UNIVERSITY OF CANBERRA
AUSTRALIA'S CAPITAL UNIVERSITY

Faculty of Science and Technology

Assignment Coversheet

Student ID number & Student Name	u3253198 Vidun Singhabahu
Unit name	Software Technology
Unit number	4483
Unit Tutor	Pranav Gupta
Assignment name	ST1 Capstone Project – Semester 2 2023
Due date	29/10/2023
Date submitted	29/10/2023

You must keep a photocopy or electronic copy of your assignment.

Student declaration

I certify that the attached assignment is my own work. Material drawn from other sources has been appropriately and fully acknowledged as to author/creator, source, and other bibliographic details.

Signature of student: VS **Date: 27/10/23**

Table of Contents

Introduction

This report describes the contents of the Python Capstone Project for the ST1 unit by following the requirements outlined in the assessment outline. The work presented throughout this project was based on the allocated dataset about COVID-19 CT scan lesion segmentation on Kaggle.

COVID-19 or Coronavirus was a global health crisis that originated in Wuhan, China and spread globally in late December 2019. It is a large family of viruses that cause respiratory infections (Healthdirect, 2022). Most people affected by the virus experience mild to moderate respiratory illness and soon recover without requiring special treatment. However, some will become seriously ill and require medical attention (World Health Organization, 2023).

A computerized tomography (CT) scan combines a series of X-ray images taken from different angles around your body and uses computer processing to create cross-sectional images of the bones, blood vessels and soft tissues inside your body (Mayo Clinic, 2022). Relating to my dataset, CT has played a vital role in providing useful information for clinicians to detect COVID-19 (National Library of Medicine, 2022). However, it is stated that differentiating infected regions from these CT scans can be quite difficult. This is why it is required that an applicable tool is developed to automatically do this segmentation from CT scans and then further evaluate the performance.

This report demonstrates the progression of this tool in the form of methods on Google Colab and TeachableMachinewithGoogle. Based on an image driven approach, consisting of the EDA, PDA, and implementation and deployment on Google Colab, this project would be documented and completed.

Methodology

1. Design & Development

To first perform the Exploratory Data Analysis (EDA) and then the Predictive Data Analytics (PDA) to display the images and other graphs. Once these are completed, it will assist in solving the problem at hand.

2. Implementation

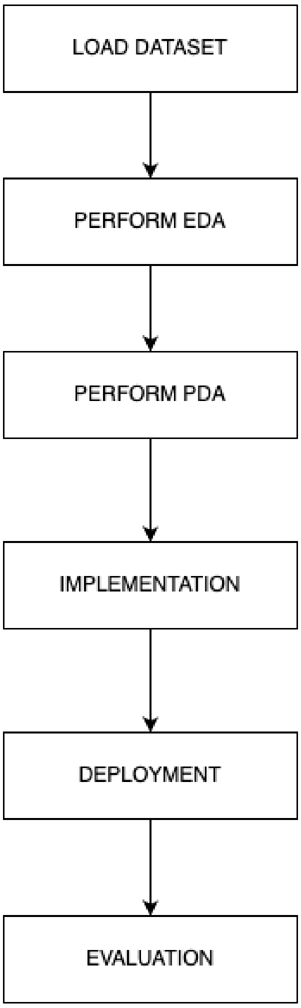
The learning model will be created through the use of TeachableMachinewithGoogle.

3. Deployment

After successfully implementing, it will be deployed on Google Colab with the aid of TeachableMachinewithGoogle.

Design & Development

Flowchart:



Algorithm for dataset:

LOAD DATASET

PERFORM EDA

PERFORM PDA

IMPLEMENTATION

DEPLOYMENT

EVALUATION

Dataset description:

The dataset that I have been allocated was the COVID-19 CT scan lesion segmentation dataset. It stated that a large lung CT scan dataset for COVID-19 was built by curating data from 7 public datasets. This dataset combines the lesion masks and their related frames of these 3 public datasets. The dataset consists of two classes, frames and masks, both holding 2729 files in each. This dataset will help with the end goal of image classification to see which class the image belongs to when uploaded.

Exploratory Data Analysis

Questions:

1. How do images from different classes look like.
2. How does the images from different classes look like with geometrical transformations.
3. What is impact of noising and denoising operations on image quality.
4. How discriminative are the salient features such as edges and corners for images corresponding to each class.
5. How discriminative are the images from different categories in terms of illumination and lighting artefacts.

```
#Needed for EDA in Google colab

import pandas as pd

from google.colab import drive

drive.mount('/content/drive/', force_remount=True)
```

```
Mounted at /content/drive/
```

```
#Install tensorflow

!pip install tensorflow==2.9.1
```

```
Requirement already satisfied: tensorflow==2.9.1 in
/usr/local/lib/python3.10/dist-packages (2.9.1)
Requirement already satisfied: absl-py>=1.0.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1)
(1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1)
(1.6.3)
Requirement already satisfied: flatbuffers<2,>=1.12 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1) (1.12)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1)
(0.4.0)
Requirement already satisfied: google-pasta>=0.1.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1)
(0.2.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1)
(1.59.0)
Requirement already satisfied: h5py>=2.9.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1)
(3.9.0)
```

```
Requirement already satisfied: keras<2.10.0,>=2.9.0rc0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1)
(2.9.0)
Requirement already satisfied: keras-preprocessing>=1.1.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1)
(1.1.2)
Requirement already satisfied: libclang>=13.0.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1)
(16.0.6)
Requirement already satisfied: numpy>=1.20 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1)
(1.23.5)
Requirement already satisfied: opt-einsum>=2.3.2 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1)
(3.3.0)
Requirement already satisfied: packaging in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1) (23.2)
Requirement already satisfied: protobuf<3.20,>=3.9.2 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1)
(3.19.6)
Requirement already satisfied: setuptools in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1)
(67.7.2)
Requirement already satisfied: six>=1.12.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1)
(1.16.0)
Requirement already satisfied: tensorboard<2.10,>=2.9 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1)
(2.9.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1)
(0.34.0)
Requirement already satisfied: tensorflow-estimator<2.10.0,>=2.9.0rc0
in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1)
(2.9.0)
Requirement already satisfied: termcolor>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1)
(2.3.0)
Requirement already satisfied: typing-extensions>=3.6.6 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1)
(4.5.0)
Requirement already satisfied: wrapt>=1.11.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.9.1)
(1.14.1)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
/usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0-
>tensorflow==2.9.1) (0.41.2)
Requirement already satisfied: google-auth<3,>=1.6.3 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.10,>=2.9-
>tensorflow==2.9.1) (2.17.3)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.10,>=2.9-
>tensorflow==2.9.1) (0.4.6)
Requirement already satisfied: markdown>=2.6.8 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.10,>=2.9-
>tensorflow==2.9.1) (3.5)
```

```
Requirement already satisfied: requests<3,>=2.21.0 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.10,>=2.9-
>tensorflow==2.9.1) (2.31.0)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.10,>=2.9-
>tensorflow==2.9.1) (0.6.1)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.10,>=2.9-
>tensorflow==2.9.1) (1.8.1)
Requirement already satisfied: werkzeug>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.10,>=2.9-
>tensorflow==2.9.1) (3.0.0)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3-
>tensorboard<2.10,>=2.9->tensorflow==2.9.1) (5.3.1)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3-
>tensorboard<2.10,>=2.9->tensorflow==2.9.1) (0.3.0)
Requirement already satisfied: rsa<5,>=3.1.4 in
/usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3-
>tensorboard<2.10,>=2.9->tensorflow==2.9.1) (4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
/usr/local/lib/python3.10/dist-packages (from google-auth-
oauthlib<0.5,>=0.4.1->tensorboard<2.10,>=2.9->tensorflow==2.9.1)
(1.3.1)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.10,>=2.9->tensorflow==2.9.1) (3.3.0)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.10,>=2.9->tensorflow==2.9.1) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.10,>=2.9->tensorflow==2.9.1) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.10,>=2.9->tensorflow==2.9.1) (2023.7.22)
Requirement already satisfied: MarkupSafe>=2.1.1 in
/usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1-
>tensorboard<2.10,>=2.9->tensorflow==2.9.1) (2.1.3)
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in
/usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1-
>google-auth<3,>=1.6.3->tensorboard<2.10,>=2.9->tensorflow==2.9.1)
(0.5.0)
Requirement already satisfied: oauthlib>=3.0.0 in
/usr/local/lib/python3.10/dist-packages (from requests-oauthlib>=0.7.0-
>google-auth-oauthlib<0.5,>=0.4.1->tensorboard<2.10,>=2.9-
>tensorflow==2.9.1) (3.2.2)
```

```python
# import system libs

import os
import time
import shutil
import pathlib
import itertools
```

```python
# import data handling tools

import cv2
import numpy as np
import pandas as pd
import seaborn as sns
sns.set_style('darkgrid')
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report

# import Deep learning Libraries

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam, Adamax
from tensorflow.keras.metrics import categorical_crossentropy
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten,
Dense, Activation, Dropout, BatchNormalization
from tensorflow.keras import regularizers

# Ignore Warnings

import warnings

warnings.filterwarnings("ignore")

print ('modules loaded')
```

```
modules loaded
```

```python
#Read Data and Store it in a dataframe

# Generate data paths with labels

#Use your own dataset path in Google drive

data_dir = '/content/drive/MyDrive/CAPSTONE_PROJECT/CapstoneData'

filepaths = []

labels = []



folds = os.listdir(data_dir)
```

```python
for fold in folds:

    foldpath = os.path.join(data_dir, fold)

    filelist = os.listdir(foldpath)

    for file in filelist:

        fpath = os.path.join(foldpath, file)

        filepaths.append(fpath)

        labels.append(fold)

# Concatenate data paths with labels into one dataframe

Fseries = pd.Series(filepaths, name= 'filepaths')

Lseries = pd.Series(labels, name='labels')

df = pd.concat([Fseries, Lseries], axis= 1)
```

```python
#EDA Step 1:How is the data distribution

df.head(5)
```

