2019-20

Biological Database (BIT2002)

Project Report

Tuberculosis Diagnosis using Artificial Intelligence on X Ray Images

SIDDHARTH GARG   (18BCB0038)
AMRIT MAHAJAN (18BCB0073)

- SUBMITTED TO – DR SUDANDIRA DOSS

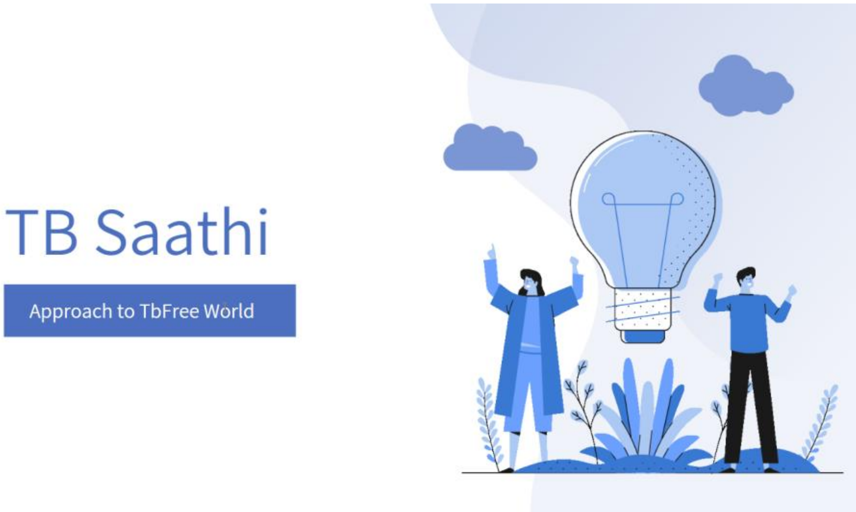
Slot- G1 + TG1

Table of Contents

Abstract

Tuberculosis (TB) is an infectious disease usually caused by Mycobacterium tuberculosis (MTB) bacteria. Tuberculosis generally affects the lungs, but can also affect other parts of the body. Most infections show no symptoms, in which case it is known as latent tuberculosis. The bacteria that cause TB are spread when an infected person coughs or sneezes. Most people infected with the bacteria that cause tuberculosis don't have symptoms. Treatment isn't always required for those without symptoms. Patients with active symptoms will require a long course of treatment involving multiple antibiotics.

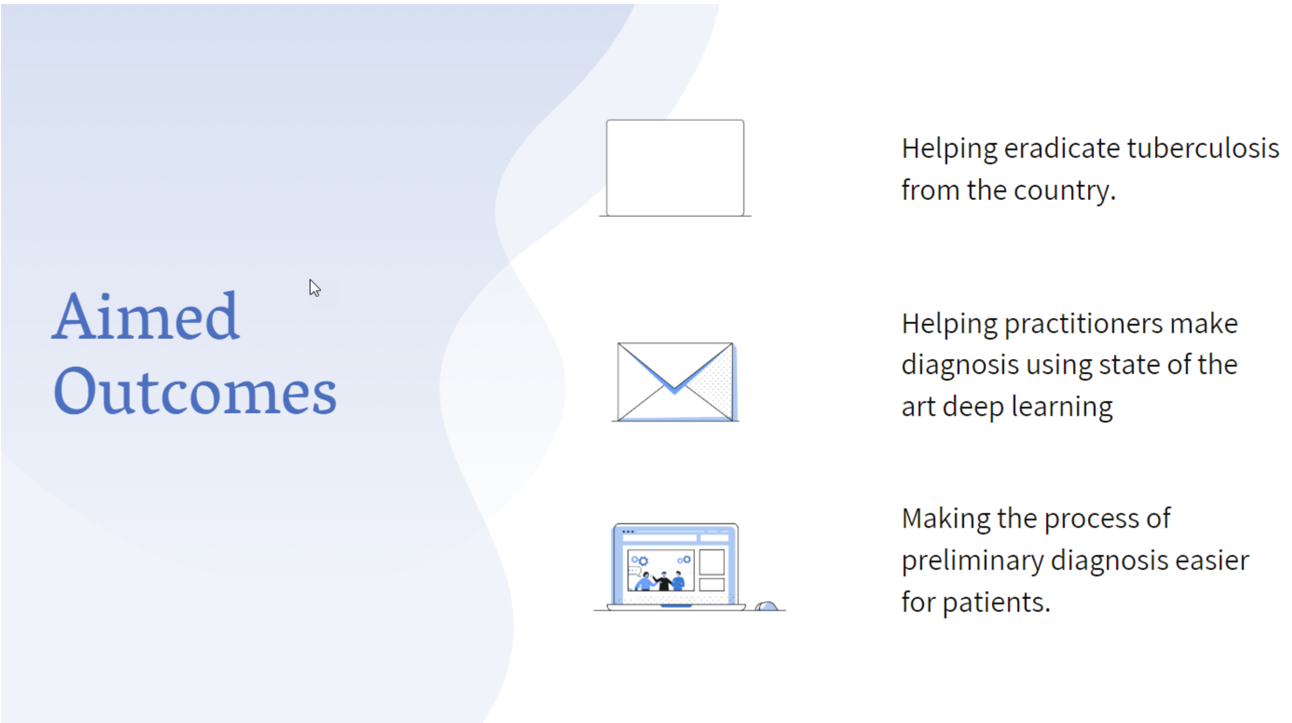
- Tuberculosis is curable and preventable.
- Tuberculosis (TB) is caused by bacteria (Mycobacterium tuberculosis) that most often affect the lungs
- In India each year 220,000 deaths are reported due to TB. 2.2 million cases of TB for India out of a global incidence of 9.6 million cases.



