Low-Level Design Document

Knee MRI Classification

Revision Number: 2.0

Imran
Knee MRI Intern Team

Document version control

Date Issued	Version	Description	Author
09-06-2021	1.0	First copy of LLD	Imran
12-07-2021	2.0		

Contents

Abstract

Magnetic resonance imaging (MRI) of the knee is the preferred method for diagnosing knee injuries. However, interpretation of knee MRI is time-intensive and subject to diagnostic error and variability. An automated system for interpreting knee MRI could prioritize high-risk patients and assist clinicians in making diagnoses. Deep learning methods, in being able to automatically learn layers of features, are well suited for modelling the complex relationships between medical images and their interpretations.

1. Introduction

1.1. Why this Low-Level Design Document?

The technical design document gives a design blueprint of the Knee MRI Classification project. This document communicates the technical details of the solution proposed.

Once agreed as the basis for the building of the project, the flowchart and assumptions will be used as a platform from which the solution will be designed.

Note: All the code will be written in python version 3.7

1.2. Scope

This document captures the different workflows involved to build the solution, exceptions in the workflows and any assumptions that have been considered. Changes to this business process may constitute a request for change and will be subject to the agreed agility program change procedures.

.

2. Project Description

2.1. Introduction

The knee is one of the largest and most complex joints in the body. The knee joins the thigh bone (femur) to the shin bone (tibia). The smaller bone that runs alongside the tibia (fibula) and the kneecap (patella) are the other bones that make the knee joint. Tendons connect the knee bones to the leg muscles that move the knee joint. Ligaments join the knee bones and provide stability to the knee:

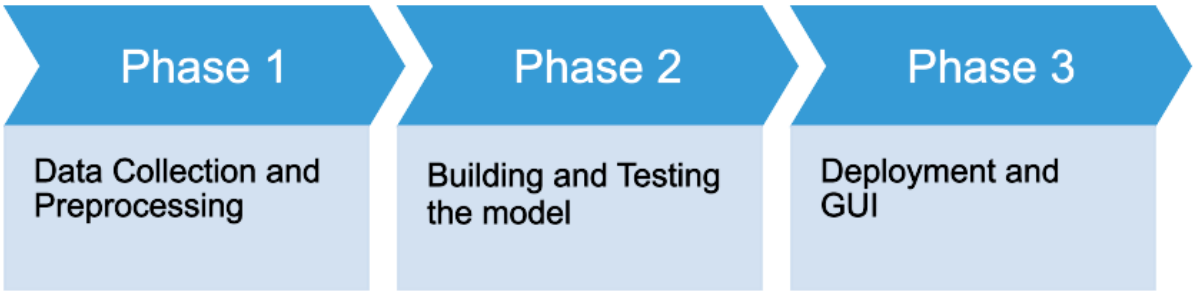
- Anterior Cruciate Ligament(ACL) prevents the femur from sliding backwards on the tibia.
- Posterior Cruciate Ligament(PCL) prevents the femur from sliding forward on the tibia
- Medial and Lateral Collateral Ligaments prevent the femur from sliding side to side.
- Two C-shaped pieces of cartilage called the medial and lateral menisci to act as shock absorbers between the femur and tibia.

Magnetic Resonance Imaging (MRI) of the knee uses a powerful magnetic field, radio waves and a computer to produce detailed pictures of the structures within the knee joint. It is typically used to help diagnose or evaluate pain, weakness, swelling or bleeding in and around the joint and it can help determine whether you require surgery. In order to better confront the ever-growing workload of musculoskeletal (MSK) radiologists, automated tools for patients' triage are becoming a real need, reducing delays in the reading of pathological cases.

2.2 High-Level Objectives

1. To enable reading/loading of MRI Image by the user.
2. To pre-process the MRI image for enhanced quality before further processing.
3. To classify the image into normal and abnormal using Deep Learning Techniques
4. If abnormal, predicting the type and location of the injury.
5. To perform statistical analysis of the results and display them in a user-friendly manner.

The project can be divided into multiple phases:



