```
In [4]:
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
import seaborn as sns
from numpy import dstack
```

```
In [5]:
def loadingsingle(filepath):
    df = pd.read_csv(filepath, delim_whitespace = True, header = None)
    return df.values
def loadinggroup(filenames,prefix = ''):
    loaded = list()
    for name in filenames:
        data = loadingsingle(prefix + name)
        loaded.append(data)

    loaded = dstack(loaded)
    return loaded

def loadingdataset(group, prefix = ''):
    filepath = prefix + group + '/Inertial Signals/'

    filenames = list()

    filenames += ['body_acc_x_' + group + '.txt', 'body_acc_y_' + group + '.txt', 'body_acc_z_' + group + '.txt']
    filenames += ['total_acc_x_' + group + '.txt', 'total_acc_y_' + group + '.txt', 'total_acc_z_' + group + '.txt']
    filenames += ['body_gyro_x_' + group + '.txt', 'body_gyro_y_' + group + '.txt', 'body_gyro_z_' + group + '.txt']

    X = loadinggroup(filenames,filepath)
    y = loadingsingle(prefix + group + '/y_' + group + '.txt')
    return X,y
```

```
In [6]:
trainx,trainy = loadingdataset('train',r'D:\driveDOCUMENTS\3337\TASK3\UCI HAR Dataset\UCI HAR Dataset')
testx,testy = loadingdataset('test',r'D:\driveDOCUMENTS\3337\TASK3\UCI HAR Dataset\UCI HAR Dataset')
```

```
In [7]:
print(trainx.shape,trainy.shape)
print(testx.shape,testy.shape)
```

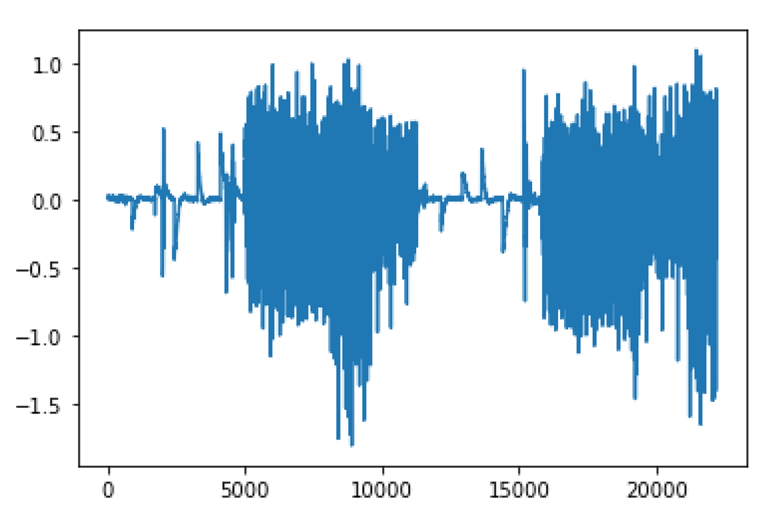
```
(7352, 128, 9) (7352, 1)
(2947, 128, 9) (2947, 1)
```

```
In [131…
tr1 = trainx[:347,:64,0]
tr2 = trainx[:347,:64,1]
tr3 = trainx[:347,:64,2]
tr4 = trainx[:347,:64,3]
tr5 = trainx[:347,:64,4]
tr6 = trainx[:347,:64,5]
tr7 = trainx[:347,:64,6]
tr8 = trainx[:347,:64,7]
tr9 = trainx[:347,:64,8]
trall = (tr1,tr2)
#plt.plot(tr3)
#tr11=pd.DataFrame(trall)
#plt.plot(tr1)
```

plt.figure(figsize = (16,8),dpi=150) tr1.plot(label='tr1')

```
In [139…
one =pd.DataFrame(tr1.flatten(),columns = ['1'])
two =pd.DataFrame(tr2.flatten(),columns = ['1'])
three =pd.DataFrame(tr3.flatten(),columns = ['1'])
four =pd.DataFrame(tr4.flatten(),columns = ['1'])
five =pd.DataFrame(tr5.flatten(),columns = ['1'])
six =pd.DataFrame(tr6.flatten(),columns = ['1'])
seven =pd.DataFrame(tr7.flatten(),columns = ['1'])
eight =pd.DataFrame(tr8.flatten(),columns = ['1'])
nine =pd.DataFrame(tr9.flatten(),columns = ['1'])
q=pd.read_csv('y_train.txt',header=None,delim_whitespace=True)
meow=pd.concat([q,jk],axis=1)
plt.plot(nine)
```

```
Out[139]: [<matplotlib.lines.Line2D at 0x1f4449017f0>]
```

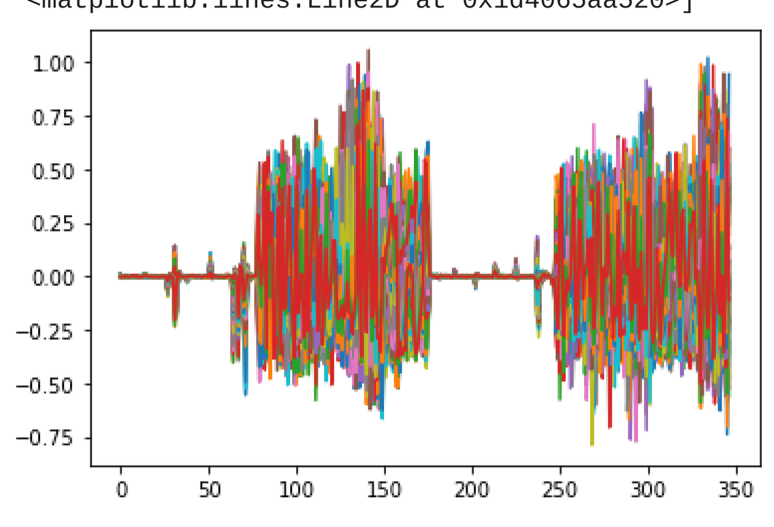


```
In [118…
##for i in range(len(jk)):
    #if(meow[0][i]==5):
        #plt.plot(jk['1'][0:i+64], color='red')
#ytrain=y_train.flatten()
