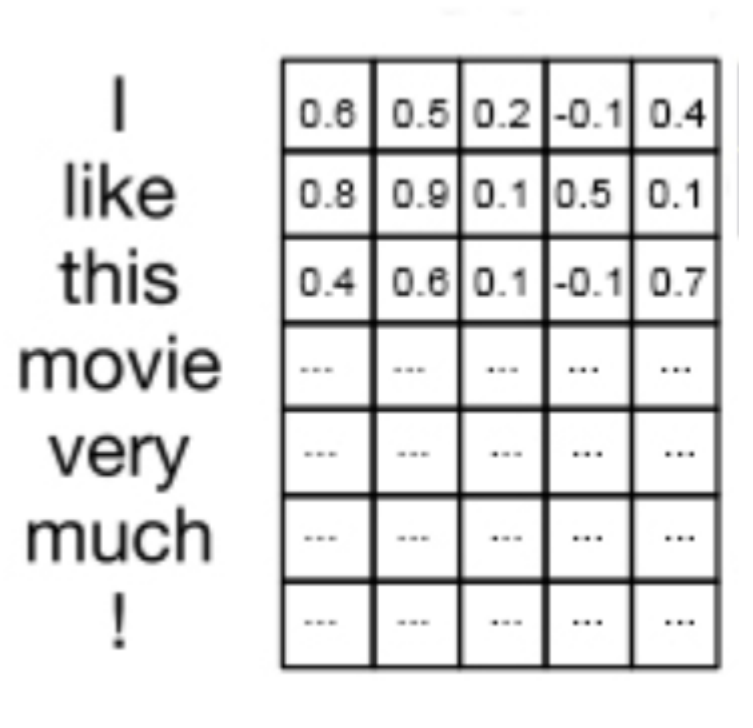
Model-1: Using 1D convolutions with word embeddings

Encoding of the Text --> For a given text data create a Matrix with Embedding layer as shown Below. In the example we have considered d = 5, but in this assignment we will get d = dimension of Word vectors we are using.
i.e if we have maximum of 350 words in a sentence and embedding of 300 dim word vector, we result in 350*300 dimensional matrix for each sentance as output after embedding layer

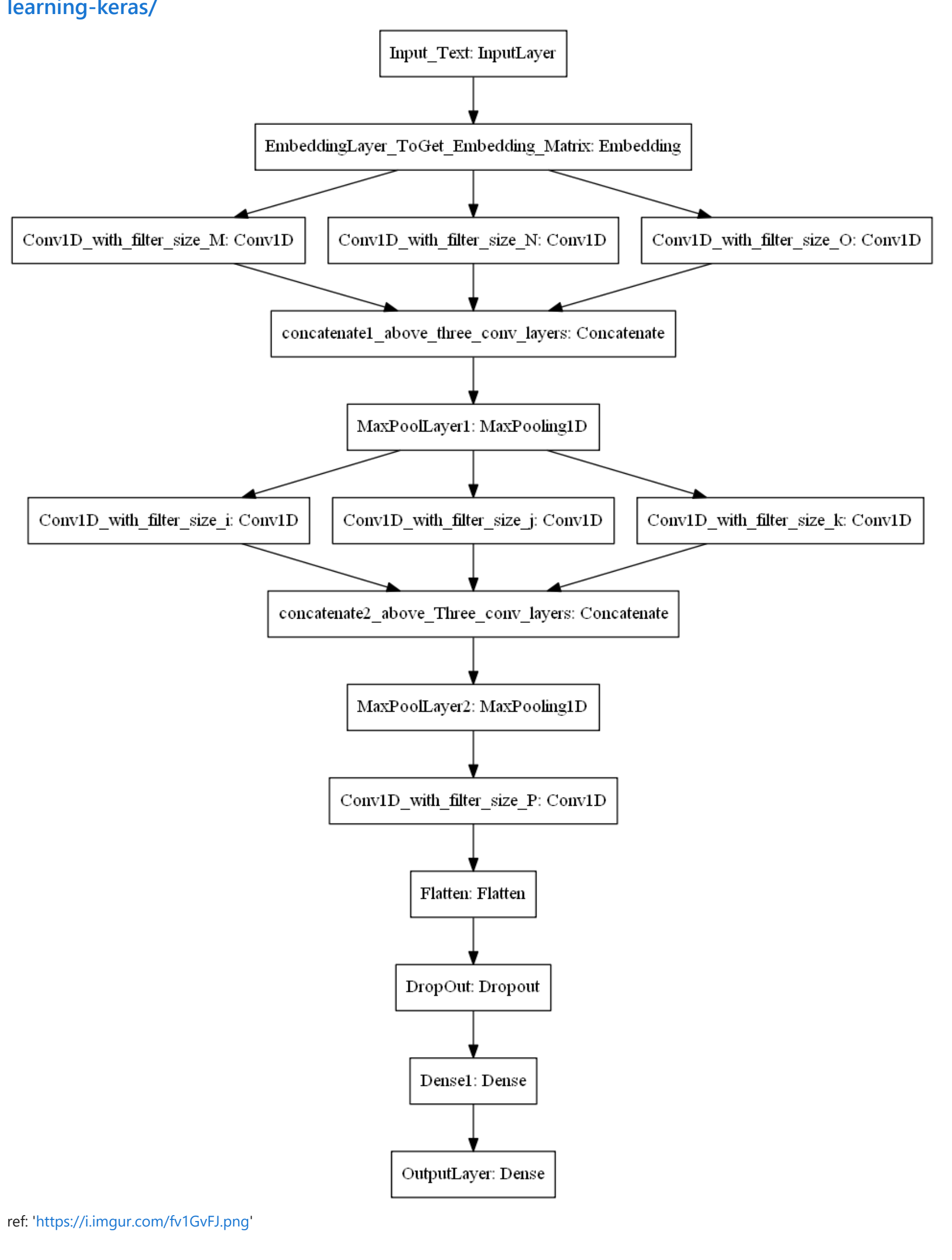


Ref: https://i.imgur.com/kiVQuk1.png

Reference:
https://stackoverflow.com/a/43399308/4084039
https://missinglink.ai/guides/keras/keras-conv1d-working-1d-convolutional-neural-networks-keras/

How EMBEDDING LAYER WORKS

Go through this blog, if you have any doubt on using predefined Embedding values in Embedding layer - https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/



ref: 'https://i.imgur.com/fv1GvFJ.png'

1. all are Conv1D layers with any number of filter and filter sizes, there is no restriction on this.
2. use concatenate layer is to concatenate all the filters/channels.
3. You can use any pool size and stride for maxpooling layer.
4. Don't use more than 16 filters in one Conv layer becuase it will increase the no of params. (Only recommendation if you have less computing power)
5. You can use any number of layers after the Flatten Layer.

