MW-1D-CNN

January 26, 2020

1 CNN 1D IoT Classification Model

1.1 Importing Libraries

[1]:
```
from __future__ import print_function
import h5py
import numpy as np
import matplotlib.pyplot as plt
from sklearn.utils import class_weight
from sklearn.metrics import classification_report
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Input, Concatenate
from keras.layers import Conv1D, MaxPooling1D, AveragePooling1D
from keras.utils import plot_model
from keras.models import Model
from hyperopt import Trials, STATUS_OK, tpe
from hyperas import optim
from hyperas.distributions import choice, uniform
from keras.utils import multi_gpu_model
from PIL import Image
import matplotlib.pyplot as plt
import pandas as pd
import copy
import tensorflow as tf
%matplotlib inline
```

```
Using TensorFlow backend.
c:\users\mrathbun2018\.conda\envs\mattwork\lib\site-
packages\tensorflow\python\framework\dtypes.py:516: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
c:\users\mrathbun2018\.conda\envs\mattwork\lib\site-
packages\tensorflow\python\framework\dtypes.py:517: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
```

```
c:\users\mrathbun2018\.conda\envs\mattwork\lib\site-
packages\tensorflow\python\framework\dtypes.py:518: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
c:\users\mrathbun2018\.conda\envs\mattwork\lib\site-
packages\tensorflow\python\framework\dtypes.py:519: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
c:\users\mrathbun2018\.conda\envs\mattwork\lib\site-
packages\tensorflow\python\framework\dtypes.py:520: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
c:\users\mrathbun2018\.conda\envs\mattwork\lib\site-
packages\tensorflow\python\framework\dtypes.py:525: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
c:\users\mrathbun2018\.conda\envs\mattwork\lib\site-
packages\tensorboard\compat\tensorflow_stub\dtypes.py:541: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
c:\users\mrathbun2018\.conda\envs\mattwork\lib\site-
packages\tensorboard\compat\tensorflow_stub\dtypes.py:542: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
c:\users\mrathbun2018\.conda\envs\mattwork\lib\site-
packages\tensorboard\compat\tensorflow_stub\dtypes.py:543: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
c:\users\mrathbun2018\.conda\envs\mattwork\lib\site-
packages\tensorboard\compat\tensorflow_stub\dtypes.py:544: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
c:\users\mrathbun2018\.conda\envs\mattwork\lib\site-
packages\tensorboard\compat\tensorflow_stub\dtypes.py:545: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
c:\users\mrathbun2018\.conda\envs\mattwork\lib\site-
packages\tensorboard\compat\tensorflow_stub\dtypes.py:550: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
```

```
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
```

1.2 Open and Read Data

[2]:
```
def data():
    import tensorflow as tf
    hdf5_path = 'Data/dataset.hdf5'
    subtract_mean = True

    hdf5_file = h5py.File(hdf5_path, "r")

    if subtract_mean:
        mm = hdf5_file["train_mean"][...,0]
        mm = mm[np.newaxis, ...]

    data_num = hdf5_file["train_flow"].shape[0]

    #batch_size = 512
    num_classes = 2
    epochs = 30

    flow_rows, flow_cols = 298, 17

    x_train = hdf5_file["train_flow"][...,0]

    if subtract_mean:
        x_train -= mm

    y_train = hdf5_file["train_labels"][:,...]
    hdf5_file.close()

    hdf5_path = 'Data/dataset-IoT.hdf5'
    hdf5_file = h5py.File(hdf5_path, "r")


    x_test = hdf5_file["IoT_flow"][...,0]
    if subtract_mean:
        x_test -= mm

    y_test = hdf5_file["labels"][:,...]

    hdf5_file.close()

    class_weights = class_weight.compute_class_weight('balanced',
                                                      np.unique(y_train),
                                                      y_train)
    d_class_weights = dict(enumerate(class_weights))
    #print(d_class_weights)

    input_shape = (x_train.shape[1], x_train.shape[2])

    #print('x_train shape:', x_train.shape)
    #print(x_train.shape[0], 'train samples')
    #print(x_test.shape[0], 'test samples')

    y_train = keras.utils.to_categorical(y_train, num_classes)
    y_test = keras.utils.to_categorical(y_test, num_classes)
    return x_train, y_train, x_test, y_test
```

