modelGRU2-gru

September 18, 2024

1 model 1

sara CNN + GRU + GRU

2 ---

3 imports

[1]:
```
import tensorflow as tf
from keras.models import load_model
from keras.callbacks import ModelCheckpoint, EarlyStopping
from keras_tqdm import TQDMNotebookCallback
import numpy as np
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 keras.utils import to_categorical
from 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 *
```

```
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()))
```

3.0.2 Load pre procssed Data

[3]:
```
with open('../data/pickles/train_and_test_data_sentences_snp_2class.pickle',␣
 ↪'rb') as handle:

    W_train = pickle.load(handle)
    d1_train = pickle.load(handle)
    d2_train = pickle.load(handle)
    Y_train = pickle.load(handle)
    Tr_word_list = pickle.load(handle)

    W_test = pickle.load(handle)
    d1_test = pickle.load(handle)
    d2_test = pickle.load(handle)
    Y_test = pickle.load(handle)
    Te_word_list = pickle.load(handle)
```

```
    word_vectors = pickle.load(handle)
    word_dict = pickle.load(handle)
    d1_dict = pickle.load(handle)
    d2_dict = pickle.load(handle)
    label_dict = pickle.load(handle)
    MAX_SEQUENCE_LENGTH = pickle.load(handle)
```

3.0.3 Prepare Word Embedding Layer

[4]:
```
EMBEDDING_DIM=word_vectors.shape[1]
embedding_matrix=word_vectors

def create_embedding_layer(l2_reg=0.1,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)
```

3.0.4 Create the Model

[5]:
```
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)

    x = Conv1D(256, 7, activation='relu')(embedded_sequences)
    x = MaxPooling1D(3)(x)
    x = Dropout(0.5)(x)

    x = Conv1D(128, 3, activation='relu')(x)
    x = MaxPooling1D(3)(x)
    x = Dropout(0.5)(x)



    conv_sequence=GlobalMaxPooling1D()(x)    #x = Flatten()(x)


```

```
    forward = GRU (80,recurrent_dropout=0.05)(embedded_sequences)
    backward = GRU(80, go_backwards=True,recurrent_dropout=0.
↪05)(embedded_sequences)
    lstm_sequence = concatenate([forward,backward])
    merge = concatenate([conv_sequence,lstm_sequence])
    x = Dense(256, activation='relu', kernel_regularizer=regularizers.l2(0.
↪01))(merge)
    x = Dropout(0.5)(x)
    preds = Dense(2, activation='softmax')(x)
    model = Model(sequence_input, preds)
    model.compile(loss='mean_squared_error',optimizer='adam',metrics=['acc',f1])
    #model.summary()
    return model
```

[6]:
```
model = build_model()
model.summary()
```

```
Model: "model_1"
________________________________________________________________________________
__________________
Layer (type)                    Output Shape         Param #     Connected to
================================================================================
==================
input_1 (InputLayer)            (None, 91)           0
________________________________________________________________________________
__________________
embedding_1 (Embedding)         (None, 91, 200)      555000      input_1[0][0]
________________________________________________________________________________
__________________
conv1d_1 (Conv1D)               (None, 85, 256)      358656
embedding_1[0][0]
________________________________________________________________________________
__________________
max_pooling1d_1 (MaxPooling1D)  (None, 28, 256)      0           conv1d_1[0][0]
________________________________________________________________________________
__________________
dropout_1 (Dropout)             (None, 28, 256)      0
max_pooling1d_1[0][0]
________________________________________________________________________________
__________________
conv1d_2 (Conv1D)               (None, 26, 128)      98432       dropout_1[0][0]
________________________________________________________________________________
__________________
max_pooling1d_2 (MaxPooling1D)  (None, 8, 128)       0           conv1d_2[0][0]
________________________________________________________________________________
__________________
dropout_2 (Dropout)             (None, 8, 128)       0
max_pooling1d_2[0][0]
```

```
______________________________________________________________________________
__________________
gru_1 (GRU)                     (None, 80)          67440
embedding_1[0][0]

______________________________________________________________________________
__________________
gru_2 (GRU)                     (None, 80)          67440
embedding_1[0][0]

______________________________________________________________________________
__________________
global_max_pooling1d_1 (GlobalM (None, 128)         0           dropout_2[0][0]
______________________________________________________________________________
__________________
concatenate_1 (Concatenate)     (None, 160)         0           gru_1[0][0]
                                                                gru_2[0][0]

______________________________________________________________________________
__________________
concatenate_2 (Concatenate)     (None, 288)         0
global_max_pooling1d_1[0][0]
concatenate_1[0][0]

______________________________________________________________________________
__________________
dense_1 (Dense)                 (None, 256)         73984
concatenate_2[0][0]

______________________________________________________________________________
__________________
dropout_3 (Dropout)             (None, 256)         0           dense_1[0][0]
______________________________________________________________________________
__________________
dense_2 (Dense)                 (None, 2)           514         dropout_3[0][0]
==============================================================================
==================
Total params: 1,221,466
Trainable params: 666,466
Non-trainable params: 555,000

______________________________________________________________________________
__________________
```

