National College of Ireland

Configuration Manual

MSc Research Project
MSc. Data Analytics

Prashant Digambar Waghela
Student ID: x20207786

School of Computing
National College of Ireland

Supervisor: Prof. Vladimir Milosavljevic

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: PRASHANT DIGAMBAR WAGHELA

Student ID: X20207786

Programme: MSc. Data Analytics **Year:** 2021-22

Module: MSc. Research Project

Lecturer: Prof. VLADIMIR MILOSAVLJEVIC

Submission Due Date: 15th August 2022

Project Title: MULTIMODAL FAKE NEWS AND TAMPERED IMAGE DETECTION USING TRANSFORMER AND CNN-BASED ALGORITHMS

Word Count: 2014 **Page Count:** 11

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: Prashant Digambar Waghela

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

Prashant Digambar Waghela
Student ID: x20207786

1 Introduction

The project configuration manual provides step-by-step instructions that researchers need to follow to successfully implement this project. It provides information about various system specification requirements along with the necessary steps that need to be remembered before running all the scripts.

2 System Specification

The first section covers the hardware, software, and library specification details on which this research project was implemented.

2.1 Hardware Used for the Project Implementation

CPU	8-core CPU (Apple M1 Chip)
GPU	Integrated 8-core GPU with 2.6 teraflops of throughput
RAM	16GB of LPDDR4X-4266 MHz SDRAM
SSD	256GB
Operating System	macOS

2.2 Libraries Used

Libraries Required	Pandas, NumPy, Keras, TensorFlow, Matplotlib, Sci-Kit Learn

2.3 Software Requirements

Software Required	Google Colab, Jupyter Notebook, Anaconda Navigator

3 Data Source

The dataset used in this project was private in nature and proper permission was taken from Sharma, D.K. and Garg, S., (2021) wherein GitHub access was provided to download the data. An ethics declaration form for the same has already been submitted to the academic institution.

4 Image Data Preparation

Note: The 'InitialDataClean_ImageFolderCreation.ipynb' and 'Image Data_Preprocessing.ipynb' needs to be strictly run on the local Jupyter Notebook environment.

Important Code Snippets in 'InitialDataClean_ImageFolderCreation.ipynb'

This is the first python notebook that needs to be run to download the images in the local folder.

```
#Read the '.tsv' file from the local system and convert it into a pandas
data frame

import pandas as pd
df =
pd.read_table('/Users/prashantwaghela/Desktop/FakeNewsDataset/multimodal_on
ly_samples/multimodal_data.tsv')
print(df.head())

#Dropping unwanted columns

df=df.drop(['author',
'clean_title','created_utc','hasImage','id','linked_submission_id','num
_comments','score','subreddit','upvote_ratio','3_way_label','6_way_labe
l'], axis=1)

#Column renaming, removal of N/A rows, and index resetting

df.columns = ['domain', 'img_url', 'news_title', 'is_fake']
df=df.dropna()
df=df.reset_index()
```

The above 3 codes were needed for the initial cleaning of the data frame. The below code is a custom script to download images through URLs and store them in the local system within train and test folders.

```
#Image Folder creation by URL checking and storing them in the local
system

import urllib.request
arr_remove=[]
def download_image(i,url, file_path):
    file_name = 'image-{}.jpg'.format(i)
```

```
    full_path = '{}{}'.format(file_path, file_name)
    try:
        urllib.request.urlretrieve(url, full_path)
        print('{} saved.'.format(file_name))
    except urllib.error.HTTPError as e:
        urllib.error.HTTPError == 'HTTP Error 404: Not Found'
        arr_remove.append(i)
        i+=1
        print('{}'.format(e))
    return None

FileName='finaldata_file.csv'

urls=pd.read_csv(FileName)
print(urls)

for i, url in enumerate(urls.values):
    if((i <= 0.8*len(df)) & (df.iat[i,4] == 0.0)):
        FilePath=
'/Users/prashantwaghela/Desktop/FakeNewsDataset/multimodal_only_samples
/images/train/fake/'
    elif((i <= 0.8*len(df)) & (df.iat[i,4] == 1.0)):
        FilePath=
'/Users/prashantwaghela/Desktop/FakeNewsDataset/multimodal_only_samples
/images/train/true/'
    elif((i > 0.8*len(df)) & (df.iat[i,4] == 0.0)):
        FilePath=
'/Users/prashantwaghela/Desktop/FakeNewsDataset/multimodal_only_samples
/images/test/fake/'
    elif((i > 0.8*len(df)) & (df.iat[i,4] == 1.0)):
        FilePath=
'/Users/prashantwaghela/Desktop/FakeNewsDataset/multimodal_only_samples
/images/test/true/'
    download_image(i, url[3], FilePath)
```

5 Step 2

Important Code Snippets in ‘Image Data_Preprocessing.ipynb’

```
from PIL import Image
Image.MAX_IMAGE_PIXELS = 1000000000
import os
import pandas as pd

#Image pre-processing by Resizing and storing in the local system
```

```
files_testt=os.listdir("/Users/prashantwaghela/Desktop/FakeNewsDataset/
multimodal_only_samples/images/test/true/")
files_testf=os.listdir("/Users/prashantwaghela/Desktop/FakeNewsDataset/
multimodal_only_samples/images/test/fake/")
files_traint=os.listdir("/Users/prashantwaghela/Desktop/FakeNewsDataset
/multimodal_only_samples/images/train/true/")
files_trainf=os.listdir("/Users/prashantwaghela/Desktop/FakeNewsDataset
/multimodal_only_samples/images/train/fake/")

extensions=['jpg','jpeg','JPEG']
```

The below 'for loop' is useful in resizing the images to a standard size of '200 x 200' and it needs to be iterated for all the above-defined variables files_testf, files_traint, and files_trainf

```
for i in files_testt:
    ext=i.split('.')[-1]
    if ext in extensions:

im=Image.open("/Users/prashantwaghela/Desktop/FakeNewsDataset/multimoda
l_only_samples/images/test/true/"+ i)
        im_resized=im.resize((200,200))

filepath="/Users/prashantwaghela/Desktop/FakeNewsDataset/multimodal_onl
y_samples/images_resized/test_resized/true_resized/" + i
        im_resized.convert('RGB').save(filepath)
```

6 Permanently Move Images to Google Drive

The above code sections will provide us with the downloaded and resized images that are usable for model implementation. Since a multimodal algorithm is used, the RAM and GPU requirements are higher and hence, the model could not be trained on the local system. Thus, the downloaded image folders and the clean data frame need to be moved into a Google Drive location.

7 Migration to Google Colab Pro

This part of the project covers the necessary steps to implement the unique multimodal algorithms on the Google Colab Pro Notebook. This step is chosen because of the high RAM and GPU requirements for the model training phase.

First, mount the Colab notebook with Google Drive so that the data frame and all the images could be accessed for the model-building process.

```
from google.colab import drive
drive.mount('/content/drive')
```

We need to add a custom ‘for loop’ to create a new column which consists of all the uploaded google drive image paths.

```
old_filelength=38520
train_split_val = 29699    #This value is 80% of entire dataset length
FilePath_arr=[]
img_number_arr=[]
for i in range(0,len(df)):
    if((i <= train_split_val) & (df.iat[i,5] == 0)):

FilePath_arr.append('/content/drive/MyDrive/images_resized/train_resize
d/fake_resized/image-' + str(df.iat[i,1]) + '.jpg')
        img_number_arr.append(df.iat[i,1])
    elif((i <= train_split_val) & (df.iat[i,5] == 1)):

FilePath_arr.append('/content/drive/MyDrive/images_resized/train_resize
d/true_resized/image-' + str(df.iat[i,1]) + '.jpg')
        img_number_arr.append(df.iat[i,1])
    elif((i > train_split_val) & (df.iat[i,5] == 0)):

FilePath_arr.append('/content/drive/MyDrive/images_resized/test_resized
/fake_resized/image-' + str(df.iat[i,1]) + '.jpg')
        img_number_arr.append(df.iat[i,1])
    elif((i > train_split_val) & (df.iat[i,5] == 1)):

FilePath_arr.append('/content/drive/MyDrive/images_resized/test_resized
/true_resized/image-' + str(df.iat[i,1]) + '.jpg')
        img_number_arr.append(df.iat[i,1])
```

The next important thing is to clean the text data and for that, we have used NLTK libraries which is visible from the below code snippet.

```
#Importing necessary libraries

from tensorflow.keras.layers import Embedding
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.text import one_hot
from tensorflow.keras import layers

voc_size=10000

import nltk
import re
from nltk.corpus import stopwords
nltk.download('stopwords')
```

```
#News text is cleaned using regular expression which removes special
characters from the text.
#Porter Stemming is used to remove stop words from the text to simplify
it for the model.

from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
corpus = []
for i in range(0, len(merged_df)):
    print(i)
    news = re.sub('[^a-zA-Z0-9]', ' ', merged_df.iat[i,5])
    news = news.lower()
    news = news.split()

    news = [ps.stem(word) for word in news if not word in
stopwords.words('english')]
    news = ' '.join(news)
    corpus.append(news)
```

The implementation of all the above steps will provide the pre-processed image and text data set which could now be utilised for the model training phase.

7.1 NPZ Array Creation

One custom requirement of a multimodal algorithm with different input data streams is that it needs an n-dimensional NumPy array where a single index would provide the model with the text, image and target column data for that specific data row. It basically simplifies the way in which multimodal data could be accessed. To implement this we need to save '.npz' files on Google Drive which contains this set of information in a single index. The next code snippet will help us in implementing the same.

```
import cv2

npz_paths = []

for i, row in df_final.iterrows():
  picture_path = row['Path']
  print(i)
  npz_path = picture_path.split('.')[0] + '.npz'
  npz_paths.append(npz_path)

  pic_bgr_arr = cv2.imread(picture_path)
  pic_rgb_arr = cv2.cvtColor(pic_bgr_arr, cv2.COLOR_BGR2RGB)

  preprocessed_news = row['preprocessed_newsText']
  news = np.array(preprocessed_news)
```

```
  fake = row['is_fake']
  np.savez_compressed(npz_path, pic=pic_rgb_arr, news=news, fake=fake)
```

The 'npz_paths' array should be converted into a list and it should be added to the data frame so that it could be utilised during the model building phase.

7.2 Data Splitting

A custom function to split the data into training, validation and testing sets is implemented in the study.

```
def get_X_y(df):

  X_pic, X_news= [], []
  y=[]

  for name in df['NPZ_Path_new']:
    loaded_npz = np.load(name)
    print(name)
    pic = loaded_npz['pic']
    X_pic.append(pic)

    news = loaded_npz['news']
    X_news.append(news)

    y.append(loaded_npz['fake'])

  X_pic, X_news = np.array(X_pic), np.array(X_news)
  y = np.array(y)

  return (X_pic, X_news), y
```

The above-defined function if called using the data frame along with its index values will help in achieving the data split as required for the model.

8 Model Implementation

This section will discuss the important code snippets to define the multimodal architectures.

8.1 Model 1 (BERT+CNN)

This code snippet to build the multimodal algorithm could be seen in 'CNN_BERT_Final_Multimodal.ipynb'

The model definition for this first multimodal algorithm is as shown below. The below code will create our unique multimodal architecture.

```
# Define the Picture (CNN) Stream and some Conv2D layers along with Max
Pooling

img_input = tf.keras.layers.Input(shape=(200, 200, 3))
x = tf.keras.layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(200, 200, 3))(img_input)
x = tf.keras.layers.MaxPooling2D(2, 2)(x)
x = tf.keras.layers.Conv2D(64, (3, 3), activation='relu')(x)
x = tf.keras.layers.MaxPooling2D(2, 2)(x)
x = tf.keras.layers.Conv2D(128, (3, 3), activation='relu')(x)
x = tf.keras.layers.MaxPooling2D(2, 2)(x)
x = tf.keras.layers.Conv2D(128, (3, 3), activation='relu')(x)
x = tf.keras.layers.MaxPooling2D(2, 2)(x)
x = tf.keras.layers.Flatten()(x)
x = tf.keras.layers.Dropout(0.5)(x)
x = tf.keras.layers.Dense(512, activation='relu')(x)
x = tf.keras.Model(inputs=img_input, outputs=x)


# Define the BERT Model
bert_preprocess =
hub.KerasLayer("https://tfhub.dev/tensorflow/bert_en_uncased_preprocess
/3")
bert_encoder =
hub.KerasLayer("https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-
768_A-12/4")

text_input = tf.keras.layers.Input(shape=(), dtype=tf.string,
name='text')
preprocessed_text = bert_preprocess(text_input)
outputs = bert_encoder(preprocessed_text)
y = tf.keras.layers.Flatten()(outputs['pooled_output'])
y = tf.keras.layers.Dropout(0.5, name="dropout2")(y)
y = tf.keras.layers.Dense(768, activation='relu')(y)
y = tf.keras.Model(inputs=[text_input], outputs = [y])


# Concatenate the two streams together

combined = tf.keras.layers.concatenate([x.output, y.output])
z = layers.Dense(64, activation="relu")(combined)
z = layers.Dropout(0.2, name="dropout3")(z)
z = layers.Dense(64, activation="relu")(z)


# Define output node of 1 categorical neuron (classification task)

z = tf.keras.layers.Dense(1, activation="sigmoid")(z)
```

```
# Define the final model

model = Model(inputs=[x.input, y.input], outputs=z)
```

Use of Early Stopping:

The use of early stopping patience helps in avoiding data overfitting while performing model training. The patience value can be changed as per the available system configuration. However, the general practice is to set its value as 3.

```
from keras.callbacks import ModelCheckpoint, EarlyStopping

es = EarlyStopping(patience=3)
cp = ModelCheckpoint('/content/drive/MyDrive/model_latest_BERT_CNN',
save_best_only=True, save_weights_only=True)
cb=[es,cp]
```

8.2 Model 2 (BERT + InceptionV3)

This code snippet to build the multimodal algorithm could be seen in 'Inception_BERT_Final_Multimodal.ipynb'

The initial CNN stream from the above multimodal algorithm can be replaced with a pre-trained InceptionV3 model to identify the performance difference between the two models.

```
# Define the Inception Stream

x = tf.keras.layers.Flatten()(inception.output)
x = tf.keras.layers.Dropout(0.1)(x)
x = tf.keras.layers.Dense(512, activation='relu')(x)
x = tf.keras.Model(inputs=inception.input, outputs=x)

# Define the BERT Model

text_input = tf.keras.layers.Input(shape=(), dtype=tf.string,
name='text')
preprocessed_text = bert_preprocess(text_input)
outputs = bert_encoder(preprocessed_text)
y = tf.keras.layers.Flatten()(outputs['pooled_output'])
y = tf.keras.layers.Dropout(0.5, name="dropout2")(y)
y = tf.keras.layers.Dense(768, activation='relu')(y)
y = tf.keras.Model(inputs=[text_input], outputs = [y])


# Concatenate the two streams together

combined = tf.keras.layers.concatenate([x.output, y.output])
```

```
z = layers.Dense(64, activation="relu")(combined)


# Define output node of 1 categorical neuron (classification task)

z = tf.keras.layers.Dense(1, activation="sigmoid")(combined)


# Define the final model

model = Model(inputs=[x.input, y.input], outputs=z)
```

8.3 Model 3 (XML_RoBERTa + InceptionV3)

This code snippet to build the multimodal algorithm could be seen in 'CNN_XML_RoBERTa_MultiModal_Final.ipynb'

```
# Define the Picture (CNN) Stream and some Conv2D layers along with Max
Pooling

img_input = tf.keras.layers.Input(shape=(200, 200, 3))
x = tf.keras.layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(200, 200, 3))(img_input)
x = tf.keras.layers.MaxPooling2D(2, 2)(x)
x = tf.keras.layers.Conv2D(64, (3, 3), activation='relu')(x)
x = tf.keras.layers.MaxPooling2D(2, 2)(x)
x = tf.keras.layers.Conv2D(128, (3, 3), activation='relu')(x)
x = tf.keras.layers.MaxPooling2D(2, 2)(x)
x = tf.keras.layers.Flatten()(x)
x = tf.keras.layers.Dropout(0.2)(x)
x = tf.keras.layers.Dense(512, activation='relu')(x)
x = tf.keras.Model(inputs=img_input, outputs=x)


# Define the XML_RoBERTa Model

xml_RoBERTa_preprocess =
hub.KerasLayer("https://tfhub.dev/jeongukjae/xlm_roberta_multi_cased_pr
eprocess/1")
xml_RoBERTa_encoder =
hub.KerasLayer("https://tfhub.dev/jeongukjae/xlm_roberta_multi_cased_L-
12_H-768_A-12/1")

text_input = tf.keras.layers.Input(shape=(), dtype=tf.string,
name='text')
preprocessed_text = xml_RoBERTa_preprocess(text_input)
outputs = xml_RoBERTa_encoder(preprocessed_text)
y = tf.keras.layers.Flatten()(outputs['pooled_output'])
```

```
y = tf.keras.layers.Dropout(0.2, name="dropout2")(y)
y = tf.keras.layers.Dense(768, activation='relu')(y)
y = tf.keras.Model(inputs=[text_input], outputs = [y])


# Concatenate the two streams together

combined = tf.keras.layers.concatenate([x.output, y.output])
z = layers.Dense(64, activation="relu")(combined)

# Define output node of 1 categorical neuron (classification task)

z = tf.keras.layers.Dense(1, activation="sigmoid")(z)


# Define the final model

model = Model(inputs=[x.input, y.input], outputs=z)
```

The implementation of the BERT and XML_RoBERTa unimodal models is the same with the absence of the CNN and Keras concatenation stream.

References

Sharma, D.K. and Garg, S., 2021. IFND: a benchmark dataset for fake news detection. *Complex & Intelligent Systems*, pp.1-21.