#ytrain =pd.DataFrame(y_train.flatten(),columns = ['0'])
ytrain=y_train.values.flatten()
```

```
Out[118]: array([5, 5, 5, ..., 2, 2, 2], dtype=int64)
```

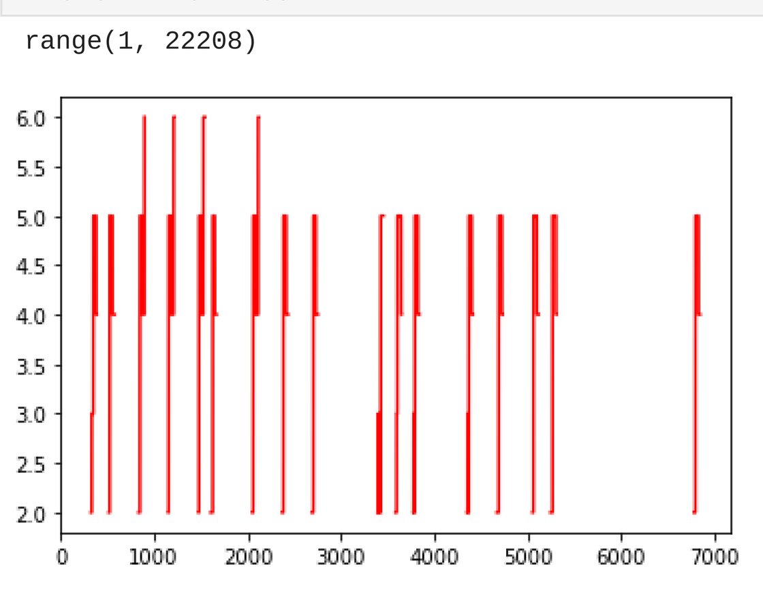
```
In [31]:
o=pd.DataFrame(tr1)
lo= pd.concat([qnew,o],axis=1)
#jjj =pd.DataFrame(lo.flatten(),columns = ['1'])
qnew=q[:347]
plt.plot(tr1)
```

```
Out[31]: [<matplotlib.lines.Line2D at 0x1d406581df0>,
 <matplotlib.lines.Line2D at 0x1d406581e50>,
 <matplotlib.lines.Line2D at 0x1d406581f70>,
 <matplotlib.lines.Line2D at 0x1d40658f0d0>,
 <matplotlib.lines.Line2D at 0x1d40658f1f0>,
 <matplotlib.lines.Line2D at 0x1d40658f310>,
 <matplotlib.lines.Line2D at 0x1d40658f430>,
 <matplotlib.lines.Line2D at 0x1d40658f550>,
 <matplotlib.lines.Line2D at 0x1d40658f670>,
 <matplotlib.lines.Line2D at 0x1d40658f790>,
 <matplotlib.lines.Line2D at 0x1d406581e20>,
 <matplotlib.lines.Line2D at 0x1d40658f8b0>,
 <matplotlib.lines.Line2D at 0x1d40658fac0>,
 <matplotlib.lines.Line2D at 0x1d40658fbe0>,
 <matplotlib.lines.Line2D at 0x1d40658fd00>,
 <matplotlib.lines.Line2D at 0x1d40658fe20>,
 <matplotlib.lines.Line2D at 0x1d40658ff40>,
 <matplotlib.lines.Line2D at 0x1d4065960a0>,
 <matplotlib.lines.Line2D at 0x1d4065961c0>,
 <matplotlib.lines.Line2D at 0x1d4065962e0>,
 <matplotlib.lines.Line2D at 0x1d406596400>,
 <matplotlib.lines.Line2D at 0x1d406596520>,
 <matplotlib.lines.Line2D at 0x1d406596640>,
 <matplotlib.lines.Line2D at 0x1d406596760>,
 <matplotlib.lines.Line2D at 0x1d406596880>,
 <matplotlib.lines.Line2D at 0x1d4065969a0>,
 <matplotlib.lines.Line2D at 0x1d406596ac0>,
 <matplotlib.lines.Line2D at 0x1d406596be0>,
 <matplotlib.lines.Line2D at 0x1d406596d00>,
 <matplotlib.lines.Line2D at 0x1d406596e20>,
 <matplotlib.lines.Line2D at 0x1d406596f40>,
 <matplotlib.lines.Line2D at 0x1d40659c0a0>,
 <matplotlib.lines.Line2D at 0x1d40659c1c0>,
 <matplotlib.lines.Line2D at 0x1d40659c2e0>,
 <matplotlib.lines.Line2D at 0x1d40659c400>,
 <matplotlib.lines.Line2D at 0x1d40659c520>,
 <matplotlib.lines.Line2D at 0x1d40659c640>,
 <matplotlib.lines.Line2D at 0x1d40659c760>,
 <matplotlib.lines.Line2D at 0x1d40659c880>,
 <matplotlib.lines.Line2D at 0x1d40659c9a0>,
 <matplotlib.lines.Line2D at 0x1d40659cac0>,
 <matplotlib.lines.Line2D at 0x1d40659cbe0>,
 <matplotlib.lines.Line2D at 0x1d40659cd00>,
 <matplotlib.lines.Line2D at 0x1d40659ce20>,
 <matplotlib.lines.Line2D at 0x1d40659cf40>,
 <matplotlib.lines.Line2D at 0x1d4065a40a0>,
 <matplotlib.lines.Line2D at 0x1d4065a41c0>,
 <matplotlib.lines.Line2D at 0x1d4065a42e0>,
 <matplotlib.lines.Line2D at 0x1d4065a4400>,
 <matplotlib.lines.Line2D at 0x1d4065a4520>,
 <matplotlib.lines.Line2D at 0x1d4065a4640>,
 <matplotlib.lines.Line2D at 0x1d4065a4760>,
