Zagazig University

Faculty Of Computer And Informatics

Department Of CS

B. Eng. Final Year Project

Intelligent Magnetic Resonance Imaging Scan

(I-MRI scan)

By:

Mohamed Tarek Refaat - Mohamed Abdelnaby Abdelwahab

Mohamed Qadry Ahmed Amin - Mohamed Mostafa Mohamed Migahed

Sophy Maher Hamdy - Amr Mohamed Faruq

Supervised By: *Dr. Alaa El-Ghamry*

ACKNOWLEDGMENT

We are honoured at the beginning of this graduation project; which we hope you will like and be appreciated we extend my sincere thanks to the respectable Professor Alaa El-Ghamry Who made an unrivalled effort in providing guidance for this work to be at the required level Note that we admit to failing to thank Because what he did is greater than thanks and appreciation But he is the father who does not wait for his sons to thank And the professor, who was overwhelmed by the success of his students To be the pillar and support of this nation in the paths of progress.

We would like to thank all the faculty members for helping us reach this level of scientific work for this project. Thank you very much, all of my professors, the honorable faculty members of our college, Faculty of Computing and Information, Zagazig University.

DECLARATION

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Bachelor of Science in *(insert title of degree for which registered)* is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: *Mohamed Tarek Refaat*

Mohamed Abdelnaby Abdelwahab

Mohamed Qadry Ahmed Amin

Mohamed Mostafa Mohamed Migahed

Sophy Maher Hamdy

Amr Mohamed Faruq

Registration No.: _____

Date: 8, August 2020.

ABSTRACT

While magnetic resonance imaging (MRI) is frequently considered the "best quality level" diagnostic imaging methodology for recognition of meniscal abnormalities, it is related with misdiagnosis in as high as 47% [1] of cases, is exorbitant, and isn't promptly accessible to countless patients. The purpose of this project reduces the MRI knee error rate to see the most appropriate solution to treat knee problems. We developed a framework for deciphering knee MRI could organize high-chance patients and help clinicians in making analyze. Deep learning techniques, in having the option to consequently learn layers of highlights, are appropriate for demonstrating the mind-boggling connections between clinical pictures and their understandings. In this investigation we built up a profound learning model for distinguishing general abnormalities and specific diagnoses ([ACL] tears and meniscal tears) on knee MRI tests. We at that point estimated the impact of giving the model's forecasts to clinical specialists during understanding.

What does our project do?

- *Our deep learning model anticipated 3 results for knee MRI tests (ACL tears, meniscal tears, and general abnormal) very quickly and with comparable execution to that of general radiologists.*

Why did we do this project?

- *To reduce the MRI knee error rate to see the most appropriate solution to treat knee problems*.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ACRONYMS/ABBREVIATIONS

1. (ML) …… Machine Learning is the scientific study of algorithms in PC used patterns
2. (DL) ……..…… Deep learning is a class of machine learning that uses multi-layers
3. (OpenCV) …… Open Computer Vision (Image Processing Library)
4. (HTML) ……..……Hyper Text Markup Language
5. (CSS) ……Cascading Style Language
6. (MRI) ……Magnetic resonance imaging
7. (MSK) ……..…… Musculoskeletal
8. (ACL) …… Anterior Cruciate Ligament
9. (ACC) ……..…… Accuracy
10. (Val) ……...……Validation
11. (e_no) ……Number of Epochs
12. (CNN) ……Convolutional Neural Network
13. (GUI) ……..… Graphical User Interface
14. **(**AlexNet**)** …… Is a Convolution Neural Network Created by Alex Krizhevsky
15. (SVM) …… Support Vector Machine
16. (LR) ……... Logistic Regression

LIST OF UTILIZED STANDARDS

Hardware Requirements:

1. **CPU: 3 GHz**
2. **GPU: NVIDIA TITAN RTX**
3. **RAM: 16 GB**
4. **HDD: 500GB**

Environment Impact:

The Project Is Familiar with Every Kind of Users. It Can Be Used by Anyone.

Manufacturability:

the project can be used in a wide range of people. it can give a high, accurate performance with the required results. its cost of resources is so little.

Health:

One of its components and stages is a health care state. especially, the knee problem. so, it can be so benefit in the human health. orthopaedics can get helpful application use as a flexible and usable app for knee problems.

Sustainability:

the project is expected to improve its performance in the future. and, entering more parts are used in MRIs such as the shoulder, spine, etc.

LIST OF REALISTIC CONSTAINTS

- Deep Learning: Is a basic class of the meaning of Machine Learning. It is used in many AI applications in various fields. One of its most popular Architecture is the (DNNs) Deep Neural Networks, that is depend on the meaning of dividing big data into many stages and apply all of its components separately.

- Deep Neural Networks (DNNs): it has many architectures like convolution neural network (CNN) and recurrent neural network (RNN) and their architectures.

- Convolution neural network (CNN): it is layered model processing the depend on multi-layer perceptron (neuron which is building unit in neural network) and it consists of input layer, one or more hidden layers and output layer. CNN use a variation of multilayer perceptron's designed to require minimal preprocessing. CNN is used in many applications like image classification, image segmentation. it is used in computer vision and to train the models on three stages:

1. Convolution layer: to pass the input to the next layer.
2. Pooling layer: to combine the outputs of neurons clusters at into single neuron in the next layer.
3. Fully connected layers: connect every neuron in one layer to every neuron in another layer.

- We had run our dataset on two different approaches that is considered as an improvement architecture of CNN.it is compared in our research for choosing the best on in training the MRI knee dataset in 3 different injuries each injury is binary classified. that approaches are:

1. Ensamble learning with transfer learning: each plane with one type of injury is trained separately so we have 3 planes (sagittal, coronal, axial) and 3 different injuries(abnormality, meniscal tear, ACL tear) so we end with 9 trained models each input is(1*S*3*256*256) where S is number of slices or frames , 3 is the color channel, 1 is the number of batches. After train the 9 models combine the three models that intersect in the same injury type in logistic regression binary classifier or support vector machine (SVM) binary classifier. In the transfer learning we use Alex-net as the pre-train model.
2. Multi-Input Convolutional Neural Network with the transfer learning: instead of train 9 models, just train only 3 models where each one is specialized in one injury. each model consists of 3 input layers for the 3 planes for just one type of injury and use pre-trained model Alex-net.

- we classify our data into 3 type of injuries (abnormality, meniscal tear, ACL tear) each type is binary classified. we choose approach Multi-Input Convolutional Neural Network with the transfer learning where it was the best in both training time and results

- We have written deep learning code in python using PyTorch framework for the build the model, train the model and save the training parameters of the model.

- We have built our desktop application with PYQT5 where the application can be used easily with different types of users.

- We use OpenCV (open computer vision) library for some features.

- The GUI consists of name, the blood type and age of patient, then adding the slices for each sagittal, coronal, axial planes then the results will appear for each injury and can transfer the slices to video with extension .mp4 with wanted number of slices per second.

- Future work:

1. Make this application as a website and mobile application.
2. Add new features like segmentation using deep learning approach and 3D reconstruction.
3. Entering more parts are used in MRIs such as the shoulder, spine, etc.

Chapter One

1 Introduction

1.1 Preamble

- We have consistently been astonished by significant AI applications that change individuals' lives and one of the regions where we see this change happening these days is medicinal services. Artificial intelligence and health care are two areas we never thought would unite.

- While magnetic resonance imaging (MRI) is frequently considered the "best quality level" diagnostic imaging methodology for recognition of meniscal abnormalities, it is related with misdiagnosis in as high as 47% [1] of cases, is exorbitant, and isn't promptly accessible to countless patients. The purpose of this project reduces the MRI knee error rate to see the most appropriate solution to treat knee problems.

1.2 Knee Problems:

1.2.1 Anterior Cruciate Ligament (ACL) tear:

- In this project, we will explicitly concentrate on Anterior Cruciate Ligament (ACL) tears which are the most widely recognized knee injuries.

- ACL tears happen when the anterior cruciate ligament is either extended, incompletely torn, or totally torn. The most widely recognized injury is a finished tear.

- It is commonly reported that female subjects have a greater incidence of **ACL tear** than male subjects [2].
- The typical symptoms for an anterior cruciate ligament (ACL) injury include:

  - ✓ hearing a "popping" noise.
  - ✓ feeling the knee giving out.
  - ✓ sudden instability.
  - ✓ pain or discomfort.
  - ✓ swelling within 24 hours.
  - ✓ loss of full range of motion.
  - ✓ joint tenderness.

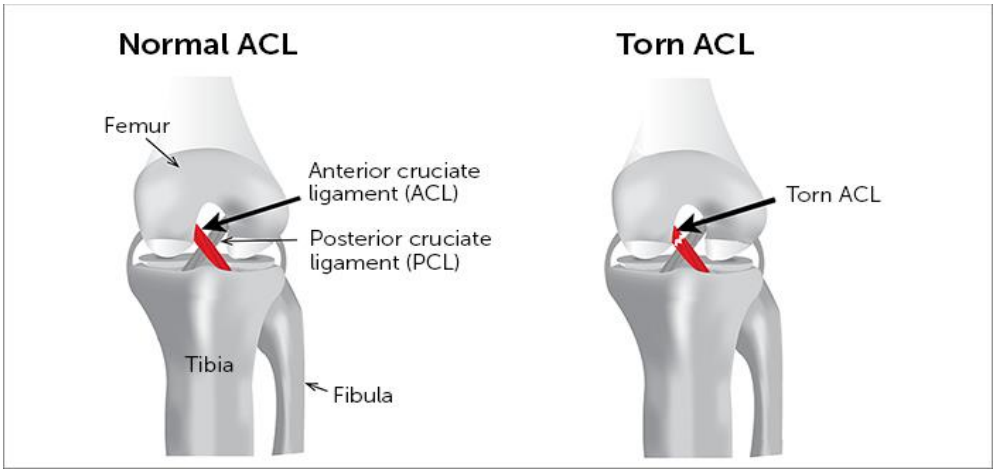


Figure 1-1: (ACL) tear

- Richard Nordnvall of the Karolinska University Hospital and colleagues used a national patient database to find out how many Swedes with ACL and how many had surgery to repair them between 2002 and 2009. in total, 56,659 people had an ACL rupture during the study period. This, they say, means an average of 78 rupture cases per 100,000 Swedish citizens.

- More than 100,000 anterior cruciate ligament injuries have occurred in the United States of America this year [2].

1.2.2 Meniscus tear:

A torn meniscus is one of the most widely recognized knee wounds. Any action that makes you powerfully curve or turn your knee, particularly when putting your full weight on it, can prompt a torn meniscus.

Symptoms

If you've torn your meniscus, you might have the following signs and symptoms in your knee [3]:

- ✓ A popping sensation.
- ✓ Swelling or stiffness.
- ✓ Pain, especially when twisting or rotating your knee.
- ✓ Difficulty straightening your knee fully.
- ✓ Feeling as though your knee is locked in place when you try to move it.
- ✓ Feeling of your knee giving way.

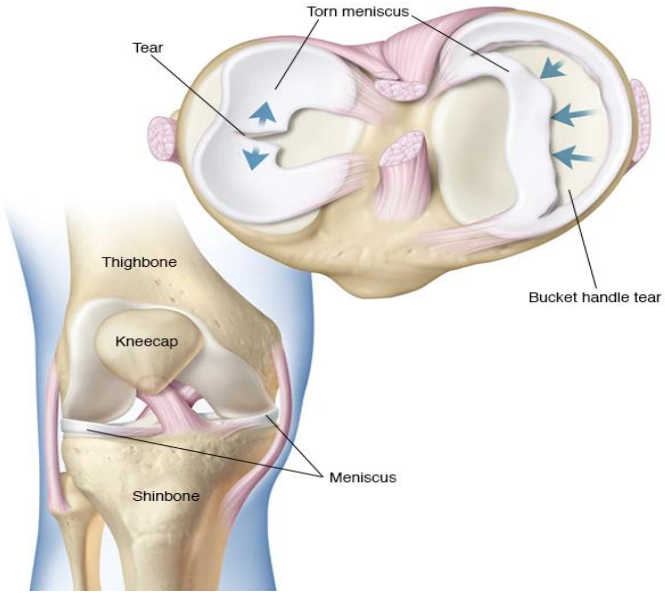


Figure 1-2: Meniscus tear

❖ To analyze knee injuries, we normally use Magnet Resonance Imaging.

1.3 Magnetic Resonance Imaging (MRI)

Magnetic resonance imaging (MRI) of the knee is the favored technique for diagnosing knee wounds. Be that as it may, translation of knee MRI is time-intensive and subject to diagnostic error and variability. An idea in a software program to diagnose MRI for knee. From the start, we were an amateur and We knew nothing about this field. So, We started researching in scientific journals and learned, Magnetic resonance imaging (MRI) of the knee is the standard-of-care imaging methodology to assess knee issues, and progressively musculoskeletal (musculoskeletal (MSK) radiologists are extraordinarily prepared radiologists who analyse conditions that influence the bones, joints, and delicate tissues in the furthest points.) MRI assessments are performed on the knee than on some other district of the body. MRI has more than once exhibited high exactness for the determination of meniscal and cruciate tendon pathology and is routinely used to distinguish the individuals who might profit by a medical procedure. Besides, the negative prescient estimation of knee MRI is almost 100%, so MRI fills in as a non-invasive strategy to preclude carefully scatters, for example, (ACL) tears. In any case, the multidimensional and multi-planar properties of MRI need to date restricted the materialness of customary picture examination strategies to knee MRI.

- MRI is a clinical imaging strategy utilized in radiology to shape an image of the anatomy and the physiological procedures of the body.
- MRI is utilized to analyze how well you reacted to treatment just as distinguishing tears and basic issues, for example, heart failure, brain injury, blood vessel disease, and so forth.
- An MRI check is certifiably not a solitary picture of an organ. It's really a lot of various pictures or cuts stacked together in a volume.

➢ There are a lot of MRI knee protocols. The most famous of these are protocols:

- **PD weighted**

☒ *plane* (sagittal, coronal, axial**).**

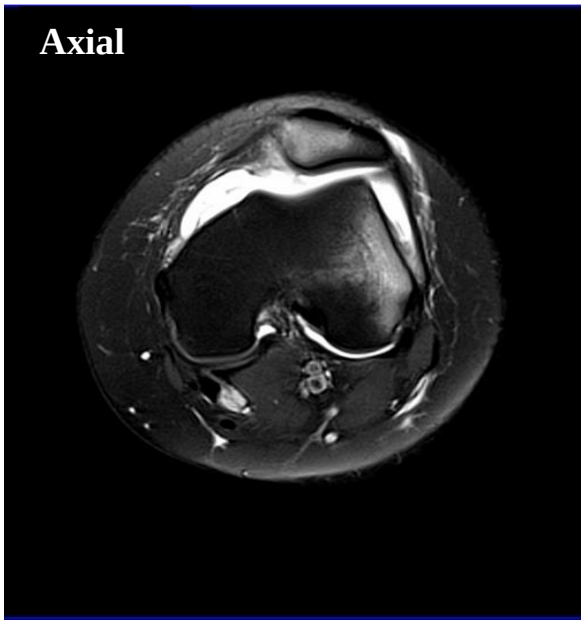


Figure 1-3: MRI Axial PD

- **T1 weighted**

☒ *plane:* at least one, often axial or coronal

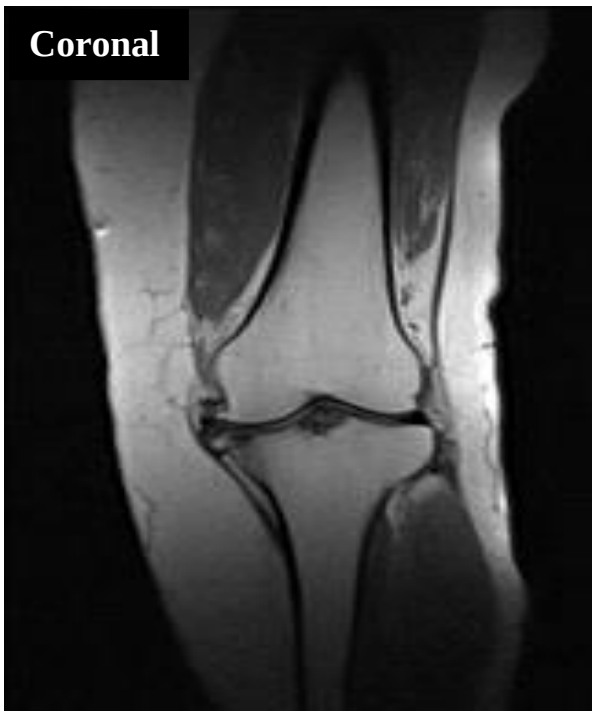


Figure 1-4: MRI Coronal T1

- **T2 weighted (fat-saturated)**

☒ *plane* (sagittal, coronal, axial**).**

Figure 1-5: MRI Sagittal T2

1.4 I MRI-Scan Datasets:

- **I MRI-Scan** datasets is 1250 cases a knee MRI dataset.
- Divided into training set 1130 case and validation set 120 cases and, is composed as follows:
  - **Train**
    - Axial (folder contains train axial dataset)
    - Coronal (folder contains train coronal dataset)
    - Sagittal (folder contains train sagittal dataset)
    - train-abnormal.csv (the file contains the result of abnormal cases if positive or negative)
    - train-acl.csv (the file contains the result of ACL cases if positive or negative)
    - train-meniscus.csv (the file contains the result of meniscus cases if positive or negative)

- **Valid**
  - Axial (folder contains valid axial dataset)
  - Coronal (folder contains valid coronal dataset)
  - Sagittal (folder contains valid sagittal dataset)
  - valid-abnormal.csv (the file contains the result of abnormal cases if positive or negative)
  - valid-acl.csv (the file contains the result of ACL cases if positive or negative)
  - valid-meniscus.csv (the file contains the result of meniscus cases if positive or negative)

1.5 About Deep Learning Model

- Deep learning draws near, in having the option to naturally learn layers of highlights, are appropriate for displaying the intricate connections between clinical pictures and their understandings. As of late, such methodologies have beaten conventional picture examination techniques and empowered critical advancement in clinical imaging errands, including skin disease arrangement, diabetic retinopathy location, and lung knob discovery. Earlier utilization of profound figuring out how to knee MRI has been restricted to ligament division and ligament sore location.

