JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

R-CNN based Diagnosis of Tuberculosis from Chest X-ray Images

Srinivasa Raghavan S,Muthukumaran S, Manju Bala P
Student, Student, Associative Professor
IFET college of Engineering

CHAPTER 1 INTRODUCTION

Medical X-rays are images are generally used to diagnose some sensitive human body parts such as bones, chest, teeth, skull, etc. Medical experts have used this technique for several decades to explore and visualize fractures or abnormalitiesin body organs. This is due to the fact that X-rays are very effective diagnostic tools in revealing the pathological alterations, in addition to its non- invasive characteristics and economic considerations. Chest diseases can be shown in CXR images in the form of cavitations, consolidations, infiltrates, blunted costophrenic angles, and small broadly distributed nodules. The interpretation of a chest X-ray can diagnose many conditions and diseases such as pleurisy, effusion, pneumonia, bronchitis, infiltration, nodule, atelectasis, pericarditis, cardiomegaly, pneumothorax, fractures and many others.

Classifying the chest x-ray abnormalities is considered a tough task for radiologists. Hence, over the past decades, computer aided diagnosis (CAD) systems have been developed to extract useful information from X-rays to help doctors in having a quantitative insight about an X-ray. However, those CADsystems haven't achieved a significance level to make decisions on the type of conditions of diseases in an X-ray. Thus, the role of them was left as visualization functionality that helps doctors in making decisions.

Recently, accurate images classification has been achieved by deep learning based systems. Those deep networks showed superhuman accuracies in performing such tasks. This success motivated the researchers to apply those networks on medical images for diseases classification tasks and the results showed that deep networks can efficiently extract useful features that distinguish different images classes. Convolutional neural networks have been applied to various medical imagesdiagnosis and classification due to its power of extracting different level features from images.

Traditional networks have been also used in classifying medical diseases, however, their performance was not as efficient as the deep networks in terms of accuracy, computation time, and minimum square error achieved. In this work, deeplearning based networks are employed to classify most common thoracic diseases. Two Region based Convolutional Neural Network are examined in this study to classify the chest X-rays into two common classes: normal and abnormal which mayhave different types of diseases that may be found in chest X-ray, i.e, Atelectasis, Cardiomegaly, Effusion, Infiltration, Mass, Nodule, Pneumonia, Pneumothorax, Consolidation, Edema, Emphysema, Fibrosis. In this work, we aim to train the deep network on the same number of chest X-ray images and evaluate their performances in classifying different chest X-rays. The data used in obtained from two public databases which are the Shenzhen Hospital X-ray Set / China data set: X-ray images in this data set (Fushman et al., 2005), and the Montgomery County X- ray Set (Fushman et al., 2016).

1.1 AIM OF THE PROJECT

Doctors and radiologists are still categorizing the chest X-rays in a manual manner based on some visual examinations. Therefore, There is a need for an automatic and intelligent systems that has the capability of accurate classification of chest X-rays into normal and abnormal images. Thus, in this work we aim to use a powerful deep network in such classification tasks. The deep network that is selected to be used in called the Region based Convolutional Neural Network which showed a great efficacy in different classification tasks in the medial field. Furthermore, the network is also examined on enhanced and unenhanced images,

which aims to demonstrate the effects of medical image processing and enhancement on the performance of neural network.

1.2 SIGNIFICANCE OF THE STUDY

As most of the previous work are conducted using convolutional neural networks (CNN) and support vector machine (SVM), this work is ought to investigate the performance of other types of deep networks in classifying the abnormality of chest X-rays. Thus, a deep network named as Region based Convolutional Neural Network is selected to be used as the brain behind this study. This network is expected to well perform since it has been applied to various medical classification tasks where it achieved high accuracies and because the data available are enough for training it and achieving a small error.

CHAPTER 2 LITERATURE SURVEY

2.1 INTRODUCTION

In this chapter, a brief review of applications in medical imaging, and diagnosisis presented. Also, the applied approach to the classification of the segmented images,pattern recognition, is introduced. Furthermore, background on related image processing and feature extraction techniques as are considered in this thesis are discussed sufficiently. Artificial neural networks, the backbone of machine learning, and which has also been used extensively in this work for the classification phase are introduced; including the particular algorithms for the supervised and unsupervised learning.

2.2 RELATED WORKS

In a past work, described the segmentation of chest X-ray using convolutional neural network. In their work, they introduced image segmentation into bone tissue and non-bone tissue. The aim of their work was to develop an automatic or an intelligent segmentation system for chest X-rays. The system was established to have the capability to segment bone tissues from the rest of the image.

They were able to achieve the aim of the research by using a convolutional neural network, which was tasked with examining raw image pixels and hence classifying them into "bone tissue" or "non-bone tissue". The convolutional neural networks were trained on the image patches collected from the chest X-ray images.

It was recorded in their work that the automatic segmentation of chest X-rays using the convolutional neural networks, and approaches suggested in their research produced plausible performance.

In another recent research, "lung Cancer Classification using Image Processing",presented the application of some image processing techniques in the classification ofpatients chest X-rays into whether cancer is present or not (benign or malignant). In this work, it was shown that by extracting some geometric features that are essential to the classification of the images such area, perimeter, diameter, and irregularity; an automatic classification system was developed.

Furthermore, in the same research, texture features were considered for a parallel comparison of results on the classification accuracy. The texture features used in the work are average gray level, standard deviation, smoothness, third moment, uniformity, and entropy. The back propagation neural network was used as the classifier, and an accuracy of 83% was recorded in the work (Patil and Kuchanur, 2018).

In this thesis the classification of chest X-ray radiographs into two classes has been achieved using artificial neural networks. The two classes are the normal one which has no disease or conditions, and the abnormal one which may have any types of diseases that may encounter the chest organs including heart, lungs. Chest etc… A deep network called Region based Convolutional Neural Network is used for (RCNN), which relies on a supervised and unsupervised learning algorithms was used to train the network on the images collected for the research.

A few works have applied deep learning techniques for chest X-ray images using different datasets. The work by Shin et al. (Shin et al., 2016) used CNN with Recurrent Neural Network (RNN) to classify and annotate diseases from chest x-ray images by utilizing OpenI dataset. Then, Bar et al. (Bar et al., 2016) made use of Sheba Medical Center dataset applied CNN (Donahue et al., 2014) with Kruskal– Wallis feature selection (Hollander & Wolfe, 1999). Meanwhile, Islam et al. (Islam, Aowal, Minhaz, & Ashraf, 2017) explored DCNN to localize features on chest x-ray images by utilizing severaldatasets from Indiana, JSRT and Shenzhen datasets.

There are several works (X. Wang et al., 2017b) ,(Rajpurkar et al., 2017b) that utilized similar dataset, namely, ChestX-ray8 dataset was releasedby Wang et al. (X. Wang et al., 2017b) and later, it is known as ChestX-ray14. In (X. Wang et al., 2017b), they applied Deep CNN to classify those images and predict the related eight diseases. It is then followed by several works, namely Rajpurkar et al. (Rajpurkar et al., 2017b) proposed CheXNet's algorithm consists of 121 layers of CNN, and Yao etal. (Yao et al., 2017) that proposed 2D ConvNet that combined a densely connected image encoder witha recurrent neural network model decoder.

Furthermore, Guan et al. (Guan et al., 2018) proposed a three- branch Attention Guided CNN which learns froma global CNN branch, Gundel et al. (Gundel et al., 2018) proposed a Location Aware Dense Networks to classify pathologies in chest X-ray images, and Sedai et al. (Sedai et al., 2018) that proposed a weakly supervised method based on CNN which can predict the location of chest pathologies in x-ray images. Meanwhile, Abiyev et al. (Abiyev & Ma'aitah, 2018) presented some comparison between Backpropagation neural network, competitive neural network and CNN on classifying chest x-ray images.

A few enhanced models of CNN based on region proposals has been proposed to address localization problem. In particular, Girshick et al. (Girshick et al., 2014) proposed Regional-CNN to improve the precision of localization. Meanwhile, Wang et al. (L. Wang et al., 2017) they addressed guide-wire detection based on real-time approach with CNN. Furthermore, Regional Proposal Network is used to extract guide-wires features. R-CNN is then been enhanced with Fast R-CNN (Girshick, 2015) to address the performance issue.

It is then followed by Faster R-CNN (Ren et al., 2017) to further improve its performance. The work by Ruhan Sa et al. (Sa et al., 2017) has applied Faster R-CNNfor detecting landmark points in lateral lumbar x-ray images. They demonstrated thata higher accuracy can be achieved even with a small dataset of x-ray images by tuning the deep learning network on naturalimages. Our work follows this direction by focusing on the chest x-ray images for addressing the classification.

2.3 FEATURES EXTRACTION IN MEDICINE

Pattern recognition is the process of developing systems that have the capability to identify patterns; while patterns can be seen as a collection of descriptive attributes that distinguishes one pattern or object from the other. Itis the study of how machines perceive their environment, and therefore capable of making logical decisions through learning or experience. During the development of pattern recognition systems, we are interested in the manner in which patterns are modeled and hence knowledge represented in such systems. Several advances in machine vision have helped revamp the field of pattern recognition by suggesting novel and more sophisticated approaches to representing knowledge in recognition systems; building on more appreciable understanding of pattern recognition as achieved in the human visual processing.

Typical pattern recognition as the following important phases for the realization of its purpose for decision making or identification.

- Data acquisition: This is the stage in which the data relevant to the recognition task are collected.
- Pre-processing: It is at this stage that the data received in the data acquisitionstage is manipulated into a form suitable for the next phase of the system. Also, noise is removed in this stage, and pattern segmentation may be carried out.
- Feature extraction/selection: This stage is where the system designer determines which features are significant and therefore important to thelearning of the classification task.
- Features: The attributes which describe the patterns.
- Model learning/ estimation: This is the phase where the appropriate model for the recognition problem is determined based on the nature of the application.
- The selected model learns the mapping of pattern features to their corresponding classes.
- Model: This is the particular selected model for learning the problem, the model is tuned using the features extracted from the preceding phase.
- Classification: This is the phase where the developed model is simulated with patterns for decision making. The performance parameters used for accessing such models include recognition rate, specificity, accuracy, and achieved mean squared error (MSE).
- Post-processing: The outputs of the model are sometimes required to be processed into a

form suitable for the decision making phase stage. Confidence in decision can be evaluated at this stage, and performance augmentation may be achieved.

- Decision: This is the stage in which the system supplies the identification predicted by the developed model.

There exist several approaches to the problem of pattern recognition such as syntactic analysis, statistical analysis, template matching, and machine learning usingartificial neural networks.

Syntactic approach uses a set of feature or attribute descriptors to define a pattern, common feature descriptors include horizontal and vertical strokes, term stroke analysis; more compact descriptors such as curves, edges, junctions, corners, etc., which is termed geometric features analysis. Generally, it is the job of the system designer to craft such rules that distinguish one pattern or object from another.The designer is meant to explore attribute descriptors which are unique to identify each pattern, and where there seems to a conflict of identification rules such ascan be observed in identifying they have same geometric feature descriptors save that one is the inverted form of the other, the system designer is meant to explore other techniques of resolving such issues (Yumusak and Temurtas, 2017). Statistical pattern analysis uses probability theory and decision to infer the suitable model for the recognition tasks.

