Guide to reproduce the experiments

This document describes the steps to reproduce the experiments carried out during the investigation.

First, the description of the experiments with the C-Town benchmark and later with the E-Town benchmark are presented.

Experiments for C-Town:

Detection

1. It is necessary to use the BATADAL dataset available at https://www.batadal.net/data.html

Follow *Cross_validation* script

2. In C-town/Main (detection) subdirectory is located the script *Cross_validation.m*. The cross-validation process is carried out to choose the best hyper-parameters for training the Autoencoder. In addition, the AEWMA parameters are adjusted in order to obtain a low rate of false alarms.
    a) Open *Cross_validation.m* script. Initially, the complete dataset is uploaded, the dataset1 is assigned to vector X and the parameters are established

```
clear
clc

load('batadal.mat')

X = dataset1;
```

b) The function *Remove_var_cat* receives dataset1 and the outputs are X1 which represents the data from continuous variables and D1 which represents the data from discrete variables. Following, only the vector X1 is used.

```
[X1, D1] = Remove_var_cat(X);       % X1: continuous variables of dataset 1
                                    % D1: discrete variables of dataset
```

c) The outliers of the dataset X1 are eliminated by using the function *Remove_Outliers00*.

```
[X1] = Remove_Outliers00(X1);            % Remove Outliers of dataset1
```

d) The dataset X1 is normalized with mean value zero (0), and standard deviation one (1).

```
[X1_,m,sigma] = zscore(X1); % Normalize data with mean value=0 and
standard deviation=1
```

For the cross-validation process, it will be necessary to separate the data into several sets. In this case, the proportion 0.2 is selected (prop=0.2 for ktot=5), which means that it is divided into 5 subsets where one of them will be used for validation and the rest for training. The

GetData_CV function receives the normalized data set, the number of subsets, and the ratio. It returns the training and validation subsets (XTrain and XVal respectively).

In the script, the dimension selection of the latent space from 21 to 25 neurons through cross-validation is completed.

```
%------------------------Parameters--------------------------
[totalData,~] = size(X);
folds = 5;
prop = 0.2;
ktot = 5;

dim = 25;
since = 21;

atkAEs_det = cell(folds,dim);
MSEs = zeros(folds,dim);

[XTrain,XVal] = GetData_CV(X1_,ktot,prop);  % Data set for Cross Validation
```

The MATLAB function *trainAutoencoder* is used for training. The MATLAB function *predict* is used to obtain the possible output of the Autoencoder. The best hyperparameters are selected.

```
for ii = since:dim
    for j = 1:folds

    X1 = XTrain(:,:,j);
    [samples,var] = size(X1);

    % ------------------- TRAIN -----------------------
    Xtrain = X1';
    Xval = XVal(:,:,j)';


        % Train
        AE = trainAutoencoder(Xtrain, ii, 'MaxEpochs', 3000,...
            'EncoderTransferFunction','logsig',...
            'DecoderTransferFunction','purelin',...
            'ShowProgressWindow',true,...
            'L2WeightRegularization', 0.001,...
            'SparsityRegularization',1,...
            'SparsityProportion',0.99);

        % Validation
        Xval_ = predict(AE,Xval);
        atkAEs_det{j,ii} = AE;
        MSE = mse(Xval-Xval_);
        MSEs(j,ii) = MSE;
```

```
        disp("Dim : "+ii+" Fold : "+j+" MSE: "+MSE+"");

    end
end

% Selection of best hyperparameters

mse_average = sum(MSEs);
fold = ones(size(mse_average))*folds;
mse_average = mse_average./fold
```

Follow the *Autoencoders_Train* script

3. Autoencoder training for cyber-attack detection. The *Autoencoders_Train.m* script located in C-town/Main(detection) subdirectory is used for such purpose. Steps 2a, 2b, and 2c are repeated.

```
clear
clc

load('batadal.mat')

[X1, D1] = Remove_var_cat(dataset1);        % Xn: continuous variables of
datasetn
                                              % Dn: discrete variables of

datasetn
[X1] = Remove_Outliers00(X1);               % Remove Outliers of dataset1
[samples,var] = size(X1);

[X1_,m,sigma] = zscore(X1);                  % Normalice normal data
```

d. Unlike the cross-validation process, the entire training dataset is used here. The Autoencoder is trained with the MATLAB *trainAutoencoder* function.

```
%% -------------------- TRAIN -----------------------
Xtrain = X1_';
Xval = X1_';

dim = 24;

   %% Train Autoencoder
    AE_det = trainAutoencoder(Xtrain, dim, 'MaxEpochs', 3000,...
        'EncoderTransferFunction','logsig',...
        'DecoderTransferFunction','purelin',...
        'ShowProgressWindow',true,...
        'L2WeightRegularization', 0.001,...
        'SparsityRegularization',1,...
        'SparsityProportion',0.99);
```

e. The Xval_ variable represents the prediction of the Autoencoder when ingesting the data after being trained.

```
    Xval_ = predict(AE_det,Xval);                    % Output prediction
```

f. The SPE is calculated with the original data and the prediction. This gives information about the error obtained when data from normal system operation is analyzed.

```
    SPE = diag(sqrt((Xval - Xval_)' * (Xval - Xval_)));      % Prediction
error (SPE)
    MSE = mse(Xval - Xval_)
```

g. Save in the *atksAEs_det.mat* variable the trained Autoencoder and the SPE for normal system operation.

```
    %% Save
    save atkAEs_det.mat AE_det SPE
```

