BeFree-3class-gru

September 18, 2024

1 Evaluation using the BeFree corpus

1.0.1 GAD dataset

To obtain a large benchmark of Gene Disease Associations along with associated sentences from literature, we used the corpus generated by BeFree system based on Genetic Association Database (GAD)

2

3 imports

[1]:
```
import tensorflow as tf
import keras
from keras.models import load_model
from keras.callbacks import ModelCheckpoint, EarlyStopping
from keras_tqdm import TQDMNotebookCallback
import numpy as np
np.random.seed(1337)
from keras_tqdm import TQDMNotebookCallback
import nltk
import xml.etree.ElementTree as ET
import pandas as pd
import os
import string
from nltk.tokenize import TreebankWordTokenizer
from numpy.random import random_sample
import re
import pickle
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

from keras.layers import Embedding, Flatten,LSTM, GRU
from keras.layers.convolutional import Conv2D, MaxPooling2D
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential, Model
```

```
from keras.layers import Dense, Dropout, Activation,  Input,␣
↪merge,Conv1D,MaxPooling1D,GlobalMaxPooling1D,Convolution1D
from keras import regularizers
from sklearn.metrics import precision_recall_fscore_support
from sklearn.model_selection import StratifiedKFold
import matplotlib.pyplot as plt
from keras.layers import Concatenate, concatenate
from keras import backend as K
from keras.layers import multiply
from keras.layers import merge
from keras.layers.core import *
from keras.layers.recurrent import LSTM
from keras.models import *
random_seed=1337
```

```
Using TensorFlow backend.
```

3.0.1 Define Callback functions to generate Measures

[2]:
```
from keras import backend as K

def f1(y_true, y_pred):
    def recall(y_true, y_pred):
        true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
        possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
        recall = true_positives / (possible_positives + K.epsilon())
        return recall

    def precision(y_true, y_pred):
        true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
        predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
        precision = true_positives / (predicted_positives + K.epsilon())
        return precision
    precision = precision(y_true, y_pred)
    recall = recall(y_true, y_pred)
    return 2*((precision*recall)/(precision+recall+K.epsilon()))
```

4 Experiments to reproduce the results of Table 7

4.0.1 Evaluation results for multi-class classification

4.0.2 Load Prerocssed Data

[3]:
```
with open('../data/pickles/befree_3class_crawl-300d-2M.pickle', 'rb') as handle:
    gene_id_list = pickle.load(handle)
    gene_symbol_list = pickle.load(handle)
    disease_id_list = pickle.load(handle)
    X_train = pickle.load(handle)
```

```
    distance1_vectors = pickle.load(handle)
    distance2_vectors = pickle.load(handle)
    Y_train = pickle.load(handle)
    word_list = pickle.load(handle)
    word_vectors = pickle.load(handle)
    word_dict = pickle.load(handle)
    distance1_dict = pickle.load(handle)
    distance2_dict = pickle.load(handle)
    label_dict = pickle.load(handle)
    MAX_SEQUENCE_LENGTH = pickle.load(handle)
print ("word_vectors",len(word_vectors))
```

```
word_vectors 6766
```

4.0.3 Create Position Embedding Vectors

[4]:
```
import keras
from keras_pos_embd import TrigPosEmbedding

model = keras.models.Sequential()
model.add(TrigPosEmbedding(
    input_shape=(None,),
    output_dim=20,                      # The dimension of embeddings.
    mode=TrigPosEmbedding.MODE_EXPAND,  # Use `expand` mode
    name='Pos-Embd',
))
model.compile('adam', keras.losses.mae, {})


d1_train_embedded=model.predict(distance1_vectors)

d1_train_embedded.shape

d2_train_embedded=model.predict(distance2_vectors)

d2_train_embedded.shape
```

[4]:
```
(5330, 81, 20)
```

4.0.4 Prepare Word Embedding Layer

[5]:
```
EMBEDDING_DIM=word_vectors.shape[1]
print("EMBEDDING_DIM=",EMBEDDING_DIM)
embedding_matrix=word_vectors

def create_embedding_layer(l2_reg=0.01,use_pretrained=True,is_trainable=False):

    if use_pretrained:
```

```
        return Embedding(len(word_dict)␣
↪,EMBEDDING_DIM,weights=[embedding_matrix],input_length=MAX_SEQUENCE_LENGTH,trainable=is_tra
↪l2(l2_reg))

    else:
        return Embedding(len(word_dict)␣
↪,EMBEDDING_DIM,input_length=MAX_SEQUENCE_LENGTH)
```

```
EMBEDDING_DIM= 300
```