3.0.5 Run the Evaluation on the test dataset

[7]:
```python
param='macro'
epochs =50
batch_size = 32
# simple early stopping
history=model.fit(W_train,␣
↪Y_train,epochs=epochs,validation_data=(W_test,Y_test),␣
↪batch_size=batch_size,verbose=1,callbacks=[TQDMNotebookCallback()])
```

```
predicted = np.argmax(model.predict(W_test), 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)
print("Precision:{:.2f}% Recall:{:.2f}% Fscore:{:.2f}% ".format(prec*100,␣
 ↪reca*100, fscore*100))
```

```
Train on 935 samples, validate on 365 samples

Training:   0%|          | 0/50 [00:00<?, ?it/s]

Epoch 1/50

Epoch 0:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 294s 315ms/step - loss: 5011.4001 -
acc: 0.7230 - f1: 0.7228 - val_loss: 5010.8479 - val_acc: 0.6712 - val_f1:
0.6799
Epoch 2/50

Epoch 1:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 6ms/step - loss: 5010.3759 - acc:
0.7412 - f1: 0.7368 - val_loss: 5010.1164 - val_acc: 0.6712 - val_f1: 0.6799
Epoch 3/50

Epoch 2:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 5s 6ms/step - loss: 5009.8826 - acc:
0.7465 - f1: 0.7531 - val_loss: 5009.7907 - val_acc: 0.6795 - val_f1: 0.6877
Epoch 4/50

Epoch 3:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 6ms/step - loss: 5009.6459 - acc:
0.7722 - f1: 0.7707 - val_loss: 5009.5914 - val_acc: 0.6986 - val_f1: 0.7059
Epoch 5/50

Epoch 4:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 6ms/step - loss: 5009.4571 - acc:
0.8193 - f1: 0.8202 - val_loss: 5009.4373 - val_acc: 0.7644 - val_f1: 0.7494
Epoch 6/50

Epoch 5:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 6ms/step - loss: 5009.4127 - acc:
0.7947 - f1: 0.7851 - val_loss: 5009.5163 - val_acc: 0.6712 - val_f1: 0.6799
Epoch 7/50

Epoch 6:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 6ms/step - loss: 5009.3668 - acc:
0.8193 - f1: 0.8202 - val_loss: 5009.3728 - val_acc: 0.7726 - val_f1: 0.7762
Epoch 8/50
```

```
Epoch 7:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 6ms/step - loss: 5009.3045 - acc:
0.8439 - f1: 0.8479 - val_loss: 5009.3182 - val_acc: 0.8027 - val_f1: 0.8125
Epoch 9/50

Epoch 8:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 6ms/step - loss: 5009.2520 - acc:
0.8770 - f1: 0.8765 - val_loss: 5009.2858 - val_acc: 0.8164 - val_f1: 0.8255
Epoch 10/50

Epoch 9:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 6ms/step - loss: 5009.2686 - acc:
0.8567 - f1: 0.8604 - val_loss: 5009.2735 - val_acc: 0.7890 - val_f1: 0.7995
Epoch 11/50

Epoch 10:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 6ms/step - loss: 5009.2427 - acc:
0.8610 - f1: 0.8609 - val_loss: 5009.2809 - val_acc: 0.8082 - val_f1: 0.8177
Epoch 12/50

Epoch 11:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 5s 6ms/step - loss: 5009.2306 - acc:
0.8770 - f1: 0.8802 - val_loss: 5009.2815 - val_acc: 0.7918 - val_f1: 0.7983
Epoch 13/50

Epoch 12:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 6ms/step - loss: 5009.1927 - acc:
0.9059 - f1: 0.9046 - val_loss: 5009.2724 - val_acc: 0.7945 - val_f1: 0.7971
Epoch 14/50

Epoch 13:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 6ms/step - loss: 5009.1930 - acc:
0.9059 - f1: 0.9083 - val_loss: 5009.2510 - val_acc: 0.8192 - val_f1: 0.8281
Epoch 15/50

Epoch 14:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 6ms/step - loss: 5009.1997 - acc:
0.8909 - f1: 0.8900 - val_loss: 5009.2416 - val_acc: 0.8164 - val_f1: 0.8255
Epoch 16/50

Epoch 15:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 6ms/step - loss: 5009.1849 - acc:
0.9037 - f1: 0.9062 - val_loss: 5009.2710 - val_acc: 0.7863 - val_f1: 0.7893
Epoch 17/50

Epoch 16:   0%|          | 0/935 [00:00<?, ?it/s]
