

Configuration Manual

MSc Research Project
Data Analytics

Parag Suresh Joshi
Student ID: x19212071

School of Computing
National College of Ireland

Supervisor: Noel Cosgrave

National College of Ireland
Project Submission Sheet
School of Computing



Student Name:	Parag Suresh Joshi
Student ID:	x19212071
Programme:	Data Analytics
Year:	2021
Module:	MSc Research Project
Supervisor:	Noel Cosgrave
Submission Due Date:	16/12/2021
Project Title:	Configuration Manual
Word Count:	1790
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:	
Date:	30th January 2022

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

Parag Suresh Joshi
x19212071

1 Introduction

The configuration manual provides details for the research and its implementation on the source machine. Next Section provides details of the hardware configuration of the source machine. An equivalent or better hardware is expected for effective research replication. Subsequent Section gives information on the Software requirements needed prior to research code execution. The last section gives details about the relevant project configurations needed in the source code for execution of research code.

2 Hardware Details

The research project code was executed on a standalone machine with the below hardware configuration.

Table 1: Machine Configuration

Feature	Config
Operating System	Windows 10 Home (64-bit)
System Memory	24 GB
Processor	Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz
Graphics Card	NVIDIA GeForce GTX 960M (2 GB Memory)
Disk Space	1 TB

The machine with above config was able to handle operations for a dataset of 400 videos. However, GPU Memory issues were observed for a dataset of size 5K videos and hence, it is recommended to use a better GPU machine with higher GPU Memory.

It is also recommended to have a minimum of 100GB of storage free. The DFDC 5K video dataset is a zip file of size 40 GB and would temporarily need double the storage size for extraction of the videos from the zip file. Moreover, the research dumps processed data into a numpy file for quicker processing in next iteration(s) whose size varies from 200MB to 3GB.

3 Software Requirements

The source code for the research was implemented in python. Python can be downloaded for the respective Operating Systems from its official website¹. Python also provides an

¹Python downloads: <https://www.python.org/downloads/>

installation and configuration guide².

Microsoft Visual Studio Code was the preferred IDE for the source code execution. Visual Studio Code for the respective Operating system can be downloaded from its official website³.

Python version 3.9.0 was used for compiling and executing the source code. Below table gives details about the libraries that are required for the research code.

Table 2: Python Library Details

Library	Version	Installation command
cv2	4.5.3.56	pip install opencv-python
decord	0.6.0	pip install decord
keras	2.6.0	pip install keras
keras-tuner	1.1.0	pip install keras-tuner
keras-video-generators	1.0.14	pip install keras-video-generators
matplotlib	3.4.3	pip install matplotlib
numpy	1.19.5	pip install numpy
pandas	1.3.4	pip install pandas
tensorflow-gpu	2.6.0	pip install tensorflow ⁴
tensorflow-Probability	0.14.1	pip install tensorflow-probability
sklearn	1.0	pip install scikit-learn

4 Project Configuration

This section describes the Project artefact information required for successful source code execution. Below subsection gives information on the source of dataset used for the research. Next subsection provide information on the directory structure required for the project. The final subsection provides information on all the variables that are configurable in the source code.

4.1 Dataset Details

Deepfake Detection Challenge (DFDC) Dataset was used for this research. The DFDC Dataset has 2 versions, 5K Videos and 124K videos, which can be downloaded from official site⁵. For development purposes, DFDC also provides a demo dataset of 400 videos which can be downloaded from kaggle⁶.

For this research, both, the development version (400 Videos) and the official version (5K videos) have been used. After downloading the dataset, extract the contents in any folder. The folder can then be set appropriately in the code config variables.

²Python Setup and Usage: <https://docs.python.org/3/using/index.html>

³VS Code downloads: <https://code.visualstudio.com/Download>

⁵DFDC Dataset: <https://ai.facebook.com/datasets/dfdc/>

⁶DFDC Development Dataset: <https://www.kaggle.com/c/deepfake-detection-challenge/data>

4.2 Folder Config

Figure 1 shows the folder structure hierarchy for this research.

Name	Date modified	Type
Code	12/14/2021 12:53 PM	File folder
Data	12/13/2021 3:37 PM	File folder
test_videos	10/15/2021 11:46 AM	File folder
train_sample_videos	10/15/2021 12:02 PM	File folder

Figure 1: Directory Structure

- The Source code file 'BNN_Code.py' is located inside folder 'Code'.
- The folders 'test_videos' and 'train_sample_videos' contain the DFDC dataset of 400 videos.
- DFDC dataset consisting of 5K videos has been extracted inside 'Data' folder.