 <matplotlib.lines.Line2D at 0x1d4065a4880>,
 <matplotlib.lines.Line2D at 0x1d4065a49a0>,
 <matplotlib.lines.Line2D at 0x1d4065a4ac0>,
 <matplotlib.lines.Line2D at 0x1d4065a4be0>,
 <matplotlib.lines.Line2D at 0x1d4065a4d00>,
 <matplotlib.lines.Line2D at 0x1d4065a4e20>,
 <matplotlib.lines.Line2D at 0x1d4065a4f40>,
 <matplotlib.lines.Line2D at 0x1d4065aa0a0>,
 <matplotlib.lines.Line2D at 0x1d4065aa1c0>,
 <matplotlib.lines.Line2D at 0x1d4065aa2e0>,
 <matplotlib.lines.Line2D at 0x1d4065aa400>,
 <matplotlib.lines.Line2D at 0x1d4065aa520>]
```

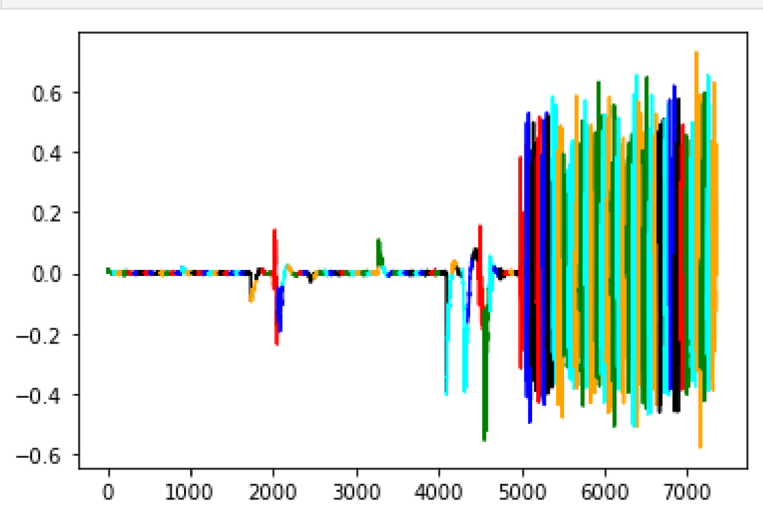


```
In [110…
for i in range(1, len(meow), 64):
    if(meow[0][i] == 2):
        #print('hi')
        plt.plot(meow[0][i:i+64], color = 'red')
#meow['1'][3]
#meow[0][65]
range(1,len(meow))
```

```
Out[110]: range(1, 22208)
```



```
In [29]:
plt.figure()
for i in range(1, len(meow), 64):
    if(meow[0][i] == 1):
        plt.plot(meow['1'][i:i+64], color = 'red')
    if(meow[0][i] == 2):
        plt.plot(meow['1'][i:i+64], color = 'blue')
    if (meow[0][i] == 3):
        plt.plot(meow['1'][i:i+64], color = 'orange')
    if (meow[0][i] == 4):
        plt.plot(meow['1'][i:i+64], color = 'black')
    if(meow[0][i]==5):
        plt.plot(meow['1'][i:i+64], color = 'green')
    if(meow[0][i] == 6):
        plt.plot(meow['1'][i:i+64], color = 'cyan')
```



```
In [15]:
from keras.models import Sequential
from keras.layers import Dense, SimpleRNN,LSTM
from tensorflow import keras
from sklearn.model_selection import KFold
from sklearn.model_selection import GridSearchCV
from tensorflow.keras.layers import Dense, SimpleRNN, LSTM, Dropout
from tensorflow.keras.models import Sequential
from scikeras.wrappers import KerasClassifier

def rnnDef(units):
    model = Sequential()
    model.add(SimpleRNN(20,input_shape = (64,9)))
    model.add(Dense(units,activation = 'relu'))
    model.add(Dense(6,activation = 'softmax'))

    model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
    return model

#model = Sequential()
#model.add(SimpleRNN(20, input_shape=(128,9)))
#model.add(Dense(units=10, activation='relu'))
#model.add(Dense(units=6, activation='softmax'))
#model.compile(loss="categorical_crossentropy", optimizer='adam',metrics=["accuracy"])
kf = KFold(n_splits=5)
clf = KerasClassifier(rnnDef, units = 15, verbose = 0)
y_train_cat = keras.utils.to_categorical(trainy-1, num_classes=6)
epochs = [5,30]
units = [20,40]
param_grid = dict(units = units, epochs = epochs)
grid = GridSearchCV(estimator = clf, param_grid = param_grid, n_jobs = -1, cv = kf, error_score='raise')
grid_results = grid.fit(trainx[:,:64,:],y_train_cat)
print("Best param = %s" % (grid_result.best_params_))
print(grid_results)
#model.fit(trainx,y_train_cat,epochs=2)
#y_test_cat = keras.utils.to_categorical(testy-1, num_classes=6)
#loss,accuracy = model.evaluate(testx, y_test_cat)
#print(accuracy)
```

```
Best param = {'epochs': 10, 'units': 30}
GridSearchCV(cv=KFold(n_splits=5, random_state=None, shuffle=False),
             error_score='raise',
             estimator=KerasClassifier(model=<function rnnDef at 0x0000021498EAB430>, units=15, verbose=0),
             n_jobs=-1, param_grid={'epochs': [5, 30], 'units': [20, 40]})
```

```
In [28]:
def rnnDef2(units):
    model = Sequential()
    model.add(LSTM(20,input_shape = (64,9)))
    model.add(Dense(units,activation = 'relu'))
    model.add(Dense(6,activation = 'softmax'))

    model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
    return model

#model = Sequential()
#model.add(SimpleRNN(20, input_shape=(128,9)))
#model.add(Dense(units=10, activation='relu'))
#model.add(Dense(units=6, activation='softmax'))
#model.compile(loss="categorical_crossentropy", optimizer='adam',metrics=["accuracy"])
kf = KFold(n_splits=5)
clf = KerasClassifier(rnnDef2, units = 15, verbose = 0)
y_train_cat = keras.utils.to_categorical(trainy-1, num_classes=6)
epochs = [5,30]
units = [20,40]
param_grid = dict(units = units, epochs = epochs)
grid = GridSearchCV(estimator = clf, param_grid = param_grid, n_jobs = -1, cv = kf, error_score='raise')
grid_results = grid.fit(trainx[:,:64,:],y_train_cat)
print("Best param = %s" % (grid_result.best_params_))
print(grid_results)
#model.fit(trainx,y_train_cat,epochs=2)
#y_test_cat = keras.utils.to_categorical(testy-1, num_classes=6)
#loss,accuracy = model.evaluate(testx, y_test_cat)
#print(accuracy)
```

```
Best param = {'epochs': 10, 'units': 30}
GridSearchCV(cv=KFold(n_splits=5, random_state=None, shuffle=False),
             error_score='raise',
             estimator=KerasClassifier(model=<function rnnDef2 at 0x00000214AA174700>, units=15, verbose=0),
             n_jobs=-1, param_grid={'epochs': [5, 30], 'units': [20, 40]})
```

```
In [33]:
model = Sequential()
model.add(LSTM(20, input_shape=(128,9)))
model.add(Dense(units=30, activation='relu'))
model.add(Dense(units=6, activation='softmax'))
model.compile(loss="categorical_crossentropy", optimizer='adam',metrics=["accuracy"])
model.fit(trainx,y_train_cat,epochs=10)
y_test_cat = keras.utils.to_categorical(testy-1, num_classes=6)
loss,accuracy = model.evaluate(testx, y_test_cat)
print(accuracy)
```

```
Epoch 1/10
230/230 [==============================] - 5s 14ms/step - loss: 1.2951 - accuracy: 0.4883
Epoch 2/10
230/230 [==============================] - 3s 14ms/step - loss: 0.6574 - accuracy: 0.7289
