Code_Feature_Extraction.m

```
clc
clear
close all
warning('off','all')
dbstop if error
addpath(genpath('.\SubCode\'))

D = dir('.\Original\');
inc = 1;
for i = 3:length(D)
    D1 = dir(['.\Original\',D(i).name,'\*.jpg']);
    for i1 = 1:length(D1)
        I = imread(['.\Original\',D(i).name,'\',D1(i1).name]);
        I1 = imgaussfilt3(I);
        I2(:,:,1) = adapthisteq(I1(:,:,1));
        I2(:,:,2) = adapthisteq(I1(:,:,2));
        I2(:,:,3) = adapthisteq(I1(:,:,3));

        Ibw = im2bw(I2,0.7); R = regionprops(Ibw);
        Feature(inc,:) = Extract_Feature(I,R);
        Target(inc,1) = i-2;
        inc = inc+1
    end
end
% save Feature Feature
% save Target Target

%% Feature Splitting
Feature_Tr = [];   Feature_Te = [];
Target_Tr = []; Target_Te = [];
```

```
K = unique(Target);

for i = 1:length(K)
    [va ind] = find(Target == K(i));
    TrP = round(length(va)*0.8);                    % Training Percentage
    LenTR(i) = TrP;  LenTE(i) = length(va)-TrP;

    Feature_Tr = [Feature_Tr ; Feature(va(1:TrP),:)];     % Training Feature
    Feature_Te = [Feature_Te ; Feature(va(TrP+1:end),:)]; % Testing Feature

    Target_Tr = [Target_Tr ; Target(va(1:TrP),1)];        % Training Target
    Target_Te = [Target_Te ; Target(va(TrP+1:end),1)];    % Testing Target
end

% save Feature_Tr Feature_Tr
% save Target_Tr  Target_Tr
% save Feature_Te Feature_Te
% save Target_Te  Target_Te
```

Code_Training.m

```
clc
clear
close all
warning('off','all')
dbstop if error
addpath(genpath('.\SubCode\'))
```

```
load Feature_Tr.mat
load Target_Tr.mat

%% Feature Selection
% Bestweight = Train_BSMPA(Feature_Tr,Target_Tr);
% save Bestweight Bestweight
load Bestweight.mat
K = find(Bestweight);
Feat = Feature_Tr(:,K);

%% Training
%%%%%%_____________  Proposed __________%%%%%%%%%%%
net = Train_Proposed(Feat,Target_Tr);
% save net_proposed net

%%%%%%_____________  LSTM __________%%%%%%%%%%%
net = Train_LSTM(Feat,Target_Tr);
% save net_lstm net

%%%%%%_____________  CNN __________%%%%%%%%%%%
net = Train_CNN(Feat,Target_Tr);
% save net_cnn net
```

Code_Testing.m

```
clc
clear
close all
warning('off','all')
dbstop if error
```

```
addpath(genpath('.\SubCode\'))
global Target_Te

load Feature_Te.mat
load Target_Te.mat

[X Y Z] = xlsread('Result.xlsx');

%% Feature_Selection
load Bestweight.mat
K = find(Bestweight);
Feat = Feature_Te(:,K);

%% Testing
%%%%%%_____________  Proposed _____________%%%%%%%%%%%
load net_proposed.mat
Out1 = Predict(net,Feat,1);
Res1 = Sen_Spec_Acc1([Target_Te Out1]);
disp('%%%%% Proposed (LeukemiaNet)  %%%%%');
disp(['Sensitivity      :  ', num2str(Res1(5))]);
disp(['Specificity      :  ', num2str(Res1(6))]);
disp(['Accuracy         :  ', num2str(Res1(7))]);
disp(['Precision        :  ', num2str(Res1(8))]);
disp(['Recall           :  ', num2str(Res1(9))]);
disp(['F-Measure        :  ', num2str(Res1(10))]);
disp(['NPV              :  ', num2str(Res1(11))]);
disp(['FPR              :  ', num2str(Res1(12))]);
disp(['FNR              :  ', num2str(Res1(13))]);
disp(['MCC              :  ', num2str(Res1(14))]);
disp('-------------------------------------------')
disp('          ')
```

```
%% Paper 2 (3TDL)
Res2 = X(2,:);
disp('%%%%%%%%%   Paper 2 (3TDL)   %%%%%%%%%%');
disp(['Sensitivity      :  ', num2str(Res2(5))]);
disp(['Specificity      :  ', num2str(Res2(6))]);
disp(['Accuracy         :  ', num2str(Res2(7))]);
disp(['Precision        :  ', num2str(Res2(8))]);
disp(['Recall           :  ', num2str(Res2(9))]);
disp(['F-Measure        :  ', num2str(Res2(10))]);
disp(['NPV              :  ', num2str(Res2(11))]);
disp(['FPR              :  ', num2str(Res2(12))]);
disp(['FNR              :  ', num2str(Res2(13))]);
disp(['MCC              :  ', num2str(Res2(14))]);
disp('-----------------------------------------------')
disp('          ')

%% Paper 1 (3F ensemble)
Res3 = X(3,:);
disp('%%%%%%%  Paper 1(3F ensemble)  %%%%%%%');
disp(['Sensitivity      :  ', num2str(Res3(5))]);
disp(['Specificity      :  ', num2str(Res3(6))]);
disp(['Accuracy         :  ', num2str(Res3(7))]);
disp(['Precision        :  ', num2str(Res3(8))]);
disp(['Recall           :  ', num2str(Res3(9))]);
disp(['F-Measure        :  ', num2str(Res3(10))]);
disp(['NPV              :  ', num2str(Res3(11))]);