Template pattern matching uses the technique of collecting perfect or standard examples for each distinct pattern or object considered in the recognition task. It is with these perfect examples that the test patterns are compared. It is usually the work of the system designer to craft the techniques with which pattern variations or dissimilarities from the templates are measured, and hence determine decision boundaries as to accept or reject a pattern being a member of a particular class. Euclidean distance is a common used function to measure the distance between two vectors in n- dimensional space.

Template matching can either be considered as global or local depending on the approach and aim for which the recognition system is designed. In global template matching, the whole pattern for recognition is used to compare the whole perfect example pattern; whereas in local template matching, a region of the pattern for classification is used to compare acorresponding region in the perfect template.

Artificial neural networks, on the other hand, are considered intelligent pattern recognition systems due to their capability to learn from examples ina phase known as training. These systems have sufficed in lots of pattern recognition systems; the ease with which same learning algorithms can be applied to various recognition tasks is motivating.

In this approach, the designer is allowed to focus on determining features to be extracted for learning by the designed systems, rather than expending a huge amount of time, resources, and labour in understanding the whole details of the application domain; instead, the system learns relevant features that distinguish one pattern from the other.

CHAPTER 3 EXISTING SYSTEM

3.1 INTRODUCTION

Computer aided diagnosis (CAD) systems can play important role in the mass screening of pulmonary TB by analyzing the chest X-ray images. The accessibility of large-scale labelled datasets and deep convolutional neural networks (CNNs) has led to huge success in image recognition. CNNs allow data-driven, highly representative, hierarchical image features to be learned from adequate training data, but obtaining data sets in the medical imaging domain as comprehensively annotated, as ImageNet remains a challenge. It is also worth noting that the healthcare sector is entirely different from any other field being a high priority sector with customers willing to pay for highest quality of care and services.

3.2 METHOD

This method consists of a series of steps, including transfer learning, freezing network, feature extraction and classifi- cation tasks using supervised learning. We conducted three experiments to verify the feasibility of our method. In the first study, combining the coordinate attention mechanism with VGG16 network, they usedVGG16-CoordAttention network model, which is trained by freezing the network to evaluate its classification effect on the Shenzhen dataset. In the second study, they used four representative models and our model as the backbone network to evaluate the effect of frozen network training. In the third study, we conduct five crossvalidations of the proposed method to verify the robustness of the model.

3.3 DISADVANTAGES

- Other classification models having less accuracy for the Chest X rayclassification compared to R-CNN classifier.
- Compared to our Proposed system the existing system had high data loss ratio.
- Existing system algorithm have few layers compared to Proposed model.

CHAPTER 4 PROPOSED SYSTEM

4.1 INTRODUCTION

The overall methodology of this is illustrated using Architecture. Two different databases were created for this study. One was for lung segmentation while the other one was for TB classification. The classification technique was implemented on a python platform and dimensions, maximum iterations, perplexity- effective number of neighbors etc. parameters were modified from default values in order to confirm the performance of the best network.

4.2 NETWORKS TRAINING AND PERFORMANCE EVALUATION

4.2.1 Methodology:

This study presents an original research for the diagnosis of chest X- rays using deep learning. A deep network named as Region based Convolutional Neural Network (RCNN) is selected to be used as the brain this work. This selection came from the few researches that were conducted for the chest X-rays classification using this kind of networks. Thus, there is a need to investigate the effectiveness and performance of Region based Convolutional Neural Network in classifying the chest X-rays and detecting whether a radiograph has a disease or it is normal (healthy).

Two auto-encoder networks were used to build the proposed Region basedConvolutional Neural Network that is then used to be as the intelligent classifier of the chest X-ray images. The auto-encoder was first trained layer by layer using greedy layer wise training until a network of two hidden layer, one input, and one output network is formed. Therefore, these trained auto-encoders were all stacked together and the proposed Region based Convolutional Neural Network is formed.

The proposed network is trained to classify chest images into normal which have no abnormalities or diseased images regardless of the type of the disease. A sample of the database normal and abnormal chest X-rays is shown in Figure 4.2.

Note that in this work, two deep models are employed. Both models are Region based Convolutional Neural Networks with the same learning parameters, however, for the first model, which we call RCNN1, the chest X-rays are fed directly into network, without processing and enhancement. The second network model, which is called RCNN2, was trained on images that are processed and enhanced before being fed into network. The aim of the use of two models is to investigate the effects of processing and image enhancement on the auto-encoder training and testing performance.

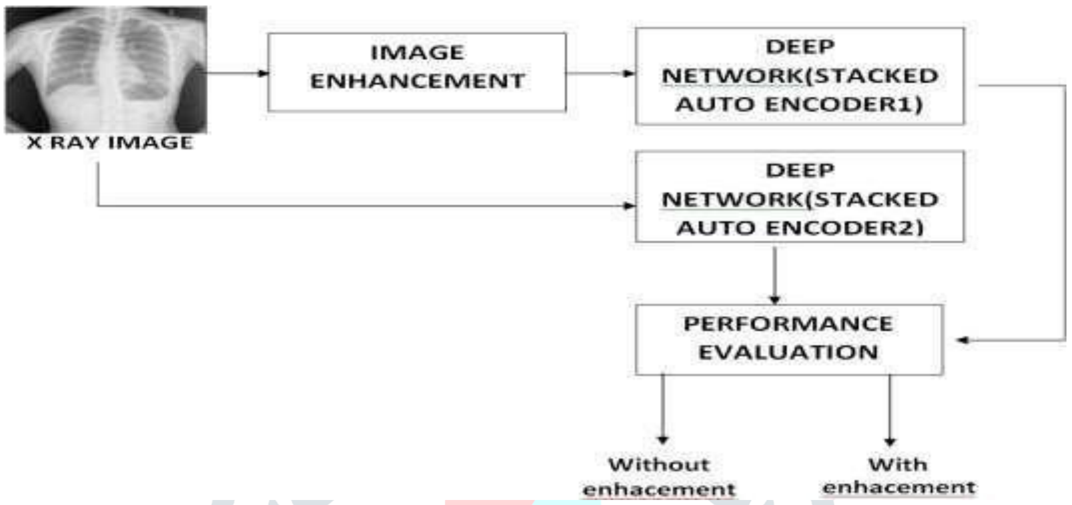


Fig 4.1: Flowchart of the proposed methodology

Figure 4.1 shows the workflow of the proposed methodology. As seen, the network model is trained first on the chest images without enhancement and the network is then tested and the performance is evaluated. Same network is then trained and tested on same images but here they are enhanced using image Histogram equalization and similarly, the network is also evaluated and tested in order to investigate the one that outperforms in terms of accuracy and less error achieved.

Image Enhancement

Image enhancement is the process of adjusting digital images so that the results are more suitable for display or further image analysis. The aim of image enhancement is to improve the interpretability or perception of information in images for human viewers, or to provide `better' input for other automated image processing techniques. Image enhancement is the modification of image, by changing the pixel brightness values, to improve its visual impact. Image enhancement techniques are performed by deriving the new brightness value for a pixel either from its existing value or from the brightness values of a set of surrounding pixels.

Stack Auto Encoder

Stack Auto encoder is a kind of unsupervised learning structure that owns three layers: input layer, hidden layer, and output layer. Some datasets have a complex relationship within the features. Thus, using only one Autoencoder is not sufficient.A single Autoencoder might be unable to reduce the dimensionality of the input features. Therefore for such use cases, we use stacked autoencoders. Autoencoders are a type of deep learning algorithm that are designed to receive an input and transform it into a different representation. They play an important part in image construction.

Performance Evaluation

A performance evaluation is a meaningful and computable measure used for quantitatively evaluating. the performance of any algorithm. Consider the process of. assessing image quality.

4.2.2 Database:

A deep network an intelligent classifier that is hungry for data. The more data it is trained on the more intelligent it will be. Therefore, there is need for a good database that has good number of normal and abnormal images to train and test the developed network.

The first database contains chest X-rays of both normal and abnormal cases and they were acquired aspart of the routine care at Shenzhen Hospital. The images are of JPEG format and there 340 normal x-rays and 275 abnormal x-rays showing various aspects of tuberculosis.

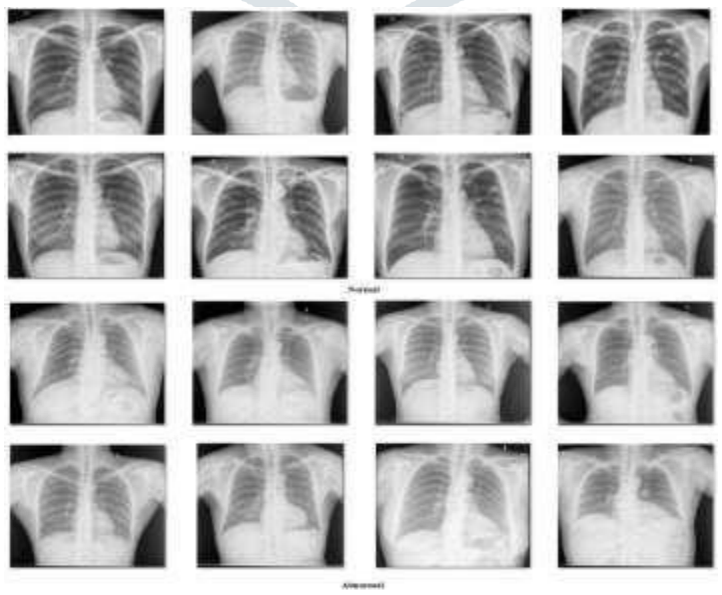


Fig 2: A sample of the database images

Figure 2 shows a sample of the chest X-rays found in the database used for training and testing the employed models. Note that the figure shows the two classes of the database images: the normal and the abnormal.

Table 1 shows the number of images found in the database in addition to the training and testing ratios used.

Table 4.1: Dataset 1 and data division

Image sets	No. of images	Normal images	Abnormal images
Training sets	400	200	200
Testing sets	215	140	75
Both sets	615	340	275

Table 2 shows the number of X-rays in the dataset 2 and its data division.

Table 4.2: Dataset 2 and data division

Image sets	**No. of images**	**Normal images**	**Abnormal images**
Training sets	70	40	30
Testing sets	68	40	28
Both sets	138	80	58

4.2.3 Training the Deep Models

In this section the training of the two deep models which are RCNN1 and RCNN 2 is discussed. Note that the RCNN1 is the Region based Convolutional Neural Network network that uses X- ray images without enhancement While RCNN2 is the same network but with enhanced images as inputs. It is important to mention that RCNN1 and RCNN2 are both trained on the same number of images which is 470 images; among them 240 are normal and 230 are abnormal.

Note that the networks are first pre-trained as they are deep networks. Pre- training means that the networks are first trained layer by layer using Greedy-layer wise training (Hinton, 2006). In this phase there is no output labeling because the network is trained here to reconstruct its input from the extracted features in the hidden layer, which is why the number of output neurons is equal to the number of input neurons which is 4096.

Once the networks finish pre-training, it is then fine-tuned using the conventional backpropagation algorithm. Here, the input images are labeled therefore, output neurons are two which means that network is being trained to classify the images into two classes: normal and abnormal.

Table 3 shows the training and testing data from the two datasets. It is seen that the network is trained on 470 images and tested on 283 chest X- rays.