	filepaths	labels
0	/content/drive/MyDrive/CAPSTONE_PROJECT/Capsto...	frames
1	/content/drive/MyDrive/CAPSTONE_PROJECT/Capsto...	frames
2	/content/drive/MyDrive/CAPSTONE_PROJECT/Capsto...	frames
3	/content/drive/MyDrive/CAPSTONE_PROJECT/Capsto...	frames
4	/content/drive/MyDrive/CAPSTONE_PROJECT/Capsto...	frames

```python
df.tail(5)
```

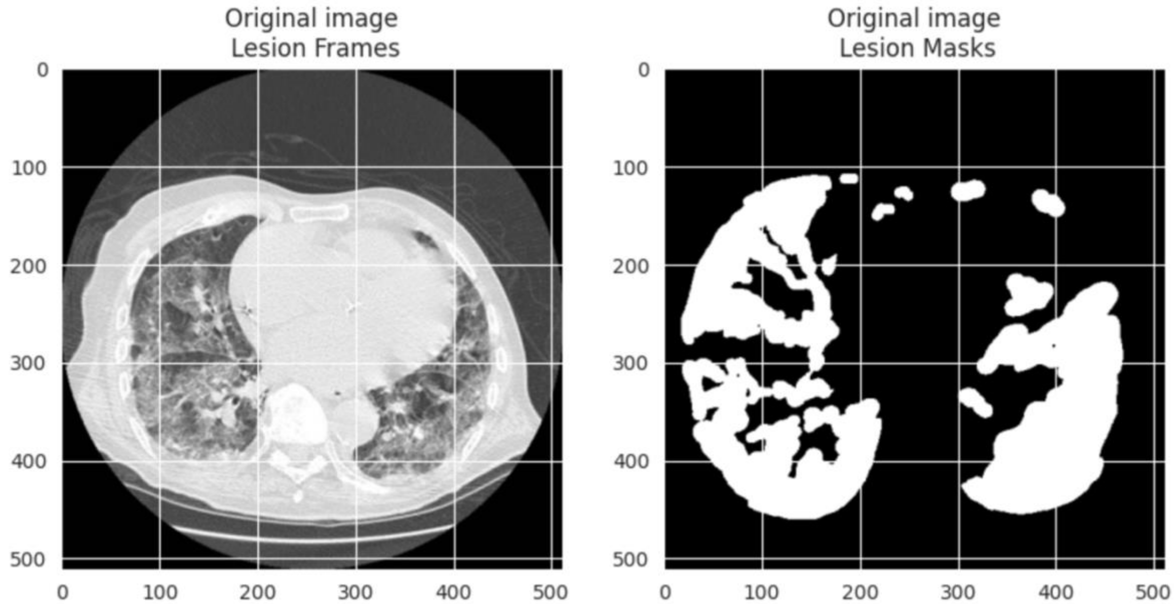
	filepaths	labels
995	/content/drive/MyDrive/CAPSTONE_PROJECT/Capsto...	masks
996	/content/drive/MyDrive/CAPSTONE_PROJECT/Capsto...	masks
997	/content/drive/MyDrive/CAPSTONE_PROJECT/Capsto...	masks
998	/content/drive/MyDrive/CAPSTONE_PROJECT/Capsto...	masks
999	/content/drive/MyDrive/CAPSTONE_PROJECT/Capsto...	masks

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   filepaths  1000 non-null   object
 1   labels     1000 non-null   object
dtypes: object(2)
memory usage: 15.8+ KB
```

```
#EDA Q1: How do images from different classes look like (Read and
Display Images)

import cv2
import matplotlib.pyplot as plt

%matplotlib inline

img_path_1 =
'/content/drive/MyDrive/CAPSTONE_PROJECT/CapstoneData/frames/bjorke_1.p
ng'

img_1 = cv2.imread(img_path_1)

img_path_2 =
'/content/drive/MyDrive/CAPSTONE_PROJECT/CapstoneData/masks/bjorke_2.pn
g'

img_2 = cv2.imread(img_path_2)

plt.figure(figsize=(10, 10))

plt.subplot(121)

plt.imshow(img_1),plt.title('Original image\n Lesion Frames')
```

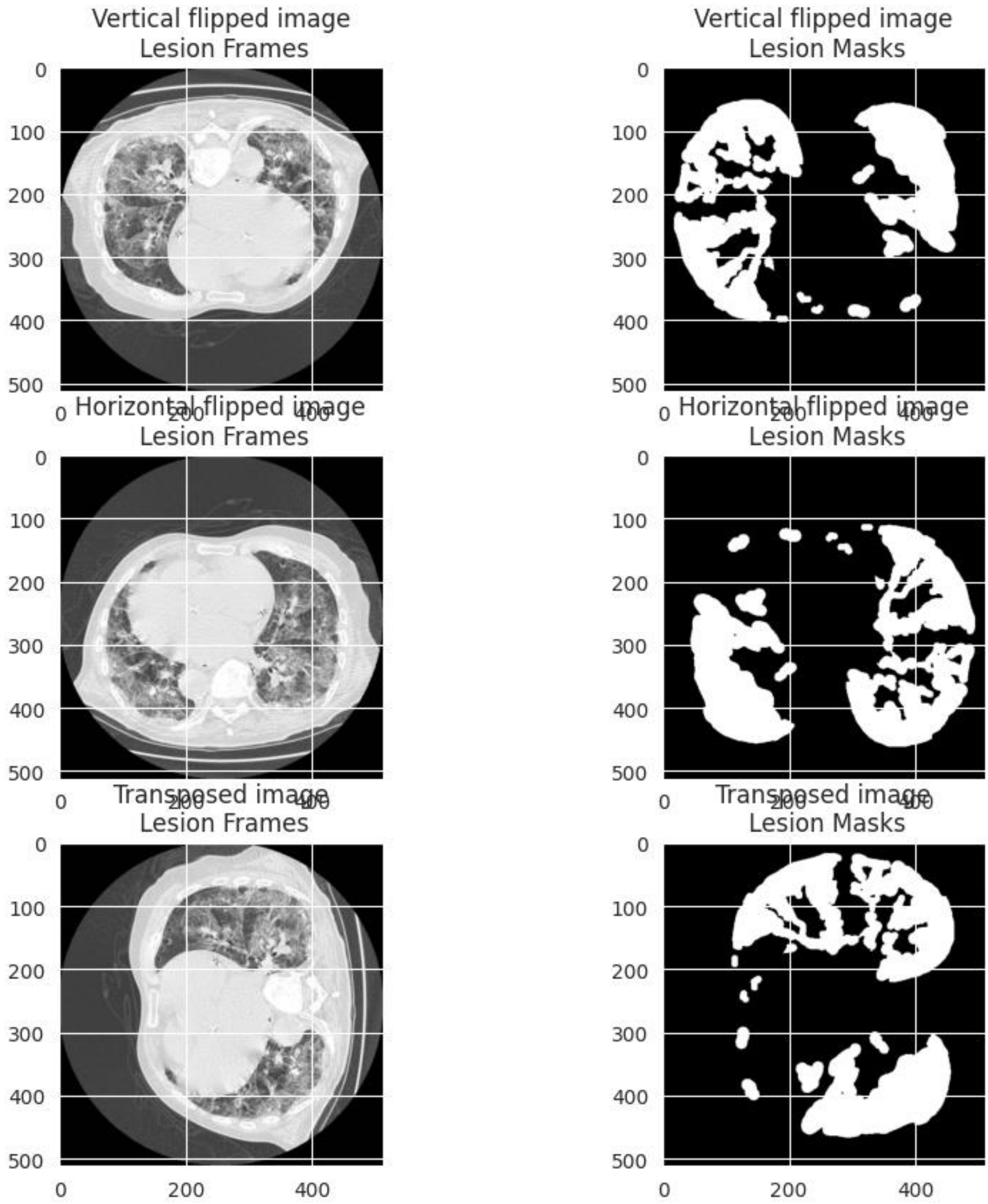
```
plt.subplot(122)