```

```
935/935 [==============================] - 6s 6ms/step - loss: 5009.1692 - acc:
0.9176 - f1: 0.9161 - val_loss: 5009.2201 - val_acc: 0.8356 - val_f1: 0.8437
Epoch 18/50

Epoch 17:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 6ms/step - loss: 5009.1906 - acc:
0.8663 - f1: 0.8661 - val_loss: 5009.2271 - val_acc: 0.8027 - val_f1: 0.8125
Epoch 19/50

Epoch 18:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 6ms/step - loss: 5009.1877 - acc:
0.8845 - f1: 0.8875 - val_loss: 5009.2210 - val_acc: 0.8110 - val_f1: 0.8203
Epoch 20/50

Epoch 19:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 6ms/step - loss: 5009.1667 - acc:
0.9187 - f1: 0.9208 - val_loss: 5009.2050 - val_acc: 0.8301 - val_f1: 0.8385
Epoch 21/50

Epoch 20:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 6ms/step - loss: 5009.1556 - acc:
0.9241 - f1: 0.9223 - val_loss: 5009.2030 - val_acc: 0.8329 - val_f1: 0.8411
Epoch 22/50

Epoch 21:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 6ms/step - loss: 5009.1641 - acc:
0.9102 - f1: 0.9088 - val_loss: 5009.2141 - val_acc: 0.8438 - val_f1: 0.8516
Epoch 23/50

Epoch 22:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 5s 6ms/step - loss: 5009.1726 - acc:
0.9027 - f1: 0.9052 - val_loss: 5009.2351 - val_acc: 0.8055 - val_f1: 0.8113
Epoch 24/50

Epoch 23:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 6ms/step - loss: 5009.1762 - acc:
0.8952 - f1: 0.8868 - val_loss: 5009.2104 - val_acc: 0.8466 - val_f1: 0.8542
Epoch 25/50

Epoch 24:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 6ms/step - loss: 5009.1733 - acc:
0.9080 - f1: 0.9067 - val_loss: 5009.2073 - val_acc: 0.8219 - val_f1: 0.8231
Epoch 26/50

Epoch 25:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 5s 6ms/step - loss: 5009.1585 - acc:
0.9230 - f1: 0.9213 - val_loss: 5009.2123 - val_acc: 0.8384 - val_f1: 0.8425
Epoch 27/50
```

```
Epoch 26:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 6ms/step - loss: 5009.1451 - acc:
0.9380 - f1: 0.9321 - val_loss: 5009.1966 - val_acc: 0.8603 - val_f1: 0.8672
Epoch 28/50

Epoch 27:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 6ms/step - loss: 5009.1407 - acc:
0.9305 - f1: 0.9323 - val_loss: 5009.1951 - val_acc: 0.8603 - val_f1: 0.8672
Epoch 29/50

Epoch 28:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 6ms/step - loss: 5009.1447 - acc:
0.9316 - f1: 0.9333 - val_loss: 5009.1967 - val_acc: 0.8521 - val_f1: 0.8594
Epoch 30/50

Epoch 29:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 6ms/step - loss: 5009.1396 - acc:
0.9380 - f1: 0.9359 - val_loss: 5009.1985 - val_acc: 0.8493 - val_f1: 0.8530
Epoch 31/50

Epoch 30:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 6ms/step - loss: 5009.1474 - acc:
0.9262 - f1: 0.9281 - val_loss: 5009.2027 - val_acc: 0.8192 - val_f1: 0.8243
Epoch 32/50

Epoch 31:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 6ms/step - loss: 5009.1443 - acc:
0.9273 - f1: 0.9292 - val_loss: 5009.1895 - val_acc: 0.8493 - val_f1: 0.8568
Epoch 33/50

Epoch 32:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 6ms/step - loss: 5009.1343 - acc:
0.9422 - f1: 0.9438 - val_loss: 5009.1806 - val_acc: 0.8767 - val_f1: 0.8828
Epoch 34/50

Epoch 33:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 6ms/step - loss: 5009.1392 - acc:
0.9422 - f1: 0.9438 - val_loss: 5009.1895 - val_acc: 0.8575 - val_f1: 0.8646
Epoch 35/50

Epoch 34:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 6ms/step - loss: 5009.1388 - acc:
0.9390 - f1: 0.9406 - val_loss: 5009.1883 - val_acc: 0.8685 - val_f1: 0.8750
Epoch 36/50

Epoch 35:   0%|          | 0/935 [00:00<?, ?it/s]
```

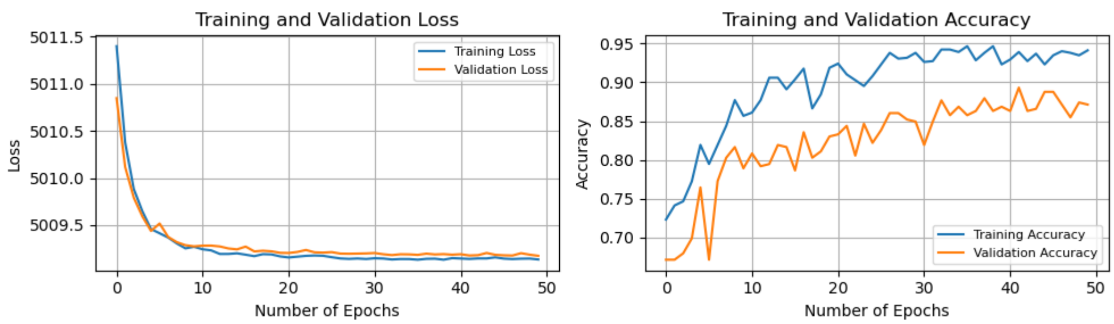
```
935/935 [==============================] - 6s 7ms/step - loss: 5009.1319 - acc:
0.9465 - f1: 0.9479 - val_loss: 5009.1831 - val_acc: 0.8575 - val_f1: 0.8646
Epoch 37/50

Epoch 36:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 6ms/step - loss: 5009.1412 - acc:
0.9283 - f1: 0.9265 - val_loss: 5009.1961 - val_acc: 0.8630 - val_f1: 0.8698
Epoch 38/50

Epoch 37:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 7ms/step - loss: 5009.1428 - acc:
0.9380 - f1: 0.9359 - val_loss: 5009.1876 - val_acc: 0.8795 - val_f1: 0.8816
Epoch 39/50

Epoch 38:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 7ms/step - loss: 5009.1319 - acc:
0.9465 - f1: 0.9479 - val_loss: 5009.1911 - val_acc: 0.8630 - val_f1: 0.8698
Epoch 40/50

Epoch 39:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 7ms/step - loss: 5009.1487 - acc:
0.9230 - f1: 0.9250 - val_loss: 5009.1849 - val_acc: 0.8685 - val_f1: 0.8750
Epoch 41/50

Epoch 40:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 7ms/step - loss: 5009.1446 - acc:
0.9294 - f1: 0.9312 - val_loss: 5009.1898 - val_acc: 0.8630 - val_f1: 0.8698
Epoch 42/50

Epoch 41:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 6ms/step - loss: 5009.1403 - acc:
0.9390 - f1: 0.9369 - val_loss: 5009.1765 - val_acc: 0.8932 - val_f1: 0.8984
Epoch 43/50

Epoch 42:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 7ms/step - loss: 5009.1454 - acc:
0.9273 - f1: 0.9292 - val_loss: 5009.1799 - val_acc: 0.8630 - val_f1: 0.8698
Epoch 44/50

Epoch 43:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 7ms/step - loss: 5009.1453 - acc:
0.9369 - f1: 0.9348 - val_loss: 5009.2033 - val_acc: 0.8658 - val_f1: 0.8724
Epoch 45/50

Epoch 44:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 7ms/step - loss: 5009.1565 - acc:
0.9230 - f1: 0.9250 - val_loss: 5009.1838 - val_acc: 0.8877 - val_f1: 0.8932
Epoch 46/50
```

```
Epoch 45:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 7ms/step - loss: 5009.1435 - acc:
0.9348 - f1: 0.9327 - val_loss: 5009.1784 - val_acc: 0.8877 - val_f1: 0.8932
Epoch 47/50

Epoch 46:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 7ms/step - loss: 5009.1389 - acc:
0.9401 - f1: 0.9379 - val_loss: 5009.1760 - val_acc: 0.8712 - val_f1: 0.8776
Epoch 48/50

Epoch 47:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 7ms/step - loss: 5009.1427 - acc:
0.9380 - f1: 0.9396 - val_loss: 5009.2010 - val_acc: 0.8548 - val_f1: 0.8620
Epoch 49/50

Epoch 48:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 6ms/step - loss: 5009.1440 - acc:
0.9348 - f1: 0.9365 - val_loss: 5009.1851 - val_acc: 0.8740 - val_f1: 0.8802
Epoch 50/50

Epoch 49:   0%|          | 0/935 [00:00<?, ?it/s]

935/935 [==============================] - 6s 6ms/step - loss: 5009.1344 - acc:
0.9412 - f1: 0.9427 - val_loss: 5009.1727 - val_acc: 0.8712 - val_f1: 0.8776
Precision:86.24% Recall:84.03% Fscore:84.98%
```