4.0.5 Prepare Attention Mechanism

[6]:

```
INPUT_DIM = 2
TIME_STEPS = MAX_SEQUENCE_LENGTH
def attentionNew(inputs):
    inputs = Lambda(lambda x: tf.keras.backend.tanh(x))(inputs)
    input_dim = int(inputs.shape[2])
    a = Permute((2, 1))(inputs)
    a = Dense(TIME_STEPS, activation='softmax')(a)
    a_probs = Permute((2, 1))(a)
    output_attention_mul = multiply([inputs, a_probs])
    output_attention_mul = Lambda(lambda x: tf.keras.backend.
↪tanh(x))(output_attention_mul)
    return output_attention_mul
```

4.0.6 Create the Model

[7]:

```
# set parameter for metric calculation, 'macro' for multiclass classification
param='macro'
def build_model():

    sequence_input = Input(shape=(MAX_SEQUENCE_LENGTH,), dtype='int32')
  ␣
↪embedding_layer=create_embedding_layer(use_pretrained=True,is_trainable=False)
    embedded_sequences = embedding_layer(sequence_input)


    pos_embedd_1=Input(shape=(MAX_SEQUENCE_LENGTH,20), dtype='float32')
    pos_embedd_2=Input(shape=(MAX_SEQUENCE_LENGTH,20), dtype='float32')

    embedded_sequences =␣
↪concatenate([embedded_sequences,pos_embedd_1,pos_embedd_2])


    x = Conv1D(32, 5, activation='relu')(embedded_sequences)
```

```
    x = MaxPooling1D(3)(x)
    x = Dropout(0.1)(x)
    conv_sequence_w5=GlobalMaxPooling1D()(x)    #x = Flatten()(x)



    x = Conv1D(64, 3, activation='relu')(embedded_sequences)
    x = MaxPooling1D(3)(x)
    x = Dropout(0.1)(x)
    conv_sequence_w4=GlobalMaxPooling1D()(x)    #x = Flatten()(x)



    x = Conv1D(128, 3, activation='relu')(embedded_sequences)
    x = MaxPooling1D(3)(x)
    x = Dropout(0.1)(x)
    conv_sequence_w3=GlobalMaxPooling1D()(x)    #x = Flatten()(x)

    forward = GRU(100, recurrent_dropout=0.
↪05,return_sequences=True)(embedded_sequences)
    backward = GRU(100, go_backwards=True,recurrent_dropout=0.
↪05,return_sequences=True)(embedded_sequences)
    attention_forward=attentionNew(forward)
    attention_backward=attentionNew(backward)
    lstm_sequence = concatenate([attention_forward,attention_backward])

    lstm_sequence = Flatten()(lstm_sequence)
    merge =␣
↪concatenate([conv_sequence_w5,conv_sequence_w4,conv_sequence_w3,lstm_sequence])
    x = Dense(64, activation='relu', kernel_regularizer=regularizers.l2(0.
↪1))(merge)
    x = Dropout(0.1)(x)
    preds = Dense(3, activation='softmax')(x)
    model = Model(inputs=[sequence_input,␣
↪pos_embedd_1,pos_embedd_2],outputs=preds)
    opt=tf.keras.optimizers.Adam()
    model.
↪compile(loss='categorical_crossentropy',optimizer=opt,metrics=['acc',f1])
    return model
```

