# Char_feature_Model_Prediction

April 22, 2021

[ ]:
```
import tensorflow as tf
import numpy as np

from tensorflow.keras.layers import Embedding, Conv1D, MaxPool1D, Flatten,␣
↪Dropout, Dense, Input
import datetime
from sklearn.metrics import f1_score
from tensorflow.keras.initializers import he_normal
from tensorflow.keras.models import Model
from tensorflow.keras.utils import plot_model
import matplotlib.pyplot as plt

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

## 1 Loading numpy array from saved .npy file

[ ]:
```
"""

vocab_size
embedding_dim
embedding_matrix

/content/drive/MyDrive/Colab Notebooks/CNN on Text data/From local workspace/
↪X_test.npy
/content/drive/MyDrive/Colab Notebooks/CNN on Text data/From local workspace/
↪X_train.npy
/content/drive/MyDrive/Colab Notebooks/CNN on Text data/From local workspace/
↪char_y_test_ohe.npy
/content/drive/MyDrive/Colab Notebooks/CNN on Text data/From local workspace/
↪char_y_train_ohe.npy
/content/drive/MyDrive/Colab Notebooks/CNN on Text data/From local workspace/
↪embedding_matrix.npy
```

```
"""

vocab_size = 40
embedding_dim = 300
embedding_matrix = np.load("/content/drive/MyDrive/Colab Notebooks/CNN on Text␣
↪data/From local workspace/embedding_matrix.npy")

X_train = np.load('/content/drive/MyDrive/Colab Notebooks/CNN on Text data/From␣
↪local workspace/X_train.npy')
char_y_train_ohe = np.load('/content/drive/MyDrive/Colab Notebooks/CNN on Text␣
↪data/From local workspace/char_y_train_ohe.npy')

X_test = np.load('/content/drive/MyDrive/Colab Notebooks/CNN on Text data/From␣
↪local workspace/X_test.npy')
char_y_test_ohe = np.load('/content/drive/MyDrive/Colab Notebooks/CNN on Text␣
↪data/From local workspace/char_y_test_ohe.npy')
```

[ ]:
```
print("X_train          {}".format(X_train.shape))
print("char_y_train_ohe {}".format(char_y_train_ohe.shape))

print("X_test           {}".format(X_test.shape))
print("char_y_test_ohe  {}".format(char_y_test_ohe.shape))

print("embedding_matrix {}".format(embedding_matrix.shape))
```

```
X_train          (14121, 2500)
char_y_train_ohe (14121, 20)
X_test           (4707, 2500)
char_y_test_ohe  (4707, 20)
embedding_matrix (40, 300)
```

## 2 Char Model - 2

[ ]:
```
tf.keras.backend.clear_session()


input_layer = Input(shape=(2500,))

# Embedding layer
embedding_layer = Embedding(input_dim = vocab_size,
                            output_dim = embedding_dim,
                            weights=[embedding_matrix],
                            input_length=1900,
                            trainable=False)(input_layer)
```

```
# conv1D - kernel_size = 35
conv1D_1 = Conv1D(filters=45,
                  kernel_size=(5),
                  activation='relu',
                  name="conv1",
                  kernel_initializer=he_normal(),
                  kernel_regularizer=tf.keras.regularizers.l1(0.001))␣
 ↪(embedding_layer)

# conv1D - kernel_size = 3
conv1D_2 = Conv1D(filters=50,
                  kernel_size=(6),
                  activation='relu',
                  name="conv2",
                  kernel_initializer=he_normal(),
                  kernel_regularizer=tf.keras.regularizers.l1(0.001)) (conv1D_1)


# maxpool 1D
Max_pool_1 = MaxPool1D(pool_size=2, strides=1 ,padding='valid',␣
 ↪name='Pool1')(conv1D_2)


# conv1D - kernel_size = 5
conv1D_3 = Conv1D(filters=55,
                  kernel_size=(7),
                  activation='relu',
                  name="conv3",
                  kernel_initializer=he_normal(),
                  kernel_regularizer=tf.keras.regularizers.l1(0.001))␣
 ↪(Max_pool_1)