Project: TB Saathi – Approach to a TB Free World

<u>Aim</u>

Using state of the art technology Artificial Intelligence providing an easy , trustable and affordable tuberculosis Diagnosis.



Novelty

1. Integrated application

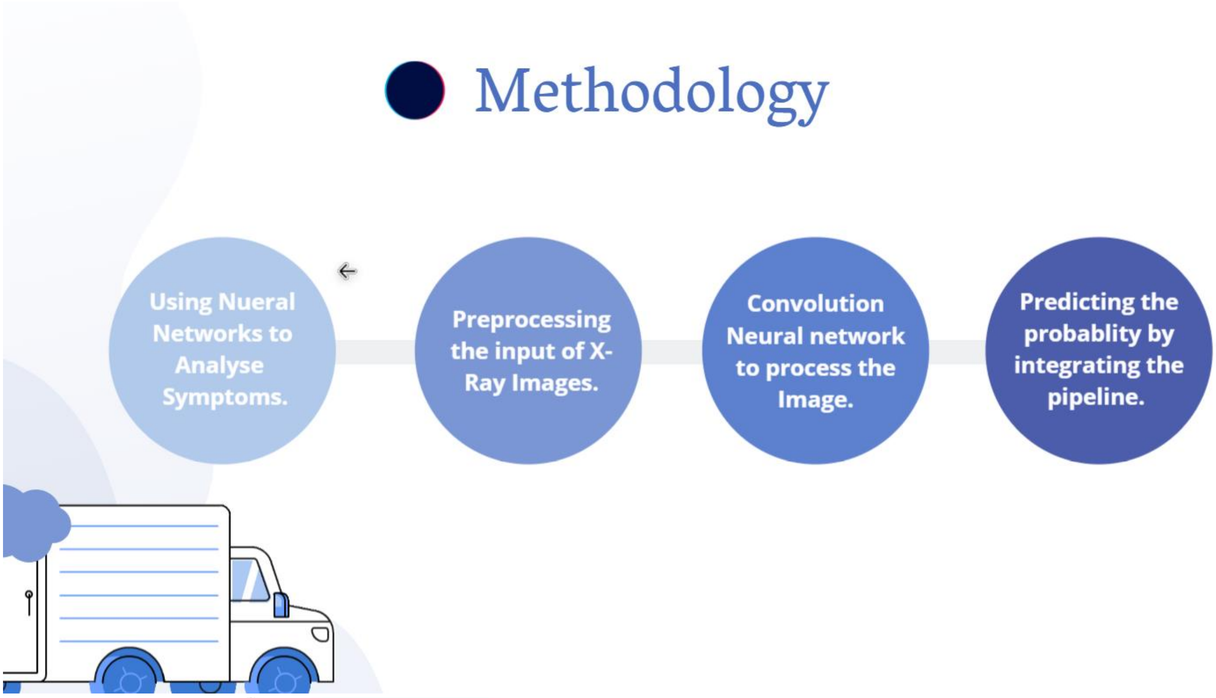
A single tool to take symptoms and X ray image as an input , process and predict the probability of tuberculosis using advanced deep neural network architectures and report to nearest health centres to initiate treatment

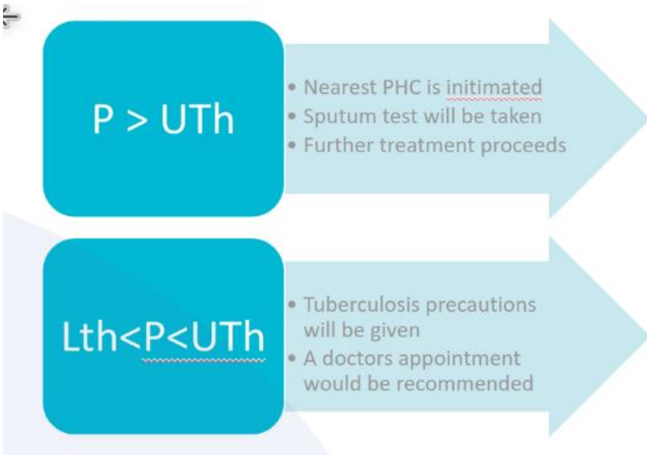
2. Communication to Government Healthcare

As soon as a new patient's record is added the nearest PHC is intimated. Rural areas would be highly benefited as non availability of PHCs in proximity causes them to look for informal practitioners..

Our Methodology

1. Using Nueral Networks to Analyse Symptoms.
2. Pre-processing the input of X-Ray Images.
3. Convolution Neural network to process the Image.
4. Predicting the probability by integrating the pipeline.
5. Setting Thresholds and finding severity of the case





Dataset

The standard digital image database for Tuberculosis is created by the National Library of Medicine, Maryland, USA in collaboration with Shenzhen No.3 People's Hospital, Guangdong Medical College, Shenzhen, China. The Chest X-rays are from out-patient clinics, and were captured as part of the daily routine using Philips DR Digital Diagnose systems. Number of X-rays:

336 cases with manifestation of tuberculosis, and 326 normal cases. Image parameters: Format: PNG Image size varies for each X-ray. It is approximately 3K x 3K. Problem Statement: Using Deep Learning detect Tuberculosis of a patient by analysing his X-Ray report.

