Configuration Manual

MSc Research Project
Data Analytics

Nikhil Reddy Byreddy
Student ID: 17136563

School of Computing
National College of Ireland

Supervisor: Vladimir Milosavljevic

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Nikhil Reddy Byreddy
Student ID:	17136563
Programme:	Data Analytics
Year:	2019
Module:	MSc Research Project
Supervisor:	Vladimir Milosavljevic
Submission Due Date:	12/12/2019
Project Title:	Configuration Manual
Word Count:	500
Page Count:	7

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:	12th December 2019

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

Nikhil Reddy Byreddy
17136563

1 Pre Requisites

The prerequisites required to implement this project are to have a valid google colab ID either in ios or windows os and python latest version needs to be installed.

2 Access Colab and Import necessary libraries

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

3 Import Libraries

```
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.utils import to_categorical
from keras.preprocessing import image
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical
from tqdm import tqdm
import cv2     # for capturing videos
import math    # for mathematical operations
import matplotlib.pyplot as plt    # for plotting the images
%matplotlib inline
import pandas as pd
from keras.preprocessing import image   # for preprocessing the images
import numpy as np    # for mathematical operations
from keras.utils import np_utils
from skimage.transform import resize   # for resizing images
from sklearn.model_selection import train_test_split
from glob import glob
from tqdm import tqdm

import os
from os.path import join
import argparse
import subprocess
import cv2
from tqdm import tqdm
```

Figure 1: Import libraries

4 Defined Functions

```python
def extract_frames(data_path, output_path, method='cv2'):
    """Method to extract frames, either with ffmpeg or opencv. FFmpeg won't
    start from 0 so we would have to rename if we want to keep the filenames
    coherent."""
    os.makedirs(output_path, exist_ok=True)
    if method == 'ffmpeg':
        subprocess.check_output(
            'ffmpeg -i {} {}'.format(
                data_path, join(output_path, '%04d.png')),
            shell=True, stderr=subprocess.STDOUT)
    elif method == 'cv2':
        reader = cv2.VideoCapture(data_path)
        count = 0
        #frame_num = 0
        while reader.isOpened():
            success, image = reader.read()
            if success:
              cv2.imwrite(join(output_path, '{:04d}.png'.format(count)),image)
              count += 30 # i.e. at 30 fps, this advances one second
              reader.set(1, count)
            else:
              reader.release()
              break
    else:
        raise Exception('Wrong extract frames method: {}'.format(method))
```

Figure 2: To Extract Frames From Videos

```python
##Code to get faces from images
path = "/content/gdrive/My Drive/data/original_sequences/c40/images1/"

facedata = "/content/gdrive/My Drive/haarcascade_frontalface_default.xml"
cascade = cv2.CascadeClassifier(facedata)

for root, _, files in os.walk(path):
    current_directory_path = os.path.abspath(root)
    for f in files:
        name, ext = os.path.splitext(f)
        #print(name)
        if ext == ".png":
            current_image_path = os.path.join(current_directory_path, f)
            current_image = cv2.imread(current_image_path)
            for i, face in enumerate(cascade.detectMultiScale(current_image,scaleFactor=1.1,minNeighbors = 6)):
                x, y, w, h = face
                sub_face = current_image[y:y + h, x:x + w]
                cv2.imwrite(os.path.join("/content/gdrive/My Drive/data/original_sequences/c40/maja/", "{}_{}.png".f
```

Figure 3: To extract face from frames

```
#plt.rcParams['figure.figsize'] = (8, 8)
fig = plt.figure()
a = fig.add_subplot(1, 2, 1)
img = mpimg.imread("/content/gdrive/My Drive/data/plot/image.png")
imgplot = plt.imshow(img, aspect='auto')
a.set_title('Full Image',color = 'red')
plt.colorbar(ticks=[0.1, 0.3, 0.5, 0.7], orientation='horizontal')
a = fig.add_subplot(1, 2, 2)
img = mpimg.imread("/content/gdrive/My Drive/data/plot/Face.png")
imgplot = plt.imshow(img, aspect='auto')
imgplot.set_clim(0.0, 0.7)
a.set_title('Face Extracted Image',color = 'green')
plt.colorbar(ticks=[0.1, 0.3, 0.5, 0.7], orientation='horizontal')
```

```
<matplotlib.colorbar.Colorbar at 0x7ff34c6f5550>
```

