

Configuration Manual

MSc Research Project
Data Analytics

Srivenkateswara Rao Vatti

Student ID: x18181104

School of Computing
National College of Ireland

Supervisor: Rashmi Guptha

National College of Ireland
Project Submission Sheet
School of Computing



Student Name:	Srivenkateswara Rao Vatti
Student ID:	x18181104
Programme:	Data Analytics
Year:	2020
Module:	MSc Research Project
Supervisor:	Rashmi Guptha
Submission Due Date:	17/08/2020
Project Title:	Configuration Manual
Word Count:	XXX
Page Count:	15

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:	Srivenkateswara Rao Vatti
Date:	27th September 2020

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	<input type="checkbox"/>
Attach a Moodle submission receipt of the online project submission , to each project (including multiple copies).	<input type="checkbox"/>
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.	<input type="checkbox"/>

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

Srivenkateswara Rao Vatti
x18181104

Contents

1	Introduction	2
2	Application Environment	2
2.1	Hardware	2
2.2	Software and tools	2
2.2.1	Google Colab	3
2.2.2	Google Drive	3
2.2.3	Python	4
2.3	Major python Libraries used across artefacts	4
3	Application overview	4
3.1	Structure of the data	4
3.2	Authentication and authorization	5
3.3	Data pre-processing and transformation	8
3.4	CNN training	10
3.4.1	Pre-trained Block	11
3.4.2	Common block	12
3.4.3	Saving the model	12
3.4.4	Run the model	13
3.5	CNN testing	13

3.5.1	Plotting the graphs	14
3.5.2	Printing the metrics	15
4	Future Work	15

1 Introduction

The basic aim of the current research is to apply a set of convolutional neural network(CNN) models implemented with transfer learning to detect and classify test the accuracy and other metrics. The pre-trained models considered as part of this analysis is VGG16, VGG19, Inception V3, ResNet50 and Xception. Initial blocks of these pre-trained models are extracted and integrated with a common block of additional convolutional layers. Apart from these models, a CNN model is implemented from scratch with a set of convolutional layers and a comparative analysis is carried out in terms of performance and efficiency of the models.

2 Application Environment

2.1 Hardware

A laptop with the configuration shown in the Table 1 is utilized for this research.

Table 1: Hardware

Feature	Value
Processor	Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz 1.80 GHz
RAM	8 GB
System Type	64-bit Operating System, x64-based processor
Hard disk	1 TB

2.2 Software and tools

The research is based on Google Colab platform along with Google drive for data storage. The high level design of the platform is shown in Figure 1

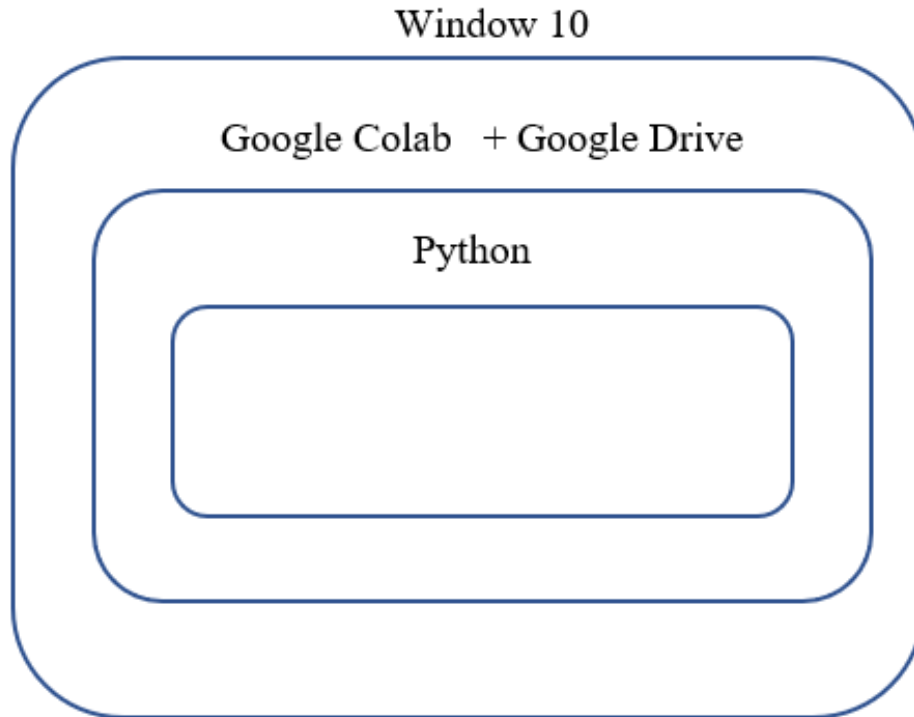


Figure 1: Platform design

2.2.1 Google Colab

This is a web-based platform implemented by Google for implementing machine learning and deep learning models. Google Colab is built upon Jupiter notebook on which CNN architectures can be run with python as the programming language. TensorFlow and Keras are the primary libraries used in this research and these packages will be loaded onto the Colab environment for model implementation. TPU runtime available free of cost in Colab and is utilized since the size of the data set used is around 8GB. Colab is integrated with GitHub for code configuration purposes.

2.2.2 Google Drive

The data set acquired for this analysis is uploaded into google drive and then loaded into Colab for further analysis. The primary reason for choosing google drive as a data repository for this analysis is that the size of the data set is around 8 GB and the storing and processing of this huge data are tedious and local machines like laptops or desktops are not an ideal choice for this purpose.

2.2.3 Python

Python is the programming language opted for implementing CNN models as part of this research and advanced packages such as TensorFlow and Keras for deep learning purposes can be imported through python. Apart from these, python also comes with data visualization libraries such as matplotlib for implementing graphs and charts.

2.3 Major python Libraries used across artefacts

The Table 2 explains various python libraries used as part of this analysis.

Table 2: Libraries in Python used

Library	Usage
NumPy	This library is for doing mathematical operations.
CV2	It is for reading an image from a physical path
matplotlib	Plotting graphs
OS	File operations like creating, moving
pydrive	Authentication and authorization between Google Colab and Google Drive.
TensorFlow	Machine learning model development
Keras	Machine learning model development including pre-trained models.

3 Application overview

3.1 Structure of the data

The dataset used for this research is around 8 GB in size and is acquired from the respective data source and stored in Google drive for further analysis. A link to the google drive folder which contains the data can be found in ¹. The screen shown in Figure 2 explains the folder structure of the tomato pest data. As per the figure, the data is segregated into three folders of images and the other three folders of the respective labels of the images. The folders GREENPATROL_DB_V1.0.zip, GREENPATROL_DB_V2.0.zip and GREENPATROL_DB_V3.0.zip contain the images. The folders LABELS_V1.0.zip, LABELS_V2.0.zip and LABELS_V3.0.zip contain the labels in the form of annotation files which are in XML format. The reason for splitting the image data into three separate folders is to avoid timeout errors while loading images onto google Colab environment.

¹Data Link: <https://drive.google.com/drive/u/1/folders/1aXb25Wp9sG7mL6AQgYY9ElrGcERi95JU>








Name ↓	Owner	Last modified	File size
 predefined_classes.txt	me	Jun 11, 2020 me	71 bytes
 LABELS_V3.0.zip	me	Jun 27, 2020 me	711 KB
 LABELS_V2.0.zip	me	Jul 25, 2020 me	884 KB
 LABELS_V1.0.zip	me	Jun 21, 2020 me	1 MB
 GREENPATROL_DB_V3.0.zip	me	Jun 26, 2020 me	2 GB
 GREENPATROL_DB_V2.0.zip	me	Jun 26, 2020 me	2 GB
 GREENPATROL_DB_V1.0.zip	me	Jun 21, 2020 me	4 GB

Figure 2: Data structure

3.2 Authentication and authorization

The data is fetched from google drive onto google Colab for further image processing activities such as cropping, resizing and extracting bounding boxes. This data retrieval from google Colab needs to be authenticated when accessing google drive. The following code will be executed for this purpose.

```
import os
from os import listdir
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials
import numpy as np
auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)
```

A link will be prompted as shown in the Figure 3



Figure 3: Authentication and authorization step 2

Select the google account that is associated with google drive in which the data resides as shown in Figure 4.

