National College of Ireland

# Configuration Manual

MSc Research Project
Programme Name

Alexander Albert Irudayaraj
Student ID: 20172176

School of Computing
National College of Ireland

Supervisor: Dr. Cristina Hava Muntean

**National College of Ireland**

**MSc Project Submission Sheet**

**School of Computing**

| | |
|---|---|
| **Student Name:** | Alexander Albert Irudayaraj |
| **Student ID:** | 20172176 |
| **Programme:** | MSc in Data Analytics **Year:** 2021-2022 |
| **Module:** | MSc Research Project |
| **Lecturer:** | Dr. Cristina Hava Muntean |
| **Submission Due Date:** | 31-01-2022 |
| **Project Title:** | Kidney Stone Detection using Deep Learning Methodologies |
| **Word Count:** | 1347 **Page Count:** 7 |

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:** .....

**Date:** 31-01-2022

**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

Alexander Albert Irudayaraj
Student ID: 20172176

# 1 Introduction

This document gives a detailed description of the steps to be followed to replicate the results of the research that have been enlisted in the project report. The document covers the different tools required to simulate the experiments and also provides a detailed explanation of the steps to be followed for code execution.

# 2 Environment Setup

## 2.1 Google Collab

The research involves the modelling of 4 specific neural networks to examine their suitability to perform detection of kidney stones in CT scan images. The Python programming language has been utilised to achieve the model construction. For executing the code, the Google Collab integrated development environment has been used. The completed code has been submitted in the form of. ipynb notebooks. These Python notebooks can be imported into Google collab as follows:

1. Go to the URL of the Google Collab[1] and sign in with Google account of choice.
2. Go to File-> Open notebook and click on Upload as shown in Figure 1 and upload the. ipynb file of choice.

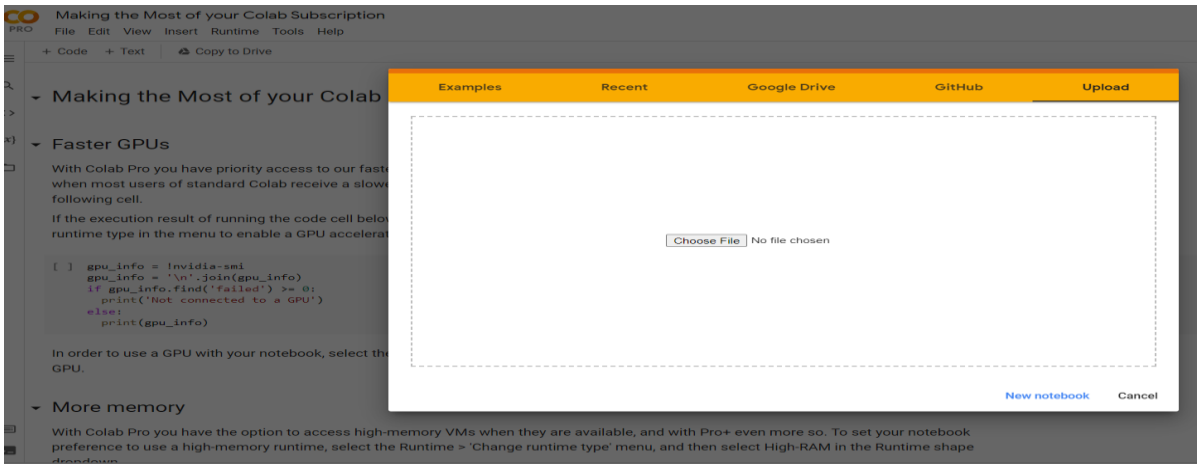



Figure 1: Google Collab uploading .pynb files from local system

[1] https://colab.research.google.com/?utm_source=scs-index

3. The .ipynb file of selection will be opened in Google Collab. Next click on the Connect button in the top right corner of the notebook. This will connect to a runtime in the cloud.
4. Once connection is successful, click on the RAM, Disk icon near the top right corner and click on Change Runtime Type. Make the same selections in the dialog box that appears as shown in the Figure 2. Please note that Runtime-shape will possess the value High-RAM after upgrading to Collab Pro.