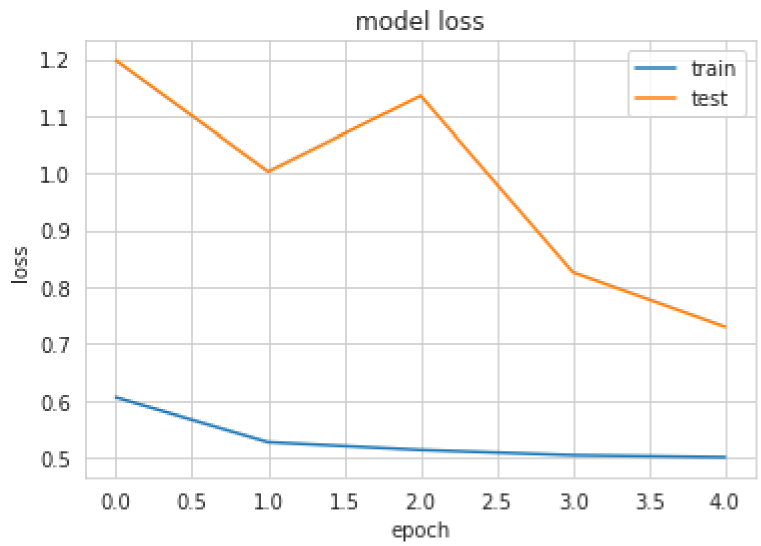
Download link:
http://openi.nlm.nih.gov/imgs/collections/ChinaSet_AllFiles.zi
p

Deep Learning problem

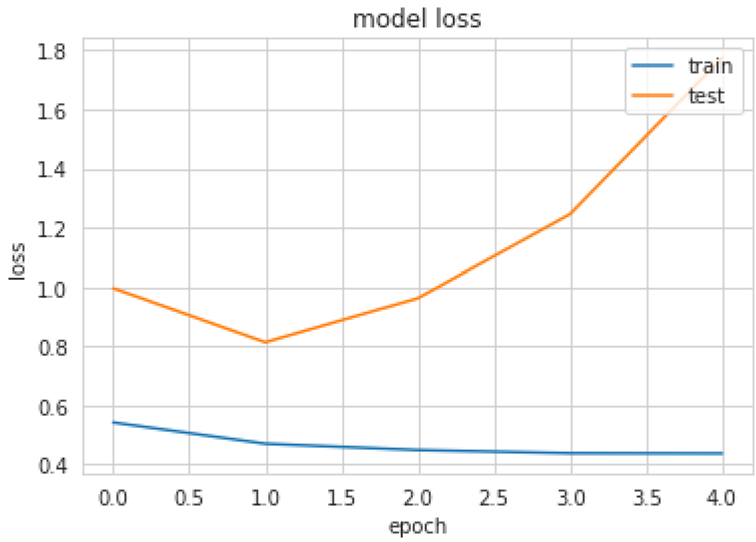
1. Using VGG-19, ResNet50 and Inception-V3 networks build models.
2. Use accuracy to compare performance of the three models.
3. Write a function to tune the threshold of the best model such that the AUC of the model is above x%(let default be 90%) and plot the confusion matrix with that tuned threshold.