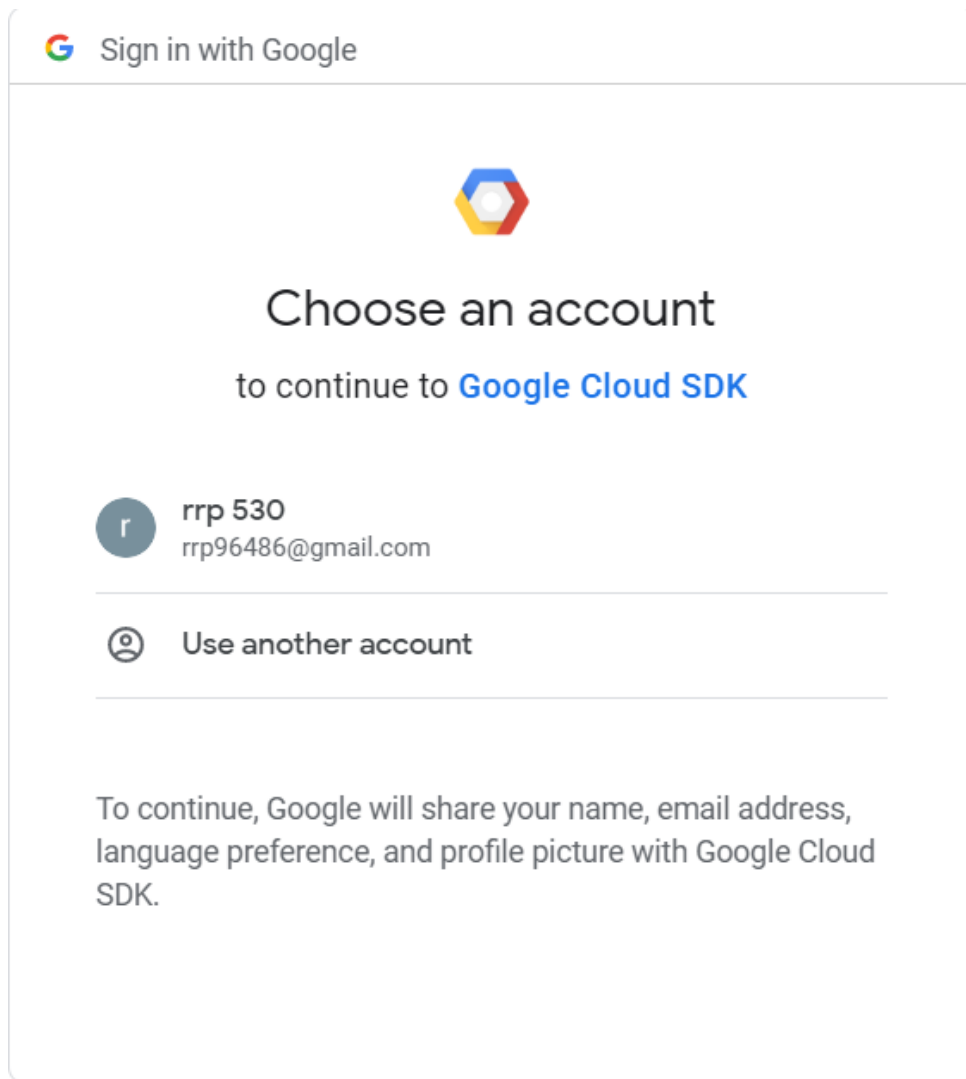


Figure 4: Authentication and authorization step 3









Click on Allow as shown in Figure 5.



Google Cloud SDK wants to access your Google Account

 rrp96486@gmail.com

This will allow **Google Cloud SDK** to:

-  See, edit, create, and delete all of your Google Drive files 
-  View and manage your data across Google Cloud Platform services 
-  View and manage your Google Compute Engine resources 
-  View and manage your applications deployed on Google App Engine 

Make sure you trust Google Cloud SDK

You may be sharing sensitive info with this site or app. Learn about how Google Cloud SDK will handle your data by reviewing its terms of service and privacy policies. You can always see or remove access in your [Google Account](#).

[Learn about the risks](#)

Cancel

Allow

Figure 5: Authentication and authorization step 4

The screen will be redirected to a page where we can copy a pass code and need to be pasted in the text box shown in Figure 3 so that the google Colab platform will be given access to fetch the data from google drive.



Sign in

Please copy this code, switch to your application and paste it there:

```
4/2wE3WJD_5edpYLUw0_S24SHGE0xZg-  
B0cNVBr11ABzto6IEptM_W_e0
```



Figure 6: Authentication and authorization step 5

3.3 Data pre-processing and transformation

There are two supporting files used in this analysis apart from primary model files and these files named as CreatePestImageData.py and SupportingFunctions.py. These files can be uploaded and instances of these files can be created on Colab environment by executing the following two blocks of code.

```
from google.colab import files  
files.upload()
```

```
import CreatePestImageData as Imagedataset  
import SupportingFunctions as support
```

There should be a nested folder structure created before executing the rest of the code. There should a parent folder with the name “Pests” and two nested folders underneath it called “Images” and “Labels”. Please find the Figure 7 for reference.

