National College of Ireland

Configuration Manual

MSc Research Project
M.Sc., in Cyber Security

Rithin krishna Dilipkumar
Student ID: 20199830

School of Computing
National College of Ireland

Supervisor: Mr. Imran Khan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: ……Rithin krishna Dilipkumar……

Student ID: ……20199830……

Programme: ……. M.Sc., Cyber Security…. **Year:** Sept 2021-Sept 2022

Module: …… **MSc Research Project**……

Supervisor: ……Mr. Imran Khan……

Submission Due Date: ……..15th August 2022……

Project Title: …… Medical Image Forgery Detection……

Word Count: ……748…… **Page Count**……10……

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: …… Rithin krishna Dilipkumar……

Date: ……15th August 2022……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

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

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

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Rithin krishna Dilipkumar
Student ID: 20199830

1 Introduction

The Configuration Manual offers particular information about the hardware, software, and tool requirements for implementing and carrying out the Research Project. It also outlines the step-by-step process for running the code to observe how it is implemented and what outcomes are produced.

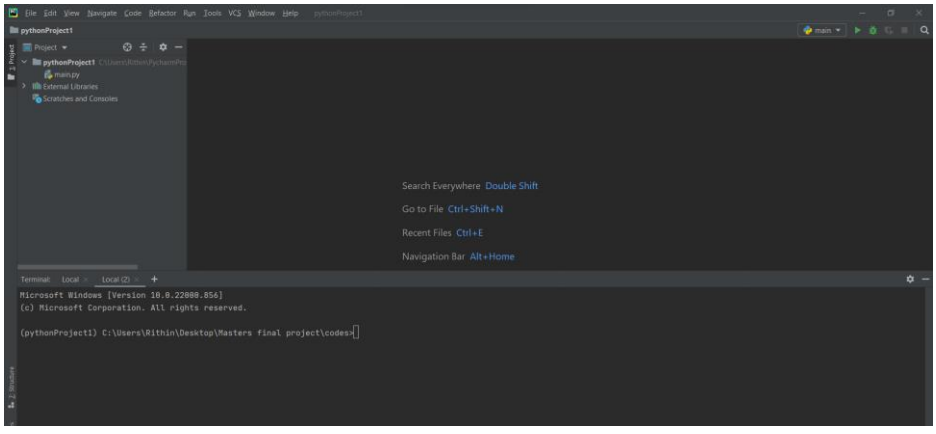
2 Environmental Setup

The following hardware and software specifications and configurations are used to accomplish the suggested solution:

- Operations System: Windows 10
- Operating System Type: 64-bit operating system, x64-based processor.
- Processor: Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz 2.59 GHz
- Memory: 16.0 GB
- Program Language: Python 3.7
- Environment: Jupyter Notebook 6.4.12
- IDE: PyCharm 2022.2

3 Installation of Tools

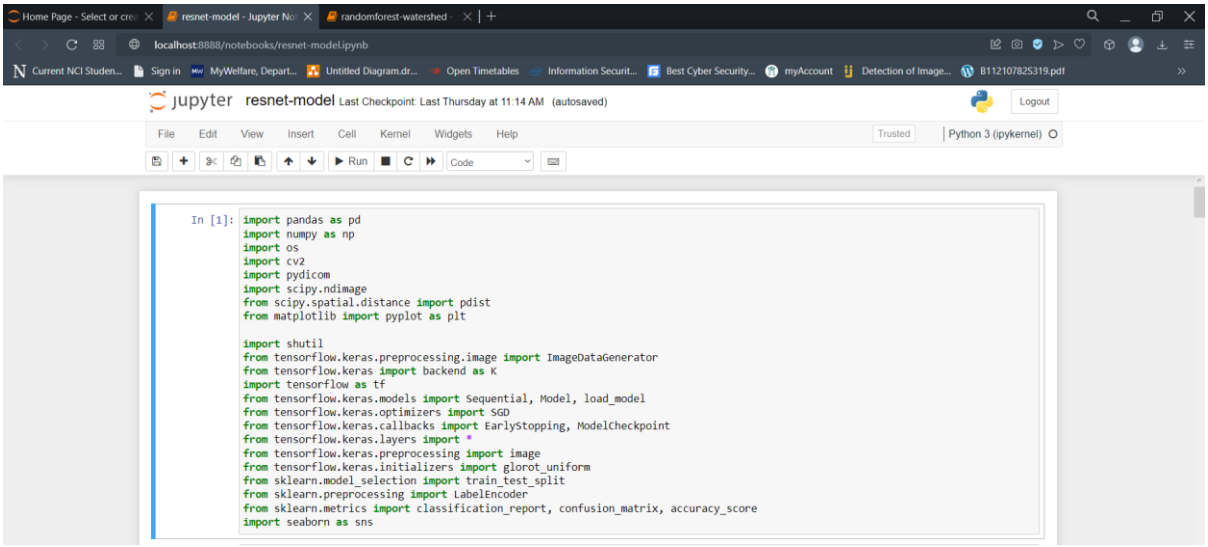
PyCharm 2022.2 suitable environment for effective Python, web, and data science development, PyCharm is a specialized Python Integrated Development Environment (IDE) that offers a wide range of crucial tools for Python developers and helps to successfully launch the required application and tools.