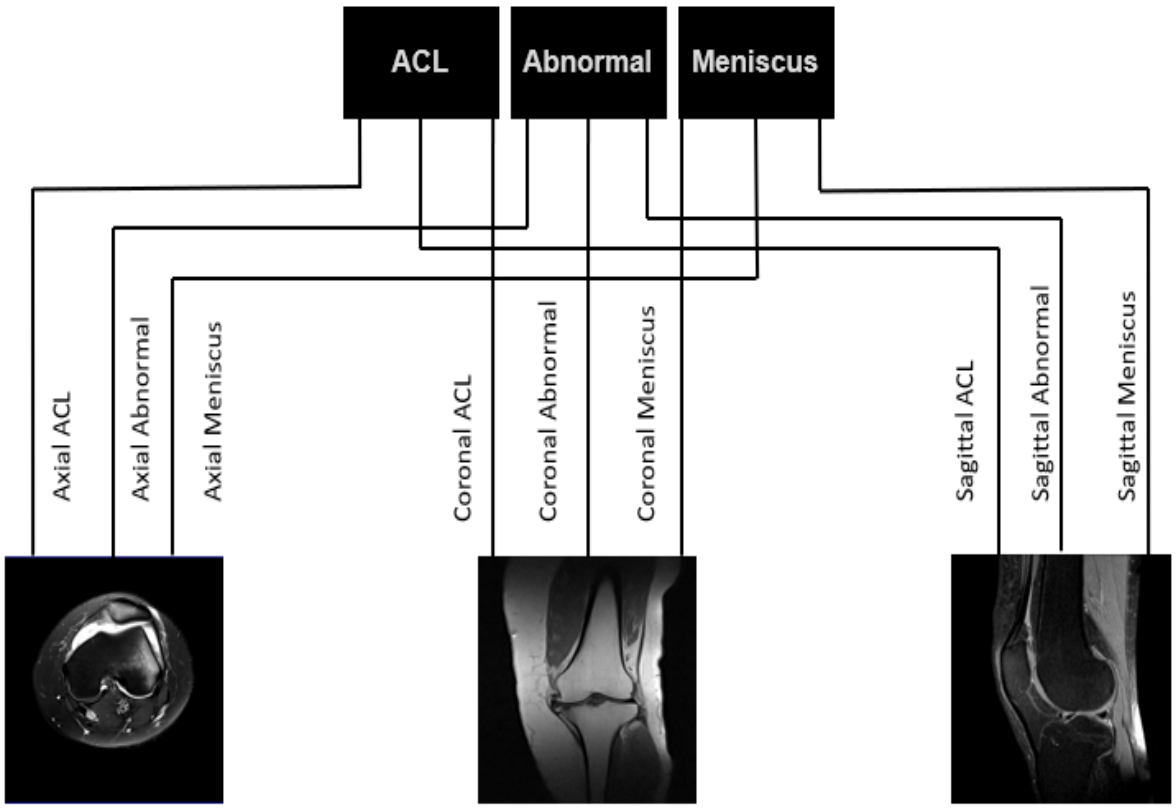
- We present **IMRIScan**, a completely automated deep learning model for deciphering knee MRI and contrast the model's exhibition with that of general radiologists. What's more, we assess changes in the analytic presentation of clinical specialists when the computerized profound learning model forecasts are given during understanding. At long last, we assess our model's presentation on

an openly accessible outer dataset of knee MRI tests marked for abnormalities and specific diagnoses ([ACL] tears and meniscal tears)

- The essential structure square of our expectation framework is **I MRI-Scan**, a convolutional neural network (CNN) mapping a 3-dimensional MRI arrangement to a likelihood.

- The contribution to **I MRI-Scan** has measurements 256 × 256 × 3 × s, where s is the number of pictures in the MRI arrangement (3 is the number of shading channels). To begin with, every 2-dimensional MRI picture cut is gone through an element extractor to acquire a s × 7 × 7 × 256 tensor containing highlights for each cut. A worldwide normal pooling layer is then applied to decrease these highlights to s × 256. We at that point applied max-pooling across cuts to get a 256-dimensional vector, which is passed to a completely associated layer to get a forecast likelihood.

- Since **I MRI-Scan** produces a forecast for each of the sagittal T2, coronal T1, and axial PD arrangement, we train a strategic relapse to weight the expectations from the 3 arrangement and create a solitary yield for every test.



We will discuss on the next sections:

- ❖ The Market and Literature Survey.
- ❖ The Methodology.
- ❖ The System Analysis and Design.
- ❖ The Experiments and Results.
- ❖ The Appendices.
- ❖ The References.

Chapter Two

2 Market and Literature Survey

- **I MRI-Scan** is used by radiologist and Orthopedic specialist but it is used specially by the radiologist.it is the best representation of deep learning and in public artificial intelligence and make it possible for any doctor or surgeon to use it as a service for an important role

- Our project is focusing on the MRI on the knee. And as discussed before these characteristics are great reasons for supporting our research, our theory or technical marketing. our application intelligent Magnetic Resonance imaging scan (I MRI-Scan) will be available for the medical society and medical student. this application has the ability to give an analysis of the MRI knee scan and other characteristics. **I MRI-Scan** is used too in medical rays' centers and hospitals and medical clinics for Orthopedic surgeon and Bone specialist.

  1. **I MRI-Scan** can used for the medical student in the radiology specialization and Orthopedic surgery specialization as helpful software for train the student to differentiate between the injured scan and the non-injured scan.
  2. **I MRI-Scan** can be used in medical rays' centers and clinics as helpful tool for make more accurate examination and take correct decision and write more accurate report for make surgery or something else for treatment.
  3. For the clinics of the clubs whose the professional sports men who got an injuries in knee, they want to get up faster in this kind of situations if a surgery is take place.

- For the market surveys, our project **I MRI-Scan** will be available as web service on the internet soon but now a desktop application is available.

- Theoretically, MRI for knee injuries classification is very important in medical field and medical society.

- In artificial intelligence the deep learning fields are always the most used and popular applications because of its different applications that cover almost every technique. Deep learning has tools and ability to work with different algorithms and environments to achieve its required tasks. Convolutional neural network was almost a part of all search papers that talk about the deep learning algorithms especially the

classification and multi-input convolutional neural network. In image classification the CNNs was the best and most accurate choice.

- Convolutional neural networks (CNN) are analogous to traditional artificial neural networks (ANN) in that they are comprised of neurons that self-optimize through learning stage. Each neuron will still receive an input and perform an operation like (scalar product followed by applied activation function)-the basis of countless ANN's. from the input raw slices to the final output of the class score for each type of injuries of the knee, the entire of network will express a single perceptive score function (the weight).

- CNNs are comprised of three types of layers. The types of layers are convolution layer, pooling layer (has two main types like average pooling and max pooling), fully connected layers. When those layers are stacked, the CNN architecture is formed.
  As this figure show the CNN architecture:

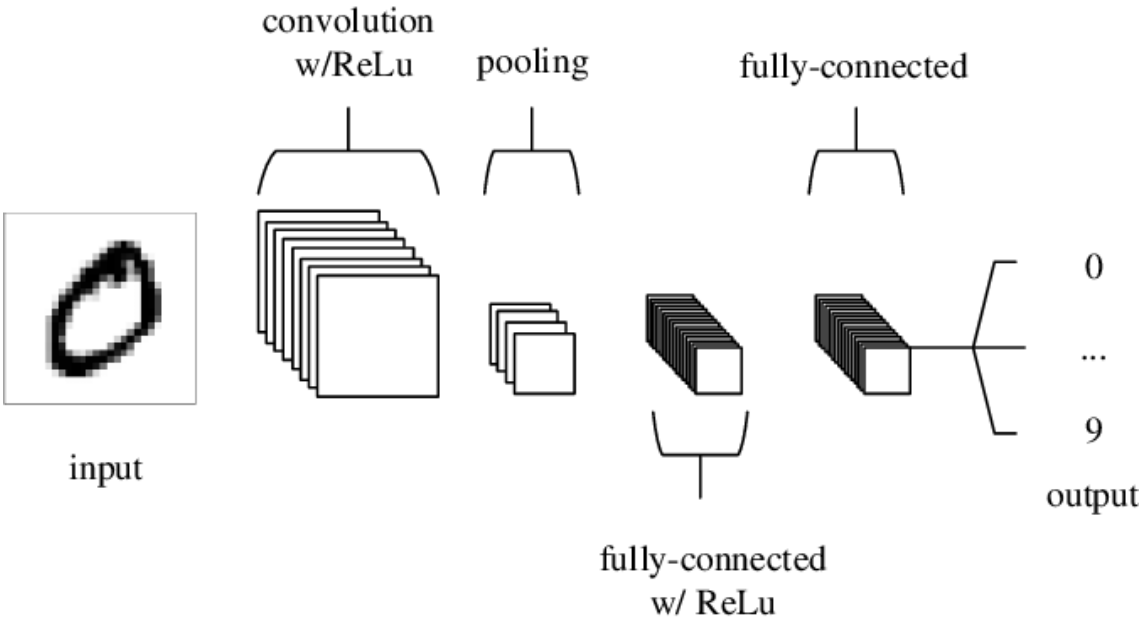


Figure 2-1: A simple CNN architecture.

- The layers of the CNNs: the input layers – the convolutional layers – the pooling layers – the fully connected layers. Through this simple method of transformation, CNNs are able to transform the original input layers by layer using convolutional and down sampling techniques to produce class score for classification or regression purposes.

- The convolutional layer plays a vital role in how the CNN works. the layers parameters focus around use of learnable kernels or filters(matrix of parameters).these kernels are usually small in spatial dimensionality but spreads along the entirely of depth of the input .when the data hits a convolutional layer, the layer convolves each kernel (filter) across the spatial dimensionality of the input to produce activation maps. This figure show the activations of the first convolution layer:

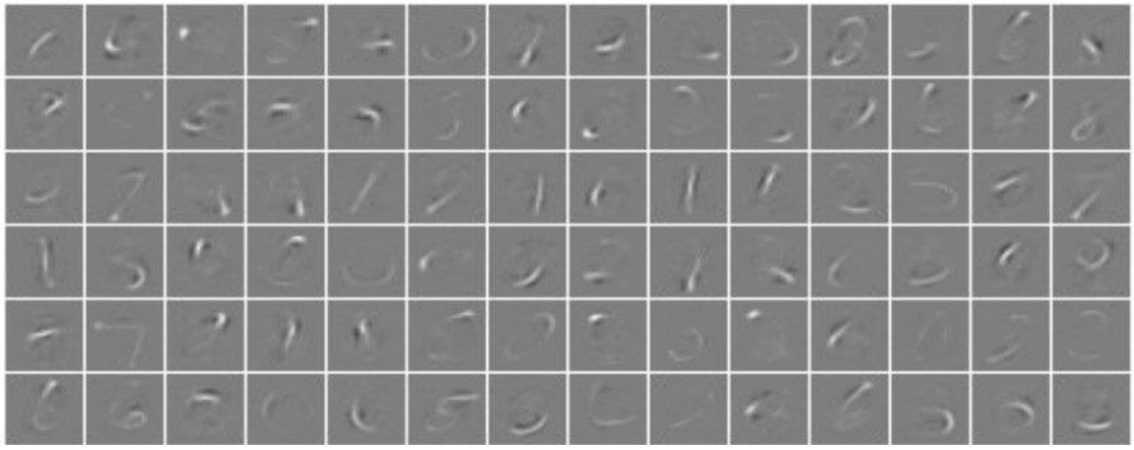


Figure 2-2: Activations taken from the first convolutional layer.

- As we glide through the input, the scalar product is calculated for each value in the kernel (filter). the network will learn kernels that fired when they see a specific feature at given spatial position of the input and it known as activations. this figure show how networks will learn kernels:

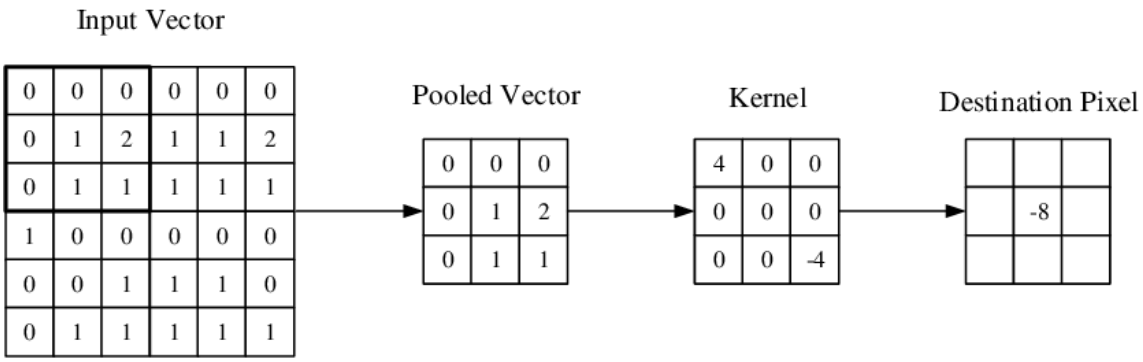


Figure 2-3: A visual representation of convolutional layer.

- the training of ANNs on input such as images results in models of which are too big to train effectively this comes down to fully connected manner of standard ANN neurons, so mitigate against this every neuron in convolutional layer is connected only with small region of the input .the dimensionality of region is commonly referred to as the receptive field size of the neuron .the magnitude of connectivity through the depth is nearly equal to the depth of the input . for example if the input

image is 3*64*64 to the network where 3 is the number of color channels and 128 is height and width of the image and we set receptive field size as 6*6 we would have total of 108 weights(kernel width*kernel height*number of color channels) within the convolutional layer to put this into perspective, a standard neuron seen in other forms of the ANN would contain 12,288 weight each .

- The depth of the output volume produced by the convolutional layers can be manually set through the number of neurons within the layer to the same region of the input .this can be seen with other forms of ANNs, where the all of the neurons in the hidden layer are directly connected to every single neuron . Reducing this hyperparameter can minimize the total number of neurons of the network but it can reduce the pattern recognition capabilities of the model. we are able to define of the stride in which we set the depth around the spatial dimensionality of the input in order to place receptive field. for example, if the stride set to be 1 then there will be high overlapping produce very large activation. on the side, setting the stride to great number will reduce the amount of overlapping and produce output of lower spatial dimensions.

- Zero padding is the simple process of padding the border of the input and it's an effective method to give further control as the dimensionality of the output volumes. It is important to understand that through using these techniques, we will alter the spatial dimensionality of the convolutional layers output to calculate this, we can this formula:

$$\mathbf{(V-R) + (2Z/S) + 1}$$

- Sharing parameters works on the assumption that if one region feature is useful to compute at set spatial region, then it is likely to be useful in another region. if we constraint each individual activation map within the output volume to the same weights and bias, then we will see massive reduction in the number parameters being produced by convolutional layer. As result of this as the backpropagation stage occurs, each neuron in the output will represent the overall gradient of which can be totaled across the depth that is the only updating a single set of weights as opposed to every single one.

- Another CNN architecture is to stack two convolutional layers before each pooling layer. this is strongly recommended as stacking multiple convolutional layers allows for more complex features of the input vector to be selected. this figure shows another common CNN architecture:

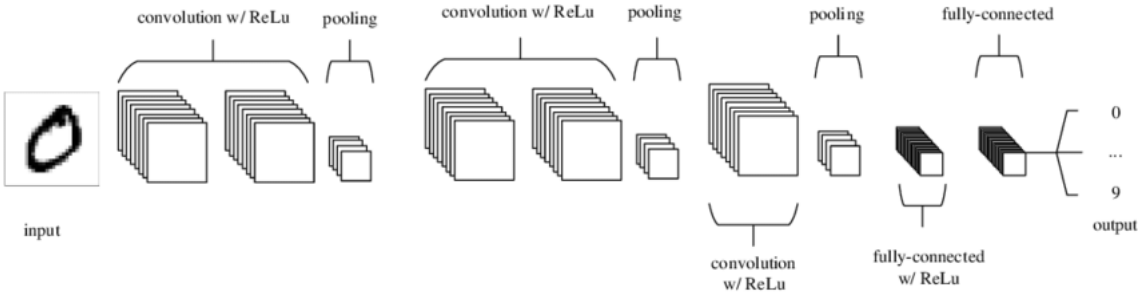


Figure 2-4: A common form of CNN architecture.