plt.imshow(img_2),plt.title('Original image\n Lesion Masks')

#################################################################
```



```
#EDA Q2: How does the images from different classes look like with
geometrical transformations (vertical flipping, horizontal flipping,
transposing)

import cv2

import matplotlib.pyplot as plt

%matplotlib inline



img_path_1 =
'/content/drive/MyDrive/CAPSTONE_PROJECT/CapstoneData/frames/bjorke_1.p
ng'

img_1 = cv2.imread(img_path_1)

img_path_2 =
'/content/drive/MyDrive/CAPSTONE_PROJECT/CapstoneData/masks/bjorke_2.pn
g'

img_2 = cv2.imread(img_path_2)
```

```
#Basic image manipulation (rotating/flipping/transpose)

flip_img_v1=cv2.flip(img_1,0) # vertical flip

flip_img_v2=cv2.flip(img_2,0) # vertical flip

#horizontal flip

flip_img_h1=cv2.flip(img_1,1) # horizontal flip

flip_img_h2=cv2.flip(img_2,1) # horizontal flip

#transpose

transp_img_1=cv2.transpose(img_1,1) # transpose

transp_img_2=cv2.transpose(img_2,1) # transpose



plt.figure(figsize=(10,10))

plt.subplot(321)

plt.imshow(flip_img_v1),plt.title('Vertical flipped image\n Lesion
Frames')

plt.subplot(322)

plt.imshow(flip_img_v2),plt.title('Vertical flipped image\n Lesion
Masks')

plt.subplot(323)

plt.imshow(flip_img_h1), plt.title('Horizontal flipped image\n Lesion
Frames')

plt.subplot(324)

plt.imshow(flip_img_h2), plt.title('Horizontal flipped image\n Lesion
Masks')

plt.subplot(325)

plt.imshow(transp_img_1),plt.title('Transposed image\n Lesion Frames')
```

```
plt.subplot(326)

plt.imshow(transp_img_2),plt.title('Transposed image\n Lesion Masks')

######################################################################
```

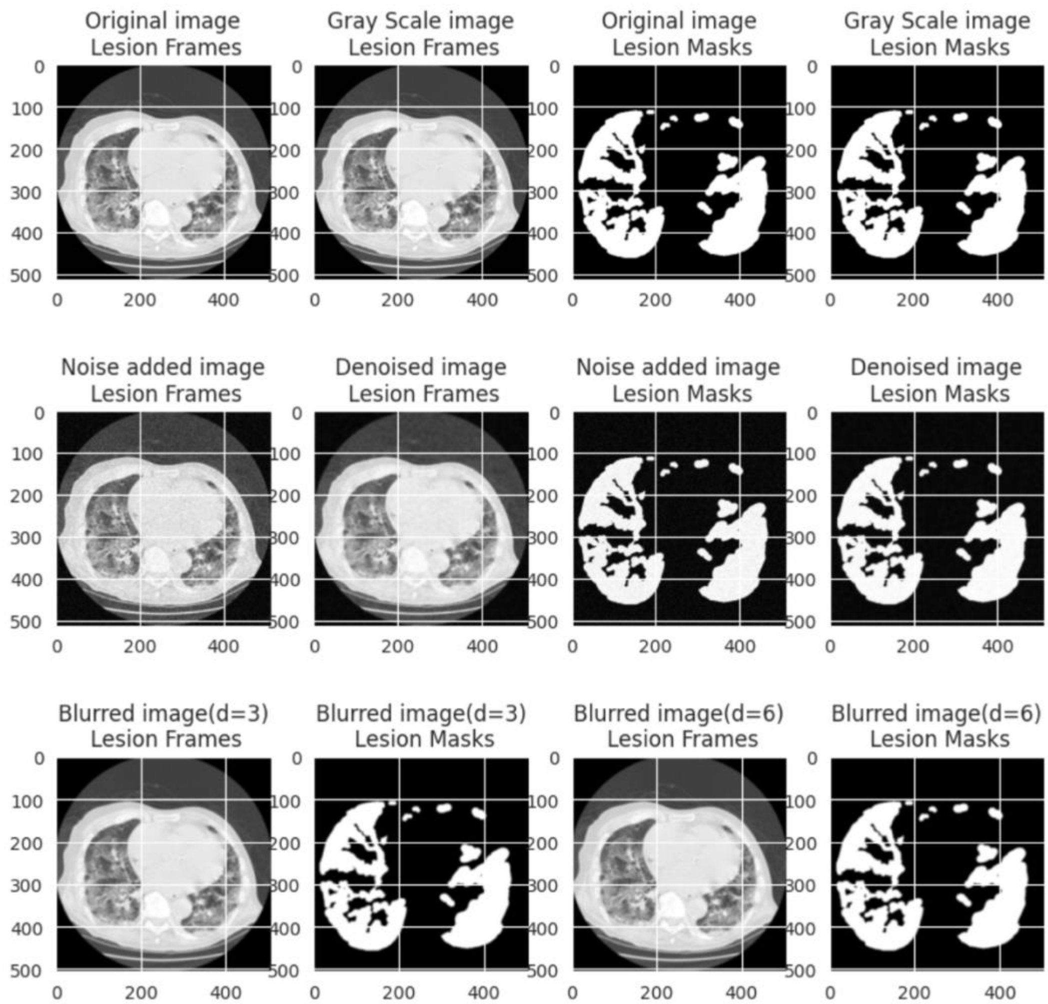



```
#EDA Q3: What is impact of noising and denoising operations on image
quality (aka Colour and Texture Analysis)

#Conversion to Gray scale image needed for colour and texture analysis
```

```
import cv2

import numpy as np

import matplotlib.pyplot as plt

import skimage

import skimage.color as skic

import skimage.filters as skif

import skimage.data as skid

import skimage.util as sku

%matplotlib inline



img_path_1 =
'/content/drive/MyDrive/CAPSTONE_PROJECT/CapstoneData/frames/bjorke_1.p
ng'

img_1 = cv2.imread(img_path_1)

img_path_2 =
'/content/drive/MyDrive/CAPSTONE_PROJECT/CapstoneData/masks/bjorke_2.pn
g'

img_2 = cv2.imread(img_path_2)



#gray scale conversion

img_1_gray = skic.rgb2gray(img_1)

img_2_gray = skic.rgb2gray(img_2)



# We add Gaussian noise and denoise using denoise_tv_bregman approach

#for img_1 and img_2

img_1_n = sku.random_noise(skic.rgb2gray(img_1))

img_1_d = skimage.restoration.denoise_tv_bregman(img_1_n, 5.)
```

```
img_2_n = sku.random_noise(skic.rgb2gray(img_2))

img_2_d = skimage.restoration.denoise_tv_bregman(img_2_n, 5.)