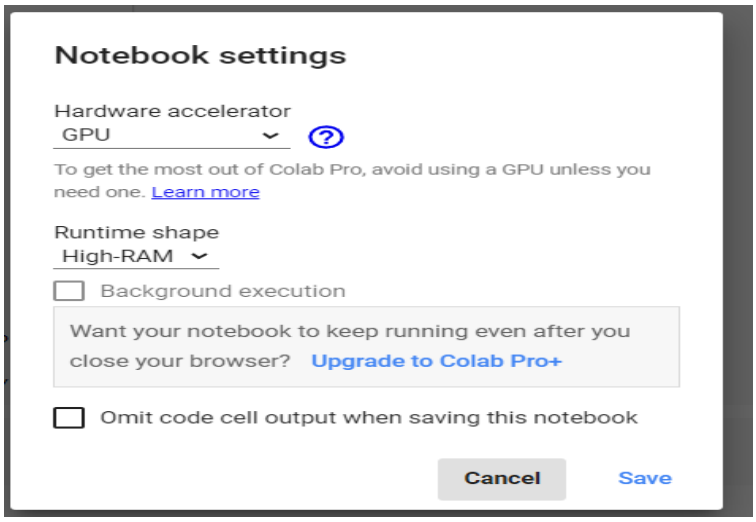



Figure 2: Runtime Selection in Jupyter Notebook

5. Code Execution of each cell in the Notebook can be performed by clicking on the Run cell button just to the left of each cell.

## 2.2 Google Drive

Since Google Collab is a cloud-based environment for program execution, the dataset also has to be placed in a cloud storage environment called Google drive to access from the Collab environment. The following steps are to be performed to store the unmodified dataset and the cropped dataset in the drive.

1. Go to the URL of the Google Drive Homepage[2].
2. Sign in using the same Google account through which Google Collab was accessed.
3. Once logged in, click on the New button and choose Folder upload.
4. Upload the Final_Dataset and Fully_Edited_Dataset folders [after creating the folders as per Section 3]. These folders should contain the unmodified dataset and cropped dataset respectively.

[2] https://www.google.com/drive/

5. To access these folders that are now part of your drive, the code block shown in Figure 3 is to be executed in Google Collaboratory. This process is called the drive mount and is essential for data access from the drive.

```
[ ]  from google.colab import drive
     drive.mount('/content/drive',force_remount=True)

     Mounted at /content/drive
```

Figure 3: Mounting the drive for accessing the data from the drive

# 3 Data Preparation

The dataset to be used for the modelling process is to be downloaded from the URL[3] through the Download ZIP option in Github. The dataset will contain two folders namely Train and Test. These two folders are to be moved to a new folder called Final_Dataset in the following hierarchical order: Final_Dataset->Second Dataset-> Kidney_stone_detection-main->Dataset and this folder is to be uploaded to Google drive as described in Section 2.2. The Final_Dataset thus now contains the unmodified data. Alternatively, the Final_Dataset folder can be downloaded directly from the Onedrive link [4]

For creating the cropped dataset, each image in the dataset is to be cropped and in case of an image with kidney stone, just the kidney with the unwanted particles is to be cropped, excluding the other regions of the CT scan image. This operation is to be performed using the Windows Photos application. This cropped dataset is to be placed in the Fully_Edited_Dataset folder and it must be uploaded to the drive. The dataset containing the cropped images and used for the experiments has been uploaded to Onedrive[5] for replication of results.

# 4 Code Execution Steps

The codebase consists of two distinct folders Code_for_cropped_dataset, Code_for_cropped_dataset pertaining to code for original dataset and cropped dataset. Each of these folders will contain 4 files corresponding to the 4 neural network algorithms and each file is appropriately named. Each file will contain the following steps, such as Importing Dependent Libraries, Data Access from Drive and Data Split, Data Augmentation, Model Import and Model Modification, Model Compilation and Results generation, Model Evaluation. Since these steps have been followed in a uniform way for all code files, they are

[3] https://github.com/yildirimozal/Kidney_stone_detection
[4] https://studentncirl-my.sharepoint.com/:f:/g/personal/x20172176_student_ncirl_ie/Ej-KidqjvglDrdhVv1rhlyIBT422OYbc6aT7VqZDGDKvRw?e=CdbKZM
[5] https://studentncirl-my.sharepoint.com/:f:/g/personal/x20172176_student_ncirl_ie/Ej-KidqjvglDrdhVv1rhlyIBT422OYbc6aT7VqZDGDKvRw?e=CdbKZM

explained with one code file [KidneyStone_ResNet_Proper_Unmodified_Dataset.ipynb] as the example.