Comparison of different model performances

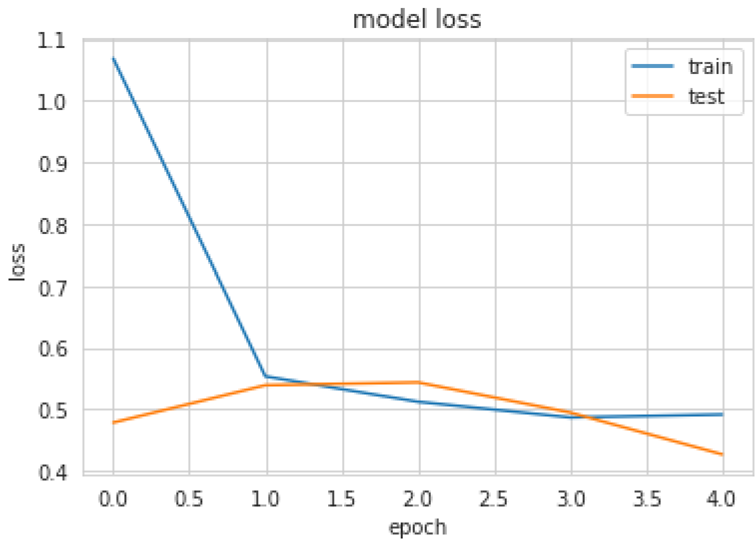
Inception



ResNet 50



VGG 19



Summary of pre existing models

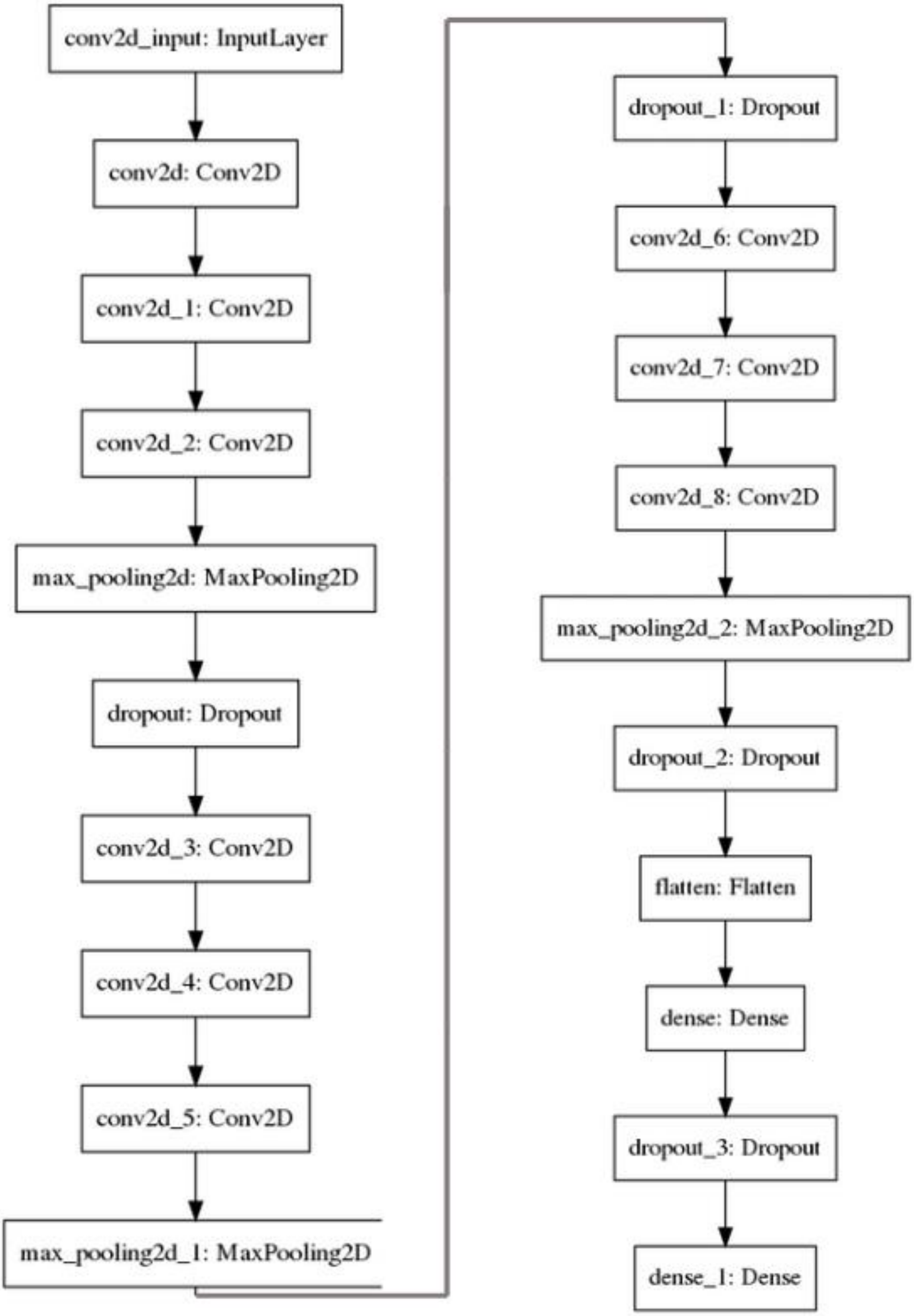
```
print(x)
```

```
+--------------+-----------+---------------+
|    Model     | Test loss | Test Accuracy |
+--------------+-----------+---------------+
| Inception V3 |    0.56   |      0.78     |
|  Resnet 50   |    2.04   |      0.43     |
|    VGG 19    |    0.42   |      0.78     |
+--------------+-----------+---------------+
```