Epoch 3/10
230/230 [==============================] - 3s 14ms/step - loss: 0.5229 - accuracy: 0.7871
Epoch 4/10
230/230 [==============================] - 3s 14ms/step - loss: 0.6321 - accuracy: 0.7618
Epoch 5/10
230/230 [==============================] - 3s 14ms/step - loss: 0.4709 - accuracy: 0.8211
Epoch 6/10
230/230 [==============================] - 3s 14ms/step - loss: 0.4231 - accuracy: 0.8433
Epoch 7/10
230/230 [==============================] - 3s 14ms/step - loss: 0.3941 - accuracy: 0.8569
Epoch 8/10
230/230 [==============================] - 3s 14ms/step - loss: 0.3510 - accuracy: 0.8745
Epoch 9/10
230/230 [==============================] - 3s 14ms/step - loss: 0.2994 - accuracy: 0.8969
Epoch 10/10
230/230 [==============================] - 3s 14ms/step - loss: 0.2627 - accuracy: 0.9089
93/93 [==============================] - 1s 4ms/step - loss: 0.8602 - accuracy: 0.7469
0.7468612194061279
```

```
In [10]:
model = Sequential()
model.add(SimpleRNN(20, input_shape=(128,9)))
model.add(Dense(units=30, activation='relu'))
model.add(Dense(units=6, activation='softmax'))
model.compile(loss="categorical_crossentropy", optimizer='adam',metrics=["accuracy"])
model.fit(trainx,y_train_cat,epochs=10)
y_test_cat = keras.utils.to_categorical(testy-1, num_classes=6)
loss,accuracy = model.evaluate(testx, y_test_cat)
print(accuracy)
```

```
Epoch 1/10
230/230 [==============================] - 2s 7ms/step - loss: 1.1903 - accuracy: 0.5499
Epoch 2/10
230/230 [==============================] - 2s 7ms/step - loss: 0.7987 - accuracy: 0.6477
Epoch 3/10
230/230 [==============================] - 2s 7ms/step - loss: 0.6573 - accuracy: 0.6974
Epoch 4/10
230/230 [==============================] - 1s 6ms/step - loss: 0.5952 - accuracy: 0.7301
Epoch 5/10
230/230 [==============================] - 1s 6ms/step - loss: 0.5599 - accuracy: 0.7504
Epoch 6/10
230/230 [==============================] - 1s 6ms/step - loss: 0.5161 - accuracy: 0.7806
Epoch 7/10
230/230 [==============================] - 1s 6ms/step - loss: 0.4695 - accuracy: 0.7988
Epoch 8/10
230/230 [==============================] - 2s 7ms/step - loss: 0.4402 - accuracy: 0.8173
Epoch 9/10
230/230 [==============================] - 1s 6ms/step - loss: 0.4236 - accuracy: 0.8338
Epoch 10/10
230/230 [==============================] - 1s 6ms/step - loss: 0.3618 - accuracy: 0.8526
93/93 [==============================] - 0s 2ms/step - loss: 0.6342 - accuracy: 0.7774
0.7774007320404053
```

```
In [ ]:
model = Sequential()
model.add(SimpleRNN(20, input_shape=(128,9)))
model.add(Dense(units=30, activation='relu'))
model.add(Dense(units=6, activation='softmax'))
