Triplet Network architecture
- Tests both architectures with different parameters
- Visualize sample's embeddings in a 2D space as well as the clusters they will form
- Find the best parameters for each architecture
- Conclude if the main goal has been reached

4 Methodology

The solution proposed for reaching the goal of this project was developed using the programing language Python. This language was chosen due to the great amount of libraries and community support that it offers. The main reason why the libraries that work with Python is important is because Tensorflow will be used for this project.

Tensorflow is an open-source multi-platform software library developed by the Google Brain Team within Google's Machine Intelligence research organization for inside use purposes and was finally published on the 9th of November of 2015. This library is used for Machine learning purposes, however, it is mainly used for implementing and training

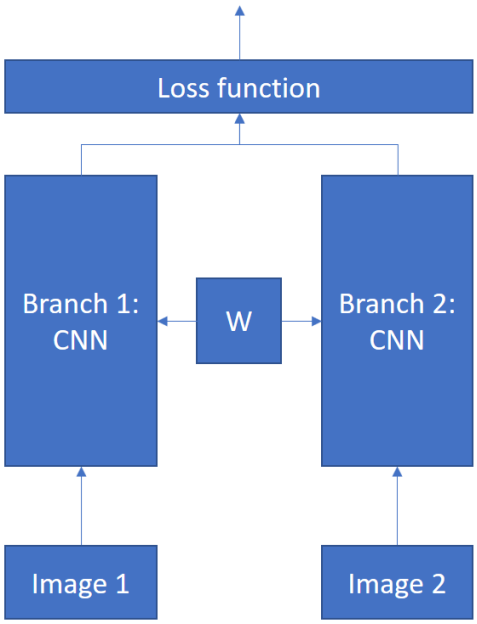

Fig. 1: Siamese network architecture schematics

Deep learning techniques as Neural networks for a great variety of backgrounds and goals.

The main advantage, and the reason why it was selected for being used in this project, is that offers the possibility to use the GPU as a processing unit in a simple way. Thanks to this library, the code for configuring and running the Embedding learning techniques which will be used, the Siamese and the Triplet network, can be implemented. This is because GPUs are faster, compared to a CPU, when running the code that will be implemented for this tasks and therefore the time for each test will be shorter which will result in a greater amount of tests.

5 Background

It is important to introduce some of the techniques, processes and tools that were part of the resolution of the project before further explaining is done.

Siamese network

This is basically a Metric learning technique which uses two identical branches, which consist of the same Neural Network architecture, as can be seen in Fig. 1 [2]. It will be used to train the data and generate a model which for this project will be classifying kidney stones. Each of the branches will output a set of features, instead of a predicted class as for regular Neural networks used for classifying, and this features are obtained from the samples that get into the branch. Obtaining a set of features from each sample means that a set of cluster, for each class can be obtained, and that classifying can be reduced to finding the closer cluster to a sample.

This technique was suggested for achieving the goal of this project to test its capabilities and how appropriate is for situations where a reduced set of samples is found in

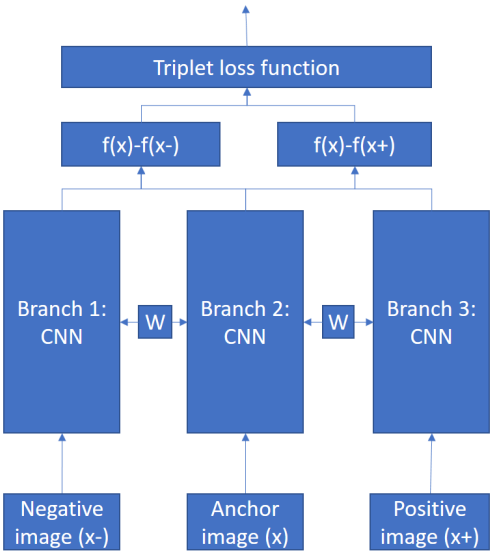

Fig. 2: Triplet network architecture schematics

relation to the number of classes which is usually bigger than average.

Triplet network

This is another approach for reaching the goal of this project and as well as the Siamese network, it is a Metric learning technique. Therefore, the Triplet network technique will also offer the possibility to create clusters for each class and classify the samples by finding the nearest cluster. However, this particular architecture, as the name suggests, uses three identical branches which share the same Neural network architecture and parameters, which can be seen in Fig. 2. Because of this, the input of the network will be three samples, instead of a couple.

Convolution Neural Network

This technique is similar to regular Neural networks, as they are both layered structure with an input, which will be the kidney samples in this case; and an output, which will be the embeddings obtained from introducing the sample data in the network. The main difference with a Neural Network would be that CNNs are mainly used for image processing and that is also how they are explicitly built and what, therefore, they are expecting as inputs.

To do this, the CNN contains a set of layers: the first one considered the input layer, where the sample image will be introduced; a final layer, know as the output layer; and medium layers known as hidden layers. All this layers consist of groups of neurons that posses weight values, that change overtime due to learning, and biases. Each layer will receive an input, which will be processed, weights will be update and an activation function will be run.