After extracting datasets in their respective folders, make changes in source code file. The below code snippet provides information on the configuration variables for the same.

```
1 *****
2 # Config
3 *****
4 CODE_PATH = os.path.dirname(__file__)
5
6 boolisPreviewData = True
7
8 if( boolisPreviewData ):
9     #5K Videos Preview Dataset
10    TRAIN_DATA_PATH= os.path.join(CODE_PATH,"../Data/dfdc_preview_set/")
11    TEST_DATA_PATH= os.path.join(CODE_PATH,"../Data/dfdc_preview_set/")
12    TRAIN_INFO_PATH =
13    ↪ os.path.join(CODE_PATH,"../Data/dfdc_preview_set/dataset.json")
14    TEST_INFO_PATH =
15    ↪ os.path.join(CODE_PATH,"../Data/dfdc_preview_set/dataset.json")
16    MODEL_NAME = "Seq_Model_5000" # Model files would be created under
17    ↪ this folder name
18
19    TRAIN_FEATURE_FILE = "arrFeatures_5000.npy"
20    TRAIN_MASK_FILE = "arrMask_5000.npy"
21    TRAIN_LABEL_FILE = "arrLabels_5000.npy"
22    TEST_FEATURE_FILE = "arrFeatures_Test_5000.npy"
23    TEST_MASK_FILE = "arrMask_Test_5000.npy"
24    TEST_LABEL_FILE = "arrLabels_Test_5000.npy"
25 else:
```

```

23  #400 Videos Demo Dataset
24  TRAIN_DATA_PATH= os.path.join(CODE_PATH,"../train_sample_videos/")
25  TEST_DATA_PATH= os.path.join(CODE_PATH,"../test_videos/")
26  TRAIN_INFO_PATH =
    ↪ os.path.join(CODE_PATH,"../train_sample_videos/metadata.json")
27  TEST_INFO_PATH =
    ↪ os.path.join(CODE_PATH,"../test_videos/metadata.json")
28  MODEL_NAME = "Seq_Model_400" # Model files would be created under
    ↪ this folder name
29
30  TRAIN_FEATURE_FILE = "arrFeatures_400.npy"
31  TRAIN_MASK_FILE = "arrMask_400.npy"
32  TRAIN_LABEL_FILE = "arrLabels_400.npy"
33  TEST_FEATURE_FILE = "arrFeatures_Test_400.npy"
34  TEST_MASK_FILE = "arrMask_Test_400.npy"
35  TEST_LABEL_FILE = "arrLabels_Test_400.npy"

```

The variable 'boolIsPreviewData' controls the dataset to be used for each operation. If True, Preview Data set with 5K variables will be used and if False, development dataset with 400 videos would be used. Make any folder path changes as applicable inside the appropriate variables. The variables have been described below.

- The variable 'CODE_PATH' is the absolute folder location of the source code file. (No Changes required in this)
- 'TRAIN_DATA_PATH' is the relative folder location of the Training Dataset
- 'TEST_DATA_PATH' is the relative folder location of the Test Dataset
- 'TRAIN_INFO_PATH' is the relative folder location of the metadata config file for training
- 'TEST_INFO_PATH' is the relative folder location of the metadata config file for testing

4.3 Source Code Config

1. The source code execution is controlled by the booleans shown in the below code snippet.

```

1  # Booleans to control flow of the code
2  boolPrintFileInfo = False #Toggle to enable/disable printing of
    ↪ file resolution information
3  boolPlotGraphs = False #Toggle to plot graphs for EDA
4  boolRebuildFrame = False #Toggle to rebuild training set frame
    ↪ data
5  boolReBuildModel = False #Toggle to rebuild model
6  boolRebuildTestFrames = False #Toggle to rebuild testing set
    ↪ frame data.

```

- 'boolPrintFileInfo' is used to enable/disable resolution information stats for video files. This flag is redundant for now and was only used to check file metadata during development.
- 'boolPlotGraphs' is used to enable/disable graph plots. False implies that graphs will not be plotted. This flag can be False since plots are only needed for EDA.
- 'boolRebuildFrame' is used to enable/disable reprocessing of frames from videos and feature extraction on the Training dataset. For value True, it will reprocess the frames and perform feature extraction and then dump the feature extraction output to a numpy file and then send the output to the next function. For value False, it will simply read the numpy file and pass the datastructure to the next function. The numpy file will be saved in the source code folder as per the values specified in the variables 'TRAIN_FEATURE_FILE', 'TRAIN_MASK_FILE' and 'TRAIN_LABEL_FILE' shown in Figure ???. Set the boolean to True for first execution, and False then onwards, as long as no frame related configs have been modified. The frame related configs have been described in the subsequent image.
- 'boolReBuildModel' is used to rebuild the model and test its performance after any model specific changes have been done. The Value True indicates model will be rebuilt and the config will then be stored in source code folder. The value False indicates model will be read from the stored location.
- 'boolRebuildTestFrames' is used to enable/disable reprocessing of frames and feature extraction for Test dataset. The behavior is same as mentioned for 'boolRebuildFrame'.