In [ ]:
```
pip install -U tensorflow-estimator
```
```
Requirement already satisfied: tensorflow-estimator in /usr/local/lib/python3.7/dist-packages (2.8.0)
```

In [ ]:
```
## Embedding layer
from keras.layers import Embedding

embedding_layer = Embedding(len(tokenizer.word_index)+1, 50, embeddings_initializer=tf.keras.initializers.Const
```

In [ ]:
```
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input,Dense,Conv1D,Flatten,Embedding,MaxPool1D,concatenate,Dropout
from tensorflow.keras.callbacks import ModelCheckpoint,TensorBoard,EarlyStopping
from tensorflow.keras.optimizers import Adam


input_layer = Input(shape=(maxlen))

embed = embedding_layer(input_layer)

#tf.keras.layers.Conv1D(
#    filters =  Integer, the dimensionality of the output space (i.e. the number of output filters in the conv
#    kernel_size = An integer or tuple/list of a single integer, specifying the length of the 1D convolution wi
#    activation=relu,
#    kernel_initializer=he_normal,
#    kernel_regularizer='tf.keras.regularizers.l2'
#)

Conv1D_m = Conv1D(25,4,activation="relu",kernel_initializer =tf.keras.initializers.he_normal(),kernel_regulariz
Conv1D_n = Conv1D(30,4,activation="relu",kernel_initializer =tf.keras.initializers.he_normal(),kernel_regulariz
Conv1D_o = Conv1D(35,4,activation="relu",kernel_initializer = tf.keras.initializers.he_normal(),kernel_regulari

second_layer = concatenate([Conv1D_m,Conv1D_n,Conv1D_o])

max_pool_1 = MaxPool1D(3)(second_layer)

Conv1D_i = Conv1D(25,3,activation="relu",kernel_initializer = tf.keras.initializers.he_normal(),kernel_regulari
Conv1D_j = Conv1D(30,3,activation="relu",kernel_initializer = tf.keras.initializers.he_normal(),kernel_regulari
Conv1D_k =  Conv1D(35,3,activation="relu",kernel_initializer = tf.keras.initializers.he_normal(),kernel_regular

third_layer = concatenate([Conv1D_i,Conv1D_j,Conv1D_k])

max_pool_2 = MaxPool1D(3)(third_layer)

fourth_layer = Conv1D(30,3,activation='relu',kernel_initializer = tf.keras.initializers.he_normal(seed=42),kern

flatten = Flatten()(fourth_layer)

dropout_layer = Dropout(0.2)(flatten)

dense_layer = Dense(64,activation="relu",kernel_initializer = tf.keras.initializers.he_normal())(dropout_layer)

output_layer = Dense(20,activation="softmax",kernel_initializer= tf.keras.initializers.glorot_normal())(dense_l


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

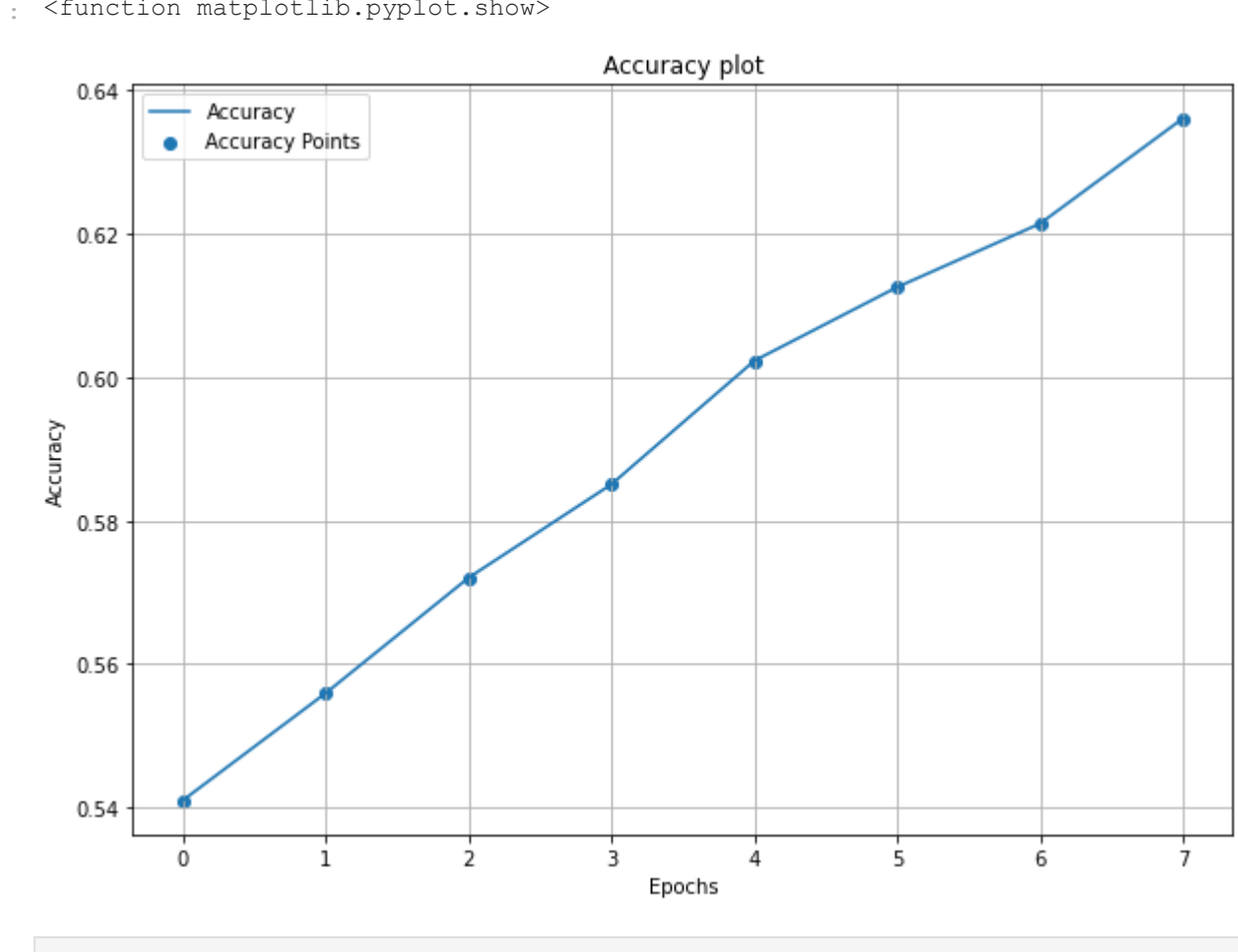
In [ ]:
```
model.summary()
```
```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 9622)]       0           []

 embedding (Embedding)          (None, 9622, 50)     5348350     ['input_1[0][0]']

 conv1d (Conv1D)                (None, 9619, 25)     5025        ['embedding[0][0]']

 conv1d_1 (Conv1D)              (None, 9619, 30)     6030        ['embedding[0][0]']

