HAR LSTM model

In [1]:

```
import pandas as pd
import numpy as np
```

executed in 3.24s, finished 2018-11-08T17:52:21+05:30

In [2]:

```
# Activities are the class labels
# It is a 6 class classification
ACTIVITIES = {
    0: 'WALKING',
    1: 'WALKING_UPSTAIRS',
    2: 'WALKING_DOWNSTAIRS',
    3: 'SITTING',
    4: 'STANDING',
    5: 'LAYING',
}

# Utility function to print the confusion matrix
def confusion_matrix(Y_true, Y_pred):
    Y_true = pd.Series([ACTIVITIES[y] for y in np.argmax(Y_true, axis=1)])
    Y_pred = pd.Series([ACTIVITIES[y] for y in np.argmax(Y_pred, axis=1)])

    return pd.crosstab(Y_true, Y_pred, rownames=['True'], colnames=['Pred'])
```

executed in 5ms, finished 2018-11-08T17:52:21+05:30

Data

In [3]:

```
# Data directory
DATADIR = 'UCI_HAR_Dataset'
```

executed in 4ms, finished 2018-11-08T17:52:23+05:30

In [4]:

```
# Raw data signals
# Signals are from Accelerometer and Gyroscope
# The signals are in x,y,z directions
# Sensor signals are filtered to have only body acceleration
# excluding the acceleration due to gravity
# Triaxial acceleration from the accelerometer is total acceleration
SIGNALS = [
    "body_acc_x",
    "body_acc_y",
    "body_acc_z",
    "body_gyro_x",
    "body_gyro_y",
    "body_gyro_z",
    "total_acc_x",
    "total_acc_y",
    "total_acc_z"
]
```

executed in 4ms, finished 2018-11-08T17:52:24+05:30

In [5]:

```
# Utility function to read the data from csv file
def _read_csv(filename):
    return pd.read_csv(filename, delim_whitespace=True, header=None)

# Utility function to load the load
def load_signals(subset):
    signals_data = []

    for signal in SIGNALS:
        filename = f'UCI_HAR_Dataset/{subset}/Inertial Signals/{signal}_{subset}.txt'
        signals_data.append(
            _read_csv(filename).as_matrix()
        )

    # Transpose is used to change the dimensionality of the output,
    # aggregating the signals by combination of sample/timestep.
    # Resultant shape is (7352 train/2947 test samples, 128 timesteps, 9 signals)
    return np.transpose(signals_data, (1, 2, 0))
```

executed in 14ms, finished 2018-11-08T17:52:24+05:30

In [6]:

```
def load_y(subset):
    """
    The objective that we are trying to predict is a integer, from 1 to 6,
    that represents a human activity. We return a binary representation of
    every sample objective as a 6 bits vector using One Hot Encoding
    (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.get_dummies.html)
    """
    filename = f'UCI_HAR_Dataset/{subset}/y_{subset}.txt'
    y = _read_csv(filename)[0]

    return pd.get_dummies(y).as_matrix()
```

executed in 4ms, finished 2018-11-08T17:52:25+05:30

In [7]:

```
def load_data():
    """
    Obtain the dataset from multiple files.
    Returns: X_train, X_test, y_train, y_test
    """
    X_train, X_test = load_signals('train'), load_signals('test')
    y_train, y_test = load_y('train'), load_y('test')

    return X_train, X_test, y_train, y_test
```

executed in 4ms, finished 2018-11-08T17:52:26+05:30

In [8]:

```
# Importing tensorflow
np.random.seed(42)
import tensorflow as tf
tf.set_random_seed(42)
```

executed in 10.3s, finished 2018-11-08T17:52:37+05:30

In [9]:

```
# Configuring a session
session_conf = tf.ConfigProto(
    intra_op_parallelism_threads=1,
    inter_op_parallelism_threads=1
)
```

executed in 4ms, finished 2018-11-08T17:52:37+05:30

In [10]:

```
# Import Keras
from keras import backend as K
sess = tf.Session(graph=tf.get_default_graph(), config=session_conf)
K.set_session(sess)
```

executed in 4.53s, finished 2018-11-08T17:52:44+05:30

```
Using TensorFlow backend.
```

In [11]:

```
# Importing libraries
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers.core import Dense, Dropout
```

executed in 4ms, finished 2018-11-08T17:52:53+05:30

In [ ]:

```

```

executed in 4ms, finished 2018-11-05T10:37:48+05:30

In [12]:

```
# Utility function to count the number of classes
def _count_classes(y):
    return len(set([tuple(category) for category in y]))
```

executed in 4ms, finished 2018-11-08T17:52:56+05:30

In [13]:

```
# Loading the train and test data
X_train, X_test, Y_train, Y_test = load_data()
```

executed in 4.63s, finished 2018-11-08T17:53:02+05:30

```
C:\Users\Shrikant\Anaconda3\envs\tf15\lib\site-packages\ipykernel_launcher.p
y:12: FutureWarning: Method .as_matrix will be removed in a future version.
Use .values instead.
  if sys.path[0] == '':
C:\Users\Shrikant\Anaconda3\envs\tf15\lib\site-packages\ipykernel_launcher.p
y:11: FutureWarning: Method .as_matrix will be removed in a future version.
Use .values instead.
  # This is added back by InteractiveShellApp.init_path()
```

In [14]:

```
timesteps = len(X_train[0])
input_dim = len(X_train[0][0])
n_classes = _count_classes(Y_train)

print("timestep of vector dimension is:{}".format(timesteps))
print("input dimension is  :{}".format(input_dim))
print("Numbers of windows with overlapping is:{}".format(len(X_train)))
print("Number of classes is:{}".format(n_classes))
```

executed in 19ms, finished 2018-11-08T17:53:06+05:30

```
timestep of vector dimension is:128
input dimension is  :9
Numbers of windows with overlapping is:7352
Number of classes is:6
```

In [15]:

```
n_classes
```

executed in 10ms, finished 2018-11-08T17:53:07+05:30

Out[15]:

```
6
```

Defining the Architecture of LSTM

In [16]:

```
# Initializing parameters
epochs = 150
batch_size = 128
n_hidden = 32
```

executed in 5ms, finished 2018-11-08T17:53:40+05:30

In [17]:

```
# Initiliazing the sequential model
model1 = Sequential()
# Configuring the parameters
model1.add(LSTM(n_hidden,input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model1.add(Dropout(0.5))
# Adding a dense output layer with sigmoid activation
model1.add(Dense(n_classes,activation='sigmoid'))
model1.summary()

```

executed in 271ms, finished 2018-11-08T17:53:45+05:30

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_1 (LSTM)                (None, 32)                5376      
_________________________________________________________________
dropout_1 (Dropout)          (None, 32)                0         
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 198       
=================================================================
Total params: 5,574
Trainable params: 5,574
Non-trainable params: 0
_________________________________________________________________
```

In [18]:

```
# Compiling the model
model1.compile(loss='categorical_crossentropy',
               optimizer='rmsprop',
               metrics=['accuracy'])

```

executed in 34ms, finished 2018-11-08T17:53:49+05:30

In [19]:

```
# Training the model
history = model1.fit(X_train,Y_train,batch_size=batch_size,validation_data=(X_test, Y_
                     epochs=epochs)
```

executed in 50m 26s, finished 2018-11-08T18:44:17+05:30

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/150
7352/7352 [==============================] - 19s 3ms/step - loss: 1.5712 -
acc: 0.3894 - val_loss: 1.4528 - val_acc: 0.4259
Epoch 2/150
7352/7352 [==============================] - 17s 2ms/step - loss: 1.3602 -
acc: 0.4429 - val_loss: 1.4376 - val_acc: 0.4296
Epoch 3/150
7352/7352 [==============================] - 18s 2ms/step - loss: 1.2929 -
acc: 0.4668 - val_loss: 1.3488 - val_acc: 0.4245
Epoch 4/150
7352/7352 [==============================] - 18s 2ms/step - loss: 1.2183 -
acc: 0.4872 - val_loss: 1.2905 - val_acc: 0.3990
Epoch 5/150
7352/7352 [==============================] - 19s 3ms/step - loss: 1.1837 -
acc: 0.4852 - val_loss: 1.1706 - val_acc: 0.4659
Epoch 6/150
7352/7352 [==============================] - 18s 2ms/step - loss: 1.0878 -
acc: 0.5143 - val loss: 1.0978 - val acc: 0.4880
```

In [20]:

```
# Confusion Matrix
print(confusion_matrix(Y_test, model1.predict(X_test)))
```

executed in 8.90s, finished 2018-11-08T18:45:04+05:30

```
Pred                LAYING  SITTING  STANDING  WALKING  WALKING_DOWNSTAIRS
\
True

LAYING                 510        0         0        0                   0

SITTING                  2      392        81        0                   1

STANDING                 0      107       423        2                   0

WALKING                  0        0         6      459                   5

WALKING_DOWNSTAIRS       0        0         0        1                 418

WALKING_UPSTAIRS         0        1         1       10                  17


Pred                WALKING_UPSTAIRS
True
LAYING                            27
SITTING                           15
STANDING                           0
WALKING                           26
WALKING_DOWNSTAIRS                 1
WALKING_UPSTAIRS                 442
```

In [21]:

```
score = model1.evaluate(X_test, Y_test)
```

executed in 9.33s, finished 2018-11-08T18:45:17+05:30

```
2947/2947 [==============================] - 9s 3ms/step
```

In [22]:

```
score
```

executed in 12ms, finished 2018-11-08T18:45:21+05:30

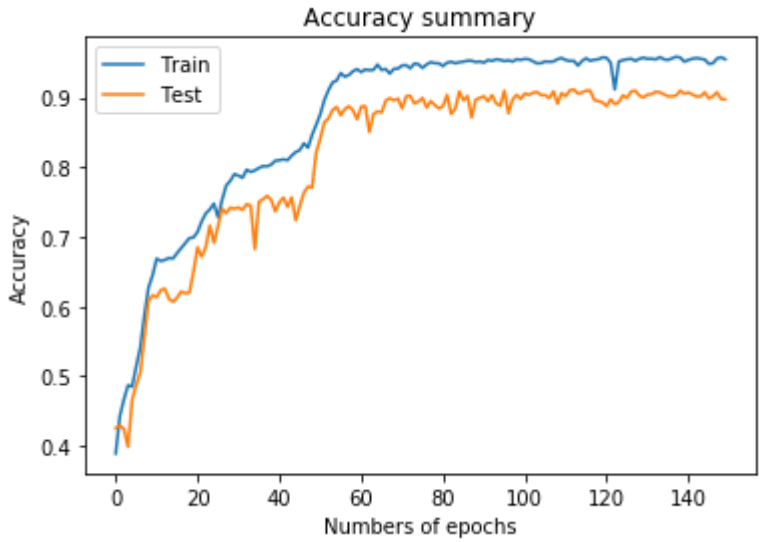
Out[22]:

```
[0.379760792548977, 0.8971835765184933]
```

In [23]:

```
import matplotlib.pyplot as plt
%matplotlib inline
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.xlabel("Numbers of epochs")
plt.ylabel("Accuracy")
plt.title("Accuracy summary")
plt.legend(['Train','Test'],loc='upper left')
plt.show()
```

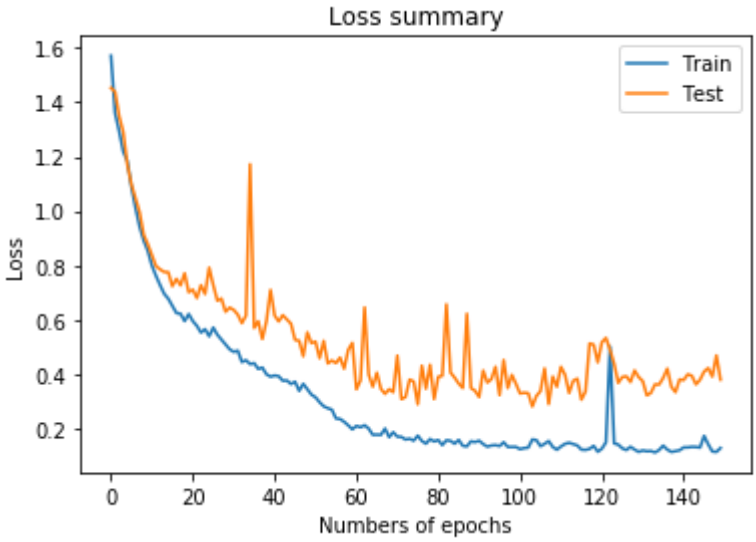
executed in 378ms, finished 2018-11-08T18:45:26+05:30



In [24]:

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.xlabel("Numbers of epochs")
plt.ylabel("Loss")
plt.title("Loss summary")
plt.legend(['Train','Test'],loc='best')
plt.show()
```

executed in 146ms, finished 2018-11-08T18:45:33+05:30



model 2 with single LSTM and 64 LSTM neuron units

In [25]:

```
# Initiliazing the sequential model
model2 = Sequential()
# Configuring the parameters
model2.add(LSTM(64,input_shape=(timesteps, input_dim)))
# Adding a dropout layer
model2.add(Dropout(0.5))
# Adding a dense output layer with sigmoid activation
model2.add(Dense(n_classes,activation='sigmoid'))
model2.summary()

```

executed in 354ms, finished 2018-11-08T18:47:01+05:30

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_2 (LSTM)                (None, 64)                18944     
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0         
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 390       
=================================================================
Total params: 19,334
Trainable params: 19,334
Non-trainable params: 0
_________________________________________________________________
```

In [26]:

```
# Compiling the model
model2.compile(loss='categorical_crossentropy',optimizer='rmsprop',metrics=['accuracy'
```

executed in 37ms, finished 2018-11-08T18:47:14+05:30

In [27]:

```
# Training the model
history2 = model2.fit(X_train,Y_train,batch_size=batch_size,validation_data=(X_test, Y
                      epochs=150)
```

executed in 51m 4s, finished 2018-11-08T19:38:46+05:30

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/150
7352/7352 [==============================] - 19s 3ms/step - loss: 1.4832 - a
cc: 0.3648 - val_loss: 1.4112 - val_acc: 0.3682
Epoch 2/150
7352/7352 [==============================] - 20s 3ms/step - loss: 1.2628 - a
cc: 0.4634 - val_loss: 1.2453 - val_acc: 0.4408
Epoch 3/150
7352/7352 [==============================] - 23s 3ms/step - loss: 1.2136 - a
cc: 0.4771 - val_loss: 1.2227 - val_acc: 0.5029
Epoch 4/150
7352/7352 [==============================] - 19s 3ms/step - loss: 1.1383 - a
cc: 0.4924 - val_loss: 1.1433 - val_acc: 0.4869
Epoch 5/150
7352/7352 [==============================] - 18s 2ms/step - loss: 1.1020 - a
cc: 0.5069 - val_loss: 1.1588 - val_acc: 0.4574
Epoch 6/150
7352/7352 [==============================] - 19s 3ms/step - loss: 1.0799 - a
cc: 0.5207 - val_loss: 1.0750 - val_acc: 0.5341
Epoch 7/150
7352/7352 [==============================] - 19s 3ms/step - loss: 0.9994 - a
cc: 0.5653 - val_loss: 0.9610 - val_acc: 0.5965
Epoch 8/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.8668 - a
cc: 0.6279 - val_loss: 0.8924 - val_acc: 0.5799
Epoch 9/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.8110 - a
cc: 0.6510 - val_loss: 0.8679 - val_acc: 0.6522
Epoch 10/150
7352/7352 [==============================] - 18s 2ms/step - loss: 0.7680 - a
cc: 0.6663 - val_loss: 0.7847 - val_acc: 0.6960
Epoch 11/150
7352/7352 [==============================] - 20s 3ms/step - loss: 0.7441 - a
cc: 0.6704 - val_loss: 0.8231 - val_acc: 0.6647
Epoch 12/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.6934 - a
cc: 0.7029 - val_loss: 0.7502 - val_acc: 0.7007
Epoch 13/150
7352/7352 [==============================] - 22s 3ms/step - loss: 0.7004 - a
cc: 0.7198 - val_loss: 0.6977 - val_acc: 0.7143
Epoch 14/150
7352/7352 [==============================] - 22s 3ms/step - loss: 0.6211 - a
cc: 0.7435 - val_loss: 0.7697 - val_acc: 0.6895
Epoch 15/150
7352/7352 [==============================] - 22s 3ms/step - loss: 0.6125 - a
cc: 0.7360 - val_loss: 0.8903 - val_acc: 0.6559
Epoch 16/150
7352/7352 [==============================] - 22s 3ms/step - loss: 0.5748 - a
cc: 0.7560 - val_loss: 0.6490 - val_acc: 0.7170
Epoch 17/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.5482 - a
```