3. Workflow Overall

Conceptual architecture of the overall workflow is given in Figure 1

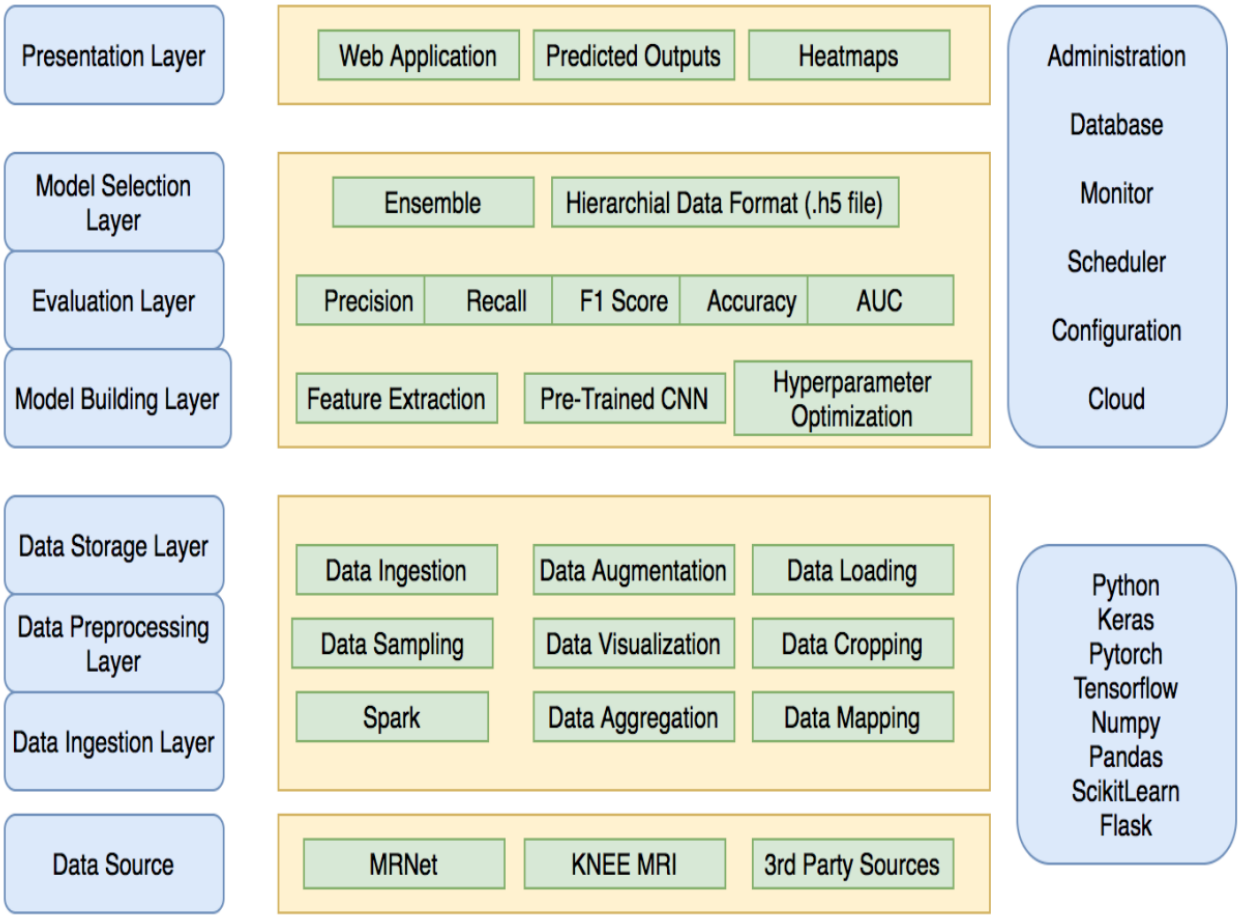


Figure 1: Conceptual Architecture

3.1 Data Collection and Preprocessing

3.1.1 Exploratory Data Analysis

The dataset employed in this research is MRNet V1 taken from Stanford ML Group (link) [1]. The same can be downloaded through Kaggle (link) [2].

It consists of a total of 1370 images divided into three classes namely - Abnormal exams (80.6%), ACL tears (23.3%), and Meniscal tears (37.1%).

This dataset is split into 3 sets as given in Table 1.

Table 1

Training set	1130 exams	1088 patients	Available
Validation set	120 exams	111 patients	Available
Test set	120 exams	113 patients	Not Available

The dataset consists of a train and validation set, including **csv** files for the conclusions of the three conditions.

The train and validation sets are divided into three planes - **axial, sagittal and coronal**

Each exam series is provided as a **.npy** file.

Some sample images are shown in Figure 2 to visualize the knee anatomy present in all the three planes.

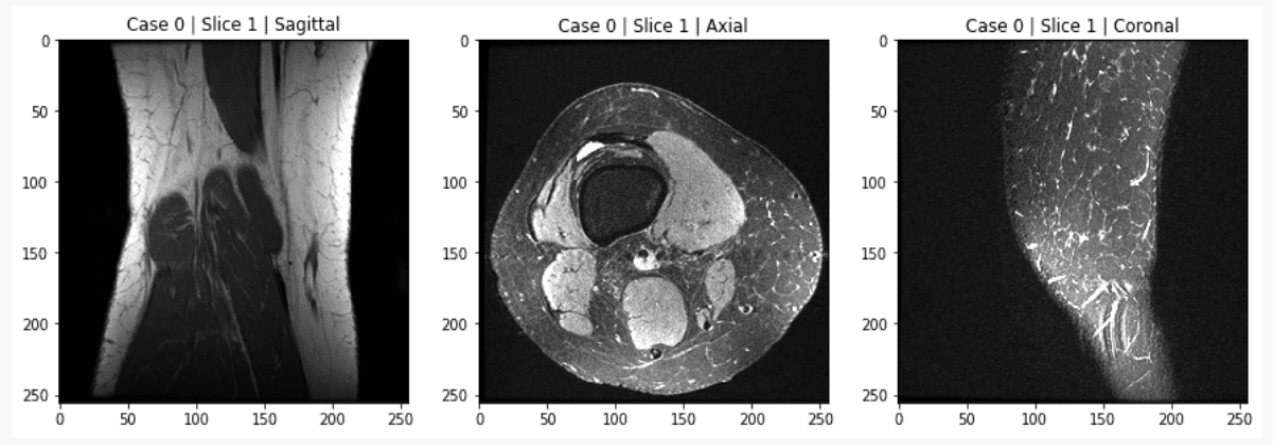


Figure 2 : Sagittal, meniscus and coronal Knee MRI image slice

The flowchart given in Figure 3 and 4 shows the hierarchical distribution of images.