Table 4.3: Training and testing data from the two databases

Image sets	No. of images	Training	Testing
Dataset 1	615	400	215
Dataset 2	138	70	68
Both sets	753	470	283

4.2.4 RCNN1 Training

RCNN1 is a Region based Convolutional Neural Network that is trained on 470 images that are fed into it without any processing or enhancement technique. This deep model is composed of one input layer of 4096 neurons since the input images size is 64*64 pixels; two hidden layers of 100 and 65 neurons, respectively. Also, it has an output layer of two neurons as the output classes are only two. Figure 15 shows the architecture of the RCNN1. Table 4 shows the values of the learning parameters of theRCNN1 when it is trained on 200 images.

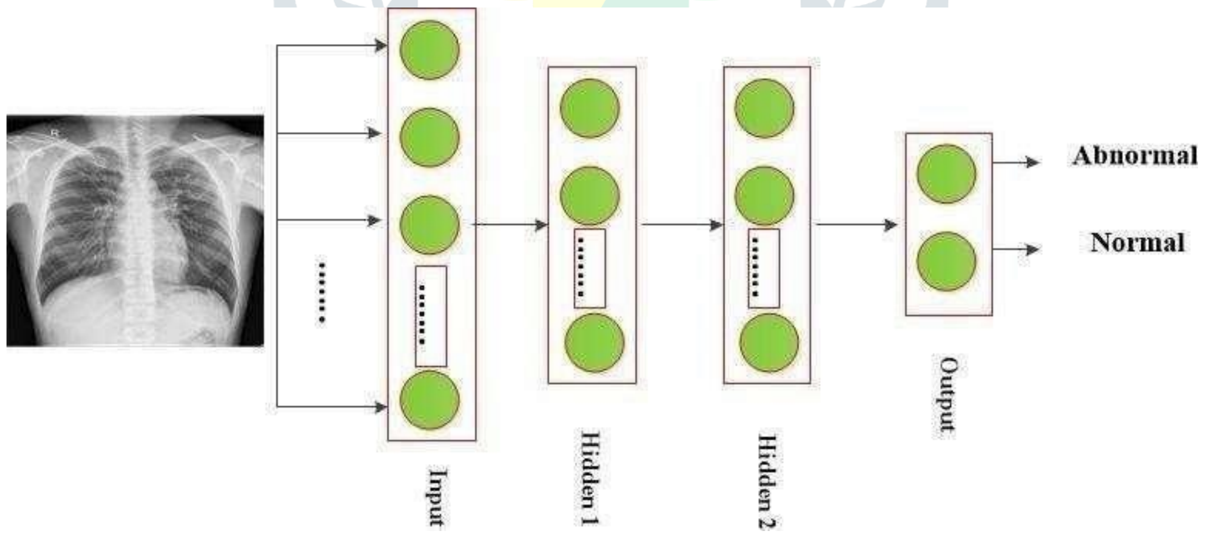


Fig. 4.3: The deep network model 1 (RCNN1) structure

4.2.5 UNDERSTANDING REGION-BASED CONVOLUTIONAL NEURAL NETWORK

Instead of working on a massive number of regions, the RCNN algorithm proposes a bunch of boxes in the image and checks if any of these boxes contain any object. RCNN uses selective search to extract these boxes from an image (these boxes are called regions).

Let’s first understand what selective search is and how it identifies the different regions. There are basically four regions that form an object: varying scales, colors, textures, and enclosure. Selective search identifies these patterns in the image and based on that, proposes various regions. Here is a brief overview of how selective search works:

- It first takes an image as input.
- Then, it generates initial sub-segmentations so that we have multiple regions from this image.
- The technique then combines the similar regions to form a larger region (based on color similarity, texture similarity, size similarity, and shape compatibility).
- Finally, these regions then produce the final object locations (Region of Interest).

Below is a succint summary of the steps followed in RCNN to detect objects:

1. We first take a pre-trained convolutional neural network.
2. Then, this model is retrained. We train the last layer of the network based on the number of classes that need to be detected.
3. The third step is to get the Region of Interest for each image. We then reshape all these regions so that they can match the CNN input size.
4. After getting the regions, we train SVM to classify objects and background. For each class, we train one binary SVM.
5. Finally, we train a linear regression model to generate tighter bounding boxes for each identified object in the image.

4.3 ADVANTAGES OF PROPOSED SYSTEM

- R-CNN classification having high accuracy for the chest X ray classification compared to proposed system algorithm.
- Low data loss Ratio.
- Prediction algorithm having huge layers.

4.4 SYSTEM ARCHITECTURE

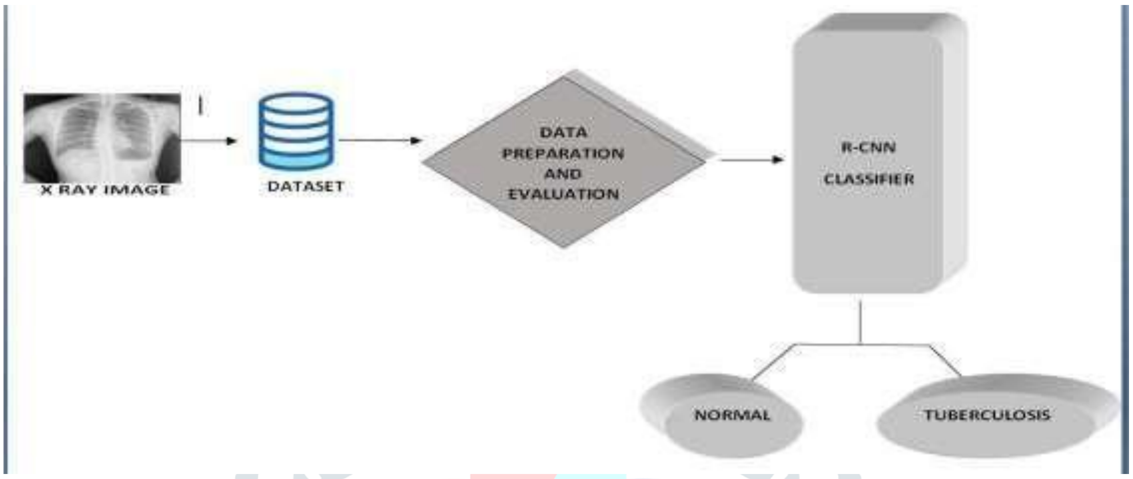


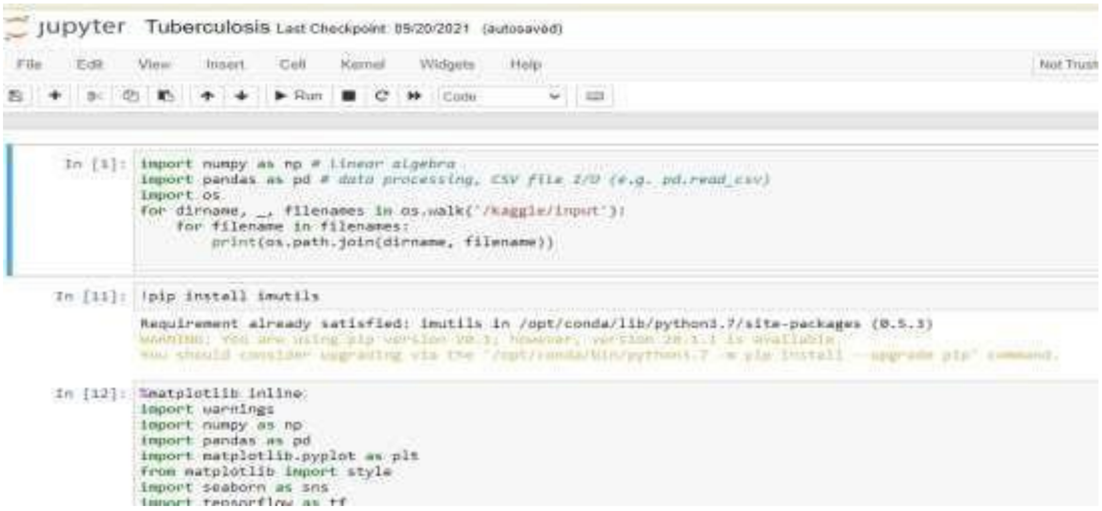
Fig 4.4: System Architecture

4.5 MODULE DESCRPTIONData Collection:

❖ This is the first real step towards the real development of a machine learning model, collecting data. This is a critical step that will cascade in how good the model will be, the more and better data that we get, the better our model will perform.

❖ There are several techniques to collect the data, like web scraping, manual interventions and etc.

❖ Comparison of Deep Learning Algorithms for Predicting Ibs Images taken from kaggle and some other source.


jupyter Tuberculosis Last Checkpoint: 05/20/2021 (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Not Trust
Run
Code

```
In [1]: import numpy as np # linear algebra
        import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
        import os
        for dirname, _, filenames in os.walk('/kaggle/input'):
            for filename in filenames:
                print(os.path.join(dirname, filename))

In [11]: !pip install imutils

         Requirement already satisfied: imutils in /opt/conda/lib/python3.7/site-packages (0.5.3)
         WARNING: You are using pip version 20.3; however, version 20.1.1 is available.
         You should consider upgrading via the '/opt/conda/bin/python3.7 -m pip install --upgrade pip' command.

In [12]: %matplotlib inline
         import warnings
         import numpy as np
         import pandas as pd
         import matplotlib.pyplot as plt
         from matplotlib import style
         import seaborn as sns
         import tensorflow as tf
```

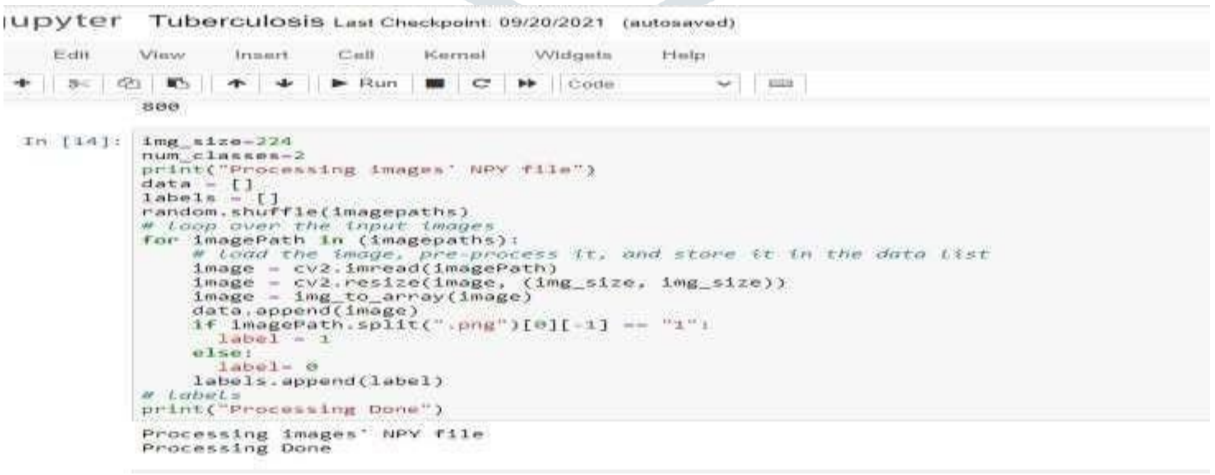
Fig 4.5 Collect the data from the kaggle

Data Preparation:

❖ Data preparation is a crucial step in developing accurate and reliable ɜep learning models for x-ray image classification. By carefully selecting, cleaning, and preprocessing the data, you can ensure that the model is trained on high- quality data that is relevant to the classification task at hand.