# conv1D - kernel_size = 6
conv1D_4 = Conv1D(filters=60,
                  kernel_size=(8),
                  activation='relu',
                  name="conv4",
                  kernel_initializer=he_normal(),
                  kernel_regularizer=tf.keras.regularizers.l1(0.001)) (conv1D_3)

# maxpool 1D
Max_pool_2 = MaxPool1D(pool_size=2, strides=1 ,padding='valid',␣
 ↪name='Pool2')(conv1D_4)

# conv1D - p = 11
conv1D_5 = Conv1D(filters=65,
                  kernel_size=(11),
```

```
                  activation='relu',
                  name="conv5",
                  kernel_initializer=he_normal(),
                  kernel_regularizer=tf.keras.regularizers.l1(0.001))␣
↪(Max_pool_2)

# flatten
flatten_1 = Flatten(name="flatten")(conv1D_5)

# drop out
dropout1 = Dropout(0.3, name="dropout") (flatten_1)

# dense
Dense_1 = Dense(50,
                activation='relu',
                name='Dense_1',
                kernel_initializer=he_normal(),
                kernel_regularizer=tf.keras.regularizers.l1(0.001))(dropout1)

# output
output_layer = Dense(20, activation='softmax', name="Output")(Dense_1)

model = Model(inputs=input_layer, outputs=output_layer)
```

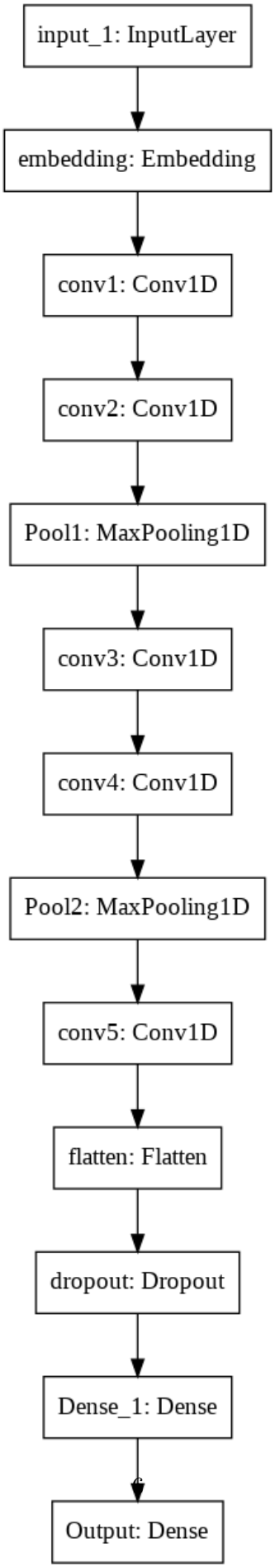
[ ]:
```
model.summary()
```

```
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 2500)]            0
_________________________________________________________________
embedding (Embedding)        (None, 2500, 300)         12000
_________________________________________________________________
conv1 (Conv1D)               (None, 2496, 45)          67545
_________________________________________________________________
conv2 (Conv1D)               (None, 2491, 50)          13550
_________________________________________________________________
Pool1 (MaxPooling1D)         (None, 2490, 50)          0
_________________________________________________________________
conv3 (Conv1D)               (None, 2484, 55)          19305
_________________________________________________________________
conv4 (Conv1D)               (None, 2477, 60)          26460
_________________________________________________________________
Pool2 (MaxPooling1D)         (None, 2476, 60)          0
_________________________________________________________________
conv5 (Conv1D)               (None, 2466, 65)          42965
```