## 4.1 Importing Dependent Libraries

The neural networks that have been modelled in the research have used in-built libraries such as Keras to achieve functionalities like base model importing. The code to import dependent libraries is shown in the Figure 4.

```
[ ]  # Commented out IPython magic to ensure Python compatibility.
     import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import skimage.io
     import tensorflow
     import tqdm
     import glob

     from tqdm import tqdm

     from skimage.io import imread, imshow
     from skimage.transform import resize

     from tensorflow.keras.preprocessing.image import ImageDataGenerator
     from tensorflow.keras.applications.vgg16 import VGG16
     from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import InputLayer, Dense, Flatten, BatchNormalization, Dropout, Activation
     from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
     from tensorflow.keras.preprocessing.image import load_img, img_to_array
```

Figure 4: Importing of necessary libraries

## 4.2 Data Access from Drive and Data Split

The datasets that are available in the drive are to be accessed using the flow_from_directory API and data split for training and validation are specified using ImageDataGenerator function. The code for performing these operations are shown in Figure 5.

```
[ ]  train_path = '/content/drive/MyDrive/Final_Dataset/Second Dataset/Kidney_stone_detection-main/Dataset/Train'
     test_path = '/content/drive/MyDrive/Final_Dataset/Second Dataset/Kidney_stone_detection-main/Dataset/Test'

[ ]  train_datagen = ImageDataGenerator(rescale= 1./255,validation_split=0.2)

[ ]  test_datagen = ImageDataGenerator(rescale= 1./255)

[ ]  validation_datagen=ImageDataGenerator(rescale= 1./255)

[ ]  batch_size = 256

[ ]  train_generator = train_datagen.flow_from_directory(
             train_path, target_size= (224,224),
             batch_size = batch_size,
             color_mode= "rgb",
             class_mode= "categorical",
             subset='training',
             )

     test_generator = test_datagen.flow_from_directory(
             test_path,
             target_size= (224,224),
             batch_size = batch_size,
             color_mode= "rgb",
             class_mode= "categorical")

     validation_generator=train_datagen.flow_from_directory(
             train_path,
             target_size= (224,224),
             batch_size = batch_size,
             color_mode= "rgb",
             class_mode= "categorical",

             subset='validation',
```

Figure 5: Accessing Data from drive and performing data split

## 4.3 Data Augmentation

The code to be executed for performing the augmentation techniques is shown in Figure 6. The code also shows the different augmentation parameters to be specified to perform desired augmentation.

```
data_augmentation = tf.keras.Sequential([
  tf.keras.layers.RandomFlip('horizontal'),
  tf.keras.layers.RandomRotation(0.2)

])
```

Figure 6: Code for performing Data Augmentation

## 4.4 Model Import and Model Specification

The base neural network models are to be imported from keras libraries. An example for importing the base ResNet model is shown in Figure 7. The base model is then modified by the addition of some extra layers to make the model suitable for classification using the kidney stone detection as shown in Figure 8.

```
[ ]
    from tensorflow.keras.applications.resnet50 import ResNet50

    base_model = ResNet50(input_shape=(224,224,3),
                          include_top=False,
                          weights="imagenet")

    Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
    94773248/94765736 [==============================] - 4s 0us/step
    94781440/94765736 [==============================] - 4s 0us/step
```

Figure 7 : Model Import from Keras API

```
[ ]  resnet_model=Sequential()
     resnet_model.add(base_model)
     resnet_model.add(Dropout(0.2))
     resnet_model.add(Flatten())
     resnet_model.add(BatchNormalization())
     resnet_model.add(Dense(1024,kernel_initializer='he_uniform'))
     resnet_model.add(BatchNormalization())
     resnet_model.add(Activation('relu'))
     resnet_model.add(Dropout(0.2))
     resnet_model.add(Dense(1024,kernel_initializer='he_uniform'))
     resnet_model.add(BatchNormalization())
     resnet_model.add(Activation('relu'))
     resnet_model.add(Dropout(0.2))
     resnet_model.add(Dense(2,activation='sigmoid'))
```

Figure 8 : Extra layers added to the base model.