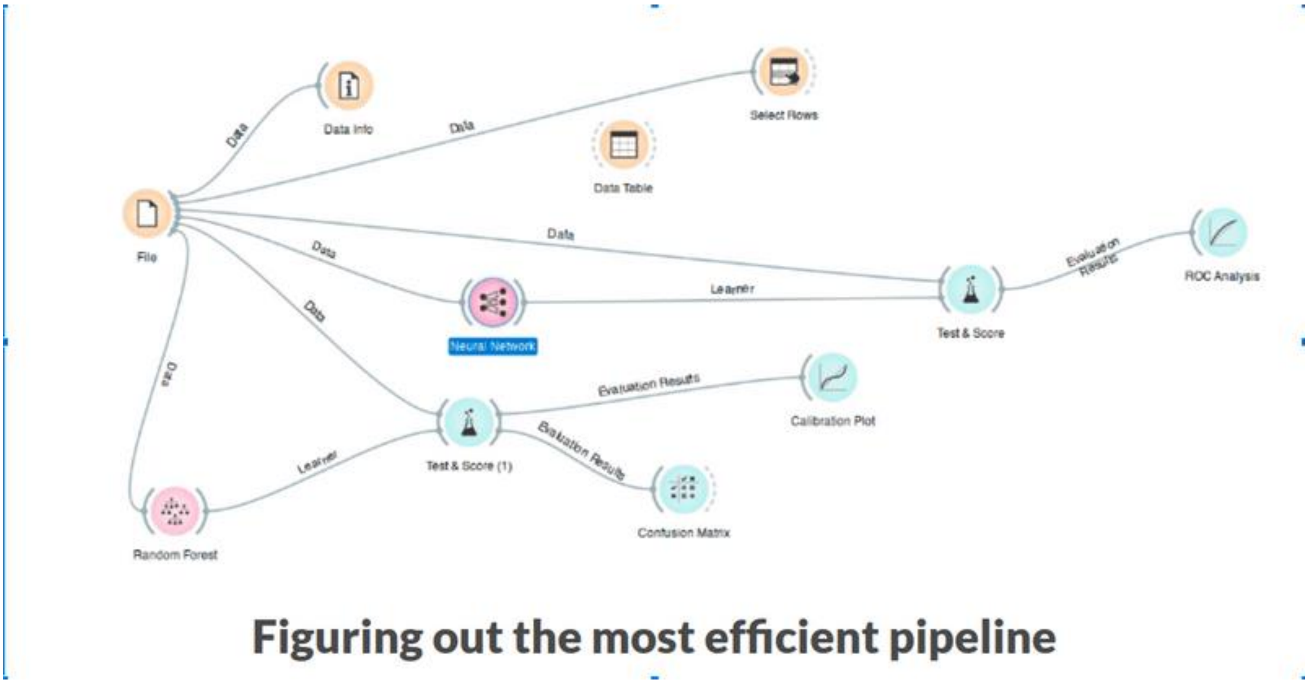
Our custom Model

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 94, 94, 32)        896
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 92, 92, 32)        9248
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 90, 90, 32)        9248
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 45, 45, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 45, 45, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 43, 43, 64)        18496
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 41, 41, 64)        36928
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 39, 39, 64)        36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 19, 19, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 19, 19, 64)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 17, 17, 128)       73856
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 15, 15, 128)       147584
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 13, 13, 128)       147584
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 128)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 6, 6, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 4608)              0
_________________________________________________________________
dense (Dense)                (None, 256)               1179904
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 514
=================================================================
Total params: 1,661,186
Trainable params: 1,661,186
Non-trainable params: 0
_________________________________________________________________
```

Convolution Artificial Neural Network Architecture Used.



Exploring Pipelines

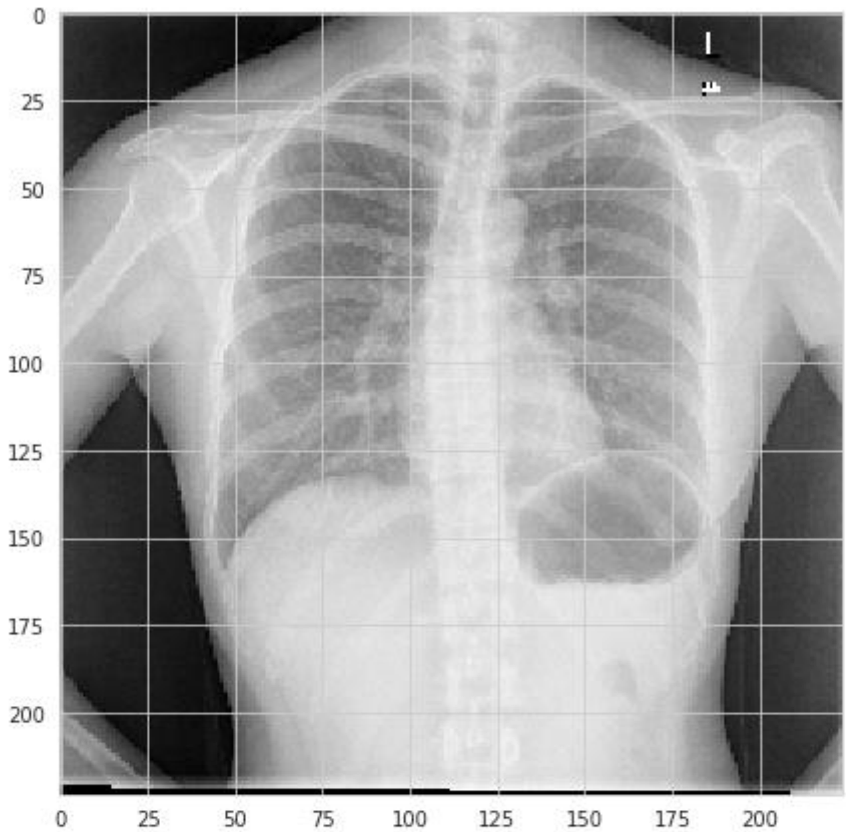


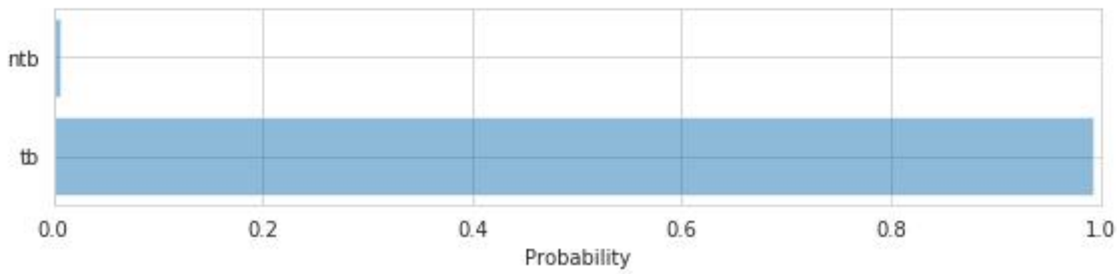
Result (Test Case)

In [0]:
```
# visualizing a test case
img = image.load_img('test/tb/CHNCXR_0328_1.png', target_size=(HEIGHT, WIDTH))
preds = predict(model, img)