```
_________________________________________________________________
flatten (Flatten)            (None, 160290)            0
_________________________________________________________________
dropout (Dropout)            (None, 160290)            0
_________________________________________________________________
Dense_1 (Dense)              (None, 50)                8014550
_________________________________________________________________
Output (Dense)               (None, 20)                1020
=================================================================
Total params: 8,197,395
Trainable params: 8,185,395
Non-trainable params: 12,000
_________________________________________________________________
```

[ ]:
```
plot_model(model, to_file='Model_2_char_embedding.png')
```

[ ]:


input_1: InputLayer
embedding: Embedding
conv1: Conv1D
conv2: Conv1D
Pool1: MaxPooling1D
conv3: Conv1D
conv4: Conv1D
Pool2: MaxPooling1D
conv5: Conv1D
flatten: Flatten
dropout: Dropout
Dense_1: Dense
Output: Dense

## 3 Training Model

[ ]:
```
class F1_Callback(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs=None):
        y_train_post = []
        y_train_pred_post = []
        y_test_post = []
        y_test_pred_post = []

        y_train_pred_pre = self.model.predict(X_train)
        y_test_pred_pre = self.model.predict(X_test)

        for i, item in enumerate(char_y_train_ohe):
            y_train_post.append(np.argmax(item))

        for i, item in enumerate(y_train_pred_pre):
            y_train_pred_post.append(np.argmax(item))

        for i, item in enumerate(char_y_test_ohe):
            y_test_post.append(np.argmax(item))

        for i, item in enumerate(y_test_pred_pre):
            y_test_pred_post.append(np.argmax(item))

        train_f1 = f1_score(y_train_post, y_train_pred_post, average='micro')
        test_f1  = f1_score(y_test_post, y_test_pred_post, average='micro')

        print("Train F1 score {}".format(train_f1))
        print("Test F1 score {}".format(test_f1))
        train_f1_score_list.append(train_f1)
        test_f1_score_list.append(test_f1)
```

[ ]:
```
opt = tf.keras.optimizers.Adam()


model.compile(optimizer=opt,
```

```
              loss='categorical_crossentropy',
              metrics=['accuracy'])

f1_callback = F1_Callback()

train_f1_score_list = []
test_f1_score_list = []


filepath = "weight_model2.h5"
checkpoint_callback = tf.keras.callbacks.ModelCheckpoint(filepath,␣
 ↪monitor='val_accuracy', verbose=1, save_best_only=True, mode='max')

# reduce_lr_callback = tf.keras.callbacks.
 ↪ReduceLROnPlateau(monitor='val_loss',mode='min', factor=0.01, patience=2,␣
 ↪min_lr=0.00000001)

es_callback = tf.keras.callbacks.EarlyStopping(monitor='val_loss', mode='min',␣
 ↪verbose=1, patience=2)

nan_callback = tf.keras.callbacks.TerminateOnNaN()

log_dir="logs/fit/model_2_" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,␣
 ↪histogram_freq=1, write_graph=True, write_grads=True)
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the
`TensorBoard` Callback.
```

[ ]:
```
history = model.fit(X_train,
          char_y_train_ohe,
          validation_data=(X_test, char_y_test_ohe),
          verbose=1,
          epochs=100,
          batch_size=300,
          callbacks=[es_callback, tensorboard_callback, checkpoint_callback,␣
 ↪nan_callback, f1_callback])  # starts training
```