[8]:
```
model = build_model()
model.summary()
```

```
Model: "model_1"
________________________________________________________________________________
__________________
```

```
Layer (type)                    Output Shape         Param #     Connected to
================================================================================
==================
input_1 (InputLayer)            (None, 81)           0
--------------------------------------------------------------------------------
------------------
embedding_1 (Embedding)         (None, 81, 300)      2029800     input_1[0][0]
--------------------------------------------------------------------------------
------------------
input_2 (InputLayer)            (None, 81, 20)       0
--------------------------------------------------------------------------------
------------------
input_3 (InputLayer)            (None, 81, 20)       0
--------------------------------------------------------------------------------
------------------
concatenate_1 (Concatenate)     (None, 81, 340)      0
embedding_1[0][0]
                                                                 input_2[0][0]
                                                                 input_3[0][0]
--------------------------------------------------------------------------------
------------------
gru_1 (GRU)                     (None, 81, 100)      132300
concatenate_1[0][0]
--------------------------------------------------------------------------------
------------------
gru_2 (GRU)                     (None, 81, 100)      132300
concatenate_1[0][0]
--------------------------------------------------------------------------------
------------------
lambda_1 (Lambda)               (None, 81, 100)      0           gru_1[0][0]
--------------------------------------------------------------------------------
------------------
lambda_3 (Lambda)               (None, 81, 100)      0           gru_2[0][0]
--------------------------------------------------------------------------------
------------------
permute_1 (Permute)             (None, 100, 81)      0           lambda_1[0][0]
--------------------------------------------------------------------------------
------------------
permute_3 (Permute)             (None, 100, 81)      0           lambda_3[0][0]
--------------------------------------------------------------------------------
------------------
dense_1 (Dense)                 (None, 100, 81)      6642        permute_1[0][0]
--------------------------------------------------------------------------------
------------------
dense_2 (Dense)                 (None, 100, 81)      6642        permute_3[0][0]
--------------------------------------------------------------------------------
------------------
permute_2 (Permute)             (None, 81, 100)      0           dense_1[0][0]
```

```
____________________________________________________________________________________________________
permute_4 (Permute)              (None, 81, 100)      0           dense_2[0][0]
____________________________________________________________________________________________________
conv1d_1 (Conv1D)                (None, 77, 32)       54432
concatenate_1[0][0]
____________________________________________________________________________________________________
conv1d_2 (Conv1D)                (None, 79, 64)       65344
concatenate_1[0][0]
____________________________________________________________________________________________________
conv1d_3 (Conv1D)                (None, 79, 128)      130688
concatenate_1[0][0]
____________________________________________________________________________________________________
multiply_1 (Multiply)            (None, 81, 100)      0           lambda_1[0][0]
                                                                  permute_2[0][0]
____________________________________________________________________________________________________
multiply_2 (Multiply)            (None, 81, 100)      0           lambda_3[0][0]
                                                                  permute_4[0][0]
____________________________________________________________________________________________________
max_pooling1d_1 (MaxPooling1D)   (None, 25, 32)       0           conv1d_1[0][0]
____________________________________________________________________________________________________
max_pooling1d_2 (MaxPooling1D)   (None, 26, 64)       0           conv1d_2[0][0]
____________________________________________________________________________________________________
max_pooling1d_3 (MaxPooling1D)   (None, 26, 128)      0           conv1d_3[0][0]
____________________________________________________________________________________________________
lambda_2 (Lambda)                (None, 81, 100)      0
multiply_1[0][0]
____________________________________________________________________________________________________
lambda_4 (Lambda)                (None, 81, 100)      0
multiply_2[0][0]
____________________________________________________________________________________________________
dropout_1 (Dropout)              (None, 25, 32)       0
max_pooling1d_1[0][0]
____________________________________________________________________________________________________
dropout_2 (Dropout)              (None, 26, 64)       0
max_pooling1d_2[0][0]
```

```
__________________________________________________________________________________________________
dropout_3 (Dropout)             (None, 26, 128)      0
max_pooling1d_3[0][0]
__________________________________________________________________________________________________
concatenate_2 (Concatenate)     (None, 81, 200)      0           lambda_2[0][0]
                                                                 lambda_4[0][0]
__________________________________________________________________________________________________
global_max_pooling1d_1 (GlobalM (None, 32)           0           dropout_1[0][0]
__________________________________________________________________________________________________
global_max_pooling1d_2 (GlobalM (None, 64)           0           dropout_2[0][0]
__________________________________________________________________________________________________
global_max_pooling1d_3 (GlobalM (None, 128)          0           dropout_3[0][0]
__________________________________________________________________________________________________
flatten_1 (Flatten)             (None, 16200)        0
concatenate_2[0][0]
__________________________________________________________________________________________________
concatenate_3 (Concatenate)     (None, 16424)        0
global_max_pooling1d_1[0][0]
global_max_pooling1d_2[0][0]
global_max_pooling1d_3[0][0]
                                                                 flatten_1[0][0]
__________________________________________________________________________________________________
dense_3 (Dense)                 (None, 64)           1051200
concatenate_3[0][0]
__________________________________________________________________________________________________
dropout_4 (Dropout)             (None, 64)           0           dense_3[0][0]
__________________________________________________________________________________________________
dense_4 (Dense)                 (None, 3)            195         dropout_4[0][0]
==================================================================================================
Total params: 3,609,543
Trainable params: 1,579,743
Non-trainable params: 2,029,800
__________________________________________________________________________________________________
```

[9]:
```python
validation_split_rate = 0.1
skf = StratifiedKFold(n_splits=5,shuffle=True, random_state=42)
Y = [np.argmax(y, axis=None, out=None) for y in Y_train]
for tr_index, te_index in skf.split(X_train,Y):
    test_index = te_index
    train_index = tr_index

trainRate = (len(train_index)/len(Y))*100
testRate = (len(test_index)/len(Y))*100
print ("TrainRate:{:.2f}% testRate:{:.2f}% validation:{:.2f}%  ".
 ↪format(trainRate,testRate, trainRate*validation_split_rate))
X_train, X_test = X_train[train_index], X_train[test_index]
pos_train1, pos_test1 = d1_train_embedded[train_index],␣
 ↪d1_train_embedded[test_index]
pos_train2, pos_test2 = d2_train_embedded[train_index],␣
 ↪d2_train_embedded[test_index]
y_train, y_test = Y_train[train_index], Y_train[test_index]



# # Saving the training data split as a pickle file
# training_data = {
#     'X_train': X_train,
#     'pos_train1': pos_train1,
#     'pos_train2': pos_train2,
#     'y_train': y_train
# }

# with open('training_data12.pkl', 'wb') as f:
#     pickle.dump(training_data, f)

# # Saving the testing data split as a pickle file
# testing_data = {
#     'X_test': X_test,
#     'pos_test1': pos_test1,
#     'pos_test2': pos_test2,
#     'y_test': y_test
# }

# with open('testing_data12.pkl', 'wb') as f:
#     pickle.dump(testing_data, f)
```

```
TrainRate:80.00% testRate:20.00% validation:8.00%
```