#Noise reduction using Gaussian Blur

d=3

img_1_blur3 = cv2.GaussianBlur(skic.rgb2gray(img_1), (2*d+1, 2*d+1), -
1)[d:-d,d:-d]

img_2_blur3 = cv2.GaussianBlur(skic.rgb2gray(img_2), (2*d+1, 2*d+1), -
1)[d:-d,d:-d]

img_1_blur6 = cv2.GaussianBlur(skic.rgb2gray(img_1), (2*d+1, 2*d+1), -
1)[d:-d,d:-d]

img_2_blur6 = cv2.GaussianBlur(skic.rgb2gray(img_2), (2*d+1, 2*d+1), -
1)[d:-d,d:-d]



plt.figure(figsize=(10,10))

#VisualisingGray scale images visualisation

plt.subplot(341), plt.imshow(img_1),plt.title('Original image\n Lesion
Frames')

plt.subplot(342), plt.imshow(img_1_gray, cmap = 'gray'),plt.title('Gray
Scale image\n Lesion Frames')

plt.subplot(343), plt.imshow(img_2),plt.title('Original image\n Lesion
Masks')

plt.subplot(344), plt.imshow(img_2_gray, cmap = 'gray'),plt.title('Gray
Scale image\n Lesion Masks')



#Visualising Noising-Denoising images

plt.subplot(345), plt.imshow(img_1_n,cmap = 'gray'), plt.title('Noise
added image\n Lesion Frames')
```

```
plt.subplot(346), plt.imshow(img_1_d,cmap = 'gray'),plt.title('Denoised image\n Lesion Frames')

plt.subplot(347), plt.imshow(img_2_n,cmap = 'gray'),plt.title('Noise added image\n Lesion Masks')

plt.subplot(348), plt.imshow(img_2_d,cmap = 'gray'),plt.title('Denoised image\n Lesion Masks')



#Visualising Noise Reduction with Gaussian Blurring

plt.subplot(349), plt.imshow(img_1_blur3,cmap = 'gray'), plt.title('Blurred image(d=3)\n Lesion Frames')

plt.subplot(3,4,10), plt.imshow(img_2_blur3,cmap = 'gray'),plt.title('Blurred image(d=3)\n Lesion Masks')

plt.subplot(3,4,11), plt.imshow(img_1_blur6,cmap = 'gray'),plt.title('Blurred image(d=6)\n Lesion Frames')

plt.subplot(3,4,12), plt.imshow(img_2_blur6,cmap = 'gray'),plt.title('Blurred image(d=6)\n Lesion Masks')

#################################################################
```



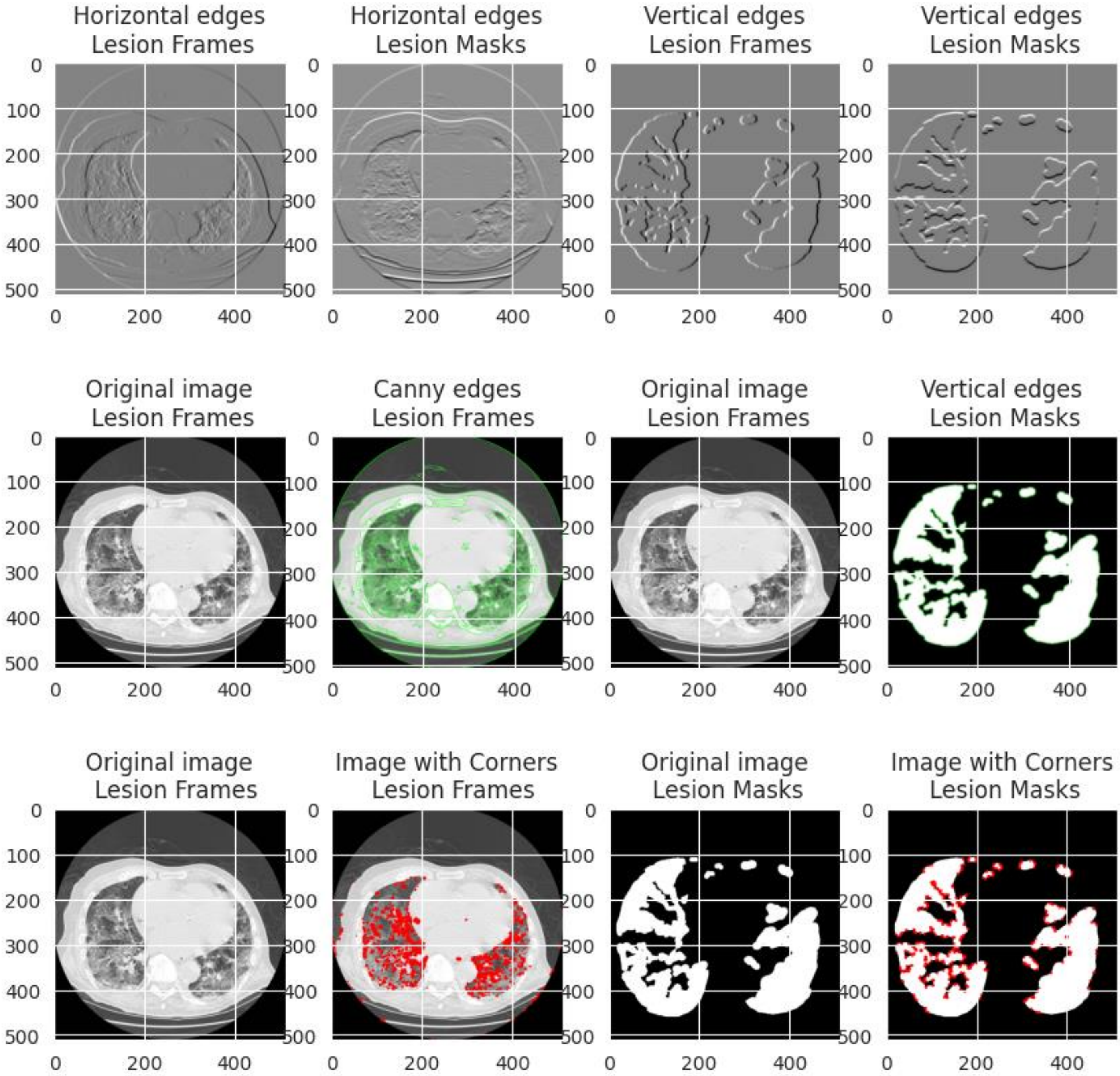
```
#EDA Q4: How discriminative are the salient features such as edges and
#corners for images corresponding to each class

#Conversion to Gray scale image needed for extracting edges and corners

import cv2
import numpy as np
import matplotlib.pyplot as plt
import skimage
import skimage.color as skic
import skimage.filters as skif
import skimage.data as skid
import skimage.util as sku

%matplotlib inline

img_path_1 =
'/content/drive/MyDrive/CAPSTONE_PROJECT/CapstoneData/frames/bjorke_1.p
ng'

img_1 = cv2.imread(img_path_1)

img_path_2 =
'/content/drive/MyDrive/CAPSTONE_PROJECT/CapstoneData/masks/bjorke_2.pn
g'

img_2 = cv2.imread(img_path_2)

#Sobel edge detector

#edge detector works on gray scale images

sobel_img_1=cv2.cvtColor(img_1,cv2.COLOR_BGR2GRAY)

sobel_img_2=cv2.cvtColor(img_2,cv2.COLOR_BGR2GRAY)

sobelx_img_1 = cv2.Sobel(sobel_img_1,cv2.CV_64F,1,0,ksize=9)

sobely_img_1 = cv2.Sobel(sobel_img_1,cv2.CV_64F,0,1,ksize=9)

sobelx_img_2 = cv2.Sobel(sobel_img_2,cv2.CV_64F,1,0,ksize=9)

sobely_img_2 = cv2.Sobel(sobel_img_2,cv2.CV_64F,0,1,ksize=9)