- Then the results it is clear that convolutional neural networks differ to other forms of ANN in that instead of focusing on the entirely the problem domain, knowledge about the specific type of input is exploited. This in turn allows for much simpler network architecture to be set up.

- On the next Chapter we will discuss the section of "*Methodology*".

Chapter Three

3 Methodology

3.1 Programming Languages:

- ***Python Programming Languages***: Is the main language of our project as we work on deep learning.
- (CSS - XML - PYQT5) ➔ GUI

3.2 Frameworks & Libraries Used:

- ***NumPy***➔ Is a library used for multi-dimensional matrices and arrays, along with a large collection of high-level mathematical functions to operate on these arrays.
- ***Pandas***➔ Is a library used for data manipulation and analysis.
- ***Matplotlib***➔ It provides an object-oriented API for embedding plots into applications.
- ***Torch***➔ Is an open-source ML library, a logical processing system, and a content language dependent on the Lua programming language.
- ***MaxPooling***➔ For modeling the Pooling layers.
- ***Dense***➔ For modeling the Dense layers.
- ***Conv***➔ For modeling the convolutional layers.
- ***OS***➔ For accessing the device directories.

3.3 Environments Used:

- ***Colab***➔ Is a Python development environment that runs in the browser using Google Cloud.

- ***Anaconda*** ➔ As a launcher environment and for installing important libraries and software and run it.

- ***Python IDE*** ➔ It was one of the basic environments used in project programming. Because of its very important libraries and very easy tools. And easy to import from any other languages and environments.

- ***Jupyter Notebook*** ➔ It is the best and most familiar console for running the neural networks models. On it you can run Python Programming and import more libraries.

- ***Spyder*** ➔ It is an embedded program into the anaconda program. It is a complete python environment to and you can run it separated.

- ***PyCharm*** ➔ Is an integrated development environment IDE used in computer programming, specifically for the Python language. Our full project was running on it.

- ***Github*** ➔ is a United States-based global company that provides hosting for software development and version control using Git.

3.4 MRI Knee Dataset:

Dataset is split in training set (1130 cases), and validation set (120 cases) and is organized as follows:

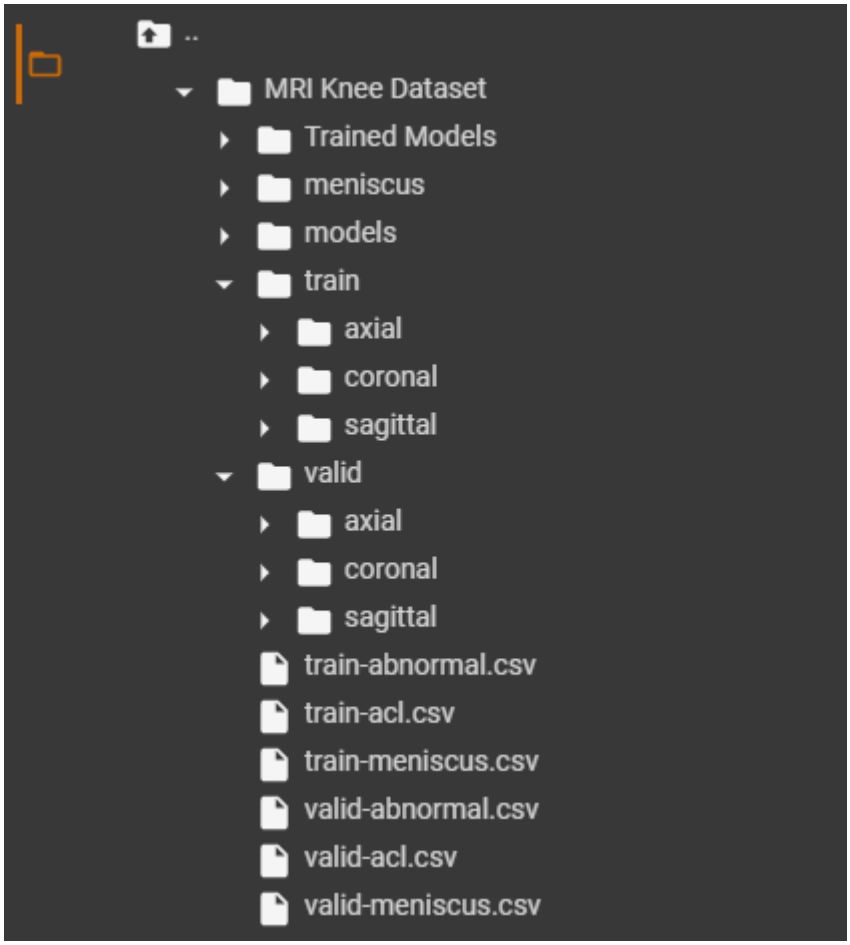


Figure 3-1: Dataset.

- Each case in both train and valid folder has 3 MRI scans taken from different planes (axial / coronal / sagittal).
- (train-abnormal.csv / train-acl.csv / train-meniscus.csv) indicate the labels of each train case.
- (valid-abnormal.csv / valid-acl.csv / valid-meniscus.csv) indicate the labels of each valid case.
- To make a proper decision regarding a case, the radiologist usually look at MRI scans from different planes in order to have a global view.

3.5 Explore the MRI Knee Dataset:

1. The csv files (train-abnormal.csv / train-acl.csv / train-meniscus.csv) have the same structure with two columns:
   - Case: is the ID of the case.
   - Abnormal: is a (0/1) value indicate the tears (ACL / Abnormal / Meniscus).

As we're going to show.

```
import numpy as np
import pandas as pd

train_acl = pd.read_csv('train-acl.csv' , header=None ,
                        names=['Case' , 'Abnormal'] ,
                        dtype={'Case':str , 'Abnormal': np.int64})
print(train_acl.head())

#output
    Case   Abnormal
0   0000      0
1   0001      1
2   0002      0
3   0003      0
4   0004      0
```

Train_acl.csv file

```
import numpy as np
import pandas as pd

train_men = pd.read_csv(' train-meniscus.csv' , header=None ,
                        names=['Case' , 'Abnormal'] ,
                        dtype={'Case':str , 'Abnormal': np.int64})
print(train_men.head())

#output
    Case   Abnormal
0   0000      0
1   0001      1
2   0002      0
3   0003      1
4   0004      0
```

Train_meniscus.csv file

```
import numpy as np
import pandas as pd

train_abnormal = pd.read_csv(' train-abnormal.csv' , header=None ,
                        names=['Case' , 'Abnormal'] ,
                        dtype={'Case':str , 'Abnormal': np.int64})
print(train_abnormal.head())

#output
    Case   Abnormal
0   0000      1
1   0001      1
2   0002      1
3   0003      1
4   0004      1
```

Train_abnormal.csv file

2. And each file contains 1130 case.

```
print('ACL Shape: ' , train_acl.shape)
print('Meniscus Shape: ' , train_acl.shape)
print('Abnormal Shape: ' , train_acl.shape)

#output
ACL Shape: (1130 , 2)
Meniscus Shape: (1130 , 2)
Abnormal Shape: (1130 , 2)
```

3. The csv files (valid-abnormal.csv / valid-acl.csv / valid-meniscus.csv) have the same structure with two columns:
   - Case: is the ID of the case.
   - Abnormal: is a (0/1) value indicate the tears (ACL / Abnormal / Meniscus).

As we're going to show one example of them.

```
import numpy as np
import pandas as pd

valid_acl = pd.read_csv('valid-acl.csv' , header=None ,
                        names=['Case' , 'Abnormal'] ,
                        dtype={'Case':str , 'Abnormal': np.int64})
print(valid_acl.head())
print(valid_acl.shape)

#output
    Case   Abnormal
0   1130      0
1   1131      0
2   1132      0
3   1133      0
4   1134      0

(120 , 2)
```

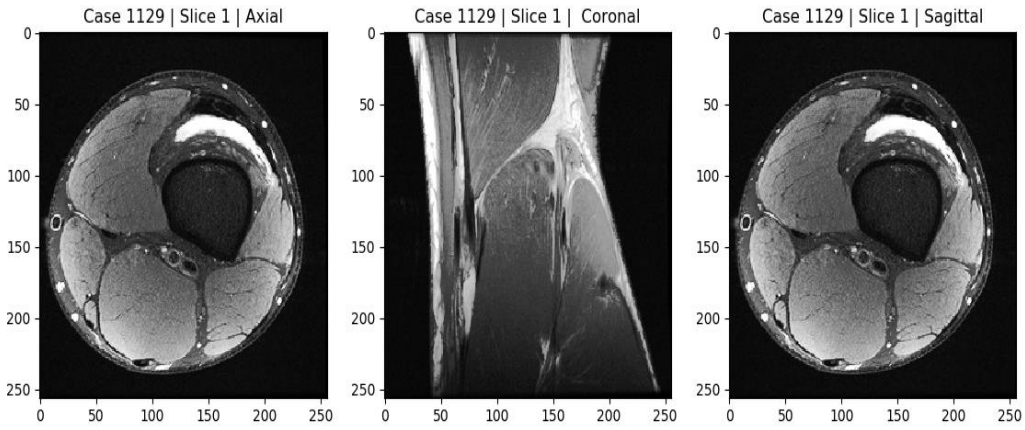
valid_acl.csv file & shape

4. Each MRI scan is a tensor of n **slices** and each slice is a gray scale image of size (256,256).

```
import numpy as np
import pandas as pd

case = '1129'

mri_axial = np.load('1129.npy')
mri_coronal = np.load('1129.npy')
mri_sagittal = np.load('1129.npy')

print(mri_axial.shape)
print(mri_coronal.shape)
print(mri_sagittal.shape)

#output
(43 , 256 , 256)
(35 , 256 , 256)
(34 , 256 , 256)
```

3.6 Visualize the First Slice Of Each Plane On Case 1129:

```
import matplotlib.pyplot as plt

fig , (ax1 , ax2 , ax3) = plt.subplots(1 , 3 , figsize=(15 , 5))

ax1.imshow(mri_axial[0 , : , : ] , 'gray')
ax1.set_title('Case 1129 | Slice 1 | Axial')

ax2.imshow(mri_coronal[0 , : , : ] , 'gray')
ax2.set_title('Case 1129 | Slice 1 |  Coronal')

ax3.imshow(mri_axial[0 , : , : ] , 'gray')
ax3.set_title('Case 1129 | Slice 1 | Sagittal')

plt.show()
```



Output of case 1129

3.7 Some Consideration About the Data:

The slices are significantly different from a plane to another. This is the first thing we noticed as a non-specialist.

Within a given plane, the slices may substantially differ as well. In fact, and we'll see it later, some slices can better highlight an (ACL, Abnormal, Meniscus) tear.

3.8 Building I MRI-Scan Model:

We are going to use MRI dataset to build a Convolutional Neural Network (CNN) that detects and classifies each tear from MRI scans. We will implement it in **PyTorch** to fully take advantage of the capabilities of this framework.

❖ **Note that:**

1. Each patient has 3 MRI scans captured on 3 distinct planes.
2. Each MRI scan has a different number of slices.
3. We are going to build 3 independent CNN models that allow to classify each disease.
4. Each of these networks will specialize in detecting three planes (axial, coronal, and sagittal).

3.8.1 Training procedure:

❖ The training of the 3 models is done through the minimization of the cross-entropy loss using **Adam** optimizer.

❖ During training the gradient of the loss is computed on each training example using the **backpropagation algorithm** and the network parameters are then adjusted in the opposite direction of the gradient.

❖ During training some geometric transformations are applied on the input MRI scans. These transformations are label-invariant. They are meant to bring diversity in the dataset and **increase** the **stability** of the model while **decrease** its tendency to **overfitting**. This procedure is called **data augmentation**.

3.8.2 Code structure:

We are organized the source code in three main classes.

❖ Model class:

a. Define the model architecture in the forward method.
b. Performs transfer learning by loading pre-trained CNN model.

- ❖ DataLoader class:
  a. Load the MRI dataset and applies data augmentation.
- ❖ Training class:
  a. Instantiates the model, the dataloaders on train and validation sets, the optimizer (Adam), and the loss.
  b. Loops over train data to update model's weights.
  c. Loops over validation data to evaluate the model's performance.

3.8.3 The model class architecture and code:

```
import torch
import torch.nn as nn
from torchvision import models

class MRI_alex(nn.Module):

    def __init__(self, training=True):
        super().__init__()
        self.axial_net = models.alexnet(pretrained=training)
        self.sagit_net = models.alexnet(pretrained=training)
        self.coron_net = models.alexnet(pretrained=training)

        self.gap_axial = nn.AdaptiveAvgPool2d(1)
        self.gap_sagit = nn.AdaptiveAvgPool2d(1)
        self.gap_coron = nn.AdaptiveAvgPool2d(1)
        self.classifier = nn.Linear(3*256, 1)

    def forward(self,vol_axial, vol_sagit, vol_coron):
        vol_axial = torch.squeeze(vol_axial, dim=0)
        vol_sagit = torch.squeeze(vol_sagit, dim=0)
        vol_coron = torch.squeeze(vol_coron, dim=0)

        vol_axial = self.axial_net.features(vol_axial)
        vol_sagit = self.sagit_net.features(vol_sagit)
        vol_coron = self.coron_net.features(vol_coron)

        vol_axial = self.gap_axial(vol_axial).view(vol_axial.size(0), -1)
        x = torch.max(vol_axial, 0, keepdim=True)[0]

        vol_sagit = self.gap_sagit(vol_sagit).view(vol_sagit.size(0), -1)
        y = torch.max(vol_sagit, 0, keepdim=True)[0]

        vol_coron = self.gap_coron(vol_coron).view(vol_coron.size(0), -1)
        z = torch.max(vol_coron, 0, keepdim=True)[0]

        w = torch.cat((x, y, z), 1)
```

```
        out = self.classifier(w)
        return out
```

We define the model as a class called MRI_alex that inherits from the torch.nn.Modules class
In the constructor:

- We define 3 objects from the pretrained **AlexNet** model*.
- 3 objects from the pooling layer.
- The dense layer that acts as a classification layer.

* we use 3 objects because we have 3 planes (axial plane / coronal plane / sagittal plane).

***AlexNet** model:

AlexNet is an incredibly powerful model capable of achieving high accuracies on very challenging datasets. However, removing any of the convolutional layers will drastically degrade AlexNet's performance. AlexNet is a leading architecture for any object-detection task and may have huge applications in the computer vision sector of artificial intelligence problems. In the future, AlexNet may be adopted more than CNNs for image tasks.

The **AlexNet** model is a convolution neural network created by Alex Krizhevsky, Geoffrey Hinton and Ilya Sutskever. They trained their network on 1.2 million high-resolution images into 1000 different classes with 60 million parameters and 650,000 neurons. The training was done on two GPUs with split layer concept because GPUs were a little bit slow at that time.

The AlexNet architecture consists of five convolutional layers, some of which are followed by maximum pooling layers and then three fully connected layers and finally a 1000-way softmax classifier.

In the original paper, all the layers are divided into two to train them on separate GPUs. Since it is a complex arrangement and difficult to understand, we will implement AlexNet model in one-layer concept.

In the forward method, we write the forward pass that operations the network performs on the input until it computes the predictions.

3.8.4 The DataLoader class:

We define a custom Dataset object that load the MRI data in the main program. using Map-style datasets. To create the **dataloader**, we define a class called **Dataset** that inherits from the class: ***torch.utils.data.Dataset.***

```
class Dataset(data.Dataset):
    def __init__(self, datadir, tear_type, use_gpu , transform=None):
        super().__init__()
        self.use_gpu = use_gpu
        label_dict = {}
        self.paths = []
        self.transform = transform

        abnormal_label_dict = {}
        if datadir[-1]=="/":
            datadir = datadir[:-1]
        self.datadir = datadir

        for i,line in enumerate(open(datadir+''+tear_type+'.csv').readlines()):
            line = line.strip().split(',')
            filename = line[0]
            label = line[1]
            label_dict[filename] = int(label)

        for i,line in enumerate(open(datadir+''+"abnormal"+'.csv').readlines()):
            line = line.strip().split(',')
            filename = line[0]
            label = line[1]
            abnormal_label_dict[filename] = int(label)

        for filename in os.listdir(os.path.join(datadir, "axial")):
            if filename.endswith(".npy"):
                self.paths.append(filename)

        self.labels = [label_dict[path.split(".")[0]] for path in self.paths]
        self.abnormal_labels = [abnormal_label_dict[
                                path.split(".")[0]] for path in self.paths]

        if tear_type != "abnormal":
           temp_labels = [self.labels[i] for i in range(
                  len(self.labels)) if self.abnormal_labels[i]==1]
```

```
            neg_weight = np.mean(temp_labels)
        else:
            neg_weight = np.mean(self.labels)

        self.weights = [neg_weight, 1 - neg_weight]


    def weighted_loss(self, prediction, target):
        weights_npy = np.array([self.weights[int(t[0])] for t in target.data])
        weights_tensor = torch.FloatTensor(weights_npy)
        if self.use_gpu:
            weights_tensor = weights_tensor.cuda()
        loss = F.binary_cross_entropy_with_logits(prediction, target,
                                  weight=Variable(weights_tensor))
        return loss


    def __getitem__(self, index):
        filename = self.paths[index]
        vol_axial = np.load(os.path.join(self.datadir, "axial", filename))
        vol_sagit = np.load(os.path.join(self.datadir, "sagittal", filename))
        vol_coron = np.load(os.path.join(self.datadir, "coronal", filename))


        # axial
        pad = int((vol_axial.shape[2] - INPUT_DIM)/2)
        vol_axial = vol_axial[:,pad:-pad,pad:-pad]

        vol_axial = preprocess(vol_axial)

        vol_axial_tensor = torch.FloatTensor(vol_axial)

        # sagittal
        pad = int((vol_sagit.shape[2] - INPUT_DIM)/2)
        vol_sagit = vol_sagit[:,pad:-pad,pad:-pad]

        vol_sagit = preprocess(vol_sagit)

        vol_sagit_tensor = torch.FloatTensor(vol_sagit)

        # coronal
        pad = int((vol_coron.shape[2] - INPUT_DIM)/2)
        vol_coron = vol_coron[:,pad:-pad,pad:-pad]

        vol_coron = preprocess(vol_coron)

        vol_coron_tensor = torch.FloatTensor(vol_coron)
```

```
        label_tensor = torch.FloatTensor([self.labels[index]])


        return vol_axial_tensor, vol_sagit_tensor, vol_coron_tensor,
               label_tensor , self.abnormal_labels[index]


    def __len__(self):
        return len(self.paths)
```

- In the constructor of MRIDataset we define a set of arguments:
  - ✓ **datadir**: is the path of the dataset, we load the dataset on Drive and mount it by colab.
  - ✓ **tear_type**: either ACL, Abnormal, Meniscus.
  - ✓ **use_gpu**: is a Boolean value to use GPU or not.
  - ✓ **transform**: the series of data augmentation operations. If none, no data augmentation.

- In the remaining part of the constructor, we prepare the paths, the labels, and the weights that correspond to each data sample.
- In the __getitem__ function we return the MRI-scan .npy file, the labels, and the weight after applying minor preprocessing and eventual data augmentation. We divided it into three parts (axial / coronal / sagittal) each part load and preprocess the data.

We did some data processing as normalizing the data, and standardization as follow:

```
INPUT_DIM = 224
MAX_PIXEL_VAL = 255
MEAN = 58.09
STDDEV = 49.73


def preprocess(series):
        series = torch.tensor(np.stack((series,)*3, axis=1))
```

```
        series = np.dot(np.divide((series - series.min()) ,
                        (series.max() - series.min())) , MAX_PIXEL_VAL)
        series = np.divide((series - MEAN) , STDDEV)
        return series
```

- Having a feature on a similar scale can help the gradient descent coverage more quickly towards the minima. So, we use Normalization and Standardization techniques.
- Normalization: is a scaling technique in which values are shifted and re-scaled so that they end up ranging between 0 and 1. it also known as Min-Max scaling.
- Normalization is good to use when you know that the distribution of your data does not follow a Gaussian distribution. This can be useful in algorithms that do not assume any distribution of the data like K-Nearest Neighbors and Neural Networks.
- Standardization: is another scaling technique where the values are centered around the mean with a unit standard deviation. This means that the mean of the attributes becomes 0, and the resultant distribution has a unit standard deviation.
- Standardization can be helpful in cases where the data follows s Gaussian distribution. However, this does not have to be necessarily true. Also, unlike normalization, standardization does not have a bounding range. So, even if you have outliers in your data, they will not be affected by standardization.

```
def __len__(self):
        return len(self.paths)
```

In __ len __ function, that return the length of the data.

3.8.5 let's start loading the data:

```
def load_data(task, use_gpu):
    train_dir = "/content/drive/My Drive/MRI Knee Dataset/train"
    valid_dir = "/content/drive/My Drive/MRI Knee Dataset/valid"

    train_dataset = Dataset(train_dir, task, use_gpu  )
    valid_dataset = Dataset(valid_dir, task, use_gpu )

    train_loader = data.DataLoader(train_dataset, batch_size=1,
                                   num_workers=11, shuffle=True)
    valid_loader = data.DataLoader(valid_dataset, batch_size=1,
                                   num_workers=11, shuffle=False)
```

```
    return train_loader, valid_loader
```

These datasets are now passed to a **DataLoader** which a handy **PyTorch** object that allows to efficiently iterate over the data by leveraging batching, shuffling, multiprocessing, and data augmentation.

In this function, we select train and valid data paths.

Then we define the data augmentation pipeline:

```
from torchvision import transforms
import torchvision.transforms.functional as TF
from torchsample.transforms import RandomRotate,
     RandomTranslate, RandomFlip, ToTensor, Compose, RandomAffine


train_transform = transforms.Compose([
            transforms.ToPILImage(),
            transforms.RandomHorizontalFlip(),
            transforms.RandomAffine(25, translate = (0.1, 0.1)),
            transforms.ToTensor()
        ])
```

Augmentation process

→ ***CLASS torchvision.transforms.Compose(transforms)***

Compose several transforms together.

→ ***CLASS torchvision.transforms.ToPILImage(mode=None)***

Convert a Tensor or numpy array to PIL image.

→ ***CLASS torchvision.transforms.RandomHorizontalFlip( )***

Horizontally flip the given image randomly with a given probability.

→ ***CLASS torchvision.transforms.RandomAffine( )***

Random Affine transformation of the image keeping center invariant.

→ ***CLASS torchvision.transforms.ToTensor( )***

Convert a PIL image or numpy array to Tensor.

3.8.6 First function in the training process

```
def run_model(model,loader,train=False,optimizer=None,abnormal_model_path=None):
    preds = []
    labels = []

    if train:
        model.train()
    else:
        if abnormal_model_path:
            abnormal_model = MRI_alex()
            state_dict = torch.load(abnormal_model_path)
            abnormal_model.load_state_dict(state_dict)
            abnormal_model.cuda()
            abnormal_model.eval()
        model.eval()

    total_loss = 0.0
    num_batches = 0

    for batch in tqdm(loader):
        vol_axial, vol_sagit, vol_coron, label, abnormal = batch
        if train:
            if abnormal_model_path and not abnormal:
                continue
            optimizer.zero_grad()
        if loader.dataset.use_gpu:
            vol_axial,vol_sagit,vol_coron = vol_axial.cuda(),
                                            vol_sagit.cuda(), vol_coron.cuda()

            label = label.cuda()
            vol_axial,vol_sagit,vol_coron =Variable(vol_axial) ,
                                    Variable(vol_sagit) , Variable(vol_coron)
        label = Variable(label)

        logit = model.forward(vol_axial, vol_sagit, vol_coron)

        loss = loader.dataset.weighted_loss(logit, label)
        total_loss += loss.item()

        pred = torch.sigmoid(logit)

        pred_npy = pred.data.cpu().numpy()[0][0]

        if abnormal_model_path and not train:
            abnormal_logit = abnormal_model.forward(
```

```
                                        vol_axial,vol_sagit,vol_coron)
            abnormal_pred = torch.sigmoid(abnormal_logit)
            abnormal_pred_npy = abnormal_pred.data.cpu().numpy()[0][0]
            pred_npy = pred_npy * abnormal_pred_npy


        label_npy = label.data.cpu().numpy()[0][0]


        preds.append(pred_npy)
        labels.append(label_npy)


        if train:
            loss.backward()
            optimizer.step()
        num_batches += 1


    avg_loss = total_loss / num_batches

    fpr, tpr, threshold = metrics.roc_curve(labels, preds)
    auc = metrics.auc(fpr, tpr)


    if abnormal_model_path and not train:
        del abnormal_model


    return avg_loss, auc, preds, labels
```

run_model function

We start by setting the model to train model, and initialize the lists that will contain predictions, true labels, and the losses on each individual sample.

Then we loop over the dataloader. At each step:

- A single MRI scan and its label and weight are passed to GPU.
- The network computes a forward pass on the MRI scan which results in a prediction.
- The loss between the prediction and the true label is computed.
- Backpropagation of the loss, computation of the gradients.

❖ Weights update by the optimizer.

3.8.7 Train function:

```
def train(rundir, task, epochs, learning_rate, use_gpu,model,
          abnormal_model_path=None):


    train_loader, valid_loader = load_data(task, use_gpu)

    model = model
    if use_gpu:
        model = model.cuda()


    optimizer = torch.optim.Adam(model.parameters(),
                     learning_rate, weight_decay=0.01)
    scheduler = torch.optim.lr_scheduler.ReduceLROnPlateau(
                    optimizer, patience=5, factor=0.3, threshold=1e-4)


    best_val_loss = float('inf')


    start_time = datetime.now()


    for epoch in range(epochs):
        change = datetime.now() - start_time
        print('starting epoch {}. time passed: {}  : lr = {} '.format(
                                    epoch+1, str(change) , learning_rate))

        train_loss, train_auc, _, _ = run_model(model,train_loader, train=True,
                   optimizer=optimizer , abnormal_model_path=abnormal_model_path)


        print(f'train loss: {train_loss:0.4f}')
        print(f'train AUC: {train_auc:0.4f}')


        val_loss, val_auc, _, _ = run_model(model, valid_loader ,
                                abnormal_model_path=abnormal_model_path)

        print(f'valid loss: {val_loss:0.4f}')
        print(f'valid AUC: {val_auc:0.4f}')
```

```
        scheduler.step(val_loss)
```

In train function we pass:

→ run_dir: path the folder of data.

→ task: choose (ACL / Abnormal / Meniscus)

→ Epochs: the number of epochs to train for.

→ Learning rate: Learning rate is a hyper-parameter that controls how much we are adjusting the weights of our network with respect the loss gradient.

We used 1***e-05*** as a default value.

→ Use_gpu: true or false.

→ Model: MRI_alex class.

Then we defined the optimizer and schedular:

```
optimizer = torch.optim.Adam(model.parameters(),
                      learning_rate, weight_decay=0.01)
scheduler = torch.optim.lr_scheduler.ReduceLROnPlateau(
                     optimizer, patience=5, factor=0.3, threshold=1e-4)
```

Then we start the training

```
Downloading: "https://download.pytorch.org/models/alexnet-owt-4df8aa71.pth" to /root/.cache/torch/ch
100% 233M/233M [00:05<00:00, 45.9MB/s]

  0%|          | 0/1130 [00:00<?, ?it/s]starting epoch 1. time passed: 0:00:00.000012  : lr = 1e-05
100%|██████████| 1130/1130 [04:33<00:00,  4.13it/s]
train loss: 0.2359
train AUC: 0.6482
100%|██████████| 120/120 [00:34<00:00,  3.48it/s]
  0%|          | 0/1130 [00:00<?, ?it/s]valid loss: 0.2847
valid AUC: 0.8841
starting epoch 2. time passed: 0:05:25.396414  : lr = 1e-05
  2%|▏         | 20/1130 [00:08<05:48,  3.19it/s]
```

First epoch in train ACL model

3.8.8 Function to Evaluate the models:

```
def evaluate(split, model_path, diagnosis, use_gpu):
    train_loader, valid_loader = load_data(diagnosis, use_gpu)


    model = MRI_alex()
    state_dict = torch.load(model_path,map_location(None if use_gpu else 'cpu'))
    model.load_state_dict(state_dict)


    if use_gpu:
        model = model.cuda()


    if split == 'train':
        loader = train_loader
    elif split == 'valid':
        loader = valid_loader
    else:
        raise ValueError("split must be 'train', 'valid', or 'test'")


    loss, auc, preds, labels = run_model(model, loader)


    print(f'{split} loss: {loss:0.4f}')
    print(f'{split} AUC: {auc:0.4f}')


    return preds, labels
```

- In this function we use it to evaluate each model after training it.
- This function call load data functions then split data.
- After split the data call run_model function that return loss and accuracy then print them.

3.9 System Implementation:

❖ Graphical User Interface done by XML and CSS

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>965</width>
    <height>582</height>
   </rect>
  </property>
  <property name="sizePolicy">
   <sizepolicy hsizetype="Minimum" vsizetype="Minimum">
    <horstretch>0</horstretch>
    <verstretch>0</verstretch>
   </sizepolicy>
  </property>
  <property name="windowTitle">
   <string>IMRI Scanner</string>
  </property>
  <property name="windowIcon">
   <iconset>
    <normaloff>../resources/icons8-neuroscience-experimer
    50.png</iconset>
  </property>
  <property name="styleSheet">
   <string notr="true">QPushButton
{
border: 0px solid lightgrey;
color:white;
}
QPushButton:hover
{
border: 1px solid rgb(0, 120, 210);
}
```

Figure 3-2: GUI XML and CSS tags.

❖ Main Activity:

MainApp is the main class to customize GUI.

```
MainUI, _ = loadUiType(r'GUI\main.ui')

class MainApp(QMainWindow, MainUI):

    def __init__(self,parent= None):
        super(MainApp, self).__init__(parent)
        QMainWindow.__init__(self)
        self.setupUi(self)

        self.setFixedWidth(965)
        self.setFixedHeight(585)

        self.db = DB()

        self.handle_buttons()
        self.handle_table_tab()

        self.threadpool = QThreadPool()
        self.abnormal, self.acl, self.men = 0,0,0
        self.dialog = None
        self.extract_dialog = None
        return
```

❖ It inherit from PyQt5.QtWidgets.QMainWindow and class which connected to GUI using PyQt5.uic.loadUiType.

❖ At constructor

- ✓ we define a set of arguments.
- ✓ we set fixed size to main frame.
- ✓ we get object from created database class DB.
- ✓ we handle GUI buttons signals and slots.
- ✓ we handle table tab to create columns and search group signal and slots.
- ✓ we get object from PyQt5.QtCore.QThreadPool to handle GUI threads,
- ✓ we set default values for model predictions.
- ✓ we identify edit_dialog which edit patient data.
- ✓ we identify extract_dialog which extract patient data.

❖ Signal and Slots

- ☒ Signals and slots are used for communication between objects, when we change one widget, we often want another widget to be notified or action to be done.
- ☒ A signal is emitted when a particular event occurs.

☒ A slot is a function that is called in response to a particular signal.

❖ Buttons Signal and Slots

```
def handle_buttons(self):
    self.btn_axial.clicked.connect(lambda x: self.get_images(0))
    self.btn_coronal.clicked.connect(lambda x: self.get_images(1))
    self.btn_sagittal.clicked.connect(lambda x: self.get_images(2))
    self.btn_examine.clicked.connect(self.examine)
    self.tabWidget.currentChanged.connect(lambda x: self.update_table(
        self.db.get_all_data(50)))
    self.tabWidget.setTabEnabled(2, False)

    self.tableWidget.cellDoubleClicked.connect(self.cell_clicked)
    self.menu_excel_file.triggered.connect(self.extract_xlsx)
    self.tabWidget.setCurrentIndex(0)
```

❖ When (axial, coronal, sagittal) buttons clicked the slot get_images will be implemented.

```
def get_images(self, key):

    loc = QFileDialog.getOpenFileNames(self,
            'Open File', directory =
            os.path.join(os.environ["HOMEPATH"], "Desktop"),
            filter='Image (*.jpg *.JPG *.png *.PNG *.jpeg *.JPEG)')
    loc = str(loc[0])[:]
    num_images = len(loc.split(','))
    if key == 0:
        self.axial_images_location = loc
        if loc == '[]': self.lbl_axial_counter.setText('no image selected')
        else: self.lbl_axial_counter.setText('{} images selected'.format(num_images))
    elif key == 1:
        self.coronal_images_location = loc
        if loc == '[]': self.lbl_coronal_counter.setText('no image selected')
        else: self.lbl_coronal_counter.setText('{} images selected'.format(num_images))
    else :
        self.sagittal_images_location = loc
        if loc == '[]': self.lbl_sagittal_counter.setText('no image selected')
        else: self.lbl_sagittal_counter.setText('{} images selected'.format(num_images))
    return
```

❖ In get_images slot we take the image location from user using QFileDialog.

❖ When examine button clicked slot examine will be implemented.

```
def examine(self):
    now = DateTime()

    if self.axial_images_location == '[]'
    or self.coronal_images_location == '[]'
    or self.sagittal_images_location == '[]':
        QMessageBox.warning(self, "Error!", "can\'t load images!")
        return

    if not self.check_text_constrains(self.get_name()):
        QMessageBox.warning(self, "Error!", "name field can't contain (' , \)")
        return

    if not self.check_text_constrains(self.get_note()):
        QMessageBox.warning(self, "Error!", "note field can't contain (' , \)")
        return
    self.btn_examine.setEnabled(False)
    self.lbl_state.setText('in progress...')
    self.th = threads.ExamineThread(
        self.get_thread_data,
        self.axial_images_location,
        self.coronal_images_location,
        self.sagittal_images_location,
        self.get_name(),
        self.get_age(),
        self.get_blood(),
        self.get_note(),
        now.Date() + ' ' + now.AMPM())
    self.th.signals.finished.connect(self.on_thread_finished)
    self.threadpool.start(self.th)
```

- At slot examine:
  - We put constrains to make sure images can be accessed.
  - we put constrains on text field and note field to not contain (‘, “, \).

- Then we send all data fields to the examiner thread.