❖ Image resizing involves scaling the images to a consistent size, while normalization involves transforming the pixel values to a common range. Data augmentation techniques include random cropping, rotation, and flipping, which can increase the diversity of the dataset and improve the model's robustness.

Fig 4.6 Data preprocessing


jupyter Tuberculosis Last Checkpoint: 09/20/2021 (autosaved)
Edit View Insert Cell Kernel Widgets Help
Run
Code

```
         800

In [14]: img_size=224
         num_classes=2
         print("Processing images' NPY file")
         data = []
         labels = []
         random.shuffle(imagepaths)
         # Loop over the input images
         for imagePath in (imagepaths):
             # load the image, pre-process it, and store it in the data list
             image = cv2.imread(imagePath)
             image = cv2.resize(image, (img_size, img_size))
             image = img_to_array(image)
             data.append(image)
             if imagePath.split(".png")[0][-1] == "1":
               label = 1
             else:
               label= 0
             labels.append(label)
         # Labels
         print("Processing Done")

         Processing images' NPY file
         Processing Done
```

Model Selection:

❖ While creating a machine learning model, we need two dataset, one fortraining andother for testing. But now we have only one. So lets split this in two with a ratio of 80:20. We will also divide the data frame into feature column and labelcolumn.

❖ Here we imported train_test_split function of sklearn. Then use it to split the dataset. Also, *test_size = 0.2*, it makes the split with 80% as train dataset and 20% as test dataset.

❖ The Deep_state parameter seeds random number generator thathelps to splitthe dataset.

❖ The function returns four datasets. Labelled them as train_x, train_y,test_x, test_y.If we see shape of this datasets we can see the split of dataset.

❖ We will use R-CNN Classifier, which fits multiple decision tree to the data. Finally Itrain the model by passing train_x, train_y to the fit method.

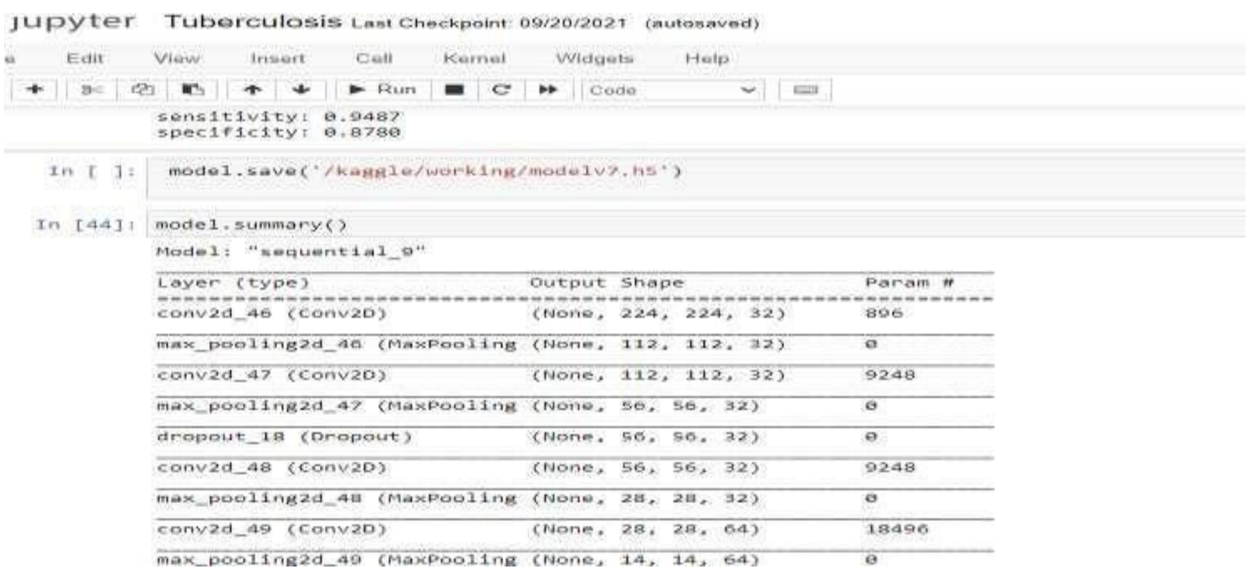
❖ Once the model is trained, we need to Test the model. For that we willpass test xto the predict method.

Saving the trained Model:

❖ Once you're confident enough to take your trained and tested modelinto the production-ready environment, the first step is to save it into a .h5 or . pkl fileusing a library like pickle.

❖ Make sure you have pickle installed in your environment.

❖ Next, let's import the module and dump the model into .pkl file.


jupyter Tuberculosis Last Checkpoint: 09/20/2021 (autosaved)
Edit
View
Insert
Cell
Kernel
Widgets
Help
Run
Code

```
sensitivity: 0.9487
specificity: 0.8780

In [ ]:  model.save('/kaggle/working/modelv7.h5')

In [44]: model.summary()
Model: "sequential_9"
Layer (type)                 Output Shape              Param #
================================================================
conv2d_46 (Conv2D)           (None, 224, 224, 32)      896
max_pooling2d_46 (MaxPooling (None, 112, 112, 32)      0
conv2d_47 (Conv2D)           (None, 112, 112, 32)      9248
max_pooling2d_47 (MaxPooling (None, 56, 56, 32)        0
dropout_18 (Dropout)         (None, 56, 56, 32)        0
conv2d_48 (Conv2D)           (None, 56, 56, 32)        9248
max_pooling2d_48 (MaxPooling (None, 28, 28, 32)        0
conv2d_49 (Conv2D)           (None, 28, 28, 64)        18496
max_pooling2d_49 (MaxPooling (None, 14, 14, 64)        0
```

Fig 4.7 Data model saving and summary

4.6 SYSTEM IMPLEMENTATION

4.6.1 Data Preparation

A few chest x-ray image datasets can be used to research about chest x- ray image classification as mentioned in. In this work, we have chosen ChestX-ray dataset from kaggle to complement the related works that have applied CNN for chest x-ray image classification and detection.

Each image has been annotated with one of different tuberculosis category. These images were scaled and standardized into 1024 x 1024 pixels. Then the images were changed to grayscale color since some of the images were in blue color.

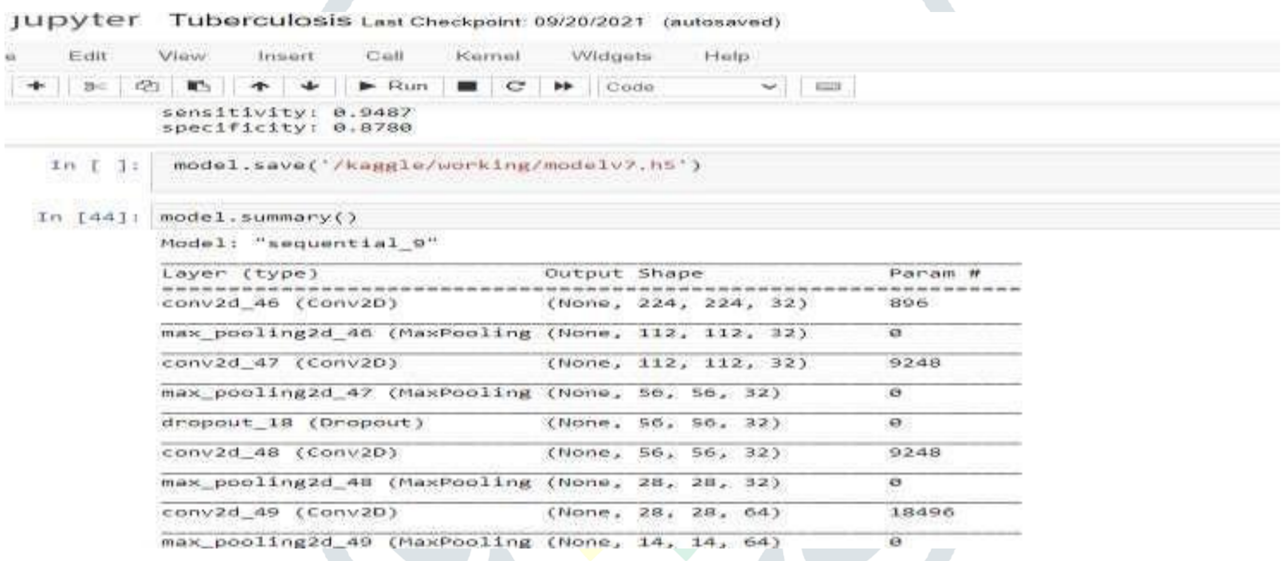
In relation to this work, we have decided to select 200 chest x-ray images randomly because we have to manually set the regions' labels for the training purpose. For labeling, we have used Labellum application.

In addition, from 200 images, we ensure half of the images have finding, whilst another half of the images are normal. All images are in png format. We also have simplified them from tuberculosis categories into two categories which are normal chest x-ray and tuberculosis chest x-ray.

4.6.2 Model Building and Training

The aim of the model is to address the tuberculosis. For the model building and training, we have used Tensor flow library which is an open source library for high performance numerical computation. We also have used Anaconda virtual environment for training the model.

The dataset that we have selected is randomly split into 80% for training and 20% for testing. Wethen used the training dataset to train the model and testing dataset to assess the model. The loss values have recorded while the trained model progresses to understand the quality of the prediction.


jupyter Tuberculosis Last Checkpoint: 09/20/2021 (autosaved)
Edit View Insert Cell Kernel Widgets Help
Run
Code

```
            sensitivity: 0.9487
            specificity: 0.8780

In [ ]:     model.save('/kaggle/working/modelv7.h5')

In [44]:    model.summary()
            Model: "sequential_9"
            Layer (type)                   Output Shape              Param #
            ===================================================================
            conv2d_46 (Conv2D)             (None, 224, 224, 32)      896
            max_pooling2d_46 (MaxPooling (None, 112, 112, 32)      0
            conv2d_47 (Conv2D)             (None, 112, 112, 32)      9248
            max_pooling2d_47 (MaxPooling (None, 56, 56, 32)        0
            dropout_18 (Dropout)           (None, 56, 56, 32)        0
            conv2d_48 (Conv2D)             (None, 56, 56, 32)        9248
            max_pooling2d_48 (MaxPooling (None, 28, 28, 32)        0
            conv2d_49 (Conv2D)             (None, 28, 28, 64)        18496
            max_pooling2d_49 (MaxPooling (None, 14, 14, 64)        0
```

Fig 4.8 Data model Processing

4.6.3 Model Testing and Validation

Model testing and validation phase focuses on three aspects, namely, effectiveness, performance evaluation, and time comparison. Each of them is discussed as follows:

- The effectiveness of the model is done through live webcam. Validated pictureswill be marked with either yellow or green box with an accuracypercentage to show the confidence level of the prediction.
- For the model performance evaluation, the complete trained model isvalidated with another 100 randomly chest x-ray images taken from Chest X-ray dataset.