[10]:
```python
# Load the training data from the pickle file
with open('training_data.pkl', 'rb') as f:
```

```
    train_data = pickle.load(f)

# Load the testing data from the pickle file
with open('testing_data.pkl', 'rb') as f:
    test_data = pickle.load(f)

# Extract data from the loaded dictionaries
X_train = train_data['X_train']
pos_train1 = train_data['pos_train1']
pos_train2 = train_data['pos_train2']
y_train = train_data['y_train']


X_test = test_data['X_test']
pos_test1 = test_data['pos_test1']
pos_test2 = test_data['pos_test2']
y_test = test_data['y_test']
print(X_train.shape)
print(X_test.shape)
```

```
(4264, 81)
(1066, 81)
```

4.0.7 Run the Evaluation using 10 fold Cross Validation

[11]:
```
MaxEpochs =50
batchsize =32
validation_split_rate = 0.1
history=model.fit([X_train,pos_train1,pos_train2],␣
 ↪y_train,validation_split=validation_split_rate ,epochs=MaxEpochs,␣
 ↪batch_size=batchsize,verbose=1)
```

```
Train on 3837 samples, validate on 427 samples
Epoch 1/50
3837/3837 [==============================] - 87s 23ms/step - loss: 2780.3121 -
acc: 0.4454 - f1: 0.1384 - val_loss: 2779.7211 - val_acc: 0.4333 - val_f1:
0.0397
Epoch 2/50
3837/3837 [==============================] - 21s 5ms/step - loss: 2779.6360 -
acc: 0.4699 - f1: 0.1608 - val_loss: 2779.6530 - val_acc: 0.4333 - val_f1:
0.1856
Epoch 3/50
3837/3837 [==============================] - 21s 5ms/step - loss: 2779.5958 -
acc: 0.4788 - f1: 0.2140 - val_loss: 2779.6675 - val_acc: 0.4450 - val_f1:
```

```
0.0709
Epoch 4/50
3837/3837 [==============================] - 21s 5ms/step - loss: 2779.5398 -
acc: 0.4827 - f1: 0.2854 - val_loss: 2779.7274 - val_acc: 0.5012 - val_f1:
0.3381
Epoch 5/50
3837/3837 [==============================] - 21s 5ms/step - loss: 2779.4688 -
acc: 0.5442 - f1: 0.4584 - val_loss: 2779.6347 - val_acc: 0.4567 - val_f1:
0.4231
Epoch 6/50
3837/3837 [==============================] - 21s 5ms/step - loss: 2779.4364 -
acc: 0.5603 - f1: 0.5021 - val_loss: 2779.5545 - val_acc: 0.5363 - val_f1:
0.5024
Epoch 7/50
3837/3837 [==============================] - 21s 6ms/step - loss: 2779.4069 -
acc: 0.6185 - f1: 0.5916 - val_loss: 2779.6308 - val_acc: 0.5480 - val_f1:
0.5146
Epoch 8/50
3837/3837 [==============================] - 21s 5ms/step - loss: 2779.3545 -
acc: 0.6706 - f1: 0.6566 - val_loss: 2779.5228 - val_acc: 0.5386 - val_f1:
0.5019
Epoch 9/50
3837/3837 [==============================] - 21s 5ms/step - loss: 2779.3112 -
acc: 0.7047 - f1: 0.6950 - val_loss: 2779.6049 - val_acc: 0.6136 - val_f1:
0.5956
Epoch 10/50
3837/3837 [==============================] - 21s 6ms/step - loss: 2779.2644 -
acc: 0.7415 - f1: 0.7366 - val_loss: 2779.6261 - val_acc: 0.5504 - val_f1:
0.5353
Epoch 11/50
3837/3837 [==============================] - 23s 6ms/step - loss: 2779.2131 -
acc: 0.7709 - f1: 0.7635 - val_loss: 2779.4703 - val_acc: 0.6253 - val_f1:
0.6110
Epoch 12/50
3837/3837 [==============================] - 21s 6ms/step - loss: 2779.1677 -
acc: 0.7965 - f1: 0.7918 - val_loss: 2779.5100 - val_acc: 0.6183 - val_f1:
0.6134
Epoch 13/50
3837/3837 [==============================] - 21s 6ms/step - loss: 2779.1289 -
acc: 0.8144 - f1: 0.8134 - val_loss: 2779.4898 - val_acc: 0.6323 - val_f1:
0.6286
Epoch 14/50
3837/3837 [==============================] - 21s 5ms/step - loss: 2779.1118 -
acc: 0.8210 - f1: 0.8216 - val_loss: 2779.7811 - val_acc: 0.5761 - val_f1:
0.5672
Epoch 15/50
3837/3837 [==============================] - 21s 6ms/step - loss: 2779.0709 -
acc: 0.8455 - f1: 0.8457 - val_loss: 2779.5833 - val_acc: 0.6393 - val_f1:
```