```
cc: 0.7662 - val_loss: 0.6397 - val_acc: 0.7486
Epoch 18/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.5436 - a
cc: 0.7767 - val_loss: 0.6099 - val_acc: 0.7438
Epoch 19/150
7352/7352 [==============================] - 22s 3ms/step - loss: 0.5051 - a
cc: 0.7836 - val_loss: 0.6790 - val_acc: 0.7184
Epoch 20/150
7352/7352 [==============================] - 22s 3ms/step - loss: 0.4921 - a
cc: 0.7942 - val_loss: 0.5517 - val_acc: 0.7540
Epoch 21/150
7352/7352 [==============================] - 19s 3ms/step - loss: 0.4834 - a
cc: 0.7905 - val_loss: 0.5913 - val_acc: 0.7516
Epoch 22/150
7352/7352 [==============================] - 17s 2ms/step - loss: 0.4942 - a
cc: 0.7923 - val_loss: 0.6294 - val_acc: 0.7618
Epoch 23/150
7352/7352 [==============================] - 18s 2ms/step - loss: 0.6722 - a
cc: 0.7386 - val_loss: 0.6196 - val_acc: 0.7282
Epoch 24/150
7352/7352 [==============================] - 20s 3ms/step - loss: 0.4726 - a
cc: 0.8088 - val_loss: 1.3371 - val_acc: 0.6261
Epoch 25/150
7352/7352 [==============================] - 19s 3ms/step - loss: 0.4262 - a
cc: 0.8300 - val_loss: 0.4911 - val_acc: 0.7927
Epoch 26/150
7352/7352 [==============================] - 18s 2ms/step - loss: 0.4452 - a
cc: 0.8430 - val_loss: 0.6864 - val_acc: 0.7662
Epoch 27/150
7352/7352 [==============================] - 19s 3ms/step - loss: 0.4289 - a
cc: 0.8369 - val_loss: 0.5333 - val_acc: 0.7492
Epoch 28/150
7352/7352 [==============================] - 20s 3ms/step - loss: 0.4063 - a
cc: 0.8589 - val_loss: 0.5419 - val_acc: 0.7879
Epoch 29/150
7352/7352 [==============================] - 19s 3ms/step - loss: 0.3880 - a
cc: 0.8576 - val_loss: 0.4309 - val_acc: 0.8202
Epoch 30/150
7352/7352 [==============================] - 19s 3ms/step - loss: 0.3455 - a
cc: 0.8829 - val_loss: 0.3907 - val_acc: 0.8385
Epoch 31/150
7352/7352 [==============================] - 60s 8ms/step - loss: 0.3029 - a
cc: 0.9051 - val_loss: 0.3570 - val_acc: 0.8633
Epoch 32/150
7352/7352 [==============================] - 38s 5ms/step - loss: 0.2915 - a
cc: 0.9070 - val_loss: 0.5759 - val_acc: 0.7957
Epoch 33/150
7352/7352 [==============================] - 20s 3ms/step - loss: 0.2799 - a
cc: 0.9153 - val_loss: 0.4633 - val_acc: 0.8412
Epoch 34/150
7352/7352 [==============================] - 20s 3ms/step - loss: 0.2561 - a
cc: 0.9204 - val_loss: 0.3793 - val_acc: 0.8772
Epoch 35/150
7352/7352 [==============================] - 18s 3ms/step - loss: 0.2228 - a
cc: 0.9268 - val_loss: 0.6284 - val_acc: 0.8154
Epoch 36/150
7352/7352 [==============================] - 18s 2ms/step - loss: 0.2249 - a
cc: 0.9289 - val_loss: 0.3307 - val_acc: 0.8877
Epoch 37/150
7352/7352 [==============================] - 18s 2ms/step - loss: 0.2200 - a
cc: 0.9289 - val_loss: 0.4617 - val_acc: 0.8639
```

```
Epoch 38/150
7352/7352 [==============================] - 18s 2ms/step - loss: 0.2146 - a
cc: 0.9289 - val_loss: 0.5114 - val_acc: 0.8280
Epoch 39/150
7352/7352 [==============================] - 18s 2ms/step - loss: 0.2208 - a
cc: 0.9274 - val_loss: 0.4075 - val_acc: 0.8755
Epoch 40/150
7352/7352 [==============================] - 20s 3ms/step - loss: 0.2252 - a
cc: 0.9293 - val_loss: 0.5927 - val_acc: 0.8609
Epoch 41/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.2046 - a
cc: 0.9329 - val_loss: 0.3456 - val_acc: 0.8744
Epoch 42/150
7352/7352 [==============================] - 22s 3ms/step - loss: 0.1863 - a
cc: 0.9350 - val_loss: 0.4016 - val_acc: 0.8660
Epoch 43/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1613 - a
cc: 0.9429 - val_loss: 0.5411 - val_acc: 0.8765
Epoch 44/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1788 - a
cc: 0.9391 - val_loss: 0.4106 - val_acc: 0.8911
Epoch 45/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1865 - a
cc: 0.9370 - val_loss: 0.5549 - val_acc: 0.8843
Epoch 46/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1680 - a
cc: 0.9419 - val_loss: 0.3981 - val_acc: 0.8897
Epoch 47/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1705 - a
cc: 0.9430 - val_loss: 0.4493 - val_acc: 0.8911
Epoch 48/150
7352/7352 [==============================] - 22s 3ms/step - loss: 0.1515 - a
cc: 0.9440 - val_loss: 0.4525 - val_acc: 0.8924
Epoch 49/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1570 - a
cc: 0.9456 - val_loss: 0.3376 - val_acc: 0.8931
Epoch 50/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1379 - a
cc: 0.9486 - val_loss: 0.4029 - val_acc: 0.8972
Epoch 51/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1378 - a
cc: 0.9501 - val_loss: 0.5187 - val_acc: 0.8863
Epoch 52/150
7352/7352 [==============================] - 22s 3ms/step - loss: 0.1511 - a
cc: 0.9455 - val_loss: 0.2806 - val_acc: 0.9060
Epoch 53/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1529 - a
cc: 0.9446 - val_loss: 0.5930 - val_acc: 0.8738
Epoch 54/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1618 - a
cc: 0.9464 - val_loss: 0.4203 - val_acc: 0.9131
Epoch 55/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1450 - a
cc: 0.9459 - val_loss: 0.3740 - val_acc: 0.9233
Epoch 56/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1380 - a
cc: 0.9482 - val_loss: 0.3030 - val_acc: 0.9213
Epoch 57/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1297 - a
cc: 0.9505 - val_loss: 0.3824 - val_acc: 0.9138
Epoch 58/150
```

```
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1385 - a
cc: 0.9506 - val_loss: 0.4033 - val_acc: 0.8985
Epoch 59/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1209 - a
cc: 0.9525 - val_loss: 0.5000 - val_acc: 0.8931
Epoch 60/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1559 - a
cc: 0.9489 - val_loss: 0.4502 - val_acc: 0.9030
Epoch 61/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1270 - a
cc: 0.9508 - val_loss: 0.3570 - val_acc: 0.9114
Epoch 62/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1326 - a
cc: 0.9501 - val_loss: 0.3969 - val_acc: 0.9057
Epoch 63/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1289 - a
cc: 0.9463 - val_loss: 0.3674 - val_acc: 0.8890
Epoch 64/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1398 - a
cc: 0.9465 - val_loss: 0.3477 - val_acc: 0.9162
Epoch 65/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1213 - a
cc: 0.9521 - val_loss: 0.3281 - val_acc: 0.9108
Epoch 66/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1188 - a
cc: 0.9525 - val_loss: 0.3743 - val_acc: 0.9026
Epoch 67/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1493 - a
cc: 0.9471 - val_loss: 0.3318 - val_acc: 0.9074
Epoch 68/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1248 - a
cc: 0.9501 - val_loss: 0.3559 - val_acc: 0.8931
Epoch 69/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1653 - a
cc: 0.9476 - val_loss: 0.2894 - val_acc: 0.9114
Epoch 70/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1320 - a
cc: 0.9490 - val_loss: 0.3508 - val_acc: 0.9030
Epoch 71/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1236 - a
cc: 0.9513 - val_loss: 0.3051 - val_acc: 0.9077
Epoch 72/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1458 - a
cc: 0.9490 - val_loss: 0.3473 - val_acc: 0.9077
Epoch 73/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1564 - a
cc: 0.9461 - val_loss: 0.3579 - val_acc: 0.8955
Epoch 74/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1271 - a
cc: 0.9509 - val_loss: 0.2656 - val_acc: 0.9189
Epoch 75/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1459 - a
cc: 0.9479 - val_loss: 0.4907 - val_acc: 0.8714
Epoch 76/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1146 - a
cc: 0.9557 - val_loss: 0.2982 - val_acc: 0.9209
Epoch 77/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1258 - a
cc: 0.9533 - val_loss: 0.4305 - val_acc: 0.8907
Epoch 78/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1213 - a
```

```
cc: 0.9523 - val_loss: 0.3253 - val_acc: 0.9067
Epoch 79/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1133 - a
cc: 0.9539 - val_loss: 0.3694 - val_acc: 0.9101
Epoch 80/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1140 - a
cc: 0.9540 - val_loss: 0.3964 - val_acc: 0.9104
Epoch 81/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1223 - a
cc: 0.9484 - val_loss: 0.3332 - val_acc: 0.9121
Epoch 82/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1063 - a
cc: 0.9535 - val_loss: 0.3501 - val_acc: 0.9121
Epoch 83/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1111 - a
cc: 0.9548 - val_loss: 0.3935 - val_acc: 0.9114
Epoch 84/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1146 - a
cc: 0.9555 - val_loss: 0.3456 - val_acc: 0.9118
Epoch 85/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1082 - a
cc: 0.9557 - val_loss: 0.3571 - val_acc: 0.9172
Epoch 86/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1224 - a
cc: 0.9517 - val_loss: 0.3891 - val_acc: 0.9050
Epoch 87/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1322 - a
cc: 0.9506 - val_loss: 0.3589 - val_acc: 0.9186
Epoch 88/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1157 - a
cc: 0.9504 - val_loss: 0.3559 - val_acc: 0.9182
Epoch 89/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1245 - a
cc: 0.9506 - val_loss: 0.3787 - val_acc: 0.9203
Epoch 90/150
7352/7352 [==============================] - 22s 3ms/step - loss: 0.1097 - a
cc: 0.9532 - val_loss: 0.4464 - val_acc: 0.9033
Epoch 91/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1263 - a
cc: 0.9490 - val_loss: 0.5101 - val_acc: 0.8992
Epoch 92/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1092 - a
cc: 0.9544 - val_loss: 0.4781 - val_acc: 0.9101
Epoch 93/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1692 - a
cc: 0.9463 - val_loss: 0.3106 - val_acc: 0.9165
Epoch 94/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1118 - a
cc: 0.9544 - val_loss: 0.4521 - val_acc: 0.9203
Epoch 95/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1107 - a
cc: 0.9535 - val_loss: 0.4251 - val_acc: 0.9070
Epoch 96/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1014 - a
cc: 0.9561 - val_loss: 0.4965 - val_acc: 0.8928
Epoch 97/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1043 - a
cc: 0.9550 - val_loss: 0.4268 - val_acc: 0.8894
Epoch 98/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1075 - a
cc: 0.9533 - val_loss: 0.5299 - val_acc: 0.9057
```

```
Epoch 99/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1508 - a
cc: 0.9444 - val_loss: 0.3580 - val_acc: 0.9223
Epoch 100/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1079 - a
cc: 0.9551 - val_loss: 0.4130 - val_acc: 0.9141
Epoch 101/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1013 - a
cc: 0.9544 - val_loss: 0.5840 - val_acc: 0.8870
Epoch 102/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1184 - a
cc: 0.9539 - val_loss: 0.3601 - val_acc: 0.9080
Epoch 103/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.0991 - a
cc: 0.9601 - val_loss: 0.4175 - val_acc: 0.9264
Epoch 104/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1003 - a
cc: 0.9574 - val_loss: 0.3852 - val_acc: 0.9111
Epoch 105/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.1017 - a
cc: 0.9551 - val_loss: 0.4332 - val_acc: 0.9060
Epoch 106/150
7352/7352 [==============================] - 22s 3ms/step - loss: 0.0992 - a
cc: 0.9572 - val_loss: 0.4754 - val_acc: 0.9040
Epoch 107/150
7352/7352 [==============================] - 19s 3ms/step - loss: 0.1007 - a
cc: 0.9547 - val_loss: 0.4354 - val_acc: 0.9152
Epoch 108/150
7352/7352 [==============================] - 19s 3ms/step - loss: 0.1077 - a
cc: 0.9573 - val_loss: 0.5059 - val_acc: 0.8941
Epoch 109/150
7352/7352 [==============================] - 18s 2ms/step - loss: 0.1083 - a
cc: 0.9565 - val_loss: 0.4554 - val_acc: 0.9128
Epoch 110/150
7352/7352 [==============================] - 17s 2ms/step - loss: 0.0925 - a
cc: 0.9592 - val_loss: 0.4940 - val_acc: 0.9118
Epoch 111/150
7352/7352 [==============================] - 20s 3ms/step - loss: 0.1036 - a
cc: 0.9565 - val_loss: 0.4497 - val_acc: 0.9091
Epoch 112/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.0985 - a
cc: 0.9585 - val_loss: 0.3859 - val_acc: 0.8982
Epoch 113/150
7352/7352 [==============================] - 20s 3ms/step - loss: 0.0986 - a
cc: 0.9589 - val_loss: 0.4506 - val_acc: 0.8996
Epoch 114/150
7352/7352 [==============================] - 20s 3ms/step - loss: 0.1007 - a
cc: 0.9578 - val_loss: 0.4147 - val_acc: 0.9165
Epoch 115/150
7352/7352 [==============================] - 20s 3ms/step - loss: 0.0890 - a
cc: 0.9603 - val_loss: 0.5087 - val_acc: 0.9094
Epoch 116/150
7352/7352 [==============================] - 19s 3ms/step - loss: 0.0956 - a
cc: 0.9611 - val_loss: 0.3479 - val_acc: 0.9230
Epoch 117/150
7352/7352 [==============================] - 18s 2ms/step - loss: 0.0896 - a
cc: 0.9591 - val_loss: 0.4622 - val_acc: 0.9186
Epoch 118/150
7352/7352 [==============================] - 18s 2ms/step - loss: 0.0918 - a
cc: 0.9595 - val_loss: 0.4069 - val_acc: 0.9284
Epoch 119/150
```

```
7352/7352 [==============================] - 20s 3ms/step - loss: 0.0850 - a
cc: 0.9612 - val_loss: 0.4365 - val_acc: 0.9125

Epoch 120/150
7352/7352 [==============================] - 23s 3ms/step - loss: 0.0876 - a
cc: 0.9611 - val_loss: 0.4409 - val_acc: 0.9145
Epoch 121/150
7352/7352 [==============================] - 20s 3ms/step - loss: 0.0880 - a
cc: 0.9607 - val_loss: 0.5901 - val_acc: 0.9009
Epoch 122/150
7352/7352 [==============================] - 17s 2ms/step - loss: 0.0940 - a
cc: 0.9616 - val_loss: 0.4040 - val_acc: 0.9006
Epoch 123/150
7352/7352 [==============================] - 17s 2ms/step - loss: 0.0872 - a
cc: 0.9600 - val_loss: 0.4892 - val_acc: 0.9101
Epoch 124/150
7352/7352 [==============================] - 17s 2ms/step - loss: 0.0841 - a
cc: 0.9597 - val_loss: 0.7206 - val_acc: 0.8972
Epoch 125/150
7352/7352 [==============================] - 17s 2ms/step - loss: 0.0898 - a
cc: 0.9596 - val_loss: 0.4337 - val_acc: 0.9101
Epoch 126/150
7352/7352 [==============================] - 18s 2ms/step - loss: 0.0876 - a
cc: 0.9618 - val_loss: 0.4438 - val_acc: 0.8985
Epoch 127/150
7352/7352 [==============================] - 17s 2ms/step - loss: 0.1154 - a
cc: 0.9592 - val_loss: 0.7080 - val_acc: 0.8951
Epoch 128/150
7352/7352 [==============================] - 18s 2ms/step - loss: 0.0991 - a
cc: 0.9610 - val_loss: 0.5178 - val_acc: 0.9009
Epoch 129/150
7352/7352 [==============================] - 18s 3ms/step - loss: 0.0741 - a
cc: 0.9642 - val_loss: 0.4806 - val_acc: 0.9019
Epoch 130/150
7352/7352 [==============================] - 18s 2ms/step - loss: 0.0819 - a
cc: 0.9633 - val_loss: 0.3862 - val_acc: 0.9026
Epoch 131/150
7352/7352 [==============================] - 18s 2ms/step - loss: 0.0809 - a
cc: 0.9621 - val_loss: 0.5859 - val_acc: 0.9097
Epoch 132/150
7352/7352 [==============================] - 18s 2ms/step - loss: 0.0902 - a
cc: 0.9615 - val_loss: 0.4389 - val_acc: 0.9084
Epoch 133/150
7352/7352 [==============================] - 17s 2ms/step - loss: 0.0849 - a
cc: 0.9614 - val_loss: 0.4882 - val_acc: 0.9077
Epoch 134/150
7352/7352 [==============================] - 17s 2ms/step - loss: 0.0802 - a
cc: 0.9621 - val_loss: 0.6001 - val_acc: 0.8985
Epoch 135/150
7352/7352 [==============================] - 17s 2ms/step - loss: 0.0801 - a
cc: 0.9631 - val_loss: 0.4122 - val_acc: 0.9148
Epoch 136/150
7352/7352 [==============================] - 17s 2ms/step - loss: 0.0743 - a
cc: 0.9630 - val_loss: 0.4895 - val_acc: 0.9070
Epoch 137/150
7352/7352 [==============================] - 17s 2ms/step - loss: 0.0762 - a
cc: 0.9653 - val_loss: 0.4474 - val_acc: 0.9108
Epoch 138/150
7352/7352 [==============================] - 17s 2ms/step - loss: 0.0748 - a
cc: 0.9637 - val_loss: 0.4843 - val_acc: 0.8958
Epoch 139/150
```

```
7352/7352 [==============================] - 27s 4ms/step - loss: 0.0837 - a
cc: 0.9627 - val_loss: 0.5640 - val_acc: 0.9053
Epoch 140/150
7352/7352 [==============================] - 24s 3ms/step - loss: 0.0897 - a
cc: 0.9631 - val_loss: 0.5382 - val_acc: 0.8897
Epoch 141/150
7352/7352 [==============================] - 22s 3ms/step - loss: 0.0808 - a
cc: 0.9637 - val_loss: 0.6747 - val_acc: 0.8924
Epoch 142/150
7352/7352 [==============================] - 21s 3ms/step - loss: 0.0883 - a
cc: 0.9607 - val_loss: 0.5981 - val_acc: 0.9101
Epoch 143/150
7352/7352 [==============================] - 18s 2ms/step - loss: 0.0738 - a
cc: 0.9653 - val_loss: 0.5623 - val_acc: 0.9080
Epoch 144/150
7352/7352 [==============================] - 17s 2ms/step - loss: 0.0868 - a
cc: 0.9596 - val_loss: 0.5183 - val_acc: 0.9002
Epoch 145/150
7352/7352 [==============================] - 17s 2ms/step - loss: 0.0800 - a
cc: 0.9633 - val_loss: 0.6155 - val_acc: 0.9114
Epoch 146/150
7352/7352 [==============================] - 18s 2ms/step - loss: 0.0770 - a
cc: 0.9656 - val_loss: 0.5205 - val_acc: 0.9057
Epoch 147/150
7352/7352 [==============================] - 18s 2ms/step - loss: 0.0914 - a
cc: 0.9642 - val_loss: 0.4668 - val_acc: 0.9091
Epoch 148/150
7352/7352 [==============================] - 17s 2ms/step - loss: 0.0718 - a
cc: 0.9663 - val_loss: 0.6681 - val_acc: 0.8992
Epoch 149/150
7352/7352 [==============================] - 17s 2ms/step - loss: 0.0803 - a
cc: 0.9611 - val_loss: 0.6983 - val_acc: 0.9023
Epoch 150/150
7352/7352 [==============================] - 18s 2ms/step - loss: 0.0777 - a
cc: 0.9659 - val_loss: 0.5977 - val_acc: 0.9040
```

In [28]:

```
# Confusion Matrix
print(confusion_matrix(Y_test, model2.predict(X_test)))
```

executed in 12.0s, finished 2018-11-08T19:39:26+05:30

Pred \ True	LAYING	SITTING	STANDING	WALKING	WALKING_DOWNSTAIRS
LAYING	510	0	0	0	0
SITTING	0	410	80	1	0
STANDING	0	122	410	0	0
WALKING	0	0	0	459	22
WALKING_DOWNSTAIRS	0	0	0	1	417
WALKING_UPSTAIRS	0	1	0	0	12

Pred True	WALKING_UPSTAIRS
LAYING	27
SITTING	0
STANDING	0
WALKING	15
WALKING_DOWNSTAIRS	2
WALKING_UPSTAIRS	458

In [29]:

```
score = model2.evaluate(X_test, Y_test)
```

executed in 8.97s, finished 2018-11-08T19:39:37+05:30

```
2947/2947 [==============================] - 9s 3ms/step
```

In [30]:

```
score
```

executed in 353ms, finished 2018-11-08T19:39:39+05:30

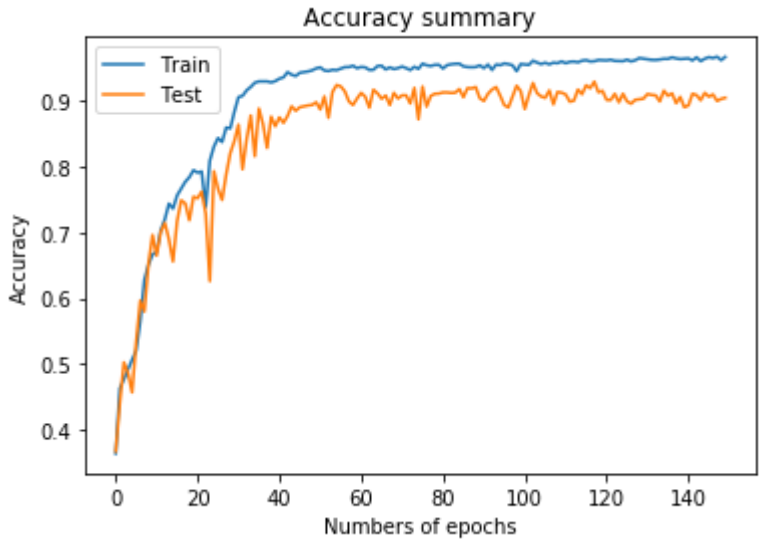
Out[30]:

```
[0.5977042395600973, 0.9039701391245334]
```

In [31]:

```
import matplotlib.pyplot as plt
%matplotlib inline
plt.plot(history2.history['acc'])
plt.plot(history2.history['val_acc'])
plt.xlabel("Numbers of epochs")
plt.ylabel("Accuracy")
plt.title("Accuracy summary")
plt.legend(['Train','Test'],loc='best')
plt.show()
```

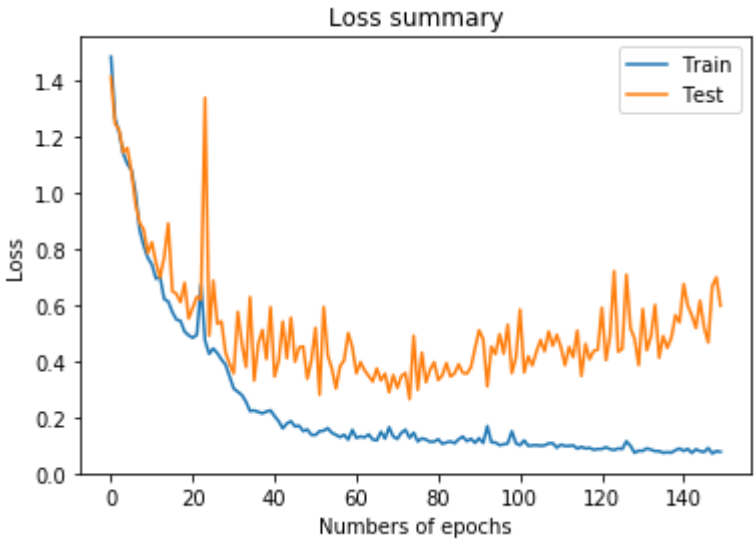
executed in 588ms, finished 2018-11-08T19:39:45+05:30



In [32]:

```
import matplotlib.pyplot as plt
%matplotlib inline
plt.plot(history2.history['loss'])
plt.plot(history2.history['val_loss'])
plt.xlabel("Numbers of epochs")
plt.ylabel("Loss")
plt.title("Loss summary")
plt.legend(['Train','Test'],loc='best')
plt.show()
```

executed in 779ms, finished 2018-11-08T19:39:50+05:30



Model 3 with 2 LSTM layers and 32 LSTM neurons units

In [33]:

```
# Initiliazing the sequential model
model3 = Sequential()
# Configuring the parameters
model3.add(LSTM(32,input_shape=(timesteps, input_dim),return_sequences=True))
# Adding a dropout layer
model3.add(Dropout(0.5))
model3.add(LSTM(32))
model3.add(Dropout(0.5))
# Adding a dense output layer with sigmoid activation
model3.add(Dense(n_classes,activation='sigmoid'))
model3.summary()

```

executed in 4.85s, finished 2018-11-08T19:47:26+05:30

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_3 (LSTM)                (None, 128, 32)           5376
_________________________________________________________________
dropout_3 (Dropout)          (None, 128, 32)           0
_________________________________________________________________
lstm_4 (LSTM)                (None, 32)                8320
_________________________________________________________________
dropout_4 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_3 (Dense)              (None, 6)                 198
=================================================================
Total params: 13,894
Trainable params: 13,894
Non-trainable params: 0
_________________________________________________________________
```

In [34]:

```
# Compiling the model
model3.compile(loss='categorical_crossentropy',optimizer='rmsprop',metrics=['accuracy'
```

executed in 250ms, finished 2018-11-08T19:47:52+05:30

In [35]:

```
# Training the model
history3 = model3.fit(X_train,Y_train,batch_size=batch_size,validation_data=(X_test, Y
                      epochs=100)
```

executed in 59m 27s, finished 2018-11-08T20:47:43+05:30

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/100
7352/7352 [==============================] - 50s 7ms/step - loss: 1.5089 -
acc: 0.4200 - val_loss: 1.3547 - val_acc: 0.4208
Epoch 2/100
7352/7352 [==============================] - 35s 5ms/step - loss: 1.2725 -
acc: 0.4882 - val_loss: 1.2151 - val_acc: 0.5110
Epoch 3/100
7352/7352 [==============================] - 34s 5ms/step - loss: 1.1583 -
acc: 0.5243 - val_loss: 1.1136 - val_acc: 0.5219
Epoch 4/100
7352/7352 [==============================] - 35s 5ms/step - loss: 1.0467 -
acc: 0.5827 - val_loss: 1.0232 - val_acc: 0.5619
Epoch 5/100
7352/7352 [==============================] - 36s 5ms/step - loss: 0.9231 -
acc: 0.6175 - val_loss: 0.9304 - val_acc: 0.6291
Epoch 6/100
7352/7352 [==============================] - 36s 5ms/step - loss: 0.8508 -
acc: 0.6378 - val loss: 0.8052 - val acc: 0.6783
```

In [36]:

```
# Confusion Matrix
print(confusion_matrix(Y_test, model3.predict(X_test)))
```

executed in 17.8s, finished 2018-11-08T20:59:58+05:30

Pred \ True	LAYING	SITTING	STANDING	WALKING	WALKING_DOWNSTAIRS
LAYING	537	0	0	0	0
SITTING	2	362	116	1	1
STANDING	0	45	486	1	0
WALKING	0	0	0	473	1
WALKING_DOWNSTAIRS	0	0	0	3	401
WALKING_UPSTAIRS	0	0	0	23	1

Pred True	WALKING_UPSTAIRS
LAYING	0
SITTING	9
STANDING	0
WALKING	22
WALKING_DOWNSTAIRS	16
WALKING_UPSTAIRS	447

In [37]:

```
scores = model3.evaluate(X_test, Y_test)
```

executed in 16.7s, finished 2018-11-08T21:00:15+05:30

```
2947/2947 [==============================] - 17s 6ms/step
```

In [38]:

```
scores
```

executed in 14ms, finished 2018-11-08T21:00:18+05:30

Out[38]:

```
[0.4626269695854334, 0.9182219205972175]
```

In [39]:

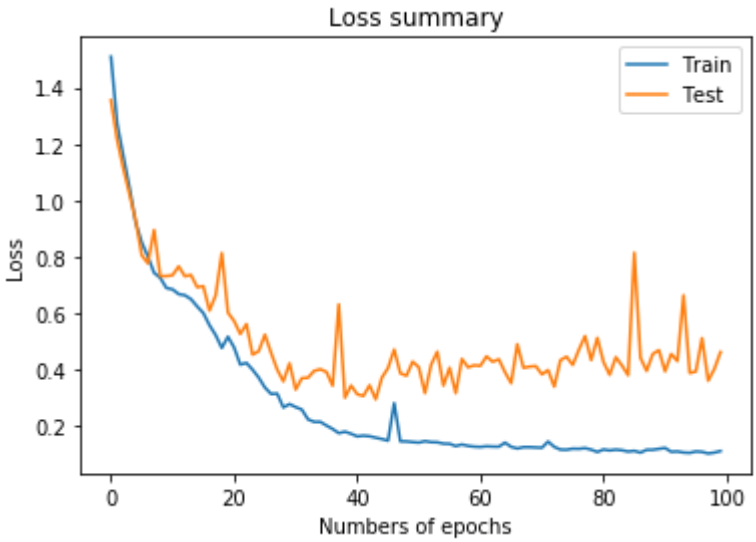
```
import matplotlib.pyplot as plt
%matplotlib inline
plt.plot(history3.history['acc'])
plt.plot(history3.history['val_acc'])
plt.xlabel("Numbers of epochs")
plt.ylabel("Accuracy")
plt.title("Accuracy summary")
plt.legend(['Train','Test'],loc='best')
plt.show()
```

executed in 403ms, finished 2018-11-08T21:00:32+05:30

In [40]:

```
import matplotlib.pyplot as plt
%matplotlib inline
plt.plot(history3.history['loss'])
plt.plot(history3.history['val_loss'])
plt.xlabel("Numbers of epochs")
plt.ylabel("Loss")
plt.title("Loss summary")
plt.legend(['Train','Test'],loc='best')
plt.show()
```

executed in 304ms, finished 2018-11-08T21:00:39+05:30



Model 4 with 2 LSTM layers and 64 LSTM neurons units

In [44]:

```
# Initiliazing the sequential model
model4 = Sequential()
# Configuring the parameters
model4.add(LSTM(64,input_shape=(timesteps, input_dim),return_sequences=True))
# Adding a dropout layer
model4.add(Dropout(0.5))
model4.add(LSTM(64))
model4.add(Dropout(0.5))
# Adding a dense output layer with sigmoid activation
model4.add(Dense(n_classes,activation='sigmoid'))
model4.summary()

```

executed in 294ms, finished 2018-11-08T21:07:04+05:30

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_7 (LSTM)                (None, 128, 64)           18944
_________________________________________________________________
dropout_7 (Dropout)          (None, 128, 64)           0
_________________________________________________________________
lstm_8 (LSTM)                (None, 64)                33024
_________________________________________________________________
dropout_8 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_5 (Dense)              (None, 6)                 390
=================================================================
Total params: 52,358
Trainable params: 52,358
Non-trainable params: 0
_________________________________________________________________
```

In [45]:

```
# Compiling the model
model4.compile(loss='categorical_crossentropy',optimizer='rmsprop',metrics=['accuracy'
```

executed in 61ms, finished 2018-11-08T21:07:07+05:30

In [46]:

```
# Training the model
history4 = model4.fit(X_train,Y_train,batch_size=batch_size,validation_data=(X_test, Y
                      epochs=100)
```

executed in 58m 19s, finished 2018-11-08T22:05:29+05:30

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/100
7352/7352 [==============================] - 35s 5ms/step - loss: 1.3523 -
acc: 0.4470 - val_loss: 1.5567 - val_acc: 0.3125
Epoch 2/100
7352/7352 [==============================] - 36s 5ms/step - loss: 1.1117 -
acc: 0.5214 - val_loss: 1.0850 - val_acc: 0.5545
Epoch 3/100
7352/7352 [==============================] - 34s 5ms/step - loss: 0.9476 -
acc: 0.5887 - val_loss: 0.9974 - val_acc: 0.5948
Epoch 4/100
7352/7352 [==============================] - 33s 4ms/step - loss: 0.8514 -
acc: 0.6049 - val_loss: 0.8233 - val_acc: 0.6396
Epoch 5/100
7352/7352 [==============================] - 33s 4ms/step - loss: 0.7628 -
acc: 0.6511 - val_loss: 0.9773 - val_acc: 0.5979
Epoch 6/100
7352/7352 [==============================] - 33s 4ms/step - loss: 0.7403 -
acc: 0.6479 - val loss: 0.7885 - val acc: 0.6393
```

In [47]:

```
# Confusion Matrix
print(confusion_matrix(Y_test, model4.predict(X_test)))
```

executed in 17.7s, finished 2018-11-08T22:06:42+05:30

Pred \ True	LAYING	SITTING	STANDING	WALKING	WALKING_DOWNSTAIRS
LAYING	510	0	25	0	0
SITTING	3	436	51	0	1
STANDING	0	112	417	2	0
WALKING	0	2	1	466	0
WALKING_DOWNSTAIRS	0	0	0	0	415
WALKING_UPSTAIRS	0	0	0	2	5

Pred True	WALKING_UPSTAIRS
LAYING	2
SITTING	0
STANDING	1
WALKING	27
WALKING_DOWNSTAIRS	5
WALKING_UPSTAIRS	464

In [48]:

```
score4 = model4.evaluate(X_test, Y_test)
```

executed in 16.6s, finished 2018-11-08T22:07:08+05:30

```
2947/2947 [==============================] - 17s 6ms/step
```

In [50]:

```
score4
```

executed in 6ms, finished 2018-11-08T22:07:18+05:30

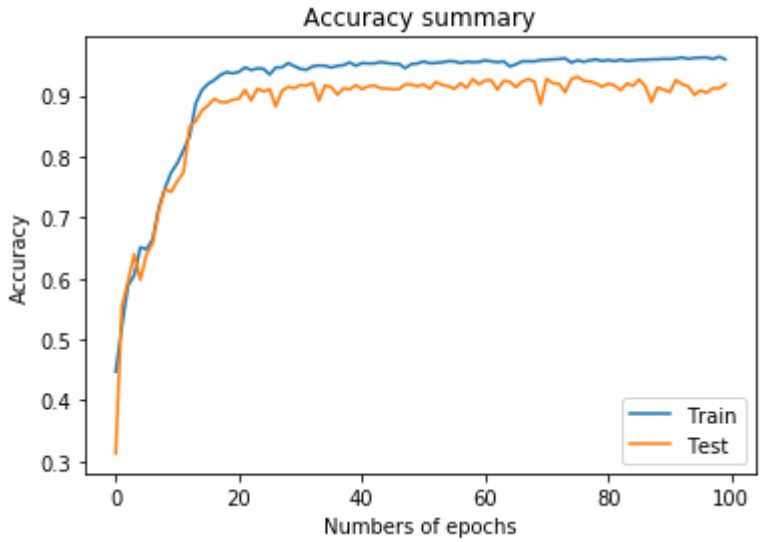
Out[50]:

```
[0.4141320268444936, 0.9189005768578216]
```

In [51]:

```
import matplotlib.pyplot as plt
%matplotlib inline
plt.plot(history4.history['acc'])
plt.plot(history4.history['val_acc'])
plt.xlabel("Numbers of epochs")
plt.ylabel("Accuracy")
plt.title("Accuracy summary")
plt.legend(['Train','Test'],loc='best')
plt.show()
```

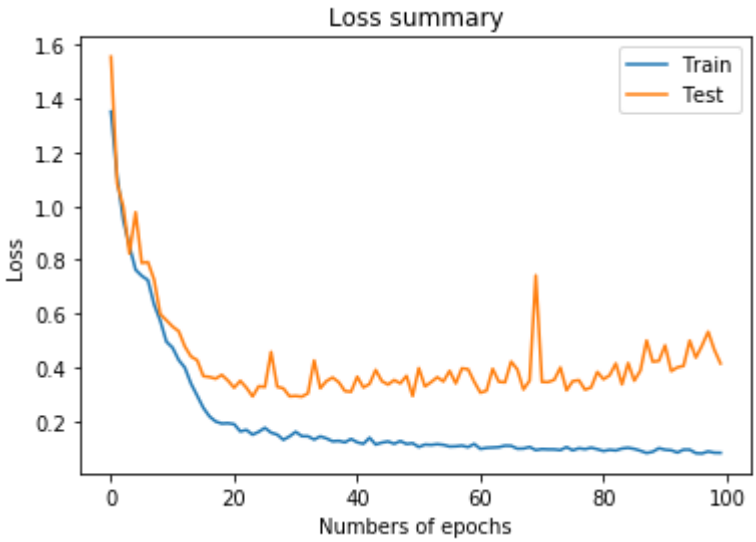
executed in 160ms, finished 2018-11-08T22:07:59+05:30



In [52]:

```
import matplotlib.pyplot as plt
%matplotlib inline
plt.plot(history4.history['loss'])
plt.plot(history4.history['val_loss'])
plt.xlabel("Numbers of epochs")
plt.ylabel("Loss")
plt.title("Loss summary")
plt.legend(['Train','Test'],loc='best')
plt.show()
```

executed in 644ms, finished 2018-11-08T22:08:07+05:30



In [59]:

```
# Initiliazing the sequential model
model5 = Sequential()
# Configuring the parameters
model5.add(LSTM(128,input_shape=(timesteps, input_dim),return_sequences=True))
# Adding a dropout layer
model5.add(Dropout(0.5))

model5.add(LSTM(64,return_sequences=True))
model5.add(Dropout(0.5))

model5.add(LSTM(32))
model5.add(Dropout(0.5))
# Adding a dense output layer with sigmoid activation
model5.add(Dense(n_classes,activation='sigmoid'))
model5.summary()
```

executed in 534ms, finished 2018-11-08T22:15:25+05:30

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_20 (LSTM)               (None, 128, 128)          70656
_________________________________________________________________
dropout_18 (Dropout)         (None, 128, 128)          0
_________________________________________________________________
lstm_21 (LSTM)               (None, 128, 64)           49408
_________________________________________________________________
dropout_19 (Dropout)         (None, 128, 64)           0
_________________________________________________________________
lstm_22 (LSTM)               (None, 32)                12416
_________________________________________________________________
dropout_20 (Dropout)         (None, 32)                0
_________________________________________________________________
dense_8 (Dense)              (None, 6)                 198
=================================================================
Total params: 132,678
Trainable params: 132,678
Non-trainable params: 0
_________________________________________________________________
```

In [60]:

```
# Compiling the model
model5.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

executed in 30ms, finished 2018-11-08T22:15:27+05:30

In [61]:

```
# Training the model
history5 = model5.fit(X_train,Y_train,batch_size=batch_size,validation_data=(X_test, Y
                      epochs=100)
```

executed in 1h 43m 60s, finished 2018-11-08T23:59:27+05:30

```
Train on 7352 samples, validate on 2947 samples
Epoch 1/100
7352/7352 [==============================] - 53s 7ms/step - loss: 1.5547 -
acc: 0.3655 - val_loss: 1.7976 - val_acc: 0.2827
Epoch 2/100
7352/7352 [==============================] - 55s 8ms/step - loss: 1.4058 -
acc: 0.3942 - val_loss: 1.3183 - val_acc: 0.5165
Epoch 3/100
7352/7352 [==============================] - 67s 9ms/step - loss: 1.2437 -
acc: 0.4761 - val_loss: 1.2336 - val_acc: 0.4425
Epoch 4/100
7352/7352 [==============================] - 80s 11ms/step - loss: 1.2448
- acc: 0.4359 - val_loss: 1.1914 - val_acc: 0.4941
Epoch 5/100
7352/7352 [==============================] - 84s 11ms/step - loss: 1.0748
- acc: 0.5467 - val_loss: 0.9838 - val_acc: 0.6094
Epoch 6/100
7352/7352 [==============================] - 86s 12ms/step - loss: 1.0465
- acc: 0.5505 - val loss: 1.0104 - val acc: 0.5684
```

In [62]:

```
# Confusion Matrix
print(confusion_matrix(Y_test, model5.predict(X_test)))
```

executed in 33.7s, finished 2018-11-09T00:03:25+05:30

Pred \ True	LAYING	SITTING	STANDING	WALKING	WALKING_DOWNSTAIRS
LAYING	537	0	0	0	0
SITTING	22	414	54	0	1
STANDING	0	107	424	1	0
WALKING	0	0	0	432	7
WALKING_DOWNSTAIRS	0	0	0	0	412
WALKING_UPSTAIRS	0	0	1	0	22

Pred True	WALKING_UPSTAIRS
LAYING	0
SITTING	0
STANDING	0
WALKING	57
WALKING_DOWNSTAIRS	8
WALKING_UPSTAIRS	448

In [63]:

```
score5 = model5.evaluate(X_test, Y_test)
```

executed in 27.4s, finished 2018-11-09T00:03:56+05:30

```
2947/2947 [==============================] - 27s 9ms/step
```

In [64]:

```
score5
```

executed in 8ms, finished 2018-11-09T00:03:59+05:30

Out[64]:

```
[0.3722532238988386, 0.9049881235154394]
```

In [65]:

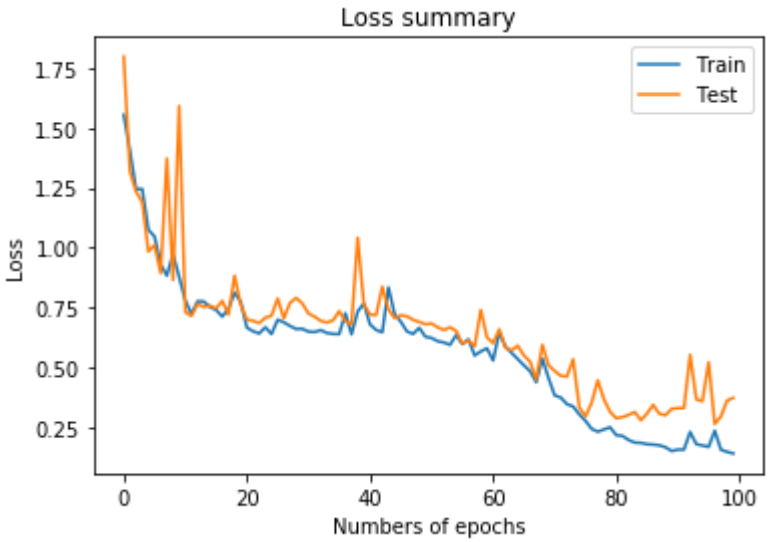
```
import matplotlib.pyplot as plt
%matplotlib inline
plt.plot(history5.history['acc'])
plt.plot(history5.history['val_acc'])
plt.xlabel("Numbers of epochs")
plt.ylabel("Accuracy")
plt.title("Accuracy summary")
plt.legend(['Train','Test'],loc='best')
plt.show()
```

executed in 177ms, finished 2018-11-09T00:04:39+05:30

In [66]:

```
import matplotlib.pyplot as plt
%matplotlib inline
plt.plot(history5.history['loss'])
plt.plot(history5.history['val_loss'])
plt.xlabel("Numbers of epochs")
plt.ylabel("Loss")
plt.title("Loss summary")
plt.legend(['Train','Test'],loc='best')
plt.show()
```

executed in 136ms, finished 2018-11-09T00:05:24+05:30



Analysis of Human Activity Recoginition using deep learning models

In [70]:

```
df = pd.DataFrame(columns=['Model Nos','LSTM Layres','LSTM Cells','Epochs','Accuracy %
df = df.append(pd.DataFrame([['model 1','1 layers','32','150',0.8971*100,0.3797*100]],
                            columns=['Model Nos','LSTM Layres','LSTM Cells','Epochs','A
df = df.append(pd.DataFrame([['model 2','1 layers','64','150',0.9039*100,0.5977*100]],
                            columns=['Model Nos','LSTM Layres','LSTM Cells','Epochs','A
df = df.append(pd.DataFrame([['model 3','2 layers','32','100',0.9182*100,0.4626*100]],
                            columns=['Model Nos','LSTM Layres','LSTM Cells','Epochs','A

df = df.append(pd.DataFrame([['model 4','2 layers','64','100',0.9189*100,0.4141*100]],
                            columns=['Model Nos','LSTM Layres','LSTM Cells','Epochs','A

df = df.append(pd.DataFrame([['model 5','3 layers','128','100',0.9049*100,0.3722*100]]
                            columns=['Model Nos','LSTM Layres','LSTM Cells','Epochs','A


df.reset_index(drop=True,inplace=True)
```

executed in 18ms, finished 2018-11-09T00:18:11+05:30

In [71]:

```
1 df
```

executed in 11ms, finished 2018-11-09T00:18:12+05:30

Out[71]:

	Model Nos	LSTM Layres	LSTM Cells	Epochs	Accuracy %	Loss %
0	model 1	1 layers	32	150	89.71	37.97
1	model 2	1 layers	64	150	90.39	59.77
2	model 3	2 layers	32	100	91.82	46.26
3	model 4	2 layers	64	100	91.89	41.41
4	model 5	3 layers	128	100	90.49	37.22

Conclusion

- We did Exploratory data analysis on HAR dataset.
- We applied the classical machine learning algorithm such as Logistic Regression,Liner SVC,SVM with kernels,Decision Tree,Gradient Boosting Decision Tree,Random Forest.
- We tuned the hyperparametrs to find out best hyperparametres for each ML algorithms using GridSearchCV.
- Plotted the confusion matrix.
- Applied the deep learning model such as **LSTM**(Long Short Term Memory).
- Applied **LSTM** with differents numbers of layers and nuerons to find out the best accuracy for this case study.
- Classical ML algorithm give the higher accuracy(Best accuracy is :**96.81%** for **Liner SVC**) compare to deep learning model **LSTM**(Best accuracy is **91.89%**).
- plotted train and validation error for LSTM.
- Here we are getting low accuracy for deep learning models **LSTM** than classical ML algorithms because deep learning model is applied where there is large numbers of data points.

In [ ]:

```
1
```