3.2 Jupyter Notebook 6.4.12

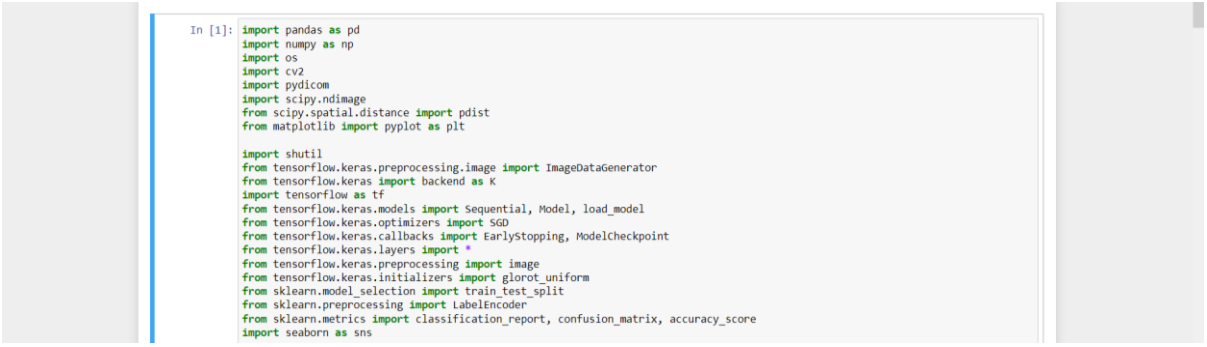
After installing PyCharm, we can use the terminal to start Jupyter Notebook and then navigate to the folder containing the code we wish to run.

Then, click the program's execution button. It will be added to Jupyter Notebook, as seen in the accompanying picture. Click "Run" to run the code and view the results after that.



4 Execution of Code for ResNet50 Model

- The other libraries (pandas, numpy, and cv2) necessary for putting the code into practice were installed.



- The dataset uses the DICOM format to save CT scans, which were then transformed into regular images by obtaining the necessary libraries.

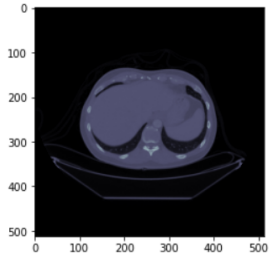
In [2]:
```
def load_dicom(path2scan_dir):
    dicom_folder = path2scan_dir
    dcms = os.listdir(dicom_folder)
    first_slice_data = pydicom.read_file(os.path.join(path2scan_dir,dcms[0]))
    first_slice = first_slice_data.pixel_array
    orientation = np.transpose(first_slice_data.ImageOrientationPatient) #zyx format
    spacing_xy = np.array(first_slice_data.PixelSpacing, dtype=float)
    spacing_z = np.float(first_slice_data.SliceThickness)
    spacing = np.array([spacing_z, spacing_xy[1], spacing_xy[0]]) #zyx format

    scan = np.zeros((len(dcms),first_slice.shape[0],first_slice.shape[1]))
    raw_slices=[]
    indexes = []
    for dcm in dcms:
        slice_data = pydicom.read_file(os.path.join(dicom_folder,dcm))
        slice_data.filename = dcm
        raw_slices.append(slice_data)
        indexes.append(float(slice_data.ImagePositionPatient[2]))
    indexes = np.array(indexes,dtype=float)

    raw_slices = [x for _, x in sorted(zip(indexes, raw_slices))]
    origin = np.array(raw_slices[0][0x00200032].value) #origin is assumed to be the image location of the first slice
    if origin is None:
        origin = np.zeros(3)
    else:
        origin = np.array([origin[2],origin[1],origin[0]]) #change from x,y,z to z,y,x

    for i, slice in enumerate(raw_slices):
        scan[i, :, :] = slice.pixel_array
    return scan, spacing, orientation, origin, raw_slices
```

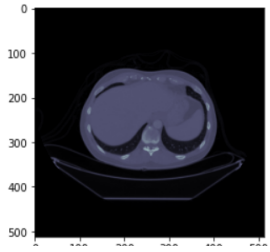
- The necessary libraries were imported, and a dataset of human lungs-related medical images was loaded into the model.