1.3 Buld the CNN 1D Model

[3]:
```
def create_model(x_train, y_train, x_test, y_test):
    num_classes = 2
    high = {{choice([40,50,60,70,80])}}

    filter_lenghts =  [int(i) for i in np.arange(2,high,2)]
    print(filter_lenghts)
    convs = []
    maxlen = 298
    batch_size = {{choice([256,512,1024])}}
    epochs = 30
    numFilters={{choice([32,64,128])}}
    activations={{choice(['relu', 'sigmoid', 'tanh'])}}
    dropoutVal = {{uniform(0.1, 0.3)}}
    lr = {{uniform(0.0009, 0.00225)}}
    adam = keras.optimizers.Adam(lr=lr)
    rmsprop = keras.optimizers.RMSprop(lr=lr)
    sgd = keras.optimizers.SGD(lr=lr)

    choiceval = {{choice(['adam', 'sgd', 'rmsprop'])}}
    if choiceval == 'adam':
        optim = adam
    elif choiceval == 'rmsprop':
        optim = rmsprop
    else:
        optim = sgd
    input_flow = Input(shape=input_shape)

    convs= {}
    mpoolings = {}
    flattens = {}
    convs_out = []
    for i in filter_lenghts:
       ␣
 ↪convs[str(i)+'_convolution']=Conv1D(filters=numFilters,kernel_size=i,padding="valid",activa

        mpoolings[str(i)+'_maxpooling'] = MaxPooling1D(pool_size= maxlen - i +␣
 ↪1)(convs[str(i)+'_convolution'])
        flattens[str(i)+'_flattenout'] =␣
 ↪Flatten()(mpoolings[str(i)+'_maxpooling'])
        convs_out.append(flattens[str(i)+'_flattenout'])
    out = Concatenate()(convs_out)
    dropout = Dropout(dropoutVal)(out)
    dense = Dense(64, activation='relu')(dropout)
    dense2 = Dense(32, activation='relu')(dense)
    dropout2 = Dropout(dropoutVal)(dense2)
    end = Dense(num_classes, activation='softmax')(dropout2)

    model = Model(inputs=input_flow, outputs=end)
    #model.summary()
    try:
        model = multi_gpu_model(model, gpus = 4)
    except:
        pass
    model.
 ↪compile(loss='binary_crossentropy',optimizer=optim,metrics=['accuracy'])
    model.fit(x_train, y_train,batch_size=batch_size, epochs=30, verbose=0,␣
 ↪validation_split=0.2, class_weight=class_weights, shuffle=True)
    score = model.evaluate(x_test, y_test, verbose=0)
    loss = score[0]
    return {'loss': loss, 'status': STATUS_OK, 'model': model}
```