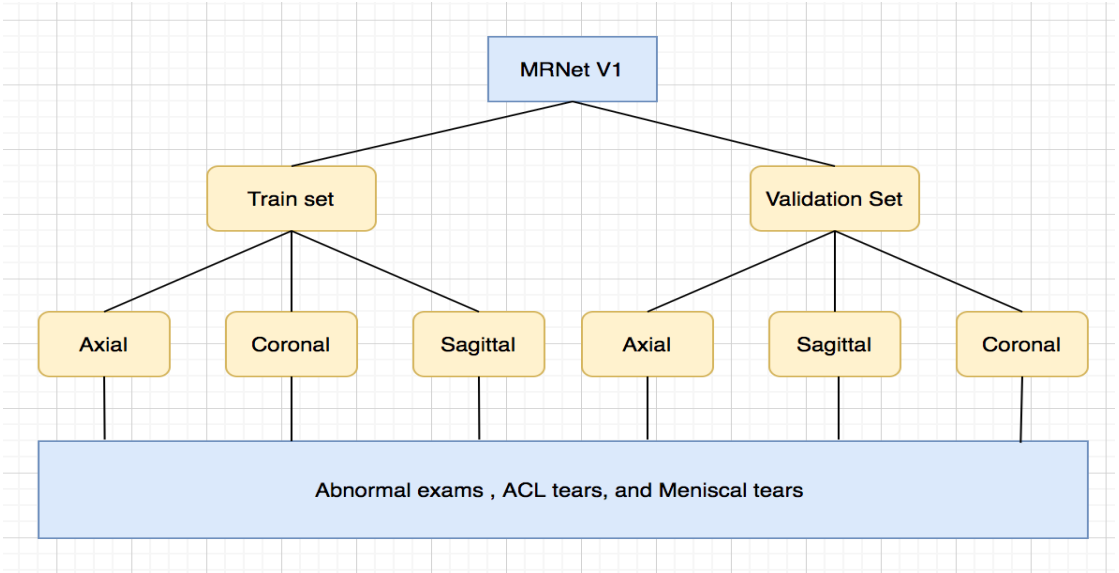
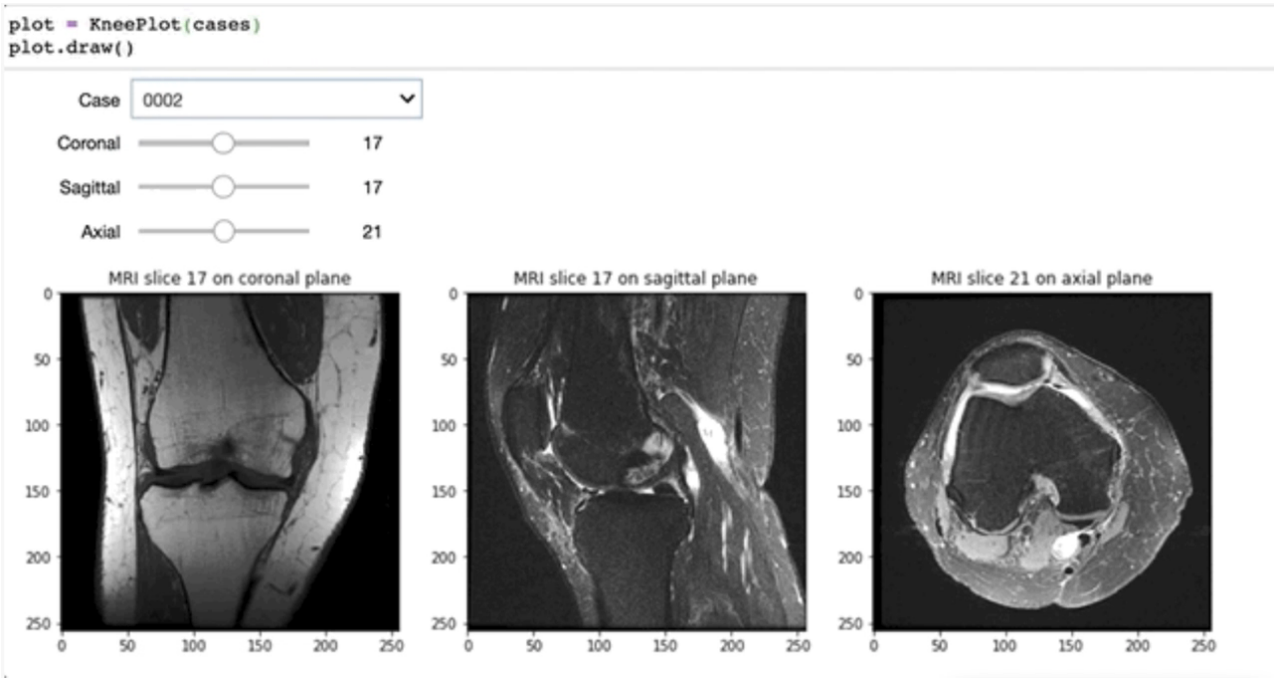


Figure 3: Tree Distribution of MRNet Dataset

```
.
├── train
│   ├── axial
│   ├── coronal
│   └── sagittal
├── train-abnormal.csv
├── train-acl.csv
├── train-meniscus.csv
├── valid
│   ├── axial
│   ├── coronal
│   └── sagittal
├── valid-abnormal.csv
├── valid-acl.csv
└── valid-meniscus.csv
```

Figure 4: Hierarchical Distribution of MRNet

Each MRI scan is a tensor of s slices. Each slice is a grayscale image of size (256, 256).



Click on the video above to play

The number of slices in each MRI scan fluctuates from 17 to 61 (mean 31.48, SD 7.07).

In most cases, the quantity of pictures falls in the middle of 20 and 40.

The minimum and maximum number of slices for each plane is given in Table 3.

Table 3

Set	Plane	Minimum	Maximum
	Axial	19	61
Training set	Coronal	17	58
	Sagittal	17	51
	Axial	20	52
Validation set	Coronal	17	48
	Sagittal	21	45

Table 4 and 5 shows the distribution of classes for training and validation exams.

There are inconsistencies between the train and validation dataset

For example, the distribution for training and validation set is 38.3% and 16.7% separately for Abnormal class with no ACL tear and no meniscal tear.

Table 4

Set	Abnormal	ACL	Meniscus	Exams	Percentage
Training	0	0	0	217	19.2
	1	0	0	433	38.3
	1	0	1	272	24.1
	1	1	0	83	7.3
	1	1	1	125	11.1
				1130	100%

Table 5

Set	Abnormal	ACL	Meniscus	Exams	Percentage
Validation	0	0	0	25	20.8
	1	0	0	20	16.7
	1	0	1	21	17.5
	1	1	0	23	19.2
	1	1	1	31	25.8
				120	100%

3.1.2 Dataset Distribution

The distribution of the dataset is given in Table 6. A standard train test split method is used.

120 exams from the train set are transferred using various sampling methods to ensure equal examples of abnormal, ACL tear and meniscal tear in the validation set (Table 6).

Table 6

Training set	1010 exams
Validation set	120 exams
Testing set	120 exams

3.1.3 Data Augmentation

Various Data Augmentation Techniques can be applied to enhance the accuracy. As recommended in [4], we need to be very careful using augmentations.

Image Data Generator [3] is used from the keras preprocessing package to generate batches of image data using methods like horizontal flip, vertical flip, rescale, zoom range, rotation range, width and height shift range.

It's worth considering that we have to perform data augmentation only on the training set i.e. 1010 exams (after train test split). This is useful to prevent data leakage problems.