plot_preds(np.asarray(img), preds)
preds
```

Out[14]:
```
array([0.9937085 , 0.00629148], dtype=float32)
```





Output Tuberculosis Positive

(99.37 % confidence)

Observations

1. Among all the models used VGG19 gave the a test accuracy of 78%. We got an validation accuracy of nearly 80% in that case.

2. VGG19 model also had lowest test loss of 0.42.

3. By taking a treshold of 0.8 we got a sensitivity of 0.73.

4. We have seen that our model work really good as confusion matrix had higher true positives.

5. We have got specificity as nan because the class levels we considered are only 1, so in that case our 'fpr' rate turned out to be nan after calculation.

6. We also visualised that our prediction for the x-ray image was remarkable with test accuracy of >90%

Codebase

```
#!/usr/bin/env python
# coding: utf-8

# # Chest X-RAY Tuberculosis Analysis using Deep Learning

# **Description:** The standard digital image database for Tuberculosis is created b
y the National Library of Medicine, Maryland, USA in collaboration with Shenzhen No.
3 People’s Hospital, Guangdong Medical College, Shenzhen, China. The Chest X-
rays are from out-
patient clinics, and were captured as part of the daily routine using Philips DR Dig
ital Diagnose systems.
# Number of X-rays:
# - 336 cases with manifestation of tuberculosis, and
# - 326 normal cases.
# Image parameters:
# - Format: PNG
# - Image size varies for each X-ray. It is approximately 3K x 3K.
#

# **Problem Statement:** Using Deep Learning detect Tuberculosis of a patient by ana
lysing his X-Ray report.

# ### Deep Learning problem

# 1. Using VGG-19, ResNet50 and Inception-V3 networks build models.
# 2. Use accuracy to compare performance of the three models.
# 3. Write a function to tune the threshold of the best model such that the AUC of t
he model is above x%(let default be 90%) and plot the confusion matrix with that tun
ed threshold.

# In[1]:


# initiating tensorflow and check GPU connection
import tensorflow as tf
tf.test.gpu_device_name()


# In[2]:


#Import modules
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Activation, Dense, Dropout
from keras import optimizers
from sklearn.metrics import confusion_matrix,accuracy_score, roc_curve, auc
get_ipython().run_line_magic('matplotlib', 'inline')
sns.set_style("whitegrid")


# In[ ]:


# Download the dataset to colab directory
%%capture
get_ipython().system('wget  http://openi.nlm.nih.gov/imgs/collections/ChinaSet_AllFi
les.zip')


# In[4]:


# Unzip the zipped folder and store data in Train folder
get_ipython().system('unzip ChinaSet_AllFiles.zip')
get_ipython().system('mv ChinaSet_AllFiles train')
get_ipython().system('find train/CXR_png -size  0 -print0 |xargs -0 rm --')


# In[ ]:


# creating two seperate data, one for patients with TB other with Non TB.
from glob import glob
ntb = glob('train/CXR_png/*_0.png')
tb = glob('train/CXR_png/*_1.png')


# In[ ]:


# create a directory called train_main and store data with TB and NonTB seperately i
n two folders.
get_ipython().system('mkdir train_main')
get_ipython().system('mkdir train_main/ntb')
files = ' '.join(ntb)
get_ipython().system('mv -t train_main/ntb $files')


# In[ ]:
```

```
get_ipython().system('mkdir train_main/tb')
files = ' '.join(tb)
get_ipython().system('mv -t train_main/tb $files')


# In[ ]:


ntb_1 = glob('train_main/ntb/*.png')
tb_1 = glob('train_main/tb/*.png')


# In[ ]:


# splitting data into train and test with 10% test data
from sklearn.model_selection import train_test_split
ntb_train, ntb_test = train_test_split(ntb_1, test_size=0.10)
tb_train, tb_test = train_test_split(tb_1, test_size=0.10)


# In[ ]:


# create new directories for test data
get_ipython().system('mkdir test')
get_ipython().system('mkdir test/ntb')
files = ' '.join(ntb_test)
get_ipython().system('mv -t test/ntb $files')

get_ipython().system('mkdir test/tb')
files = ' '.join(tb_test)
get_ipython().system('mv -t test/tb $files')


# In[ ]:


# defining test and train models in constants
TRAIN_DIR = 'train_main'
TEST_DIR = 'test'


# ### Inception V3:

# In[12]:


from keras.models import Model
from keras.layers import Dense, GlobalAveragePooling2D, Dropout
```

```
from keras.applications.inception_v3 import InceptionV3, preprocess_input

CLASSES = 2

# setup model
base_model = InceptionV3(weights='imagenet', include_top=False)

x = base_model.output
x = GlobalAveragePooling2D(name='avg_pool')(x)
x = Dropout(0.4)(x)
predictions = Dense(CLASSES, activation='softmax')(x)
model = Model(inputs=base_model.input, outputs=predictions)

# transfer learning
for layer in base_model.layers:
    layer.trainable = False

model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['accuracy'])


# ** Image data augmentation is used to expand the training dataset in order to impr
ove the performance and ability of the model to generalize.**