[8]:

```
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)
```

```python
# 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()
```



[9]:
```python
# Predict on the training dataset
train_predictions = np.argmax(model.predict(W_train), axis=1)

# Calculate accuracy
train_accuracy = accuracy_score(np.argmax(Y_train, axis=1), train_predictions)

# Print accuracy
print("Accuracy on Training Data: {:.3f}".format(train_accuracy))

# Print classification report
print("Classification Report on Training Data:")
print(classification_report(np.argmax(Y_train, axis=1), train_predictions))
```

```
Accuracy on Training Data: 0.956
Classification Report on Training Data:
              precision    recall  f1-score   support

           0       0.94      0.88      0.91       233
           1       0.96      0.98      0.97       702

    accuracy                           0.96       935
```

```
   macro avg       0.95      0.93      0.94       935
weighted avg       0.96      0.96      0.96       935
```

[11]:

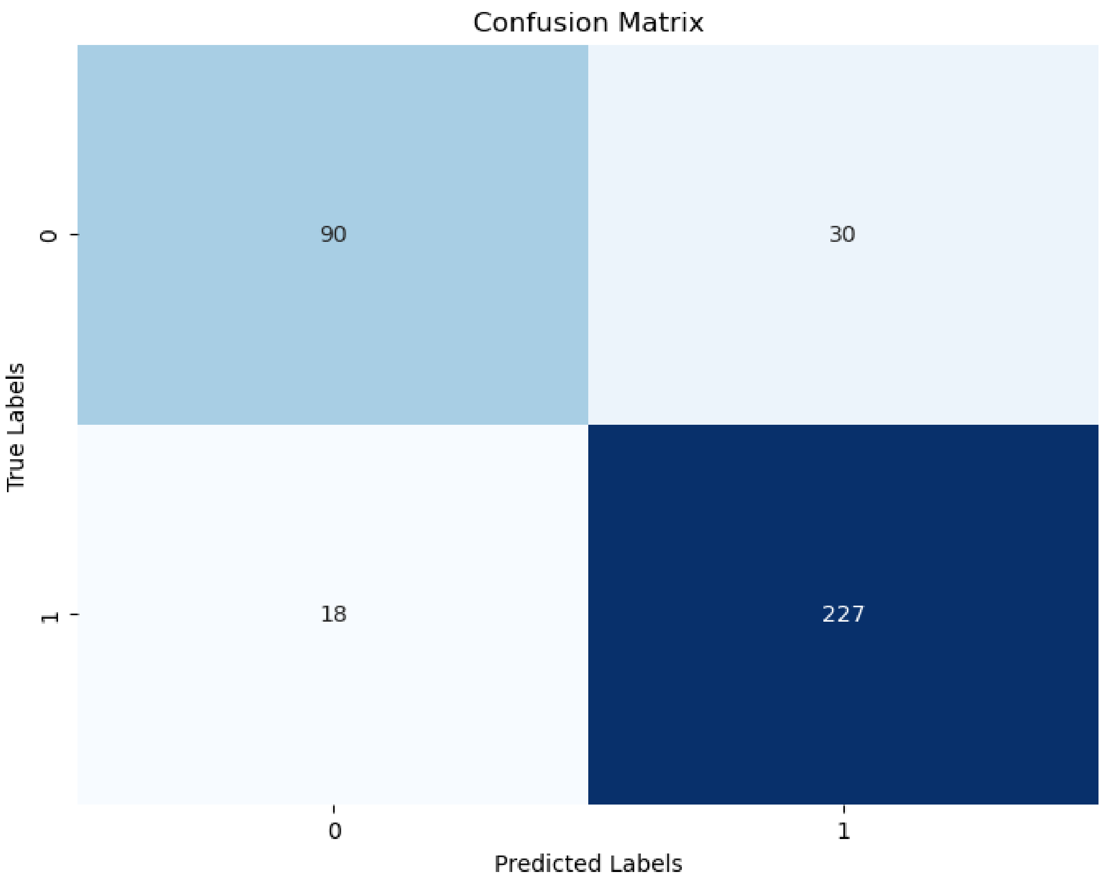
```
print('Running predictions...')
all_predictions, all_labels = [], []
labels = np.argmax(Y_test, axis=1)
y_pred = np.argmax(model.predict(W_test), axis=1)
all_predictions.extend(y_pred.astype('int32'))
all_labels.extend(labels.astype('int32'))
all_labels = np.array(all_labels)