Follow *Autoencoders_Test* script

4. To check the effectiveness of the above process, the *Autoencoders_Test.m* script located in in C-town/Main(detection) subdirectory is used.
    a. The three data sets are loaded, and steps 2.a, 2.b and 2.c are repeated in each of the sets, obtaining X1, X2, and X3 from dataset1, dataset2 and dataset3, respectively.
        - The test data set is manually normalized by using the mean and standard deviation of the training data set.

```
clear
clc

load('batadal.mat')
load('atkAEs_det.mat')
load('Batadal_result.mat')

% Xn: continuous variables of datasetn
% Dn: discrete variables of datasetn
[X1, D1] = Remove_var_cat(dataset1);
[X2, D2] = Remove_var_cat(dataset2);
[X3, D3] = Remove_var_cat(dataset3);

[X1] = Remove_Outliers00(X1);              % Remove Outliers of dataset1

[samples,~] = size(X3);

[X1_,m,sigma] = zscore(X1);                 % Normalice normal data
[~,var] = size(X3);

for i = 1:samples                           % Normalice attack data
    aux = X3(i,:) - m;
    for j = 1:var
        Xtest(i,j) = aux(j)/sigma(j);
    end

end
```

b. In *Xtest* is stored the data from dataset3. In *Ytest* is stored the actual outputs that identify as “1” an attack and “0” a non-attack (this is used for testing). P_SPE represents the new SPE obtained by comparing the original data (*Xtest*) with the prediction given by the Autoencoder (*P_Xtest*).

```
%% TEST

P_Xtest = predict(AE_det,Xtest);                                         %
Output prediction
P_SPE = diag(sqrt((Xtest - P_Xtest)' * (Xtest - P_Xtest)));              %
Prediction error (SPE)
```

c. The *AEWMA* function located in C-town/Main(detection) subdirectory is used to classify observations into attacks or non-attacks. AEWMA receives as parameters *P_SPE, SPE, minimum delta, maximum delta*, and *arl*. The function returns the *atks_det* array with the classification of each of the samples.

```
arl = 400;
d2 = 5;     % maximum delta
d = 0.25;   % minimun delta

[atks_det,~] = AEWMA(P_SPE,SPE,d,d2,arl);  % Classify output in attack (1)
or not attack (0)
atks_det = atks_det';
```

d. The indices *TPR, TNR, Sclf, FAR, FDR* and precision are calculated.

```
%% Classification check

Result = zeros(7,1);

    tot_A = 0;
    tot_NA = 0;
    TP = 0;                     % System under attack and attack detected
    TN = 0;                     % System without attack and without attack
detection
    FP = 0;                     % System under attack and without attack
detection
    FN = 0;                     % System without attack and attack
detected

    for ii = 1:samples
            if((Ytest(ii,1) == 1) && (atks_det(ii,1) == 1))
                TP = TP + 1;
            elseif ((Ytest(ii,1) == 0) && (atks_det(ii,1) == 0))
                TN = TN + 1;
            elseif ((Ytest(ii,1) == 0) && (atks_det(ii,1) == 1))
                FP = FP + 1;
            elseif ((Ytest(ii,1) == 1) && (atks_det(ii,1) == 0))
                FN = FN + 1;
```

```
            end
    end

    TPR = TP/(TP + FN);                                         %TPR = TP/(TP +
FN) True Positive Rate
    TNR = TN/(FP + TN);                                         %TNR = TN/(FP +
TN) True Negative Rate
    Sclf = (TPR + TNR)/2;
    FAR = (FP/(FP+TN))*100;
    FDR = TPR*100;
    TP;
    FN;
    recall = TP/(TP+FN)
    precision = TP/(TP+FP)
    F1 = 2*(recall*precision)/(recall+precision)


ms(1,1) = FDR;
ms(1,2) = FAR;
ms(1,3) = arl;
ms(1,4) = d;
ms(1,5) = d2;

X = ['ARL :',num2str(arl)];
disp(X)
disp('')
```

e. To check the rapid detection, the STTD index is found

```
%% Checking fast detection

gamma = 0.5;
na = 0;
td = 0;
t0 = 0;
TTD = zeros(1,1);
flag = 0;
delta_t = zeros(1,1);

for ii = 1:samples

    if(ii == 1)

        if(Ytest(ii,1) == 1)
            t0 = ii;
            na = na + 1;
        end
        if(atks_det(ii,1) == 1)
            td = ii;
            flag = 1;
        end

    elseif(Ytest(ii,1)==1 && Ytest(ii-1,1)==0)
```

```
        t0 = ii;
        na = na + 1;
        if(atks_det(ii,1)== 1)
            td = ii;
            flag = 1;
        end
    elseif(Ytest(ii,1)==1 && Ytest(ii-1,1)==1)
        if(atks_det(ii,1)==1 && flag == 0)
            td = ii;
            flag = 1;
        end
    elseif(Ytest(ii,1)==0 && Ytest(ii-1,1)==1)
        flag = 0;
        TTD(na,1) = td - t0;
        delta_t(na,1) = ii - t0;
    end
end

sum_aux = 0;
for ii = 1:na

    aux = TTD(ii,1)/delta_t(ii,1);
    sum_aux = sum_aux + aux;

end
```