 conv1d_2 (Conv1D)              (None, 9619, 35)     7035        ['embedding[0][0]']

 concatenate (Concatenate)      (None, 9619, 90)     0           ['conv1d[0][0]',
                                                                  'conv1d_1[0][0]',
                                                                  'conv1d_2[0][0]']

 max_pooling1d (MaxPooling1D)   (None, 3206, 90)     0           ['concatenate[0][0]']

 conv1d_3 (Conv1D)              (None, 3204, 25)     6775        ['max_pooling1d[0][0]']

 conv1d_4 (Conv1D)              (None, 3204, 30)     8130        ['max_pooling1d[0][0]']

 conv1d_5 (Conv1D)              (None, 3204, 35)     9485        ['max_pooling1d[0][0]']

 concatenate_1 (Concatenate)    (None, 3204, 90)     0           ['conv1d_3[0][0]',
                                                                  'conv1d_4[0][0]',
                                                                  'conv1d_5[0][0]']

 max_pooling1d_1 (MaxPooling1D)  (None, 1068, 90)    0           ['concatenate_1[0][0]']

 conv1d_6 (Conv1D)              (None, 1066, 30)     8130        ['max_pooling1d_1[0][0]']

 flatten (Flatten)              (None, 31980)        0           ['conv1d_6[0][0]']

 dropout (Dropout)              (None, 31980)        0           ['flatten[0][0]']

 dense (Dense)                  (None, 64)           2046784     ['dropout[0][0]']

 dense_1 (Dense)                (None, 20)           1300        ['dense[0][0]']

==================================================================================================
Total params: 7,447,044
Trainable params: 2,098,694
Non-trainable params: 5,348,350
__________________________________________________________________________________________________
```

In [ ]:
```
import tensorflow_addons as tfa
from tensorflow_addons.metrics import F1Score

checkpoint = ModelCheckpoint(filepath='best_model_1.h5',verbose=1,monitor='val_accuracy',mode='max',save_best_o

early_stop = EarlyStopping(monitor="val_accuracy",mode='max',patience=2)

log_dir = "logs"
tensorboard = TensorBoard(log_dir=log_dir,histogram_freq=1,write_graph=True)

callbacks =[checkpoint,early_stop,tensorboard]

model.compile(loss='categorical_crossentropy', optimizer=Adam(learning_rate=0.001), metrics=['accuracy',F1Score

history = model.fit(x_train,y_train,epochs=15,verbose=2,validation_data=(x_test,y_test),batch_size =64,callback
```
```
Epoch 1/15
/usr/local/lib/python3.7/dist-packages/keras/engine/training.py:2034: UserWarning: Metric F1Score implements a `reset_states()` method; rename it to `reset_state()` (without the final "s"). The name `reset_states()` has been deprecated to improve API consistency.
  m.reset_state()
Epoch 1: val_accuracy improved from -inf to 0.49862, saving model to best_model_1.h5
221/221 - 940s - loss: 1.5223 - accuracy: 0.5410 - f1_score: 0.5410 - val_loss: 1.6694 - val_accuracy: 0.4986 - val_f1_score: 0.4986 - 940s/epoch - 4s/step
Epoch 2/15

Epoch 2: val_accuracy improved from 0.49862 to 0.50414, saving model to best_model_1.h5
221/221 - 914s - loss: 1.4875 - accuracy: 0.5560 - f1_score: 0.5560 - val_loss: 1.6810 - val_accuracy: 0.5041 - val_f1_score: 0.5041 - 914s/epoch - 4s/step
Epoch 3/15

Epoch 3: val_accuracy improved from 0.50414 to 0.53197, saving model to best_model_1.h5
221/221 - 906s - loss: 1.4562 - accuracy: 0.5721 - f1_score: 0.5721 - val_loss: 1.5736 - val_accuracy: 0.5320 - val_f1_score: 0.5320 - 906s/epoch - 4s/step
Epoch 4/15

Epoch 4: val_accuracy improved from 0.53197 to 0.55428, saving model to best_model_1.h5
221/221 - 908s - loss: 1.4238 - accuracy: 0.5851 - f1_score: 0.5851 - val_loss: 1.5562 - val_accuracy: 0.5543 - val_f1_score: 0.5543 - 908s/epoch - 4s/step
Epoch 5/15

Epoch 5: val_accuracy improved from 0.55428 to 0.55513, saving model to best_model_1.h5
221/221 - 902s - loss: 1.3925 - accuracy: 0.6023 - f1_score: 0.6023 - val_loss: 1.5702 - val_accuracy: 0.5551 - val_f1_score: 0.5551 - 902s/epoch - 4s/step
Epoch 6/15

Epoch 6: val_accuracy improved from 0.55513 to 0.57977, saving model to best_model_1.h5
221/221 - 907s - loss: 1.3668 - accuracy: 0.6126 - f1_score: 0.6126 - val_loss: 1.5333 - val_accuracy: 0.5798 - val_f1_score: 0.5798 - 907s/epoch - 4s/step
Epoch 7/15

Epoch 7: val_accuracy did not improve from 0.57977
221/221 - 897s - loss: 1.3519 - accuracy: 0.6214 - f1_score: 0.6214 - val_loss: 1.5771 - val_accuracy: 0.5528 - val_f1_score: 0.5528 - 897s/epoch - 4s/step
Epoch 8/15

Epoch 8: val_accuracy did not improve from 0.57977
221/221 - 898s - loss: 1.3224 - accuracy: 0.6361 - f1_score: 0.6361 - val_loss: 1.5795 - val_accuracy: 0.5702 - val_f1_score: 0.5702 - 898s/epoch - 4s/step
```

In [ ]:
```
accuracy_array = history.history['accuracy']
f1Score_array = history.history['f1_score']
loss_array = history.history['loss']