all_predictions = np.array(all_predictions)

correct_pred_count = (all_labels == all_predictions).sum()
test_acc = correct_pred_count / len(all_labels)
# show the the accuracy of testing data
print('We got %d of %d correct (or %.3f accuracy)' % (correct_pred_count,␣
↪len(all_labels), test_acc))
print('Accuracy:', accuracy_score(y_true=all_labels, y_pred=all_predictions))


# 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))
```

```
Running predictions…
```

```
We got 318 of 365 correct (or 0.871 accuracy)
Accuracy: 0.8712328767123287
              precision    recall  f1-score   support

           0     0.8411    0.7500    0.7930       120
           1     0.8837    0.9306    0.9066       245

    accuracy                         0.8712       365
   macro avg     0.8624    0.8403    0.8498       365
weighted avg     0.8697    0.8712    0.8692       365

 Precision:86.24% Recall:84.03% Fscore:84.98%
```

[12]:

```
from sklearn.metrics import confusion_matrix
import seaborn as sns

# Get the predicted labels
predicted_labels = np.argmax(model.predict(W_test), axis=1)

# Create the confusion matrix
cm = confusion_matrix(np.argmax(Y_test, axis=1), predicted_labels)

# Plot the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```


Confusion Matrix
True Labels
0
1
90
30
18
227
0
1
Predicted Labels

```
[ ]:
```

```
[ ]:
```