MINI PROJECT REPORT

ON

HUMAN ACTIVITY RECOGNISATION USING RNN AND LSTM

NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA

SURATHKAL, 575025

COURSE TITLE

Soft Computing (IT355)

SUBMITTED BY:

1. Sowmya (15IT246)

2. Deena (15IT209)

3. Alekhya (15IT226)

4. Neha (15IT223)

SUBMITTED TO:

Dr. Nagamma Patil (Course Instructor)

CERTIFICATE

This is to certify that the project entitled **"Human Activity Recognization Using RNN and LSTM"** has been presented by Deena , Alekhya, Sowmya, Neha students of third year (V sem), B.Tech (IT), Department of Information Technology, National Institute of Technology Karnataka, Surathkal, on November 10, 2017, during the odd semester of the academic year 2017- 2018, in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Information Technology at NITK, Surathkal.

Place: NITK, Surathkal

Date: November 10, 2017 (Signature of the Examiner)

ACKNOWLEDGEMENT

This project would not have been possible without the kind support and help of many individuals and organizations. We would like to extend our sincere thanks to all of them. We are highly indebted to the Department of Information Technology, NITK for their guidance and constant supervision as well as for providing necessary information and resources for the project and also for their support in completing the project.

We would like to express our gratitude towards our teacher Dr Nagamma Patil for her kind co-operation and encouragement which helped us in completion of this project. Our thanks and appreciations also go to our group members in developing the project and people who have willingly helped us out with their abilities.

DECLARATION

We, Deena(15IT209), Alekhya(15IT226), Sowmya(15IT246), Neha(15IT223) students of 5th semester B. Tech in Information Technology, National Institute of Technology Karnataka, Surathkal, hereby declare that the mini project work entitled **"Human Activity Recognization Using RNN and LSTM"** submitted to Mrs. Nagamma Patil, Assistant Professor, Department of Information Technology, NITK Surathkal during the academic year 2017-2018, is a record of an original work done by us under the guidance of her. This project work is submitted in partial fulfillment of the requirements for the award of the degree of the Bachelor of Technology in Information Technology at NITK Surathkal.

Date : November 10, 2017

Place : NITK Surathkal

INDEX

List of Figures

1. INTRODUCTION

Human activity recognition (HAR) tasks have traditionally been solved using engineered features obtained by heuristic processes. Current research suggests that deep convolution neural networks are suited to automate feature extraction from raw sensor inputs. However, human activities are made of complex sequences of motor movements, and capturing this temporal dynamics is fundamental for successful HAR. Based on the recent success of recurrent neural networks for time series domains, we propose a generic deep framework for activity recognition based on convolutional and LSTM recurrent units, which: (i) is suitable for multimodal wearable sensors; (ii) can perform sensor fusion naturally; (iii) does not require expert knowledge in designing features; and (iv) explicitly models the temporal dynamics of feature activations.

Human activity recognition using smart phones dataset and an LSTM RNN. Classifying the type of movement amongst six categories:

- WALKING,
- WALKING_UPSTAIRS,
- WALKING_DOWNSTAIRS,
- SITTING,
- STANDING,
- LAYING.

2. LITERATURE SURVEY

2.1 RELATED WORK

What is an RNN?

As explained in this project, an RNN takes many input vectors to process them and output other vectors. It can be roughly pictured like in the fig 1 below, imagining each rectangle has a vectorial depth and other special hidden quirks in the image below. In our case, the "many to one" architecture is used: we accept time series of feature vectors (one vector per time step) to convert them to a probability vector at the output for classification. Note that a "one to one" architecture would be a standard feedforward neural network.

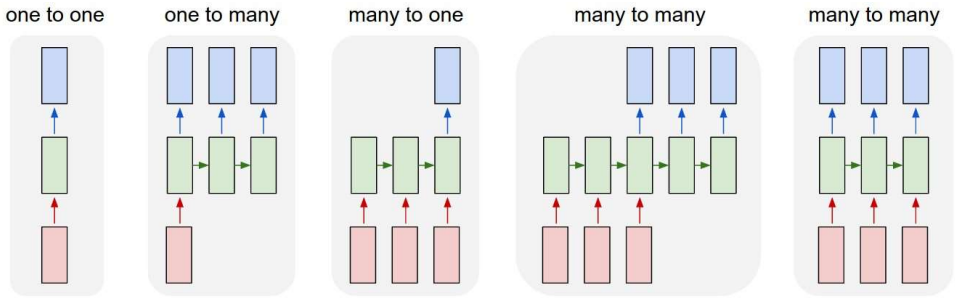

Fig1: Types of architectures of RNN

An LSTM is an improved RNN. It is more complex, but easier to train, avoiding what is called the vanishing gradient problem.

Recurrent networks, on the other hand, take as their input not just the current input example they see, but also what they have perceived previously in time. Here's a diagram of an early, simple recurrent net, where the *BTSXPE* at the bottom of the drawing represents the input example in the current moment, and *CONTEXT UNIT* represents the output of the previous moment.

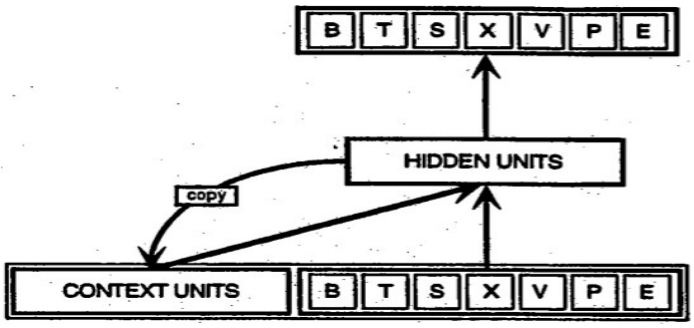

Fig 2: Example of showing simple recurrent net

The decision a recurrent net reached at time step t-1 affects the decision it will reach one moment later at time step t. So recurrent networks have two sources of input, the present and the recent past, which combine to determine how they respond to new data, much as we do in life.

HOW RNN IS DIFFERENT FROM OTHER FEED FORWARD NEURAL NETWORK?

Recurrent networks are distinguished from feedforward networks by that feedback loop connected to their past decisions, ingesting their own outputs moment after moment as input. It is often said that recurrent networks have memory.[2] Adding memory to neural networks has a purpose: There is information in the sequence itself, and recurrent nets use it to perform tasks that feedforward networks can't.

That sequential information is preserved in the recurrent network's hidden state, which manages to span many time steps as it cascades forward to affect the processing of each new example. It is finding correlations between events separated by many moments, and these correlations are called "long-term dependencies", because an event downstream in time depends upon, and is a function of, one or more events that came before.

LSTMs for Human Activity Recognition::

LSTM Networks:

Long Short Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies.They work tremendously well on a large variety of problems, and are now widely used.

LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn.

All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.

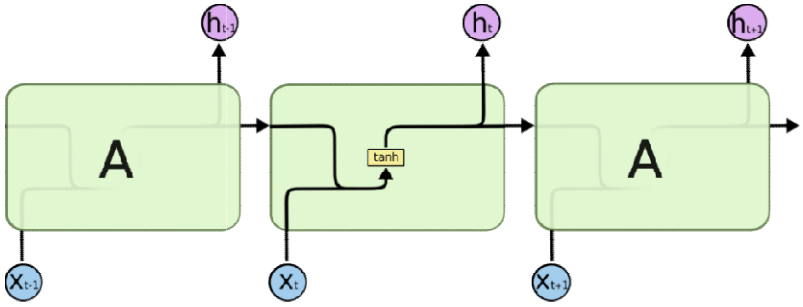

Fig 3: The repeating module in a standard RNN contains a single layer.

LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.

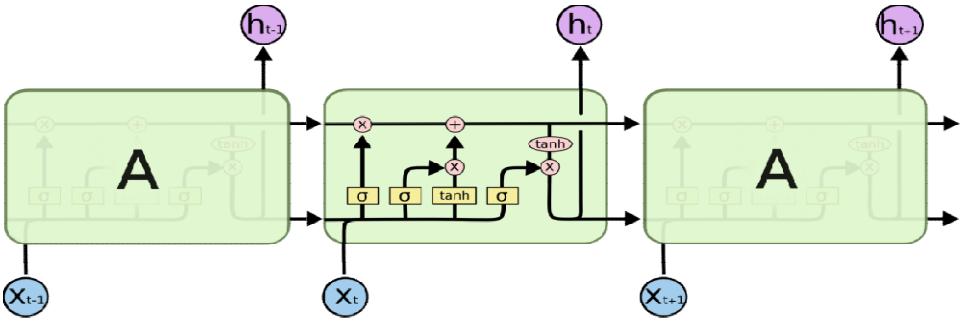

Fig 4: The repeating module in an LSTM contains four interacting layers.

In the above diagram, each line carries an entire vector, from the output of one node to the inputs of others. The pink circles represent pointwise operations, like vector addition, while the yellow boxes are learned neural network layers. Lines merging denote concatenation, while a line forking denote its content being copied and the copies going to different locations.

2.2 PROBLEM STATEMENT

Human activity recognition using smartphones dataset and an LSTM RNN. An LSTM Neural Network (implemented in TensorFlow) is trained for Human Activity Recognition (HAR) from accelerometer data.

2.3 OBJECTIVES

The objectives of this project are the following:

- This project present LSTM: a deep learning framework composed of convolutional and LSTM recurrent layers, that is capable of automatically learning feature representations and modelling the temporal dependencies
- Also demonstrate that this framework is suitable for activity recognition from wearable sensor data by using it on two families of human activity recognition problems, that of static/periodic activities (modes of locomotion and postures) and that of sporadic activities (gestures).
- Shows that the framework can be applied seamlessly to different sensor modalities individually and that it can also fuse them to improve performance. We demonstrate this on accelerometers, gyroscopes and combinations thereof.

3. PROPOSED METHODOLOGY

- Here download_dataset.py file will download dataset. And if dataset is already there then it prints its location.
- Load_X will train and test the neural network inputs
- Load_Y will tain and test the neural network outputs
- There are some core parameter definitions for the training.
- The whole neural network's structure could be summarised by enumerating those parameters and the fact an LSTM is used
- LSTM_RNN Function returns a tensorflow LSTM (RNN) artificial neural network from given parameters.
- Our LSTM (covered in the previous part of the series) model expects fixed-length sequences as training data. We'll use a familiar method for generating these. Each generated sequence contains 200 training examples.
- Our training dataset has drastically reduced size after the transformation.
- The shape of our tensor looks kinda strange. Let's transform it into sequences of 128 rows, each containing x, y and z. Let's apply a one-hot encoding to our labels, as well
- Finally, split the data into training and test (20%) set. Split data because rnn cell needs a list of inputs for the RNN inner loop.
- Our model contains 2 fully-connected and 2 LSTM layers (stacked on each other) with 32 units each.
- Get last time step's output feature for a "many to one" style classifier.
- extract_batch_size ,function to fetch a "batch_size" amount of data from "(X|y)_train" data.
- one_hot(y_): function to encode output labels from number indexes
- Now build the neural network::
    - Get graph input and output and save them.
    - Randomly initialize weights and biases of out and hidden layers.
    - Now calculate loss,as it prevents this overkill neural network to overfit the data.
    - And also calculate cost,optimizer, correct_pred,accuracy.
- Now train the neural network::
    - Launch the graph.
    - Perform Training steps with "batch_size" amount of example data at each loop.
    - Evaluate network only at some steps for faster training.

- Now evaluate on the test data(no learning made here).
- Calculate accuracy for test data.
- Finally print the results for batch loss and accuracy.

4. RESULTS AND ANALYSIS

Output for downloading and showing the data path for data set ::

```
/home/sowmya/LSTM-Human-Activity-Recognition-master
1.png     LSTM_files  Screenshot from 2017-11-04 20-24-29.png
data      LSTM.ipynb  Screenshot from 2017-11-04 20-50-55.png
last.png  lstm.py     Untitled1.ipynb
LICENSE  README.md  Untitled.ipynb
/home/sowmya/LSTM-Human-Activity-Recognition-master/data
download_dataset.py  source.txt  UCI HAR Dataset  UCI HAR Dataset.zip
Downloading...
Dataset already downloaded. Did not download twice.
Extracting...
Dataset already extracted. Did not extract twice.
/home/sowmya/LSTM-Human-Activity-Recognition-master/data
download_dataset.py  source.txt  UCI HAR Dataset  UCI HAR Dataset.zip
/home/sowmya/LSTM-Human-Activity-Recognition-master
1.png     LSTM_files  Screenshot from 2017-11-04 20-24-29.png
data      LSTM.ipynb  Screenshot from 2017-11-04 20-50-55.png
last.png  lstm.py     Untitled1.ipynb
LICENSE  README.md  Untitled.ipynb
Dataset is now located at: data/UCI HAR Dataset/
```

Output showing dataset's shape and normalisation::

Some useful info to get an insight on dataset's shape and normalisation: (X shape, y shape, every X's mean, every X's standard deviation) (2947, 128, 9) (2947, 1) 0.0991399 0.395671 .The dataset is therefore properly normalised, as expected, but not yet one-hot encoded.

Output showing accuracy and batch loss::

Training iter #1500: Batch Loss = 3.948874, Accuracy = 0.13333334028720856

PERFORMANCE ON TEST SET: Batch Loss = 3.062919855117798, Accuracy = 0.20563285052776337

Training iter #30000: Batch Loss = 1.300843, Accuracy = 0.6966666579246521

PERFORMANCE ON TEST SET: Batch Loss = 1.4882047176361084, Accuracy = 0.5866983532905579

Training iter #60000: Batch Loss = 1.191107, Accuracy = 0.7606666684150696

PERFORMANCE ON TEST SET: Batch Loss = 1.2652212381362915, Accuracy = 0.7319307923316956

Training iter #90000: Batch Loss = 1.005859, Accuracy = 0.8193333148956299

PERFORMANCE ON TEST SET: Batch Loss = 1.145902395248413, Accuracy = 0.7807940244674683

...

...

...

Training iter #2160000: Batch Loss = 0.259621, Accuracy = 0.987333357334137

PERFORMANCE ON TEST SET: Batch Loss = 0.5876603126525879, Accuracy = 0.8934509754180908

Training iter #2190000: Batch Loss = 0.279541, Accuracy = 0.9833333492279053

PERFORMANCE ON TEST SET: Batch Loss = 0.6032086610794067, Accuracy = 0.8920936584472656

Optimization Finished!

FINAL RESULT: Batch Loss = 0.6084891557693481, Accuracy = 0.8897183537483215

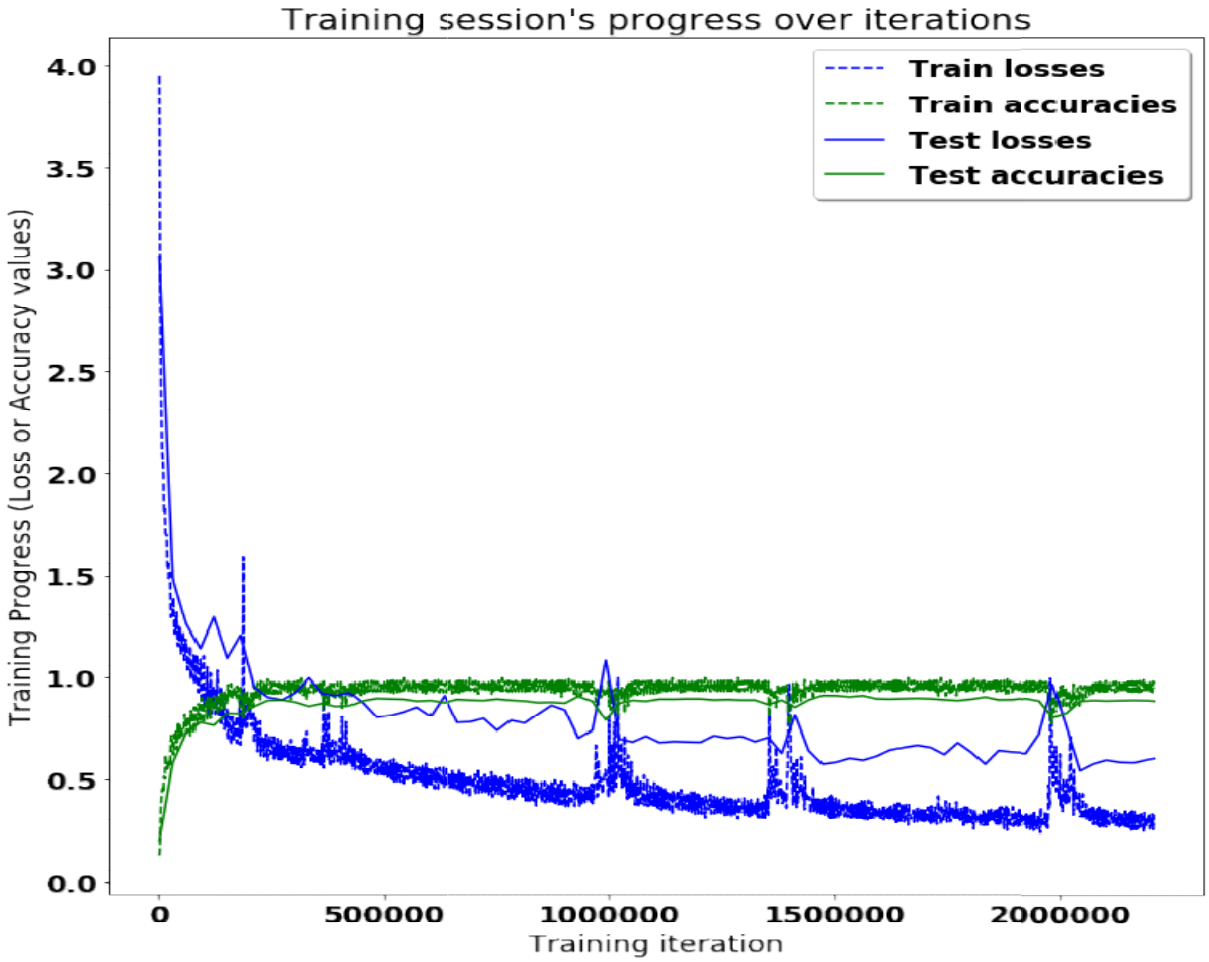

Fig 5:Output showing graph of training session's progress over iterations

Output showing testing accuracy and confusion matrix::

Testing Accuracy: 88.97183537483215%

Precision: 88.99611282611261%

Recall: 88.97183576518493%

f1_score: 88.91043032299801%

Confusion Matrix:

[[461 3 31 1 0 0]

[32 413 26 0 0 0]

[15 7 398 0 0 0]

[0 18 0 399 68 6]

[3 7 0 108 414 0]

[0 0 0 0 0 537]]

Confusion matrix (normalised to % of total test data):

[[15.64302731 0.10179844 1.0519172 0.03393281 0. 0.]

[1.08585 14.01425171 0.88225317 0. 0. 0.]

[0.5089922 0.2375297 13.50525951 0. 0. 0.]

[0. 0.61079061 0. 13.5391922 2.30743122 0.20359688]

[0.10179844 0.2375297 0. 3.6647439 14.04818439 0.]

[0. 0. 0. 0. 0. 18.22192001]]

Note: training and testing data is not equally distributed amongst classes,

so it is normal that more than a 6th of the data is correctly classifier in the last category.