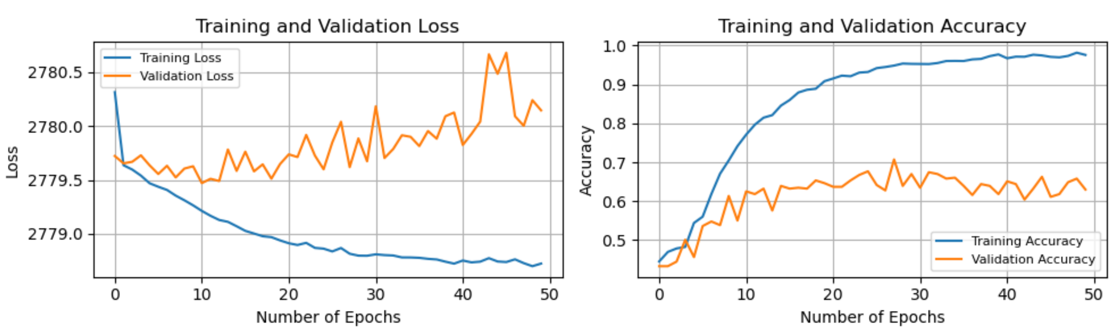
```
0.6329
Epoch 16/50
3837/3837 [==============================] - 21s 5ms/step - loss: 2779.0282 -
acc: 0.8598 - f1: 0.8580 - val_loss: 2779.7611 - val_acc: 0.6323 - val_f1:
0.6300
Epoch 17/50
3837/3837 [==============================] - 21s 6ms/step - loss: 2779.0035 -
acc: 0.8791 - f1: 0.8780 - val_loss: 2779.5790 - val_acc: 0.6347 - val_f1:
0.6281
Epoch 18/50
3837/3837 [==============================] - 21s 5ms/step - loss: 2778.9785 -
acc: 0.8861 - f1: 0.8858 - val_loss: 2779.6440 - val_acc: 0.6323 - val_f1:
0.6332
Epoch 19/50
3837/3837 [==============================] - 22s 6ms/step - loss: 2778.9699 -
acc: 0.8887 - f1: 0.8867 - val_loss: 2779.5114 - val_acc: 0.6534 - val_f1:
0.6472
Epoch 20/50
3837/3837 [==============================] - 21s 5ms/step - loss: 2778.9397 -
acc: 0.9080 - f1: 0.9064 - val_loss: 2779.6474 - val_acc: 0.6464 - val_f1:
0.6400
Epoch 21/50
3837/3837 [==============================] - 21s 5ms/step - loss: 2778.9134 -
acc: 0.9150 - f1: 0.9139 - val_loss: 2779.7363 - val_acc: 0.6370 - val_f1:
0.6327
Epoch 22/50
3837/3837 [==============================] - 21s 6ms/step - loss: 2778.8974 -
acc: 0.9223 - f1: 0.9212 - val_loss: 2779.7114 - val_acc: 0.6370 - val_f1:
0.6398
Epoch 23/50
3837/3837 [==============================] - 21s 5ms/step - loss: 2778.9171 -
acc: 0.9208 - f1: 0.9202 - val_loss: 2779.9166 - val_acc: 0.6534 - val_f1:
0.6350
Epoch 24/50
3837/3837 [==============================] - 21s 5ms/step - loss: 2778.8707 -
acc: 0.9302 - f1: 0.9302 - val_loss: 2779.7252 - val_acc: 0.6674 - val_f1:
0.6565
Epoch 25/50
3837/3837 [==============================] - 21s 5ms/step - loss: 2778.8636 -
acc: 0.9317 - f1: 0.9309 - val_loss: 2779.5980 - val_acc: 0.6768 - val_f1:
0.6812
Epoch 26/50
3837/3837 [==============================] - 21s 5ms/step - loss: 2778.8377 -
acc: 0.9416 - f1: 0.9413 - val_loss: 2779.8469 - val_acc: 0.6417 - val_f1:
0.6401
Epoch 27/50
3837/3837 [==============================] - 21s 5ms/step - loss: 2778.8694 -
acc: 0.9445 - f1: 0.9445 - val_loss: 2780.0383 - val_acc: 0.6276 - val_f1:
```

```
0.6256
Epoch 28/50
3837/3837 [==============================] - 22s 6ms/step - loss: 2778.8163 -
acc: 0.9481 - f1: 0.9484 - val_loss: 2779.6181 - val_acc: 0.7073 - val_f1:
0.7038
Epoch 29/50
3837/3837 [==============================] - 21s 6ms/step - loss: 2778.7988 -
acc: 0.9533 - f1: 0.9529 - val_loss: 2779.8841 - val_acc: 0.6393 - val_f1:
0.6300
Epoch 30/50
3837/3837 [==============================] - 22s 6ms/step - loss: 2778.7978 -
acc: 0.9526 - f1: 0.9523 - val_loss: 2779.6721 - val_acc: 0.6698 - val_f1:
0.6715
Epoch 31/50
3837/3837 [==============================] - 21s 6ms/step - loss: 2778.8109 -
acc: 0.9523 - f1: 0.9527 - val_loss: 2780.1814 - val_acc: 0.6347 - val_f1:
0.6350
Epoch 32/50
3837/3837 [==============================] - 21s 5ms/step - loss: 2778.8038 -
acc: 0.9520 - f1: 0.9520 - val_loss: 2779.7025 - val_acc: 0.6745 - val_f1:
0.6754
Epoch 33/50
3837/3837 [==============================] - 21s 6ms/step - loss: 2778.8002 -