It is regular for CNNs to start with a Convolutional set of layers and end with a set of Fully connected layers. Each convolutional layer takes as input one image, which will be one of the samples views, or the output from another convolutional layer. The input is then convolved using a set

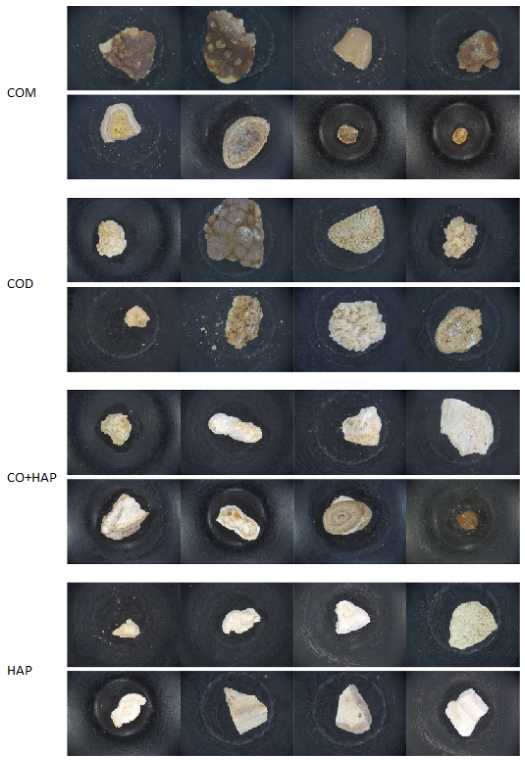

Fig. 3: Kidney stone sample images from the inner and outer surface views

of filters or kernels and an output feature map is obtained, which can be used as an input to an activation function.

Another difference respect regular Neural networks is that while they work with vectors, CNNs work with three dimensional matrices. However, the Fully connected layers will not work properly with three dimensional matrices and thus, the input to the first Fully connected layer has to be turned into a flattened vector using, for example, a Flat pool function.

6 Dataset

This section will explain in detail the dataset of samples which will be trained for generating a predictive model us-

Table 1: This table displays the kidney stone classification system which is used, the amount of stone samples per each class and the percentage of these over the dataset

Class	Number of samples	Percentage
COM	90	19.82
COD	139	30.62
CO+HAP	63	13.88
HAP	19	4.19
STR	38	8.37
BRU	14	3.08
UA	61	13.44
UA+CO	30	6.61
Total	454	

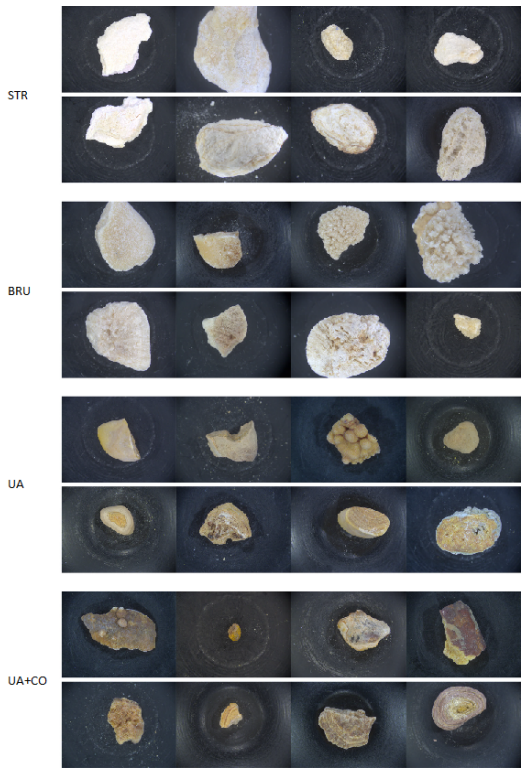

Fig. 4: Kidney stone sample images continuation from Fig. 3

ing the Embedding learning techniques mentioned before. Overall, the dataset is formed by 454 different samples which are divided into 8 different classes and each stone sample consists of 4 different images or views.

The 8 classes mentioned are the ones which are being used on this project, due to their availability. This classification is a summary in which classes are mainly separated by its main components:

- Calcium oxalate monohydrate (COM)
- Calcium oxalate dihydrate (COD)
- Mixed calcium oxalate and hydroxiapatite (CO - HAP)
- Hydroxiapatite (HAP)
- Struvite (STR)
- Brushite (BRU)
- Uric acid anhydrous and dihydrate (UA)
- Mixed uric acid and calcium oxalate (UA + CO)

The use of images from the exterior as well as from the interior is needed due to the fact that some types of stones could be easily confused if the pictures from the outer portion where used. This is important because some stones are differentiable from the traits that can be found only in the inner part [1].

The main particularity of this dataset is the sample amount, which is limited, and therefore will normally result in an insufficient amount of samples for training and testing the generated model. In addition, kidney stone classes have big differences in what amount of stone samples is referred, which can be seen in table 1 where classes like COM or COD represent a great percentage of the total samples, in contrast with classes like BRU or HAP. This results in an unbalanced dataset which independently of the technique used is considered a great adversity within applied Machine learning processes.

However, if there were not enough complications for this particular dataset, kidney stones sample textures and color can be easily confused even for the trained experts as mentioned before. This is the reason why images from the outer and the inner surfaces are used. Some example samples, and the different inner and outer views can be seen in Fig. 3 and Fig. 4.

