```
% ---- Autoencoder 1

%--------------------Train net1 and net2 ------------------------

%Train the networks and save after training.

[xTrain, tTrain, xValid, tValid, xTest, tTest] = LoadMNIST(3);
trainingSetSize = 50000;
validSetSize = 10000;

xTrainTmp = xTrain;
xTrain = zeros(784,trainingSetSize);
size(xTrainTmp,4)
for i = 1:trainingSetSize
    img = xTrainTmp(:,:,i);
    imgVector = img(:);
    xTrain(:,i) = imgVector;
end
xTrainNormed = normalize(xTrain,'range');
xTrain = xTrainNormed;


xValidTmp = xValid;
xValid = zeros(784,validSetSize);
for i = 1:validSetSize
    img = xValidTmp(:,:,i);
    imgVector = img(:);
    xValid(:,i) = imgVector;
end
xValidNormed = normalize(xValid,'range');
xValid = xValidNormed;


    options = trainingOptions('adam', ...
    'InitialLearnRate', 0.00100,...
    'MaxEpochs',800, ...
    'MiniBatchSize',8192, ...
    'ValidationData',{xValid,xValid}, ...
    'Shuffle', 'every-epoch',...
    'Plots','training-progress');

layers1 = [
     sequenceInputLayer(784)
     fullyConnectedLayer(50, 'WeightsInitializer', 'glorot')
     reluLayer

     fullyConnectedLayer(2, 'WeightsInitializer', 'glorot')
     reluLayer


     fullyConnectedLayer(784, 'WeightsInitializer', 'glorot')
     reluLayer
```

```
    regressionLayer];

layers2 = [
     sequenceInputLayer(784)
     fullyConnectedLayer(50, 'WeightsInitializer', 'glorot')
     reluLayer

     fullyConnectedLayer(4, 'WeightsInitializer', 'glorot')
     reluLayer


     fullyConnectedLayer(784, 'WeightsInitializer', 'glorot')
     reluLayer


    regressionLayer];

net1 = trainNetwork(xTrain, xTrain, layers1, options);

net2 = trainNetwork(xTrain, xTrain, layers2, options);

save;
     %---------- END TRAINING----------

% --------- ANALYSIS START--------
load('matlab.mat');

xTestTmp = xTest;
xTest = zeros(784,1000);
for i = 1:1000
    img = xTestTmp(:,:,i);
    imgVector = img(:);
    xTest(:,i) = imgVector;
end
xTestNormed = normalize(xTest,'range');
xTest = xTestNormed;

%Layers
[layerEncoderNet1, layerDecoderNet1] = SeperateEncoderDecoder(net1,2);
[layerEncoderNet2, layerDecoderNet2] = SeperateEncoderDecoder(net2,4);

%Assemble network
netEncoder1 = assembleNetwork(layerEncoderNet1);
netEncoder2 = assembleNetwork(layerEncoderNet2);

netDecoder1 = assembleNetwork(layerDecoderNet1);
netDecoder2 = assembleNetwork(layerDecoderNet2);


%After a lot of testing in PlotAndCompareImages, 0's and 1's
%could quite often be reproduced.
%Encode first network and plot 0's and 1's.
```

```
bottleNeckOutput = predict(netEncoder1,xTest);
zeroPairs = [];
zeroPairsIndx = 1;

onePairs = [];
onePairsIndx = 1;

restPairs = [];
restPairsIndx = 1;
for i = 1:1000
    t = tTest(i);
    t = double(t);
    neuronPair = bottleNeckOutput(:,i);
    x = neuronPair(1);
    y = neuronPair(2);
    if (t == 1)
        zeroPairs(:,zeroPairsIndx) = neuronPair;
        zeroPairsIndx = zeroPairsIndx + 1;
    elseif (t == 2)
        onePairs(:,onePairsIndx) = neuronPair;
        onePairsIndx = onePairsIndx + 1;

    else
        restPairs(:,restPairsIndx) = neuronPair;
        restPairsIndx = restPairsIndx + 1;
    end

end
figure(1);
scatter(zeroPairs(1,:), zeroPairs(2,:),'b');
hold on
scatter(onePairs(1,:), onePairs(2,:),'r');
hold on
scatter(restPairs(1,:), restPairs(2,:),'g');
legend('0','1','Rest');


%Decode first network, and check that ones and zeros can be generated
 by
%looking at the scatterplot and picking values within thier
 boundaries.
%Below are the random values picked.
checkZeros = [[2;6], [3; 7.5], [4;8], [1;5], [5;11]];
checkOnes = [[5;1], [7;0.5], [9;1], [5;1.5], [10;1]];

figure(2)
for i = 1:5
    outZero = predict(netDecoder1,checkZeros(:,i));
    outZero2d = reshape(outZero,28,28);
    subplot(2,5,i), imshow(mat2gray(outZero2d));

    outOne = predict(netDecoder1,checkOnes(:,i));
    outOne2d = reshape(outOne,28,28);
    subplot(2,5,5+i), imshow(mat2gray(outOne2d));
```

```
end

%For the second network..
%Digits that could be well reproduced were 1's, 0's and 9's.

%Encode second network and inspect the bottleneck output.
bottleNeckOutput2 = predict(netEncoder2,xTest);
inspectionTarget = 4;
ts = 0;
n1 = 0;
n2 = 0;
n3 = 0;
n4 = 0;
for i = 1:1000
    t = tTest(i);
    t = double(t);
    if t == inspectionTarget
       out = bottleNeckOutput2(:,i)
       n1 = n1 + out(1);
       n2 = n2 + out(2);
       n3 = n3 + out(3);
       n4 = n4 + out(4);
       ts = ts+1;
    end
end
n1Avr = n1/ts;
n2Avr = n2/ts;
n3Avr = n3/ts;
n4Avr = n4/ts;

n1Avr
n2Avr
n3Avr
n4Avr


checkZeros = [[8.47; 10.37; 9.77; 10.18], [11;10;9;10], [8;9;9;9],
 [10;8;8;10], [10;11;10;8]];
checkOnes = [[19.69; 2.10; 8.52; 6.70], [19;0.5;10;9], [18;1;9;12],
 [21;1.5;11;9], [22;1;10;10]];
checkNines = [[15.83; 9.39; 6.28; 13.71], [14;9;7;13], [15;8;5;13],
 [13;7;5;12], [14.5;8.3;7;12]];

figure(2)
for i = 1:5
    outZero = predict(netDecoder2,checkZeros(:,i));
    outZero2d = reshape(outZero,28,28);
    subplot(3,5,i), imshow(mat2gray(outZero2d));

    outOne = predict(netDecoder2,checkOnes(:,i));
    outOne2d = reshape(outOne,28,28);
    subplot(3,5,5+i), imshow(mat2gray(outOne2d));

    outNine = predict(netDecoder2,checkNines(:,i));
```

```
    outNine2d = reshape(outNine,28,28);
    subplot(3,5,10+i), imshow(mat2gray(outNine2d));
end
% ----------------------ANALYSIS END-----------------

%---- PlotAndCompareImages ------------------
load('matlab.mat');

network = net2; %Just change this (net1, net2).
compIndexes = [];
for i = 1:10
    out = Inf; % -> entry to while loop
    while((out) ~= i)
        r = randi([1,10000],1,1);
        validOut = tValid(r);
        out = double(validOut);
    end
    compIndexes(i) = r;
end
for i = 1:10

    compIndx = compIndexes(i);
    feed = xValid(:,compIndx);
    target = xValidTmp(:,:,1,compIndx);

    output = predict(network,feed);
    output2d = reshape(output,28,28);

    %plot target and output
    subplot(2,10,i), imshow(mat2gray(target));
    subplot(2,10,10+i), imshow(mat2gray(output2d));
end
%---- PlotAndCompareImages END ------------------
function [layers_encode, layers_decode] =
 SeperateEncoderDecoder(net,decInputSize)
 layers_encode= nnet.cnn.layer.SequenceInputLayer(0);
 layers_decode= nnet.cnn.layer.SequenceInputLayer(0);

  for i = 1:5
    layers_encode(i) = net.Layers(i);
  end
  layers_encode(6) = regressionLayer;

layers_decode(1) =  sequenceInputLayer(decInputSize);
 for i = 1:3
   layers_decode(i+1) = net.Layers(i+5);
 end
end
```

Published with MATLAB® R2020a