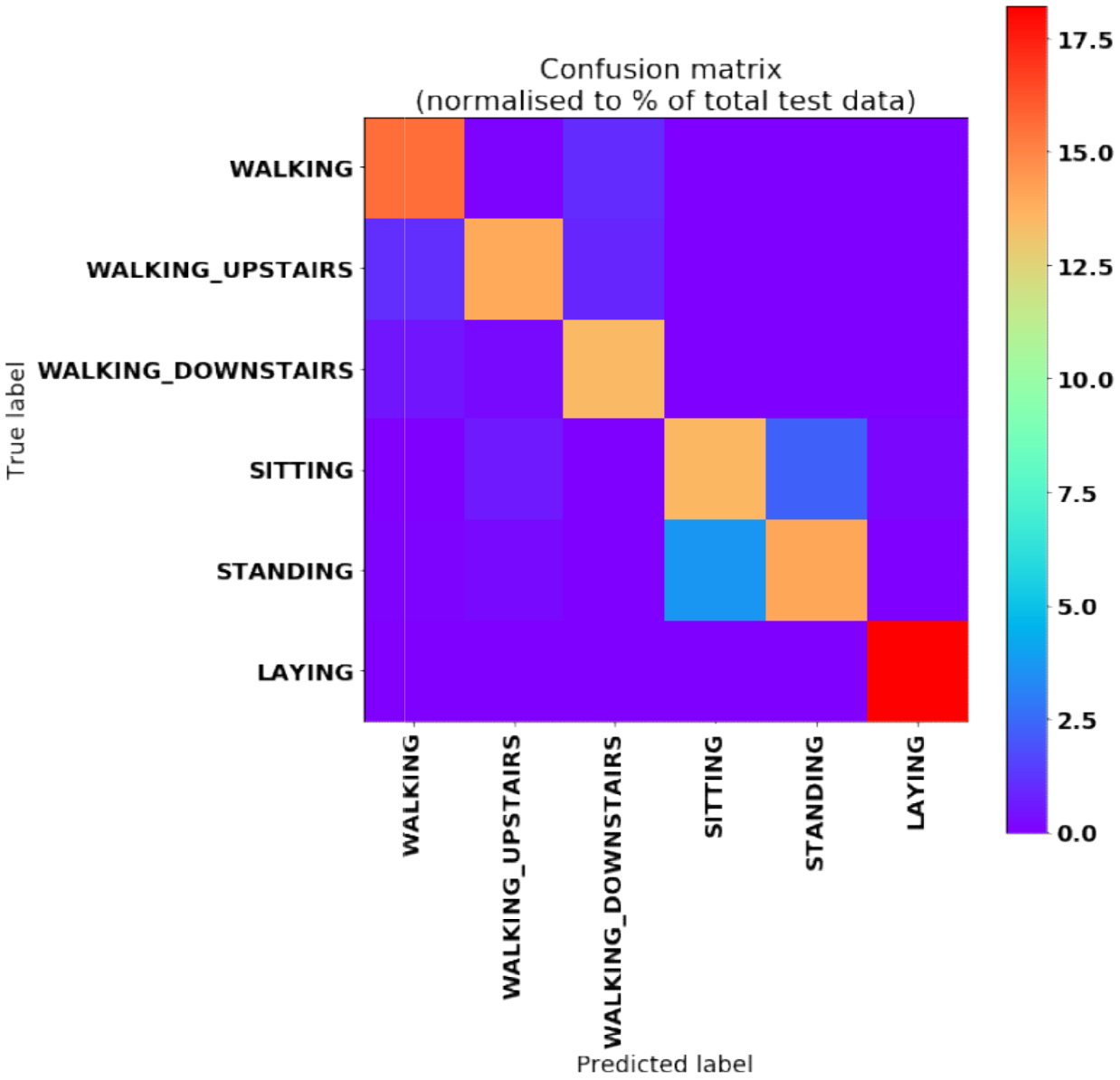

Fig 6: Output showing Confusion Matrix

5. CONCLUSION AND FUTURE WORKS

In this paper,human activities are recognised using RNN and LSTM. Outstandingly, **the final accuracy is of 88%**! And it can peak to values such as 92.73%, at some moments of luck during the training, depending on how the neural network's weights got initialized at the start of the training, randomly.

This means that the neural networks is almost always able to correctly identify the movement type! Remember, the phone is attached on the waist and each series to classify has just a 128 sample window of two internal sensors (a.k.a. 2.56 seconds at 50 FPS), so those predictions are extremely accurate.

Those are seemingly almost the same thing from the point of view of a device placed at waist level according to how the dataset was gathered. It is still possible to see a little cluster on the matrix between those classes, which drifts away from the identity.

6. REFERENCES

LSTMs for Human Activity Recognition

[1] Activity Recognition using Cell Phone Accelerometers

[2] WIreless Sensor Data Mining Lab

[3] http://www.mdpi.com/1424-8220/16/1/115/htm

[4] http://people.idsia.ch/~juergen/rnn.html

[5] http://people.idsia.ch/~juergen/rnn.html