3.1.4 Spectrum View

The spectrum view for sagittal, coronal and meniscus is presented in figure 5, 6, 7 respectively.

We can see more whiter regions at the center showing low frequency content.

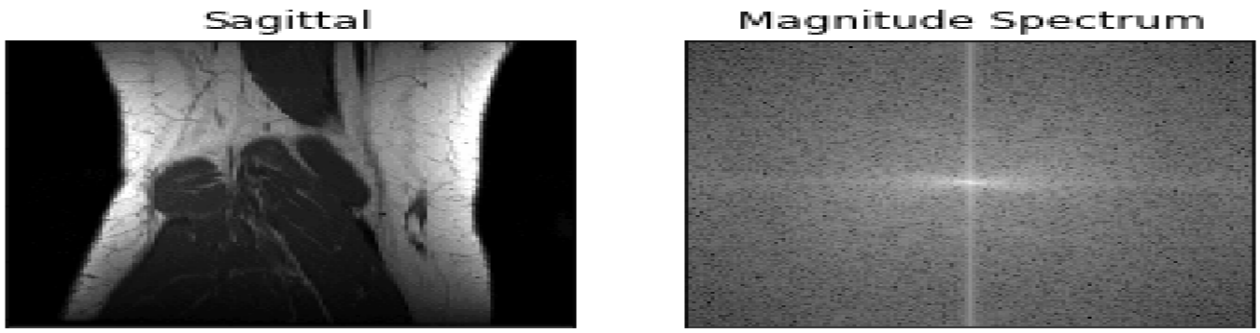


Figure 5: Magnitude Spectrum View Sagittal

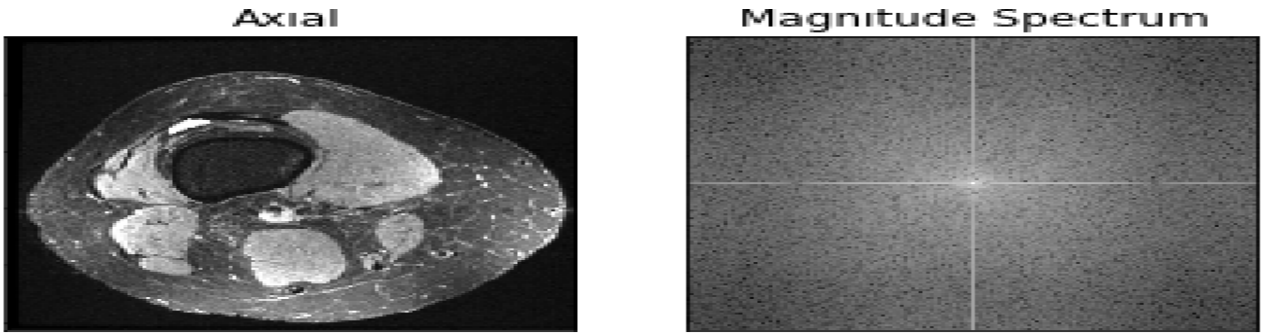


Figure 6:Magnitude Spectrum View Axial

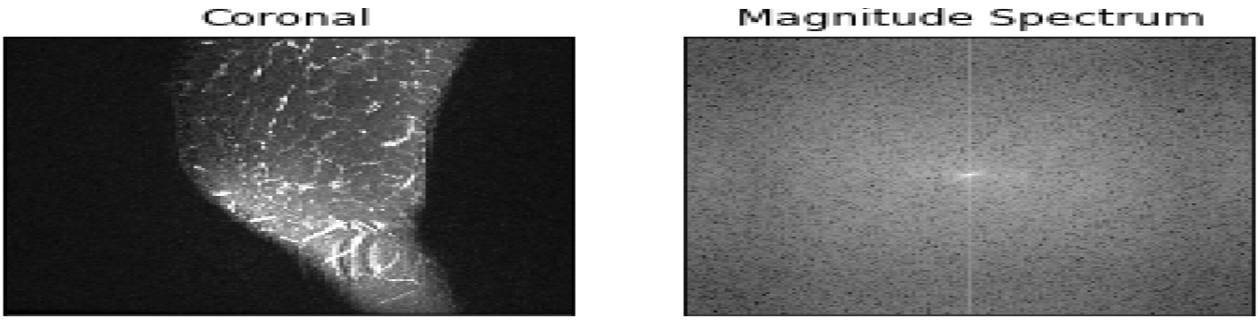


Figure 7: Magnitude Spectrum View Coronal

3.2 Model Building

3.2.1 Details of DL Model

The DL model is referred from a Research Article published in the Radiology Artificial Intell Journal. The article is titled as Fully Automated Diagnosis of Anterior Cruciate Ligament Tears on Knee MR Images by Using Deep Learning [2].

The proposed deep learning-based ACL tear detection system consisted of three separate CNN architectures.

1. The first part classifies the images that show Knee injuries from the entire MR image dataset. Here we are leveraging the concept of transfer learning utilizing Pre-Trained CNN models for instance, **VGG16, DenseNet, InceptionV3, ResNet50** etc

2. The second part isolates the anomalous region that contains the injury to narrow down the range of information. Object detection models like **YOLOv3, YOLOv4, YOLOv5, Mask R CNN** can be used.

3. The third part evaluates the isolated injuries on the image sections obtained from step two to determine the presence or absence of an injury. Again we are leveraging the concept of transfer learning here. After this, we can utilize the generated .h5 file to visualize the attention maps showing the affected region of the injury.

These three CNNs are connected in a cascaded fashion to create a fully automated processing pipeline, as shown in the figure 8.