f. With both indicators, a global indicator *S* is formulated.

```
STTD = 1 - ( (1/na)*sum_aux )
Sclf
S = gamma * STTD + (1-gamma)*Sclf
```

g. The results are shown.

```
%% To point out the False alarms in the figures
[atks_det1,signal_e] = AEWMA(P_SPE,SPE,d,d2,arl);

value_fa = 0;
times_fa = 0;
cont = 1;
for ii = 1:samples
    if(atks_det(ii,1)>Ytest(ii,1))
        times_fa(cont) = ii;
        value_fa(cont) = signal_e(1,ii);
        cont = cont +1;
    end

end


figure
```

```
plot(P_SPE);grid on; hold on; plot(Ytest*3);
legend('SPE de Xtest', 'Attacks');

figure
plot(atks_det1,'linewidth',1.5);grid on; hold on; plot(Ytest);
legend('SPE', 'Attack interval');
title('Detection')

P_SPE1 = P_SPE;

figure
a = area(atks_det1);
a.FaceColor = 'black';
a.FaceAlpha = 0.2;
a.LineStyle = 'none';
e = {'no-attack','attack'};
set(gca,'YTick',0:1:length(e)+1);
set(gca,'YTickLabel',e);
set(gca,'XTick',0:100:samples+1);
hold on;
plot(Ytest,'Color', 'black');
legend('detected attacks','real attacks')

figure
umbral = ones(samples,1)*1.0385;
falsa_alarma = atks_det1' - Ytest;
p = plot(signal_e);
p.Color = [0 0.4470 0.7410];
p.LineWidth = 1.5;
hold on;
plot(Ytest*10,'Color','black');
hold on;
p = plot(umbral,'Color','red');
p.LineWidth = 1.5;
hold on;
p = plot(times_fa,value_fa,'o');
p.Color = [0.4660 0.6740 0.1880];
ylabel('Attack Indicator');
xlabel('Observations');
legend('z(t)','time interval of attack','limit of control chart','false
alarm')


%% Save
save atks_det1.mat atks_det1
save P_SPE1.mat P_SPE1
```

Location

5. The localization experiment consists of training an Autoencoder for each ARRs. In this case, the first step is to find the variables to consider in each ARR using Structural Analysis.

a. In the C-town\Structural Analysis\src subdirectory is located the ctown.m script. In it are declared the model variables and the possible types of attacks. (model.x, model.z and model.f). (Please, read the Structural Analysis software User Manual)
b. Declare the equations that relate the variables and attacks in model.rels (Please, see the Structural Analysis Software User Manual).
c. Check the figures obtained. Obtain the ARRs.

6. In C-town\Main (localization) subdirectory is located the information related to each ARR. In each subdirectory, the reader will find all scripts and functions used for that ARR. For each ARR, it is necessary to develop steps 2 and 3 explained in detection.
7. To verify the location results the *main.m script* located in Ctown/Mail(localization) subdirectory is used.

Follow *main.m* script

a. Upload information correspondig to each ARR.

```
% MAIN Localization AE

clear
clc

load('batadal.mat')
addpath('Group 1 (data normal)')
addpath('Group 2 (data normal)')
addpath('Group 3 (data normal)')
addpath('Group 4 (data normal)')
addpath('Group 5 (data normal)')
addpath('Group 6 (data normal)')
addpath('Group 7 (data normal)')

% addpath('auxiliar')

load('atks_det1.mat')
load('atks_det2.mat')
load('atks_det3.mat')
load('atks_det4.mat')
load('atks_det5.mat')
load('atks_det6.mat')
load('atks_det7.mat')

%Initialization the variables samples, groups, atks and Y test
[samples,~] = size(atks_det1);
groups = 7;
atks = 6;
Ytest = labels3;
```

b. The *alabels* array represents the actual classification for each attack. To obtain it, the *Atk_selection* function is used. The objective is to create a labeled output (Ytest) for each DMA.

```
%% Labels of real attacks

alabels = zeros(samples,atks);
alabels(:,1) = Atk_selection(Ytest,[3 4]);      % fdma1
alabels(:,2) = Atk_selection(Ytest,[2 5]);      % fdma6
alabels(:,3) = Atk_selection(Ytest,[7]);        % fdma2
alabels(:,4) = Atk_selection(Ytest,[1]);        % fdma3
% alabels(:,5) = Atk_selection(Y_test,[ ]);     % fdma4 do not receive any
attack
alabels(:,6) = Atk_selection(Ytest,[6]);        % fdma5
```

c. The *adiagnostic* array represents the classification obtained by using the methodology proposed in the paper. It is obtained from structural analysis.

```
%% Labels of attck diagnosis

adiagnostic = zeros(samples,atks);

adiagnostic(:,1) = atks_det6.*atks_det7;            % fdma1
adiagnostic(:,2) = atks_det5.*atks_det7;            % fdma6
adiagnostic(:,3) = atks_det4.*atks_det5;            % fdma2
adiagnostic(:,4) = atks_det3.*atks_det5;            % fdma3
adiagnostic(:,5) = atks_det2.*atks_det6;            % fdma4
adiagnostic(:,6) = atks_det1.*atks_det6;            % fdma5
```