#Canny edge detector

#threshold selection
```

```
th1=30

th2=60

# Canny recommends threshold 2 is 3 times threshold 1

# you could try experimenting with this...

d=3

# gaussian blur

# this takes pixels in edgeresult where edge non-zero and colours them
bright green

edgeresult_1=img_1.copy()

edgeresult_1 = cv2.GaussianBlur(edgeresult_1, (2*d+1, 2*d+1), -1)[d:-
d,d:-d]

gray_1 = cv2.cvtColor(edgeresult_1, cv2.COLOR_BGR2GRAY)

edge_1 = cv2.Canny(gray_1, th1, th2)

edgeresult_1[edge_1 != 0] = (0, 255, 0)

edgeresult_2=img_2.copy()

edgeresult_2 = cv2.GaussianBlur(edgeresult_2, (2*d+1, 2*d+1), -1)[d:-
d,d:-d]

gray_2 = cv2.cvtColor(edgeresult_2, cv2.COLOR_BGR2GRAY)

edge_2 = cv2.Canny(gray_2, th1, th2)

edgeresult_2[edge_2 != 0] = (0, 255, 0)

#Corner detector

#detecting corners for  image_1

harris_1=img_1.copy()

#greyscale it

gray = cv2.cvtColor(harris_1,cv2.COLOR_BGR2GRAY)

gray = np.float32(gray)
```

```
blocksize=4 #

kernel_size=3 # sobel kernel: must be odd and fairly small

# run the harris corner detector

dst = cv2.cornerHarris(gray,blocksize,kernel_size,0.05) # parameters
are blocksize, Sobel parameter and Harris threshold

#result is dilated for marking the corners, this is visualisation
related and just makes them bigger

dst = cv2.dilate(dst,None)

#we then plot these on the input image for visualisation purposes,
using bright red

harris_1[dst>0.01*dst.max()]=[0,0,255]

#detecting corners for image_2

harris_2=img_2.copy()

#greyscale it

gray = cv2.cvtColor(harris_2,cv2.COLOR_BGR2GRAY)

gray = np.float32(gray)

blocksize=4 #

kernel_size=3 # sobel kernel: must be odd and fairly small

# run the harris corner detector

dst = cv2.cornerHarris(gray,blocksize,kernel_size,0.05) # parameters
are blocksize, Sobel parameter and Harris threshold

#result is dilated for marking the corners, this is visualisation
related and just makes them bigger

dst = cv2.dilate(dst,None)

#we then plot these on the input image for visualisation purposes,
using bright red

harris_2[dst>0.01*dst.max()]=[0,0,255]
```

```
#Visualisng Edges and Corners

plt.figure(figsize=(10,10))

#Visualising Sobel Edges

plt.subplot(341), plt.imshow(sobelx_img_1, cmap =
'gray'),plt.title('Horizontal edges\n Lesion Frames')

plt.subplot(342), plt.imshow(sobely_img_1, cmap =
'gray'),plt.title('Horizontal edges\n Lesion Masks')

plt.subplot(343), plt.imshow(sobelx_img_2, cmap =
'gray'),plt.title('Vertical edges\n  Lesion Frames')

plt.subplot(344), plt.imshow(sobely_img_2, cmap =
'gray'),plt.title('Vertical edges\n Lesion Masks')

#Visualising Canny Edges

plt.subplot(345), plt.imshow(img_1),plt.title('Original image\n Lesion
Frames')

plt.subplot(346), plt.imshow(edgeresult_1, cmap =
'gray'),plt.title('Canny edges\n  Lesion Frames')

plt.subplot(347), plt.imshow(img_1),plt.title('Original image\n Lesion
Frames')

plt.subplot(348), plt.imshow(edgeresult_2, cmap =
'gray'),plt.title('Vertical edges\n Lesion Masks')

#Visualising Corners

plt.subplot(349), plt.imshow(cv2.cvtColor(img_1,
cv2.COLOR_BGR2RGB)),plt.title('Original image\n  Lesion Frames')

plt.subplot(3,4,10), plt.imshow(cv2.cvtColor(harris_1,
cv2.COLOR_BGR2RGB)),plt.title('Image with Corners\n  Lesion Frames')

plt.subplot(3,4,11), plt.imshow(cv2.cvtColor(img_2,
cv2.COLOR_BGR2RGB)),plt.title('Original image\n Lesion Masks')

plt.subplot(3,4,12), plt.imshow(cv2.cvtColor(harris_2,
cv2.COLOR_BGR2RGB)),plt.title('Image with Corners\n Lesion Masks')
```

```
######################################################################
```

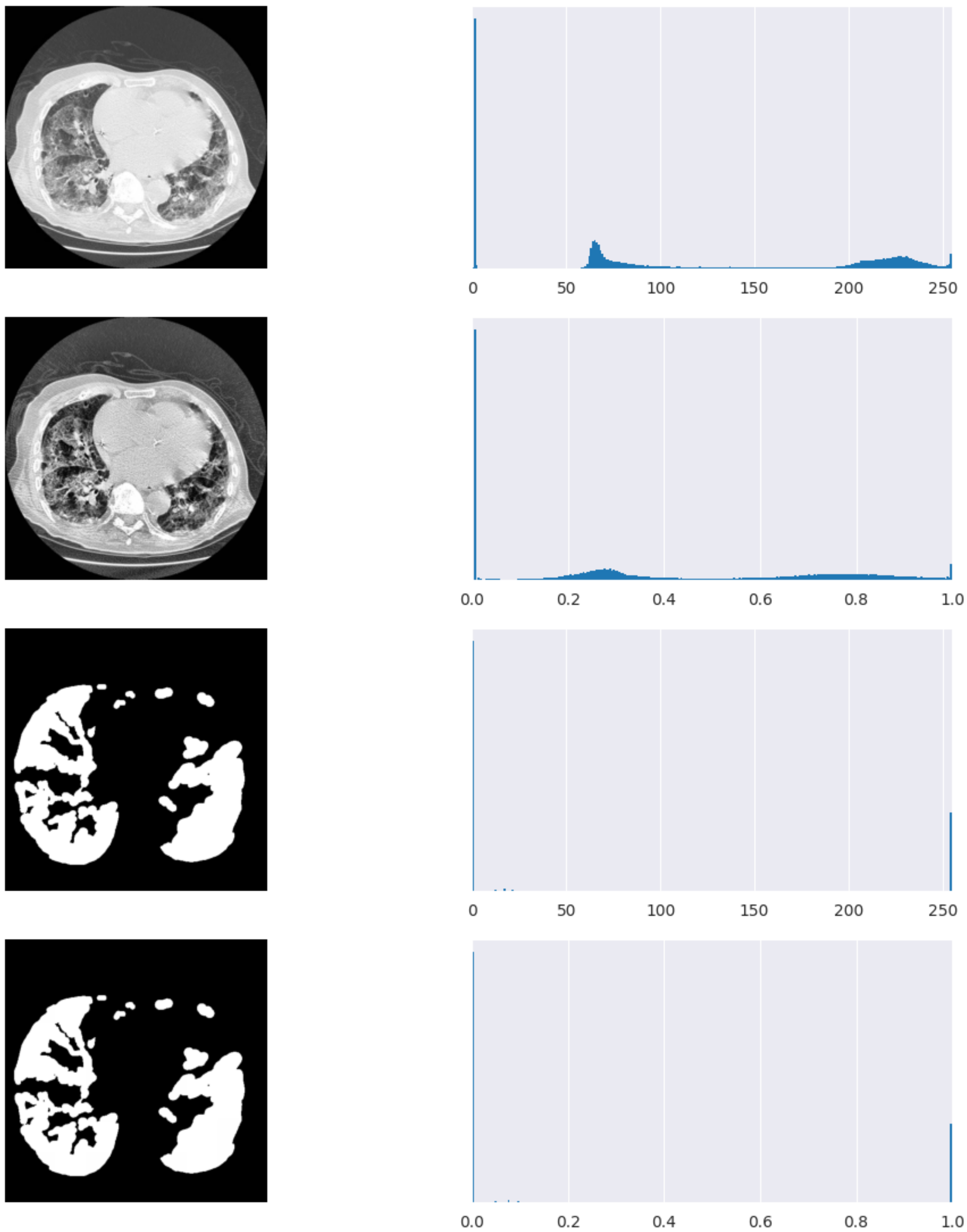


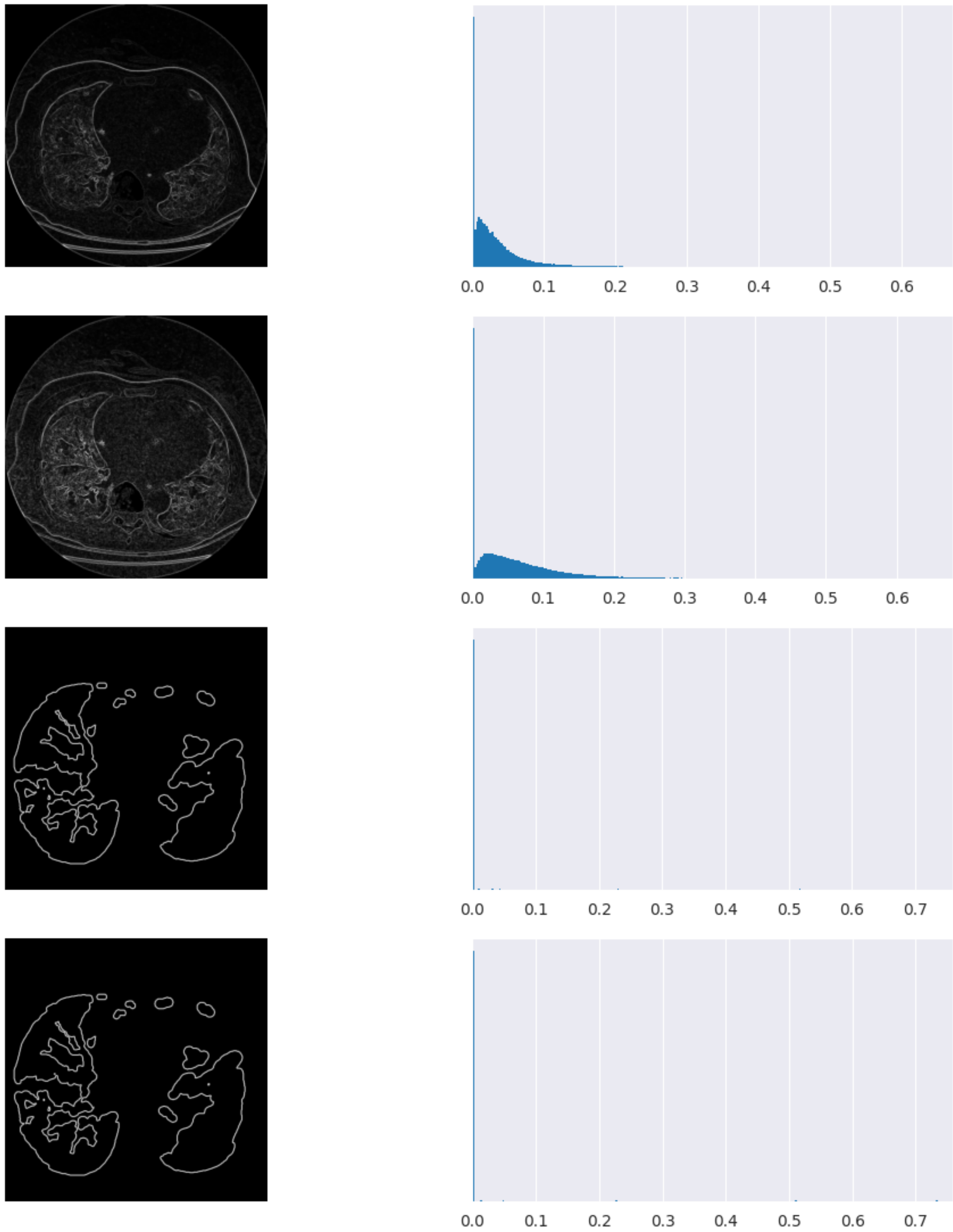