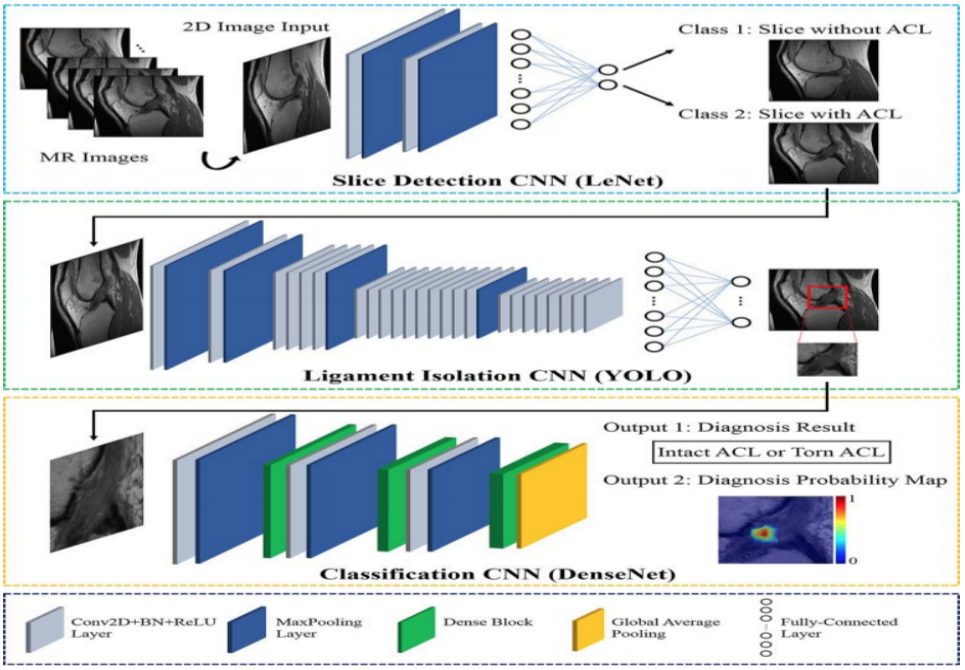


Figure 8: Model Training Architecture

3.2.2 Model Testing

Testing or Validation of Deep Learning models requires two sets of data validation and test data.

Following flowchart defines the flow of testing/validation of the proposed model.

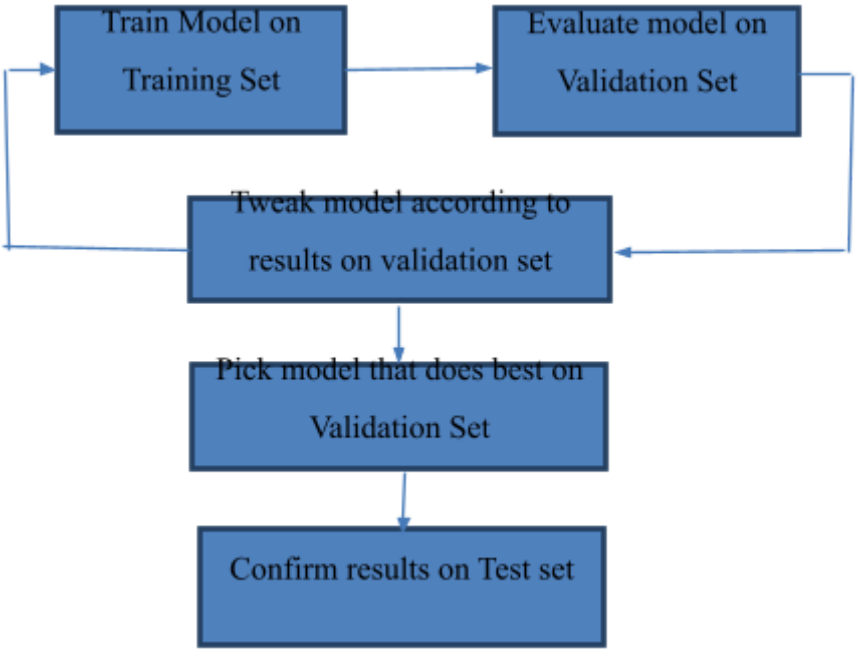


Figure 9: Model Testing Pipeline

3.2.3 Evaluation Metrics

Given Confusion Matrix in Figure 10, the following metrics can be used to measure the performance of the models.

	1	0
1	Tp	Fp
0	Fn	Tn

Figure 10: Confusion matrix

Precision = (Tp) / (Tp+Fp)

Recall = (Tp) / (Tp+Fn)

F1-score = (Tp+Tn) / (Tp+Tn+Fp+Fn)

Accuracy = 2*(Precision*Recall) / (Precision+Recall)

AUC is an area under a region operating curve which shows the relation between true positive rate and false positive rate across various thresholds.

3.2.4 Hyperparameter Tuning

Hyperparameter optimization is an important technique to improve the accuracy.

Learning rate, optimizer used, dropout rate, number of epochs are some of the parameters that need to be optimized.

Learning Rate controls how fast your neural net learns. The main goal is to minimize a loss function. If the learning rate is too high, loss will start jumping all over the place and never converge. Possible Values acc. to project : [0.0001, 0.001, 0.01, 0.1]

Adam is chosen as it is the standard most **optimizer** and works commendably well for any kind of deep learning model [41]. Possible Values acc. to project : [adam, sgd, adadelta]

Dropout is a regularization technique that randomly sets activations in a neural network to 0 with a probability rate of x. This helps prevent neural nets from overfitting (memorizing) the data as opposed to learning it. Possible Values acc. to project : [0.5, 0.3, 0.7, 0.2]

The model should be trained for an optimal number of **epochs** to mitigate overfitting. Early stopping is particularly required.

There are various HPO algorithms like **grid search, random search, genetic algorithm and optuna**.

In this project, Evolutionary algorithms are skipped due to low computation power available to train them. A simple brute force **grid search algorithm** is employed on a smal search space to optimize the parameters.

3.2.5 Attention Maps

Visualizing activation maps helps in featuring those areas of the image which are important, where activation is high.

It gives users certainty that their work is going the correct way.

Saliency Maps such as **SmoothGrad** and **vanilla saliency** [8] uses the gradients of the output layer with respect to input image which tells us how the output value changes with a change in input value

Class activation maps [7] such as **GradCAM**, **ScoreCAM**, and many more, didn't use gradients; instead they used the penultimate layer (pre-dense) to get spatial information which gets lost in dense layers.

tf-keras-vis [6] visualization toolkit is used for visualization of these maps. An example image is shown in figure 11.