In [3]:
```
# loading a image from blind category
scan_uuid = 8038
scan, spacing, orientation, origin, raw_slices = load_dicom('../Dataset/CT_Scans/EXP1_blind/'+str(scan_uuid))

print('The CT scan has the dimensions of',scan.shape,'  (z,y,x)')
```

- Displaying the sample images which are loaded

In [4]:
```
# displaying the images
for slice_indx in range(50,55,1):
    plt.imshow(scan[slice_indx,:,:],cmap='bone',vmin=-1000,vmax=2000)
    plt.show()
```





- Loading the label files (TM and FM) and then loadings images according to the labels and displaying on how many images haven been loaded. Then then the images are merged and the output is displayed.

In [6]:
```
# few images from TM and FM
scan_uuids = [4142, 2838, 2320, 5614]
finalImagesFM = None
for scan_uuid in scan_uuids:
    FM_scan, spacing, orientation, origin, raw_slices = load_dicom('../Dataset/CT_Scans/EXP1_blind/'+str(scan_uuid))
    if finalImagesFM is not None:
        finalImagesFM = np.concatenate((finalImagesFM, FM_scan))
    else:
        finalImagesFM = FM_scan
```

```
c:\users\rithin\desktop\nci study\pythonproject1\lib\site-packages\ipykernel_launcher.py:8: DeprecationWarning: `np.float` is a
deprecated alias for the builtin `float`. To silence this warning, use `float` by itself. Doing this will not modify any behavi
or and is safe. If you specifically wanted the numpy scalar type, use `np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
```

In [7]:
```
finalImagesFM.shape
```

Out[7]:
```
(1033, 512, 512)
```

In [8]:
```
scan_uuids = [2190, 7507, 3025, 3099]
finalImagesTM = None
for scan_uuid in scan_uuids:
    TM_scan, spacing, orientation, origin, raw_slices = load_dicom('../Dataset/CT_Scans/EXP1_blind/'+str(scan_uuid))
    if finalImagesTM is not None:
        finalImagesTM = np.concatenate((finalImagesTM, TM_scan))
    else:
        finalImagesTM = TM_scan
```

In [9]:
```
finalImagesTM.shape
```

Out[9]:
```
(1046, 512, 512)
```

In [10]:
```
finalImages = np.concatenate((finalImagesFM, finalImagesTM))
finalLabels = ['fake'] * finalImagesFM.shape[0] + ['real'] * finalImagesTM.shape[0]
```

In [11]:
```
finalImages.shape, len(finalLabels)
```

Out[11]:
```
((2079, 512, 512), 2079)
```

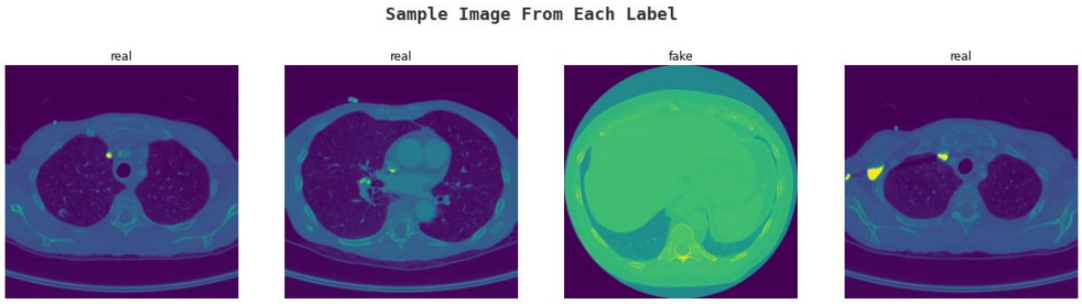
- Splitting the data in two portions, for training and testing respectively.