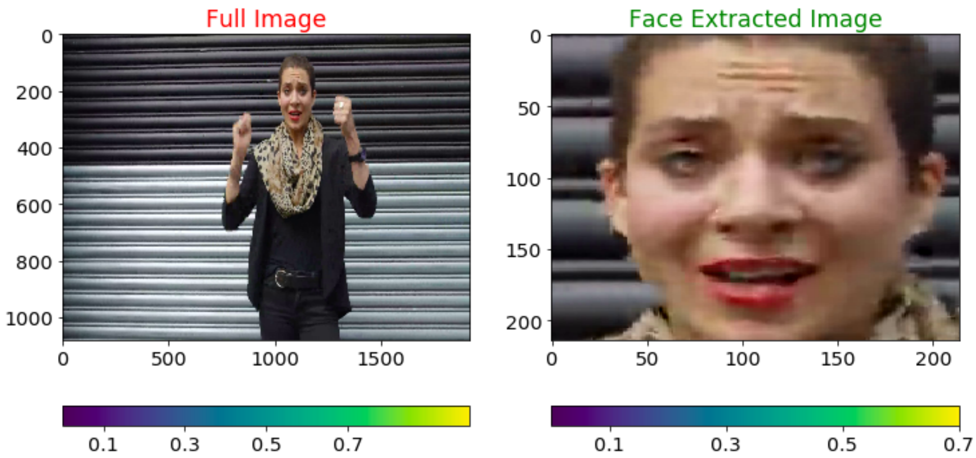

Figure 4: Face from frames

```
lap = cv2.imread("/content/gdrive/My Drive/data/plot/Face.png",0)
lap = cv2.Laplacian(lap,cv2.CV_64F)
cv2.imwrite("/content/gdrive/My Drive/data/plot/lap.png",lap)
fig = plt.figure()
a = fig.add_subplot(1, 2, 1)
img = mpimg.imread("/content/gdrive/My Drive/data/plot/Face.png")
imgplot = plt.imshow(img, aspect='auto')
a.set_title('Face Image',color = 'red')
plt.colorbar(ticks=[0.1, 0.3, 0.5, 0.7], orientation='horizontal')
a = fig.add_subplot(1, 2, 2)
img = mpimg.imread("/content/gdrive/My Drive/data/plot/lap.png")
imgplot = plt.imshow(img,cmap = 'gray', aspect='auto')
imgplot.set_clim(0.0, 0.7)
a.set_title('Laplacian Transformed',color = 'green')
plt.colorbar(ticks=[0.1, 0.3, 0.5, 0.7], orientation='horizontal')
```

```
<matplotlib.colorbar.Colorbar at 0x7ff34c244470>
```

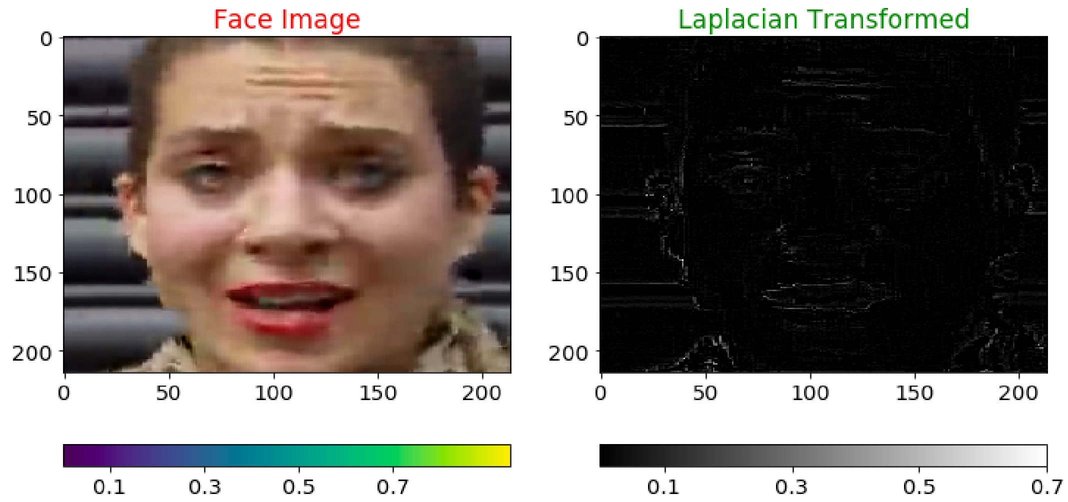

Figure 5: Laplace Transform on images

```python
edges = cv2.imread("/content/gdrive/My Drive/data/plot/Face.png",0)
edges = cv2.Canny(edges,0,100)

cv2.imwrite("/content/gdrive/My Drive/data/plot/edges.png",lap)
fig = plt.figure()
a = fig.add_subplot(1, 2, 1)
img = mpimg.imread("/content/gdrive/My Drive/data/plot/Face.png")
imgplot = plt.imshow(img, aspect='auto')
a.set_title('Face Image',color = 'red')
plt.colorbar(ticks=[0.1, 0.3, 0.5, 0.7], orientation='horizontal')
a = fig.add_subplot(1, 2, 2)
img = mpimg.imread("/content/gdrive/My Drive/data/plot/edges.png")
imgplot = plt.imshow(img,cmap = 'gray', aspect='auto')
imgplot.set_clim(0.0, 0.7)
a.set_title('Canny Edge Transformed',color = 'green')
plt.colorbar(ticks=[0.1, 0.3, 0.5, 0.7], orientation='horizontal')
```

```
<matplotlib.colorbar.Colorbar at 0x7ff34b6e0be0>
```

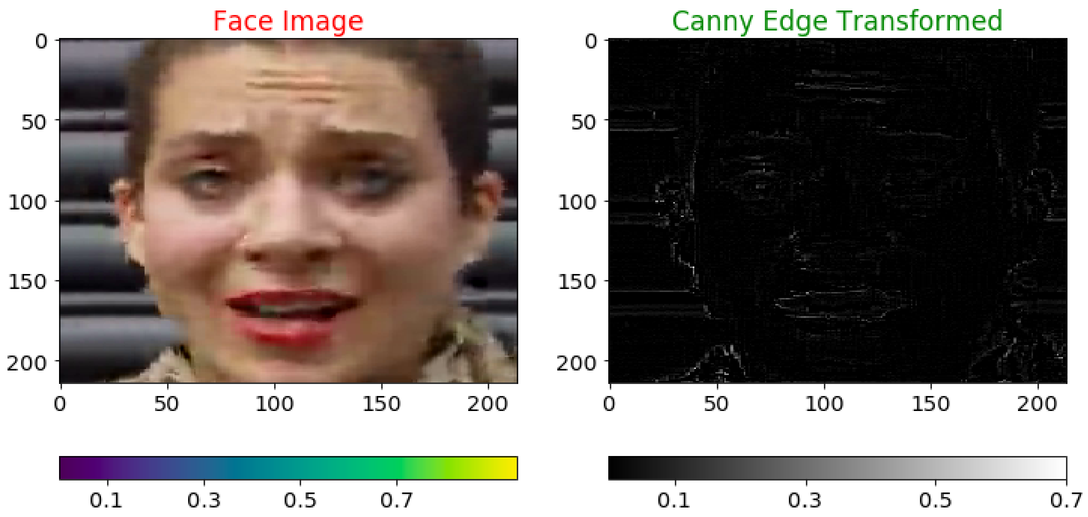

Figure 6: Canny edge Transform on images

5 Implementation

```
# Creating validation set for raw images
X_train, X_test, y_train, y_test = train_test_split(X_org, y_org, random_state=42, test_size=0.2)

# Define the model structure
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),activation='relu',input_shape=(28,28,3)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(2, activation='softmax'))


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


# Training the model with transform face images

# Fit the model
history = model.fit(X_train, y_train, epochs=50, batch_size = 20,validation_data=(X_test, y_test))
```

Figure 7: CNN

```
import strawberryfields as sf
from strawberryfields.ops import Dgate, BSgate
import tensorflow as tf
import qmlt as qm
import qmlt.tf as pl
from qmlt.tf.helpers import make_param
from qmlt.tf import CircuitLearner
```

Figure 8: QML import

```
def circuit(X):
    phi = make_param('phi', constant=2.)

    eng, q = sf.Engine(2)

    with eng:
        Dgate(X[:, 0], 0.) | q[0]
        Dgate(X[:, 1], 0.) | q[1]
        BSgate(phi=phi) | (q[0], q[1])
        BSgate() | (q[0], q[1])


    num_inputs = X.get_shape().as_list()[0]
    state = eng.run('tf', cutoff_dim=10, eval=False, batch_size=num_inputs)

    p0 = state.fock_prob([0, 2])
    p1 = state.fock_prob([2, 0])
    normalisation = p0 + p1 + 1e-10
    circuit_output = p1/normalisation

    return circuit_output
```

Figure 9: QML Circuit

```python
hyperparams = {'circuit': circuit,
               'task': 'supervised',
               'loss': myloss,
               'optimizer': 'SGD',
               'init_learning_rate': 0.5
               }

learner = CircuitLearner(hyperparams=hyperparams)

learner.train_circuit(X=X_train, Y=Y_train, steps=3,batch_size=2)

test_score = learner.score_circuit(X=X_test, Y=Y_test,
                                   outputs_to_predictions=outputs_to_predictions)
print("\nPossible scores to print: {}".format(list(test_score.keys())))
print("Accuracy on test set: ", test_score['accuracy'])
print("Loss on test set: ", test_score['loss'])

outcomes = learner.run_circuit(X=X_pred, outputs_to_predictions=outputs_to_predictions)

print("\nPossible outcomes to print: {}".format(list(outcomes.keys())))
print("Predictions for new inputs: {}".format(outcomes['predictions']))
```

Figure 10: QML circuit

```python
from time import time
t0=time()
learner.train_circuit(X=t, Y=r, steps=50,batch_size=10)
print ("training time:", round(time()-t0, 3), "s")
```

Figure 11: QML implementation