# ### Data augmentation:

# In[13]:


from keras.preprocessing.image import ImageDataGenerator

WIDTH = 299
HEIGHT = 299
BATCH_SIZE = 32

# data prep
train_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest')

validation_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
```

```
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest')

train_generator = train_datagen.flow_from_directory(
    TRAIN_DIR,
    target_size=(HEIGHT, WIDTH),
        batch_size=BATCH_SIZE,
        class_mode='categorical')

validation_generator = validation_datagen.flow_from_directory(
    TEST_DIR,
    target_size=(HEIGHT, WIDTH),
    batch_size=BATCH_SIZE,
    class_mode='categorical')


# In[14]:


EPOCHS = 5
BATCH_SIZE = 32
STEPS_PER_EPOCH = 320
VALIDATION_STEPS = 64

MODEL_FILE = 'filename.model'

history = model.fit_generator(
    train_generator,
    epochs=EPOCHS,
    steps_per_epoch=STEPS_PER_EPOCH,
    validation_data=validation_generator,
    validation_steps=VALIDATION_STEPS)

model.save(MODEL_FILE)


# In[16]:


# testing the model
model.evaluate_generator(validation_generator, steps=1,  max_queue_size=10, workers=
1, use_multiprocessing=False, verbose=0)
```

```
# In[17]:


# summarize history for loss
plt.figure()
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper right')
plt.show()


# In[ ]:






# ## RESNET 50

# In[18]:


from keras.applications.resnet50 import ResNet50, preprocess_input
HEIGHT = 300
WIDTH = 300
CLASSES = 2
base_model = ResNet50(weights='imagenet',
                      include_top=False,
                      input_shape=(HEIGHT, WIDTH, 3))


# In[ ]:


x = base_model.output
x = GlobalAveragePooling2D(name='avg_pool')(x)
x = Dropout(0.4)(x)
predictions = Dense(CLASSES, activation='softmax')(x)
model = Model(inputs=base_model.input, outputs=predictions)

# transfer learning
for layer in base_model.layers:
    layer.trainable = False

model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
```

```
              metrics=['accuracy'])


# In[20]:


from keras.preprocessing.image import ImageDataGenerator

BATCH_SIZE = 32

# data prep
train_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest')

validation_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest')

train_generator = train_datagen.flow_from_directory(
    TRAIN_DIR,
    target_size=(HEIGHT, WIDTH),
        batch_size=BATCH_SIZE,
        class_mode='categorical')

validation_generator = validation_datagen.flow_from_directory(
    TEST_DIR,
    target_size=(HEIGHT, WIDTH),
    batch_size=BATCH_SIZE,
    class_mode='categorical')


# In[21]:


EPOCHS = 5
BATCH_SIZE = 32
```

```
STEPS_PER_EPOCH = 320
VALIDATION_STEPS = 64

MODEL_FILE = 'resnet.model'

history = model.fit_generator(
    train_generator,
    epochs=EPOCHS,
    steps_per_epoch=STEPS_PER_EPOCH,
    validation_data=validation_generator,
    validation_steps=VALIDATION_STEPS)

model.save(MODEL_FILE)


# In[22]:


# testing the model
model.evaluate_generator(validation_generator, steps=1,  max_queue_size=10, workers=
1, use_multiprocessing=False, verbose=0)


# In[23]:


# summarize history for loss
plt.figure()
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper right')
plt.show()


# ## VGG 19

# In[24]:


from keras.applications import VGG19
HEIGHT = 224
WIDTH = 224
CLASSES = 2
BATCH_SIZE = 32

base_model = VGG19(weights='imagenet',
```

```
                  include_top=False,
                  input_shape=(HEIGHT, WIDTH, 3))


# In[ ]:


from keras.models import Model
from keras.layers import Dense, GlobalAveragePooling2D, Dropout
from keras.applications.vgg19 import vgg19, preprocess_input

x = base_model.output
x = GlobalAveragePooling2D(name='avg_pool')(x)
x = Dropout(0.4)(x)
predictions = Dense(CLASSES, activation='softmax')(x)
model = Model(inputs=base_model.input, outputs=predictions)

# transfer learning
for layer in base_model.layers:
    layer.trainable = False

model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['accuracy'])


# In[26]:


from keras.preprocessing.image import ImageDataGenerator


# data prep
train_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest')

validation_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
```

```
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest')

train_generator = train_datagen.flow_from_directory(
    TRAIN_DIR,
    target_size=(HEIGHT, WIDTH),
        batch_size=BATCH_SIZE,
        class_mode='categorical')

validation_generator = validation_datagen.flow_from_directory(
    TEST_DIR,
    target_size=(HEIGHT, WIDTH),
    batch_size=BATCH_SIZE,
    class_mode='categorical')


# In[27]:


EPOCHS = 5
BATCH_SIZE = 32
STEPS_PER_EPOCH = 320
VALIDATION_STEPS = 64

MODEL_FILE = 'vgg19.model'

history = model.fit_generator(
    train_generator,
    epochs=EPOCHS,
    steps_per_epoch=STEPS_PER_EPOCH,
    validation_data=validation_generator,
    validation_steps=VALIDATION_STEPS)

model.save(MODEL_FILE)


# In[28]:


# testing the model
model.evaluate_generator(validation_generator, steps=1,  max_queue_size=10, workers=
1, use_multiprocessing=False, verbose=0)


# In[29]:


# summarize history for loss
```

```
plt.figure()
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper right')
plt.show()


# ## Summary:

# In[30]:


from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ["Model", "Test loss", "Test Accuracy"]

x.add_row(["Inception V3", "0.56", "0.78"])
x.add_row(["Resnet 50", "2.04", "0.43"])
x.add_row(["VGG 19", "0.42", '0.78'])


print(x)


# By observing from the above data, we can clearly conclude that VGG19 works best wi
th highest test accuracy and lowest test loss. So we would proceed predictions with
VGG19 model.