We then compute theconfusion matrix to obtain four key values.

$$Accuracy = \frac{(TP + TN)}{(TP + FP + TN + FN)} \cdots\cdots\cdots\cdots (1)$$

$$Recall = \frac{(TP)}{(TP + FN)} \cdots\cdots\cdots\cdots (2)$$

$$Precision = \frac{(TP)}{(TP + FP)} \cdots\cdots\cdots\cdots (3)$$

$$\mathrm{F1-score} = 2 \times \frac{(Precision \times Recall)}{(Precision + Recall)} \cdots\cdots\cdots\cdots (4)$$

Accuracy

Accuracy represents the number of correctly classified data instances over the totalnumber of data instances.

Precision

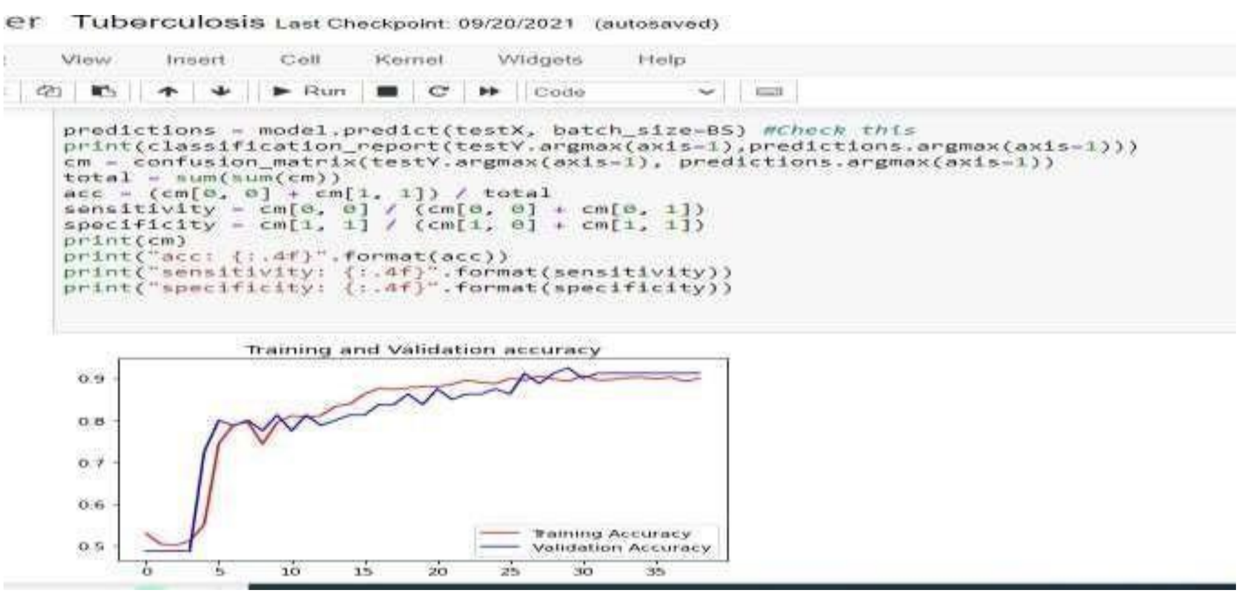
Precision should ideally be 1 (high) for a good classifier. **Precision** becomes 1 only when the numerator and denominator are equal i.e **TP = TP +FP**, this also means **FP** is zero. As **FP** increases the value of denominator becomes greater than the numerator and **precision** value decreases.

Recall

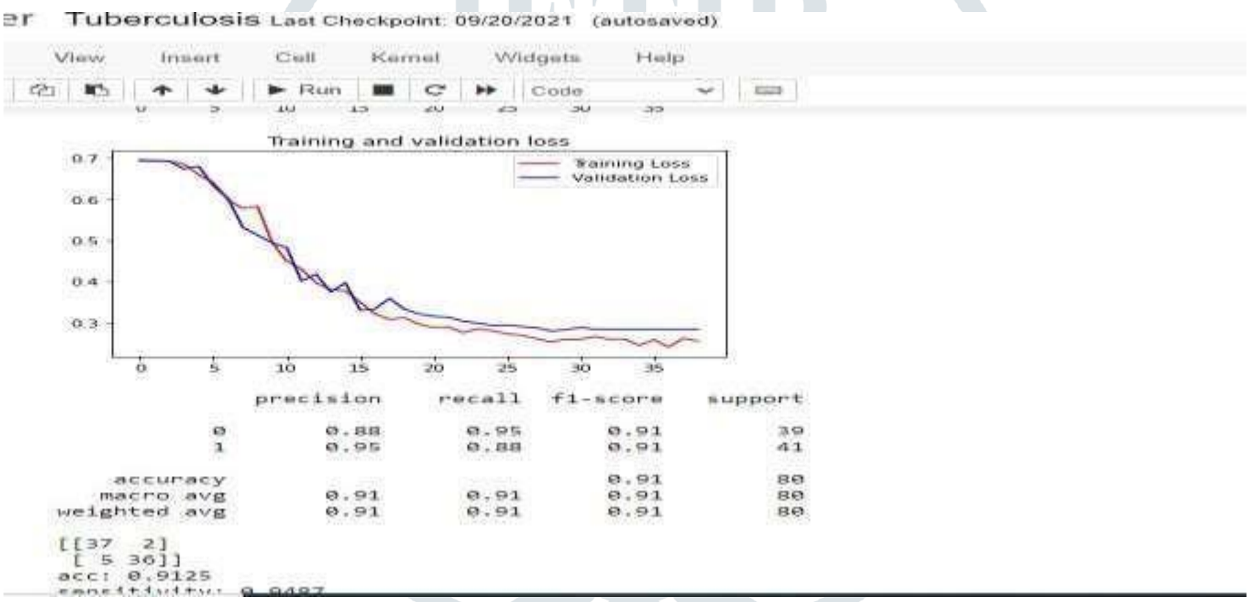
Recall should ideally be 1 (high) for a good classifier. **Recall** becomes 1 only when the numerator and denominator are equal i.e **TP = TP +FN**, this also means **FN** is zero. As **FN** increases the value of denominator becomes greater than the numerator and **recall** value decreases.

F1 Score

F1 Score becomes 1 only when **precision** and **recall** are both 1. **F1 score** becomeshigh only when both **precision** and **recall** are high. **F1 score** is the harmonic mean of **precision** and **recall** and is a better measure than **accuracy**.


er Tuberculosis Last Checkpoint: 09/20/2021 (autosaved)
View Insert Cell Kernel Widgets Help
Run
Code
Training and Validation accuracy
Training Accuracy
Validation Accuracy

```
predictions = model.predict(testX, batch_size=BS) #Check this
print(classification_report(testY.argmax(axis=1),predictions.argmax(axis=1)))
cm = confusion_matrix(testY.argmax(axis=1), predictions.argmax(axis=1))
total = sum(sum(cm))
acc = (cm[0, 0] + cm[1, 1]) / total
sensitivity = cm[0, 0] / (cm[0, 0] + cm[0, 1])
specificity = cm[1, 1] / (cm[1, 0] + cm[1, 1])
print(cm)
print("acc: {:.4f}".format(acc))
print("sensitivity: {:.4f}".format(sensitivity))
print("specificity: {:.4f}".format(specificity))
```


er Tuberculosis Last Checkpoint: 09/20/2021 (autosaved)
View Insert Cell Kernel Widgets Help
Run
Code
Training and validation loss
Training Loss
Validation Loss

```
              precision    recall  f1-score   support

           0       0.88      0.95      0.91        39
           1       0.95      0.88      0.91        41

    accuracy                           0.91        80
   macro avg       0.91      0.91      0.91        80
weighted avg       0.91      0.91      0.91        80

[[37  2]
 [ 5 36]]
acc: 0.9125
sensitivity: 0.9487
```

Fig 4.9 Data Evaluation and testing

CHAPTER 5 SYSTEM SPECIFICATION

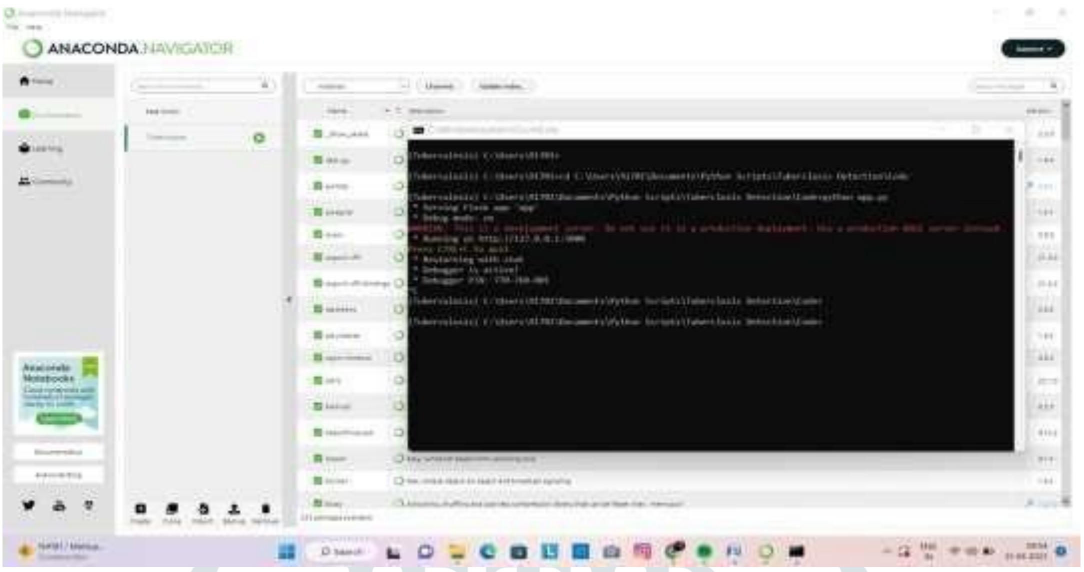
5.1 HARDWARE REQUIREMENTS

❖ Processor	: i3 PROCESSOR
❖ Hard Disk	: 500 GB.
❖ Monitor	: 15 VGA Color.
❖ Mouse	: Logitech.
❖ Ram	: 4 GB

5.2 SOFTWARE REQUIREMENTS

❖ Operating system	: Windows
❖ Coding Language	: Python
❖ Tool	: Anaconda IDE
❖ Database	: Kaggle

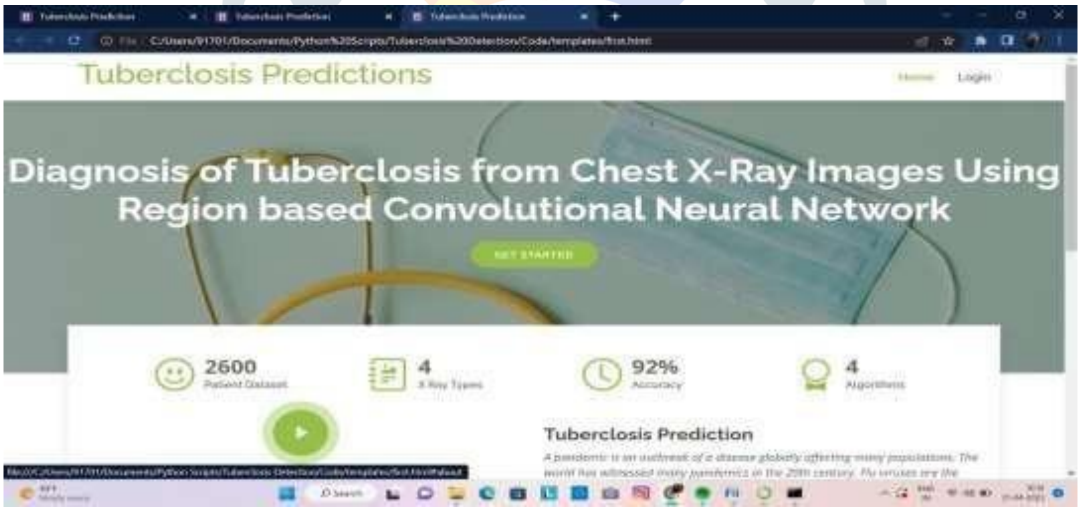
CHAPTER 6 RESULTS AND DISCUSSION



```
(tuberculosis) C:\Users\91701>
(tuberculosis) C:\Users\91701>cd C:\Users\91701\Documents\Python Scripts\Tuberclosis Detection\Code
(tuberculosis) C:\Users\91701\Documents\Python Scripts\Tuberclosis Detection\Code>python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: [illegible]
^C
(tuberculosis) C:\Users\91701\Documents\Python Scripts\Tuberclosis Detection\Code>
(tuberculosis) C:\Users\91701\Documents\Python Scripts\Tuberclosis Detection\Code>
```

Step 1: Open the Anaconda Navigator and run the terminal in Tuberculosis project. Then enter the location path of python scripts and run the terminal using PYTHON APP.PY.