```
#EDA Q5: How discriminative are the images from different categories in
terms of illumination and lighting artefacts

# i.e. what is the impact of camera effects/exposure of an image

import cv2

import matplotlib.pyplot as plt
```

```
import numpy as np

import skimage

import skimage.color as skic

import skimage.filters as skif

import skimage.data as skid

import skimage.util as sku

import skimage.exposure as skie

%matplotlib inline








img_path_1 =
'/content/drive/MyDrive/CAPSTONE_PROJECT/CapstoneData/frames/bjorke_1.p
ng'

img_1 = cv2.imread(img_path_1)

img_path_2 =
'/content/drive/MyDrive/CAPSTONE_PROJECT/CapstoneData/masks/bjorke_2.pn
g'

img_2 = cv2.imread(img_path_2)



def show(img):

    # Display the image.

    fig, (ax1, ax2) = plt.subplots(1, 2,

                                   figsize=(12, 3))

    ax1.imshow(img, cmap=plt.cm.gray)

    ax1.set_axis_off()
```

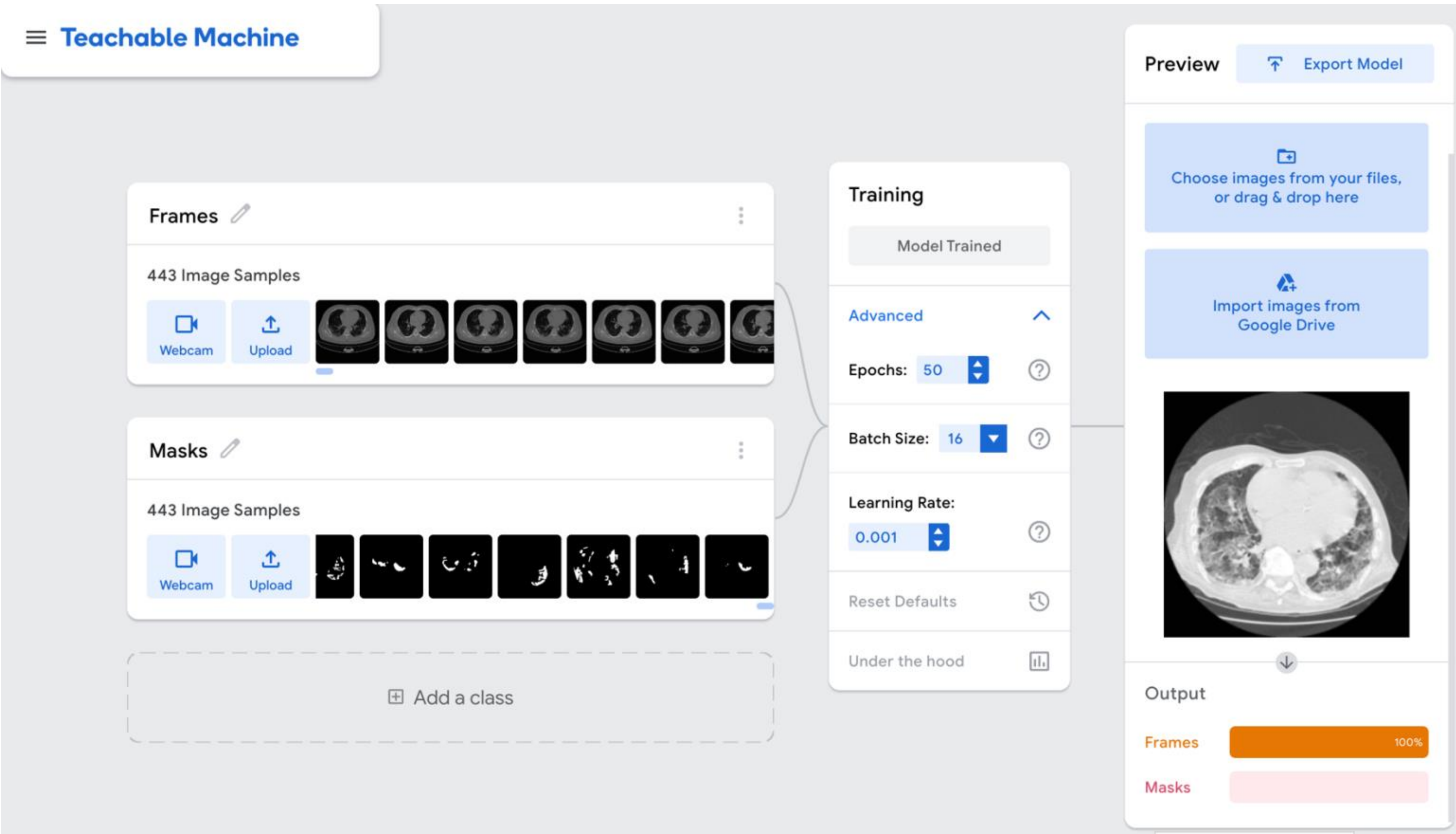
```
    # Display the histogram.
    ax2.hist(img.ravel(), lw=0, bins=256)
    ax2.set_xlim(0, img.max())
    ax2.set_yticks([])
    plt.show()
show(img_1)
# adaptive histogram equalisation
show(skie.equalize_adapthist(img_1))

show(img_2)
# adaptive histogram equalisation
show(skie.equalize_adapthist(img_2))