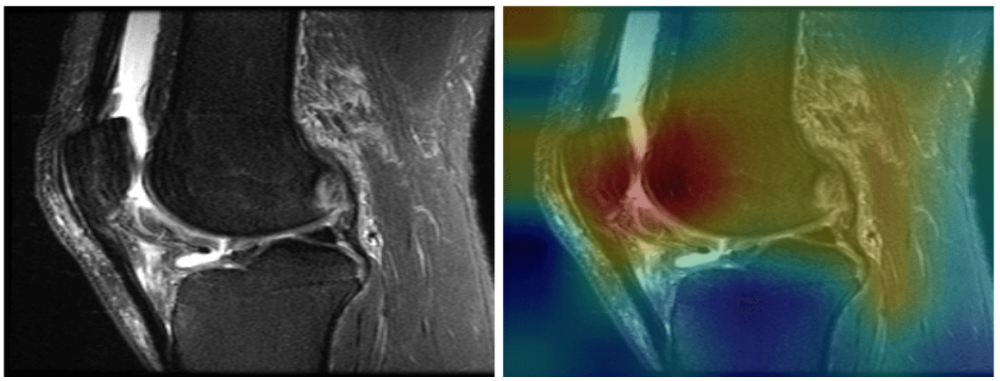
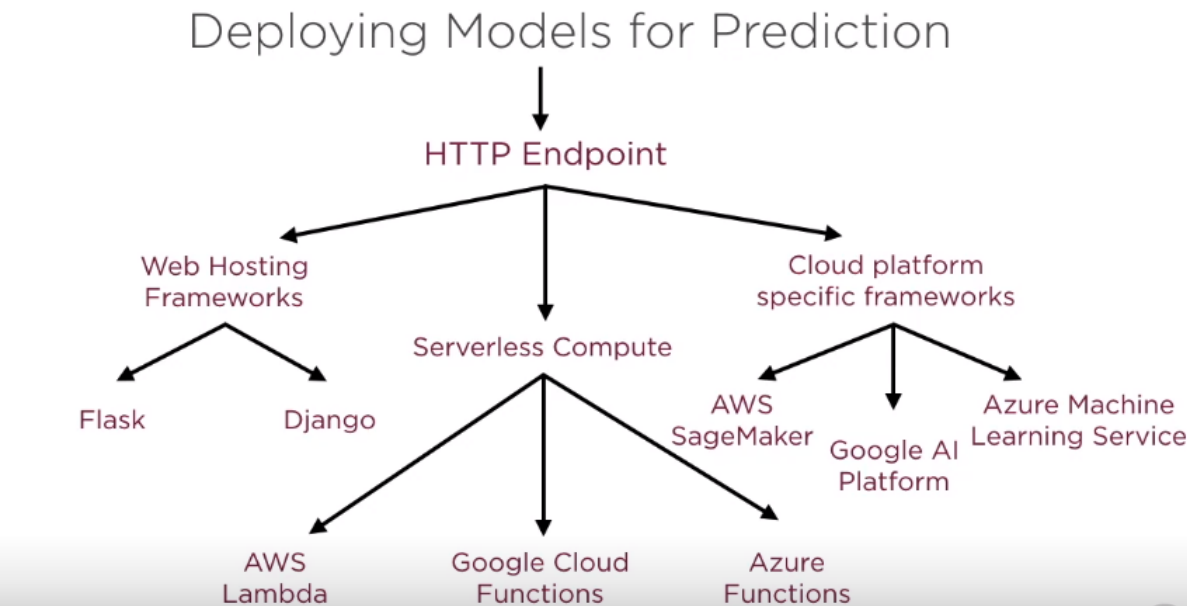
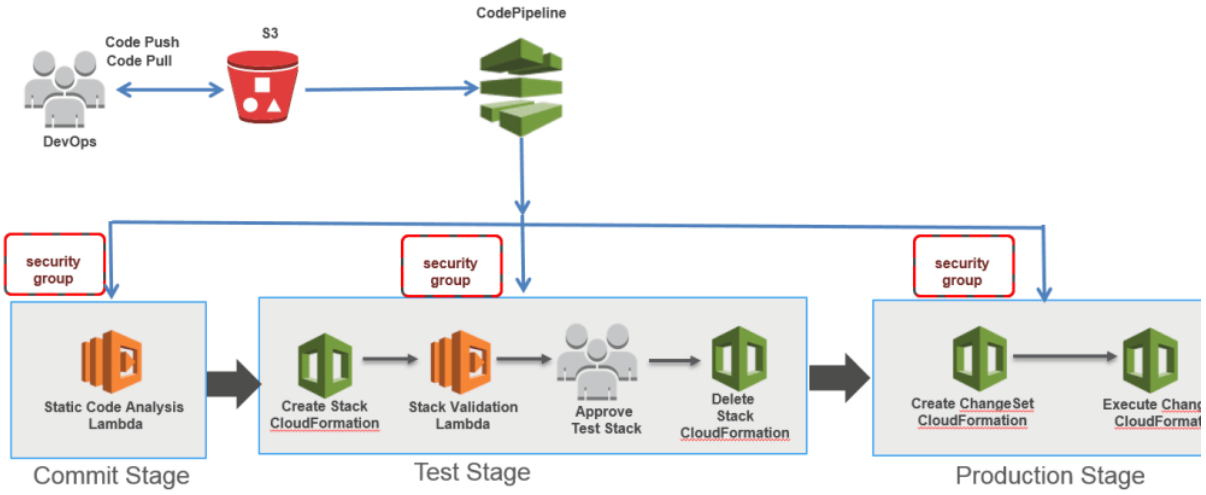
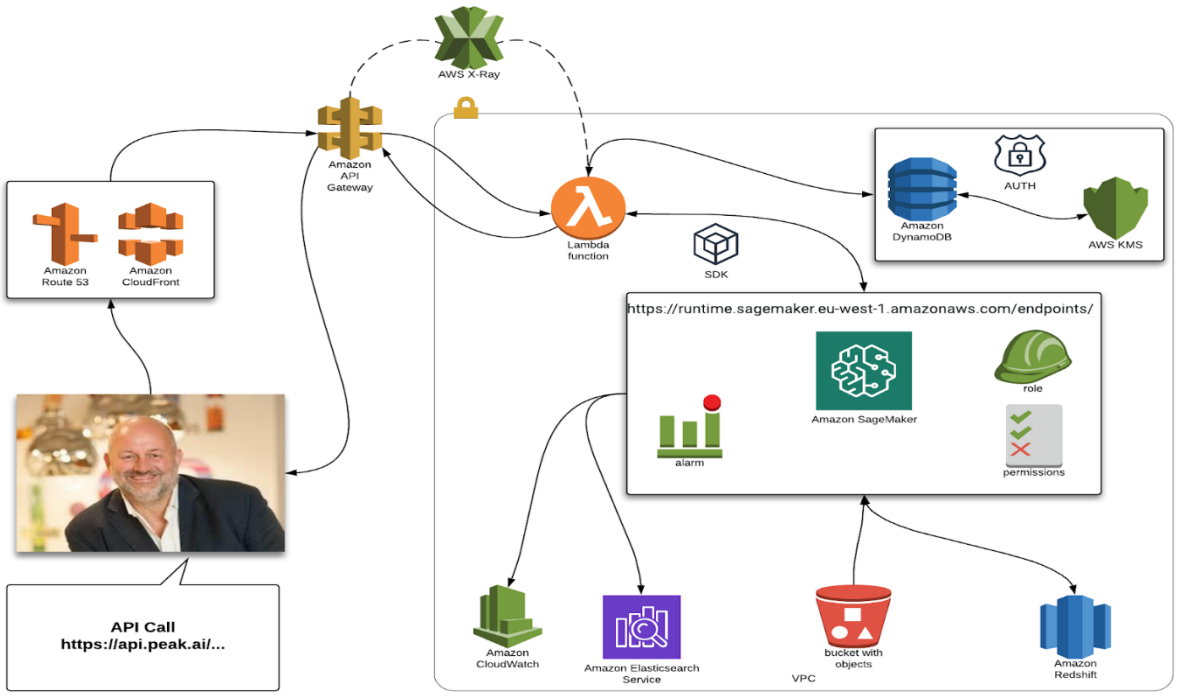