d. Verify if the attack is detected in the correct DMA. For this, the *inZone* variable is used.

```
%% Attack detected in correct zone? Yes -->inZone = 1    NO -->inZone = 0
inZone = zeros(samples,1);

magDiag = sum(adiagnostic,2);

for ii = 1:samples
    for j = 1:atks

        if(alabels(ii,j)==1 && adiagnostic(ii,j)==1)
            inZone(ii) = 1;
        end

    end
end

clear ii j
```

e. Find attack location performance metrics.

```
%% Statistics of each attack
num_atk = 0;
num_atks = zeros(samples,1);
```

```matlab
times = zeros(atks,2);
t = 1;
c_atks = 0;          % Times counter under attack
c_correct = 0;       % Times counter under attack that are located in the
correct zone
c_moreOne = 0;       % Times counter under attack that are located in more
one zone
c_ok = 0;            % Times counter under attack that are located only in
the correct zone
c_atkDet = 0;        % Times counter under attack detected

for ii = 1:samples

    if(ii == 1)
        if(Ytest(ii) == 1)
            num_atk = num_atk + 1;
            num_atks(ii) = num_atk;
        end
    end
    if(ii~=1)
        if(Ytest(ii)==1 && Ytest(ii-1) == 0)
            num_atk = num_atk + 1;
            num_atks(ii,1) = num_atk;
            times(t,1) = ii;

            % Attack start
            c_atks = c_atks + 1;
            if(inZone(ii) == 1)
                c_correct = c_correct + 1;
            end
            if(magDiag(ii)>1)
                c_moreOne = c_moreOne + 1;
            end
            if(inZone(ii) == 1 && magDiag(ii)==1)
                c_ok = c_ok + 1;
            end
            if(magDiag(ii)>0)
                c_atkDet = c_atkDet + 1;
            end
        elseif(Ytest(ii,1)==1 && Ytest(ii-1)==1)
            num_atks(ii,1) = num_atk;

            % During an attack
            c_atks = c_atks + 1;
            if(inZone(ii) == 1)
                c_correct = c_correct + 1;
            end
            if(magDiag(ii)>1)
                c_moreOne = c_moreOne + 1;
            end
            if(inZone(ii) == 1 && magDiag(ii)==1)
                c_ok = c_ok + 1;
            end
            if(magDiag(ii)>0)
                c_atkDet = c_atkDet + 1;
            end
```

```
        elseif(Ytest(ii)== 0 && Ytest(ii-1)==1)
            times(t,2) = ii;


            % Attack end
            disp(" ");
            disp("ATTACK #: "+t+"");
            aux = c_atkDet*100/c_atks;
            disp("     Localized attacks: "+aux+"%   ("+c_atkDet+")");
            aux = c_correct*100/c_atkDet;
            disp("     Attacks located in the correct area: "+aux+"%
("+c_correct+")");
            aux = c_moreOne*100/c_atkDet;
            disp("     Attacks located in more than one area: "+aux+"%
("+c_moreOne+")");
            aux = c_ok*100/c_atkDet;
            disp("Attacks located only in the correct area: "+aux+"%
("+c_ok+")");
            c_atks = 0;
            c_correct = 0;
            c_moreOne = 0;
            c_ok = 0;
            c_atkDet = 0;
            t = t + 1;

        end
    end

end

clear num_atk t
```

f. The results are shown.

```
%% Show results

figure
plot(magDiag); grid on; hold on;
xlabel('Time')
ylabel('Number of areas where an attack was detected')


figure
plot(adiagnostic(298:367,1),'o'); grid on; hold on;
plot(adiagnostic(298:367,2)*2,'o'); grid on; hold on;
plot(adiagnostic(298:367,3)*3,'o'); grid on; hold on;
plot(adiagnostic(298:367,4)*4,'o'); grid on; hold on;
plot(adiagnostic(298:367,5)*5,'o'); grid on; hold on;
plot(adiagnostic(298:367,6)*6,'o'); grid on; hold on;
title('Diagnosis during Atk1');
e = {' ','dma1','dma6','dma2','dma3','dma4','dma5'};
set(gca,'YTick',0:1:length(e)+1);
set(gca,'YTickLabel',e);
```

```
figure
plot(adiagnostic(633:697,1),'o'); grid on; hold on;
plot(adiagnostic(633:697,2)*2,'o'); grid on; hold on;
plot(adiagnostic(633:697,3)*3,'o'); grid on; hold on;
plot(adiagnostic(633:697,4)*4,'o'); grid on; hold on;
plot(adiagnostic(633:697,5)*5,'o'); grid on; hold on;
plot(adiagnostic(633:697,6)*6,'o'); grid on; hold on;
title('Diagnosis during Atk2');
e = {' ','dma1','dma6','dma2','dma3','dma4','dma5'};
set(gca,'YTick',0:1:length(e)+1);
set(gca,'YTickLabel',e);

figure
plot(adiagnostic(868:898,1),'o'); grid on; hold on;
plot(adiagnostic(868:898,2)*2,'o'); grid on; hold on;
plot(adiagnostic(868:898,3)*3,'o'); grid on; hold on;
plot(adiagnostic(868:898,4)*4,'o'); grid on; hold on;
plot(adiagnostic(868:898,5)*5,'o'); grid on; hold on;
plot(adiagnostic(868:898,6)*6,'o'); grid on; hold on;
title('Diagnosis during Atk3');
e = {' ','dma1','dma6','dma2','dma3','dma4','dma5'};
set(gca,'YTick',0:1:length(e)+1);
set(gca,'YTickLabel',e);

figure
plot(adiagnostic(938:968,1),'o'); grid on; hold on;
plot(adiagnostic(938:968,2)*2,'o'); grid on; hold on;
plot(adiagnostic(938:968,3)*3,'o'); grid on; hold on;
plot(adiagnostic(938:968,4)*4,'o'); grid on; hold on;
plot(adiagnostic(938:968,5)*5,'o'); grid on; hold on;
plot(adiagnostic(938:968,6)*6,'o'); grid on; hold on;
title('Diagnosis during Atk4');
e = {' ','dma1','dma6','dma2','dma3','dma4','dma5'};
set(gca,'YTick',0:1:length(e)+1);
set(gca,'YTickLabel',e);

figure
plot(adiagnostic(1230:1329,1),'o'); grid on; hold on;
plot(adiagnostic(1230:1329,2)*2,'o'); grid on; hold on;
plot(adiagnostic(1230:1329,3)*3,'o'); grid on; hold on;
plot(adiagnostic(1230:1329,4)*4,'o'); grid on; hold on;
plot(adiagnostic(1230:1329,5)*5,'o'); grid on; hold on;
plot(adiagnostic(1230:1329,6)*6,'o'); grid on; hold on;
title('Diagnosis during Atk5');
e = {' ','dma1','dma6','dma2','dma3','dma4','dma5'};
set(gca,'YTick',0:1:length(e)+1);
set(gca,'YTickLabel',e);

figure
plot(adiagnostic(1575:1654,1),'o'); grid on; hold on;
plot(adiagnostic(1575:1654,2)*2,'o'); grid on; hold on;
plot(adiagnostic(1575:1654,3)*3,'o'); grid on; hold on;
plot(adiagnostic(1575:1654,4)*4,'o'); grid on; hold on;
plot(adiagnostic(1575:1654,5)*5,'o'); grid on; hold on;
plot(adiagnostic(1575:1654,6)*6,'o'); grid on; hold on;
title('Diagnosis during Atk6');
```

```
e = {' ','dma1','dma6','dma2','dma3','dma4','dma5'};
set(gca,'YTick',0:1:length(e)+1);
set(gca,'YTickLabel',e);

figure
plot(adiagnostic(1941:1970,1),'o'); grid on; hold on;
plot(adiagnostic(1941:1970,2)*2,'o'); grid on; hold on;
plot(adiagnostic(1941:1970,3)*3,'o'); grid on; hold on;
plot(adiagnostic(1941:1970,4)*4,'o'); grid on; hold on;
plot(adiagnostic(1941:1970,5)*5,'o'); grid on; hold on;
plot(adiagnostic(1941:1970,6)*6,'o'); grid on; hold on;
title('Diagnosis during Atk7');
e = {' ','dma1','dma6','dma2','dma3','dma4','dma5'};
set(gca,'YTick',0:1:length(e)+1);
set(gca,'YTickLabel',e);
```

Experiments for E-Town:

Generate data sets

1. It is necessary to install Epanet2 software package. Open ETown_wPLCs.inp to perform expert analysis. The aim is to divide the network to apply the methodology. Resulting in the division of the network into three zones.
2. The datasets used for this experiment were generated with the epanetCPA-master tool downloaded from https://github.com/rtaormina/epanetCPA with the etown scenario by using the *main.m* script.

Structural Analysis

3. It is necessary to develop a structural analysis for each zone. The procedure is similar to C-Town localization experiment. An Autoencoder must be trained for each ARR. The first step is to find the variables to consider in each ARR. Develop the procedure explained in step 5 of c-town experiments.
   a. In the E-town/Structural Analysis/src folder use the *etown2_1.m, etown2_2.m, and etown2_3.m* scripts to obtain the variables of the ARRs groups for zone 1, zone 2, and zone 3 respectively.
4. In E-town/Main subdirectory are located the *Zone 1.m, Zone2.m, and Zone3.m* scripts, used for training the Autoencoders of the ARRs in the three zones.

Follow *Zone1.m* script

a. Load the data SCADA_readings.mat which contains the training and test data.

```
load('SCADA_readings.mat'); % load raw datasets
```

b. Create the labeled outputs to check the results that are obtained.

```
% labels for the dataset with the first 10 attacks
evtr{1} = (600:630);
evtr{2} = (1000:1040);
evtr{3} = (1400:1500);
evtr{4} = (2000:2100);
evtr{5} = (2600:2700);
evtr{6} = (2600:2700);
evtr{7} = (3300:3400);
evtr{8} = (3350:3450);
evtr{9} = (3400:3500);
evtr{10}= (4000:4050);

Y_val = zeros(height(SCADA_val),1);
Y_val([evtr{1} evtr{2} evtr{3} evtr{4} evtr{5} evtr{6} evtr{7} evtr{8}
evtr{9} evtr{10}]) = 1; %events flags for train
Y_val=Y_val(1:end); %to have full days in the dataset

% labels for each DMA
Y_vl = zeros(height(SCADA_val),24);
Y_vl([evtr{3} evtr{10}],6) = 1; %atk 6
Y_vl([evtr{2}],11) = 1; %atk 11
Y_vl([evtr{8}],16) = 1; %atk 16
Y_vl([evtr{9}],17) = 1; %atk 17
Y_vl([evtr{7}],18) = 1; %atk 18
Y_vl([evtr{6}],20) = 1; %atk 20
Y_vl([evtr{5}],21) = 1; %atk 21
Y_vl([evtr{4}],22) = 1; %atk 22
Y_vl([evtr{1}],24) = 1; %atk 23cd

% labels for dataset with the other 10 attacks
evts{1} =(387:487);
evts{2} =(937:987);
evts{3} =(1387:1407);
evts{4} =(1887:2037);
evts{5} =(2687:2737);
evts{6} =(2712:2762);
evts{7} =(2907:3057);
evts{8} =(3187:3257);
evts{9} =(3887:3967);
evts{10}=(3887:3977);

Y_test=zeros(height(SCADA_test),1);
Y_test([evts{1} evts{2} evts{3} evts{4} evts{5} evts{6} evts{7} evts{8}
evts{9} evts{10}])=1;   %events flags for test
Y_test=Y_test(1:end); %to have full days in the dataset

% labeld for each DMA
Y_ts = zeros(height(SCADA_test),24);
Y_ts([evts{8}],1) = 1; %atk 1
Y_ts([evts{2}],2) = 1; %atk 2
Y_ts([evts{3}],3) = 1; %atk 3
Y_ts([evts{4}],6) = 1; %atk 6
Y_ts([evts{5} evts{6}],13) = 1; %atk 13
Y_ts([evts{1} evts{7}],22) = 1; %atk 22
Y_ts([evts{9} evts{10}],24) = 1; %atk 23cd
```

- The variable *evtr* represents the time intervals in which the first 10 attacks occur.
- The variable *evts* represents the time intervals in which the last 10 attacks occur.
- The variable *Y_val* contains the labeled output of the first 10 attacks.
- The variable *Y_test* contains the labeled output of the last 10 attacks.
- The variable *Y_vl* and the variable *Y_ts* contain the labeled outputs corresponding to each DMA.

c. Structural analysis indicates that Zone 1 should have six ARRs. The next step is to train an Autoencoder for each one of them.

   - *AdmasN* variable: presents the groups of ARRs in which an attack must be detected at the same time, in order to locate it in that DMA. N is the zone number: N= 1,2,3.
   - *ARRsN* array: presents the variables present in each ARR in the different zones.

```
load('ARRs1.mat')
```

   - Save in *Xtrain* the training data corresponding to the analyzed ARR, in *X_val* and *X_test* the test data, the first 10 attacks, and the next 10 attacks respectively.

```
%% This steps should be repeated for each ARR in the analyzed zone
a = 1;

X_train = SCADA_train(:,ARRs1{a,1});
X_test = SCADA_test(:,ARRs1{a,1});
X_val = SCADA_val(:,ARRs1{a,1});
```

   - Remove constant variables with the *RemoveConstant* function which receives the training data and the test data and returns those arrays with the constants removed.

```
[Xtn,Xts] = RemoveConstant(X_train,X_test); % Remove constant variables
and time variable
[~,Xvl] = RemoveConstant(X_train,X_val); % Remove constant variables and
time variable
```

   - Normalize training data with the Matlab *zscore* command.

```
[X1_,m,sigma] = zscore(Xtn);     % Normalize data normal
```

   - Normalize test data with the *normaldata* function which receives the mean and standard deviation of the training data, and the data to be normalized.

```
[X2_] = normaldata(m,sigma,Xvl);     % Normalize data test
[X3_] = normaldata(m,sigma,Xts);     % Normalize data test
```

- Finally, train the Autoencoder with the *AEtrain* function, which receives the training data, the dimension of the latent space and the hyper-parameters L2, SR, and SP. The function returns the trained Autoencoder, the SPE, and the MSE.

```
[AEs{a,1},SPEs{a,1},MSEs{a,1}]=AEtrain(X1_,5,0.001,1,0.99);
```

- To test the data, the *AEtest* function is used, which receives the information about the corresponding Autoencoder and the test data and returns the P_SPE.

```
[P_SPE{a,2}] = AEtest(AEs{a,1},X3_);
[P_SPE{a,1}] = AEtest(AEs{a,1},X2_);
```

5. Repeat step 4 for each Zone.

Follow *MAIN.m* script

6. The verification of the obtained results is implemented in MAIN.m.
   a. Load data.

```
load('SCADA_readings.mat'); % load raw datasets
```

   b. *PepareData* function is used to obtain the labeled outputs in the same form as step 4.b from E-Town

```
[evtr,evts,evtss,Y_val,Y_vl,Y_test,Y_ts]=PepareData(SCADA_val,SCADA_test);
```

   c. Analyze the detection and location in each zone (j = zone number).
   d. Apply AEWMA similar to step 4.d in the C town experiments.
   e. Show the 24 DMA answer individually.

```
%% Step 2: AEWMA for each ARR

for j = 1:3
    %% Zone 1
    if(j==1)
        load('Response1.mat')
        load('ARRs1.mat')
        [count_RR,~] = size(P_SPE);
        d_ = 0.9;
        d2_ = 5;
        arl_ = 900;
        for i = 1:count_RR
            [atks_det{i,1},~] = AEWMA(P_SPE{i,1},SPEs{i,1},d_,d2_,arl_);
            [atks_det{i,2},~] = AEWMA(P_SPE{i,2},SPEs{i,1},d_,d2_,arl_);
        end
        [count,~] = size(admas1);
        for i = 1:count
            b_val{i,1} = atks_det(admas1{i,1},1);
            b_test{i,1} = atks_det(admas1{i,1},2);
```

```
            [cc,~] = size(b_test{i,1});
            aux = ones(size(Y_test'));
            aux2 = ones(size(Y_val'));
            for ii = 1:cc
                aux = aux .* b_test{i,1}{ii,1};
                aux2 = aux2 .* b_val{i,1}{ii,1};
            end
            det_dma{i,1}= aux2;
            det_dma{i,2}= aux;
            figure

subplot(2,1,1);area(det_dma{i,1},'FaceColor','blue','LineStyle','none','Fa
ceAlpha',0.4);hold on;plot(Y_val,'Color','k');
            title(['Response of attack to DMA',num2str(i)])
            xlabel('First 10 attacks')
            e = {'Normal','Under Attack'};
            set(gca,'YTick',0:1:length(e)+1);
            set(gca,'YTickLabel',e);
            legend({'Detected attacks','Real
attacks'},'Location','bestoutside','NumColumns',1)

subplot(2,1,2);area(det_dma{i,2},'FaceColor','blue','LineStyle','none','Fa
ceAlpha',0.4);hold on;plot(Y_test,'Color','k');
            xlabel('Another 10 attacks')
            e = {'Normal','Under Attack'};
            set(gca,'YTick',0:1:length(e)+1);
            set(gca,'YTickLabel',e);
            legend({'Detected attacks','Real
attacks'},'Location','bestoutside','NumColumns',1)
            filename = ['Response of attack to DMA',num2str(i),'.fig'];
            saveas(gcf,filename);
        end
    end
    clear atks_det
%% Zone 2
    if(j==2)
        load('Response2.mat')
        load('ARRs2.mat')
        [count_RR,~] = size(P_SPE);
        arl_ = 100;
        d2_ = 5;
        d_ = 1.5;
        for i = 1:count_RR
            [atks_det{i,1},~] = AEWMA(P_SPE{i,1},SPEs{i,1},d_,d2_,arl_);
            [atks_det{i,2},~] = AEWMA(P_SPE{i,2},SPEs{i,1},d_,d2_,arl_);
        end
        [count,~] = size(admas2);
        for i = 1:count
            b_val{7+i,1} = atks_det(admas2{i,1},1);
            b_test{7+i,1} = atks_det(admas2{i,1},2);
            [cc,~] = size(b_test{7+i,1});
            aux = ones(size(Y_test'));
            aux2 = ones(size(Y_val'));
            for ii = 1:cc
                aux = aux .* b_test{7+i,1}{ii,1};
                aux2 = aux2 .* b_val{7+i,1}{ii,1};
            end
```

```
            det_dma{7+i,1}= aux2;
            det_dma{7+i,2}= aux;
            figure

subplot(2,1,1);area(det_dma{7+i,1},'FaceColor','blue','LineStyle','none','
FaceAlpha',0.4);hold on;plot(Y_val,'Color','k');
            title(['Response of attack to DMA',num2str(7+i)])
            xlabel('First 10 attacks')
            e = {'Normal','Under Attack'};
            set(gca,'YTick',0:1:length(e)+1);
            set(gca,'YTickLabel',e);
            legend({'Detected attacks','Real
attacks'},'Location','bestoutside','NumColumns',1)

subplot(2,1,2);area(det_dma{7+i,2},'FaceColor','blue','LineStyle','none','
FaceAlpha',0.4);hold on;plot(Y_test,'Color','k');
            xlabel('Another 10 attacks')
            e = {'Normal','Under Attack'};
            set(gca,'YTick',0:1:length(e)+1);
            set(gca,'YTickLabel',e);
            legend({'Detected attacks','Real
attacks'},'Location','bestoutside','NumColumns',1)
            filename = ['Response of attack to DMA',num2str(7+i),'.fig'];
            saveas(gcf,filename);
        end
    end
    clear atks_det
    %% Zone 3
    if(j==3)
        load('Response3.mat')
        load('ARRs3.mat')
        [count_RR,~] = size(P_SPE);
        d = 0.5;
        d2 = 6;
        arl = 700;
        for i = 1:count_RR
            [atks_det{i,1},~] = AEWMA(P_SPE{i,1},SPEs{i,1},d,d2,arl);
            [atks_det{i,2},~] = AEWMA(P_SPE{i,2},SPEs{i,1},d,d2,arl);
        end
        [count,~] = size(admas3);
        for i = 1:count
            if(i==1)
                b_val{7,1} = atks_det(admas3{i,1},1);
                b_test{7,1} = atks_det(admas3{i,1},2);

                [cc,~] = size(b_test{7,1});
                aux = ones(size(Y_test'));
                aux2 = ones(size(Y_val'));
                for ii = 1:cc
                    aux = aux .* b_test{7,1}{ii,1};
                    aux2 = aux2 .* b_val{7,1}{ii,1};
                end
                det_dma{7,1}= aux2;
                det_dma{7,2}= aux;
                figure
```

```
subplot(2,1,1);area(det_dma{7,1},'FaceColor','blue','LineStyle','none','Fa
ceAlpha',0.4);hold on;plot(Y_val,'Color','k');
                title(['Response of attack to DMA',num2str(7)])
                xlabel('First 10 attacks')
                e = {'Normal','Under Attack'};
                set(gca,'YTick',0:1:length(e)+1);
                set(gca,'YTickLabel',e);
                legend({'Detected attacks','Real
attacks'},'Location','bestoutside','NumColumns',1)

subplot(2,1,2);area(det_dma{7,2},'FaceColor','blue','LineStyle','none','Fa
ceAlpha',0.4);hold on;plot(Y_test,'Color','k');
                xlabel('Another 10 attacks')
                e = {'Normal','Under Attack'};
                set(gca,'YTick',0:1:length(e)+1);
                set(gca,'YTickLabel',e);
                legend({'Detected attacks','Real
attacks'},'Location','bestoutside','NumColumns',1)
                filename = ['Response of attack to
DMA',num2str(7),'.fig'];
                saveas(gcf,filename);
            else
                b_val{10+i,1} = atks_det(admas3{i,1},1);
                b_test{10+i,1} = atks_det(admas3{i,1},2);

                [cc,~] = size(b_test{10+i,1});
                aux = ones(size(Y_test'));
                aux2 = ones(size(Y_val'));
                for ii = 1:cc
                    aux = aux .* b_test{10+i,1}{ii,1};
                    aux2 = aux2 .* b_val{10+i,1}{ii,1};
                end
                det_dma{10+i,1}= aux2;
                det_dma{10+i,2}= aux;
                figure

subplot(2,1,1);area(det_dma{10+i,1},'FaceColor','blue','LineStyle','none',
'FaceAlpha',0.4);hold on;plot(Y_val,'Color','k');
                title(['Response of attack to DMA',num2str(10+i)])
                xlabel('First 10 attacks')
                e = {'Normal','Under Attack'};
                set(gca,'YTick',0:1:length(e)+1);
                set(gca,'YTickLabel',e);
                legend({'Detected attacks','Real
attacks'},'Location','bestoutside','NumColumns',1)

subplot(2,1,2);area(det_dma{10+i,2},'FaceColor','blue','LineStyle','none',
'FaceAlpha',0.4);hold on;plot(Y_test,'Color','k');
                xlabel('Another 10 attacks')
                e = {'Normal','Under Attack'};
                set(gca,'YTick',0:1:length(e)+1);
                set(gca,'YTickLabel',e);
                legend({'Detected attacks','Real
attacks'},'Location','bestoutside','NumColumns',1)
                filename = ['Response of attack to
DMA',num2str(10+i),'.fig'];
```

```
                saveas(gcf,filename);
            end

        end
    end

end
```

f. Show and analyze the other results. In this case, it is necessary to concatenate the 20 attacks, storing them in the *accum* variable.

```
%% Step 3: Show results of detection

% Label for general detection
[samples_ts,var] = size(Y_ts);
[samples_vl,~] = size(Y_vl);
Y = zeros(samples_ts+samples_vl,1);
Y(1:samples_vl,1) = Y_val;
Y(samples_vl+1:end,1) = Y_test;

labels = zeros(samples_ts+samples_vl,var);
labels(1:samples_vl,:) = Y_vl;
labels(samples_vl+1:end,:) = Y_ts;

% Data for general detection
[atks,~] = size(det_dma);
accum = zeros(1,samples_ts+samples_vl);
for i = 1:atks
    aux = zeros(1,samples_ts+samples_vl);
    aux(1,1:samples_vl) = det_dma{i,1};
    aux(1,samples_vl+1:end) = det_dma{i,2};
    det_dma{i,3} = aux;
    accum = accum+aux;
end
```

g. Using the *CheckDetection* function, the TPR, TNR, Sclf, FAR, FDR, STTD, and S indices are obtained. The function receives the variable with the attack detection (*accum*) and the actual labeled output of the system (Y), and returns the indices.

```
[TPR,TNR,Sclf,FAR,FDR,STTD,S] = CheckDetection(accum,Y)
r = find(accum>1);
accum(r)=1;
```

h. Obtain the incidence matrix with the *IncMatrix* function. It receives the *labels* variable that represents the real labeled output of each DMA separately. The *det_dma* variable represents the labeled output obtained from each DMA separately and the *Y* variable represents the actual outputs

```
[matrix,porcentual_matrix] = IncMatrix(labels,det_dma,Y);
```

i. *plotAtks* function allows to view the detection of each attack individually.

```
figure
area(accum,'FaceColor','blue','LineStyle','none','FaceAlpha',0.4);hold
on;plot(Y,'Color','k');
e = {'Normal','Under Attack'};
set(gca,'YTick',0:1:length(e)+1);
set(gca,'YTickLabel',e);
xlabel('Time')
title('Detection of attacks')
legend({'Detected attacks','Real
attacks'},'Location','bestoutside','NumColumns',1)
filename = ['Detection of attacks.fig'];
saveas(gcf,filename);

[Results] = plotAtks(det_dma,evtr,evtss,labels);
```