National College of Ireland

Configuration Manual

MSc Research Project
MSc Data Analytics

Suchal Pote
Student ID: X21121397

School of Computing
National College of Ireland

Supervisor: Prof Athanasios Staikopoulos

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Suchal Pote
Student ID:	X21121397
Programme:	MSc Data Analytics **Year:** 2022
Module:	MSc Research Project
Lecturer:	Prof Athanasios Staikopoulos
Submission Due Date:	15th December 2022
Project Title:	Waste Classification by Using Deep Learning and Transfer Learning
Word Count:	1032 **Page Count:** 8

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: Suchal Pote

Date: 15th December 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

Suchal Pote
Student ID: X21121397

1 Introduction

This research project is carried out to classify common types of waste by using the deep learning models and the transfer learning method. Information on the data set, software, system specifications, hardware specifications, techniques, library packages, as well as programs used for developing the models in the research is included in this configuration manual. All the steps that have performed in the research is also provided in this manual. The screenshots of the codes have also provided for better understanding.

2 System Specifications

Details of the system configurations utilized in this project's implementation are given in this section.

2.1 System Hardware Configuration

Operating System:	Windows 11
Processor:	11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz
Installed RAM:	8.00 GB (7.73 GB usable)
System Type:	64-bit operating system, x64-based processor
Hard Disk:	1 TB

2.2 Software Configuration

The specifications of the used softwares and its requirements are provided in this section. In the research paper pre-trained deep learning models including Vgg16, InceptionV3, Xception, and DenseNet201 have been implemented on the Jupyter notebook version 6.4.12 with python version 3.9.7. The Jupyter notebook can be assessed by Anaconda[1] Nevigator.

- Jupyter notebook 6.4.12
- python version 3.9.7.
- Anaconda Nevigator

2.3 Libraries and packages used

This section provides libraries and packages used in the research project. The fig. 1 shows the libraries and packages used to in Vgg16 model. Fig.2 showcasing the libraries and packages used to in InceptionV3 model. Fig.3 showcasing the libraries and packages used to in Xception model. The libraries and packages used to in DenseNet20 model is presented in fig.4.

[1] https://www.anaconda.com/products/distribution

- Tensorflow version 2.9.1
- Keras
- Numpy
- Glob
- Matplotlib
- Os
- Skimage
- pylab

```
import tensorflow as tf
print(tf.__version__)
```

```
2.9.1
```

```
from platform import python_version


print("Current Python Version-", python_version())
```

```
Current Python Version- 3.9.7
```

```
from keras.layers import Input, Lambda, Dense, Flatten
from keras.models import Model
from keras.applications.vgg16 import VGG16
from keras.applications.vgg16 import preprocess_input
from keras.applications.vgg19 import VGG19
from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
import numpy as np
from glob import glob
import matplotlib.pyplot as plt
import os
from tensorflow.keras.layers import BatchNormalization
from keras.layers import Dense, GlobalAveragePooling2D, Dropout
from keras.layers import Dropout
from keras import regularizers
```

Fig. 1 Vgg16 libraries

```
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.applications.inception_v3 import preprocess_input
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_img
from tensorflow.keras.models import Sequential
import numpy as np
from glob import glob
import matplotlib.pyplot as plt
from tensorflow.keras.layers import BatchNormalization
from keras.layers import Dense, GlobalAveragePooling2D, Dropout
from keras.layers import Dropout
from keras import regularizers
```

Fig. 2 InceptionV3 libraries

```
import tensorflow as tf
print(tf.__version__)
```

```
2.9.1
```

```
from keras.layers import Input, Lambda, Dense, Flatten
from keras.models import Model
from tensorflow.keras.applications.xception import Xception
from keras.applications.vgg16 import preprocess_input
from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
import numpy as np
from tensorflow import keras
from glob import glob
import matplotlib.pyplot as plt
import os
from keras.layers import Dense, GlobalAveragePooling2D, Dropout
from keras.layers import Dropout
from keras import regularizers
```

Fig. 3 Xception libraries

```
import tensorflow as tf
print(tf.__version__)
```

```
2.9.1
```

```
from keras.layers import Input, Lambda, Dense, Flatten
from keras.models import Model
from keras.applications.densenet import DenseNet201
from keras.applications.densenet import preprocess_input
from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
import numpy as np
from glob import glob
import matplotlib.pyplot as plt
from keras.layers import Dense, GlobalAveragePooling2D, Dropout
from keras.layers import Dropout
import os
from tensorflow import keras
```