```
Epoch 1/100
48/48 [==============================] - ETA: 0s - loss: 12.6296 - accuracy:
0.0804
Epoch 00001: val_accuracy improved from -inf to 0.08540, saving model to
weight_model2.h5
Train F1 score 0.08929962467247363
Test F1 score 0.08540471637985979
48/48 [==============================] - 46s 960ms/step - loss: 12.6296 -
accuracy: 0.0804 - val_loss: 7.4854 - val_accuracy: 0.0854
```

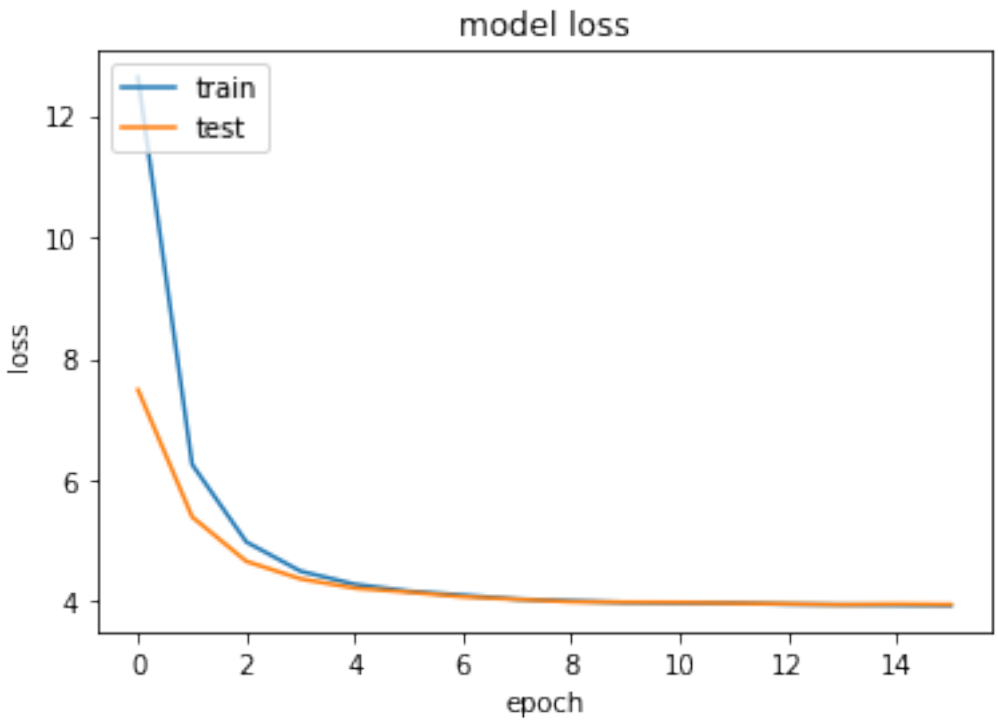
```
Epoch 2/100
47/48 [============================>.] - ETA: 0s - loss: 6.2578 - accuracy:
0.0887
Epoch 00002: val_accuracy improved from 0.08540 to 0.08668, saving model to
weight_model2.h5
Train F1 score 0.08894554210041782
Test F1 score 0.08667941363926067
48/48 [==============================] - 45s 937ms/step - loss: 6.2567 -
accuracy: 0.0886 - val_loss: 5.3937 - val_accuracy: 0.0867
Epoch 3/100
47/48 [============================>.] - ETA: 0s - loss: 4.9826 - accuracy:
0.0862
Epoch 00003: val_accuracy did not improve from 0.08668
Train F1 score 0.08830819347071737
Test F1 score 0.08434246866369237
48/48 [==============================] - 45s 933ms/step - loss: 4.9821 -
accuracy: 0.0861 - val_loss: 4.6623 - val_accuracy: 0.0843
Epoch 4/100
47/48 [============================>.] - ETA: 0s - loss: 4.5004 - accuracy:
0.0857
Epoch 00004: val_accuracy did not improve from 0.08668
Train F1 score 0.0893704411868848
Test F1 score 0.07881878053962184
48/48 [==============================] - 45s 932ms/step - loss: 4.5004 -
accuracy: 0.0856 - val_loss: 4.3743 - val_accuracy: 0.0788
Epoch 5/100
47/48 [============================>.] - ETA: 0s - loss: 4.2847 - accuracy:
0.0855
Epoch 00005: val_accuracy improved from 0.08668 to 0.08753, saving model to
weight_model2.h5
Train F1 score 0.09029105587422986
Test F1 score 0.0875292118121946
48/48 [==============================] - 45s 936ms/step - loss: 4.2848 -
accuracy: 0.0855 - val_loss: 4.2220 - val_accuracy: 0.0875
Epoch 6/100
47/48 [============================>.] - ETA: 0s - loss: 4.1714 - accuracy:
0.0873
Epoch 00006: val_accuracy did not improve from 0.08753
Train F1 score 0.08632533106720487
Test F1 score 0.08179307414489059
48/48 [==============================] - 45s 932ms/step - loss: 4.1712 -
accuracy: 0.0874 - val_loss: 4.1551 - val_accuracy: 0.0818
Epoch 7/100
47/48 [============================>.] - ETA: 0s - loss: 4.1051 - accuracy:
0.0850
Epoch 00007: val_accuracy improved from 0.08753 to 0.08774, saving model to
weight_model2.h5
Train F1 score 0.09000778981658522
```