```
class ExamineSignals(QObject):
    started = pyqtSignal()
    finished = pyqtSignal()
```

- We create two signals to be emitted on start of the thread and on the end of the thread.

- QThreadingPool start the thread by implement run function

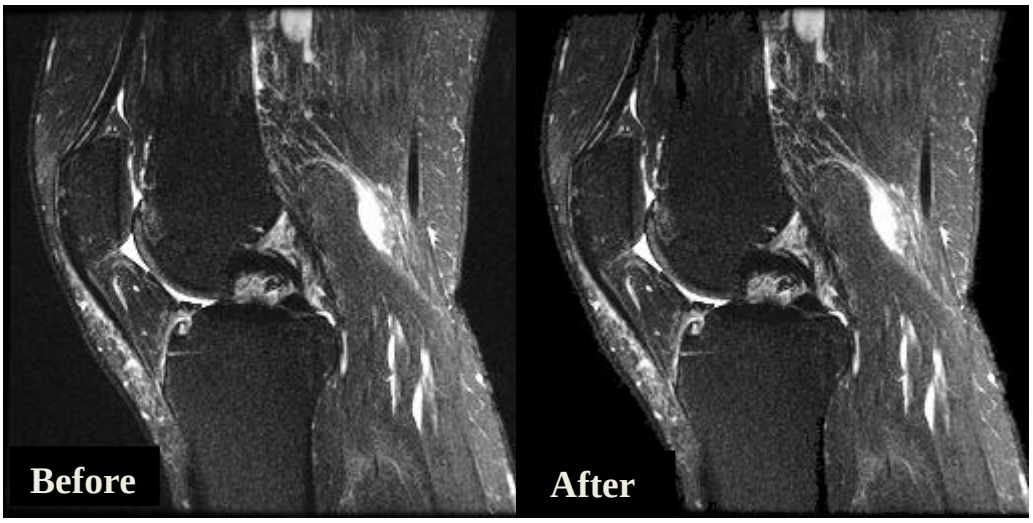
```
def run(self):
    preprocessing(self.axial_images, self.coronal_images, self.sagittal_images)
    self.examine_result_abnormal = Model(key = 'abnormal').get_prediction()
    self.examine_result_acl = Model(key = 'acl').get_prediction()
    self.examine_result_men = Model(key = 'men').get_prediction()

    self.get_data(self.examine_result_abnormal,
                  self.examine_result_acl,
                  self.examine_result_men)

    self.save_to_database()
    self.db.close()
    self.signals.finished.emit()
    return
```

- we do processes on images: image denoise, image resize, convert images to NumPy files.

- image denoise

```
def denoise_image(self, image):
    gray = cv.cvtColor(image,cv.COLOR_BGR2GRAY)
    _,edge = cv.threshold(gray,30,255,cv.THRESH_BINARY)
    contours,_ = cv.findContours(edge,cv.RETR_EXTERNAL,cv.CHAIN_APPROX_SIMPLE)
    contours = sorted(contours,key=cv.contourArea,reverse=True)[0]
    mask = np.zeros(image.shape[:2],np.uint8)
    mask = cv.drawContours(mask,[contours],-1,(255),-1)
    res = cv.bitwise_and(image,image,mask=mask)
    return res
```

- We focus on the main part of image and delete remain elements in image as example:



- image resize

```
def resize_image(self, img):
    pixles = 256
    if img.shape[0] > pixles or img.shape[1] > pixles:
        output = cv.resize(img, (pixles,pixles), interpolation=cv.INTER_CUBIC)
    else:
        output = cv.resize(img, (pixles,pixles), interpolation=cv.INTER_LINEAR)
    return output
```

- We convert image resolution to be 256x256

- convert images to NumPy files

```
def to_numpy(self, images, file_name):
    if not os.path.exists('temp'):
        os.mkdir('temp')
    result = []
    for i in images:
        i = i.strip()
        i = i[1:-1]
        temp = cv.imread(i)
        temp = self.denoise_image(temp)
        img = self.resize_image(temp)
        data = np.array(img, dtype='uint8')
        result.append(data[:,:,0])
    np.save(r'temp/{}'.format(file_name), result)
    return
```

- After remove noise from image and resize it we append all images from the same type in NumPy file.

- Then we get examine result for (Abnormal, ACL, Meniscus) from Model.get_prediction().

  - Mapping between GUI and the trained model:

```
def __init__(self, key = 'abnormal'):
    self.INPUT_DIM = 224
    self.MAX_PIXEL_VAL = 255
    self.MEAN = 58.09
    self.STDDEV = 49.73
    self.model_ab=MRI_alex(False)

    if key == 'abnormal':
        self.model_ab.load_state_dict(torch.load(r"models/abnormal.pt", map_location='cpu'))
    elif key =='acl':
        self.model_ab.load_state_dict(torch.load(r"models/acl.pt", map_location='cpu'))
    else:
        self.model_ab.load_state_dict(torch.load(r"models/men.pt", map_location='cpu'))
    self.model_ab.cuda()
```

```
def preprocess(self, series):
    pad = int((series.shape[2] - self.INPUT_DIM)/2)
    series = series[:,pad:-pad,pad:-pad]
    series = (series-np.min(series))/(np.max(series)-np.min(series))*self.MAX_PIXEL_VAL
    series = (series - self.MEAN) / self.STDDEV
    series = np.stack((series,)*3, axis=1)
    series_float = torch.FloatTensor(series)
    return series_float
```

```
def study(self, axial_path, sagit_path, coron_path):
    vol_axial = np.load(axial_path)
    vol_sagit = np.load(sagit_path)
    vol_coron = np.load(coron_path)
    vol_axial_tensor = self.preprocess(vol_axial)
    vol_sagit_tensor = self.preprocess(vol_sagit)
    vol_coron_tensor = self.preprocess(vol_coron)
    return {"axial": vol_axial_tensor,
            "sagit": vol_sagit_tensor,
            "coron": vol_coron_tensor}
```

```
def predict(self, model, tensors, abnormality_prior=None):
    vol_axial = tensors["axial"].cuda()
    vol_sagit = tensors["sagit"].cuda()
    vol_coron = tensors["coron"].cuda()
    vol_axial = Variable(vol_axial)
    vol_sagit = Variable(vol_sagit)
    vol_coron = Variable(vol_coron)
    logit = model.forward(vol_axial, vol_sagit, vol_coron)
    pred = torch.sigmoid(logit)
    pred_npy = pred.data.cpu().numpy()[0][0]

    if abnormality_prior:
        pred_npy = pred_npy * abnormality_prior
    return pred_npy
```

```
def get_prediction(self):
    self.predict(self.model_ab, self.study(axial_path, coronal_path, sagittal_path))
```

- Then we save the result data to database and send a signal to be executed after thread has been finished, this signal emitted using on_thread_finished.

```
def on_thread_finished(self):
    report = '''
    abnormal\t: {}
    acl\t\t: {}
    menalusis\t: {}
    '''.format(
        self.get_text_result(self.abnormal),
        self.get_text_result(self.acl),
        self.get_text_result(self.men))
    self.txt_report.setPlainText(report)
    self.lbl_state.setText('Done, state saved successfully!')
    self.btn_examine.setEnabled(True)
    self.reset_fieldes()
```

❖ Print data for user at the report text box.

```
class MRI_alex(nn.Module):
    def __init__(self, training=True):
        super().__init__()
        self.axial_net = models.alexnet(pretrained=training)
        self.sagit_net = models.alexnet(pretrained=training)
        self.coron_net = models.alexnet(pretrained=training)
        self.gap_axial = nn.AdaptiveAvgPool2d(1)
        self.gap_sagit = nn.AdaptiveAvgPool2d(1)
        self.gap_coron = nn.AdaptiveAvgPool2d(1)
        self.classifier = nn.Linear(3*256, 1)
        return

    def forward(self,vol_axial, vol_sagit, vol_coron):
        vol_axial = torch.squeeze(vol_axial, dim=0)
        vol_sagit = torch.squeeze(vol_sagit, dim=0)
        vol_coron = torch.squeeze(vol_coron, dim=0)
        vol_axial = self.axial_net.features(vol_axial)
        vol_sagit = self.sagit_net.features(vol_sagit)
        vol_coron = self.coron_net.features(vol_coron)
        vol_axial = self.gap_axial(vol_axial).view(vol_axial.size(0), -1)
        x = torch.max(vol_axial, 0, keepdim=True)[0]
        vol_sagit = self.gap_sagit(vol_sagit).view(vol_sagit.size(0), -1)
        y = torch.max(vol_sagit, 0, keepdim=True)[0]
        vol_coron = self.gap_coron(vol_coron).view(vol_coron.size(0), -1)
        z = torch.max(vol_coron, 0, keepdim=True)[0]
        w = torch.cat((x, y, z), 1)
        out = self.classifier(w)
        return out
```

❖ Now we handled buttons after that we will handle database tab.

```
def handle_buttons(self):
    self.btn_axial.clicked.connect(lambda x: self.get_images(0))
    self.btn_coronal.clicked.connect(lambda x: self.get_images(1))
    self.btn_sagittal.clicked.connect(lambda x: self.get_images(2))
    self.btn_examine.clicked.connect(self.examine)
    self.tabWidget.currentChanged.connect(lambda x: self.update_table(
        self.db.get_all_data(50)))
    self.tabWidget.setTabEnabled(2, False)

    self.tableWidget.cellDoubleClicked.connect(self.cell_clicked)
    self.menu_excel_file.triggered.connect(self.extract_xlsx)
    self.tabWidget.setCurrentIndex(0)
```

- At database tab user can filter data in the table by (name, age, created time).

```
def filter_(self):
    engine = self.txt_search.toPlainText()
    if self.radio_id.isChecked() == True:
        if engine == '':
            data = self.db.filter_by_name('')
        elif engine.isdigit(): data = self.db.filter_by_id(engine)
        else: data = self.db.filter_by_id('-1')
    elif self.radio_name.isChecked() == True:
        data = self.db.filter_by_name(engine)
    else:
        data = self.db.filter_by_Date(engine)

    self.update_table(data)
    return
```

- We put signal on tab clicked to execute slot update_table.

```
def update_table(self, data):
    if self.tabWidget.currentIndex() != 1: return
    self.tableWidget.setRowCount(len(data))
    for i in range(0, len(data)):
        self.tableWidget.setItem(i,0, QTableWidgetItem(str(data[i][0])))#id
        self.tableWidget.setItem(i,1, QTableWidgetItem(str(data[i][1])))#name
        self.tableWidget.setItem(i,2, QTableWidgetItem(str(data[i][2])))#age
        self.tableWidget.setItem(i,3, QTableWidgetItem(self.get_text_blood(data[i][3])))#blood
        self.tableWidget.setItem(i,4, QTableWidgetItem(self.get_text_result(data[i][7])))#abnormal
        self.tableWidget.setItem(i,5, QTableWidgetItem(self.get_text_result(data[i][8])))#acl
        self.tableWidget.setItem(i,6, QTableWidgetItem(self.get_text_result(data[i][9])))#men
        self.tableWidget.setItem(i,7, QTableWidgetItem(str(data[i][10])))#note
        self.tableWidget.setItem(i,8, QTableWidgetItem(str(data[i][11])))#created at
        for j in range(0,9):
            self.tableWidget.item(i,j).setTextAlignment(Qt.AlignHCenter)
    self.lbl_state.setText('table successfully updated!')
    return
```

- Update_table fill the table with data saved in database.

- We put signal on cell double click to execute slot cell_clicked.

```
def cell_clicked(self, row, column):
    id = self.tableWidget.item(row,0).text()
    self.start_edit_dialog(id,
                           self.db.get_name(id),
                           self.db.get_age(id),
                           self.db.get_blood(id),
                           self.db.get_note(id))
```

- Start_edit_dialog start a GUI that give user access to update row data or delete the row or extract data about the patient in clicked row that data can include personal data as (name , age , blood type , note , created time , abnormal result , ACL result , men result), images and video created from images with number of frames the user choose from combo box include (15 , 30 , 60) frames but the default value is 15 frames per second.

- The function that extract patient images:

```
def extract_images(self, id, location):
    if location == '':
        location = os.path.join(os.environ["HOMEPATH"], r"Desktop\{}".format(id))
    axial = np.load(r'database\axial_{}.npy'.format(id))
    coronal = np.load(r'database\coronal_{}.npy'.format(id))
    sagittal = np.load(r'database\sagittal_{}.npy'.format(id))
    os.mkdir(os.path.join(location, 'axial images'))
    os.mkdir(os.path.join(location, 'coronal images'))
    os.mkdir(os.path.join(location, 'sagittal images'))
    for j,i in enumerate(axial):
        cv.imwrite(os.path.join(location, 'axial images') + r'\{}.jpg'.format(j), i)
    for j,i in enumerate(coronal):
        cv.imwrite(os.path.join(location, 'coronal images') + r'\{}.jpg'.format(j), i)
    for j,i in enumerate(sagittal):
        cv.imwrite(os.path.join(location, 'sagittal images') + r'\{}.jpg'.format(j), i)
```

- The function that extract patient videos:

```
@staticmethod
def export_video(id, export_type, num_frames, file_name, save_location = 'BASE'):
    if export_type not in ['axial', 'coronal', 'sagittal']:
        export_type = 'axial'
    if num_frames > 60 or num_frames <0:
        num_frames = 30
    try:
        images_npy = np.load(r'database\{}_{}.npy'.format(export_type,id))
    except Exception:
        raise FileNotFoundError('can\'t find data/{}.npy!'.format(export_type))
    images = []
    for i in images_npy:
        cv.imwrite(r'data/temp.jpg', i)
        images.append(cv.imread(r'data/temp.jpg'))
    os.remove(r'data/temp.jpg')
    if save_location == 'BASE':
        video = cv.VideoWriter('{}/{}.mp4'.format(file_name), cv.VideoWriter_fourcc(*'MP4V'), num_frames, (256,256))
    else:
        video = cv.VideoWriter('{}/{}.mp4'.format(save_location, file_name), cv.VideoWriter_fourcc(*'MP4V'), num_frames, (256,256)
    for i in images:
        video.write(i)
    video.release()
    cv.destroyAllWindows()
```

- The menu bar has one action that extract data in database to excel sheet.

```
def extract_xlsx(self):
    try:
        loc = QFileDialog.getSaveFileName(
            self, 'Choose File', directory = os.path.dirname(__file__))
    except FileNotFoundError:
        QMessageBox.warning(self, "Error!", "can\'t find path")
    loc = str(loc[0])[:]
    if loc == '': return
    os.mkdir(loc)

    worksheet = workbook.add_worksheet()
    rows = self.db.get_all_data()
    worksheet.write(0, 0, 'ID')
    worksheet.write(0, 1, 'Name')
    worksheet.write(0, 2, 'Age')
    worksheet.write(0, 3, 'Blood')
    worksheet.write(0, 4, 'Abnormal result')
    worksheet.write(0, 5, 'Acl result')
    worksheet.write(0, 6, 'Men result')
    worksheet.write(0, 7, 'Note')
    worksheet.write(0, 8, 'Created at')

    for i,row in enumerate(rows):
        worksheet.write(i, 0, row[0])#id
        worksheet.write(i, 1, row[1])#name
        worksheet.write(i, 2, row[2])#age
        worksheet.write(i, 3, self.get_text_blood(row[3]))#blood
        worksheet.write(i, 4, self.get_text_result(row[7]))#abnormal
        worksheet.write(i, 5, self.get_text_result(row[8]))#acl
        worksheet.write(i, 6, self.get_text_result(row[9]))#men
        worksheet.write(i, 7, row[10])#note
        worksheet.write(i, 8, row[11])#time

    workbook.close()
    self.lbl_state.setText('all data successfully extracted!')
    return
```

- We use SQLite database.
- At constructor we create database if it not exists and connect if exist.

```
def __init__(self):
    try:
        self.connection = sq.connect(r'database\database.db', check_same_thread=False)
        self.cursor = self.connection.cursor()
    except Exception:
        x = open(r'database\database.db', 'w')
        x.close()
        self.connection = sq.connect(r'database\database.db', check_same_thread=False)
        self.cursor = self.connection.cursor()
        self.create_db()
```

- The structure of the table with ID primary key.

```
def create_db(self):
    target = '''
    CREATE TABLE "{0}" (
            "{1}"    INTEGER PRIMARY KEY,
            "{2}"    TEXT,
            "{3}"    INTEGER,
            "{4}"    INTEGER,
            "{5}"    TEXT NOT NULL UNIQUE,
            "{6}"    TEXT NOT NULL UNIQUE,
            "{7}"    TEXT NOT NULL UNIQUE,
            "{8}"    NUMERIC NOT NULL,
            "{9}"    NUMERIC NOT NULL,
            "{10}"   NUMERIC NOT NULL,
        "{11}"   TEXT,
        "{12}"   TEXT,
    );
    '''.format(self.TABLE, self.C_ID, self.C_NAME, self.C_AGE,
    self.C_BLOOD, self.C_AXIAL, self.C_CORONAL, self.C_SAGITTAL,
    self.C_ABNORMAL, self.C_ACL, self.C_MEN, self.C_NOTE, self.C_TIME)
    self.cursor.execute(target)
    self.connection.commit()
    return
```

- We can insert, delete, update, get data, and filter by (id, name, time).

- On the next Chapter we will talk about "*System Analysis and Design*".

Chapter Four

4 System Analysis and Design

4.1 System Request:

1. Project sponsor
2. Business need
3. Business requirement
4. Business value
5. Special issues and constraints

4.1.1 Project Sponsors:

i- Doctors (both radiologist and Orthopedic surgeon)
ii- Medical students specialized in radiology, Orthopedic surgery
iii- Medical rays' centers
iv- Hospitals

4.1.2 Business Need:

i- Increase accuracy in diagnosis
ii- Increase accuracy in take right way of treatment
iii- Visualize the scans in video frames

4.1.3 Business Requirement:

i- Give breaking opinion in A case with a difference of opinions
ii- Easy to diagnose the patient’s scan
iii- Easy in explore the patient’s scan

4.1.4 business value:

i- Increase diagnosing accuracy
ii- Decrease the percentage of take wrong decision in treatment way

4.1.5 special issues:

i- Wrong use of the software
ii- Increase in time of get result due to technical matters
iii- Overload on the software

4.1.6 feasibility analysis:

i- technical:
   a- number of slices in each plane may be very large
   b- familiar with desktop applications: very good

ii- economical:
   a- increase revenue
   b- low costs

4.2 Requirements:

1. **User requirements:**
   a. Navigation and browse through the place where the data is stored.
   b. Easily use the application**.**

2. **Business requirements:**
   a. Increase the usage of AI specifically deep learning in the medical & radiology domain.
   b. Decrease the percentage of the wrong diagnosis and wrong treatment decision.
   c. Decrease time of diagnosis the scan of the patient (case).

3. **System requirements**

i. Functional requirements:
   a. Get the diagnose from the planes of the scan: when the user enters the required data (name, age, blood type, axial slices, sagittal slices, coronal slices), he will be able to that if there is injuries (ACL, meniscus, abnormality) or not or one of them.
   b. Save the results of the diagnosis: the user can save the results for all the patients in excel sheet for future use.
   c. Save the planes slices in video: the user can save the planes slices in video with extension .mp4 with wanted speed (slice (frame) per second).
   d. The data of each patient are saved separately.

ii. Nonfunctional requirements:
   a. Security: the data of each patient (case) shall be secured and paid for.
   b. Operational: the user must use windows operating system.
   c. Cultural & political: English is the only language is used.

Use-case

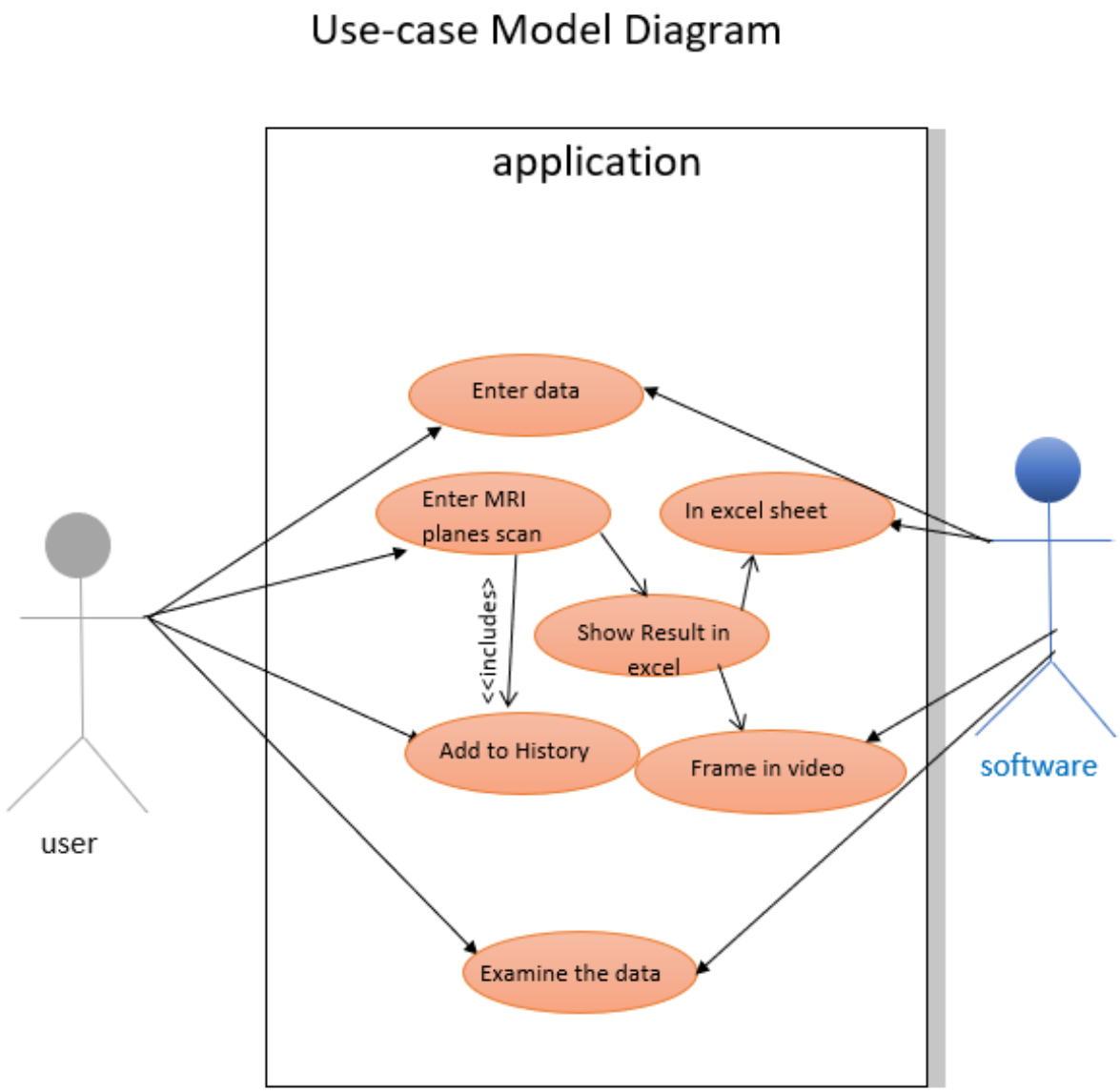


Figure 4-1: Use case for the project.

The system here is with on user that do every transaction in the application. First the User must enter his data like Name, age, blood type and Note. Second the user must enter MRI planes scan (Axial, sagittal and coronal images). Third the user clicks Examine button. Finally, the results of the scan will appear to the user.

Context Diagram (DFD-Level 0)

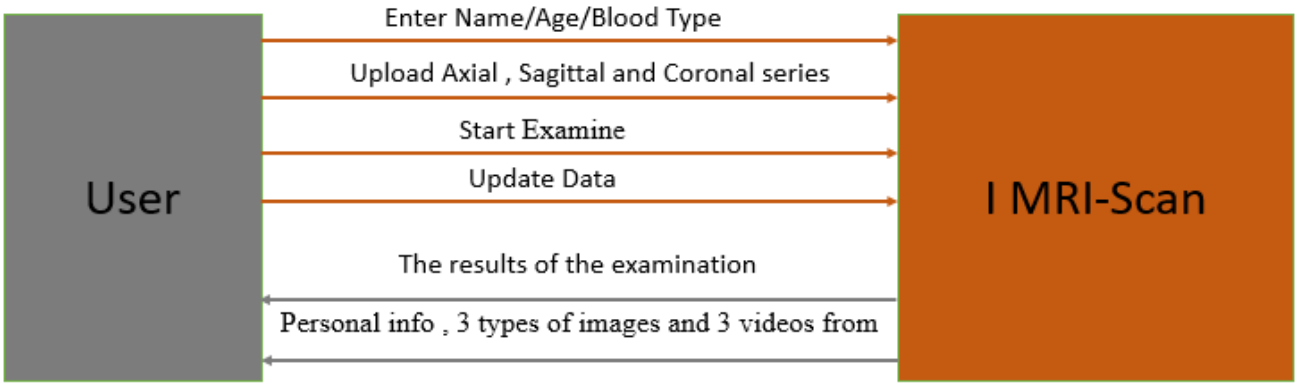


Figure 4-2: Context Diagram (DFD-level 0).

This figure shows the user transactions with the system and how to use the system. The context diagram is the (Level-0) of dataflow diagram (DFD). First the User must enter his data like Name, age, blood type and Note. Second the user must enter MRI planes scan (Axial, sagittal and coronal images). Third the user clicks Examine button. Fourth the user can update his data like (name, age and blood type) and he can't update his ID. Finally, the results of examination will appear to the user.

Data Flow Diagram (DFD)

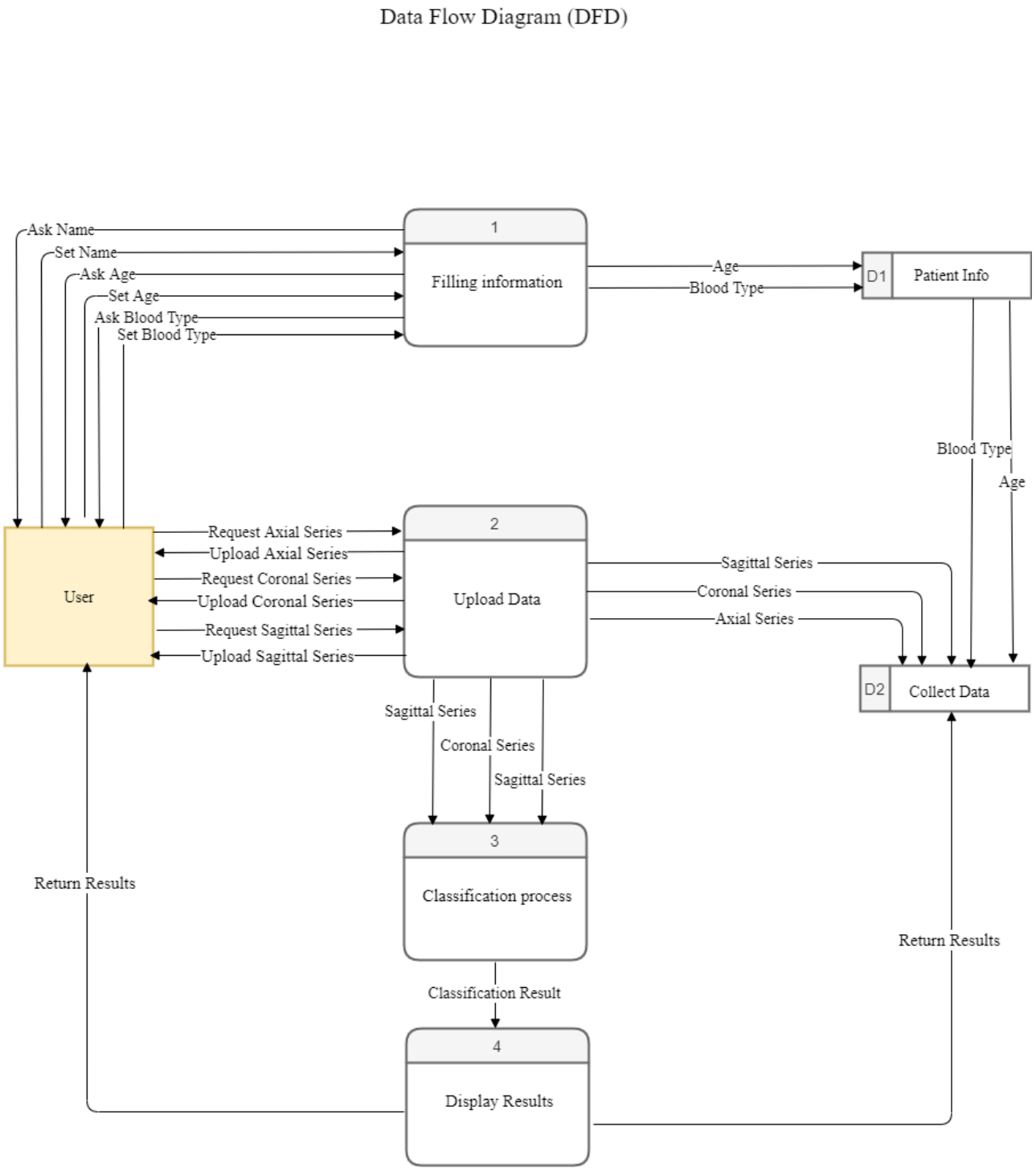


Figure 4-3: Data Flow Diagram (DFD).

- This figure shows the data flow diagram of the different processes of the user.
  - On the next Chapter we will discuss the section of "*Experiments and Results*".

Chapter Five

5 Experiments and Results

- Magnetic resonance imaging (MRI) of the knee is the favoured strategy for diagnosing knee wounds. In any case, the translation of knee MRI is time-concentrated and subject to analytic mistake and fluctuation. An automated framework for deciphering knee MRI could organize high-chance patients and help clinicians in making analyse. Deep learning techniques, in being able to automatically learn layers of features, are appropriate for demonstrating the unpredictable connections between clinical pictures and their translations. In this project, we developed a deep learning model for detecting general abnormalities and specific diagnoses (meniscal tears and anterior cruciate ligament [ACL] tears) on knee MRI exams. We at that point estimated the impact of giving the model's expectations to clinical specialists during understanding.

- The dataset consisted of 1,250 knee MRI exams. The majority vote of 3 musculoskeletal radiologists established reference standard labels on an internal validation set of 120 exams. We developed I MRI-Scan, a convolutional neural network for classifying MRI series and combined predictions from 3 series per exam using logistic regression. In detecting ACL tears, abnormalities, and meniscal tears, this model achieved area under the receiver operating characteristic curve (AUC).

- Our deep learning model can quickly create precise clinical pathology groupings of knee MRI tests from both inner and outer datasets. In addition, our outcomes bolster the statement that profound learning models can improve the exhibition of clinical specialists during clinical imaging translation. Further exploration is expected to approve the model tentatively and to decide its utility in the clinical setting.

5.1 Model:

- a convolutional neural network (CNN) mapping a 3-dimensional MRI series to a probability. The input to I MRI-Scan has dimensions s × 3 × 256 × 256, where s is the number of images in the MRI series (3 is the number of color channels). First, each 2-dimensional MRI image slice was passed through a feature extractor based on **AlexNet** to obtain a s × 256 × 7 × 7 tensor containing features for each slice.

- A global average pooling layer was then applied to reduce these features to s × 256. We then applied max pooling across slices to obtain a 256-dimensional vector, which was passed to a fully connected layer and sigmoid activation function to obtain a prediction in the 0 to 1 range. We optimized the model using binary cross-entropy loss. To account for imbalanced class sizes on all tasks, the loss for an example was scaled inversely proportionally to the prevalence of that example's class in the dataset.

- During training, the gradient of the loss was computed on each training example using the backpropagation algorithm, and I MRI-Scan parameters were adjusted in the direction opposite the gradient. Each training example was rotated randomly between –25 and 25 degrees, shifted randomly between –25 and 25 pixels, and flipped horizontally with 50% probability whenever it appeared in training. Model parameters were saved after every full pass through the training set, and the model with the lowest average loss on the tuning set was chosen for evaluation on the validation set.

- Training a CNN for image classification from scratch typically requires a dataset larger than 1,130 examples. For this reason, we initialized the weights of the **AlexNet** portion of the I MRI-Scan to values optimized on the ImageNet database of 1.2 million images across 1,000 classes, then fine-tuned these weights to fit our MRI dataset. This allowed the earlier layers of the network, which are more difficult to optimize than later layers, to immediately recognize generic features such as lines and edges. This "transfer learning" approach has similarly been applied to skin cancer and diabetic retinopathy image datasets.

- . Exams were sorted in reverse chronological order. Each exam was assigned 3 binary labels for the presence or absence of (1) any abnormality, (2) an ACL tear, and (3) a meniscal tear. Definitions for labels were as follows:

  - Abnormality: normal (all images reviewed are free of abnormalities) or abnormal (the abnormal findings in the internal validation set that were not ACL tear or meniscal tear included osteoarthritis, effusion, iliotibial band syndrome, posterior cruciate ligament tear, fracture, contusion, plica, and medial collateral ligament sprain).
  - ACL: intact (normal, mucoid degeneration, ganglion cyst, sprain) or tear (low-grade partial tear with <50% of fibers torn, high-grade partial tear with >50% of fibers torn, complete tear).
  - Meniscus: intact (normal, degenerative changes without tear, postsurgical changes without tear) or tear (increased signal reaching the articular surface on at least 2 slices or morphologic deformity).

5.2 Results:

❖ Our deep learning model predicted 3 outcomes for knee MRI exams (meniscal tears, anterior cruciate ligament [ACL] tears, and general abnormalities) in a matter of seconds and with similar performance to that of general radiologists.

- We experimented with providing model outputs to general radiologists and orthopedic surgeons during interpretation and observed statistically significant improvement in diagnosis of ACL tears with model assistance.
- When externally validated on a dataset from a different institution, the model picked up ACL tears with high discriminative ability.
- In detecting abnormalities, there were no significant differences in the performance metrics of the model and general radiologists.
- Our results demonstrate that a deep learning approach can achieve high performance in clinical classification tasks on knee MRI.
- The model achieved high specificity in detecting ACL tears on the internal validation set, which suggests that such a model, if used in the clinical workflow, may have the potential to effectively rule out ACL tears.
- The MRI model achieved state-of-the-art results on the external dataset, but only after retraining. It remains to be seen if the model would better generalize to an external dataset with more MRI series and a more similar MRI protocol.
- Deep learning has the potential to provide rapid preliminary results following MRI exams and improve access to quality MRI diagnoses in the absence of specialist radiologists.
- Providing clinical experts with predictions from a deep learning model could improve the quality and consistency of MRI interpretation.

5.2.1 Abnormal Results:

- Number of epochs = 20
- Time = 2:04:11.717334

❖ Learning rate=1e-5

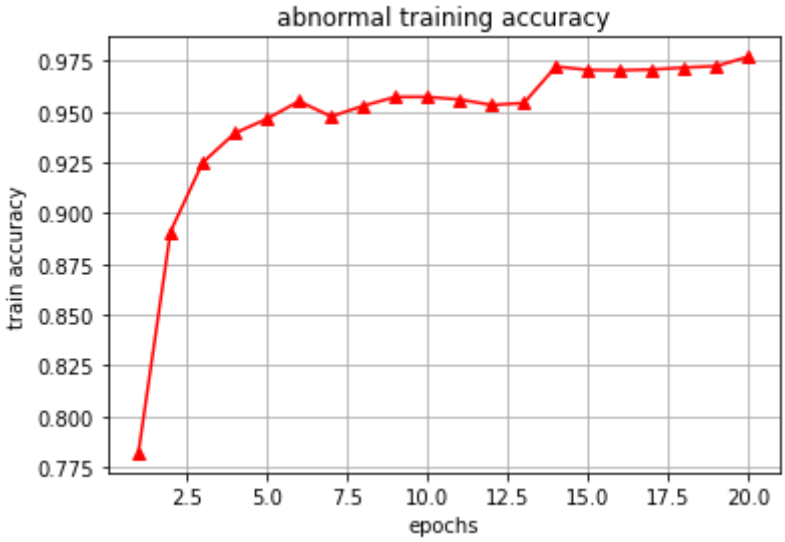


Figure 5-1: Abnormal Training Accuracy.

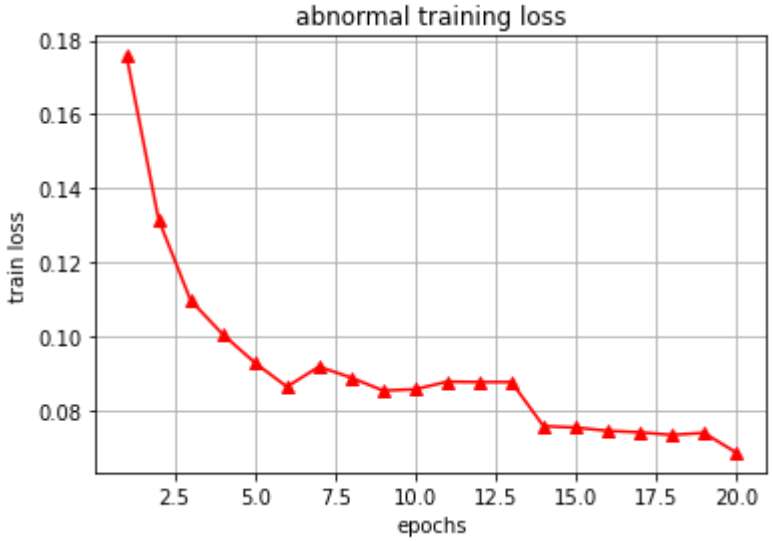


Figure 5-2: Abnormal Training Loss.

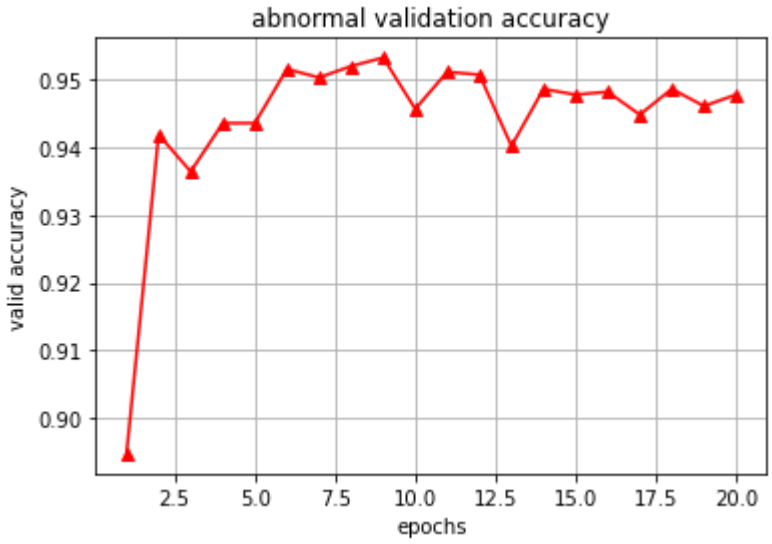


Figure 5-3: Abnormal Validation Accuracy.

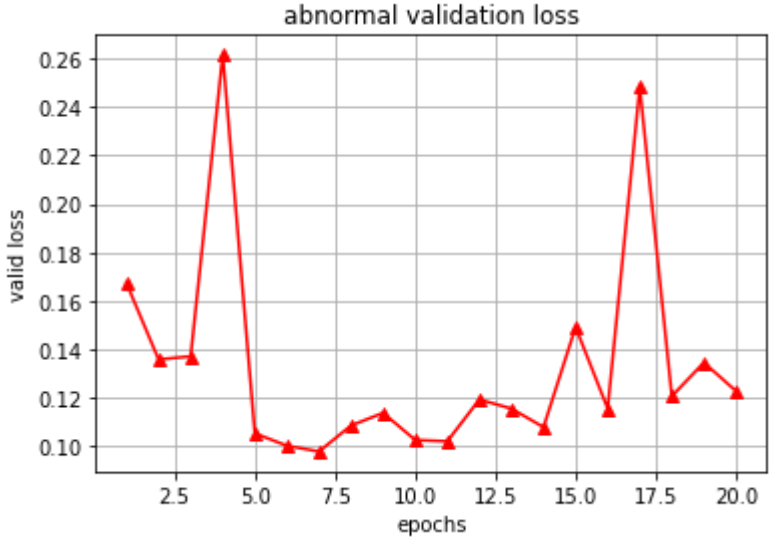


Figure 5-4: Abnormal Validation Loss.

5.2.2 ACL Results:

- Number of epochs = 20
- Time = 3:29:12.627334
- Learning rate=1e-06

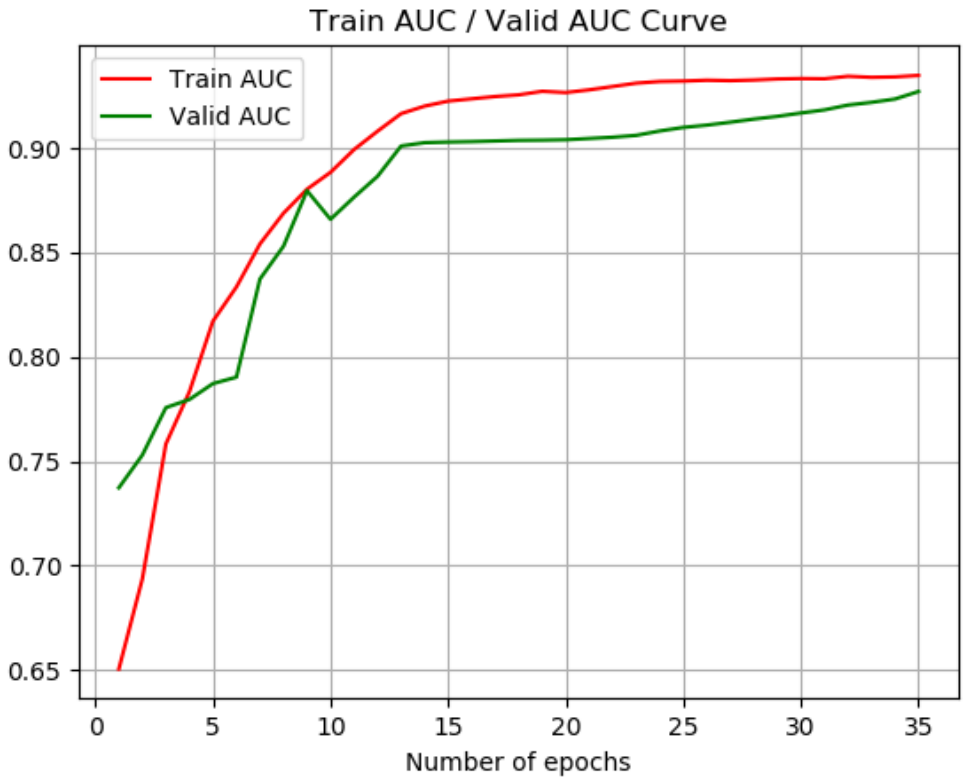


Figure 5-5: ACL train/valid ACC.

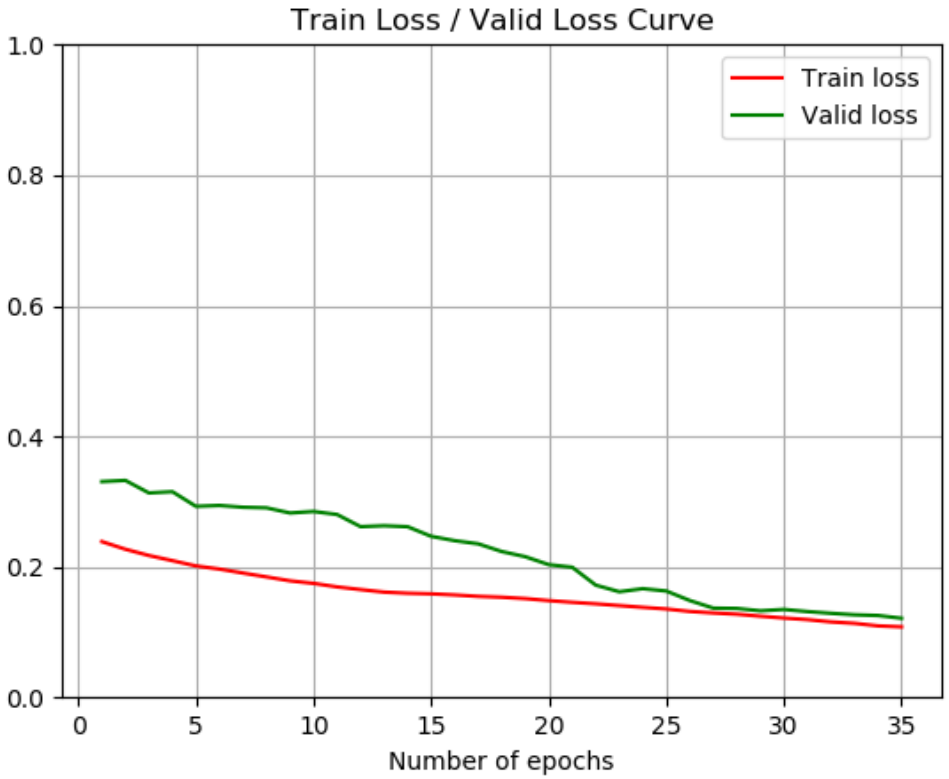


Figure 5-6: ACL train/valid loss.

5.2.3 Meniscus Results:

- Number of epochs = 40
- Learning rate = 1e-6
- Time=3:11:35.956819

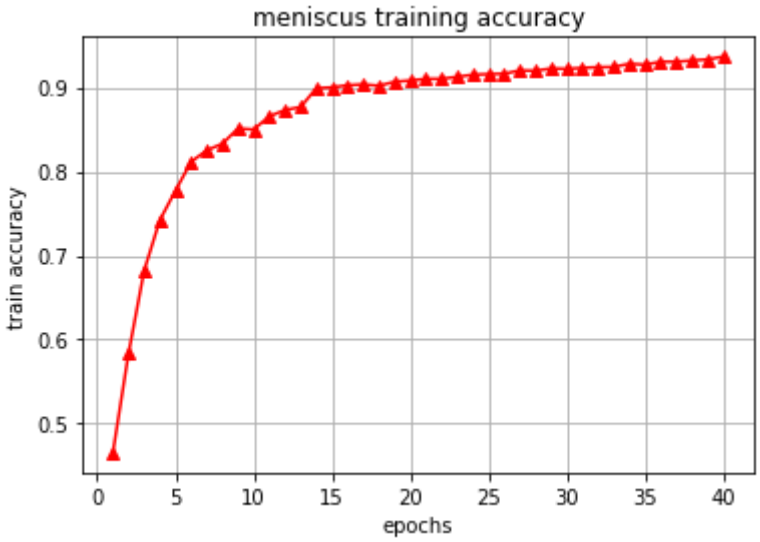


Figure 5-7: Meniscus Training Accuracy.

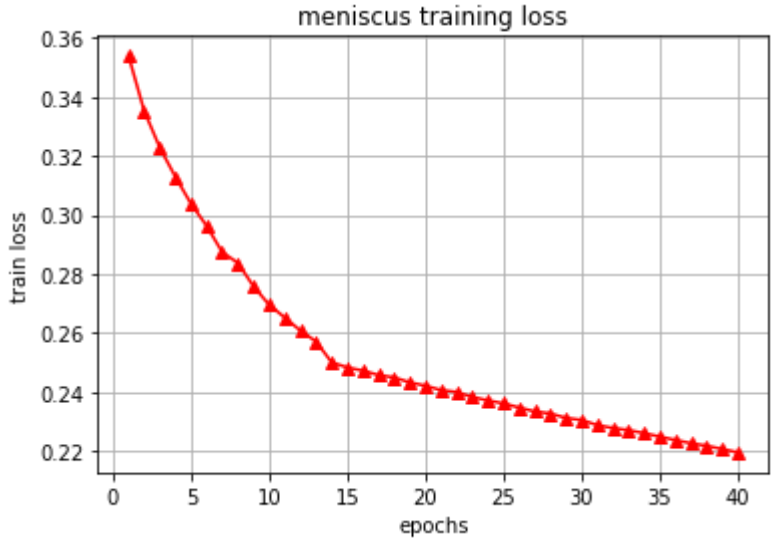


Figure 5-8: Meniscus Training Loss.

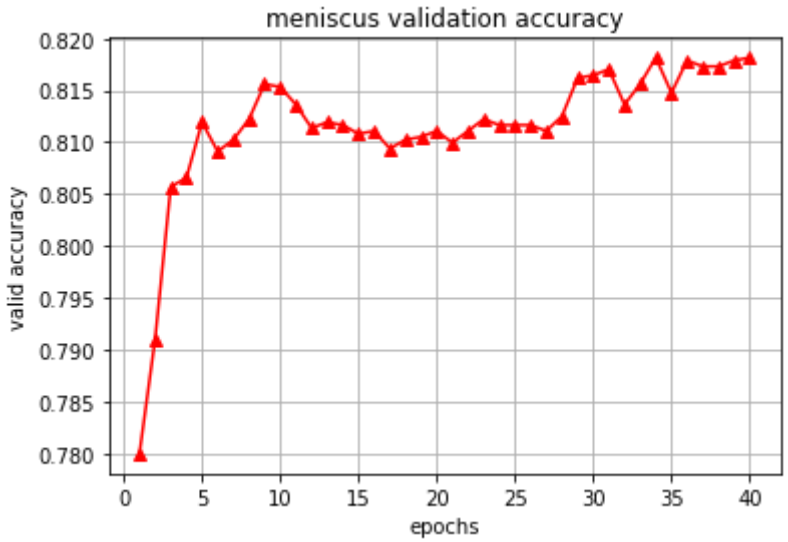


Figure 5-9: Meniscus Validation Accuracy.

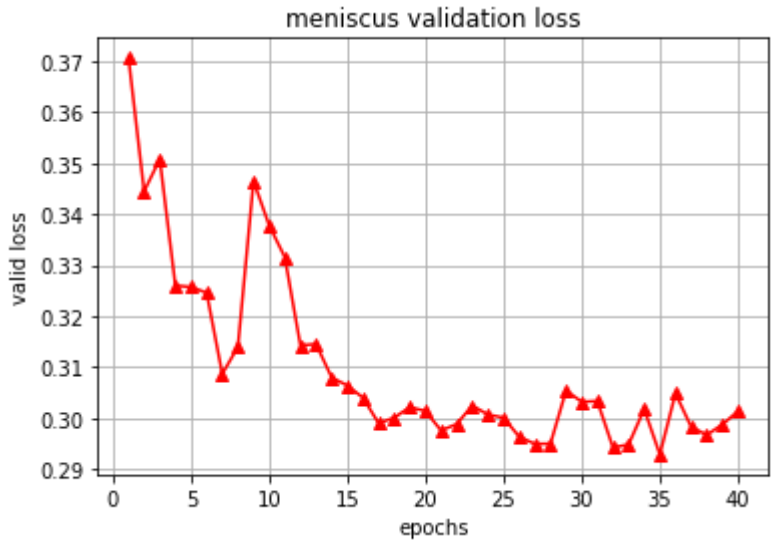


Figure 5-10: Meniscus Validation Loss.

- On the next Chapter we will discuss the section of "*Conclusion and Discussion*".

Chapter Six

6 Conclusion and Discussion

In this paper, we suggest the I MRI-Scan system that combines standard methods. Like the convolutional neural network. Experiments have shown that the combination of normalization procedures greatly improves the accuracy of the method. As shown in the results, our method achieves competitive results and provides a simpler solution. This project for "***Faculty of Computer and Information***" Zagazig University (FCI-ZU).

- We had run our dataset on two different approaches that is considered as an improvement architecture of CNN.it is compared in our research for choosing the best on in training the MRI knee dataset in 3 different injuries each injury is binary classified. that approaches are:
    1. Esambling learning with transfer learning: each plane with one type of injury is trained separately so we have 3 planes (sagittal, coronal, axial) and 3 different injuries(abnormality, meniscal tear, ACL tear) so we end with 9 trained models each input is(1*S*3*256*256) where S is number of slices or frames , 3 is the color channel, 1 is the number of batches. After train the 9 models combine the three models that intersect in the same injury type in logistic regression binary classifier or support vector machine (SVM) binary classifier. In the transfer learning we use Alex-net as the pre-train model.
    2. Multi-Input Convolutional Neural Network with the transfer learning: instead of train 9 models, just train only 3 models where each one is specialized in one injury. each model consists of 3 input layers for the 3 planes for just one type of injury and use pre-trained model Alex-net.
- we classify our data into 3 type of injuries (abnormality, meniscal tear, ACL tear) each type is binary classified. we choose approach Multi-Input Convolutional Neural Network with the transfer learning where it was the best in both training time and results

- We have written deep learning code in python using PyTorch framework for the build the model, train the model and save the training parameters of the model.