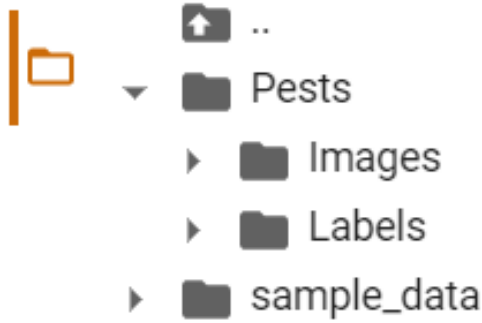


Figure 7: Data pre-processing step 2

Configuration Parameters: The Table 3 describes the methods and parameters passed those methods.

Table 3: Methods and parameters overview

Method Name	Purpose	Parameters
DownLoadFiles(folderId, drive)	To download the data files from google drive to Colab	The “ folderId ” is the unique identifier of the folder in Google Drive in which data resides and can be found in URL of the folder in the google drive. The “ drive ” is the object created as per the code shown in Figure ??
Unzip(source_Path, dest_Path)	To extract the images and annotation files from the zip files that are downloaded in the previous step. Note: This method needs to be called for all the folders of images and labels	source_Path is the path of the zip file of images or labels which are downloaded to google Colab dest_Path is the target folder which is either “Images” or “Labels” those are located in the “Pests” parent folder
fileCount(directory)	To get the number of files residing a folder.	directory - The path of the folder

<pre>prepare_dataset(path , class_object , image_Size, imagePath, ,test_size)</pre>	<p>To prepare training and validation data sets in the form of NumPy arrays by cropping the images as per the XML-based annotation files.</p>	<p>Path - the path of annotation files class_object - class labels of the image data image_Size - the size of the image that needs to be cropped imagePath - the path of image files test_size - the size of the test data set that is created.</p>
---	---	--

Please find the below code block shown for the usage of above methods in Google Colab environment.

```
support.DownloadFiles('1aXb25Wp9sG7mL6AQgYY9ElrGcERi95JU',
drive)
support.Unzip('/content/GREENPATROLDV3.0.zip',
'/content/Pests/Images')
support.fileCount('/content/Pests/Images')
dataSet = Imagedataset.createImageDataset()
class_object = {0 : 'egg_wf' ,
1 : 'egg_bt' , 2 : 'egg_ta' , 3 : 'wf' ,
4 : 'bt' , 5 :
'tomato'}
dataSet.prepare_dataset('/content/Pests/Labels' ,
class_object ,32, '/content/Pests/Images' , test_size=11241 )
(trainY , trainX) , (testY , testX ) =
(dataSet.trainY , dataSet.trainX ) ,
(dataSet.testY , dataSet.testX )
```

3.4 CNN training

The portion discussed so far is common to all the models implemented as part of this research. Once the training and testing data is prepared, both can be passed to models as part of the training. The following Table 4 describes different models and the corresponding files in the code submitted.

Table 4: Code files corresponding to models implemented

Model	Code File
CNN	Pest_Classifier_CNN.ipynb
VGG16	Pest_Classifier_VGG16.ipynb
VGG19	Pest_Classifier_VGG19.ipynb
ResNet50	Pest_Classifier_ResNet50.ipynb
InceptionV3	Pest_Classifier_InceptionV3.ipynb
Xception	Pest_Classifier_Xception.ipynb

These models are implemented with two major blocks namely pre-trained block and a common block of additional convolutional layers.