## 4.5 Model Compilation and Results Generation

The model is to be compiled using the code shown in the Figure 9. The compile function is used for the model compilation and the fit function is used to generate the appropriate results.

```
]  #OPT    = tensorflow.keras.optimizers.Adam(lr=0.0001)
   base_learning_rate = 0.0001

   resnet_model.compile(loss='binary_crossentropy',
                metrics=['accuracy','AUC','Precision','Recall'],
                optimizer=tf.keras.optimizers.Adam(learning_rate=base_learning_rate))
```

```
]  model_history=resnet_model.fit(train_generator,
                          validation_data=validation_generator,
                          epochs = initial_epochs)
```

Figure 9: Model Compilation and Results generation

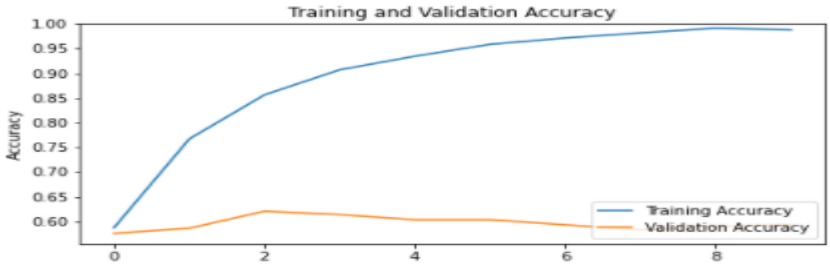
## 4.6 Model Evaluation

The model can be evaluated using the code shown in Figure 10 and Figure 11. The evaluation metrics used in this research are Accuracy, Precision and Recall. The training and Validation Accuracy are then plotted to effectively examine the classification performance of each deep learning neural network.

```
[ ]  acc = model_history.history['accuracy']
     val_acc = model_history.history['val_accuracy']
     loss = model_history.history['loss']
     val_loss = model_history.history['val_loss']
```

```
[ ]  plt.figure(figsize=(8, 8))
     plt.subplot(2, 1, 1)
     plt.plot(acc, label='Training Accuracy')
     plt.plot(val_acc, label='Validation Accuracy')
     plt.legend(loc='lower right')
     plt.ylabel('Accuracy')
     plt.ylim([min(plt.ylim()),1])
     plt.title('Training and Validation Accuracy')
```

```
Text(0.5, 1.0, 'Training and Validation Accuracy')
```




```
[ ]  plt.subplot(2, 1, 2)
     plt.plot(loss, label='Training Loss')
     plt.plot(val_loss, label='Validation Loss')
     plt.legend(loc='upper right')
     plt.ylabel('Cross Entropy')
     plt.ylim([0,1.0])
     plt.title('Training and Validation Loss')
     plt.xlabel('epoch')
     plt.show()
```

Figure 10: Model Evaluation-Part 1

```
[ ]  plt.subplot(2, 1, 2)
     plt.plot(loss, label='Training Loss')
     plt.plot(val_loss, label='Validation Loss')
     plt.legend(loc='upper right')
     plt.ylabel('Cross Entropy')
     plt.ylim([0,1.0])
     plt.title('Training and Validation Loss')
     plt.xlabel('epoch')
     plt.show()
```

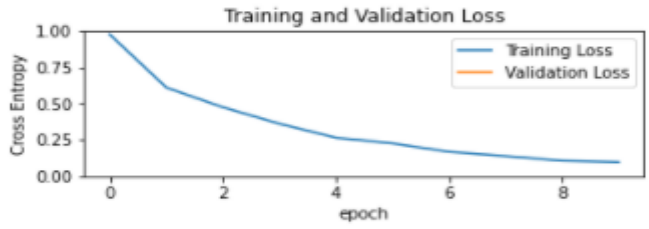



Figure 11: Model Evaluation-Part 2