1.4 Run Model

[4]:
```
x_train, y_train, x_test, y_test = data()
best_run, best_model = optim.minimize(model=create_model, data=data, algo=tpe.
 ↪suggest, max_evals=100, trials=Trials(), eval_space=True,␣
 ↪notebook_name='MW-1D-CNN')

print("Evalutation of best performing model:")
print(best_model.evaluate(x_test, y_test))
print("Best performing model chosen hyper-parameters:")
print(best_run)
```

```
>>> Imports:
#coding=utf-8

from __future__ import print_function

try:
    import h5py
except:
    pass

try:
    import numpy as np
except:
    pass

try:
    import matplotlib.pyplot as plt
except:
    pass

try:
    from sklearn.utils import class_weight
except:
    pass

try:
    from sklearn.metrics import classification_report
except:
    pass

try:
    import keras
except:
    pass

try:
    from keras.models import Sequential
except:
    pass

try:
    from keras.layers import Dense, Dropout, Flatten, Input, Concatenate
except:
    pass

try:
    from keras.layers import Conv1D, MaxPooling1D, AveragePooling1D
except:
    pass

try:
    from keras.utils import plot_model
except:
    pass
```

```
try:
    from keras.models import Model
except:
    pass

try:
    from hyperopt import Trials, STATUS_OK, tpe
except:
    pass

try:
    from hyperas import optim
except:
    pass

try:
    from hyperas.distributions import choice, uniform
except:
    pass

try:
    from keras.utils import multi_gpu_model
except:
    pass

try:
    from PIL import Image
except:
    pass

try:
    import matplotlib.pyplot as plt
except:
    pass

try:
    import pandas as pd
except:
    pass

try:
    import copy
except:
    pass

try:
    import tensorflow as tf
```

```
except:
    pass

try:
    import tensorflow as tf
except:
    pass

try:
    from sklearn.metrics import confusion_matrix
except:
    pass

try:
    from sklearn.metrics import roc_curve
except:
    pass

try:
    from sklearn.metrics import auc
except:
    pass

try:
    from sklearn.metrics import precision_recall_curve
except:
    pass

try:
    from sklearn.metrics import f1_score
except:
    pass

try:
    from sklearn.metrics import auc
except:
    pass

try:
    from sklearn.metrics import average_precision_score
except:
    pass

>>> Hyperas search space:

def get_space():
    return {
        'high': hp.choice('high', [40,50,60,70,80]),
        'batch_size': hp.choice('batch_size', [256,512,1024]),
        'numFilters': hp.choice('numFilters', [32,64,128]),
        'activations': hp.choice('activations', ['relu', 'sigmoid', 'tanh']),
        'dropoutVal': hp.uniform('dropoutVal', 0.1, 0.3),
        'lr': hp.uniform('lr', 0.0009, 0.00225),
        'choiceval': hp.choice('choiceval', ['adam', 'sgd', 'rmsprop']),
    }
```

```
>>> Data

import tensorflow as tf
hdf5_path = 'Data/dataset.hdf5'
subtract_mean = True

hdf5_file = h5py.File(hdf5_path, "r")

if subtract_mean:
    mm = hdf5_file["train_mean"][…,0]
    mm = mm[np.newaxis, …]

data_num = hdf5_file["train_flow"].shape[0]

#batch_size = 512
num_classes = 2
epochs = 30

flow_rows, flow_cols = 298, 17

x_train = hdf5_file["train_flow"][…,0]

if subtract_mean:
    x_train -= mm

y_train = hdf5_file["train_labels"][:,…]
hdf5_file.close()

hdf5_path = 'Data/dataset-IoT.hdf5'
hdf5_file = h5py.File(hdf5_path, "r")


x_test = hdf5_file["IoT_flow"][…,0]
if subtract_mean:
    x_test -= mm

y_test = hdf5_file["labels"][:,…]

hdf5_file.close()