Despite all this adversities, there are complex techniques to avoid this complications and accomplish the goals set. As mentioned before, the approach for solving the goal of this project will be by using a Siamese or Triplet network which, in theory, will help with the complexities in the dataset. However, there are other techniques that could be added, for example, in addition to the techniques mentioned, data augmentation processes were added in the previous project, already explained in the State of the art section.

Data augmentation consist of a series of transformation on the images of the dataset with its main goal being creating new sets of images from the ones that we already have.

Therefore, the reason behind the use of this technique, for this dataset, is trying to emulate a bigger amount of samples from the ones that we already have which were already mentioned that were limited, especially for some classes (HAP and BRU, as can be seen in table 1).

7 Method

After the previous analysis and understanding of the problem, the detailed implementation of the solution proposed for this project will be explained in this section.

They main task for this would be the implementation of the algorithm which creates and runs models for classifying the kidney stones samples is explained. However, to achieve this, some smaller tasks had to be completed.

The first one being a data structure and a set of functions which could handle and manage all the dataset of samples, an algorithm which will define the Siamese or Triplet network architecture for data training, in addition with the implementation of the Constitutional Neural Network (CNN) which will be used for both architectures and finally, an algorithm that can handle all the sample's embeddings for creating and displaying the clusters in which samples will be grouped.

7.1 Dataset management

A very important part, as mentioned, is how the samples are managed and fed into the training algorithm. This task consists of implementing the code that will offer the possibility to store, within a data structure, all the images from the kidney stone sample dataset. In addition, it will contain the required functions which will manage the dataset so the Embedding learning techniques will obtain the corresponding data samples in the corresponding format for the training process.

This is performed usually by the use of batches of samples, which are then introduced in the Neural network. This is the approach that will be used, however, in the case of the Siamese network, every element of the batch is a couple of stone samples, one for each branch. Therefore, for a Triplet network it will be three samples for each element of the batch.

Now that the batches are defined for each architecture, what an input sample is, has to be defined. As mentioned before, each stone sample from the dataset consists of 4 images, two from the exterior surface and two from the interior surface. This is important because the dataset is complex and full of adversities, which was detailed in previous sections. Therefore, since the goal of this project is to find the best combinations it was convenient to test several input view combinations.

7.1.1 One view

This is the simplest approach, the network's branches are build as regular CNNs, as in Fig. 1 or Fig. 2, with only an image as an input and therefore, only one set of output features. It has to be noted that the selected view is selected randomly over the 4 possible views, 2 inner and 2 outer pictures.

7.1.2 Two views

This is the second approach that was taken and it consists of using only 2 views, out of the 4 views which constitute a sample. As a result, network's branches will take 2 images as an input, which will result in 2 CNNs, one for each input image view. This approach seemed like a better approach than the previous one because it offered more information for each introduced sample.

This approach can be divided into 2 different approaches, as well. The first being, using the first view from the exterior as well as the interior.

The second one being, using an external and an internal, however, both selected randomly over the ones available. This resulted in the best approach with better results due to the considerable increase in different combination of image samples inputed in the network.

7.1.3 Four views

Finally, this is the last approach that was tested and the most complex, time and process demanding. This is because this approach uses all four views, which constitute each sample and therefore, it needs to input a total of four images, into each branch. As a result, each branch will have a total of 4 CNNs, as can be seen in Fig. 5, which will be explained in future sections.

7.2 Siamese network architecture

Once the previous task is ready and data samples can be loaded and managed correctly, the next step is using what the previous classes and functions offer and implement the algorithm which will train the data and generate a model which will classify the kidney stone samples.

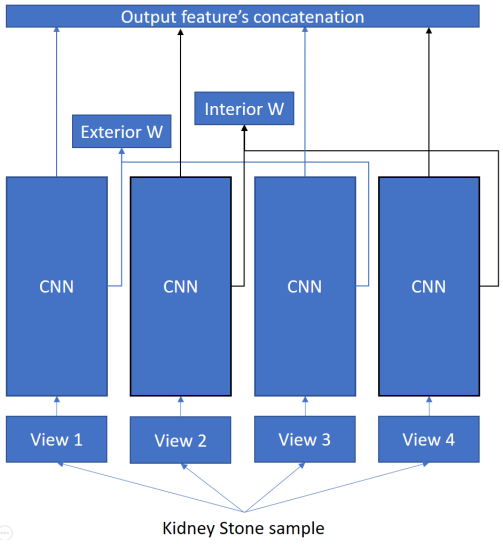

Fig. 5: Network subbranch configured for taking 4 image views from a sample

This is therefore one of the main tasks of the project, the approach will be to implement a Siamese network architecture which will train the data fed from the previously explained functions and generate a predictive model. The Siamese network will be implemented using Tensorflow and samples will be store in data structures known as Tensors.

One important aspect is that samples consist of 4 different images, which can be used in different configurations to obtain the sample features for training the data. This is a complication because, regular Convolutional Neural Networks (CNN) usually take one image as an input instead of four. This is solved by having each image view go into a separate CNN, which will have the same network architecture. This means having as much CNNs within a branch as views used as inputs, as can be seen in image Fig. 5. This will work by sharing the variables and weights between the same type of surface view (for example, exterior images with exterior images only).

Now, each time an image is introduced into a CNN an output is obtained which when implementing Embedding learning techniques, can be considered as the embeddings of the image sample, which can be interpreted as the features of the image as well.