acc: 0.9547 - f1: 0.9547 - val_loss: 2779.7859 - val_acc: 0.6698 - val_f1:
0.6690
Epoch 34/50
3837/3837 [==============================] - 21s 5ms/step - loss: 2778.7823 -
acc: 0.9599 - f1: 0.9592 - val_loss: 2779.9133 - val_acc: 0.6581 - val_f1:
0.6506
Epoch 35/50
3837/3837 [==============================] - 21s 5ms/step - loss: 2778.7819 -
acc: 0.9601 - f1: 0.9594 - val_loss: 2779.8988 - val_acc: 0.6604 - val_f1:
0.6692
Epoch 36/50
3837/3837 [==============================] - 21s 5ms/step - loss: 2778.7786 -
acc: 0.9599 - f1: 0.9603 - val_loss: 2779.8131 - val_acc: 0.6393 - val_f1:
0.6195
Epoch 37/50
3837/3837 [==============================] - 21s 5ms/step - loss: 2778.7692 -
acc: 0.9640 - f1: 0.9651 - val_loss: 2779.9524 - val_acc: 0.6159 - val_f1:
0.6145
Epoch 38/50
3837/3837 [==============================] - 21s 5ms/step - loss: 2778.7633 -
acc: 0.9653 - f1: 0.9668 - val_loss: 2779.8813 - val_acc: 0.6440 - val_f1:
0.6390
Epoch 39/50
3837/3837 [==============================] - 21s 5ms/step - loss: 2778.7430 -
acc: 0.9724 - f1: 0.9723 - val_loss: 2780.0892 - val_acc: 0.6393 - val_f1:
```

```
0.6354
Epoch 40/50
3837/3837 [==============================] - 21s 5ms/step - loss: 2778.7247 -
acc: 0.9768 - f1: 0.9771 - val_loss: 2780.1236 - val_acc: 0.6183 - val_f1:
0.6168
Epoch 41/50
3837/3837 [==============================] - 21s 5ms/step - loss: 2778.7535 -
acc: 0.9669 - f1: 0.9679 - val_loss: 2779.8234 - val_acc: 0.6511 - val_f1:
0.6406
Epoch 42/50
3837/3837 [==============================] - 21s 5ms/step - loss: 2778.7365 -
acc: 0.9711 - f1: 0.9718 - val_loss: 2779.9257 - val_acc: 0.6440 - val_f1:
0.6478
Epoch 43/50
3837/3837 [==============================] - 21s 5ms/step - loss: 2778.7431 -
acc: 0.9708 - f1: 0.9711 - val_loss: 2780.0395 - val_acc: 0.6042 - val_f1:
0.5988
Epoch 44/50
3837/3837 [==============================] - 21s 5ms/step - loss: 2778.7754 -
acc: 0.9760 - f1: 0.9762 - val_loss: 2780.6627 - val_acc: 0.6323 - val_f1:
0.6211
Epoch 45/50
3837/3837 [==============================] - 21s 5ms/step - loss: 2778.7446 -
acc: 0.9742 - f1: 0.9740 - val_loss: 2780.4818 - val_acc: 0.6628 - val_f1:
0.6643
Epoch 46/50
3837/3837 [==============================] - 21s 5ms/step - loss: 2778.7396 -
acc: 0.9705 - f1: 0.9707 - val_loss: 2780.6776 - val_acc: 0.6112 - val_f1:
0.6029
Epoch 47/50
3837/3837 [==============================] - 21s 5ms/step - loss: 2778.7645 -
acc: 0.9692 - f1: 0.9695 - val_loss: 2780.0895 - val_acc: 0.6183 - val_f1:
0.6127
Epoch 48/50
3837/3837 [==============================] - 21s 5ms/step - loss: 2778.7295 -
acc: 0.9729 - f1: 0.9744 - val_loss: 2780.0008 - val_acc: 0.6487 - val_f1:
0.6466
Epoch 49/50
3837/3837 [==============================] - 21s 5ms/step - loss: 2778.7011 -
acc: 0.9812 - f1: 0.9816 - val_loss: 2780.2382 - val_acc: 0.6581 - val_f1:
0.6499
Epoch 50/50
3837/3837 [==============================] - 20s 5ms/step - loss: 2778.7253 -
acc: 0.9755 - f1: 0.9758 - val_loss: 2780.1437 - val_acc: 0.6300 - val_f1:
0.6374
```