Step 2: After that it create one IP address and then enter the IP address into one web browser.


Tuberclosis Prediction
C:/Users/91701/Documents/Python%20Scripts/Tuberclosis%20Detection/Code/templates/first.html
Home
Login
GET STARTED

Tuberclosis Predictions

Diagnosis of Tuberclosis from Chest X-Ray Images Using Region based Convolutional Neural Network

2600 Patient Dataset

4 X-Ray Types

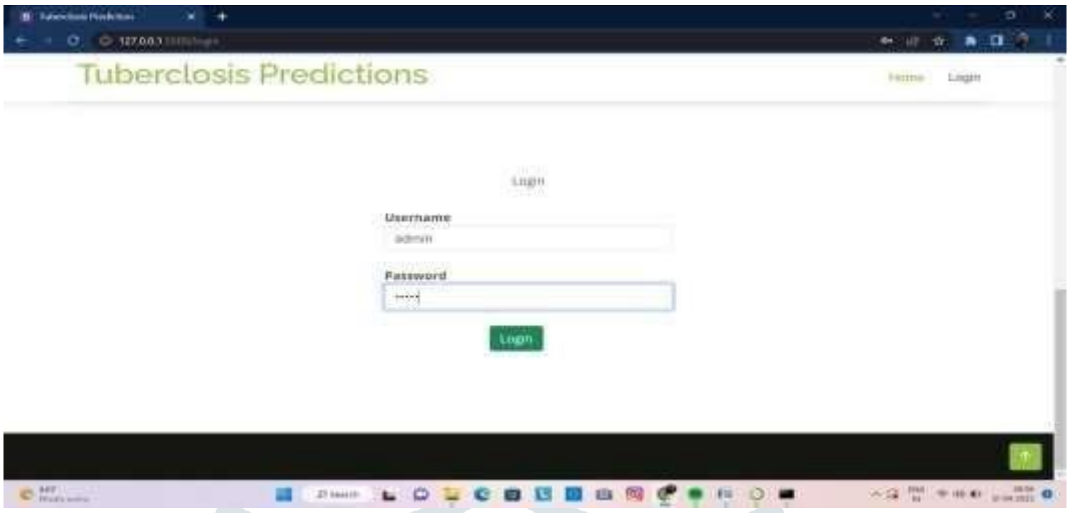
92% Accuracy

4 Algorithms

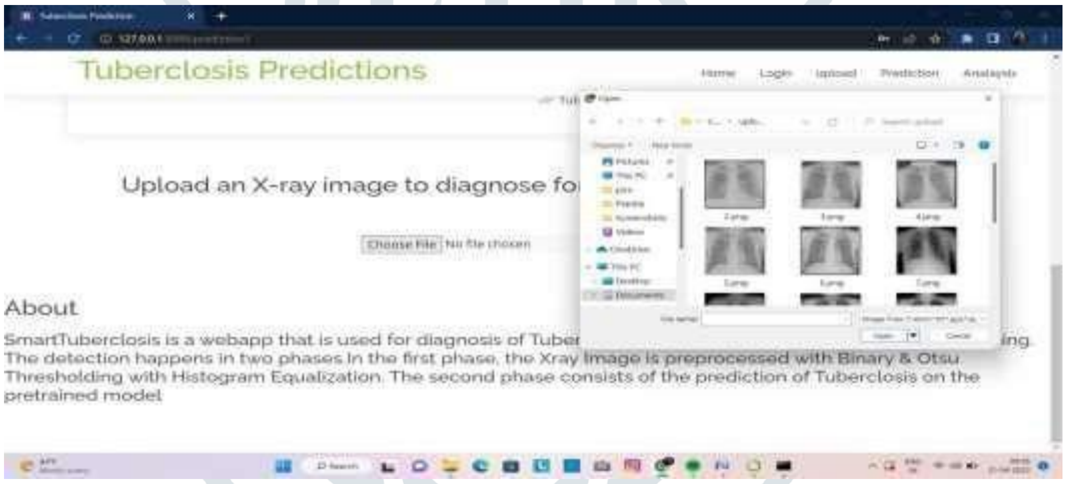
Tuberclosis Prediction

A pandemic is an outbreak of a disease globally affecting many populations. The world has witnessed many pandemics in the 20th century. Flu viruses are the

Step 3: Login the Web page as admin

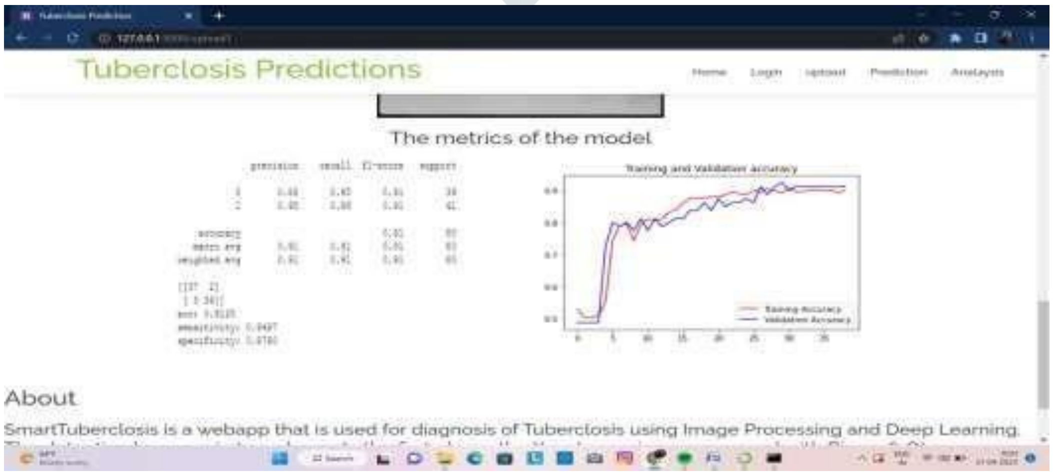


Step 4: Choose the X-Ray image to diagnosis

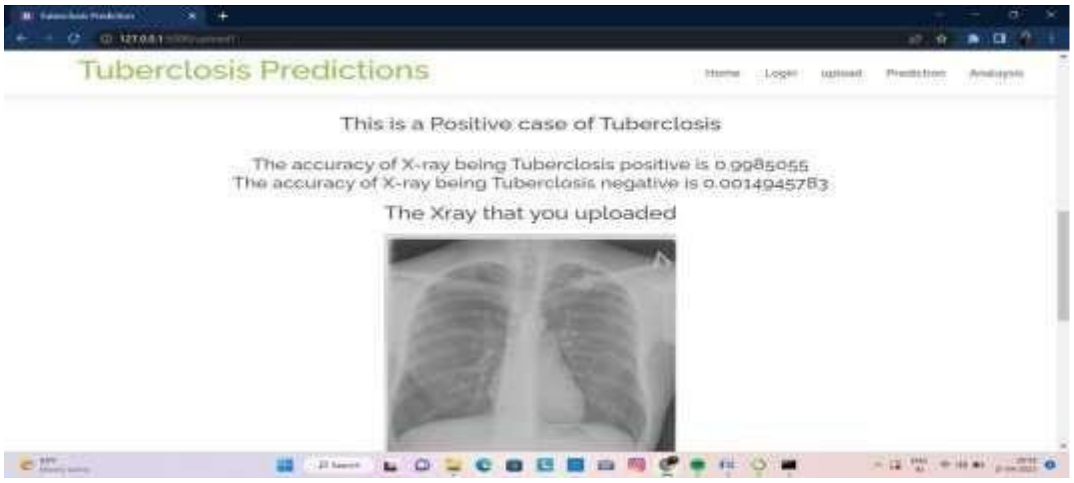

Tuberclosis Predictions
Home
Login
Upload
Prediction
Analaysis
Upload an X-ray image to diagnose fo
Choose File No file chosen
About

SmartTuberclosis is a webapp that is used for diagnosis of Tuber[illegible]ing. The detection happens in two phases In the first phase, the Xray image is preprocessed with Binary & Otsu Thresholding with Histogram Equalization. The second phase consists of the prediction of Tuberclosis on the pretrained model

Step 5: After choosing the X-ray image it will show the Positive and Negativepercentage of TB.



SmartTuberclosis is a webapp that is used for diagnosis of Tuberclosis using Image Processing and Deep Learning.

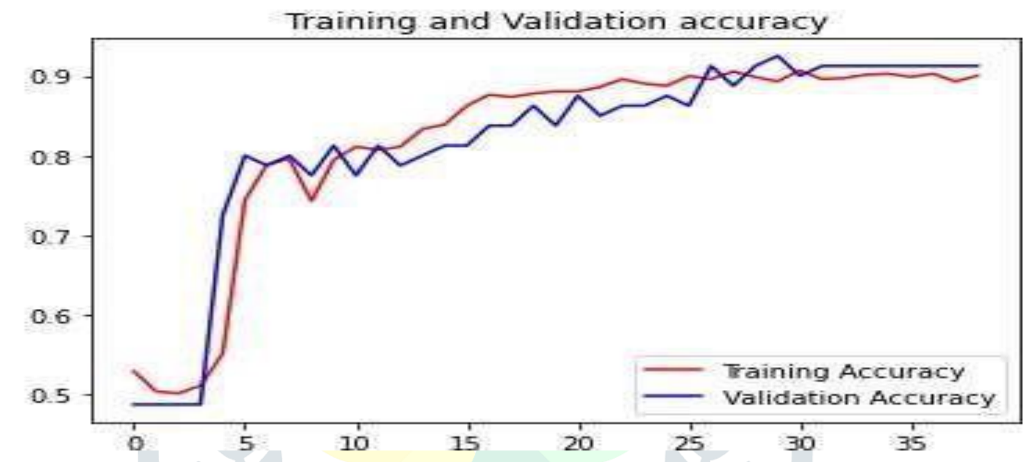

Tuberclosis Predictions

This is a Positive case of Tuberclosis

The accuracy of X-ray being Tuberclosis positive is 0.9985055
The accuracy of X-ray being Tuberclosis negative is 0.0014945783

The Xray that you uploaded

Step 6: The below image shows accuracy result



CHAPTER 7

CONCLUSION AND FUTURE ENHACEMENT

7.1 CONCLUSION

In conclusion, the testing of Region based Convolutional Neural Network showed that it gained a good capability of diagnosing the new unseen chest X-rays and correctly classifying them into normal or abnormal images. Thus, it can be stated that the RCNN can be a good classifier for the chest X-rays classification with a small margin of errors. Moreover, it is seen that the enhancement of chest X- rays using histogram equalization has a good role in improving the learning of the Region based Convolutional Neural Network, which results in a better accuracy during the testing of the network.