```
Test F1 score 0.08774166135542809
48/48 [==============================] - 45s 937ms/step - loss: 4.1048 -
accuracy: 0.0851 - val_loss: 4.0786 - val_accuracy: 0.0877
Epoch 8/100
47/48 [============================>.] - ETA: 0s - loss: 4.0403 - accuracy:
0.0871
Epoch 00008: val_accuracy did not improve from 0.08774
Train F1 score 0.08349267049075844
Test F1 score 0.08009347779902273
48/48 [==============================] - 45s 933ms/step - loss: 4.0400 -
accuracy: 0.0872 - val_loss: 4.0372 - val_accuracy: 0.0801
Epoch 9/100
47/48 [============================>.] - ETA: 0s - loss: 4.0118 - accuracy:
0.0889
Epoch 00009: val_accuracy improved from 0.08774 to 0.08902, saving model to
weight_model2.h5
Train F1 score 0.09085758798951915
Test F1 score 0.08901635861482898
48/48 [==============================] - 45s 935ms/step - loss: 4.0118 -
accuracy: 0.0889 - val_loss: 3.9970 - val_accuracy: 0.0890
Epoch 10/100
47/48 [============================>.] - ETA: 0s - loss: 3.9858 - accuracy:
0.0904
Epoch 00010: val_accuracy did not improve from 0.08902
Train F1 score 0.08901635861482898
Test F1 score 0.08264287231782452
48/48 [==============================] - 45s 931ms/step - loss: 3.9856 -
accuracy: 0.0905 - val_loss: 3.9899 - val_accuracy: 0.0826
Epoch 11/100
47/48 [============================>.] - ETA: 0s - loss: 3.9785 - accuracy:
0.0892
Epoch 00011: val_accuracy did not improve from 0.08902
Train F1 score 0.08979534027335175
Test F1 score 0.08243042277459103
48/48 [==============================] - 45s 930ms/step - loss: 3.9784 -
accuracy: 0.0892 - val_loss: 3.9872 - val_accuracy: 0.0824
Epoch 12/100
47/48 [============================>.] - ETA: 0s - loss: 3.9750 - accuracy:
0.0912
Epoch 00012: val_accuracy did not improve from 0.08902
Train F1 score 0.09128248707598612
Test F1 score 0.0839175695772254
48/48 [==============================] - 45s 929ms/step - loss: 3.9749 -
accuracy: 0.0913 - val_loss: 3.9720 - val_accuracy: 0.0839
Epoch 13/100
47/48 [============================>.] - ETA: 0s - loss: 3.9565 - accuracy:
0.0896
Epoch 00013: val_accuracy did not improve from 0.08902
```