For a normal Siamese network, whose inputs are a couple of images, the embeddings for each sample will be the output of each branch and this is what will be use for the following processes like the loss function, the cluster creation, the classifying, etc. In this project's particular case, each branch is formed by as many CNNs as input views, so each branch will output as many embeddings as input images.

This is, therefore, an adversity, because a vector is needed for the processes previous to the embedding learning and not 4. The solution proposed for this problem is concatenating all of the outputs from the CNNs, within the same branch, as one single output vector. It has to be noted that using a 4 input views from the sample will result in a output feature vector 4 times bigger than when using a single input image.

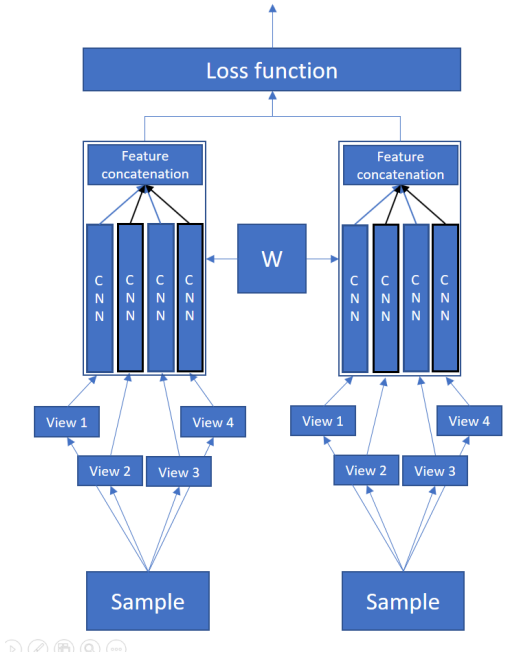

Fig. 6: Siamese architecture schematics when using 4 views per sample

For example, a Siamese network is being implemented using 4 views for each sample, as can be seen in Fig. 6, and 10 elements for each batch: there will be 8 images for each element of the batch due to it being a Siamese network and, as mentioned before, having a couple of samples for each element of the batch; this will result in a total of 80 images that are introduced into the Siamese network. As a result, because it has four views for each sample it will require four CNNs for each branch which in total will be eight CNNs and if for example, the output of each CNN is sixteen, each sample will have sixty-four values as its embedding output.

7.2.1 Siamese loss function

When implementing a Siamese network, one of the most important aspects of it is the loss function, which will be used for evaluating the training process and therefore, assisting in the learning process that will create a good kidney stone classifier.

The classification, as will be explained in following sections, will use the Euclidean distance between sample embeddings, therefore, the loss function for a Siamese network should reduce the Euclidean distance between the samples of the same type and increase the distance, at the same time, between the different types. The approach proposed for the Siamese loss function is a modified version of the Hinge and Contrastive loss functions at [2] and [3] respectively.

The loss function proposed is the following:

$$fpos(s_1, s_2) = max\{D(f(s_1, s_2) - gpos, 0\}^2$$

$$fneg(s_1, s_2) = max\{gneg - D(f(s_1, s_2), 0\}^2$$

$$L(s_1, s_2, y) = y * fpos(s_1, s_2) + (1 - y) * fneg(s_1, s_2)$$

where $D(f(s_1, s_2)$ is the Euclidean distance between the features obtained for each sample and *gpos* and *gneg* are margin constants to control the distance between samples when they are the same and different, respectively. It is recommended that *gpos* is as small as possible, even nonexistent; and *gneg*, on the other hand, needs to be as good as possible. Many margin values were tested, however the one proposed and the ones that gave the best results in the majority of the tests was 3.5 for *gpos* and 18.5 for *gneg*.

It has to be noted as well that the auxiliary functions $fpos(s_1, s_2)$ and $fneg(s_1, s_2)$ calculate the loss value itself when the samples are from the same type and when they are not respectively.

7.3 Triplet network architecture

Once the code for the Siamese network is implemented and tests to identify the best parameters have been made, the next step is to implement and test the Triplet network architecture approach for reaching the goal of this project, which is as well one of the main tasks.

As mentioned before, the class and functions are ready for loading and managing the dataset for the Siamese, however, the Triplet network will require a small change in how the samples are managed and sent into it. This is because the Triplet network will require that each element of the batch of samples consist of 3 different samples, instead of a couple, and therefore, there will be 3 branches instead of 2 [7].

Regarding the inputs, there will be 3, one for each branch, as can be seen in Fig. 2: a query sample, which is the sample which will be compared; a positive sample, which will be the same type as the query; and a negative sample, which will be different [6, 7]. Therefore, as in the Siamese network, it will need to compare both equal and different samples to learn, however, the Triplet network will learn about both in the same training step.

Like in the Siamese network approach, this technique obtains embedding features for the samples introduced, which will is the reason why samples can be mapped into an Euclidean space. This will allow for a classification using the Euclidean distance between samples in this space. To achieve this, the Triplet network, as well as the Siamese, will need a good loss function.

Since this network is working with the same dataset as the Siamese, the complications that appeared from the sample consisting of 4 different views are shared. This means that every branch of the Triplet network will have as many CNNs as views that will be used for the tests. This will result as well in as many output features as views, which will be concatenated, as in the Siamese network, and will be passed to the following processes like the loss function, the embedding visualization, the sample classifying function, etc.

7.3.1 Triplet loss function

As mentioned before, for this types of architectures, the loss function is crucial. For the Triplet network the loss function used is the Hinge loss functions [2, 6], which is a similar approach to the Hinge loss for the Siamese network.

For this particular technique, the goal is to reduce the Euclidean distance between the anchor and the positive sample, while at the same time increase the Euclidean distance

between the negative and the anchor. The function used is as follows:

$$max\{0, g + D(f(s_i^a), f(s_i^+)) - D(f(s_i^a), f(s_i^-))\}$$

where g is margin value between the distances, between the positive and the anchor and between the negative and the anchor. The f functions would be the embedding functions, which correspond to the input sample going through the network, therefore, $f(s_i)$ would be the output embeddings learned for that sample. Finally, s^a corresponds to the anchor sample, s^+ corresponds to the positive sample, with the same type as the anchor; and s^- corresponds to the negative sample.

7.4 CNN

The CNN class implementation is separated from the Siamese and the Triplet network's implementation so it can containing different architectures or configurations for the Convolution and Fully connected layers. This different configurations can be set through the Siamese or Triplet network script and its main purpose was for testing network parameters and architectures for finding the best one to achieve the goal of this project.

As mentioned before, it is regular for CNNs to end with a set of Fully connected layers, which in the case of Metric learning techniques will have as much weights as number of requested features, to define the embedding of the sample.

Many small network architectures were tested at first, resulting in insufficient results due to them not being big enough to learn the necessary features. After this tests, a bigger, more complex net was used; this CNN was based on the SigNet designed by [3] which was, at the same time, inspired by the CNN used in [4]. This resulted, as well, in insufficient results, but better than the previous nets.

Finally, after the previous tests, a conclusion was reached that to capture the most relevant features of the Kidney stone samples, the CNN had to be bigger and more complex.

Therefore, a deep residual network approach, inspired by [8], is proposed as a test. The network architecture used will be referred to as ResNet and it is formed by 10 convolutional layers in total. The layers will be explained in the following sections.

It has to be noted that all the layers within this architectures use activation functions, to add non-linearities, at the end. For this project, Rectified linear units (ReLUs) and Exponential linear units (ELUs) were tested, being the ELUs the ones chosen for the proposed approach due to the better results obtained with them.

7.4.1 Convolutional layers

For this architecture, as seen in Fig.7, we will have 10 convolutional layers (in blue): an input layer followed by a Max Pool (MP, in light gray) layer; 3 groups of 3 convolutional layers, which will end in a MP layer; and finally, an output layer, which will end up as well in a MP layer but will be followed by a Flat Pool layer, which will flatten the output matrix to prepare it for the Fully connected layers.

The main particularity, of the proposed residual network architecture, is that within the 3 groups of 3 convolutional layers, the output will be the addition of the input matrix, of the first layer of the group; and the output matrix of the last layer.

Regarding the convolutional layers, all of them will consist of 64 kernels of filters of a 3x3 size with weights initialized using the *Xavier initializer*, however, the biases will be initialized to a constant value of 0.1.

Now, regarding the Max Pooling layers will use a 2x2 windows and stride size.

7.4.2 Fully connected layers

The Fully connected layers used for this architecture consist of a regular small set of dense layers. This layers consist of an input size equal to the input vector size of the layer, the layer's output approach proposed is using a size which is half of the size of the input, for all the layers except for the output layer. Regarding the weight and bias initialization, the first one uses the *Xavier initializer* and the second is initialized using a constant of value 0.0001.

Three different approaches where chosen for output layer's size: the first one consisted of using 2 layers and making the output layer's size half of the input, for the same layer; the second approach consisting of the same as the first, however, using 3 layers, which results in an output half the size; and finally, using 4 layers, being the output size of the last one any size before running the training process.

For example, if all the approaches mentioned before have a input layer's input size of 5000. The first approach will have an output of 1250, whereas, the second one will have half of that, therefore, 625. For the last approach, the size will be whatever value is set, it could be set to 512 for the previous example.

7.5 Classification

As mentioned before, metric learning techniques as Siamese networks offer the possibility to find the features for the input samples, which will result in simple classification using, for example, the Euclidean distance. This is the case for the solution proposed for this project; samples will be classified regarding their mapping within an embedding space.

The proposed approach is, to form clusters of samples, by finding the mean point for all the output features of the same class, which represent the sample mappings. Once clusters are created, different approaches appear for choosing which is the cluster that correspond to a tested sample like *k nearest neighbors* (k-NN) or finding the nearest cluster center. The proposed approach is the last one, nearest class center, whose process is finding the closest cluster center to the sample.