7.2 FUTURE ENHANCEMENT

We have used a subset of ChestX-ray dataset and partition them into 80% for training set and 20% for testing set. We have validated the model by comparing its performance with medical representatives. The results show a reasonable accuracy as compared to the medical representatives. There are a few aspects that can be tackled in the future to improve the model performance. Firstly, the training dataset can be increased. For this reason, an alternative to manual labelling is needed to label the region. Secondy, the model validation can be compared with more medical representatives that may include radiologists.

APPENDIX-I

SOURCE CODE

APP.py

```
import jsonimport plotly
import pandas as pdimport numpy as npfrom flask import Flask
from flask import render_template, requestfrom plotly.graph_objs import Barimport joblib
from sqlalchemy import create_engine from flask import Flask, render_template,
request
import os,cv2
from keras.models import Model,load_model
#from keras.preprocessing.imageimporting_to_array
from tensorflow.keras.utils import img_to_arrayimport matplotlib.pyplot as plt
import numpy as np import tensorflow as tf app = Flask(_____name__) APP_ROOT =
os.path.dirname(os.path.abspath(_____file__))
app.config['SEND_FILE_MAX_AGE_DEFAU
LT'] = 1
size=224
def processimg(img): img=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY) img =
cv2.resize(img, (size, size))
clahe = cv2.createCLAHE(clipLimit=2.0,tileGridSize=(8, 8))
equalized_img = clahe.apply(img) crop=cv2.resize(equalized_img,(size,size)) returncrop
@app.route('/') @app.route('/first')def first():
return render_template('first.html')@app.route('/login')def login():
return render_template('login.html')def home():
return render_template('home.html')@app.route('/upload')def upload():
```

```
return render_template('upload.html') @app.route('/preview',methods=["POST"])def
preview():
if request.method == 'POST':
dataset = request.files['datasetfile'] df = pd.read_csv(dataset,encoding ='unicode_escape')
df.set_index('Id', inplace=True)return render_template("preview.html",df_view =
df) @app.route('/prediction1')def index():
return render_template('index.html') @app.route("/upload1", methods=['POST']) def
upload1():
model=load_model('modelv7.h5')print("model_loaded") target =
os.path.join(APP_ROOT,'static/xray/')
if not os.path.isdir(target):os.mkdir(target)filename = ""
for file in request.files.getlist("file"):filename = file.filename
destination = "/".join([target, filename])file.save(destination)
img = cv2.imread(destination) cv2.imwrite('static/xray/file.png',img) img=
processimg(img)
v2.imwrite('static/xray/processedfile.png',img)img = img.astype('uint8')img =
cv2.cvtColor(img, cv2.COLOR_GRAY2BGR)
img = img_to_array(img)
img = cv2.resize(img,(size,size)) img=img.reshape(1,size,size,3) img =
img.astype('float32')
img = img / 255.0
#result = model.predict(img)result = np.argmax(model.predict(img),axis=1)
#result = np.round(result).astype(int)pred=model.predict(img) neg=pred[0][0]pos=pred[0][1]
classes=['Negative','Positive']predicted=classes[result[0]]
plot_dest = "/".join([target, "result.png"])
return render_template("result.html",pred=predicted,pos=pos,neg=neg,
filename=filename)@app.route('/chart') def chart():
return   render_template('chart.html')@app.route('/prediction')
def prediction():

return render_template("home.html")@app.route('/crime')def crime():

return render_template("crime.html")@app.route('/crimes') def crimes():

return render_template("crimes.html")@app.route('/total') def total():

return render_template("total.html")@app.route('/theft')
def theft():

return        render_template('theft.html')    @app.route('/predict',methods=['POST'])   def
predict():
df= pd.read_csv("some.csv",encoding="latin-1")# Features and Labelsdf['label'] = df['l']
X = df['t']
y =  df['label']print(X)
# Extract Feature With CountVectorizer

cv = CountVectorizer(ngram_range=(1, 2))X = cv.fit_transform(X) # Fit the Datafrom
sklearn.model_selection importtrain_test_split
X_train,       X_test,     y_train,     y_test      =     train_test_split(X,     y,     test_size=0.10,
random_state=0)
#Naive Bayes Classifier

from  sklearn.naive_bayes  importMultinomialNB
clf  =  svm.SVC(kernel='linear')clf.fit(X_train,y_train)   clf.score(X_test,y_test)
if request.method == 'POST': message = request.form['message']data = [message]vect =
cv.transform(data).toarray()my_prediction = clf.predict(vect)
return render_template('result.html',prediction
= my_prediction)
```

if_____name___== '__main__':app.run(debug=True)

Lable_image.py

from_____future___import absolute_import,division,print_functionimport argparseimport

sys import time

import numpy as np import tensorflow as tf
import tensorflow.compat.v1 as tftf.disable_v2_behavior()def load_graph(model_file):

graph = tf.Graph() graph_def = tf.GraphDef()

with open(model_file, "rb") as f: graph_def.ParseFromString(f.read()) with graph.as_default(): tf.import_graph_def(graph_def) return graph

def read_tensor_from_image_file(file_name,input_height=299, input_width=299,

input_mean=0,

input_std=255): input_name = "file_reader" output_name = "normalized"file_reader = tf.read_file(file_name,input_name)

if file_name.endswith(".png"):image_reader =

tf.image.decode_png(file_reader, channels = 3, name='png_reader')elif

file_name.endswith(".gif"):

image_reader = tf.squeeze(tf.image.decode_gif(file_reader,name='gif_reader'))

elif file_name.endswith(".bmp"):

image_reader = tf.image.decode_bmp(file_reader,name='bmp_reader')else:

image_reader = tf.image.decode_jpeg(file_reader, channels =3,name='jpeg_reader')

float_caster = tf.cast(image_reader, tf.float32) dims_expander = tf.expand_dims(float_caster, 0);

resized = tf.image.resize_bilinear(dims_expander,[input_height, input_width])normalized = tf.divide(tf.subtract(resized,[input_mean]), [input_std])

sess = tf.Session()

```
result = sess.run(normalized)return result
def  load_labels(label_file):

label = [] proto_as_ascii_lines = tf.gfile.GFile(label_file).readlines()

for l in proto_as_ascii_lines:label.append(l.rstrip()) return labeldef predict(file_name):

model_file = "retrained_graph.pb"  label_file = "retrained_labels.txt"  file_name =
file_name input_height = 224

input_width  =  224

input_mean = 128

input_std = 128 input_layer = "input"output_layer = "final_result"

parser = argparse.ArgumentParser() parser.add_argument("--image", help="imageto be
processed")

parser.add_argument("--graph", help="graph/model to be executed")
parser.add_argument("--labels", help="name offile containing labels")
parser.add_argument("--input_height",type=int, help="input height")
parser.add_argument("--input_width", type=int,help="input width")
parser.add_argument("--input_mean", type=int,help="input mean")
parser.add_argument("--input_std", type=int,help="input std") parser.add_argument("--
input_layer",help="name of input layer") parser.add_argument("--
output_layer",help="name of output layer") args = parser.parse_args()if args.graph:
model_file = args.graphif args.image:
file_name  =  args.imageif  args.labels:
```

```
label_file = args.labelsif args.input_height:

input_height = args.input_heightif args.input_width:

input_width = args.input_widthif args.input_mean:

input_mean = args.input_meanif args.input_std:

input_std = args.input_stdif args.input_layer:

input_layer = args.input_layerif args.output_layer:

output_layer = args.output_layergraph = load_graph(model_file)t =

read_tensor_from_image_file(file_name,

input_height=input_height,        input_width=input_width,      input_mean=input_mean,
input_std=input_std)
input_name = "import/" + input_layer output_name = "import/" + output_layerinput_operation
= graph.get_operation_by_name(input_name);

output_operation = graph.get_operation_by_name(output_name);with

tf.Session(graph=graph) as sess:

start = time.time()

results = sess.run(output_operation.outputs[0],

{input_operation.outputs[0]:   t})end=time.time()

results      =    np.squeeze(results)    top_k    =    results.argsort()[-1:][::-1]    labels    =
load_labels(label_file)
# print('\nEvaluation time (1-image):

{:.3f}s\n'.format(end-start))

template = "{} (score={:0.5f})"for i in top_k:
print(template.format(labels[i],        results[i]))    result    =    "{:0.5f}".format(results[i])
print(result)
```

```
if float(result) >= 0.8:

return labels[i]return 0
predict('vlcsnap-2018-08-26-08h40m33s232.png')
```

Predict.py

```
from flask import Flask, render_template,requestimport os, cv2
from keras.models import Model,load_modelfrom keras.preprocessing.image import
img_to_array

import matplotlib.pyplot as pltimport numpy as npimport tensorflow as tf

app = Flask(_____name__)APP_ROOT = os.path.dirname(os.path.abspath(__file___))

app.config['SEND_FILE_MAX_AGE_DEFAULT'] = 1size=224

def processimg(img): img=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY) img =
cv2.resize(img, (size, size))

clahe = cv2.createCLAHE(clipLimit=2.0,tileGridSize=(8, 8))
equalized_img = clahe.apply(img) crop=cv2.resize(equalized_img,(size,size)) returncrop
@app.route("/")def index():
return render_template("index.html") @app.route("/upload", methods=['POST']) def
upload():
model=load_model('modelv7.h5') print("model_loaded")
```

target = os.path.join(APP_ROOT,'static/xray/')

if not os.path.isdir(target):os.mkdir(target)filename = ""

for file in request.files.getlist("file"):filename = file.filenamedestination = "/".join([target, filename])file.save(destination)

img = cv2.imread(destination) cv2.imwrite('static/xray/file.png',img) img= processimg(img)

v2.imwrite('static/xray/processedfile.png',img)img = img.astype('uint8')img =

cv2.cvtColor(img, cv2.COLOR_GRAY2BGR)

img = img_to_array(img)

img = cv2.resize(img,(size,size)) img=img.reshape(1,size,size,3) img = img.astype('float32')

img = img / 255.0

result = model.predict_classes(img)pred=model.predict(img) neg=pred[0][0]

pos=pred[0][1] classes=['Negative','Positive']predicted=classes[result[0]]plot_dest = "/".join([target, "result.png"])

return render_template("result.html", pred=predicted,pos=pos,neg=neg, filename=filename)

if_____name___== '__main__':

app.run(debug=True)