#class 1 image
img = skic.rgb2gray(img_1)
sobimg_nheq= skif.sobel(img)
show(sobimg_nheq)
img = skic.rgb2gray(skie.equalize_adapthist(img_1))
sobimg_heq_1 = skif.sobel(img)
show(sobimg_heq_1)
#class 2 image
img = skic.rgb2gray(img_2)
sobimg_nheq= skif.sobel(img)
```

```
show(sobimg_nheq)

img = skic.rgb2gray(skie.equalize_adapthist(img_2))

sobimg_heq_2 = skif.sobel(img)

show(sobimg_heq_2)
```




0.0 0.1 0.2 0.3 0.4 0.5 0.6
0.0 0.1 0.2 0.3 0.4 0.5 0.6
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7

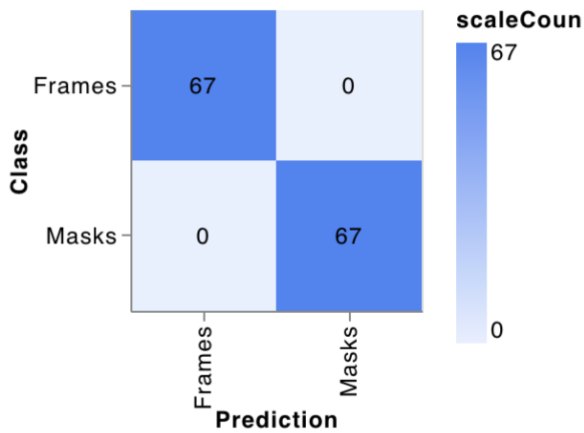
Predictive Data Analytics



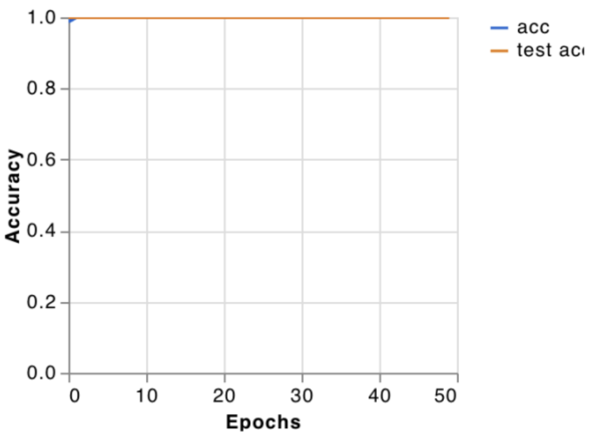
Accuracy per class

CLASS	ACCURACY	# SAMPLES
Frames	1.00	67
Masks	1.00	67

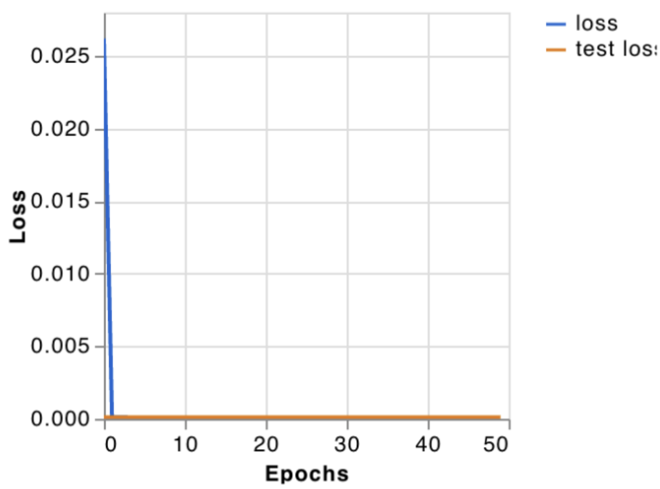
Confusion Matrix



Accuracy per epoch



Loss per epoch



Implementation & Deployment

1. The program is first implemented on TeachableMachinewithGoogle by obtaining the ‘converted.keras’ file.

2. Deployment will be done by using Google Colab as seen below or in the link attached.

Link for Google Drive:

https://drive.google.com/drive/folders/1yUiaqa27E9B_boxUA0mvaQ4dIyxmD_NP

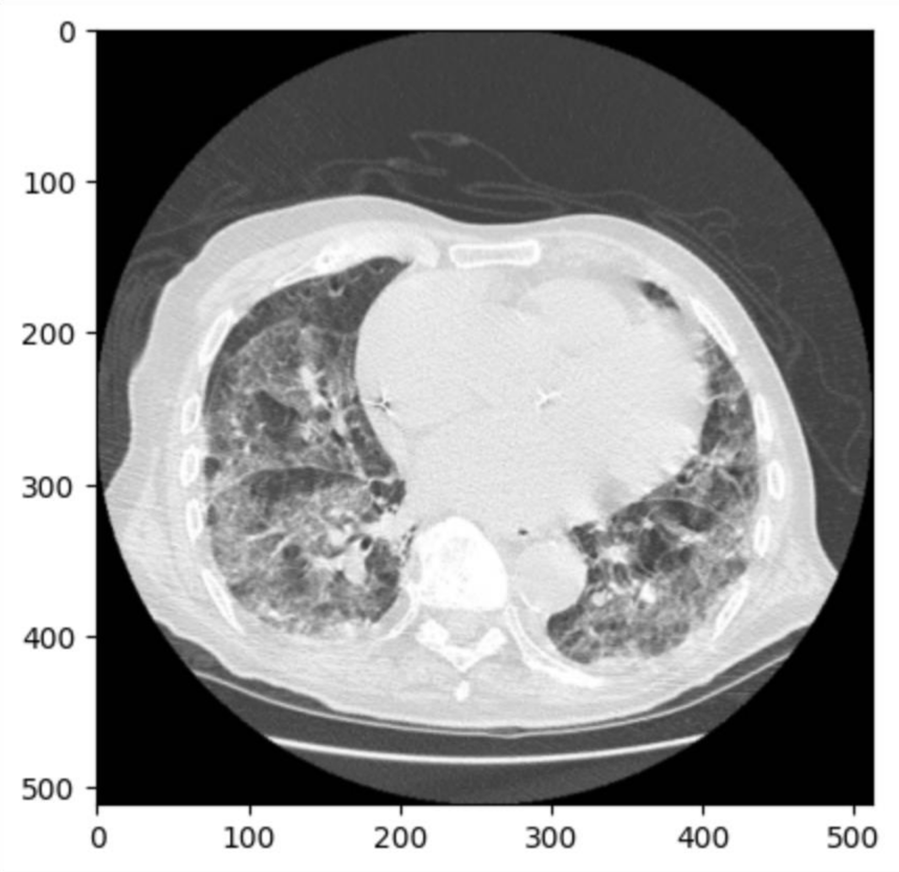
```
#Needed to get folder from Google drive for Google colab

import pandas as pd

from google.colab import drive

drive.mount('/content/drive/', force_remount=True)
Mounted at /content/drive/
```

```
from keras.models import load_model  # TensorFlow is required for Keras
to work
from PIL import Image, ImageOps  # Install pillow instead of PIL
import numpy as np

# Disable scientific notation for clarity
np.set_printoptions(suppress=True)

# Load the model
model =
load_model("/content/drive/MyDrive/CAPSTONE_PROJECT/converted_keras/ker
as_model.h5", compile=False)

# Load the labels
class_names =
open("/content/drive/MyDrive/CAPSTONE_PROJECT/converted_keras/labels.tx
t", "r").readlines()

# Create the array of the right shape to feed into the keras model
# The 'length' or number of images you can put into the array is
# determined by the first position in the shape tuple, in this case 1
data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)
```

```
# Replace this with the path to your image
image =
Image.open("/content/drive/MyDrive/CAPSTONE_PROJECT/bjorke_1.png").conv
ert("RGB")

# resizing the image to be at least 224x224 and then cropping from the
center
size = (224, 224)
image = ImageOps.fit(image, size, Image.Resampling.LANCZOS)

# turn the image into a numpy array
image_array = np.asarray(image)

# Normalize the image
normalized_image_array = (image_array.astype(np.float32) / 127.5) - 1

# Load the image into the array
data[0] = normalized_image_array

# Predicts the model
prediction = model.predict(data)
index = np.argmax(prediction)
class_name = class_names[index]
confidence_score = prediction[0][index]

# Print prediction and confidence score
print("Class:", class_name[2:], end="")
print("Confidence Score:", confidence_score)


from warnings import filterwarnings

import tensorflow as tf

from tensorflow import io

from tensorflow import image

from matplotlib import pyplot as plt

filterwarnings("ignore")

tf_img =
io.read_file("/content/drive/MyDrive/CAPSTONE_PROJECT/bjorke_1.png")

tf_img = image.decode_png(tf_img, channels=3)

print(tf_img.dtype)
```

```
plt.imshow(tf_img)

# plt.show()
```

```
1/1 [==============================] - 1s 1s/step
Class: Frames
Confidence Score: 0.99999976
<dtype: 'uint8'>
<matplotlib.image.AxesImage at 0x7b29140c3790>
```

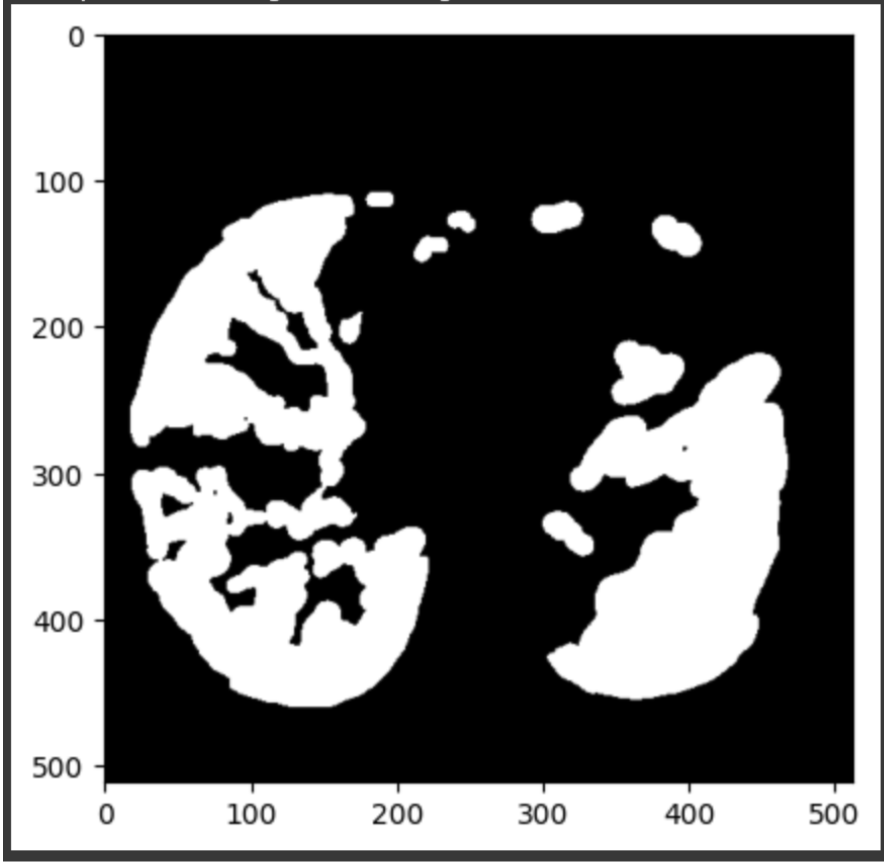



```
from keras.models import load_model  # TensorFlow is required for Keras to work
from PIL import Image, ImageOps  # Install pillow instead of PIL
import numpy as np

# Disable scientific notation for clarity
np.set_printoptions(suppress=True)

# Load the model
model =
load_model("/content/drive/MyDrive/CAPSTONE_PROJECT/converted_keras/keras_model.h5", compile=False)
```

```
# Load the labels
class_names =
open("/content/drive/MyDrive/CAPSTONE_PROJECT/converted_keras/labels.tx
t", "r").readlines()

# Create the array of the right shape to feed into the keras model
# The 'length' or number of images you can put into the array is
# determined by the first position in the shape tuple, in this case 1
data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)

# Replace this with the path to your image
image =
Image.open("/content/drive/MyDrive/CAPSTONE_PROJECT/bjorke_2.png").conv
ert("RGB")

# resizing the image to be at least 224x224 and then cropping from the
center
size = (224, 224)
image = ImageOps.fit(image, size, Image.Resampling.LANCZOS)

# turn the image into a numpy array
image_array = np.asarray(image)

# Normalize the image
normalized_image_array = (image_array.astype(np.float32) / 127.5) - 1

# Load the image into the array
data[0] = normalized_image_array

# Predicts the model
prediction = model.predict(data)
index = np.argmax(prediction)
class_name = class_names[index]
confidence_score = prediction[0][index]

# Print prediction and confidence score
print("Class:", class_name[2:], end="")
print("Confidence Score:", confidence_score)


from warnings import filterwarnings

import tensorflow as tf

from tensorflow import io

from tensorflow import image

from matplotlib import pyplot as plt
```

```
filterwarnings("ignore")

tf_img =
io.read_file("/content/drive/MyDrive/CAPSTONE_PROJECT/bjorke_2.png")

tf_img = image.decode_png(tf_img, channels=3)

print(tf_img.dtype)

plt.imshow(tf_img)

# plt.show()
```

```
1/1 [==============================] - 1s 900ms/step
Class: Masks
Confidence Score: 0.9999975
<dtype: 'uint8'>
<matplotlib.image.AxesImage at 0x7b2914e18880>
```



Conclusion

This report displays the work completed related towards the ST1 Capstone project for design, development, implementation, and deployment for the COVID-19 CT scan image classifier. First came the EDA and then the PDA for stage 2 using TeachableMachinewithGoogle and finally the deployment using Google Colab. As seen above, the confidence score of each image uploaded for classification was above 99% and very close to 100%. This means that this could be very useful for doctors for identifying and monitoring lungs that have been impacted by COVID-19.

References

Coronavirus (2023) *World Health Organization*. Available at: https://www.who.int/health-topics/coronavirus#tab=tab_1 (Accessed: 05 October 2023).

CT Scan (2022) *Mayo Clinic*. Available at: https://www.mayoclinic.org/tests-procedures/ct-scan/about/pac-20393675 (Accessed: 07 October 2023).

Salehi, M. *et al.* (2022) *Automated deep learning-based segmentation of COVID-19 lesions from chest computed tomography images*, *Polish journal of radiology*. Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9453472/ (Accessed: 18 October 2023).

What is covid-19? (2022) *healthdirect*. Available at: https://www.healthdirect.gov.au/covid-19/about#what-is (Accessed: 29 October 2023).

Journal

Week 8:

I was able to go through the assignment sheet and see the requirements of the task. I also was able to go through my dataset that was allocated to me. I watched the lecture to gain some more knowledge about the whole project.

Week 9:

I came up with the five questions that I wanted to answer while exploring my dataset that was allocated to me from Kaggle. This would set me up to progress with the rest of the project as I needed this to proceed with the EDA.

Week 10:

I was able to complete my EDA on Google Colab and start with my PDA through TeachableMachinewithGoogle. I prepared to start my implementation & deployment for stage 3 of the project.

Week 11:

I completed the implementation & deployment stage of the project by doing this on Google Colab and then began writing for the report part of the project. I was on good track with the project and had made good progress with plenty of time left.

Week 12:

I continued working on my report by following a template given. I also began to work on the presentation for the presentation/interview assessment of the project which would be in week 13 during the tutorial time.

Week 13:

I have completed my presentation and presented it in class. I was also very close to finishing the report as I had to add a few more things and the whole project would be complete.