3.4.1 Pre-trained Block

The code block shown below is the one for downloading the InceptionV3 model from Keras API.

```
import tensorflow as tf
from keras import models
from keras.applications.inception_v3 import InceptionV3
InceptionV3 = InceptionV3(input_shape=input_shape,
    include_top=False, pooling='avg', weights='imagenet')
InceptionV3.summary()
```

The code block shown below is for extraction of “conv2d_3” block for the InceptionV3 model.

```
from keras.models import Model
layerName = 'conv2d_3'
mymodel = Model(inputs= InceptionV3.input ,
    outputs=InceptionV3.get_layer(layerName).output)
mymodel.summary()
```

3.4.2 Common block

The “conv2d_3” extracted in the previous step is then integrated to a common block as shown in the below code blocks to implement the final architecture of the model which will be trained in the next stage of the implementation.

```
from keras import applications
from keras.models import Model
from keras.layers import Conv2D,
MaxPooling2D, GlobalAveragePooling2D
from keras.layers import Dense,
Dropout, Activation, Flatten
from keras.layers.normalization import
    BatchNormalization
import matplotlib.pyplot as plt
```

```
model= models.Sequential()
model.add(mymodel)
model.add(Conv2D(16, (3, 3), activation='relu',
padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(32, (3, 3), activation='relu',
padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(GlobalAveragePooling2D())
model.add(Dense(64, activation='relu'))
model.add(BatchNormalization())
model.add(Dense(6, activation='softmax'))
model.layers[0].trainable = True

# compile the model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

3.4.3 Saving the model

The model implemented can be trained using the code block shown below. This is useful to retrain the model again and again till satisfactory results are obtained.

```

from tensorflow.keras.utils import to_categorical
trainY = to_categorical(trainY)
testY = to_categorical(testY)
tf.config.experimental_run_functions_eagerly(True)
filepath="/content/drive/My Drive/Models/InceptionV3
/weights-{epoch:02d}-{accuracy:.3f}.hdf5"
from keras.callbacks import ModelCheckpoint
checkpoint = ModelCheckpoint(filepath,
    monitor='accuracy',
    verbose=1, mode='max', save_best_only=True)

callbacks_list = [checkpoint]

```

3.4.4 Run the model

The model implemented can be executed using the code block shown below

```

model.fit(trainX, trainY, verbose=1, epochs=50,
    validation_data=(testX, testY), callbacks=[checkpoint])

```

3.5 CNN testing

The implemented models can be validated with predictions on testing data and necessary graphs, key metrics can be calculated through the methods mentioned in the TABLE.

Table 5: Validation methods

Method Name	Purpose	Parameters
plot_image(i, predictions_array, true_labels, images)	This method is to display actual vs predicted results of the test data	i – looping variable predictions_array – The array of prediction outcomes by model true_labels – Actual classes of the images images - Images those are validated against true labels

PrintMetrics(testX, testY, model)	This method is for printing important metrics such as accuracy, precision, recall and F1 score with the test data	testX – NumPy array of training data testY – NumPy array of validation data model – the instance of the model.
GenerateGraphs(acc, val_acc, loss, val_loss)	This method is to draw graphs such as loss vs accuracy for both testing and training data sets.	acc – training accuracy val_acc – validation accuracy loss – training loss val_loss - validation loss

3.5.1 Plotting the graphs

The following code blocks are used for plotting the actual versus predicted labels against a random 9 images of validation data.

```
def plot_image(i, predictions_array, true_labels, images):
    predictions_array, true_label,
    img = predictions_array[i], true_labels[i], images[i]
    plt.grid(False)
    plt.xticks([])
    plt.yticks([])

    plt.imshow(img[...,0], cmap=plt.cm.binary)

    predicted_label = np.argmax(predictions_array)
    if predicted_label == true_label:
        color = 'blue'
    else:
        color = 'red'

    plt.xlabel("{} {:2.0f}% ({})" .format
(class_object[predicted_label],
100*np.max(predictions_array),
class_object[true_label]),
color=color)
```



```

num_rows = 3
num_cols = 3
num_images = num_rows*num_cols
plt.figure(figsize=(2*num_cols, 2*num_rows))
test_images = testX[:num_images]
predictions = model.predict(testX_Pred)
for i in range(num_images):
    plt.subplot(num_rows, num_cols, i+1)
    plot_image(i, predictions, testY_Pred, testX_Pred)

plt.tight_layout()
plt.show()

```

3.5.2 Printing the metrics

Various metrics such as accuracy, precision, recall and F1 score can be printed using the code block shown below

```

support.PrintMetrics(testX_Pred, testY_Pred, model)

```

4 Future Work

In the future, the research can be extended in multiple dimensions. A different set of CNN architectures can be implemented to examine the performance of the models in terms of classifying the pests on tomato plants. The model implemented in the current research can be integrated into a scouting robot so that it can detect the pests in real-time conditions. Also, the models can be deployed onto smartphones so that the leaves of the tomato plants can be scanned and the percentage of different pests that are formed can be identified on the spot.

References