print(accuracy_array)
print(f1Score_array)
print(loss_array)
```
```
[0.5410381555557251, 0.5559804439544678, 0.5720558166503906, 0.5850860476493835, 0.6022944450378418, 0.6125628352165222, 0.6214148998260498, 0.6360739469528198]
[0.5410381555557251, 0.5559804439544678, 0.5720558166503906, 0.5850860476493835, 0.6022944450378418, 0.6125628352165222, 0.6214148998260498, 0.6360739469528198]
[1.5222605466842651, 1.4874863624572754, 1.4561767578125, 1.4237747192382812, 1.392497181892395, 1.3668441772460938, 1.3519222736358643, 1.3223624229431152]
```

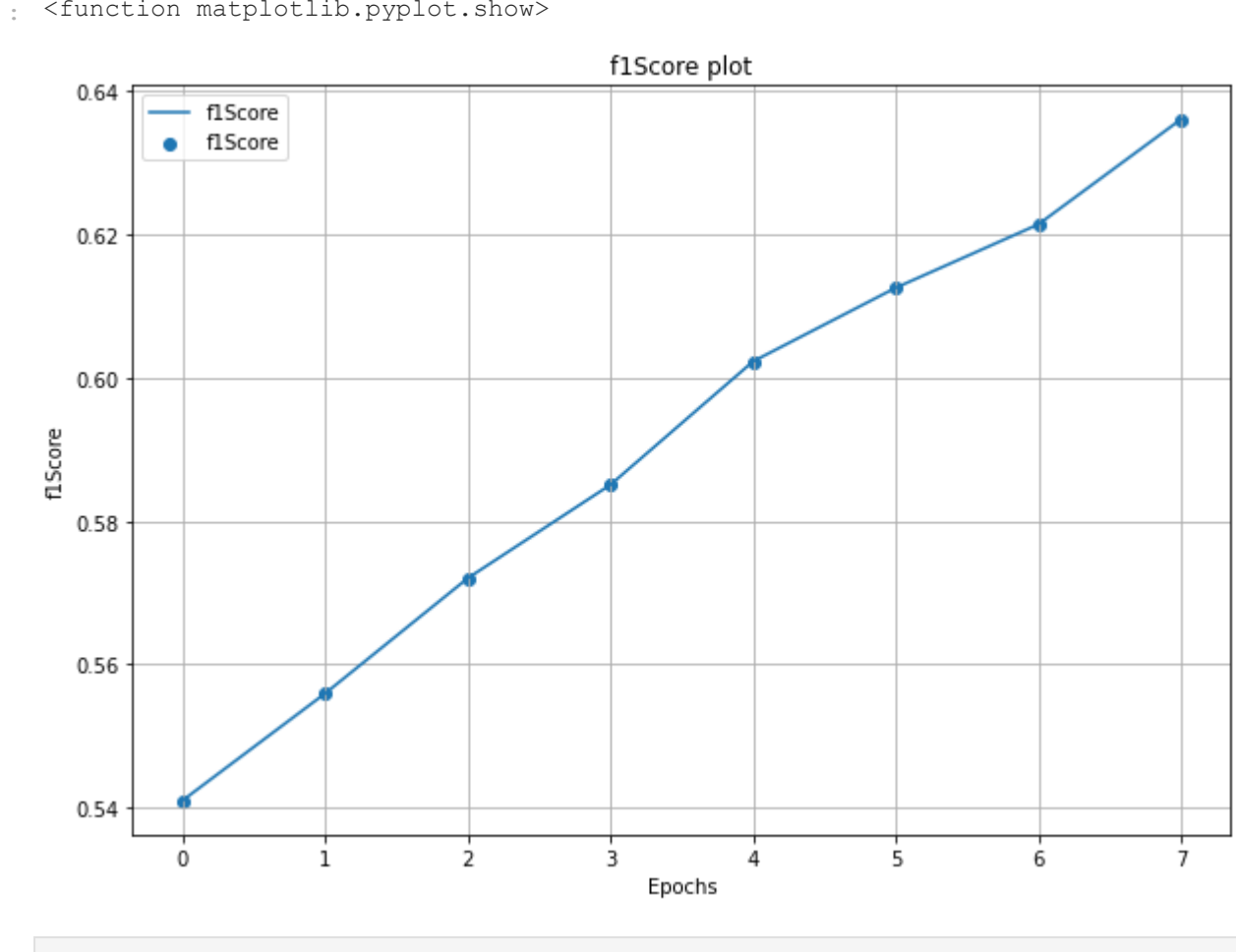
In [ ]:
```
#import maptplotlib.pyplot as plt
import matplotlib.pyplot as plt
epochs =15