Figure 11: Attention maps of KNEE MRI injury

3.3 Deployment

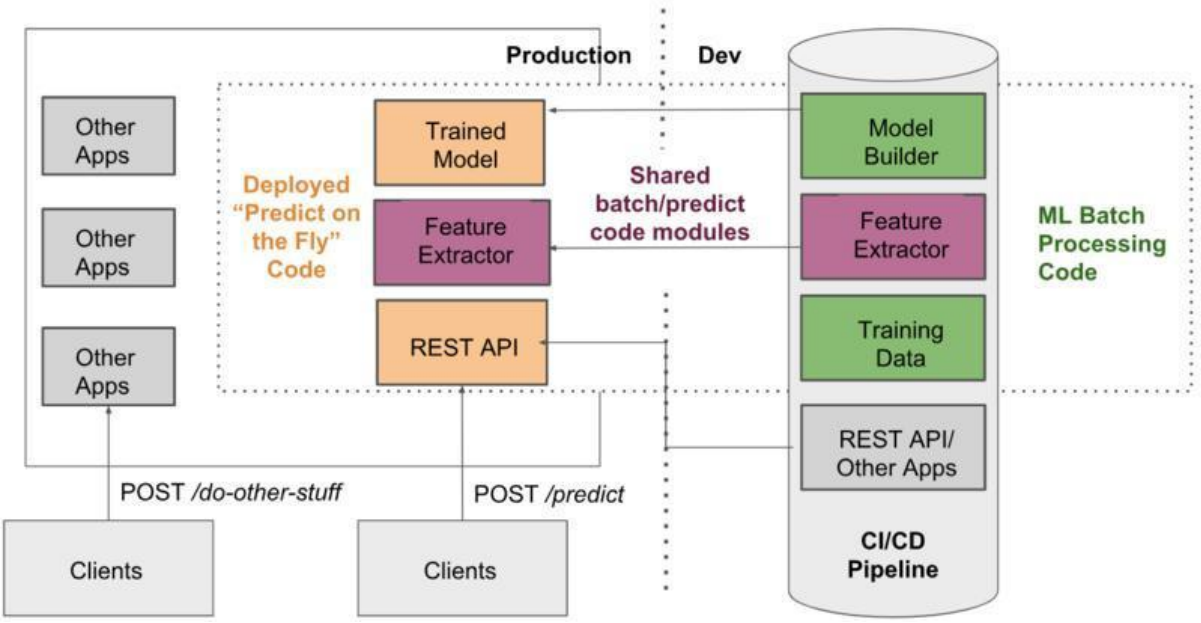



CodePipeline
S3
Code Push
Code Pull
DevOps
security group
security group
security group
Static Code Analysis Lambda
Create Stack CloudFormation
Stack Validation Lambda
Approve Test Stack
Delete Stack CloudFormation
Create ChangeSet CloudFormation
Commit Stage
Test Stage
Production Stage


AWS X-Ray
Amazon API Gateway
Amazon Route 53
Amazon CloudFront
Lambda function
SDK
Amazon DynamoDB
AUTH
AWS KMS
https://runtime.sagemaker.eu-west-1.amazonaws.com/endpoints/
Amazon SageMaker
role
alarm
permissions
API Call
https://api.peak.ai/...
Amazon CloudWatch
Amazon Elasticsearch Service
bucket with objects
VPC
Amazon Redshift

Amazon SageMaker

Amazon SageMaker is a cloud machine-learning platform that was launched in November 2017. SageMaker enables developers to create, train, and deploy machine-learning models in the cloud. SageMaker also enables developers to deploy ML models on embedded systems and edge-devices.



