# convlstm model

from numpy import mean

from numpy import std

from numpy import dstack

from pandas import read\_csv

from keras.models import Sequential

from keras.layers import Dense

from keras.layers import Flatten

from keras.layers import Dropout

from keras.layers import LSTM

from keras.layers import TimeDistributed

from keras.layers import ConvLSTM2D

from keras.utils import to\_categorical

from matplotlib import pyplot

# load a single file as a numpy array

def load\_file(filepath):

  dataframe = read\_csv(filepath, header=None, delim\_whitespace=True)

  return dataframe.values

# load a list of files and return as a 3d numpy array

def load\_group(filenames, prefix=''):

  loaded = list()

  for name in filenames:

    data = load\_file(prefix + name)

    loaded.append(data)

  # stack group so that features are the 3rd dimension

  loaded = dstack(loaded)

  return loaded

# load a dataset group, such as train or test

def load\_dataset\_group(group, prefix=''):

  filepath = prefix + group + '/Inertial Signals/'

  # load all 9 files as a single array

  filenames = list()

  # total acceleration

  filenames += ['total\_acc\_x\_'+group+'.txt', 'total\_acc\_y\_'+group+'.txt', 'total\_acc\_z\_'+group+'.txt']

  # body acceleration

  filenames += ['body\_acc\_x\_'+group+'.txt', 'body\_acc\_y\_'+group+'.txt', 'body\_acc\_z\_'+group+'.txt']

  # body gyroscope

  filenames += ['body\_gyro\_x\_'+group+'.txt', 'body\_gyro\_y\_'+group+'.txt', 'body\_gyro\_z\_'+group+'.txt']

  # load input data

  X = load\_group(filenames, filepath)

  # load class output

  y = load\_file(prefix + group + '/y\_'+group+'.txt')

  return X, y

# load the dataset, returns train and test X and y elements

def load\_dataset(prefix=''):

  # load all train

  trainX, trainy = load\_dataset\_group('train', prefix + 'data/UCI HAR Dataset/')

  print(trainX.shape, trainy.shape)

  # load all test

  testX, testy = load\_dataset\_group('test', prefix + 'data/UCI HAR Dataset/')

  print(testX.shape, testy.shape)

  # zero-offset class values

  trainy = trainy - 1

  testy = testy - 1

  # one hot encode y

  trainy = to\_categorical(trainy)

  testy = to\_categorical(testy)

  print(trainX.shape, trainy.shape, testX.shape, testy.shape)

  return trainX, trainy, testX, testy

# fit and evaluate a model

def evaluate\_model(trainX, trainy, testX, testy):

  # define model

  verbose, epochs, batch\_size = 0, 25, 64

  n\_timesteps, n\_features, n\_outputs = trainX.shape[1], trainX.shape[2], trainy.shape[1]

  # reshape into subsequences (samples, time steps, rows, cols, channels)

  n\_steps, n\_length = 4, 32

  trainX = trainX.reshape((trainX.shape[0], n\_steps, 1, n\_length, n\_features))

  testX = testX.reshape((testX.shape[0], n\_steps, 1, n\_length, n\_features))

  # define model

  model = Sequential()

  model.add(ConvLSTM2D(filters=64, kernel\_size=(1,3), activation='relu', input\_shape=(n\_steps, 1, n\_length, n\_features)))

  model.add(Dropout(0.5))

  model.add(Flatten())

  model.add(Dense(100, activation='relu'))

  model.add(Dense(n\_outputs, activation='softmax'))

  model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

  # fit network

  model.fit(trainX, trainy, epochs=epochs, batch\_size=batch\_size, verbose=verbose)

  # evaluate model

  \_, accuracy = model.evaluate(testX, testy, batch\_size=batch\_size, verbose=0)

  return accuracy

# summarize scores

def summarize\_results(scores):

  print(scores)

  m, s = mean(scores), std(scores)

  print('Accuracy: %.3f%% (+/-%.3f)' % (m, s))

# run an experiment

def run\_experiment(repeats=20):

  # load data

  trainX, trainy, testX, testy = load\_dataset()

  # repeat experiment

  scores = list()

  for r in range(repeats):

    score = evaluate\_model(trainX, trainy, testX, testy)

    score = score \* 100.0

    print('>#%d: %.3f' % (r+1, score))

    scores.append(score)

  # summarize results

  summarize\_results(scores)

# run the experiment

run\_experiment()

Result -----------------------------------------

(7352, 128, 9) (7352, 1)

(2947, 128, 9) (2947, 1)

(7352, 128, 9) (7352, 6) (2947, 128, 9) (2947, 6)

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/framework/op\_def\_library.py:263: colocate\_with (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.

Instructions for updating:

Colocations handled automatically by placer.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow\_backend.py:3445: calling dropout (from tensorflow.python.ops.nn\_ops) with keep\_prob is deprecated and will be removed in a future version.

Instructions for updating:

Please use `rate` instead of `keep\_prob`. Rate should be set to `rate = 1 - keep\_prob`.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/ops/math\_ops.py:3066: to\_int32 (from tensorflow.python.ops.math\_ops) is deprecated and will be removed in a future version.

Instructions for updating:

Use tf.cast instead.

>#1: 88.056

>#2: 90.092

>#3: 91.347

>#4: 90.770

>#5: 89.820

>#6: 91.415

>#7: 89.311

>#8: 90.159

>#9: 90.126

>#10: 90.567

>#11: 90.465

>#12: 91.483

>#13: 89.141

>#14: 91.008

>#15: 89.481

>#16: 90.227

>#17: 91.754

>#18: 91.008

>#19: 90.363

>#20: 90.974

[88.05564981336953, 90.09161859518154, 91.34713267729894, 90.77027485578554, 89.82015609093995, 91.41499830335935, 89.31116389750949, 90.15948422124194, 90.12555140821175, 90.56667797760434, 90.46487953851374, 91.48286392941975, 89.14149983033593, 91.00780454699695, 89.48082796063794, 90.22734984730234, 91.75432643366135, 91.00780454699695, 90.36308109942314, 90.97387173396675]

Accuracy: 90.378% (+/-0.891)