[12]:

```python
import matplotlib.pyplot as plt

# Training & Validation accuracy
train_loss = history.history['loss']
val_loss = history.history['val_loss']
train_acc = history.history['acc']
val_acc = history.history['val_acc']
epochs = len(train_loss)

xc = range(epochs)

plt.figure(figsize=(10, 3))

# Loss subplot
plt.subplot(1, 2, 1)
plt.plot(xc, train_loss, label='Training Loss')
plt.plot(xc, val_loss, label='Validation Loss')
plt.xlabel('Number of Epochs', fontsize=10)
plt.ylabel('Loss', fontsize=10)
plt.title('Training and Validation Loss', fontsize=12)
plt.legend(fontsize=8)
plt.grid(True)

# Accuracy subplot
plt.subplot(1, 2, 2)
plt.plot(xc, train_acc, label='Training Accuracy')
plt.plot(xc, val_acc, label='Validation Accuracy')
plt.xlabel('Number of Epochs', fontsize=10)
plt.ylabel('Accuracy', fontsize=10)
plt.title('Training and Validation Accuracy', fontsize=12)
plt.legend(fontsize=8, loc='lower right')  # Change position to lower right
plt.grid(True)

plt.tight_layout()
plt.show()
```



[13]:
```python
import torch
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪precision_recall_fscore_support

# Predict on the training dataset
train_predicted = np.argmax(model.predict([X_train, pos_train1, pos_train2]),␣
 ↪axis=1)
y_train_to_label = np.argmax(y_train, axis=1)

# Calculate accuracy, precision, recall, and F1-score for the training data
train_accuracy = accuracy_score(y_train_to_label, train_predicted)
train_prec, train_reca, train_fscore, _ =␣
 ↪precision_recall_fscore_support(y_train_to_label, train_predicted,␣
 ↪average=param)

# Print the classification report for the training data
print("Training Classification Report:")
print(classification_report(y_train_to_label, train_predicted))

# Print the precision, recall, and F1-score for the training data
print("Training Accuracy: {:.2f}%".format(train_accuracy * 100))
print("Training Precision: {:.2f}%".format(train_prec * 100))
print("Training Recall: {:.2f}%".format(train_reca * 100))
print("Training F1 Score: {:.2f}%".format(train_fscore * 100))
```

```
Training Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.88      0.89      2023
           1       0.86      0.95      0.90      1468
           2       0.86      0.77      0.81       773

    accuracy                           0.88      4264
   macro avg       0.88      0.86      0.87      4264
weighted avg       0.88      0.88      0.88      4264

Training Accuracy: 88.20%
Training Precision: 87.62%
Training Recall: 86.41%
Training F1 Score: 86.85%
```

[14]:
```python
predicted = np.argmax(model.predict([X_test,pos_test1,pos_test2]), axis=1)
y_test_to_label = np.argmax(y_test, axis=1)
prec, reca, fscore, sup = precision_recall_fscore_support(y_test_to_label,␣
 ↪predicted, average=param)
# Generate the classification report as a dictionary
```