In [15]:
```
X_train, X_test, y_train, y_test = train_test_split(
    resized_masks,
    finalLabels,
    test_size=0.2,
    random_state=42,
)

X_train = np.array(X_train)
y_train = np.array(y_train)
X_test = np.array(X_test)
y_test = np.array(y_test)

print(('X_train:', X_train.shape))
print(('y_train:', y_train.shape))
print(('X_test:', X_test.shape))
print(('y_test:', y_test.shape))
```

```
('X_train:', (1663, 256, 256))
('y_train:', (1663,))
('X_test:', (416, 256, 256))
('y_test:', (416,))
```

- Sample of training images is getting displayed.

```
In [16]: # displaying training images
k = 0
fig, ax = plt.subplots(1, 4, figsize=(20, 20))
fig.text(s='Sample Image From Each Label',
         size=18,
         fontweight='bold',
         fontname='monospace',
         y=0.62,
         x=0.4,
         alpha=0.8)
for j in [0,1,409,385]:
    while True:
        ax[k].imshow(X_train[j])
        ax[k].set_title(y_train[j])
        ax[k].axis('off')
        k += 1
        break
```



- Defining the ResNet50 model

```
In [17]: def residual_block(X_start, filters, name, reduce=False, res_conv2d=False):
    nb_filters_1, nb_filters_2, nb_filters_3 = filters
    strides_1 = [2, 2] if reduce else [1, 1]

    X = Conv2D(filters=nb_filters_1,
               kernel_size=[1, 1],
               strides=strides_1,
               padding='same',
               name=name)(X_start)
    X = BatchNormalization()(X)  # default axis-1 is ok
    X = Activation('relu')(X)

    X = Conv2D(filters=nb_filters_2,
               kernel_size=[3, 3],
               strides=[1, 1],
               padding='same')(X)
    X = BatchNormalization()(X)
    X = Activation('relu')(X)

    X = Conv2D(filters=nb_filters_3,
               kernel_size=[1, 1],
               strides=[1, 1],
               padding='same')(X)
    X = BatchNormalization()(X)

    if res_conv2d:
        X_res = Conv2D(filters=nb_filters_3,
                       kernel_size=[1, 1],
                       strides=strides_1,
                       padding='same')(X_start)
        X_res = BatchNormalization()(X_res)
    else:
        X_res = X_start

    X = Add()([X, X_res])
```

- Summary of the defined model is displayed regarding the number of layers present in it and the data was encoded into numbers because neural network does not process strings. Due to these reasons the labels were encoded to numbers.

```
In [20]: model.summary()
```

```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 256, 256, 1  0           []
                                )]

 conv1 (Conv2D)                 (None, 128, 128, 64  3200        ['input_1[0][0]']
                                )

 bn_conv1 (BatchNormalization)  (None, 128, 128, 64  256         ['conv1[0][0]']
                                )

 activation (Activation)        (None, 128, 128, 64  0           ['bn_conv1[0][0]']
                                )

 max_pooling2d (MaxPooling2D)   (None, 63, 63, 64)   0           ['activation[0][0]']

 conv2_a (Conv2D)               (None, 63, 63, 64)   4160        ['max_pooling2d[0][0]']
```

```
In [21]: le = LabelEncoder()
y_train = le.fit_transform(y_train)
y_test = le.transform(y_test)
```

- The model is being trained with successful running 100 epoch.