- We have built our desktop application with PYQT5 where the application can be used easily with different types of users.

- We use OpenCV (open computer vision) library for some features.

- The GUI consists of name, the blood type and age of patient, then adding the slices for each sagittal, coronal, axial planes then the results will appear for each injury and can transfer the slices to video with extension .mp4 with wanted number of slices per second.

Models/Loss & Acc	Abnormal	ACL	Meniscus
Training Accuracy	0.9772	0.9398	0.9365
Training Loss	0.0686	0.1085	0.2194
Validation Accuracy	0.9478	0.9271	0.8182
Validation Loss	0.1229	0.1217	0.3013

- On the next Chapter we will discuss the section of "*Appendices*". In which we will display our Application design and result.

Chapter Seven

7 Appendices

In this section we will show the application and project run. When you open the program, the main interface appears as shown**:**

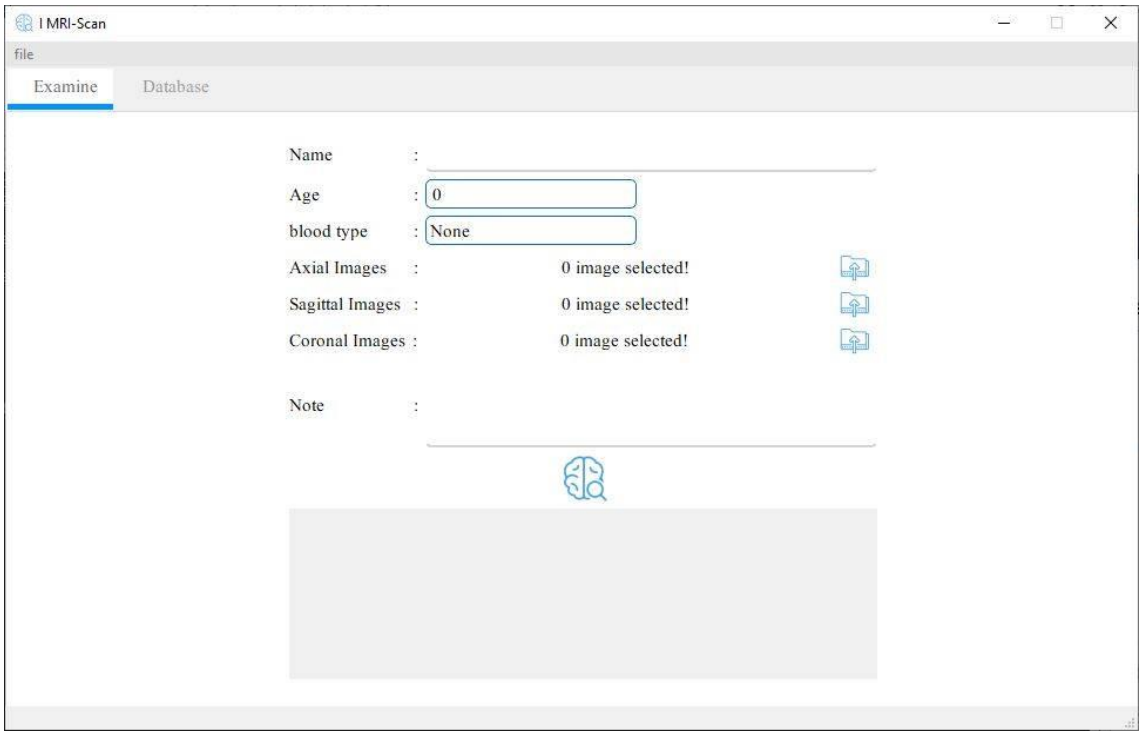


Figure 7-1: I MRI-Scan (Examine).

❖ at first menu bar, it's contained a single action that save all patients data to an excel sheet.

❖ there are two tabs (Examine, Database)

At first tab user enter data about the patient that includes:

1. Name:
   It had constrained to not contain (', ", &, ^, $, #, %).
2. Age:
   It had constrained to be entered as a number from 0-99 with 0 default value as unknown age.

3. Blood type:

   It had constrained to be entered as (None, A+, A-, B+, B-, AB+, AB-, O+, O-) with None default value as unknown type**.**

4. Axial Images:

   Browser to get Axial images.

5. Sagittal Images:

   Browser to get Sagittal images.

6. Coronal Images:

   Browser to get Coronal images.

7. Note:

   It had constrained to not contain (‘, “, &, ^, $, #, %).

8. Examine Button:

   Send all data in fields to the model after doing processing which include remove noise from image and focus on target part in image, resize image to be (256 x 256), send images to the models then it send result data to appear in the report box below it and save data in the database.

9. State bar:

   It is a label that tells user about actions that be in behind.

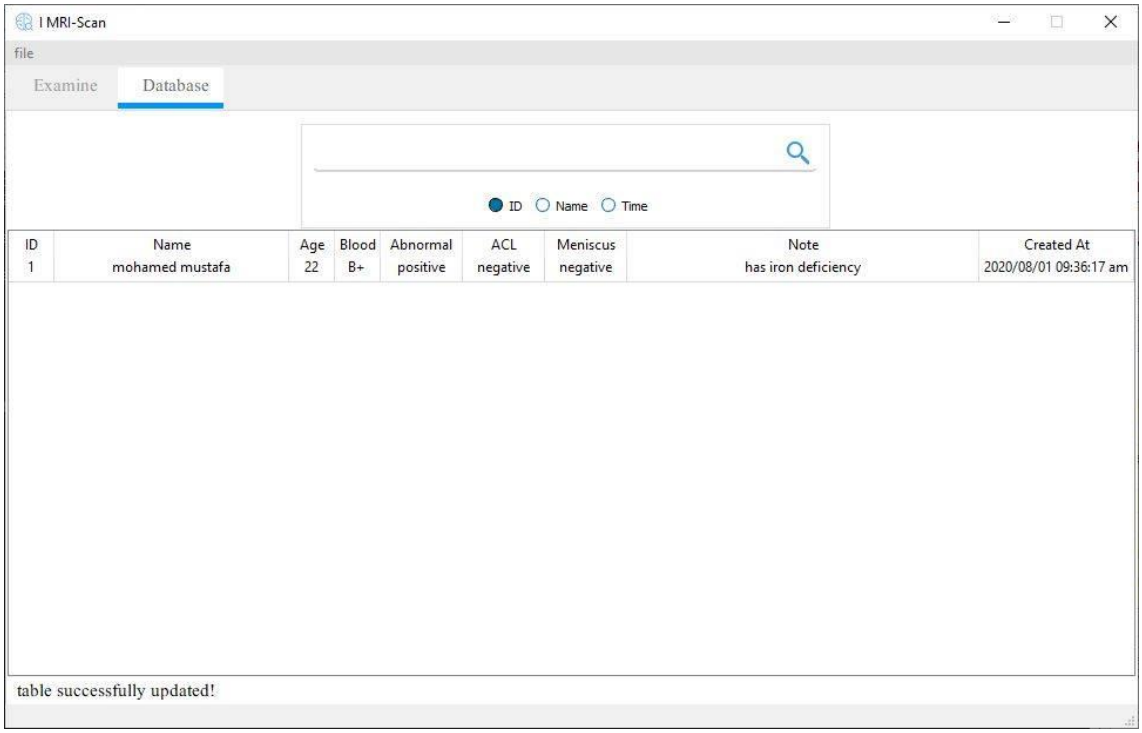


Figure 7-2: I MRI-Scan (Database).

At the second tab user can check saved data in the database:

- ❖ User can filter table by id, name and time.
- ❖ User can edit limited data or delete or extract file about patient by double click on the target row.

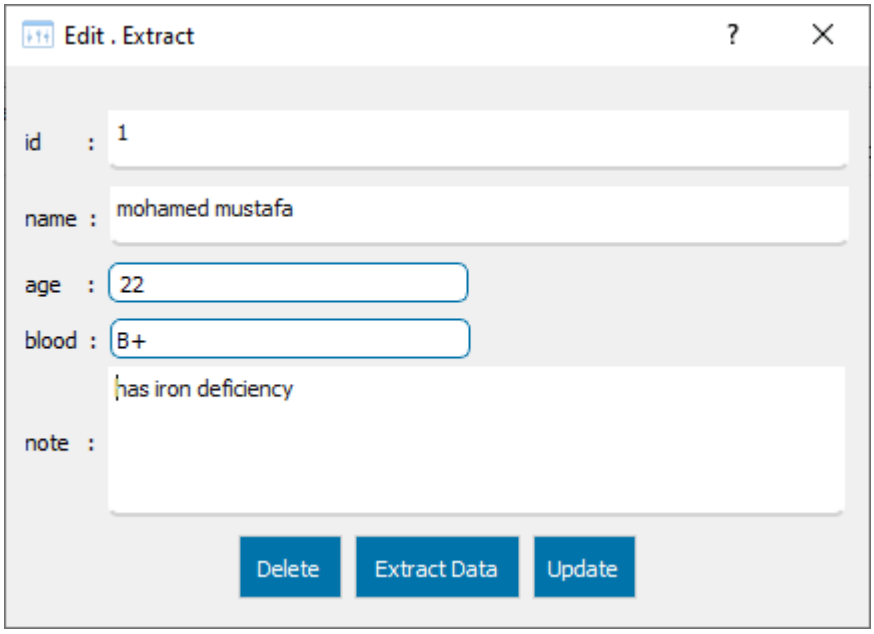


Figure 7-3: Edit Database.

- ❖ User cannot edit id field but can edit the other fields (name, age, blood, note) by clicking update button.
- ❖ User can delete the row by clicking delete button.
- ❖ User can extract the patient data by clicking extract data button.

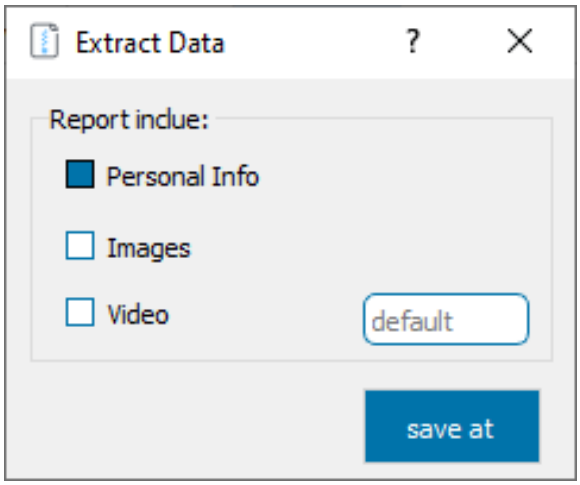


Figure 7-4: Extract Data.

1. Personal info check button:

   Saves (name, age, blood type, abnormal result, ACL result, meniscal result, note, created time) in text file.

2. Images check button:

   Saves 3 types of images (axial, sagittal, coronal).

3. Video check button:

   Saves 3 videos from (axial, sagittal, coronal) images also user can choose number of frames from spin box which include (default, 15, 30, 60) default value is 30 frame per second, also spin box is enabled after checking video check box.

4. Save at button:

   Browse the location where user will save the extracted data.

References

1. **Cook, J. L., Cook, C. R., Stannard, J. P., Vaughn, G., Wilson, N., Roller, B. L., ... & Kuroki, K. (2014). MRI versus ultrasonography to assess meniscal abnormalities in acute knees. *The journal of knee surgery*, *27*(04), 319-324.**

2. **Suter, L. G., Smith, S. R., Katz, J. N., Englund, M., Hunter, D. J., Frobell, R., & Losina, E. (2017). Projecting lifetime risk of symptomatic knee osteoarthritis and total knee replacement in individuals sustaining a complete anterior cruciate ligament tear in early adulthood. *Arthritis care & research*, *69*(2), 201-208.**

3. **Sihvonen, R., Englund, M., Turkiewicz, A., & Järvinen, T. L. (2016). Mechanical symptoms and arthroscopic partial meniscectomy in patients with degenerative meniscus tear: a secondary analysis of a randomized trial. *Annals of Internal Medicine*, *164*(7), 449-455.**

4. **Nyu´ l LG, Udupa JK. On standardizing the MR image intensity scale. Magn Reson Med. 1999; 42:1072–81. PMID: 10571928.**

5. **ImageNet Classification with Deep Convolutional Neural Networks by: Alex Krizhevsky,Ilya Sutskever,Geoffrey E. Hinton.**

6. **One weird trick for parallelizing convolutional neural networks:Alex Krizhevsky.**

7. **Tissue-Based MRI Intensity Standardization:Application toMulticentric Datasets by:Nicolas Robitaille,1 AbderazzakMouiha,1 Burt Cr´epeault,1 Fernando Valdivia,1 Simon Duchesne,1, 2 and The Alzheimer's Disease Neuroimaging Initiative3.**

8. **Multi-Input Convolutional Neural Network for Flower Grading by:Yu Sun, Lin Zhu, Guan Wang, and Fang Zhao.**

9. **MR Imaging–based Diagnosis and Classification of Meniscal Tears by: Jie C. Nguyen, MD, MS Arthur A. De Smet, MD,Ben K. Graf, MD,Humberto G. Rosas, MD.**

10. **Semi-Automated Detection of Anterior Cruciate Ligament Injury from MRI by:Ivan S˘ tajduhara,d,————————————, Mihaela Mamulab, Damir Miletic´b, Go¨zde U¨ nalc.**

11. **Histogram-based normalization technique on human brain magnetic resonance images from different acquisitions Xiofei Sun1,2,3, Lin Shi4,5, Yishan Luo1,2, Wei Yang6,7, Hongpeng Li8*, Peipeng Liang9, Kuncheng Li9,Vincent C T Mok4, Winnie C W Chu1,2,6 and Defeng Wang1,2,3,6*.**

12. **Oliphant, T. E. (2006). *A guide to NumPy* (Vol. 1, p. 85). USA: Trelgol Publishing.**

13. **McKinney, W. (2011). pandas: a foundational Python library for data analysis and statistics. *Python for High Performance and Scientific Computing*, *14*(9).**

14. **Tosi, S. (2009). *Matplotlib for Python developers*. Packt Publishing Ltd.**

15. **Collobert, R., Bengio, S., & Mariéthoz, J. (2002). *Torch: a modular machine learning software library* (No. REP_WORK). Idiap.**

16. **Rush, A. M. (2020). Torch-Struct: Deep Structured Prediction Library. *arXiv preprint arXiv:2002.00876*.**

17. **Bisong, E. (2019). Google Colaboratory. In *Building Machine Learning and Deep Learning Models on Google Cloud Platform* (pp. 59-64). Apress, Berkeley, CA.**

18. **Kadiyala, A., & Kumar, A. (2017). Applications of python to evaluate environmental data science problems. *Environmental Progress & Sustainable Energy*, *36*(6), 1580-1586.**

19. **Van Rossum, G. (2009). Python 0.**

20. **Perkel, J. M. (2018). Why Jupyter is data scientists' computational notebook of choice. *Nature*, *563*(7732), 145-147.**

21. **Raybaut, P. (2009). Spyder-Documentation. *Available online at: pythonhosted. org*.**

22. **Hu, Q., Ma, L., & Zhao, J. (2018, December). Deepgraph: A pycharm tool for visualizing and understanding deep learning models. In *2018 25th Asia-Pacific Software Engineering Conference (APSEC)* (pp. 628-632). IEEE.**

23. **Munaiah, N., Kroh, S., Cabrey, C., & Nagappan, M. (2017). Curating github for engineered software projects. *Empirical Software Engineering*, *22*(6), 3219-3253.**