```
Train F1 score 0.09347779902273211
Test F1 score 0.08774166135542809
48/48 [==============================] - 45s 930ms/step - loss: 3.9566 -
accuracy: 0.0895 - val_loss: 3.9632 - val_accuracy: 0.0877
Epoch 14/100
47/48 [============================>.] - ETA: 0s - loss: 3.9493 - accuracy:
0.0907
Epoch 00014: val_accuracy did not improve from 0.08902
Train F1 score 0.0931237164506763
Test F1 score 0.08901635861482898
48/48 [==============================] - 45s 931ms/step - loss: 3.9491 -
accuracy: 0.0907 - val_loss: 3.9507 - val_accuracy: 0.0890
Epoch 15/100
47/48 [============================>.] - ETA: 0s - loss: 3.9430 - accuracy:
0.0885
Epoch 00015: val_accuracy did not improve from 0.08902
Train F1 score 0.09022023935981871
Test F1 score 0.0841300191204589
48/48 [==============================] - 45s 929ms/step - loss: 3.9432 -
accuracy: 0.0886 - val_loss: 3.9594 - val_accuracy: 0.0841
Epoch 16/100
47/48 [============================>.] - ETA: 0s - loss: 3.9360 - accuracy:
0.0916
Epoch 00016: val_accuracy did not improve from 0.08902
Train F1 score 0.08958289073011826
Test F1 score 0.08540471637985979
48/48 [==============================] - 45s 929ms/step - loss: 3.9361 -
accuracy: 0.0917 - val_loss: 3.9523 - val_accuracy: 0.0854
Epoch 00016: early stopping
```

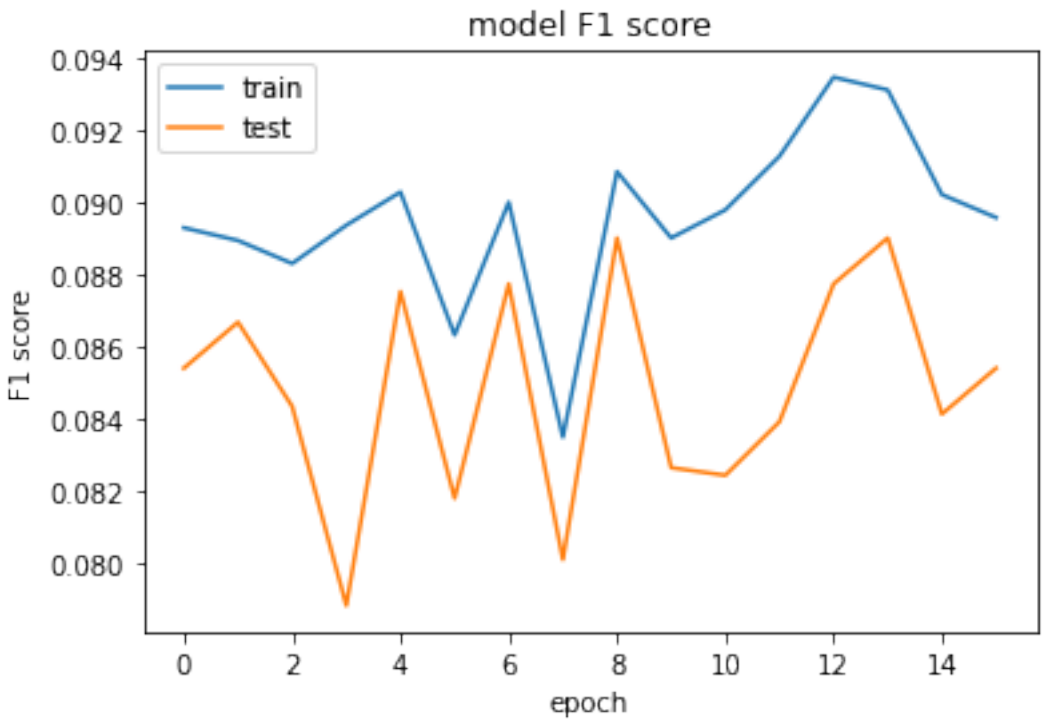
## 4 Loss

[ ]:
```
# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```




## 5 F1 Score

[ ]:
```
plt.plot(train_f1_score_list)
plt.plot(test_f1_score_list)
plt.title('model F1 score')
plt.ylabel('F1 score')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```




## 6 Tensorboard

[ ]:
```
%load_ext tensorboard
%tensorboard --logdir logs/fit
```

```
Reusing TensorBoard on port 6006 (pid 4391), started 0:56:33 ago. (Use '!kill␣
↪4391' to kill it.)
```

```
<IPython.core.display.Javascript object>
```