7.6 Cluster visualization

An interesting aspect of Metric learning techniques, as mentioned before, is that the output of the Neural Networks are embeddings or features obtained from the image, which will be used to map the samples and is what separates this technique from other Neural network techniques. In addition, this allows this technique to form clusters out of the embeddings, obtained from each sample and, therefore, simply

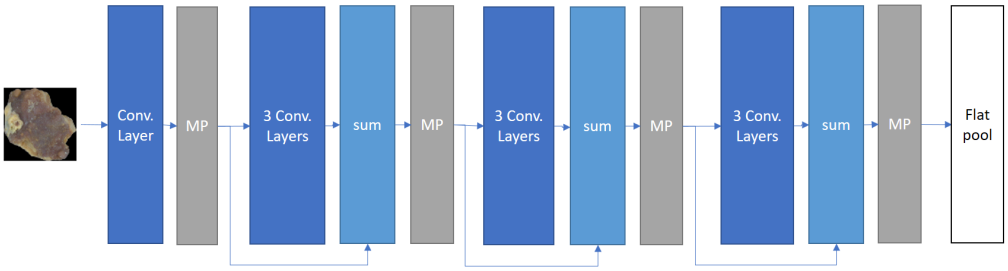

Fig. 7: Schematics for the ResNet architecture used as CNN

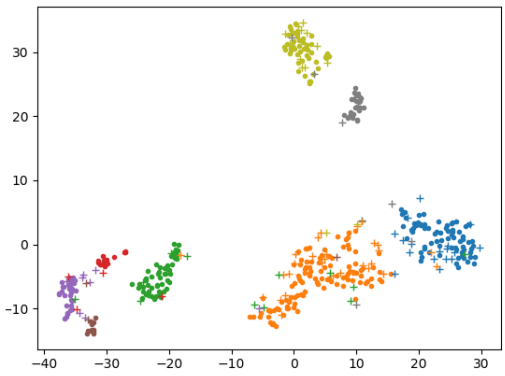

Fig. 8: Siamese network - sample mapping and cluster formation from the output embeddings, after being processed by the TSNE, obtained

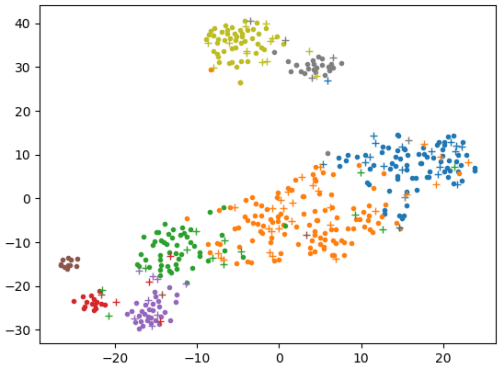

Fig. 9: Triplet network - sample mapping and cluster formation from the output embeddings, after being processed by the TSNE, obtained

classify samples by the closest cluster center to each sample, using the Euclidean distance.

Since the output can be considered as a set of features or embeddings, the idea for visualizing the clusters formed from the samples appeared. However, the output from the network, despite being a flattened vector, has a dimensionality two big for visualization in a 2D board, as suggested.

The size of this output vector is kept relatively big because a small amount of features will, result in insufficient data, as seen in the tests done. Therefore, a dimensionality reduction process is needed for preparing the output features for visualization of samples in a 2D space. The approach proposed is to use TSNE, over other techniques like PCA, adding one last layer at the end of the Fully Connected layers with as much neurons as dimensions desired, etc.

A cluster of samples example can be seen in Fig. 8 and Fig. 9 where the first one is obtained from a Siamese network training of 35K steps, 2 views randomly chosen and an embedding size of 2704; whereas, the second one is obtained from a Triplet network with the same training steps and output size but instead of 2 randomly chosen views, it uses 4. By analyzing this solely, it could be concluded that the Siamese network does a better job that the Triplet at learning, which can also be caused by the different number of views. This will be analyzed in detail in the following.

8 Results

In this section the result of the most relevant test will be presented for the different parameters that were set like steps, number of views, number of output features, etc. All this tests have been run using images of size 200x200 pixels x 3 channels, since samples consist of color images, a learning rate of 1e-4 and the ResNet explained in previous sections.

It has to be noted that, the following test were all done using, always, the same samples as a set for training and the same ones as the set for testing. The train and tests partition was done in a way that 80% of the samples from each class went to the train set and the remaining 20% to the test set. This resulted in a total of 361 samples for training and 93 samples for testing.

Also, the following tables, which represent the results from different tests, show the resulting accuracy range obtained from the several tests done. This range is formed by the minimum and the maximum, accuracy values found.

8.1 Siamese network results

8.1.1 Number of steps comparison

One of the important tests is how will the results behave regarding different training step values. This test were done using four views and a batch size of 12. The different training steps tested can be seen in table 2.

By seeing the results, it can not be denied that between 15K and 35K, the best results are obtained, specially at 20k steps. Regularly, it is expected for the results to improve at the same time that the steps increase but as can be seen,

Table 2: RESULTS ON THE TEST WITH DIFFERENT TRAINING STEPS FOR A SIAMESE NETWORK

Steps	Model accuracy (min - max)
10K	51% - 64.5%
15K	52.2% - 66.7%
20K	58% - 70%
35K	62.4% - 66.7%
50K	63.5% - 65.6%

Table 3: RESULTS WHEN USING DIFFERENT INPUT CONFIGURATIONS WHEN TRAINING A SIAMESE NETWORK

Views	Model accuracy (min - max)
1	31% - 61.3%
2 (first and second view)	55% - 62.3%
2 random	60.2% - 65.6%
4	58% - 70%

specially at 50k steps, they worsen. This may be because a slight state of overfitting is reached, where the network will not learn efficiently, when too much steps are used.

8.1.2 Number of views comparison

One important test that has to be done is this one, testing the effect of the number of different input configurations in the global accuracy of the classifier to find the best one.

The number of views, which where tested are the ones which can be seen in table 3, which are: 1 randomly chosen view, 2 views, choosing the first inner and outer image; 2 views, where the inner and outer images where chosen randomly; and finally, using the 4 views that constitute the sample.

The batch size for 4 views was set to 12, whereas, for 2 and 1 views, it was set to 20, as the tests were intended to be only for the view sizes.

After the tests, the best input configuration for achieving a better overall result is using 4 views, as expected. Followed by using 2 random views, whose overall accuracy was very close to the one obtained using 4 views.

It has to be noted one special case within the tests done, training the data using the approach, previously mentioned, for 2 random views. However, when testing, 4 views were used, which resulted in the best results compared with the rest of the tests. This results were a minimum accuracy of 65.6% and a maximum obtained value of 74.2%

Using 4 views limits the elements that can be stored in the batch, therefore, if using 2 random views and setting batch size to the maximum it can handle, accuracy results will improve definitely. This means that weights, learned when training using 2 random views, work better for the classifier and this is definitely reflected in the test results explained in the previous paragraph.

8.1.3 Number of output features comparison

The purpose of this tests was to analyze the effect of different output sizes and the precision result. As can be seen in 4, difference in sizes does not affect critically the result, however, the bigger the output the higher overall accuracy. Regarding the parameters, this tests were done using 4 views, a batch of 12 samples and 20K steps.

Table 4: RESULTS WHEN DIFFERENT OUTPUT SIZES ARE USED IN THE SIAMESE NETWORK

Output size	Model accuracy (min - max)
32	56% - 63.4%
64	59% - 66.7%
128	57% - 66.7%
2704	58% - 70%

Table 5: RESULTS ON THE TESTS WITH DIFFERENT TRAINING STEPS FOR THE TRIPLET NETWORK

Steps	Model accuracy (min - max)
15K	53.8% - 62.4%
20K	61.3% - 64.5%
35K	63.4% - 72.1%
50K	60.2% - 67.7%

8.2 Triplet network results

8.2.1 Number of steps comparison

This tests were done in order to see how the precision of the classifier improves when increasing the training step of the Triplet network. The test were done using the 4 views, which constitute a sample; a batch size of 10, and an output embedding size of 2704.

As can be seen in table 5, the tested steps were: 15K, 20K, 35K and 50K. From the results of this tests, it can be seen that numbers above 35K steps might be too much for the learning process, as happened in the Siamese network. It can be concluded that from 20K to 35k steps, the optimal range of steps is found, which is the same as for the Siamese network.

8.2.2 Number of views comparison

This tests where run in order to compare the number of sample views used as inputs and conclude which is the best for classification, using the Triplet network. The output value is the same as for the previous tests, 2704. It has to be noted

Table 6: TEST RESULTS WHEN VARYING THE NUMBER OF SAMPLE VIEWS USED FOR THE TRAINING AND TESTING PROCESSES

Views	Model accuracy (min - max)
1	36.5% - 55.9%
2 (first and second view)	47.3% - 61.3%
2 random	55.9% - 63.4%
4	53.8% - 62.4%

Table 7: RESULTS WHEN DIFFERENT OUTPUT SIZES ARE TESTED FOR A TRIPLET NETWORK

Output size	Model accuracy (min - max)
256	53.8% - 61.2%
512	52.7% - 62.4%
1352	55.9% - 62.4%
2704	61.3% - 64.5%

that this tests were run using only 15k steps and therefore, the results are not as good as expected.

The number of views, which where tested are the ones which can be seen in table 6, which are: 1 randomly chosen view, 2 views, choosing the first inner and outer image; 2 views, where the inner and outer images where chosen randomly; and finally, using the 4 views that constitute the sample.

Regarding the number of elements used for the batches, each tests used a number as big as possible for the images used, as opposite to the same test for the Siamese. For 1 view, a size of 60 was chosen; for both 2 views test, 25 elements were chosen; and finally, for 4 views, 10 elements were chosen.

Now that the batch size is as big as it can be, what was predicted in the Siamese's tests for different input configurations, which is that 2 randomly chosen views is a better approach than using 4 views. This is because using 2 randomly chosen views, the number of samples that can be used for training is bigger than using the 4 views that constitute the sample.

Finally, since training the Siamese network with 2 randomly chosen views as inputs and using 4 views, for obtaining the embeddings, for the classification; it was tested for the Triplet network as well, and it obtained the best results, as expected. The results it obtained are between the range of 61.3% and 69.9%.

After all this tests, it can be concluded that using 2 views as inputs, a random inner picture and a random outer function, from the same sample; results in a better training and learning of the data, for both Metric learning architectures.

8.2.3 Number of output features comparison

The following tests were run to compare some possible output sizes, which from the Siamese tests, it was clear that a big number was needed to map the kidney stone samples correctly. The output size chosen for the test can be seen in table 7.

It has to be noted that the tests where run using 20k steps and 4 views. Now, regarding the output sizes, they where created thanks to the approach explained in the CNN section of this article.

As for the Siamese network, the bigger the output size, the better the results. It is important to remember, from previous sections, that the output size number represented is the output from a CNN. Therefore, when using a 4 view input configuration, 4 CNNs are created for the branch for Siamese as well as for Triplet networks; the output will be 4 times the size of the output of a CNN. For example, if the output is 2704 for a cNN, when using 4 views, the total feature for a sample will be 10816.

Now, regarding how a big number of features affects the learning and training process; it should have a big effect, however, using 2 random views, does a better training than using 4 views, which has double the feature size.

Regarding how it affects the testing, it has a big effect, which can be seen when using 4 views over 2 when creating clusters and classifying samples. This can especially be seen when the training is done using 2 random views. When the test is done using 2 random views the results are (55.9% - 63.4%), whereas, when using 4 views they are (61.3% - 69.9%.)

9 CONCLUSIONS

After analyzing the previous tests and the project's process it can not be denied that, top limits exists regarding the precision of the predictive model used for the classification of kidney stone samples.

It is clear that, the dataset, is complex, unbalanced and limited in size and therefore, it is full of adversities and complications which affect reaching the goal of the project. However, as seen for the tests, using this techniques over the regular Machine learning techniques, leads to better results.

What can be concluded from the tests is that, the accuracy improved with the increase of the amount of steps, until it reached 35K, which is closer to the maximum number of steps; the best input combination would be, using 2 randomly chosen views, which has shown to be the best approach for training the data; and finally, using a big output size to create the map of the samples into a embedding space, for creating the clusters and classifying the samples, is undoubtedly the best approach.

It can not be denied that having a bigger batch of samples, when using 2 random views as an input, produces more possible samples combination than any other input approach tested. Therefore, it can learn from more sample's views combinations, which as expected, led to better results.

Overall, it has to be noted that the Metric learning techniques have resulted in a great improvement over the previous techniques. Reaching an outstanding maximum accuracy of 74%, by the Siamese network, which is 11% over the old alternative, using regular machine learning techniques. Despite the implementation of the Triplet network proposed, being an evolved and improved version of what a Siamese is, did not achieve better results than the Siamese. The main reason for this, would be the Triplet loss function used, and maybe, after modifying or tweaking the loss function parameters, it could achieve a better performance in the future.

Also, it is important to note that working with a dataset full of adversities is great for learning due to the complications and thus making a greater effort when improving the results. It has to be noted that maybe introducing some new techniques, like input and batch normalization or a bigger and more complex CNN architectures, could help as well.

Finally, it is safe to say that a bigger, better dataset would be a great improvement in the results. This means that, having a balanced number of samples for each class, but also a bigger amount of samples, could have a great effect in the overall results.

ACKNOWLEDGMENTS

I want to thank, first of all, my tutor Joan, for having no problem regarding sharing his knowledge and experience and for giving me a lot of great ideas for testing. I also want to thank all my friends and family, even though the were unfamiliar with the techniques, for sharing ideas and helping me to stay motivated.

REFERENCES

[1] Joan Serrat[1], Felipe Lumbreras[1], Francisco Blanco[2], Manuel Valiente[2], Montserrat López-Mesas[2]. myStone: A system for automatic kidney stone classification. [1]*Computer Vision Center and Department of Computer, Universitat Autònoma de Barcelona, Edifici O, 08193 Bellaterra, Spain.* [2]*Centre Grup de Tècniques de Separacióen Química, Unitat de Química Analítica, Departament de Química, Universitat Autònoma de Barcelona, Bellaterra 08193, Spain*.

[2] Hengliang Luo. Siamese Network: Architecture and Applications in Computer Vision. Dec 30, 2014.

[3] Sounak Dey[1], Anjan Dutta[1], J. Ignacio Toledo[1], Suman K.Ghosh[1], Josep Lladós[1], Umapada Pal[2]. SigNet: Convolutional Siamese Network for Writer Independent Offline Signature Verification [1]*Computer Vision Center, Computer Science Dept., Universitat Autònoma de Barcelona, Edifici O, Campus UAB, 08193 Bellaterra, Spain* [2]*Computer Vision and Pattern Recognition Unit, Indian Statistical Institute, 203, B. T. Road, Kolkata-700108, India*

[4] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *University of Toronto*

[5] Sumit Chopra, Raia Hadsell, Yann LeCun. Learning a Similarity Metric Discriminatively, with Application to Face Verification *Courant Institute of Mathematical Sciences, New York University, New York, USA*

[6] Jiang Wang[1], Yang Song[2], Thomas Leung[2], Chuck Rosenberg[2], Jingbin Wang[2], James Philbin[2], Bo Chen[3], Ying Wu[1]. Learning Fine-grained Image Similarity with Deep Ranking. [1]*Northwestern University* [2]*Google Inc.* [3]*California Institute of Technology*

[7] Elad Hoffer[1], Nir Ailon[2]. Deep metric learning using Triplet network. [1]*Department of Electrical Engineering, Technion Israel Institute of Technology* [2]*Department of Computer Science, Technion Israel Institute of Technology*

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition. *Microsoft Research, 10 Dec 2015*

[9] Christopher Olah. Visualizing MNIST: An Exploration of Dimensionality Reduction. October 9, 2014.

[10] Stanford CS class CS231n: Convolutional Neural Networks for Visual Recognition. http://cs231n.github.io/convolutional-networks/