plt.figure(figsize = (10,7))
plt.plot(range(len(accuracy_array)),accuracy_array,label='Accuracy')
plt.scatter(range(len(accuracy_array)),accuracy_array,label='Accuracy Points')
plt.title('Accuracy plot')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.grid()
plt.show
```

Out[ ]: <function matplotlib.pyplot.show>



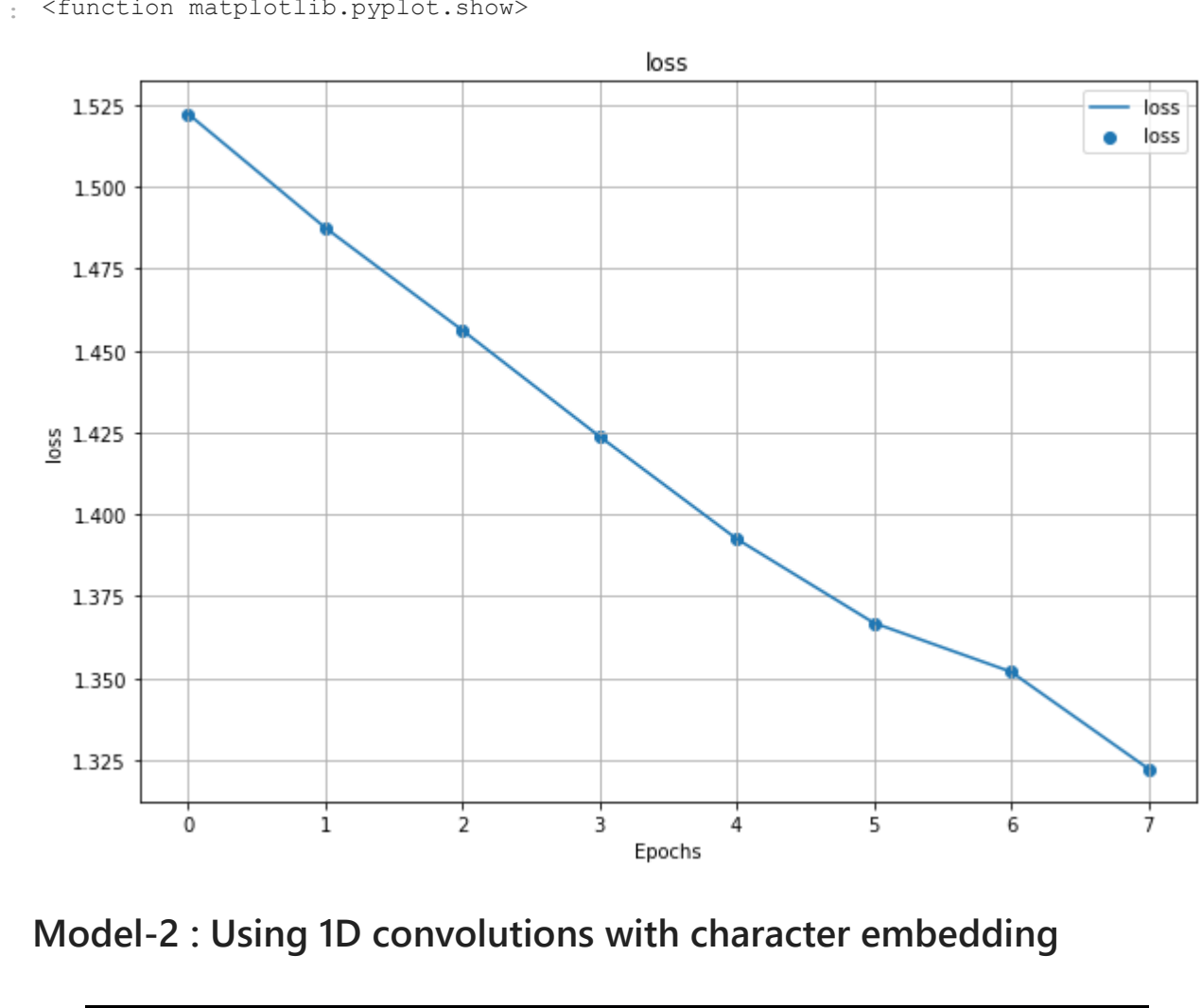
In [ ]:
```
plt.figure(figsize = (10,7))
plt.plot(range(len(f1Score_array)),f1Score_array,label='f1Score')
plt.scatter(range(len(f1Score_array)),f1Score_array,label='f1Score')
plt.title('f1Score plot')
plt.xlabel('Epochs')
plt.ylabel('f1Score')
plt.legend()
plt.grid()
plt.show
```

Out[ ]: <function matplotlib.pyplot.show>

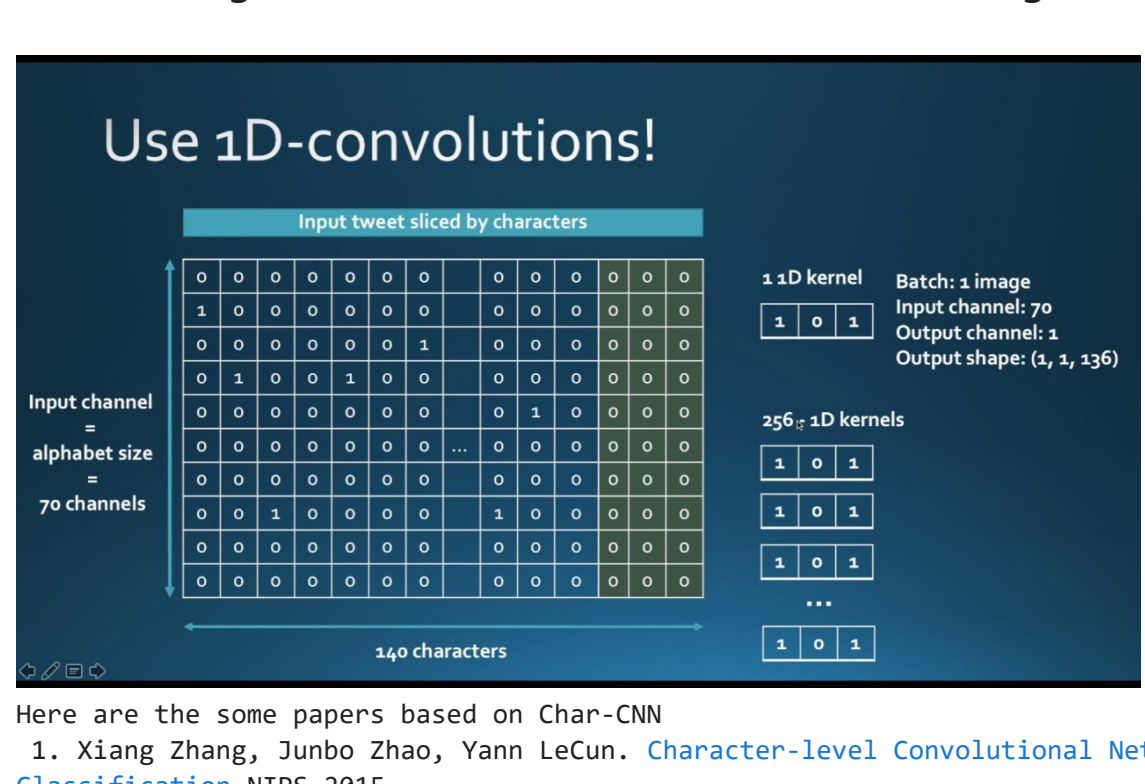


In [ ]:
```
#import maptplotlib.pyplot as plt
import matplotlib.pyplot as plt
epochs =15

plt.figure(figsize = (10,7))
plt.plot(range(len(loss_array)),loss_array,label='loss')
plt.scatter(range(len(loss_array)),loss_array,label='loss')
plt.title('loss')
plt.xlabel('Epochs')
plt.ylabel('loss')
plt.legend()
plt.grid()
plt.show
```

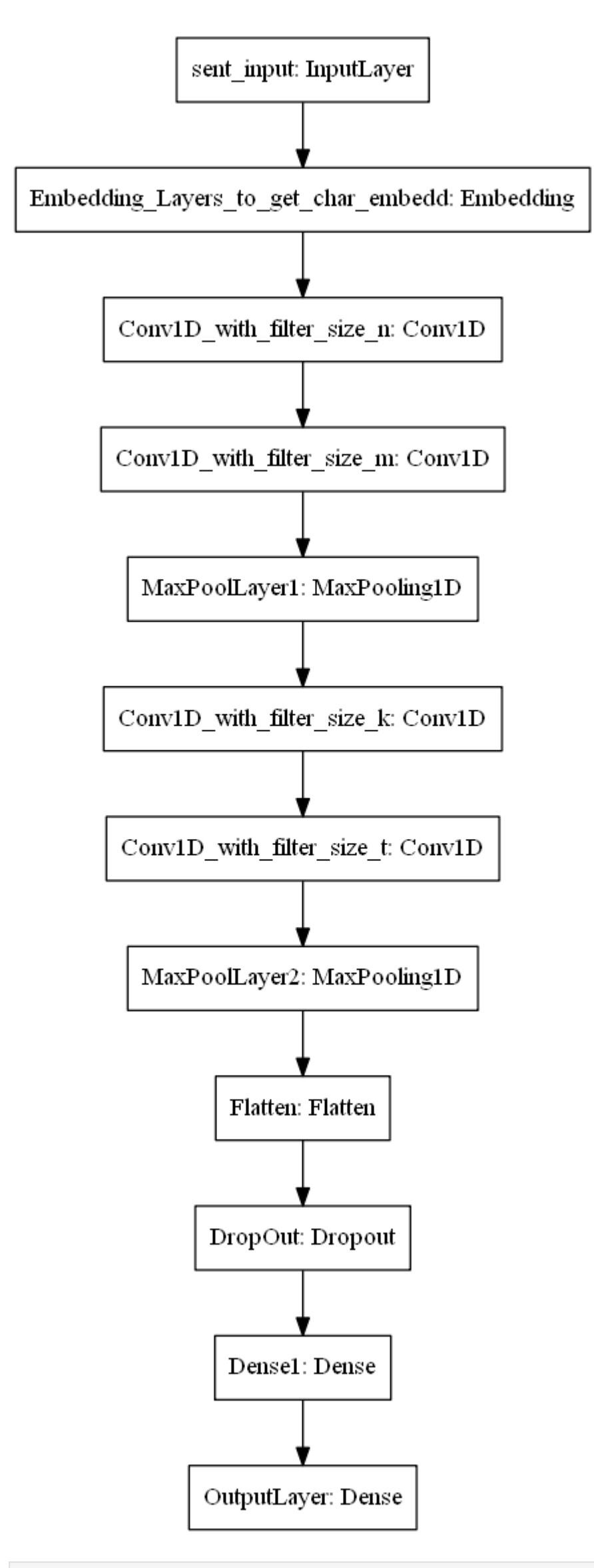
Out[ ]: <function matplotlib.pyplot.show>



Model-2 : Using 1D convolutions with character embedding



Here are the some papers based on Char-CNN
1. Xiang Zhang, Junbo Zhao, Yann LeCun. Character-level Convolutional Networks for Text Classification.NIPS 2015
2. Yoon Kim, Yacine Jernite, David Sontag, Alexander M. Rush. Character-Aware Neural Language Models. AAAI 2016
3. Shaojie Bai, J. Zico Kolter, Vladlen Koltun. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling
4. Use the pratrained char embeddings https://github.com/minimaxir/char-embeddings/blob/master/glove.840B.300d-char.txt



In [ ]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25,stratify=y)
```

In [ ]:
```
tokenize_char = Tokenizer(filters='!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n',char_level= True,oov_token='UNK')
tokenize_char.fit_on_texts(x_train)
```

In [ ]:
```
## Tokenize them
x_train = tokenize_char.texts_to_sequences(x_train)
x_test = tokenize_char.texts_to_sequences(x_test)
```

In [ ]:
```
#Applying Padding
length_of_sentences = [len(s) for s in x_train]
length_of_sentences.sort()
length_of_sentences = np.array(length_of_sentences)
```

In [ ]:
```
embedding_layer_char = Embedding(len(tokenize_char.word_index)+1,41, embeddings_initializer=tf.keras.initialize

first_layer = Input(shape=(maxlen))

embed = embedding_layer_char(first_layer)

m1 = Conv1D(64,3,activation="relu",kernel_initializer =tf.keras.initializers.he_normal(seed=42),kernel_regulari

n1 = Conv1D(64,3,activation="relu",kernel_initializer =tf.keras.initializers.he_normal(seed=42),kernel_regulari


max_pool_1 = MaxPool1D(5)(n1)

o1 = Conv1D(64,3,activation="relu",kernel_initializer = tf.keras.initializers.he_normal(seed=42),kernel_regular

i1 = Conv1D(64,3,activation="relu",kernel_initializer = tf.keras.initializers.he_normal(seed=42),kernel_regular


max_pool_2 = MaxPool1D(5)(i1)

i1 = Conv1D(64,3,activation="relu",kernel_initializer = tf.keras.initializers.he_normal(seed=42),kernel_regular


max_pool_3 = MaxPool1D(5)(i1)

flatten = Flatten()(max_pool_3)

dropout_layer = Dropout(0.5)(flatten)

dense_layer1 = Dense(256,activation="relu",kernel_initializer = tf.keras.initializers.he_normal(seed=42))(dropo


output_layer = Dense(20,activation="softmax",kernel_initializer= tf.keras.initializers.glorot_normal(seed=42))


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

In [ ]:
```
model.summary()
```
```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 9880)]            0

 embedding_1 (Embedding)     (None, 9880, 41)          1681

 conv1d_7 (Conv1D)           (None, 9878, 64)          7936

 conv1d_8 (Conv1D)           (None, 9876, 64)          12352

 max_pooling1d_2 (MaxPooling  (None, 1975, 64)         0
 1D)

 conv1d_9 (Conv1D)           (None, 1973, 64)          12352

 conv1d_10 (Conv1D)          (None, 1971, 64)          12352

 max_pooling1d_3 (MaxPooling  (None, 394, 64)          0
 1D)

 conv1d_11 (Conv1D)          (None, 392, 64)           12352

 max_pooling1d_4 (MaxPooling  (None, 78, 64)           0
 1D)

 flatten_1 (Flatten)         (None, 4992)              0

 dropout_1 (Dropout)         (None, 4992)              0

 dense_2 (Dense)             (None, 256)               1278208

 dense_3 (Dense)             (None, 20)                5140

=================================================================
Total params: 1,342,373
Trainable params: 1,340,692
Non-trainable params: 1,681
_________________________________________________________________
```

In [ ]:
```
checkpoint = ModelCheckpoint(filepath='best_model_1.h5',verbose=1,monitor='val_accuracy',mode='max',save_best_o

early_stop = EarlyStopping(monitor="val_accuracy",mode='max',patience=2)

log_dir = "logs"
tensorboard = TensorBoard(log_dir=log_dir,histogram_freq=1,write_graph=True)

callbacks =[checkpoint,early_stop,tensorboard]

model.compile(loss='categorical_crossentropy', optimizer=Adam(learning_rate=0.0001), metrics=['accuracy',F1Scor

history2 = model.fit(x_train,y_train,epochs=15,verbose=2,validation_data=(x_test,y_test),batch_size =64,callbac
```
```
Epoch 1/15
/usr/local/lib/python3.7/dist-packages/keras/engine/training.py:2034: UserWarning: Metric F1Score implements a `reset_states()` method; rename it to `reset_state()` (without the final "s"). The name `reset_states()` has been deprecated to improve API consistency.
  m.reset_state()
Epoch 1: val_accuracy improved from -inf to 0.05226, saving model to best_model_1.h5
221/221 - 646s - loss: 3.0001 - accuracy: 0.0523 - f1_score: 0.0523 - val_loss: 2.9986 - val_accuracy: 0.0523 - val_f1_score: 0.0523 - 646s/epoch - 3s/step
Epoch 2/15

Epoch 2: val_accuracy did not improve from 0.05226
221/221 - 624s - loss: 2.9983 - accuracy: 0.0523 - f1_score: 0.0523 - val_loss: 2.9983 - val_accuracy: 0.0523 - val_f1_score: 0.0523 - 624s/epoch - 3s/step
Epoch 3/15

Epoch 3: val_accuracy did not improve from 0.05226
221/221 - 655s - loss: 2.9982 - accuracy: 0.0511 - f1_score: 0.0511 - val_loss: 2.9983 - val_accuracy: 0.0523 - val_f1_score: 0.0523 - 655s/epoch - 3s/step
```

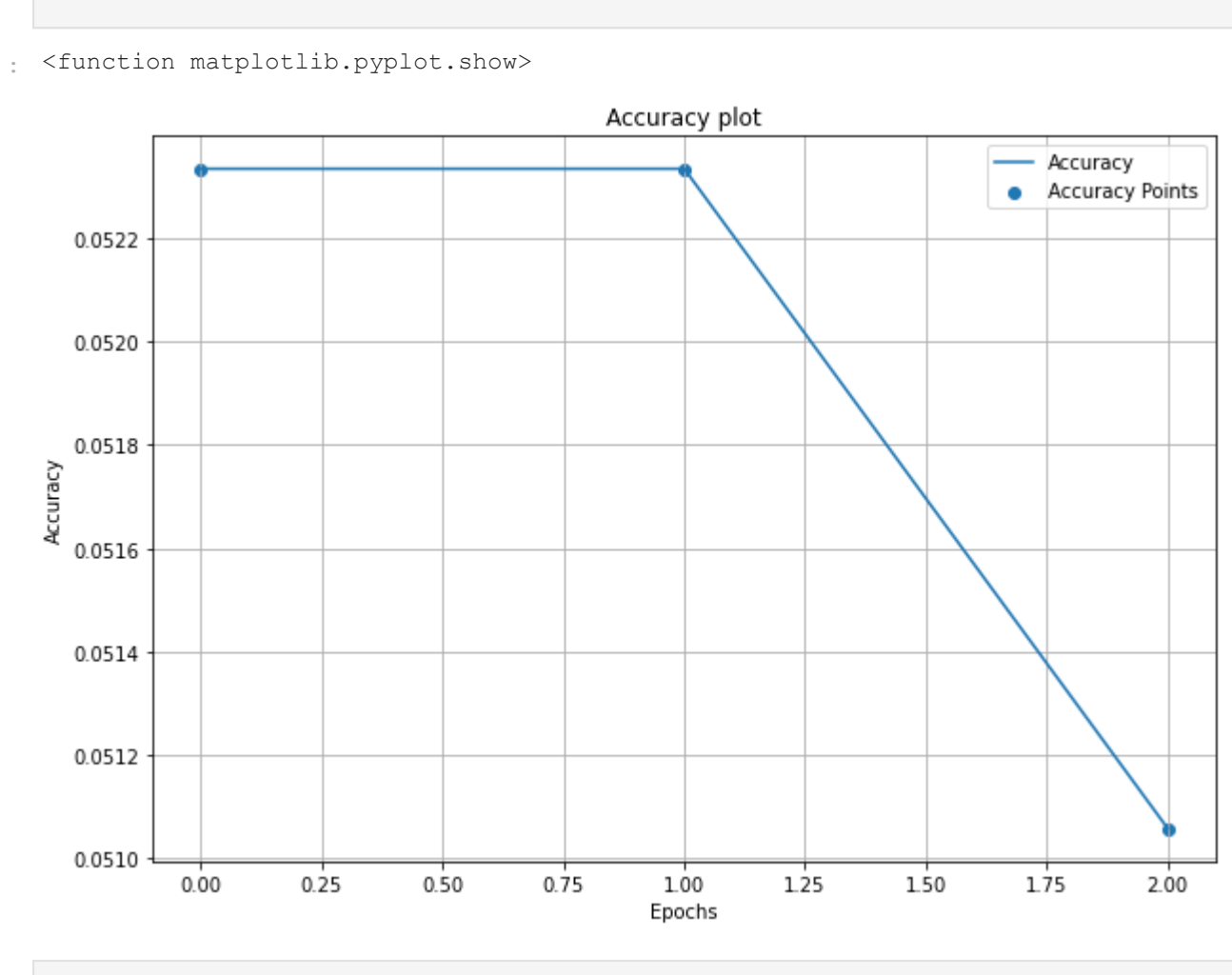
In [ ]:
```
history2.history
```

Out[ ]:
```
{'accuracy': [0.05233340337872505, 0.05233340337872505, 0.051058705896139145],
 'f1_score': [0.05233339965343475, 0.05233339965343475, 0.051058705896139145],
 'loss': [3.000108003616333, 2.9983344078063965, 2.998198986053467],
 'val_accuracy': [0.05226258933544159,
  0.05226258933544159,
  0.05226258933544159],
 'val_f1_score': [0.05226258933544159,
  0.05226258933544159,
  0.05226258933544159],
 'val_loss': [2.998629570007324, 2.998314619064331, 2.998323678970337]}
```

In [ ]:
```
accuracy_array = history2.history['accuracy']
f1Score_array = history2.history['f1_score']
loss_array = history2.history['loss']

print(accuracy_array)
print(f1Score_array)
print(loss_array)
```
```
[0.05233340337872505, 0.05233340337872505, 0.051058705896139145]
[0.05233339965343475, 0.05233339965343475, 0.051058705896139145]
[3.000108003616333, 2.9983344078063965, 2.998198986053467]
```

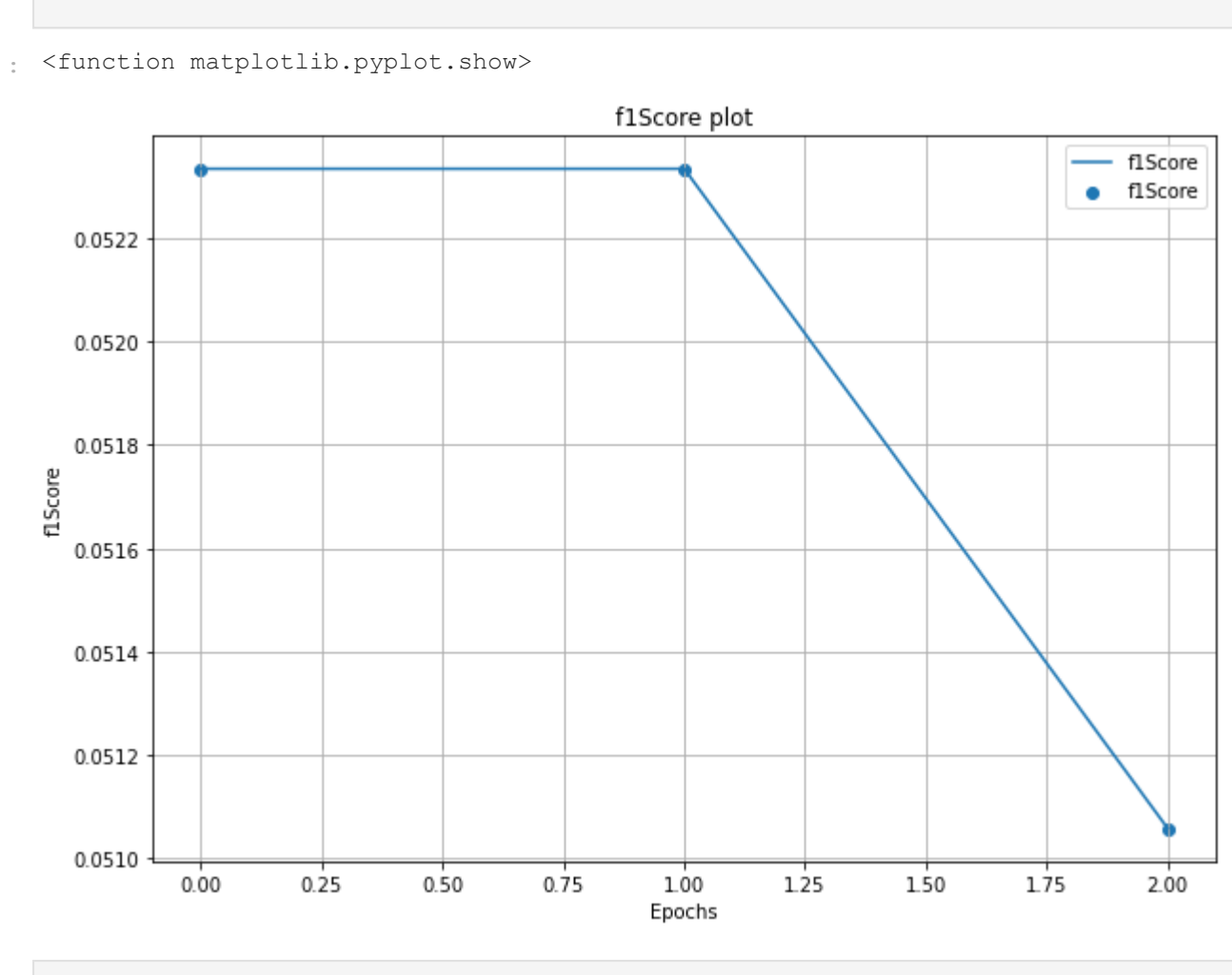
In [ ]:
```
#import maptplotlib.pyplot as plt
import matplotlib.pyplot as plt
epochs =15

plt.figure(figsize = (10,7))
plt.plot(range(len(accuracy_array)),accuracy_array,label='Accuracy')
plt.scatter(range(len(accuracy_array)),accuracy_array,label='Accuracy Points')
plt.title('Accuracy plot')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.grid()
plt.show
```

Out[ ]: <function matplotlib.pyplot.show>



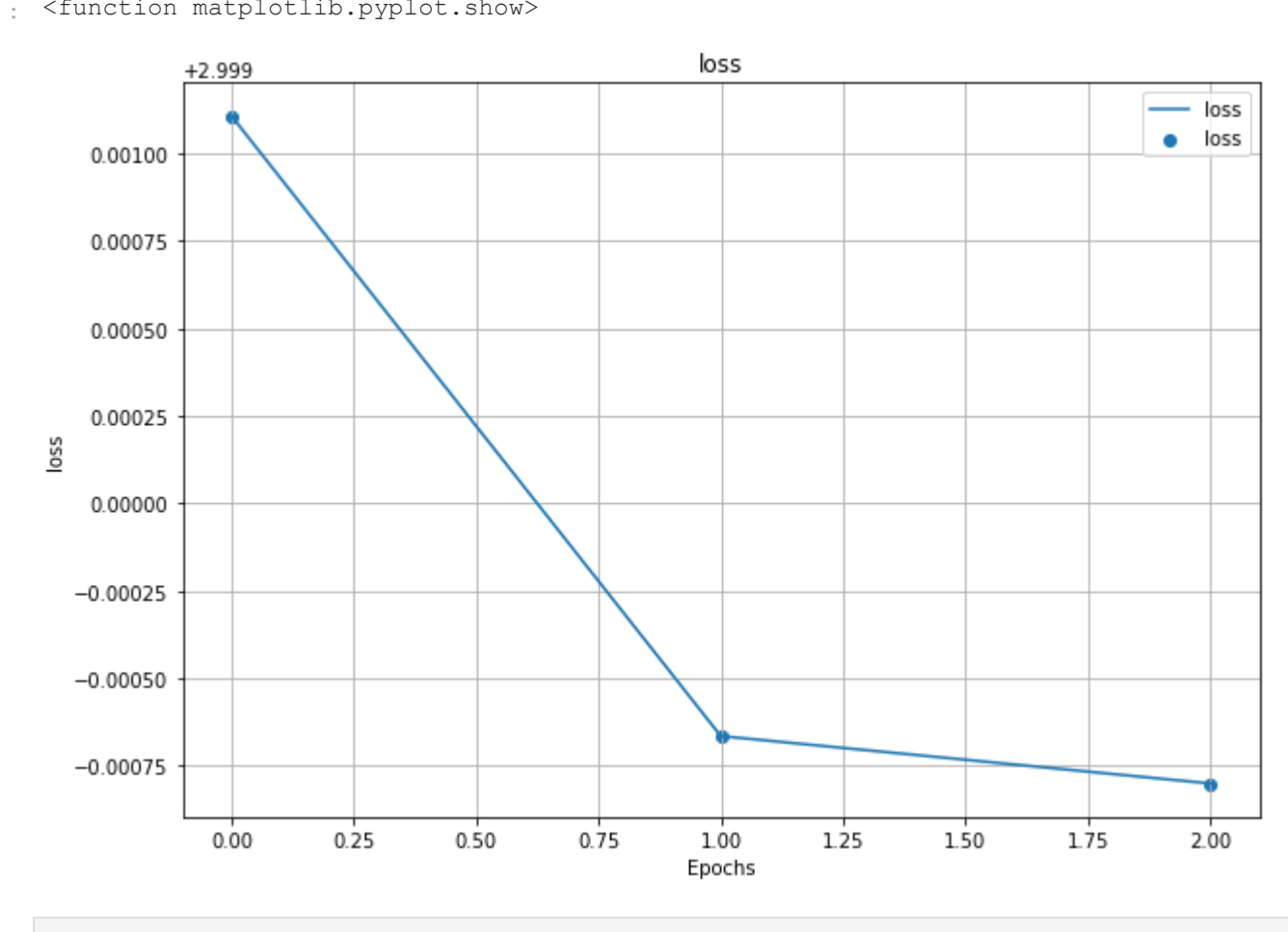
In [ ]:
```
plt.figure(figsize = (10,7))
plt.plot(range(len(f1Score_array)),f1Score_array,label='f1Score')
plt.scatter(range(len(f1Score_array)),f1Score_array,label='f1Score')
plt.title('f1Score plot')
plt.xlabel('Epochs')
plt.ylabel('f1Score')
plt.legend()
plt.grid()
plt.show
```

Out[ ]: <function matplotlib.pyplot.show>



In [ ]:
```
#import maptplotlib.pyplot as plt
import matplotlib.pyplot as plt
epochs =15

plt.figure(figsize = (10,7))
plt.plot(range(len(loss_array)),loss_array,label='loss')
plt.scatter(range(len(loss_array)),loss_array,label='loss')
plt.title('loss')
plt.xlabel('Epochs')
plt.ylabel('loss')
plt.legend()
plt.grid()
plt.show
```

Out[ ]: <function matplotlib.pyplot.show>



In [ ]: