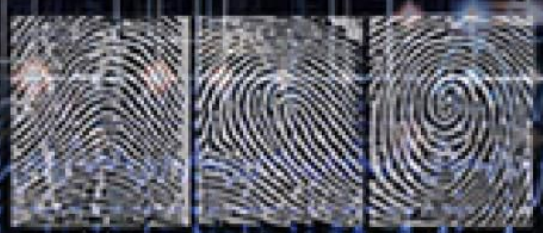
DEEP LEARNING NEURAL NETWORKS

Design and Case Studies

Daniel Graupe



World Scientific

DEEP LEARNING NEURAL NETWORKS

Design and Case Studies

Daniel Graupe

University of Illinois, Chicago, USA

World Scientific

NEW JERSEY • LONDON • SINGAPORE • BEIJING • SHANGHAI • HONG KONG • TAIPEI • CHENNAI • TOKYO

Published by

World Scientific Publishing Co. Pte. Ltd.

5 Toh Tuck Link, Singapore 596224

USA office: 27 Warren Street, Suite 401-402, Hackensack, NJ 07601

UK office: 57 Shelton Street, Covent Garden, London WC2H 9HE

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library.

DEEP LEARNING NEURAL NETWORKS
Design and Case Studies

ISBN 978-981-3146-44-0
ISBN 978-981-3146-45-7 (pbk)

Printed in Singapore

Contents

CHAPTER 8

Case Studies

Comment: In the discussions concerning the LNN-1 and LNN-2 deep learning networks in the Case Studies below, learning never stops after training but is carried out for all samples, and at all iterations from the first sample on. However, an initial number of datasets in these 2 networks is named as the training sets whereas all later datasets are considered as testing sets and performance statistics are computed only for later datasets. In this way, the number of data sets for training and for testing is the same for all neural networks considered in a given case study. Only when a pre-training algorithm such as the data-balancing pre-training algorithm of Section 6.5 above is employed, will this pre-training be considered in the training-mode statistics of comparing neural networks. Still, in none of the case studies in this text was this kind of pre-training employed.

This aspect of LAMSTAR must be taken into account when it is compared with other networks, where the meaning of training is different.

8.1. Human Activities Recognition (A Bose)

The goal of this Case Study is to apply CNN, LNN-1 and LNN-2 deep learning neural networks to the problem of recognizing human activities Classification problem and to compare the performance of these three networks and their respective computational times. All 3 networks considered in this study share the same input data and the same preprocessing. Comparison is also made with results from 18 other recently published (2012–2015) studies on the same problem and for the same database (see the Results tabulations of the present study, below).

DATA: The data for this Case Study comes from 2 human activities datasets: (1) the MSRDailyActivity3D [Microsoft. MRSDaily] from the Microsoft Research Laboratory, and (2) CAD-60 from Computer Science Department, Cornell University [Cornell. CAD60, 2009]. The data is of RGBD (RGB and Depth) images from Kincet sensors. Of 16 daily activities in MRSDaily, six were selected (Eating, Talking on cellphone, Standing up, Sitting still, Standing still, Walking). In these 6 activities 7590 different poses served for training and 600 for testing. Of 12 activities in CAD-60, 5 were selected for this study ((Brushing teeth, talking on phone, drinking water, cooking/chopping, working on computer).

PREPROCESSING: Our data is in terms of 3D images. Hence we must consider 3D Euclidean coordinates of 20 joints (see Fig. 8.1): 1. Hip Center, 2. Spine, 3. Shoulder Center, 4. Head, 5. Shoulder Right, 6. Elbow Right, 7. Wrist Right, 8. Hand Right, 9. Shoulder Left, 10. Elbow Left, 11. Wrist Left, 12. Hand Left, 13. Hip Right, 14. Knee Right, 15. Ankle Right, 16. Foot Right, 17. Hip Left, 18. Knee Left, 19. Ankle Left, 20. Foot Left. Therefore, the body orientation must be preprocessed (computed) to achieve a view-invariant activity recognition. After normalization of the above, we obtain 60 coordinates from 20 body-joints for each frame (pose of human activity). By adding zeros, we then get a 1×64 input vector that is arranged as an 8×8 input matrix and which serves as input to the neural networks considered in this Case Study.

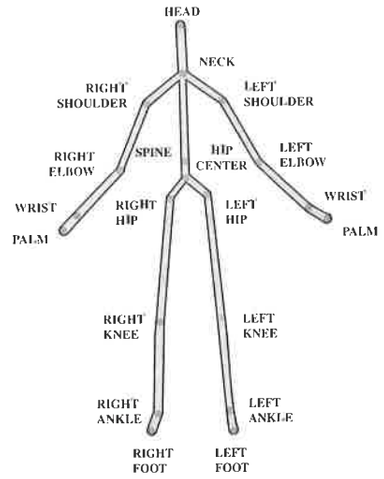


Fig. 8.1. Skeleton body joints given by Kincet.

Table 8.1a. Precision and recall of present and previous results on MRSDaily database — Human activity classification.

Method	Accuracy (%)
LOP [Wang J, 2012]	42.5
Depth motion maps [Yang X, 2012]	43.13
Joint position [Wang J, 2012]	68
Moving pose [Zanfir, 2013	73.8
Local HOV 4D [Oreifej, 2013]	80
Actionlet ensemble [Wang J, 2012]	85.75
SNV [Yang X, 2014]	86.25
HDMM+3ConvNets [Wang P, 2015]	81.88
CNN (present study)	93
LNN-1 (present study)	95.33
LNN-2 (present study)	99.67

Table 8.1b. Precision and recall of present and previous results on CD-60 database — Human activity classification.

Method	Precision	Recall
MEMM [Sung, 2011], [Sung, 2012]	67.9	55.5
SSVM [Koppula, 2013]	80.8	71.4
Structure-Motion Features [Zhang C, 2012]	86	84
NBNN [Yang X, 2013]	71.9	66.6
Image Fusion [Ni B, 2013]	75.9	69.5
Spatial-based Clustering [Gupta R, 2013]	78.1	75.4
K-means Clusterings+SVM+HMM [Gaglio, 2014]	77.3	76.7
S-ONI [Parisi, 2015]	91.9	90.2
SI Point Feature [Zhu Y, 2014]	93.2	84.6
Pose Kinetic Energy [Shan J, 2014]	93.8	94.5
CNN (present study)	92.33	93
LNN-1 (present study)	96.67	95.33
LNN-2 (present study)	100	100

Table 8.1c. Summarized comparison MRSDaily database human activity classification.

Parameter	CNN	LAMSTAR-I	LAMSTAR-II
Training time (sec)	507.30*	378.63†	429.425†
Training accuracy (%)	94.33‡	98.67‡	100‡
Testing time (sec)	172.36§	151.23§	153.365§

*Training time of 7590 training samples for 50 epochs
†Training time of 7590 training samples for threshold 0.9999
‡Testing with the same input used as training set
§Testing time of 600 test samples on trained network
¶For a trained CNN for 50 epochs
ΔFor a trained LAMSTAR/LAMSTAR II with threshold 0.9999

COMPUTATION: The CNN network receives an 8 × 8 input vector of the coordinate image above. The CNN program used in this study is the DeepLearnToolbox [Rasmus, 2012] for CNN written in MATLAB. DeepLearnToolbox is a MATLAB/Octave toolbox for deep learning — see Part 1 of Appendix A.8.1. The code for LNN-1 is in Part 2 of the same appendix, while Preprocessing is as in Part 3 of that Appendix.

RESULTS: Table 8.1a compare performances of present study (bottom 3) for poses from the MRSDaily database , while Table 8.1b compare performances of present study (bottom 3) for poses from the CD-60 database. Table 8.1c compares Computing Time and Accuracy for the 3 networks of the Present Case Study. Observe the perfect recognition by LNN-2 in the results of Table 8.1c.

References

[Cornell. CAD60, 2009] http://pr.cs.cornell.edu/humanactivities/data.php. Copyright (c) Cornell University, (2009).

[Gaglio, 2014] Gaglio S, Lo Re G, Morana M, "Human activity recognition process using 3-D posture data", in *IEEE Transactions on Human-Machine Systems* (2014).

[Gupta R, 2013] Gupta R, Chia A Y S, Rajan D, "Human activities recognition using depth images", in *Proc. of the 21st ACM International Conference on Multimedia* (2013).

[Koppula 2013] Koppula H S, Gupta R, Saxena A, "Learning human activities and object affordances from RGB-D videos", arXiv:1210.1207v2, May 2013.

[Microsoft, MRSDaily]
http://research.microsoft.com/en-us/um/people/zliu/actionrecorsrc/

[Ni B, 2013] Ni B, Pei Y, Moulin P, Yan S, "Multilevel depth and image fusion for human activity detection", *IEEE Trans. Cybernetics* (2013).

[Oreifej, 2013] Oreifej O, Liu Z, "Hon4d: Histogram of oriented 4d normals for activity recognition from depth sequences", in *CVPR* (2013).

[Parisi, 2015] Parisi G I, Weber C, Wermter S, "Self-organizing neural integration of pose-motion features for human action recognition", in *Frontier in Neurobotics* (2015).

[Rasmus, 2012] https://github.com/rasmusbergpalm/DeepLearnToolbox. Copyright (c) 2012.

[Shan J, 2014] Shan J, Akella S, "3D Human action segmentation and recognition using pose kinetic energy", in *IEEE Workshop on Advanced Robotics and its Social Impacts* (ARSO), 2014.

[Sung J, 2011] Sung J, Ponce C, Selman B, Saxena A, "Human activity detection from RGBD images", in *Proc. AAAI Workshop on Pattern, Activity and Intent Recognition* (PAIR), 2011.

[Sung J, 2012] Sung J, Ponce C, Selman B, Saxena A, "Unstructured human activity detection from RGBD images," in *Proc. ICRA* (2012).

[Wang J, 2012] Wang J, Liu Z, Wu Y, Yuan J, "Mining action let ensemble for action recognition with depth cameras", in *Proc. CVPR* (2012), Providence, Rhode Island, June 16–21, 2012.

[Wang P, 2015] Wang P, Li W, Gao Z, Zhang J, Tang C, Ogunbona P, "Deep convolutional neural networks for action recognition using depth map sequences," arXiv, preprint arXiv:1501.04686, 2015.

[Yang X, 2012] Yang X, Zhang C, Tian Y, "Recognizing actions using depth motion maps-based histograms of oriented gradient", in *ACMMM* (2012).

[Yang X, 2013] Yang X, Tian Y, "Effective 3D action recognition using eigenjoints", *Journal of Visual Communication and Image Representation* (*JVCIR*), *Special Issue on Visual Understanding and Applications with RGBD Cameras* (2013).

[Yang X, 2014] Yang X, Tian Y, "Super normal vector for activity recognition using depth sequences," in *CVPR* (2014).

[Zafir, 2013] Zanfir M, Leordeanu M, Sminchisescu C, "The moving pose: An efficient 3d kinematics descriptor for low-latency action recognition and detection", in *ICCV* (2013).

[Zhang C, 2012] Zhang C, Tian Y, "RGB-D camera-based daily living activity recognition", *Journal of Computer Vision and Image Processing* Vol. 2, No. 4, December 2012.

[Zhu Y, 2014] Zhu Y, Chen W, Guo G, "Evaluating spatiotemporal interest point features for depth-based action recognition", in *Image and Vision Computing* (2014).

8.2. Medicine: Predicting Onset of Seizures in Epilepsy (J Tran)

The goal of this Case Study is to predict onset of epileptic seizures in Epilepsy patients from Intracranial EEG (iEEG) data by using 3 deep learning neural networks, Back Propagation (BP), LNN-1 and LNN-2 and comparing performance and computational speed of these 3 networks. Prediction is carried out by detecting the 20–30 minutes Pre-ictal (pre-seizure) phase as compared with the inter-ictal phase (of no seizures) that usually lasts for days or even for a few weeks, and which precedes the ictal phase of a few minutes. Obviously, computational speed is almost as critical as accurate prediction (oncoming seizures being missed).

DATASET: The data for this case study was downloaded from https://www.kaggle.com/c/seizure-prediction/data [epilepsy soc., 2014]. Both the inter-ictal (no seizures) and the pre-ictal sections used as data were of same duration (10 minutes). The pre-ictal data was taken between 15 and 5 minutes BEFORE start of seizures. Each data window is of 30 seconds duration. All inter-ictal data were randomly chosen at least a week before or after a seizure occurred.

PREPROCESSING: For both LNN-1 and LNN-2, the input data includes the dominant frequency of the in each 1 sec. time window.

COMPUTING: The convolutional program used was a Python Lasagne version of CNN. The data was entered into the CNN network using https://lasagne.readthedocs,org/, see Appendix A.8.2 below. The LAMSTAR LNN-1 and LNNN-2 programs were both based on the Core code in Chapter 6. Both LNN-1 and LNN-2 use five SOM input layers.

RESULTS: See Table 8.2 below. Results with CNN of 71% sensitivity were reported in [Mirowski, 2009], though for a different data source (University of Freiburg, Germany) and with different preprocessing.

Table 8.2. Comparison of results — Seizure prediction.

Method	Accuracy	Training Time	Testing Time
CNN	70%	170 sec	3 sec
LNN-1	81.25%	< 1 sec	< 1 sec
LNN-2	81.25%	< 1 sec	< 1 sec

References

[epilepsy soc., 2014] American Epilepsy Society Seizure Prediction Challenge https://www.kaggle.com/c/seizure-prediction/data, 2014.

[Mirowski, 2009] Mirowski P, Madhavan D, LeCun Y, Kuzniecky R, "Classification of patterns of EEG synchronization for seizure prediction" (https://epilepsy.uni-freiburg.de/freiburg-seizureprediction-project/eeg-database/), *Clinical Neurophysiology* **120**(11): 1927–1940 (2009).

8.3. Medicine: Image Processing: Cancer Detection (D Bose)

The goal is to build a classifier that can distinguish between cancer and control patients from the mass spectrometry data. The classifiers used were BP, CNN, LNN-1, and LNN-2 deep learning neural networks. The same data and the same preprocessing was used in all neural networks used for this Case Study.

DATASET: The data in this example is from the FDA-NCI Clinical Proteomics Program Databank [1]. Specifically, the data used was of High Resolution SELDI-TOF Study Sets from the databank's link -

Appendices to Case Studies of Chapter 8

Introduction

Appendix numbers correspond to their respective subsection number in Chapter 8. It is important to note that none of these appendices is the complete code for any of the Case Studies. Their purpose is that together with the respective description in Chapter 8, the appendix may help the reader to solve the problem described in the case study. Full programs are far too long to fit a book with 20 case studies. Furthermore, the codes involve proprietary library codes that are accessible by license.

The appendices were selected, subject to the constrains mentioned, such that they give the reader insight into how to integrate data of the kind that is needed in a given study into the code described in Chapter 8 for that study. Certain appendices aim to help integrating specific preprocessing methods of a Case Study. The references given in Chapter 8 should also be consulted when attempting to reconstruct a Case Study.

For any specific application of deep learning neural networks, the reader should attempt to use library codes and integrate them, whenever possible. Selection of preprocessing algorithms requires first to understand the problem to be solved. Again, most preprocessing methods should serve to either reduce data or to add knowledge that is not built into the neural network. Again, mathematical methods for analyzing a given problem are often available in open-access library programs, say codes of spectral and wavelets analysis, for entropy analysis, DNA coding, market analysis, etc.

A.8.1. Human Activity — Codes (A Bose)

Part 1: CNN

1. *Code_CNN.m*

```
clear all; close all; clc;
load ('activity_dataset.mat');

[tsR, tsC] = size(testdata);
[trR, trC] = size(traindata);

for i = 1 : tsR
  testdata(i,3:62) = normalizeData(testdata(i,3:62));
end
for i = 1 : trR
  traindata(i,3:62) = normalizeData(traindata(i,3:62));
end
testdata = double(reshape(testdata',8,8,tsR));
traindata = double(reshape(traindata',8,8,trR));
testlabel = double(testlabel');
trainlabel = double(trainlabel');

rand('state',0)
ttr = [];
for i = 1:50
  cnn = [];
  cnn.layers = {
    struct('type', 'i') %input layer
    struct('type', 'c', 'outputmaps', 24, 'kernelsize', 5) %convolution layer
    struct('type', 's', 'scale', 2) %sub sampling layer
  };
  opts.numepochs = i;
  opts.alpha = 0.85;
  opts.batchsize = 30;

  cnn = cnnsetup(cnn, traindata, trainlabel);
  [cnn, ttr(i)] = cnntrain(cnn, traindata, trainlabel, opts);
  disp(['Total training time:' num2str(ttr(i))]);

  tic;
  [er(i), correct(i), decision{i}] = cnntest(cnn, testdata, testlabel);
  tts = toc;
  disp('--------------------------------');
end

save ttr ttr;
save er er;
save correct correct;
```



```
save decision decision;
generateConfusionMatrix(decision{50}');
figure; plot(er(1:50), 'LineWidth', 2);
xlabel('Number of epoch');
ylabel('Bit error (%)');

figure; plot(ttr(1:50), 'LineWidth', 2);
xlabel('Number of epoch');
ylabel('Training time (sec)');

figure; plot(correct(1:50)/600*100, 'LineWidth', 2);
('Number of epoch');
ylabel('Recognition rate (%)');
```

Part 2: LAMSTAR (LNN-1)

1. *Code_LAMSTAR.m*

```
clear all; close all; clc;
load ('activity_dataset.mat');

for i = 1 : size(traindata,1)
  traindata(i,3:62) = normalizeData(traindata(i,3:62));
end
disp('Training data acquisition done...');
X_train = traindata';
[row, col] = size(X_train);
numSubWords = 16;

nBit = 8;
alpha = 0.8;
tol = 1e-5;
thresh = 0.9999;

flag = zeros(1,numSubWords);
disp('Forming Sub Words');
for i = 1:size(X_train,2)
 tempX = reshape(X_train(:,i), nBit, nBit);
 for j = 1:numSubWords
  if j <= nBit
   X_in{i}(j,:) = tempX(j,:);
  else
   X_in{i}(j,:) = tempX(:,j - nBit)';
  end
 end

 check(1,:) = zeros(1, nBit);
 for k = 1:numSubWords
  for t = 1 : nBit
```

```
    if (X_in{i}(k,t) ~= check(1,t))
      X_norm{i}(k,:) = X_in{i}(k,:) / sqrt(sum(X_in{i}(k,:).^2));
    else
      X_norm{i}(k,:) = zeros(1,nBit);

    end
   end
  end
end

tic;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
disp('Dynamic Building of neurons');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Building of the first neuron is done as Kohonen Layer neuron
%(this is for all the subwords in the first input pattern for all SOM modules
i = 1;
ct = 1;
while (i <= numSubWords)
 cl = 0;
 for t = 1 : nBit
   if (X_norm{ct}(i,t)==0)
     cl = cl+1;
   end
 end
 if (cl == nBit)
  Z{ct}(i) = 0;
 elseif (flag(i) == 0)
  W{i}(:,ct) = rand(nBit, 1);
  flag(i) = ct;
  W_norm{i}(:,ct) = W{i}(:,ct)/sqrt(sum(W{i}(:,ct).^2));
  Z{ct}(i)= X_norm{ct}(i,:)*W_norm{i};
  while(Z{ct}(i) <= (1-tol)),
    W_norm{i}(:,ct) = W_norm{i}(:,ct) + alpha*(X_norm{ct}(i,:)' - W_norm{i}(:,ct));
    Z{ct}(i) = X_norm{ct}(i,:)*W_norm{i}(:,ct);
  end
 end
 r(ct,i) = 1;
 i = i+1;
end

r(ct,:) = 1;
ct = ct+1;
while (ct <= size(X_train,2))
 for i = 1 : numSubWords
   cl = 0;
   for t = 1 : nBit
     if (X_norm{ct}(i,t) == 0)
```



```
     cl = cl+1;
   end
  end
  if (cl == nBit)
   Z{ct}(i) = 0;
  else
   r(ct,i) = flag(i);
   r_new=0;
   for k = 1:max(r(ct,i)),
     Z{ct}(i) = X_norm{ct}(i,:)*W_norm{i}(:,k);
     if Z{ct}(i) >= thresh
       r_new = k;
       flag(i) = r_new;
       r(ct,i) = flag(i);
       break;
     end
   end
   if (r_new == 0)
     flag(i) = flag(i) + 1;
     r(ct,i) = flag(i);
     W{i}(:,r(ct,i)) = rand(nBit,1);
     %flag(i) = r
     W_norm{i}(:,r(ct,i)) = W{i}(:,r(ct,i))/sqrt(sum(W{i}(:,r(ct,i)).^2));
     Z{ct}(i) = X_norm{ct}(i,:)*W_norm{i}(:,r(ct,i));

     while(Z{ct}(i) <= (1-tol)),
       W_norm{i}(:,r(ct,i)) = W_norm{i}(:,r(ct,i)) + alpha*(X_norm{ct}(i,:)' -
       W_norm{i}(:,r(ct,i)));
       Z{ct}(i) = X_norm{ct}(i,:)*W_norm{i}(:,r(ct,i));
     end
    end
  end
  ct = ct+1;
end

%%%%%%%%%%%%%%%
% Link Weights
%%%%%%%%%%%%%%%
outNum = size(trainlabel,2);
ct = 1;
m_r = max(r);
for i = 1:numSubWords,
  L_w{i} = zeros(m_r(i),outNum);
end

ct = 1;
disp('Link weights and output calculations');
```

```
Z_out = zeros(size(X_train,2), outNum);
while (ct <= size(X_train,2))
 L = zeros(size(X_train,2), outNum);
 for i = 1 : numSubWords
  count = size(find(r(:,i) == r(ct,i)) , 1);
   if (r(ct,i)~=0)
    for j = 1 : outNum
      if (trainlabel(ct,j)==0)
%         L_w{i}(r(ct,i),j) = L_w{i}(r(ct,i),j)- 5;
        L_w{i}(r(ct,i),j) = L_w{i}(r(ct,i),j)/count - 5;
      else
%         L_w{i}(r(ct,i),j) = L_w{i}(r(ct,i),j) + 5;
        L_w{i}(r(ct,i),j) = L_w{i}(r(ct,i),j)/count + 5;
      end
    end
%     L(i,:) = L_w{i}(r(ct,i),:);
    L(i,:) = L_w{i}(r(ct,i),:)/count;
   end
 end
 Z_out(ct,:) = sum(L);
 ct = ct+1;
end
toc;
save W_norm W_norm
save L_w L_w
LAMSTAR_test
```

2. *generateConfusionMatrix.m*

```
function generateConfusionMatrix(predicted)
classes = [1 0 0 0 0 0;
     0 1 0 0 0 0;
     0 0 1 0 0 0;
     0 0 0 1 0 0;
     0 0 0 0 1 0;
     0 0 0 0 0 1];
fprintf('+-------------+-------------------------------------------------------------------------+-----------+\n');
fprintf('|      |Predicted Class                                                          |\n');
fprintf('+-----------+----------+----------+----------+----------+----------+----------+---------+\n');
fprintf('|Actual Class |Class1   |Class2   |Class3   |Class4   |Class5   |Class6   |Other   |\n');
fprintf('+-----------+----------+----------+----------+----------+----------+----------+----------+\n');

for i = 1 : 6
  class1 = 0;
  class2 = 0;
  class3 = 0;
  class4 = 0;
  class5 = 0;
  class6 = 0;
```



```
  other = 0;
  for j = 1 : 100
   if (predicted((i-1)*100+j, :) == classes(1,:))
     class1 = class1 + 1;
   elseif (predicted((i-1)*100+j, :) == classes(2,:))
     class2 = class2 + 1;
   elseif (predicted((i-1)*100+j, :) == classes(3,:))
     class3 = class3 + 1;
   elseif (predicted((i-1)*100+j, :) == classes(4,:))
     class4 = class4 + 1;
   elseif (predicted((i-1)*100+j, :) == classes(5,:))
     class5 = class5 + 1;
   elseif (predicted((i-1)*100+j, :) == classes(6,:))
     class6 = class6 + 1;
   else
     other = other + 1;
   end
  end
  fprintf('|Class%d\t\t |%d \t\t |%d \t\t |%d \t\t |%d \t\t |%d \t\t |%d \t\t |%d \t\t |\n', i, class1,
class2, class3, class4, class5, class6, other);
  fprintf('+----------+----------+----------+----------+----------+----------+----------+----------+\n');
end
end
```

3. *LAMSTAR_test.m*

```
%% LAMSTAR_test.m
clear all;
load W_norm
load L_w
load ('activity_dataset.mat');
nBit = 8;

for i = 1 : size(testdata,1)
  testdata(i,3:62) = normalizeData(testdata(i,3:62));
end
X_test = testdata';
[row, col] = size(X_test);
numSubWords = 16;
% To make 12 subwords
correct = 0;
wrong = 0;
errPer = 0;
for i = 1:size(X_test,2)
  tempX = reshape(X_test(:,i), nBit, nBit);
  for j = 1 : numSubWords
    if j <= nBit
      X_in{i}(j,:) = tempX(j,:);
    else
```

```
    X_in{i}(j,:) = tempX(:,j - nBit)';
  end
end

check(1,:) = zeros(1, nBit);
for k = 1 : numSubWords
 for t = 1 : nBit
   if (X_in{i}(k,t) ~= check(1,t))
    X_norm{i}(k,:) = X_in{i}(k,:) / sqrt(sum(X_in{i}(k,:).^2));
   else
    X_norm{i}(k,:) = zeros(1, nBit);
   end
 end
end

for k = 1 : numSubWords - 1
 if isempty(W_norm{k}),
   Z_out(k,:) = [0 0 0 0 0 0];
 else
   Z = X_norm{i}(k,:)*W_norm{k};
   index(k) = find((Z == max(Z)),1);
   L(k,:) = L_w{k}(index(k),:);
   Z_out(k,:) = L(k,:)*Z(index(k));
 end
end
 end
 final_Z(i,:) = sum(Z_out);
 sgm = sigmoid(final_Z(i,:));
 decision(i,:) = sgm >= max(sgm);
 err = xor(decision(i,:), testlabel(i,:));
 errPer = errPer + sum(err)/size(err,2);
 if (decision(i,:) == testlabel(i,:))
   out = 'Correct';
   correct = correct + 1;
 else
   out = 'Wrong';
   wrong = wrong + 1;
 end
 disp(['Test Pattern: ' num2str(i) ' |output: ' num2str(decision(i,:)) ' :' out]);
 if rem(i,100) == 0
   disp('-------------------------------------------------------');
 end
end
disp(['Correct: ' num2str(correct)]);
disp(['Wrong: ' num2str(wrong)]);
disp(['Bit Error (%): ' num2str(errPer/size(X_test,2)*100) '%']);
generateConfusionMatrix(decision);
```


4. *normalizeData.m*

```
% Code for normalizing MSR Daily Activity 3D Dataset & Cornell CAD-60
% Dataset
function [normalized_data] = normalizeData(Coordinates)
```

A.8.2. Predicting Seizures in Epilepsy (J Tran)

CNN: The code below uses Python's https://lasagne.readthedocs.org/

```
def cnn_preprocess(input, detect, predict):

 length = int(len(input)/23.6)
 dimension = 224
 padding = int((dimension - length)/2)
 result = []
 scaling = 0
 if detect is True:
  scaling = 1000
 if predict is True:
  scaling = 300

 while len(input) >= length:
  empty_array = create_empty_array(dimension)

 prev = -1
 for index in range(0, length):

  zero_axis = int(dimension/2)
  scale = int(scaling/zero_axis)

  if input[index] >= 0:
   row = index+padding
   col = int(input[index]/scale + zero_axis)
   if col >= dimension:
    col = dimension - 1
   if index != 0:
    if col < prev:
     for i in range(col + 1, prev):
      empty_array[row][i] = i
     prev = col
    if col > prev:
     for i in range(col - 1, prev, -1):
      empty_array[row][i] = i
     prev = col
   else:
    prev = col
   empty_array[row][col] = col
  elif input[index] < 0:
   row = index+padding
   scaled = int(input[index]/scale)
```