Configuration Manual

MSc Research Project
Data Analytics

Kishan Kumar Bajaj
Student ID: x20131241

School of Computing
National College of Ireland

Supervisor: Dr. Arghir-Nicolae Moldovan

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Kishan Kumar Bajaj
Student ID:	x20131241
Programme:	Data Analytics
Year:	2021
Module:	MSc Research Project
Supervisor:	Dr. Arghir-Nicolae Moldovan
Submission Due Date:	31/01/2022
Project Title:	Configuration Manual
Word Count:	1046
Page Count:	6

Signature:	
Date:	31st January 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.	☐

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Kishan Kumar Bajaj
x20131241

1 Introduction

This configuration manual is created to help user's understand the details and the various steps carried out for implementing the ResNet + TCN neural network architecture for classification of affective states and their levels of students during online learning. This manual lists all the artefacts that were developed for implementing this machine learning model.

The manual only lists the technical aspects of the system used to develop this application and this is supplementary to the dissertation thesis report submitted on the topic - "Classification of Affective States and their Level in a Learning Environment using Neural Networks". To understand the modelling details of this application, user's are requested to refer the project report.

2 System Configuration

This section of the manual describes the software and hardware configuration of the system that was used or is required to execute the model for classification of affective states and their levels of students during online learning. Please note, a change in software or hardware configuration to perform the experiments using the developed model may result in different result as compared to the one's reported in the project report.

2.1 Hardware

- **Operating System:** Windows 10, 64-bit
- **RAM:** 16 GB
- **Storage:** 512 GB (minimum 100 GB required to store and process the videos)
- **Processor:** Intel(R) Core(TM) i5 CPU @1.60 GHz
- **GPU:** NVIDIA GeForce MX150 2 GB (minimum 2GB dedicated NVIDIA GPU memory required)

2.2 Software

- **Package Manager:** Anaconda 4.7.12. It comes with Python distribution, hence Python installation is not required explicitly. It also provides python development environment such as Jupyter Notebook, JupyterLab, Spyder, etc.

- **Spyder:** 5.1.5, Python development environment used in this research.
- **Python:** 3.8.12
- **pytorch:** 1.10.0
- **torchvision:** 0.11.1
- **sklearn**
- **pandas:** 1.3.4
- **numpy:** 1.21.2
- **matplotlib:** 3.5.0

3 Prerequisites

This section lists the pre-requisites required before performing the experiments.

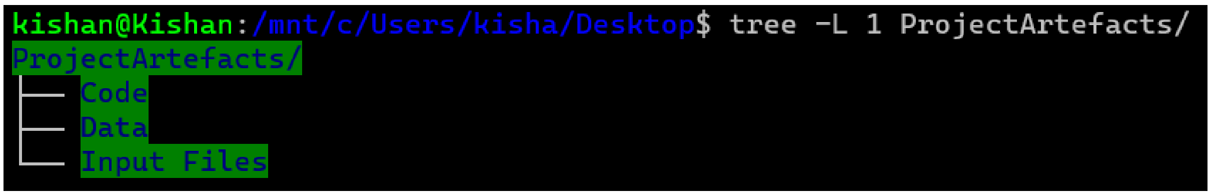
1. All softwares mentioned in Section 2.2 should be installed using the Anaconda package manager or pip utility as both of them automatically install any dependencies.
2. All the datasets should be downloaded, unzipped and stored in the current working directory. Download URLs are provided in artefacts zip.

Data Sources:

- **DAiSEE:** Dataset for Affective States in E-Environments
- **EmotiW2020:** Engagement Prediction in the Wild

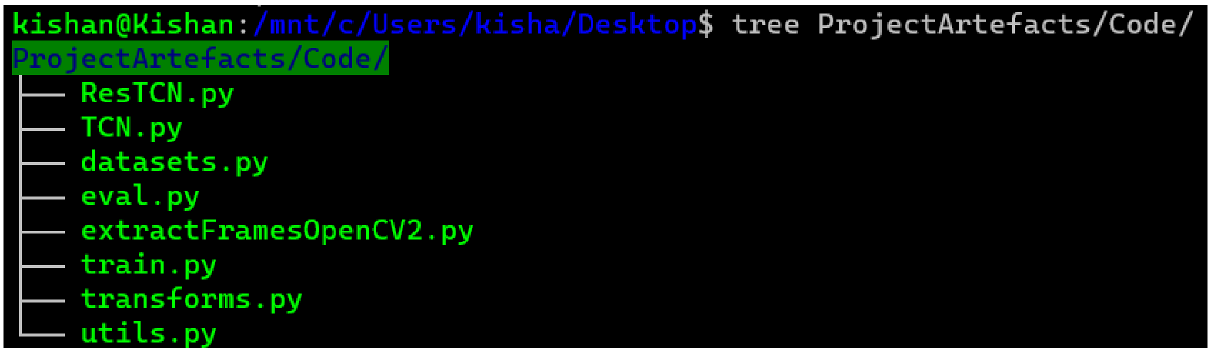
4 Project Artefacts

The artefacts are provided in x20131241_KishanKumarBajaj_ProjectArtefacts.zip file. The structure of the contents of the zip file is as seen if Figure 1. Contents of each directory and their usage is described in further sub sections.



```
kishan@Kishan:/mnt/c/Users/kisha/Desktop$ tree -L 1 ProjectArtefacts/
ProjectArtefacts/
├── Code
├── Data
└── Input Files
```

Figure 1: Project Artefacts directory structure



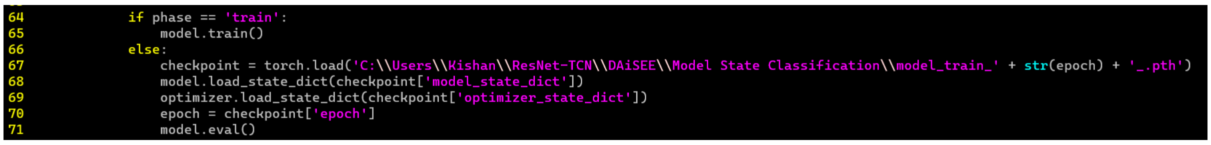
```
kishan@Kishan:/mnt/c/Users/kisha/Desktop$ tree ProjectArtefacts/Code/
ProjectArtefacts/Code/
├── ResTCN.py
├── TCN.py
├── datasets.py
├── eval.py
├── extractFramesOpenCV2.py
├── train.py
├── transforms.py
└── utils.py
```

Figure 2: Code directory contents

4.1 Code

The Code directory contains eight different Python modules as seen in Figure 2. Each python module and its usage is described further.

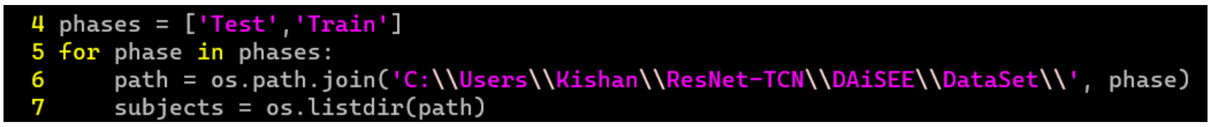
- **ResTCN.py:** This module defines the Residual network (ResNet) class and is called internally.
- **TCN.py:** This module defines the Temporal convolutional network class and is also called internally.
- **datasets.py:** This module defines the VideoDataset class which is used for parsing videos using csv input files and is also called internally.
- **eval.py:** This module is used for evaluating a trained model. The location of trained models needs to be changed in checkpoint variable at line 67 in the existing format for a Windows platform.



```
64            if phase == 'train':
65                model.train()
66            else:
67                checkpoint = torch.load('C:\\Users\\Kishan\\ResNet-TCN\\DAiSEE\\Model State Classification\\model_train_' + str(epoch) + '_.pth')
68                model.load_state_dict(checkpoint['model_state_dict'])
69                optimizer.load_state_dict(checkpoint['optimizer_state_dict'])
70                epoch = checkpoint['epoch']
71                model.eval()
```

Figure 3: eval.py saved models path

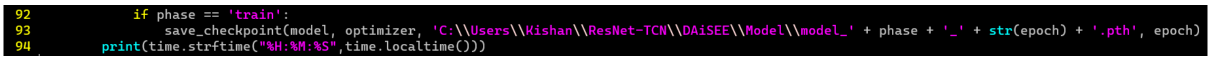
- **extractFramesOpenCV2.py:** This module is used for extracting frames from video clips. The path of the data set needs to be set in path variable at line 6 as seen in Figure 4. This path should contain the video clips split into Train/Test/Validation splits, which need to be updated in phases variable at line 4 in Figure 4.



```
4 phases = ['Test','Train']
5 for phase in phases:
6     path = os.path.join('C:\\Users\\Kishan\\ResNet-TCN\\DAiSEE\\DataSet\\', phase)
7     subjects = os.listdir(path)
```

Figure 4: extractFramesOpenCV2.py data set path

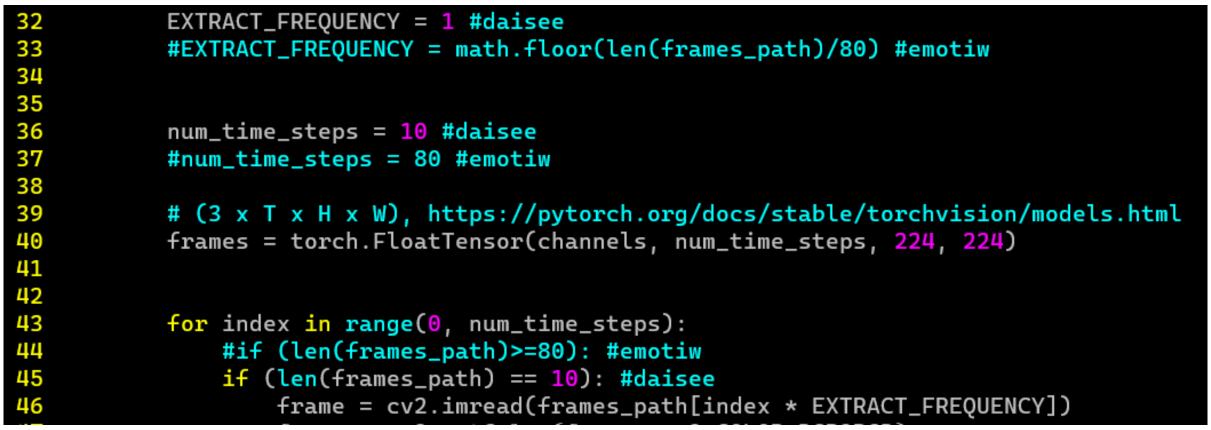
- **train.py:** This module is used to train the model. The location, where trained models are to be saved, needs to be updated in the third argument of save_checkpoint function at line 93 as seen in Figure 5.



```
92            if phase == 'train':
93                save_checkpoint(model, optimizer, 'C:\\Users\\Kishan\\ResNet-TCN\\DAiSEE\\Model\\model_' + phase + '_' + str(epoch) + '.pth', epoch)
94        print(time.strftime("%H:%M:%S",time.localtime()))
```

Figure 5: train.py model save path

- **transforms.py:** This module is used for data transformation before model training and is called internally. While training or evaluating the DAiSEE data set, line 33, 37 and 44 need to be commented as they define the configuration for EmotiW2020 data set. While evaluating the EmotiW2020 data set, line 32, 36 and 45 need to be commented as they define the configuration for DAiSEE data set. This is needed because both the data sets have videos with different length and frame rates and hence need to be transformed differently.



```
32          EXTRACT_FREQUENCY = 1 #daisee
33          #EXTRACT_FREQUENCY = math.floor(len(frames_path)/80) #emotiw
34
35
36          num_time_steps = 10 #daisee
37          #num_time_steps = 80 #emotiw
38
39          # (3 x T x H x W), https://pytorch.org/docs/stable/torchvision/models.html
40          frames = torch.FloatTensor(channels, num_time_steps, 224, 224)
41
42
43          for index in range(0, num_time_steps):
44              #if (len(frames_path)>=80): #emotiw
45              if (len(frames_path) == 10): #daisee
46                  frame = cv2.imread(frames_path[index * EXTRACT_FREQUENCY])
```

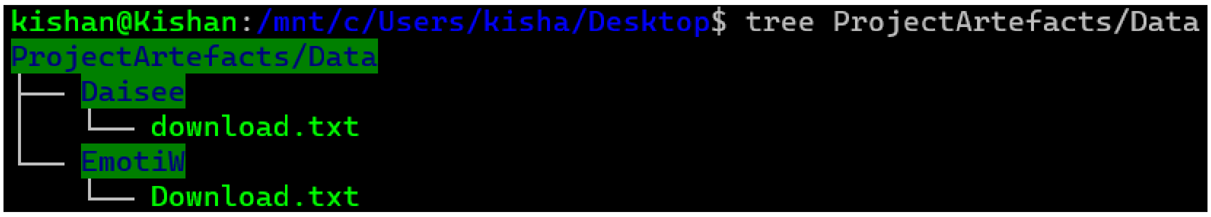
Figure 6: transforms.py DAiSEE and EmotiW2020 configuration parameters

- **utils.py:** This module defines the dataloader classes which are used for loading the model training and evaluation data and is called internally.

4.2 Data

The structure of Data directory is as seen in Figure 7. It contains one folder for each data set, DAiSEE and EmotiW2020. Folder of each data set contains a download.txt file which contains the URL to download the data set. Since the data sets were greater than 60 GB, they could not be uploaded and hence the URLs are provided instead as below:

- **DAiSEE:** `https://iith.ac.in/~daisee-dataset/`
- **EmotiW2020:** `https://1drv.ms/u/s!AkLw1EC6-UzbjPwdwfViatKBDAcgIQ?e=dDUO1U`

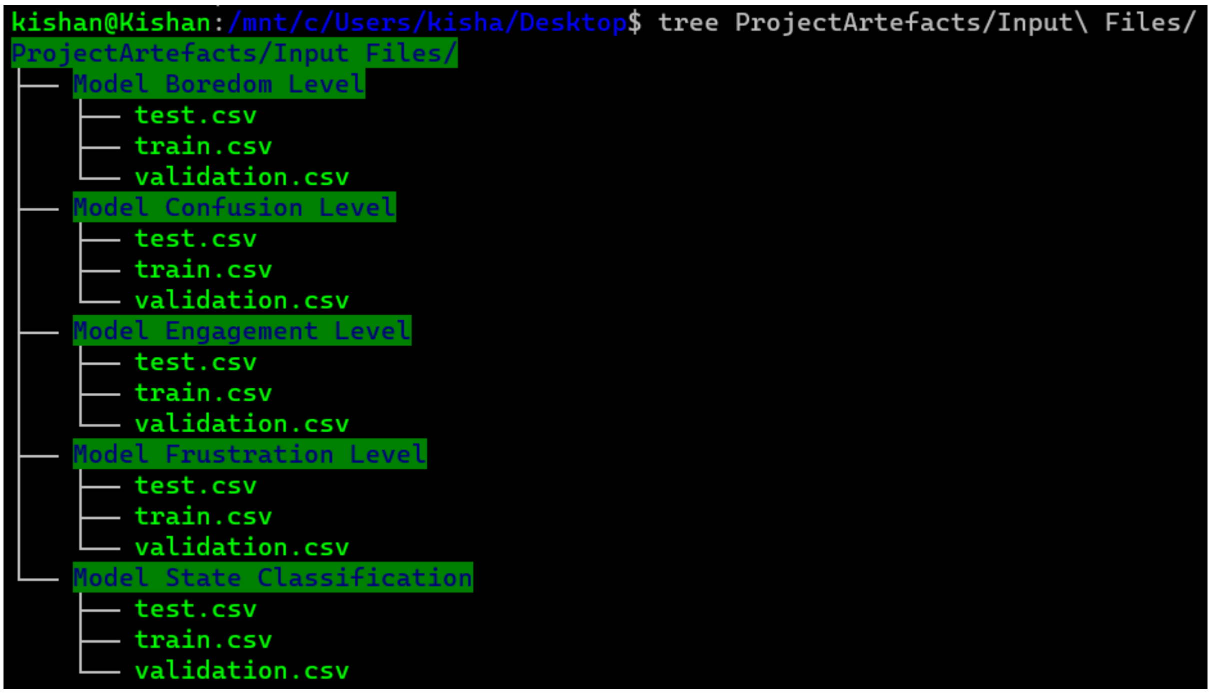


```
kishan@Kishan:/mnt/c/Users/kisha/Desktop$ tree ProjectArtefacts/Data
ProjectArtefacts/Data
├── Daisee
│   └── download.txt
└── EmotiW
    └── Download.txt
```

Figure 7: Data directory structure

4.3 Input Files

The directory structure of Input Files folder is as seen in Figure 8. It contains one directory for each of the five experiments performed in the project. Each of the five directories contain three csv files namely, test.csv, train.csv and validation.csv. These three files need to be present in the same directory where the code is executing for each experiment.



```
kishan@Kishan:/mnt/c/Users/kisha/Desktop$ tree ProjectArtefacts/Input\ Files/
ProjectArtefacts/Input Files/
├── Model Boredom Level
│   ├── test.csv
│   ├── train.csv
│   └── validation.csv
├── Model Confusion Level
│   ├── test.csv
│   ├── train.csv
│   └── validation.csv
├── Model Engagement Level
│   ├── test.csv
│   ├── train.csv
│   └── validation.csv
├── Model Frustration Level
│   ├── test.csv
│   ├── train.csv
│   └── validation.csv
└── Model State Classification
    ├── test.csv
    ├── train.csv
    └── validation.csv
```

Figure 8: Input Files directory structure

5 Commands for performing experiments

After fulfilling the requirements and changes specified in Section 2, Section 3 and Section 4, below commands can be used to train and evaluate the model as per requirement.

- Execute train.py script to start training the model. This module trains the model and saves them in the specified location.

- Execute eval.py script to load the model from specified location and test it. This script will generate the results in a text file and confusion matrix in image files for each epoch of the model in the same directory where trained models are saved.

6 Best Practices

1. Always store a backup of the raw data before performing pre-processing and modifying the raw data structure.
2. We should always keep a backup of the latest "working" version of the artefacts developed before making changes improvements/corrections to avoid losing the work done due to hardware failures or other issues.
3. The outputs generated by the program at each step such as pre-processed data, trained model, etc should be store to avoid re-running all the steps in case of issues in the later stages of development.