# #### Predictions:

# In[ ]:


# defining functions to visualise and test the data


def predict(model, img):
    """Run model prediction on image
    Args:
        model: keras model
        img: PIL format image
    Returns:
        list of predicted labels and their probabilities
    """
```

```
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)
    x = preprocess_input(x)
    preds = model.predict(x)
    return preds[0]

def plot_preds(img, preds):
    """Displays image and the top-n predicted probabilities in a bar graph
    Args:
        preds: list of predicted labels and their probabilities
    """
    labels = ("tb", "ntb")
    gs = gridspec.GridSpec(2, 1, height_ratios=[4, 1])
    plt.figure(figsize=(8,8))
    plt.subplot(gs[0])
    plt.imshow(np.asarray(img))
    plt.subplot(gs[1])
    plt.barh([0, 1], preds, alpha=0.5)
    plt.yticks([0, 1], labels)
    plt.xlabel('Probability')
    plt.xlim(0, 1)
    plt.tight_layout()


# In[ ]:


# importing packages
from keras.preprocessing import image
from keras.models import load_model
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
model = load_model(MODEL_FILE)


# In[ ]:


# visualizing a test case
img = image.load_img('test/tb/CHNCXR_0328_1.png', target_size=(HEIGHT, WIDTH))
preds = predict(model, img)

plot_preds(np.asarray(img), preds)
preds


# In[ ]:
```

```
# predicting the output of the images
y_pred = []

directory = 'test/tb/'
import os

for filename in os.listdir(directory):
    if filename.endswith(".png"):
        img = image.load_img(directory + filename, target_size=(HEIGHT, WIDTH))
        preds = predict(model, img)
        y_pred.append(preds[0])
        continue
    else:
        continue


# In[ ]:


# appending class levels to a list
path="test/"
tb_test = path + "tb/"

label_tb_t = []

for i in range(34):
    label_tb_t.append("1")


# In[ ]:


# typecasting
labels = np.asarray(label_tb_t)
labels = labels.astype(int)


# In[ ]:


y_pred_1 = np.asarray(y_pred).reshape(-1, 1)
from sklearn.preprocessing import binarize
# it will return 1 for all values above 0.7 and 0 otherwise
# results are 2D so we slice out the first column
y_pred_class = binarize(y_pred_1, 0.7)


# In[ ]:
```

```
#
from sklearn import metrics
fpr, tpr, thresholds = metrics.roc_curve(labels, y_pred_class)


# In[ ]:


def evaluate_threshold(threshold):
    print('Sensitivity:', tpr[thresholds > threshold][-1])
    print('Specificity:', 1 - fpr[thresholds > threshold][-1])


# In[ ]:


evaluate_threshold(0.8)


# In[ ]:


print(metrics.confusion_matrix(labels, y_pred_class))
```

References

1. Cao, Y., Liu, C., Liu, B., Brunette, M. J., Zhang, N., Sun, T., ... & Curioso, W. H. (2016, June). Improving tuberculosis diagnostics using deep learning and mobile health technologies among resource-poor and marginalized communities. In *2016 IEEE first international conference on connected health: applications, systems and engineering technologies (CHASE)* (pp. 274-281). IEEE.

2. Panicker, R.O., Kalmady, K.S., Rajan, J. and Sabu, M.K., 2018. Automatic detection of tuberculosis bacilli from microscopic sputum smear images using deep learning methods. Biocybernetics and Biomedical Engineering, 38(3), pp.691-699.

3. Chen, M.L., Doddi, A., Royer, J., Freschi, L., Schito, M., Ezewudo, M., Kohane, I.S., Beam, A. and Farhat, M., 2018. Deep learning predicts tuberculosis drug resistance status from whole-genome sequencing data. BioRxiv, p.275628.

4. Szegedy, C., Ioffe, S., Vanhoucke, V. and Alemi, A.A., 2017, February. Inception-v4, inception-resnet and the impact of residual connections on learning. In Thirty-first AAAI conference on artificial intelligence.

5. Alom, M.Z., Hasan, M., Yakopcic, C. and Taha, T.M., 2017. Inception recurrent convolutional neural network for object recognition. arXiv preprint arXiv:1704.07709.

6. Narin, A., Kaya, C. and Pamuk, Z., 2020. Automatic detection of coronavirus disease (covid-19) using x-ray images and deep convolutional neural networks. arXiv preprint arXiv:2003.10849.

7. Bankman, I. ed., 2008. Handbook of medical image processing and analysis. Elsevier.

8. McAuliffe, M.J., Lalonde, F.M., McGarry, D., Gandler, W., Csaky, K. and Trus, B.L., 2001, July. Medical image processing, analysis and visualization in clinical research. In Proceedings 14th IEEE Symposium on Computer-Based Medical Systems. CBMS 2001 (pp. 381-386). IEEE.