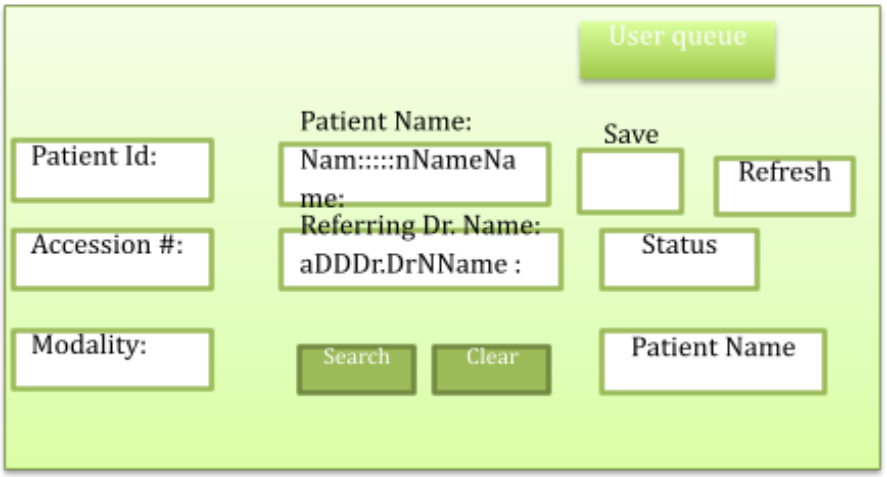
GUI SCREENS

Stepwise format of Sample GUI Screens:

Step 1: **Sign up** form

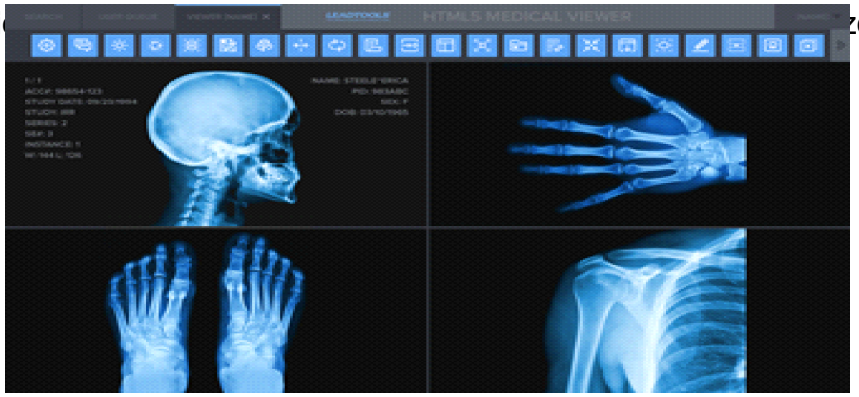
Step 2: To **login** using these inputs

Step 3: **Search** the scan from data base

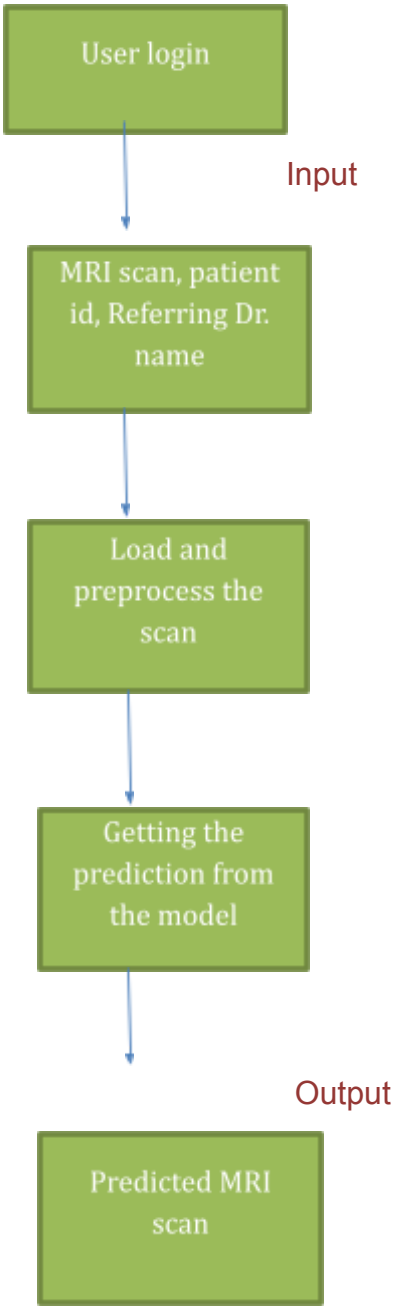


Step 4: Already studied scan Will be stored in Studies table

Pt. id	Pt. name	Access--ion	Ref. Dr name	Modality

Step 5: **vi**  zoom, rotate, arrows etc.

User I/O workflow:



2.6.3 Sample GUI

(Source)

(Source)

GUI Screens: Stepwise format of Sample GUI Screens:

3. References

1. “MRNet Dataset.” Stanford ML Group, https://stanfordmlgroup.github.io/competitions/mrnet/
2. “Mrnet Dataset.” Kaggle, https://www.kaggle.com/cjinny/mrnet-v1
3. ImageDataGenerator https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator
4. Azcona, D., McGuinness, K., & Smeaton, A. F. (2020, October). A Comparative Study of Existing and New Deep Learning Methods for Detecting Knee Injuries using the MRNet Dataset. In 2020 International Conference on Intelligent Data Science Technologies and Applications (IDSTA) (pp. 149-155). IEEE.
5. Liu, F., Guan, B., Zhou, Z., Samsonov, A., Rosas, H., Lian, K., Sharma, R., Kanarek, A., Kim, J., Guermazi, A., & Kijowski, R. (2019). Fully Automated Diagnosis of Anterior Cruciate Ligament Tears on Knee MR Images by Using Deep Learning. Radiology. Artificial intelligence, 1(3), 180091. https://doi.org/10.1148/ryai.2019180091
6. tf-keras-vis toolkit, https://pypi.org/project/tf-keras-vis Last accessed 30 Aug 2020
7. Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva and Antonio Torralba. Learning Deep Features for Discriminative Localization, 2015; arXiv:1512.04150.
8. Karen Simonyan, Andrea Vedaldi and Andrew Zisserman. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, 2013; arXiv:1312.6034.