Fig. 4 DenseNet201 libraries

3 Project Implementation

3.1 Data collection

The dataset used to classify different types of waste in this research is TrashNet[2] dataset. The dataset is openly accessible on GitHub repository for research purpose. This dataset can be used for commercial purpose, private purpose, for modification as well as distribution as mentioned in license file. No agreement license or any kind of form is required to fill to access the dataset. The repository contains link of google drive where dataset file is present. The dataset file only contains no other file than images. The collected dataset having images that are not labelled and uneven for each category. The dataset contains only 6 classes of

[2] https://github.com/garythung/trashnet

waste. So, in order to extend the study, dataset was modified, and 4 more categories have been added in the dataset[3]. The dataset has already been divided into train and test set.

3.2 Data Preparation and Transformation

The dataset used in this study was already divided into training and test sets in ratio of 50:50. Some of the images were found to be corrupted, which are eliminated in this stage. The input images are resized to 224x224 as per all model requirements as shown in fig.5 and fed them to data transformation stage. In the data transformation stage, some data augmentation techniques are performed in order to prevent from overfitting of the models and to make the model generalize. The ImageDataGenerator class has been used to conduct data augmentation steps as shown in fig.6. In this, the input normalization is performed by rescaling the input image on both train and test sets to dynamically resize pixel values to the range [0,1] from the range of [0,255] (Vo *et al.*, 2019). Due to image rescaling, the images will contribute to the total loss more evenly. Along with rescaling, image shearing is carried out with a range of 0.2. Additionally, image zooming with a range of 0.2 and random horizontal flipping are also performed. The input images after were resized to 224x224. Only image rescaling is performed on the test set.

```
# set image size
IMAGE_SIZE = [224, 224]

# adding preprocessing layer to the front of VGG
Myvgg = VGG16(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)
```

```
# set image size
IMAGE_SIZE = [224, 224]

# adding preprocessing layer to the front of VGG
inception = InceptionV3(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)
```

```
# set image size
IMAGE_SIZE = [224, 224]

# adding preprocessing layer to the front of VGG
xception = Xception(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)
```

```
# set image size
IMAGE_SIZE = [224, 224]

# adding preprocessing layer to the front of VGG
DenseNet201 = DenseNet201(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)
```

Fig 5 Image resizing in all models

```
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)

test_datagen = ImageDataGenerator(rescale = 1./255)
```

Fig 6 Data Augmentation on train and set for all models

[3] https://studentncirl-my.sharepoint.com/:f:/g/personal/x21121397_student_ncirl_ie/EqjJTe8vElhEiTn7qrPu6ioBMpWMLLE9BNuHve7LqkHLNg?e=hNLN5m

3.3 Load the models

All the models implemented in the research are pretrained models. The weights of the pre-trained models are set to the ImageNet dataset and top layer have excluded as the transfer learning method is supposed to utilize (Aral *et al.*, 2018) as showcased in fig.7.

```
# adding preprocessing layer to the front of VGG
Myvgg = VGG16(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)

# do not train current weights
for layer in Myvgg.layers:
  layer.trainable = False

# adding preprocessing layer to the front of VGG
inception = InceptionV3(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)

# do not train current weights
for layer in inception.layers:
  layer.trainable = False

# adding preprocessing layer to the front of VGG
xception = Xception(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)

# do not train current weights
for layer in xception.layers:
  layer.trainable = False

# adding preprocessing layer to the front of VGG
DenseNet201 = DenseNet201(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)

# do not train current weights
for layer in DenseNet201.layers:
  layer.trainable = False
```

Fig 7 loading of all models

3.4 Creation of model's architecture

All the models have been loaded and batch normalization, as well as drop-out layers, have been added to models. The structure of the vgg16 created in the research is presented in fig.8.

```
# preparing model structure
x = Flatten(name='flatten')(Myvgg.output)

x = Dense(4096, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.2)(x)

x = Dense(4096, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.2)(x)

predictions= Dense(len(dir), activation='softmax', kernel_initializer='random_uniform', bias_initializer='random_uniform',
bias_regularizer=regularizers.l2(0.01), name='predictions')(x)

# create a model object
model = Model(inputs=Myvgg.input, outputs=predictions)
```

Fig 8 Preparation of Vgg16 structure

The structure of the InceptionV3 created in the research is Shown in fig.9.

```
# preparing model structure

x = inception.output
x = GlobalAveragePooling2D()(x)

# let's add a fully-connected layer
x = Dense(4096, activation='relu')(x)

# add Dropout regularizer
x = Dropout(0.8)(x)
x = Dense(len(dir), activation='softmax', kernel_initializer='random_uniform', bias_initializer='random_uniform',
bias_regularizer=regularizers.l2(0.01), name='prediction')(x)


# create a model object
model = Model(inputs=inception.input, outputs=x)
```

Fig.9 Preparation of InceptionV3 structure

The structure of the Xception created in the research is presented in fig.10.

```
# preparing model structure

x = xception.output
x = GlobalAveragePooling2D()(x)
x = keras.layers.Dropout(0.2)(x)
x = Dense(len(dir), activation='softmax', name='predictions')(x)

# create a model object
model = Model(inputs=xception.input, outputs=x)
```

Fig.10 Preparation of Xception structure

The structure of the DenseNet201 created in the research is presented in fig.11.

```
# preparing model structure

x = DenseNet201.output
x = GlobalAveragePooling2D()(x)
x = keras.layers.Dropout(0.2)(x)
x = Dense(len(dir), activation='softmax', name='predictions')(x)

# create a model object
model = Model(inputs=DenseNet201.input, outputs=x)
```

Fig.11 Preparation of DenseNet201 structure

3.5 Modelling

All the models were compiled with the help of the 'adam' optimizer and 'categorical_crossentropy' loss function. The accuracy has been selected as a metrics. The fig.12 shows the model compilation which is same for all models.

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

Fig.12 Model compilation

The training and test sets were loaded for training in the batch size of 32. The image size is set to 224x224. The fig.13 represented code for the same.

```
training_set = train_datagen.flow_from_directory("C:\\Users\\SUCHAL\\Desktop\\Research in
Computing\\Thesis\\Codes\\Xception\\Dataset\\train",
                                                 target_size = (224, 224),
                                                 batch_size = 32,
                                                 class_mode = 'categorical')

test_set = test_datagen.flow_from_directory("C:\\Users\\SUCHAL\\Desktop\\Research in
Computing\\Thesis\\Codes\\Xception\\Dataset\\test",
                                            target_size = (224, 224),
                                            batch_size = 32,
                                            class_mode = 'categorical')
```

Fig.13 Train and Test set loading

The models have been trained with 15 epochs. The training of the models with 'earlystop' method is shown in fig.14.

```
# fit the model
from keras.callbacks import EarlyStopping
early_stopping = EarlyStopping(patience=10)
r = model.fit(training_set, validation_data=test_set, epochs=15, steps_per_epoch=len(training_set),
              validation_steps=len(test_set), callbacks=[early_stopping])
```

Fig.14 Model training

4 Model Evaluation

The accuracy and loss graphs for the training and test sets are both plotted to evaluate performance. A code for graph of training and test accuracy and loss is shown in Fig 15.

```
# loss
plt.plot(r.history['loss'], label='train loss')
plt.plot(r.history['val_loss'], label='val loss')
plt.legend()
plt.show()
plt.savefig('LossVal_loss')
```

```
# accuracies
plt.plot(r.history['accuracy'], label='train acc')
plt.plot(r.history['val_accuracy'], label='val acc')
plt.legend()
plt.show()
plt.savefig('AccVal_acc')
```

Fig.15 Plotting loss and accuracy graph

5 Save the models

All the models were saved in computer for future work if needed. The code to save the model is shown in fig.16. It is same for all models.

```
#Save the trained model
from tensorflow.keras.models import load_model
model.save('Model_Name.h5')
```

Fig.16 Save the model

References

Aral, R.A. *et al.* (2018) 'Classification of TrashNet Dataset Based on Deep Learning Models', in *2018 IEEE International Conference on Big Data (Big Data)*, pp. 2058–2062. Available at: https://doi.org/10.1109/BigData.2018.8622212.

Vo, A.H. *et al.* (2019) 'A novel framework for trash classification using deep transfer learning', *IEEE Access*, 7, pp. 178631–178639.