```

```
class_weights = class_weight.compute_class_weight('balanced',
                                             np.unique(y_train),
                                             y_train)
d_class_weights = dict(enumerate(class_weights))
#print(d_class_weights)

input_shape = (x_train.shape[1], x_train.shape[2])

#print('x_train shape:', x_train.shape)
#print(x_train.shape[0], 'train samples')
#print(x_test.shape[0], 'test samples')

y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)



>>> Resulting replaced keras model:
```

```
def keras_fmin_fnct(space):

    num_classes = 2
    high = space['high']

    filter_lenghts =  [int(i) for i in np.arange(2,high,2)]
    print(filter_lenghts)
    convs = []
    maxlen = 298
    batch_size = space['batch_size']
    epochs = 30
    numFilters=space['numFilters']
    activations=space['activations']
    dropoutVal = space['dropoutVal']
    lr = space['lr']
    adam = keras.optimizers.Adam(lr=lr)
    rmsprop = keras.optimizers.RMSprop(lr=lr)
    sgd = keras.optimizers.SGD(lr=lr)

    choiceval = space['choiceval']
    if choiceval == 'adam':
        optim = adam
    elif choiceval == 'rmsprop':
        optim = rmsprop
    else:
        optim = sgd
    input_flow = Input(shape=input_shape)

    convs= {}
    mpoolings = {}
    flattens = {}
    convs_out = []
    for i in filter_lenghts:
        convs[str(i)+'_convolution']=Conv1D(filters=numFilters,kernel_size=i,padding="valid",activation=activations,strides=1)(input_flow)

        mpoolings[str(i)+'_maxpooling'] = MaxPooling1D(pool_size= maxlen - i + 1)(convs[str(i)+'_convolution'])
        flattens[str(i)+'_flattenout'] = Flatten()(mpoolings[str(i)+'_maxpooling'])
        convs_out.append(flattens[str(i)+'_flattenout'])
    out = Concatenate()(convs_out)
    dropout = Dropout(dropoutVal)(out)
    dense = Dense(64, activation='relu')(dropout)
    dense2 = Dense(32, activation='relu')(dense)
    dropout2 = Dropout(dropoutVal)(dense2)
    end = Dense(num_classes, activation='softmax')(dropout2)

    model = Model(inputs=input_flow, outputs=end)
    #model.summary()
    try:
        model = multi_gpu_model(model, gpus = 4)
    except:
        pass
    model.compile(loss='binary_crossentropy',optimizer=optim,metrics=['accuracy'])
    model.fit(x_train, y_train,batch_size=batch_size, epochs=30, verbose=0, validation_split=0.2, class_weight=class_weights, shuffle=True)
    score = model.evaluate(x_test, y_test, verbose=0)
    loss = score[0]
    return {'loss': loss, 'status': STATUS_OK, 'model': model}

[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68]
  0%|
| 0/100 [00:00<?, ?it/s, best loss: ?]WARNING:tensorflow:From
c:\users\mrathbun2018\.conda\envs\mattwork\lib\site-
packages\keras\backend\tensorflow_backend.py:4070: The name tf.nn.max_pool is
deprecated. Please use tf.nn.max_pool2d instead.

WARNING:tensorflow:From c:\users\mrathbun2018\.conda\envs\mattwork\lib\site-
packages\tensorflow\python\ops\math_grad.py:1250:
add_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array_ops) is
deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From c:\users\mrathbun2018\.conda\envs\mattwork\lib\site-
```

```
packages\keras\backend\tensorflow_backend.py:422: The name tf.global_variables
is deprecated. Please use tf.compat.v1.global_variables instead.

[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]
```

```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78]
```

```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]
```

```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
44, 46, 48]
100%|                           | 100/100 [19:21:03<00:00,
696.63s/it, best loss: 0.24673511634403458]
Evalutation of best performing model:
42167/42167 [==============================] - 246s 6ms/step
[0.24673511634403458, 0.8921194076538086]
Best performing model chosen hyper-parameters:
{'activations': 'sigmoid', 'batch_size': 1024, 'choiceval': 'adam',
'dropoutVal': 0.2485206320388983, 'high': 80, 'lr': 0.001362881122433337,
'numFilters': 128}
```

1.5 Model Analysis

Classification Report

Confusion Matrix

Area Under Reciever Operating Characteristic Curve

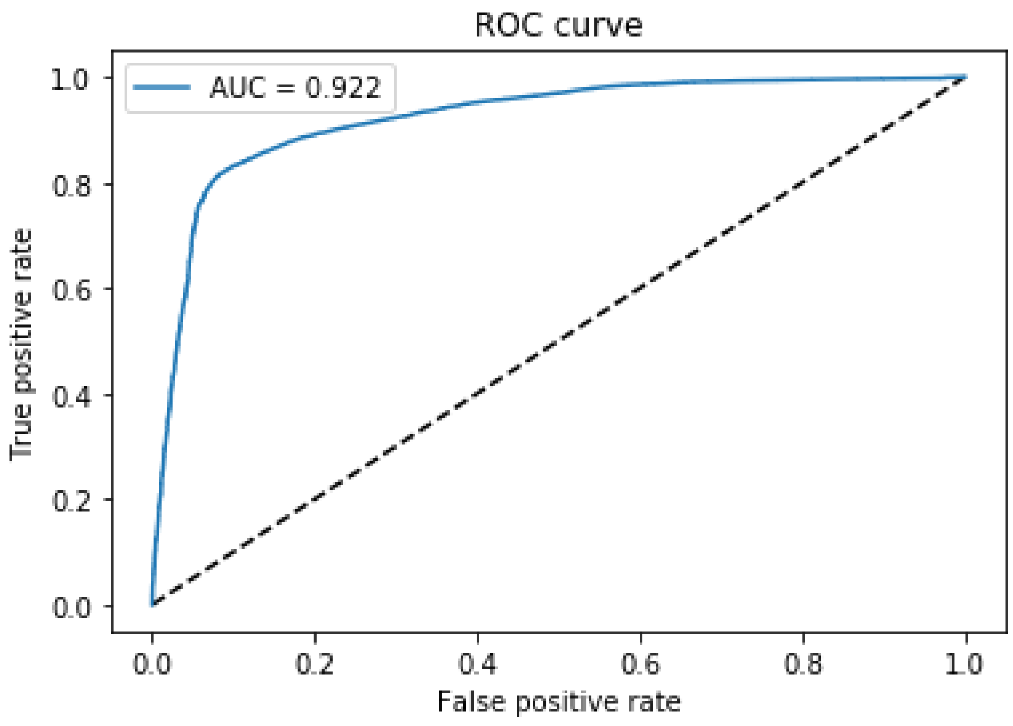
[6]:
```python
y_pred = best_model.predict(x_test)
yy_test = [np.argmax(i) for i in y_test]

yy_pred = [np.argmax(i) for i in y_pred]

print(classification_report(yy_test, yy_pred))

new = np.vstack([yy_test,yy_pred])
```

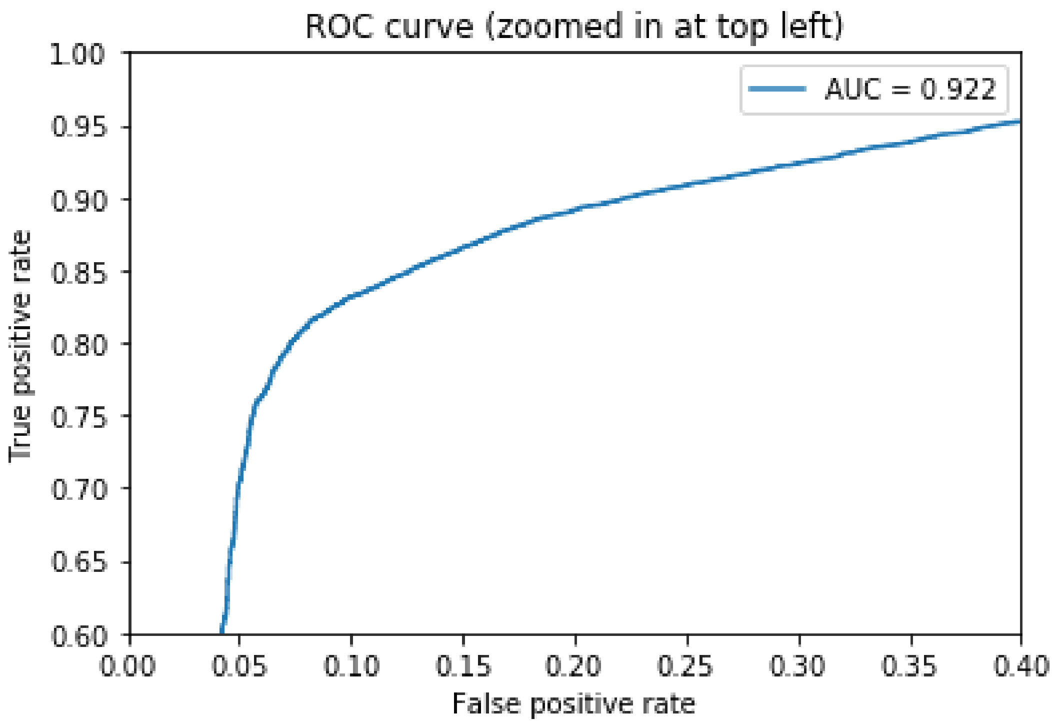
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve
from sklearn.metrics import auc

print(confusion_matrix(yy_test, yy_pred))



y_pred_keras = best_model.predict(x_test).ravel()
fpr_keras, tpr_keras, thresholds_keras = roc_curve(yy_test, y_pred[:
 ↪,0],pos_label=0)
auc_keras = auc(fpr_keras, tpr_keras)
print(auc_keras)

f1 = plt.figure()
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr_keras, tpr_keras, label='AUC = {:.3f}'.format(auc_keras))
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.show()
f1.savefig("ROC-curve-cnn1D-MW.pdf", bbox_inches='tight')

f2 = plt.figure()
plt.xlim(0, 0.4)
plt.ylim(0.6, 1)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr_keras, tpr_keras, label='AUC = {:.3f}'.format(auc_keras))
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve (zoomed in at top left)')
plt.legend(loc='best')
plt.show()
f2.savefig("ROC-curve-zoomed-cnn1D-MW.pdf", bbox_inches='tight')

from sklearn.metrics import precision_recall_curve
from sklearn.metrics import f1_score
from sklearn.metrics import auc
from sklearn.metrics import average_precision_score

precision, recall, thresholds = precision_recall_curve(yy_test,  y_pred[:
 ↪,0],pos_label=0)
# calculate F1 score
#f1 = f1_score(yy_test, y_pred)
# calculate precision-recall AUC
```

```
auc_score = auc(recall, precision)
print(auc_score)
# calculate average precision score
ap = average_precision_score(yy_test, y_pred[:,1])
print(ap)
#print('auc=%.3f ap=%.3f' % (auc, ap))
# plot no skill
f3 = plt.figure()
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the precision-recall curve for the model
plt.plot( recall, precision,marker='.')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision Recall Curve')

# show the plot
plt.show()
f3.savefig("precisionrecall-cnn1D-MW.pdf", bbox_inches='tight')
num_positive = float(np.count_nonzero(yy_test))
num_negative = float(len(yy_test) - num_positive)
pos_weight = num_negative / num_positive
weights = np.ones_like(yy_test)
weights[yy_test != np.float64(0)] = pos_weight


precision_weighted, recall_weighted, thresholds_weighted =␣
 ↪precision_recall_curve(yy_test,  y_pred[:
 ↪,0],pos_label=0,sample_weight=weights)
#calculate F1 score
#f1 = f1_score(yy_test, y_pred)
# calculate precision-recall AUC
auc_score = auc(recall_weighted, precision_weighted)
print(auc_score)
# calculate average precision score
ap = average_precision_score(yy_test, y_pred[:,1])
print(ap)
#print('auc=%.3f ap=%.3f' % (auc, ap))
# plot no skill
f4 = plt.figure()
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the weighted precision-recall curve for the model
plt.plot( recall_weighted, precision_weighted,marker='.')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Weighted Precision Recall Curve')
# show the plot
plt.show()
```

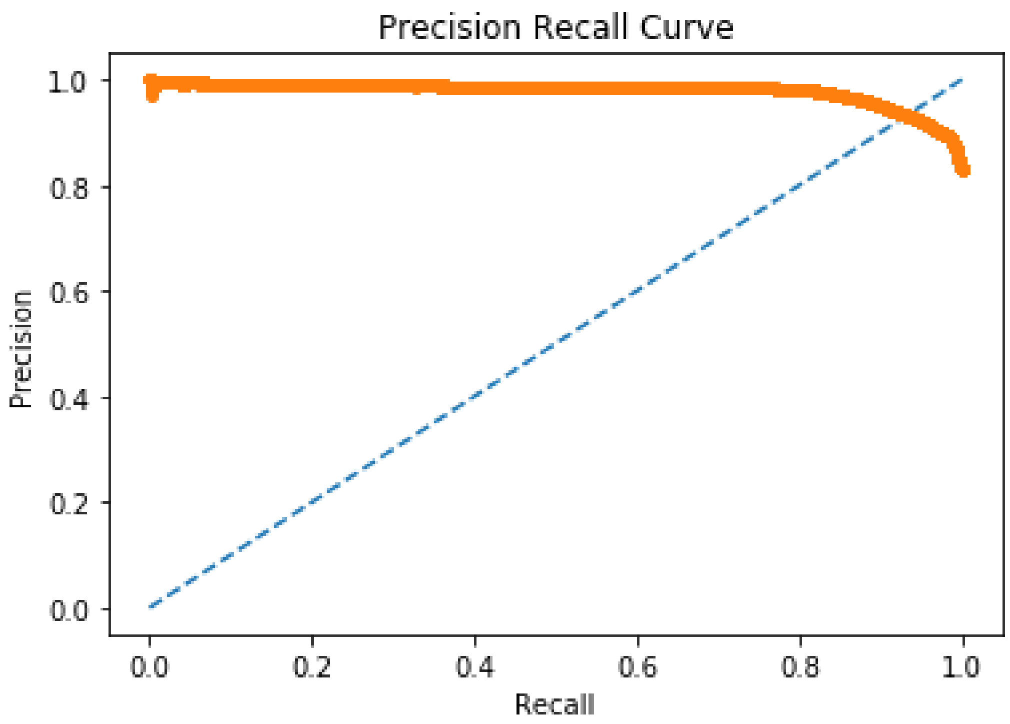
```
f4.savefig("weightedprecisionrecall-cnn1D-MW.pdf", bbox_inches='tight')

best_model.save('cnn1D-MW.h5')
```

```
              precision    recall  f1-score   support

           0       0.92      0.95      0.94     34974
           1       0.72      0.60      0.65      7193

    accuracy                           0.89     42167
   macro avg       0.82      0.78      0.80     42167
weighted avg       0.89      0.89      0.89     42167

[[33318  1656]
 [ 2893  4300]]
0.9223222969606681
```

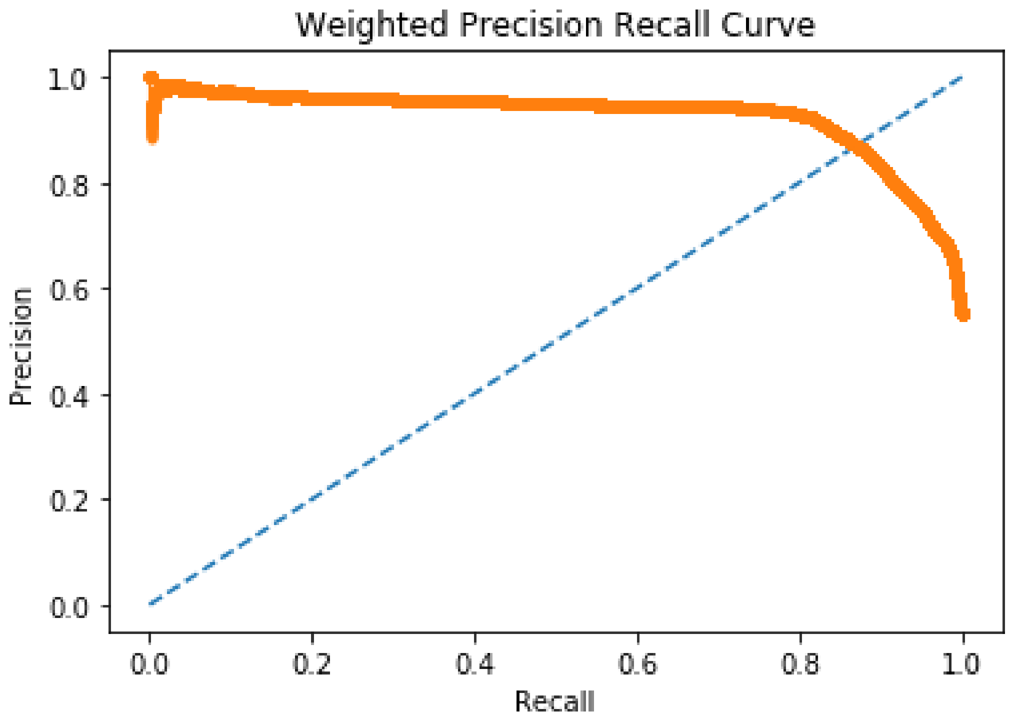




```
0.9792048339203074
0.7290148652784303
```



```
0.9265171965991505
0.7290148652784303
```



1.6 Save Model Analysis Data

[7]:
```
d = {'False Positive Rate': fpr_keras, 'True Positive Rate': tpr_keras ,␣
 ↪'Thresholds': thresholds_keras}
```

[8]:
```
roc_CNN1D = pd.DataFrame(data=d)
```

[9]:
```
roc_CNN1D.to_csv(path_or_buf ='rocCNN1D-MW.csv', index=False)
```

[10]:
```
conf = confusion_matrix(yy_test, yy_pred)
```

[11]:
```
conf1D=pd.DataFrame(data=conf)
```

[12]:
```
conf1D.to_csv(path_or_buf='ConfusionCNN1D-MW.csv',index=False)
```

[13]:
```
pd.DataFrame({"precision" : precision, "recall" :recall}).
 ↪to_csv("precisionrecall-1D-MW.csv", index=None)
```

[14]:
```
pd.DataFrame({"precision" : precision_weighted, "recall" :recall_weighted}).
 ↪to_csv("weightedprecisionrecall-1D-MW.csv", index=None)
```