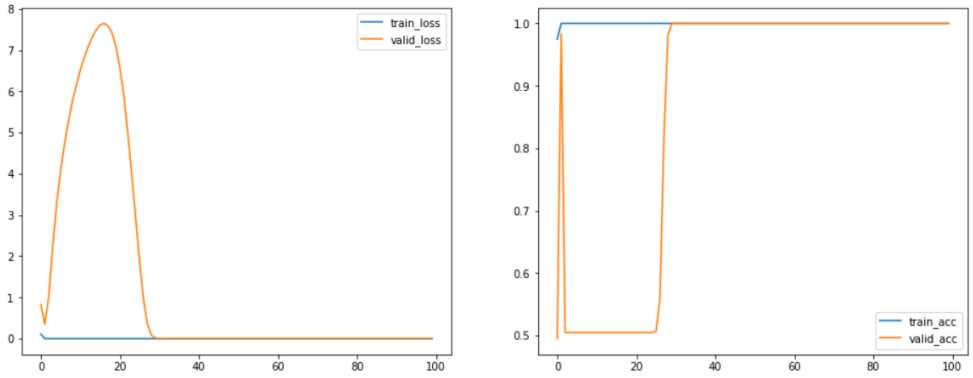
```
In [22]: hist = model.fit(x=X_train,
                         y=y_train,
                         batch_size=128,
                         epochs=100,
                         validation_data=(X_test, y_test),
                         verbose=2)
```

```
13/13 - 499s - loss: 5.6027e-07 - accuracy: 1.0000 - val_loss: 1.9056e-07 - val_accuracy: 1.0000 - 499s/epoch - 38s/step
Epoch 92/100
13/13 - 503s - loss: 7.5703e-07 - accuracy: 1.0000 - val_loss: 1.8598e-07 - val_accuracy: 1.0000 - 503s/epoch - 39s/step
Epoch 93/100
13/13 - 500s - loss: 4.6465e-07 - accuracy: 1.0000 - val_loss: 1.8340e-07 - val_accuracy: 1.0000 - 500s/epoch - 38s/step
Epoch 94/100
13/13 - 497s - loss: 2.5885e-07 - accuracy: 1.0000 - val_loss: 1.8168e-07 - val_accuracy: 1.0000 - 497s/epoch - 38s/step
Epoch 95/100
13/13 - 496s - loss: 5.3969e-07 - accuracy: 1.0000 - val_loss: 1.7738e-07 - val_accuracy: 1.0000 - 496s/epoch - 38s/step
Epoch 96/100
13/13 - 499s - loss: 6.7474e-07 - accuracy: 1.0000 - val_loss: 1.7165e-07 - val_accuracy: 1.0000 - 499s/epoch - 38s/step
Epoch 97/100
13/13 - 502s - loss: 4.1691e-07 - accuracy: 1.0000 - val_loss: 1.7079e-07 - val_accuracy: 1.0000 - 502s/epoch - 39s/step
Epoch 98/100
13/13 - 501s - loss: 5.8715e-07 - accuracy: 1.0000 - val_loss: 1.6850e-07 - val_accuracy: 1.0000 - 501s/epoch - 39s/step
Epoch 99/100
13/13 - 503s - loss: 1.0336e-06 - accuracy: 1.0000 - val_loss: 1.6620e-07 - val_accuracy: 1.0000 - 503s/epoch - 39s/step
Epoch 100/100
13/13 - 509s - loss: 4.8049e-07 - accuracy: 1.0000 - val_loss: 1.6076e-07 - val_accuracy: 1.0000 - 509s/epoch - 39s/step
```

➢ The performance outcome of the training is plotted and it is displayed as graph.

```
In [23]: fig, (ax1, ax2) = plt.subplots(nrows=1, ncols=2, figsize=[16, 6])
         ax1.plot(hist.history['loss'], label='train_loss')
         ax1.plot(hist.history['val_loss'], label='valid_loss')
         ax1.legend()
         ax2.plot(hist.history['accuracy'], label='train_acc')
         ax2.plot(hist.history['val_accuracy'], label='valid_acc')
         ax2.legend()
```

```
Out[23]: <matplotlib.legend.Legend at 0x2a49dd4db88>
```



➢ The prediction and the classification report of the model is displayed.

```
In [24]: y_pred = model.predict(X_test)

         print(('Prediction shape:', y_pred.shape))
         print(('Accuracy:', accuracy_score(y_test, np.argmax(y_pred, axis=1))))
```

```
13/13 [==============================] - 29s 2s/step
('Prediction shape:', (416, 2))
('Accuracy:', 1.0)
```

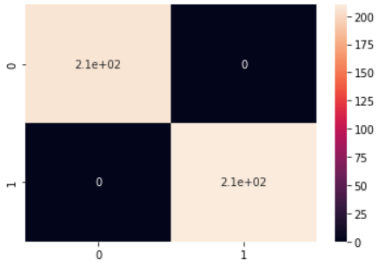
```
In [25]: print(classification_report(y_test, np.argmax(y_pred, axis=1)))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       206
           1       1.00      1.00      1.00       210

    accuracy                           1.00       416
   macro avg       1.00      1.00      1.00       416
weighted avg       1.00      1.00      1.00       416
```

➢ Toward the end as the final outcome the confusion matrix have been displayed.

```
In [26]: cm = confusion_matrix(y_test, np.argmax(y_pred, axis=1))
```

```
In [27]: sns.heatmap(cm, annot=True)
         plt.show()
```



References

[1] Jupyter Home. “Jupyter Nootebook”. [online]. Available at: https://jupyter.org. [Accessed on: 19th July 2022]

[2] Jet Brains. "PyCharm". [online]. Available at: https://www.jetbrains.com/pycharm/. [Accessed on: 19th July 2022]

……. END REPORT……