model.compile(loss="categorical_crossentropy", optimizer='adam',metrics=["accuracy"])
model.fit(trainx,y_train_cat,epochs=10)
y_test_cat = keras.utils.to_categorical(testy-1, num_classes=6)
loss,accuracy = model.evaluate(testx, y_test_cat)
print(accuracy)
```

```
In [34]:
from tensorflow.keras.layers import Dense, SimpleRNN, LSTM, Dropout, Bidirectional
model.add(Bidirectional(LSTM(20, input_shape=(128,9))))
model.add(Dense(units=30, activation='relu'))
model.add(Dense(units=6, activation='softmax'))
model.compile(loss="categorical_crossentropy", optimizer='adam',metrics=["accuracy"])
model.fit(trainx,y_train_cat,epochs=10)
y_test_cat = keras.utils.to_categorical(testy-1, num_classes=6)
loss,accuracy = model.evaluate(testx, y_test_cat)
print(accuracy)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [34], in <cell line: 2>()
      1 from tensorflow.keras.layers import Dense, SimpleRNN, LSTM, Dropout, Bidirectional
----> 2 model.add(Bidirectional(LSTM(20, input_shape=(128,9))))
      3 model.add(Dense(units=30, activation='relu'))
      4 model.add(Dense(units=6, activation='softmax'))

File D:\Downloads\anaconda\lib\site-packages\tensorflow\python\trackable\base.py:205, in no_automatic_dependency_tracking.<locals>._method_wrapper(self, *args, **kwargs)
    203 self._self_setattr_tracking = False  # pylint: disable=protected-access
    204 try:
--> 205   result = method(self, *args, **kwargs)
    206 finally:
    207   self._self_setattr_tracking = previous_value  # pylint: disable=protected-access

File D:\Downloads\anaconda\lib\site-packages\keras\utils\traceback_utils.py:70, in filter_traceback.<locals>.error_handler(*args, **kwargs)
     67     filtered_tb = _process_traceback_frames(e.__traceback__)
     68     # To get the full stack trace, call:
     69     # `tf.debugging.disable_traceback_filtering()`
---> 70     raise e.with_traceback(filtered_tb) from None
     71 finally:
     72     del filtered_tb

File D:\Downloads\anaconda\lib\site-packages\keras\engine\input_spec.py:232, in assert_input_compatibility(input_spec, inputs, layer_name)
    230     ndim = shape.rank
    231     if ndim != spec.ndim:
--> 232         raise ValueError(
    233             f'Input {input_index} of layer "{layer_name}" '
    234             "is incompatible with the layer: "
    235             f"expected ndim={spec.ndim}, found ndim={ndim}. "
    236             f"Full shape received: {tuple(shape)}"
    237         )
    238 if spec.max_ndim is not None:
    239     ndim = x.shape.rank

ValueError: Input 0 of layer "bidirectional" is incompatible with the layer: expected ndim=3, found ndim=2. Full shape received: (None, 6)
```