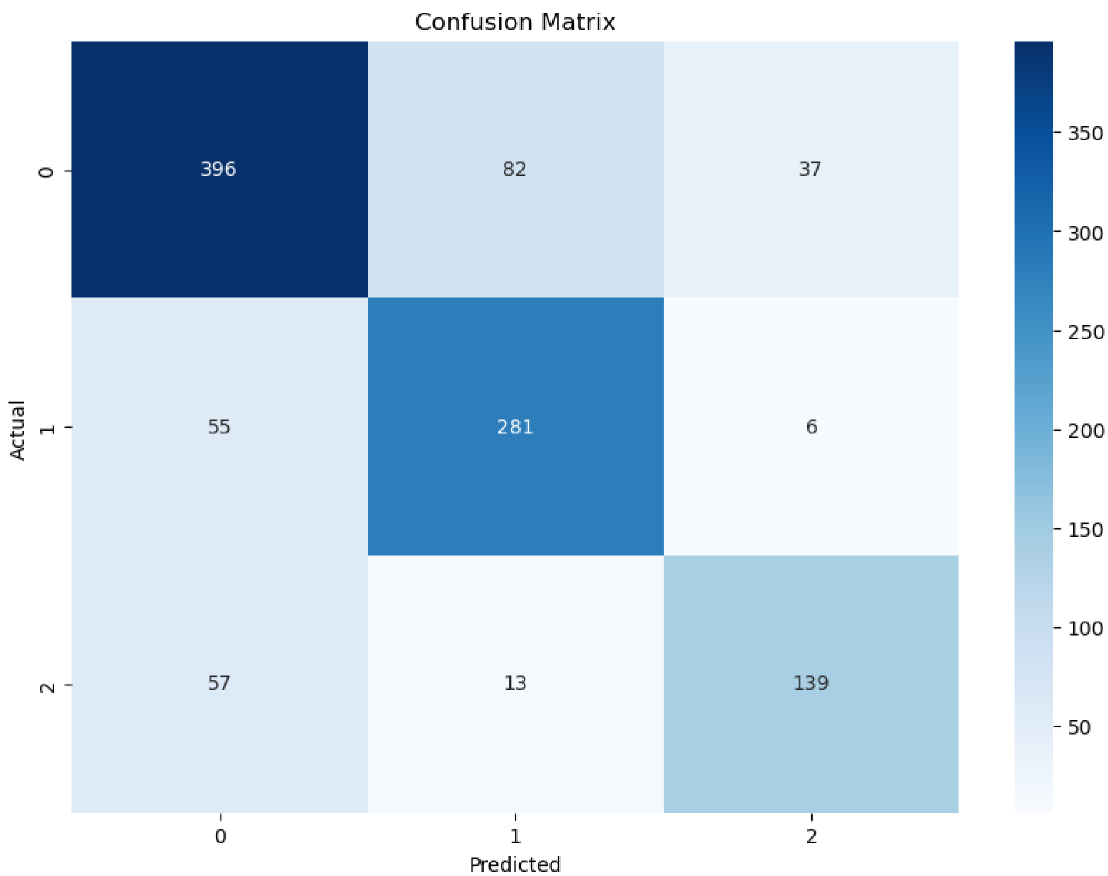
```
report_dict = classification_report(y_test_to_label, predicted,␣
↪output_dict=True)

# Create a new dictionary to hold the formatted values
formatted_report_dict = {}

# Iterate over the items in the report dictionary
for key, value in report_dict.items():
    if isinstance(value, dict):
        # Format the nested dictionary values
        formatted_report_dict[key] = {sub_key: f"{sub_value:.4f}" for sub_key,␣
↪sub_value in value.items()}
    else:
        # Format the top-level dictionary values
        formatted_report_dict[key] = f"{value:.4f}"

# Create a string representation of the formatted dictionary
formatted_report_str = classification_report(y_test_to_label, predicted,␣
↪digits=4)

# Print the formatted classification report
print(formatted_report_str)
print(" Precision:{:.2f}% Recall:{:.2f}% Fscore:{:.2f}% ".format(prec*100,␣
↪reca*100, fscore*100))
```

```
              precision    recall  f1-score   support

           0     0.7795    0.7689    0.7742       515
           1     0.7473    0.8216    0.7827       342
           2     0.7637    0.6651    0.7110       209

    accuracy                         0.7655      1066
   macro avg     0.7635    0.7519    0.7560      1066
weighted avg     0.7661    0.7655    0.7645      1066

 Precision:76.35% Recall:75.19% Fscore:75.60%
```

[15]:

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import classification_report, confusion_matrix,␣
↪precision_recall_fscore_support
from sklearn.model_selection import StratifiedKFold
# Calculate and visualize the confusion matrix
cm = confusion_matrix(y_test_to_label, predicted)
plt.figure(figsize=(10, 7))
```

```python
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['0', '1','2'], yticklabels=['0', '1','2'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

# Print precision, recall, and f-score
prec, reca, fscore, sup = precision_recall_fscore_support(y_test_to_label, predicted, average=param)
print(" Precision:{:.2f}% Recall:{:.2f}% Fscore:{:.2f}% ".format(prec*100, reca*100, fscore*100))
```




```
Precision:76.35% Recall:75.19% Fscore:75.60%
```

[ ]: