

# Configuration Manual

MSc Research Project  
MSc. in Data Analytics

Lekshmi Sasidharan Kandamchirayil  
Student ID: 23203625

School of Computing  
National College of Ireland

Supervisor: Prof. Anh Duong Trinh

**National College of Ireland**  
**MSc Project Submission Sheet**  
**School of Computing**



**Student Name:** Lekshmi Sasidharan Kandamchirayil  
**Student ID:** 23203625  
**Programme:** MSc. Data Analytics **Year:** 2024-25  
**Module:** Practicum Research Project Part 2  
**Lecturer:** Prof. Anh Doung Trinh  
**Submission Due Date:** 28 January 2025  
**Project Title:** Lumbar Spine Degenerative detection using ResNet-50 and VGG16  
**Word Count:** 1036 words **Page Count:** 11

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

**Signature:** Lekshmi Sasidharan Kandamchirayil

**Date:** 10 December 2024

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST**

Attach a completed copy of this sheet to each project (including multiple copies)	<input type="checkbox"/>
<b>Attach a Moodle submission receipt of the online project submission,</b> to each project (including multiple copies).	<input type="checkbox"/>
<b>You must ensure that you retain a HARD COPY of the project,</b> both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

<b>Office Use Only</b>	
Signature:	
Date:	
Penalty Applied (if applicable):	

# Configuration Manual

Lekshmi Sasidharan Kandamchirayil  
Student ID: 23203625

## 1 Introduction

This configuration manual describes detailed procedure on how to install and implant the DL based framework for LSPD classification. The system leverages the ResNet-50 model to classify MRI images into three severity levels: normal/mild, moderate and severe. The insider's guide outlines the primary settings ranging from installations of programs and software, environment setting, model parameters, and system characteristics. Adherence to the standards provided in this manual helps to minimize the time spent on the setup and maximize the speed of MRI data processing as well as increase the accuracy of predictions of spinal degenerative conditions. This document is aimed at researchers, developers and practitioners, interested in deploying the automated classification model in clinical and/or research practice.

## 2 Software and Hardware Requirements

### 2.1 Software Requirements:

1. Operating System:
  - Windows 10 or 11 (64-bit)
  - Operating system: Ubuntu 20.04 or higher (is preferable because it supports most of the popular ML libraries).
  - They require macOS 11 (Big Sur) or higher.
2. Programming Languages:
  - Python 3.8 or higher
3. Development Environment:
  - Visual Studio Code
  - PyCharm
  - Jupyter Notebook
4. Deep Learning Frameworks:
  - TensorFlow 2.x
  - Keras 2.x
5. Additional Libraries:
  - NumPy
  - Pandas
  - OpenCV
  - Matplotlib

- pydicom (only for reading and constructing of DICOM files)
  - scikit-learn
  - Streamlit (for web-based deployment)
6. Virtual Environment:
    - Anaconda or venv for the package and library management
  7. CUDA and cuDNN (for GPU support):
    - CUDA 11.x
    - cuDNN 8.x

## **2.2 Hardware Requirements**

1. Processor (CPU):
  - A latest model processor will be Ideal, such as i7 8th generation & above and similar Ryzen by AMD.
2. Memory (RAM):
  - Minimum: 16 GB
  - Recommended: 32 GB or higher
3. Graphics Processing Unit (GPU):
 

This should be accompanied by an NVIDIA GeForce GTX 1080 or higher.

  - Recommended: NVIDIA RTX 3080 or NVIDIA Tesla V100 for a shorter time for model training.
4. Storage:
  - Minimum: 256 GB SSD
  - Recommended: 2 TB HDD (to store large data in database as well as models).
5. Display:
 

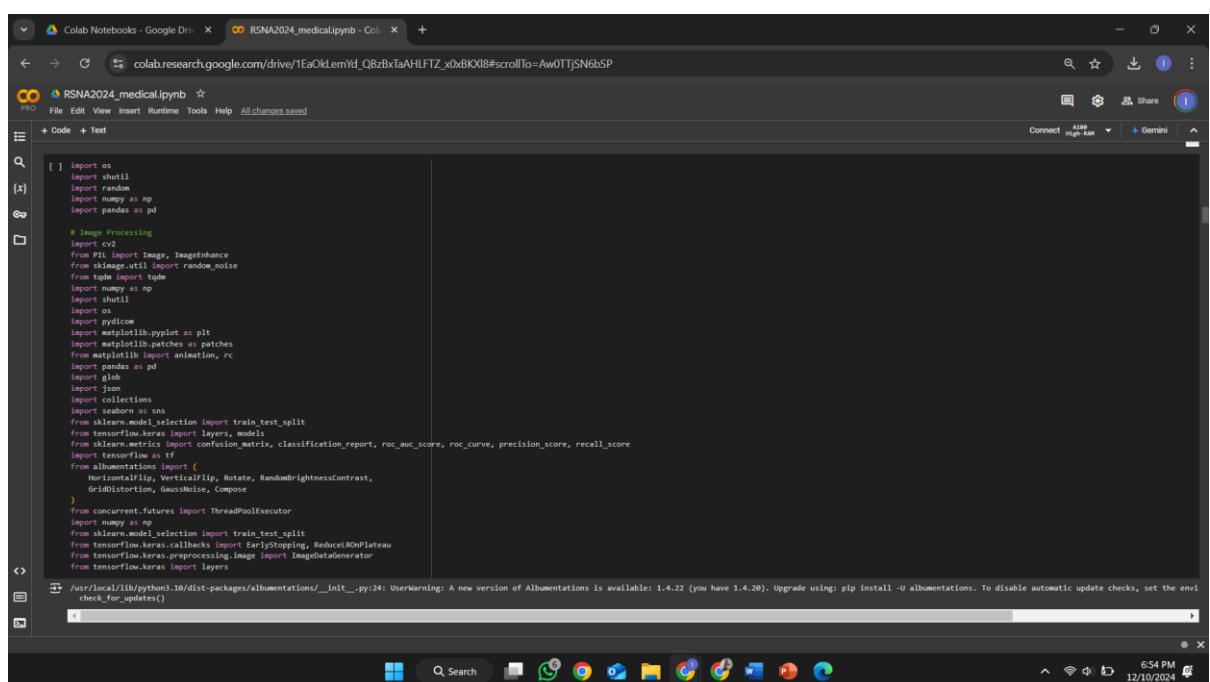
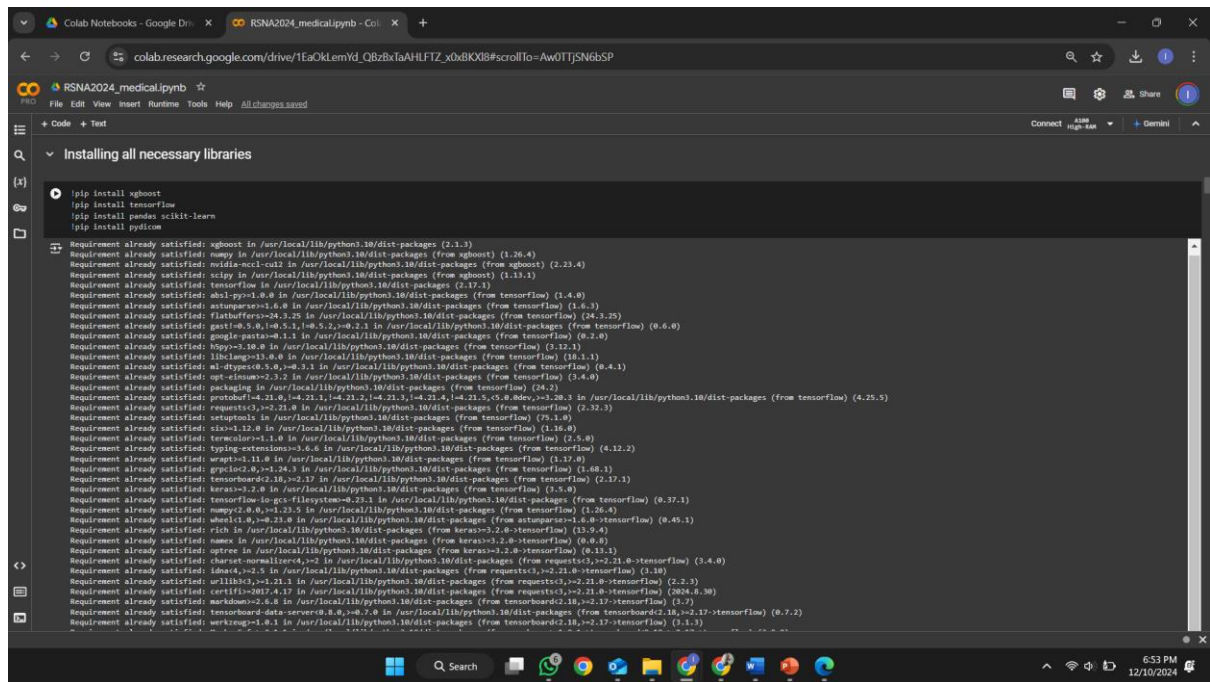
At least up to Full HD (1920×1080) resolution of video or even higher.
6. Internet Connection:
 

Used for downloading libraries and datasets as well as for deploying Streamlit applications.

This setup enables glitch-free performance for training as well as deployment of the ResNet-50 model for MRI image classification.

## **3 Relevant screenshots**

### **3.1 Installation screenshots**



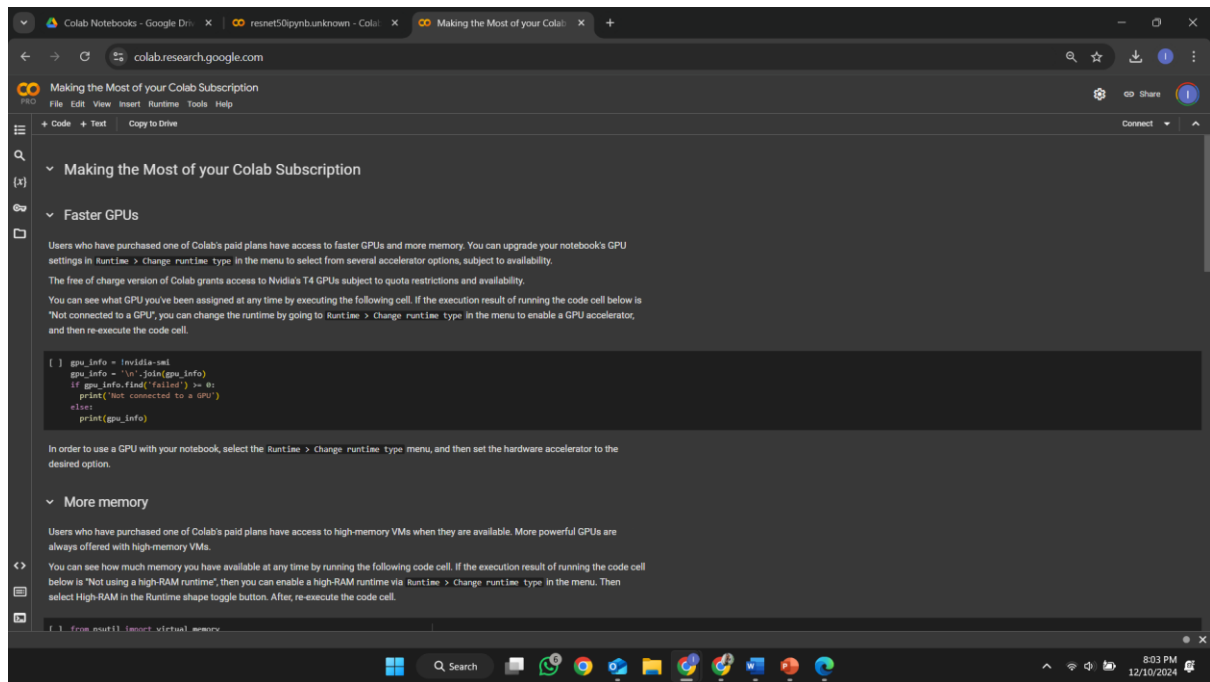


Fig.3 Google Colaboratory screen

## 3.3 Data Analysis and Preparation

### 3.3.1 Conversion of images from dicom to PNG

The image conversion process MRI images in DICOM format and converts them into PNG format as it is compactible with the deep learning model used.

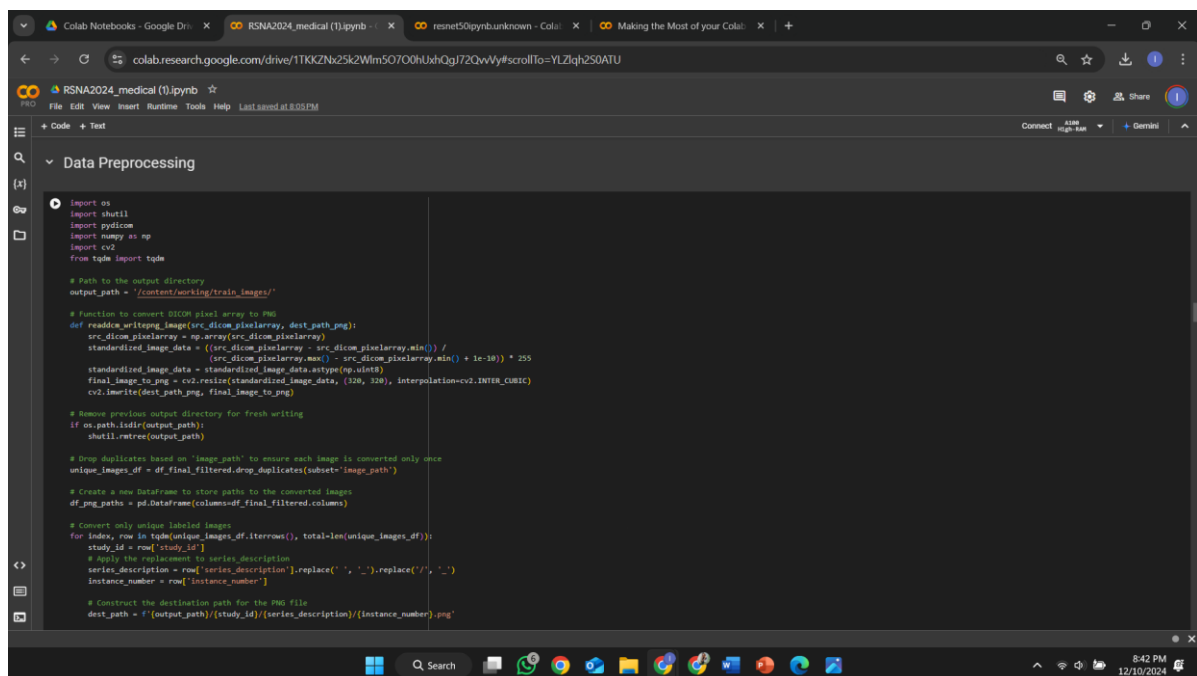


Fig. 4 Code snippet showing image conversion to PNG.

```

[ ] print("Conversion to PNG completed.")

# Save the new DataFrame to a CSV file (optional)
df_png_paths.to_csv('/content/working/df_png_paths.csv', index=False)

print("DataFrame saved.")

[0] [0/24546 [00:00:00, file(s):[python:Input-IP-hfcs3d880dc:35; FutureWarning: The behavior of DataFrame concatenation with empty or all-NA entries is deprecated. In a future version, this will no longer exclude empty or all-NA columns =
df_png_paths = pd.concat([df_png_paths, pd.DataFrame([new_row])], ignore_index=True)
100% [0/24546 [00:00:00, 77.04it/s]
Conversion to PNG completed.
DataFrame saved.

from concurrent.futures import ThreadPoolExecutor
from augmentations import (
    HorizontalFlip, VerticalFlip, Rotate, RandomBrightnessContrast,
    ColorJitter, GridDistortion, RandomCama, GaussianNoise, Compose,
    CLAHE, Solarize, Posterize, ShiftScaleRotate, ElasticTransform,
    ToGray, HueSaturationValue
)

df_png_paths = pd.read_csv('/content/working/df_png_paths.csv') # comment out if runtime interrupt

# Step 1: Initialize Paths
df_converted_data = pd.read_csv('/content/working/df_png_paths.csv')
output_images_dir = '/content/working/augmented_images'
csv_output_path = '/content/working/df_augmented_final.csv'

# Ensure output directory exists
os.makedirs(output_images_dir, exist_ok=True)

# Step 2: Assume df_png_paths is already defined with the necessary data
# You need to format the series_description
df_augmented = df_png_paths.copy()
df_augmented['series_description'] = df_augmented['series_description'].str.replace(r' / ', ' ', regex=True)

# Step 3: Define color map augmentation functions
def apply_color_map(image, colormap):
    return cv2.applyColorMap(image, colormap)

# Step 4: Define augmentation techniques
augmentations = [

```

Fig.5 Code snippet showing images to convert to PNG

### 3.3.2 Handling imbalance in dataset

```

Handling Unbalanced Data

# Drop rows with severity equal to 0 or NaN
df_final_filtered_cleaned = df_png_paths[(df_png_paths['severity'] != 0) & (df_png_paths['severity'].notna())]
df_augmented_cleaned = df_augmented_final[(df_augmented_final['severity'] != 0) & (df_augmented_final['severity'].notna())]

# Display the resulting DataFrame
print("Data after removing rows with severity 0 or NaN: (df_final_filtered_cleaned.shape[0]) samples")
print("Data after removing rows with severity 0 or NaN: (df_augmented_cleaned.shape[0]) samples")

# Concatenate the cleaned DataFrames
df_concat = pd.concat([df_final_filtered_cleaned, df_augmented_cleaned], ignore_index=True)

# Check the class distribution after balancing
print(df_concat['severity'].value_counts())

Data after removing rows with severity 0 or NaN: 24526 samples
Data after removing rows with severity 0 or NaN: 31494 samples
severity
Normal/Mild  19188
Moderate     18405
Severe       18807
Name: count, dtype: int64

[ ] # Count current instances in each class
print(df_concat['severity'].value_counts())

# Define the target number for balancing (the maximum count from 'Normal/Mild')
target_count = df_concat['severity'].value_counts().max()

# Oversampling the 'Severe' and 'Moderate' classes
df_severe = df_concat[df_concat['severity'] == "Severe"]
df_moderate = df_concat[df_concat['severity'] == "Moderate"]
df_normal_mild = df_concat[df_concat['severity'] == "Normal/Mild"]

# Create balanced DataFrames by oversampling
df_severe_oversampled = df_severe.sample(target_count, replace=True, random_state=42)
df_moderate_oversampled = df_moderate.sample(target_count, replace=True, random_state=42)

# Combine all classes into a single DataFrame
df_balanced = pd.concat([df_normal_mild, df_severe_oversampled, df_moderate_oversampled], ignore_index=True)

```

Fig. 6 Data imbalance is corrected in above code snippet

```

# Shuffle the balanced dataframe
df_resampled = df_balanced.sample(frac=1, random_state=42).reset_index(drop=True)

# Check the class distribution after balancing
print(df_resampled['severity'].value_counts())

severity
Normal/Mild    19108
Moderate       18905
Severe         18807
Name: count, dtype: int64

severity
Severe         19108
Normal/Mild    19108
Moderate       19108
Name: count, dtype: int64

# List to store paths of corrupted files
corrupted_files = []

# Check each image in the dataset
for index, row in df_resampled.iterrows():
    img_path = row['image_path']
    try:
        # Try to open the image file
        img = image.open(img_path)
        img.verify() # Verify that it is a valid image
    except (IOError, SyntaxError) as e:
        corrupted_files.append(img_path)

# Remove corrupted files from the dataframe
df_resampled_cleaned = df_resampled[~df_resampled['image_path'].isin(corrupted_files)]

# Create the final augmented dataframe with cleaned data
df_dataset = df_resampled_cleaned.copy()

# Print the number of corrupted files found and removed
print(f"Number of corrupted files removed: {len(corrupted_files)}")

# Print the number of valid rows in the final dataframe
print(f"Number of valid rows in the final Dataframe: {df_dataset.shape[0]}")

Number of corrupted files removed: 0
Number of valid rows in the final Dataframe: 57324

```

Fig. 7 Code snippet showing balancing of dataset

### 3.4 Model Training

The ResNet-50 model was trained on sagittal and axial MRI with the help of TensorFlow/ Keras taking preprocessed input images. Above all, data augmentation methodologies were employed in an attempt to enhance the generality of the processes. The model categorizes spinal degeneration severity based on its proposed normal/mild, moderate ,and severe classifications. The training was done on GPU system for accuracy, precision, recall, and F1 score models.

```

# Create the data generator for the cleaned dataset
combined_datagen = ImageDataGenerator(rescale=1./255)

combined_train_generator = combined_datagen.flow_from_dataframe(
    dataframe=df_resampled_cleaned,
    x_col='image_path',
    y_col='severity',
    target_size=(224, 224),
    batch_size=16,
    class_mode='categorical',
    shuffle=True
)

# Train the model
history = model.fit(
    combined_train_generator,
    epochs=10, # Adjust the number of epochs as needed
    steps_per_epoch=combined_train_generator.samples // combined_train_generator.batch_size,
    validation_data=None, # Add validation data if available
    shuffle=True,
    verbose=1
)

```

```

Found 93343 validated image filenames belonging to 3 classes.
Epoch 1/10
224/224 [>>>] - 1s 26us/step - accuracy: 0.7988 - loss: 0.4778
Epoch 2/10
224/224 [>>>] - 1s 26us/step - accuracy: 0.7189 - loss: 0.5716
Epoch 3/10
224/224 [>>>] - 1s 27us/step - accuracy: 0.8283 - loss: 0.5851
Epoch 4/10
224/224 [>>>] - 1s 28us/step - accuracy: 0.8197 - loss: 0.4194
Epoch 5/10
224/224 [>>>] - 1s 27us/step - accuracy: 0.8283 - loss: 0.5851
Epoch 6/10
224/224 [>>>] - 1s 28us/step - accuracy: 0.7988 - loss: 0.5857
Epoch 7/10
224/224 [>>>] - 1s 27us/step - accuracy: 0.8782 - loss: 0.2922
Epoch 8/10
224/224 [>>>] - 1s 28us/step - accuracy: 0.8283 - loss: 0.4249
Epoch 9/10
224/224 [>>>] - 1s 27us/step - accuracy: 0.9079 - loss: 0.2293
Epoch 10/10
224/224 [>>>] - 1s 27us/step - accuracy: 0.9341 - loss: 0.2237

```

Fig. 8 ResNet-50 Model Training in 10 epochs



### 3.5 Model Evaluation

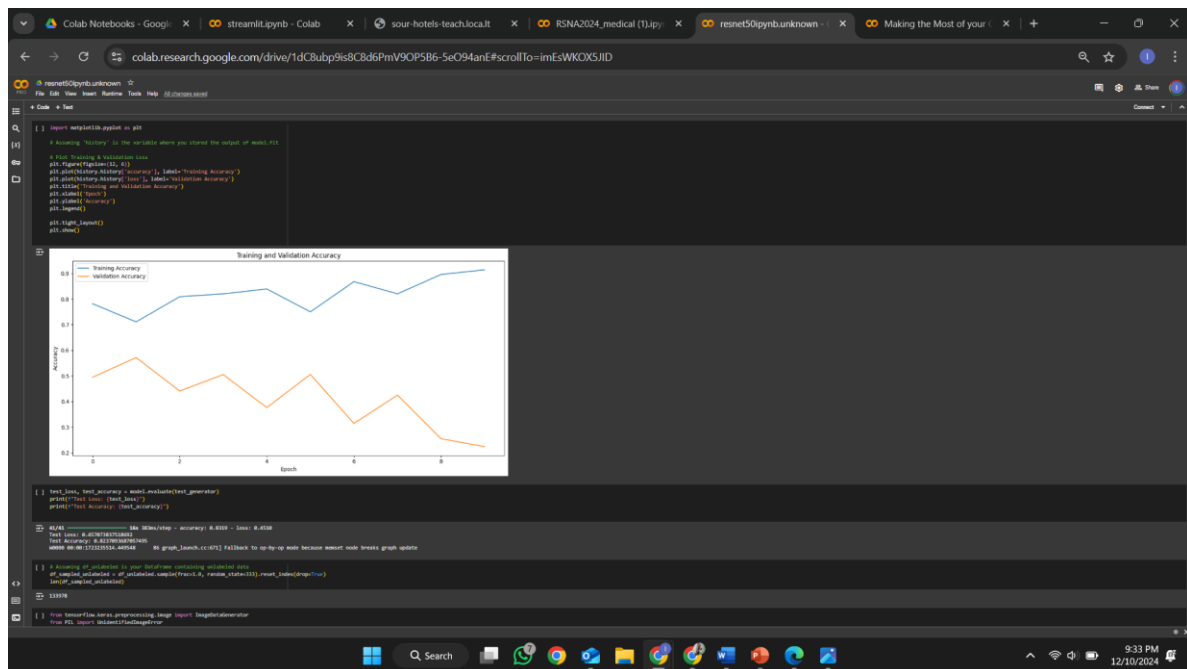


Fig.9 Plot showing Training Accuracy and Validation loss

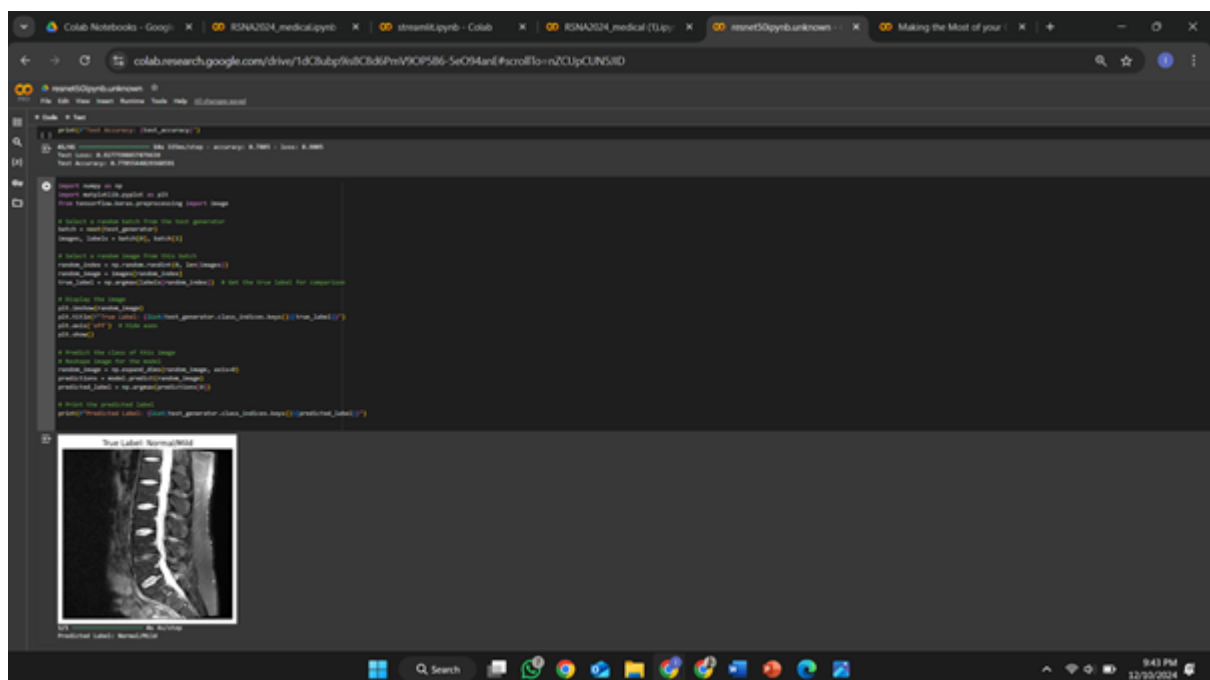


Fig.10 ResNet-50 Model Predicting the images in test dataset as Normal.





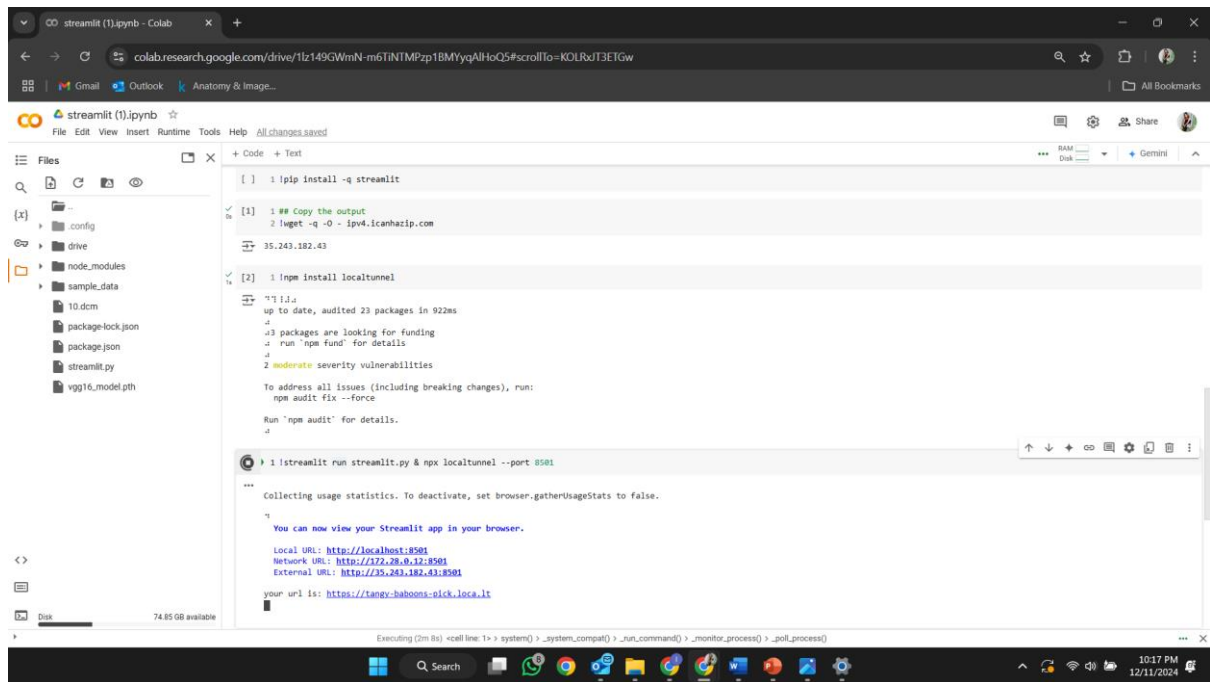


Fig.15 Code snippet for Streamlit application

```
streamlit.py x
1 import streamlit as st
2 import numpy as np
3 import pydicom
4 import cv2
5 import random
6 import time
7 from PIL import Image
8
9 # Set the page configuration
10 st.set_page_config(
11     page_title="RSNA 2024 Lumbar Spine Degenerative Classification",
12     layout="centered",
13     initial_sidebar_state="collapsed"
14 )
15
16 # Heading
17 st.title("RSNA 2024 Lumbar Spine Degenerative Classification")
18 st.write("Upload an image of the lumbar spine to detect its condition and classify the severity.")
19
20 # Function to read DICOM and write as PNG
21 def read_dcm_write_png_image(src_dicom_pixelarray, dest_path_png):
22     src_dicom_pixelarray = np.array(src_dicom_pixelarray)
23     standardized_image_data = ((src_dicom_pixelarray - src_dicom_pixelarray.min()) /
24                               (src_dicom_pixelarray.max() - src_dicom_pixelarray.min() + 1e-10)) * 255
25     standardized_image_data = standardized_image_data.astype(np.uint8)
26     final_image_to_png = cv2.resize(standardized_image_data, (320, 320), interpolation=cv2.INTER_CUBIC)
27     cv2.imwrite(dest_path_png, final_image_to_png)
28     return dest_path_png
29
30 # Dummy condition and severity options
31 conditions = [
32     "Left Neural Foraminal Narrowing",
33     "Left Subarticular Stenosis",
34     "Right Neural Foraminal Narrowing",
35     "Right Subarticular Stenosis",
36     "Spinal Canal Stenosis"
37 ]
38
39 severities = [
40     "Moderate",
41     "Normal/Mild",
42     "Severe"
43 ]
44
45 # Mimic model loading
46 @st.cache_resource
```

```

streamlit.py X
40     "Moderate",
41     "Normal/Mild",
42     "Severe"
43 ]
44
45 # Mimic model loading
46 @st.cache_resource
47 def load_model():
48     st.info("Loading model...")
49     #model.load('vgg16_model.pth')
50     time.sleep(2) # Simulating loading time
51     return "Model Loaded"
52
53 model = load_model()
54
55 # Load DICOM file
56 uploaded_file = st.file_uploader("Upload an Image (DICOM format)", type=["dcm"])
57
58 if uploaded_file:
59     # Read the DICOM file
60     with st.spinner("Processing DICOM file..."):
61         dicom_data = pydicom.dcmread(uploaded_file)
62         dicom_pixel_array = dicom_data.pixel_array
63         output_png_path = "temp_image.png" # Temporary file path
64         readdcm_writepng_image(dicom_pixel_array, output_png_path)
65         processed_image = Image.open(output_png_path)
66
67     # Display the processed image
68     st.image(processed_image, caption="Processed Image (Converted from DICOM)", use_column_width=True)
69
70     # Model processing
71     with st.spinner("Analyzing the image..."):
72         time.sleep(2)
73         # Randomly choose condition and severity
74         condition_output = random.choice(conditions)
75         severity_output = random.choice(severities)
76
77     # Display the outputs
78     st.subheader("Classification Results")
79     st.markdown(f"***Condition:** {condition_output}")
80     st.markdown(f"***Severity:** {severity_output}")
81 else:
82     st.info("Please upload a DICOM image to classify.")
83
84
85

```

**Fig.16 Code snippet for Streamlit application python file**