2. Parameters shown in the below code snippet are used to tweak and improve model performance.

```

1  SIZE = 299
2  INPUT_SIZE = (SIZE,SIZE)
3  CHANNELS = 3
4  INPUT_SHAPE = (SIZE,SIZE,CHANNELS)
5  SEQUENCE_SIZE = 80
6  NUM_FEATURES = 2048
7  EPOCH_COUNT = 150
8  REGULARIZER = 0.001
9
10 BNN_ITERATIONS = 15

```

- 'SIZE' indicates the Size * Size resolution into which the videos will be resized before feature extraction activity.
- 'SEQUENCE_SIZE' indicates the frame count that will be processed for each video
- 'EPOCH_COUNT' indicates the maximum number of epochs that will be executed during model fit activity
- 'REGULARIZER' indicates the alpha value for regularization

- 'BNN_ITERATIONS' indicates the count of networks what will be used for BNN simulation
3. The below code snippet in the source code file toggles the CPU/GPU version for tensorflow

```
os.environ["CUDA_VISIBLE_DEVICES"] = "-1"
```

For scenarios where GPU runs out of memory, the above code can be commented out to execute the tensorflow processing in CPU mode instead of GPU mode.

4. The Below code snippet in the source code configures the Feature extraction model. Changes can be applied wherever applicable.

```
1 objFeatureExtractor = tf.keras.applications.InceptionV3(
2     include_top=False, #Top layer is not the last layer
3     weights="imagenet", #pretrained weights
4     input_tensor=None, #Inputs not shared with other ANNs
5     input_shape=INPUT_SHAPE,
6     pooling="max", #global max pooling will be used
7     #classes=1000, #Optional, will be read only if
8     → 'include_top' is true
9     #classifier_activation="softmax", #Optional, will be
10    → read only if 'include_top' is true
11 )
```

5. The Below code snippet in the source code configures the Sequence Processor model. Changes can be applied wherever applicable.

```
1 #Uncomment below 2 lines to enable code execution using CuDNNGRU
2 → libraries for faster GPU code execution
3     #objGRU = tf.compat.v1.keras.layers.CuDNNGRU(1024,
4     → return_sequences=True ,
5     → kernel_regularizer=l1(REGULARIZER),
6     → recurrent_regularizer=l1(REGULARIZER),
7     → bias_regularizer=l1(REGULARIZER) )(
8     → objFrameFeaturesInput )
9     #objGRU = tf.compat.v1.keras.layers.CuDNNGRU(1024)(objGRU)
10
11
12
13
14
15
16 objGRU = keras.layers.GRU(512, return_sequences=True ,
17 → kernel_regularizer=l1(REGULARIZER),
18 → recurrent_regularizer=l1(REGULARIZER),
19 → bias_regularizer=l1(REGULARIZER) )(
20 → objFrameFeaturesInput, mask=objInputMask )
21 objGRU = keras.layers.GRU(512)(objGRU)
22
23
24 objGRU = keras.layers.Dropout(0.3)(objGRU) #Dropout layer to
25 → handle overfitting.
```



```

10
11
12     #objGRU = keras.layers.Dense(256,
13     ↪ activation="relu")(objGRU)
14
15     objGRU = tfp.layers.DenseFlipout(256, activation="relu"
16     ↪ )(objGRU) #256
17
18     #objOutput = keras.layers.Dense(len(ObjVocab),
19     ↪ activation="softmax")(objGRU)
20
21     objOutput = tfp.layers.DenseFlipout(len(ObjVocab),
22     ↪ activation="softmax")(objGRU)
23
24     self.objSequenceModel = keras.Model([objFrameFeaturesInput,
25     ↪ objInputMask], objOutput)
26
27     objOptimizer = keras.optimizers.Adam(learning_rate=0.00001)
28     ↪ #Manually Set starting value for learning rate

```

5 Artefact Details

Figure 2 shows the artefacts required for the research. The Zip files for both the datasets are of considerable sizes (500MB and 3GB) and hence, these files would be excluded from the list of artefacts. Only the single source file would be eligible for submission.




Name	Date modified	Type	Size
 BNN_Code.py	12/15/2021 9:29 AM	Python File	37 KB
 Data_400.zip	12/15/2021 9:51 AM	ZIP File	518,352 KB
 Data_5000.zip	12/15/2021 9:56 AM	ZIP File	3,052,677 KB

Figure 2: Artefact Details

The artefacts of the zip file however can be generated through code.

Figure 3 shows the contents of the zip file for DFDC demo dataset model. The contents of the zip file for DFDC preview dataset is exact same.

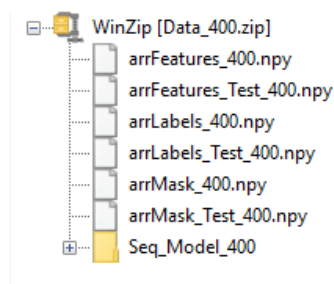


Figure 3: Zip file Details

- Setting the boolean 'boolRebuildFrame' as true in the source code will generate the arrFeatures, arrMask and arrLabels numpy files for the respective models.
- Similarly, setting the boolean 'boolRebuildTestFrames' as true in the source code will generate the arrFeatures_Test, arrMask_Test and arrLabels_Test numpy files.
- Setting the boolean 'boolReBuildModel' as true in the source code will re-train and store the created model at the specified config location.