Trained Data Model

```
{

"cells": [

{
"cell_type": "code", "execution_count": null,"metadata": {

"_cell_guid": "b1076dfc-b9ad-4769-8c92-a6c4dae69d19", "_uuid":
"8f2839f25d086af736a60e9eeb907d3b93b6e0e5"

},

"outputs": [],"source": [

"import numpy as np # linear algebra\n",

"import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)\n", "importos\n",

"for dirname, _, filenames in os.walk('/kaggle/input'):\n"," for filename in
filenames:\n",

"print(os.path.join(dirname, filename))\n"

]

},

{

"cell_type": "code", "execution_count": 11,"metadata": {}, "outputs": [

{

"name": "stdout", "output_type": "stream","text": [

"Requirement already satisfied: imutils in /opt/conda/lib/python3.7/site- packages
(0.5.3)\n",

"\u001b[33mWARNING: You are using pip version 20.1; however, version

20.1.1 is available.\n",

"You should consider upgrading via the '/opt/conda/bin/python3.7 -m pip install --
upgrade pip' command.\u001b[0m\n"

]
```

}

],

"source": [

"!pip install imutils"

]

},

{

"cell_type": "code", "execution_count": 12,"metadata": {

"_cell_guid": "79c7e3d0-c299-4dcb-8224-4455121ee9b0", "_uuid": "d629ff2d2480ee46fbb7e2d37f6b5fab8052498a"

},

"outputs": [],"source": [

"%matplotlib inline \n","import warnings\n", "import numpy as np\n","import pandasas pd\n",

"import matplotlib.pyplot as plt\n","from matplotlib import style\n", "import seabornas sns\n",

"import tensorflow as tf\n", "import cv2 \n", "import numpy as np \n", "from tqdm import tqdm\n","import os \n",

"from random import shuffle \n","import argparse\n", "import random\n", "import matplotlib\n", "matplotlib.use(\"Agg\")\n", "from imutils import paths\n",

"from sklearn.model_selection import train_test_split\n", "from sklearn.model_selection import KFold\n",

"from sklearn.metrics import accuracy_score, precision_score, recall_score, confusion_matrix, roc_curve,roc_auc_score\n", "from sklearn.model_selection import GridSearchCV\n",

"from sklearn.preprocessing import LabelEncoder\n", "from keras.preprocessing.image import

ImageDataGenerator,img_to_array\n", "from keras import backend as K\n", "from keras.models import Sequential\n","from keras.layers import Dense\n",

"from keras.optimizers import Adam,SGD,RMSprop\n", "from keras.utils import to_categorical\n",

"from keras.callbacks import ModelCheckpoint, EarlyStopping, TensorBoard, CSVLogger, ReduceLROnPlateau, LearningRateScheduler\n",

"from keras.layers import Dropout, Flatten,Activation, Dense\n", "from keras.layersimport Conv2D, MaxPooling2D,

BatchNormalization,AveragePooling2D,ZeroPadding2D\n",

"from sklearn.metrics import classification_report,confusion_matrix\n", "from keras.callbacks import EarlyStopping, ModelCheckpoint\n", "\n",

"\n", "checkpoint =

ModelCheckpoint('./base.model',monitor='val_loss',verbose=1,save_best_only= True,mode='min',\n",

"save_weights_only=False,period=1)\n","earlystop =

EarlyStopping(monitor='val_loss',min_delta=0.001,patience=10,verbose=1,mod e='auto')\n",

"reduce = ReduceLROnPlateau(monitor='val_loss',factor=0.1, patience=3, verbose=1, mode='auto')\n",

"callbacks = [earlystop,checkpoint,reduce]"

]

},

{
"cell_type": "code", "execution_count": 13,"metadata": {}, "outputs": [

{

"name": "stdout", "output_type": "stream","text": ["800\n"

]

}

],

"source": ["imagepaths=sorted(list(paths.list_images(r\"/kaggle/input/tbclahedataset/lung_clahe\")))\n",

"print(len(imagepaths))\n","\n"
]

},

{

"cell_type": "code", "execution_count": 14,"metadata": {}, "outputs": [

{

"name": "stdout", "output_type": "stream","text": [

"Processing images' NPY file\n","Processing Done\n"

]

}

],

"source": ["img_size=224\n", "num_classes=2\n", "print(\"Processing images' NPY file\")\n","data = []\n", "labels = []\n", "random.shuffle(imagepaths)\n", "# loop over the input images\n","for imagePath in (imagepaths):\n",

"   # load the image, pre-process it, and store it in the data list\n","   image = cv2.imread(imagePath)\n",

"   image = cv2.resize(image, (img_size, img_size))\n"," image = img_to_array(image)\n",

"   data.append(image)\n",

"   if imagePath.split(\".png\")[0][-1] == \"1\":\n","   label = 1\n","

        else:\n",

"   label= 0\n",

"   labels.append(label)\n", "# labels\n", "print(\"Processing Done\")"

]

},

{

"cell_type": "code", "execution_count": 15,"metadata": {},"outputs": [],"source": [

"data = np.array(data, dtype=\"float\") / 255.0\n","labels = np.array(labels)\n","\n",

"(trainX, testX, trainY, testY) = train_test_split(data,labels, test_size=0.1, random_state=40)\n",

"\n",

"# convert the labels from integers to vectors\n", "trainY = to_categorical(trainY, num_classes)\n", "testY = to_categorical(testY, num_classes)\n", "channels=(trainX.shape[3])"

]

},

{

"cell_type": "code", "execution_count": 41,"metadata": {},"outputs": [],"source": [

"model = Sequential()\n",

"model.add(Conv2D(filters = 32, kernel_size = (3,3),padding = 'Same',activation ='relu', \n",

"input_shape = (img_size,img_size,channels)))\n",

"model.add(MaxPooling2D(pool_size=(2,2)))\n","\n",

"model.add(Conv2D(filters = 32, kernel_size = (3,3),padding = 'Same',activation ='relu'))\n",

"model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))\n", "model.add(Dropout(0.4))\n",

"\n",

"model.add(Conv2D(filters = 32, kernel_size = (3,3),padding = 'Same',activation ='relu'))\n",

"model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))\n","\n",

"model.add(Conv2D(filters = 64, kernel_size = (3,3),padding = 'Same',activation ='relu'))\n",

"model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))\n", "model.add(Dropout(0.3))\n",

"\n",

"model.add(Conv2D(filters = 64, kernel_size = (3,3),padding = 'Same',activation ='relu'))\n",

"model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))\n","\n", "model.add(Conv2D(filters =128, kernel_size = (3,3),padding = 'Same',activation ='relu'))\n",

"model.add(MaxPooling2D(pool_size=(2,2),strides=(2,2)))\n", "model.add(Dropout(0.4))\n",

"\n",

"model.add(Conv2D(filters =128, kernel_size = (3,3),padding = 'Same',activation ='relu'))\n",

"model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))\n", "\n","model.add(Flatten())\n",

"model.add(Dense(256))\n","model.add(Activation('relu'))\n", "model.add(Dropout(0.5))\n", "model.add(Dense(num_classes, activation = \"softmax\"))\n"

]

},

{

"cell_type": "code", "execution_count": 42,"metadata": {},

APPENDIX-II REFERENCES

[1] K. R. Steingart, M. Henry, V. Ng, P. C. Hopewell, A. Ramsay, J. Cunningham, R. Urbanczik, M. Perkins, M. A. Aziz, and M. Pai, ‘‘Fluorescence versus conventional sputum smear microscopy for tuberculosis: A systematic review,’’ Lancet Infectious Diseases, vol. 6, no. 9, pp. 570–581, Sep. 2006.

[2] Chest Radiography in Tuberculosis Detection: Summary of Current WHO Recommendations and Guidance on Programmatic Approaches, World Health Org., Geneva, Switzerland, 2016.

[3] Q. Hou, D. Zhou, and J. Feng, ‘‘Coordinate attention for efficient mobile network design,’’ in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2021, pp. 13713–13722.

[4] K. Simonyan and A. Zisserman, ‘‘Very deep convolutional networks for large- scale image recognition,’’ 2014, arXiv:1409.1556.

[5] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, ‘‘Residual attention network for image classification,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 3156–3164.

[6] Q. Wang, B. Wu, P. Zhu, P. Li, W. Zuo, and Q. Hu, ‘‘ECA-Net: Efficient channel attention for deep convolutional neural networks,’’ in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 1– 12.

[7] S. Woo, J. Park, J. Y. Lee, and I. S. Kweon, ‘‘CBAM: Convolutional block attention module,’’ in Proc. Eur. Conf. Comput. Vis. (ECCV), 2018, pp. 3– 19.

[8] E. Showkatian, M. Salehi, H. Ghaffari, R. Reiazi, and N. Sadighi, ‘‘Deep learning-based automatic detection of tuberculosis disease in chest X-ray images,’’ Polish J. Radiol., vol. 87, no. 1, pp. 118–124, Feb. 2022.

[9] Q. Zhang, C. Bai, Z. Liu, L. T. Yang, H. Yu, J. Zhao, and H. Yuan, ‘‘A GPU- based residual network for medical image classification in smart medicine,’’ Inf.Sci., vol. 536, pp. 91–100, Oct. 2020.

[10] Y. Xie, Z. Wu, X. Han, H. Wang, Y. Wu, L. Cui, J. Feng, Z. Zhu, and Z. Chen, ‘‘Computer-aided system for the detection of multicategory pulmonary tuberculosis in radiographs,’’ J. Healthcare Eng., vol. 2020, pp. 1–12, Aug. 2020.

[11] Z. Ul Abideen, M. Ghafoor, K. Munir, M. Saqib, A. Ullah, T. Zia, S. A. Tariq, G. Ahmed, and A. Zahra, ‘‘Uncertainty assisted robust tuberculosis identification with Bayesian convolutional neural networks,’’ IEEE Access, vol. 8, pp. 22812–22825, 2020.

[12] K. Munadi, K. Muchtar, N. Maulina, and B. Pradhan, ‘‘Image enhancement for tuberculosis detection using deep learning,’’ IEEE Access, vol. 8, pp. 217897– 217907, 2020.

[13] S. Jaeger, A. Karargyris, S. Candemir, L. Folio, J. Siegelman, F. Callaghan, Z. Xue, K. Palaniappan, R. K. Singh, S. Antani, G. Thoma, Y.-X. Wang, P.- X.Lu, and C. J. McDonald, ‘‘Automatic tuberculosis screening using chest radiographs,’’ IEEE Trans. Med. Imag., vol. 33, no. 2, pp. 233–245, Feb. 2013.

[14] S. Jaeger, S. Candemir, S. Antani, Y. X. Wang, P. X. Lu, and G. Thoma, ‘‘Two public chest X-ray datasets for computer-aided screening of pulmonary diseases,’’ Quant. Imag. Med. Surg., vol. 4, p. 475, Dec. 2014.

[15] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and
N. Houlsby, ‘‘An image is worth 16 × 16 words: Transformers for imagerecognition at scale,’’ 2020, arXiv:2010.11929.

[16] K. He, X. Zhang, S. Ren, and J. Sun, ‘‘Deep residual learning for image recognition,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 770–778.

[17] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen,''MobileNetV2: Inverted residuals and linear bottlenecks,'' in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 4510–4520.

[18] D. L. Streiner and J. Cairney, ''What's under the ROC? An introduction to receiver operating characteristics curves,'' Can. J. Psychiatry, vol. 52, no. 2, pp. 121–128, Feb. 2007.

[19] Q. Wang, B. Wu, P. Zhu, P. Li, W. Zuo, and Q. Hu, ''ECA-Net: Efficient channel attention for deep convolutional neural networks,'' in Proc.IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 1– 12.

[20] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, ''Residual attention network for image classification,'' in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 3156–3164.