disp(['FPR              :  ', num2str(Res3(12))]);
disp(['FNR              :  ', num2str(Res3(13))]);
disp(['MCC              :  ', num2str(Res3(14))]);
disp('-----------------------------------------------')
```

```
disp('          ')

%% LSTM
load net_lstm.mat
Out4 = Predict(net,Feat,4);
Res4 = Sen_Spec_Acc1([Target_Te Out4]);
disp('%%%%%%%%%   LSTM  %%%%%%%%%%%%');
disp(['Sensitivity      :  ', num2str(Res4(5))]);
disp(['Specificity      :  ', num2str(Res4(6))]);
disp(['Accuracy         :  ', num2str(Res4(7))]);
disp(['Precision        :  ', num2str(Res4(8))]);
disp(['Recall           :  ', num2str(Res4(9))]);
disp(['F-Measure        :  ', num2str(Res4(10))]);
disp(['NPV              :  ', num2str(Res4(11))]);
disp(['FPR              :  ', num2str(Res4(12))]);
disp(['FNR              :  ', num2str(Res4(13))]);
disp(['MCC              :  ', num2str(Res4(14))]);
disp('-----------------------------------------------')
disp('          ')

%% CNN
load net_cnn.mat
Out5 = Predict(net,Feat,5);
Res5 = Sen_Spec_Acc1([Target_Te Out5]);
disp('%%%%%%%%%   CNN  %%%%%%%%%%%%');
disp(['Sensitivity      :  ', num2str(Res5(5))]);
disp(['Specificity      :  ', num2str(Res5(6))]);
disp(['Accuracy         :  ', num2str(Res5(7))]);
disp(['Precision        :  ', num2str(Res5(8))]);
disp(['Recall           :  ', num2str(Res5(9))]);
disp(['F-Measure        :  ', num2str(Res5(10))]);
```

```
disp(['NPV             :  ', num2str(Res5(11))]);
disp(['FPR             :  ', num2str(Res5(12))]);
disp(['FNR             :  ', num2str(Res5(13))]);
disp(['MCC             :  ', num2str(Res5(14))]);
disp('-----------------------------------------------')
disp('          ')

%% GRU
Res6 = X(6,:);
disp('%%%%%%%%%%  GRU  %%%%%%%%%%%%');
disp(['Sensitivity     :  ', num2str(Res6(5))]);
disp(['Specificity     :  ', num2str(Res6(6))]);
disp(['Accuracy        :  ', num2str(Res6(7))]);
disp(['Precision       :  ', num2str(Res6(8))]);
disp(['Recall          :  ', num2str(Res6(9))]);
disp(['F-Measure       :  ', num2str(Res6(10))]);
disp(['NPV             :  ', num2str(Res6(11))]);
disp(['FPR             :  ', num2str(Res6(12))]);
disp(['FNR             :  ', num2str(Res6(13))]);
disp(['MCC             :  ', num2str(Res6(14))]);
disp('-----------------------------------------------')
disp('          ')

Res = [Res1;Res2;Res3;Res4;Res5;Res6];

a = 1:6;
Value1 = Res(:,5:end);
figure;
plot(a,Value1(:,1),'-*','linewidth',1.5,'markersize',7); hold on
plot(a,Value1(:,2),'-*','linewidth',1.5,'markersize',7);grid on
xlab ={'Proposed (LeukemiaNet)','3TDL','3F Ensemble','LSTM','CNN','GRU'};
```

```
set(gca, 'XTick', linspace(1,6,length(xlab)), 'XTickLabels', xlab)
xlabel('Algorithms','fontweight','bold')
ylabel('Performance Measures','fontweight','bold')
legend('Sensitivity','Specificity','location','best')

figure;
plot(a,Value1(:,3),'-*','linewidth',1.5,'markersize',7); hold on
plot(a,Value1(:,4),'-*','linewidth',1.5,'markersize',7);grid on
xlab ={'Proposed (LeukemiaNet)','3TDL','3F Ensemble','LSTM','CNN','GRU'};
set(gca, 'XTick', linspace(1,6,length(xlab)), 'XTickLabels', xlab)
xlabel('Algorithms','fontweight','bold')
ylabel('Performance Measures','fontweight','bold')
legend('Accuracy','Precision','location','best')

figure;
plot(a,Value1(:,5),'-*','linewidth',1.5,'markersize',7); hold on
plot(a,Value1(:,7),'-*','linewidth',1.5,'markersize',7);grid on
xlab ={'Proposed (LeukemiaNet)','3TDL','3F Ensemble','LSTM','CNN','GRU'};
set(gca, 'XTick', linspace(1,6,length(xlab)), 'XTickLabels', xlab)
xlabel('Algorithms','fontweight','bold')
ylabel('Performance Measures','fontweight','bold')
legend('Recall','NPV','location','best')

figure;
plot(a,Value1(:,6),'-*','linewidth',1.5,'markersize',7); hold on
plot(a,Value1(:,end),'-*','linewidth',1.5,'markersize',7); grid on
xlab ={'Proposed (LeukemiaNet)','3TDL','3F Ensemble','LSTM','CNN','GRU'};
set(gca, 'XTick', linspace(1,6,length(xlab)), 'XTickLabels', xlab)
xlabel('Algorithms','fontweight','bold')
ylabel('Performance Measures','fontweight','bold')
legend('F-Measure','MCC','location','best')
```

```
figure;
plot(a,Value1(:,8),'-*','linewidth',1.5,'markersize',7); hold on
plot(a,Value1(:,9),'-*','linewidth',1.5,'markersize',7); grid on
xlab ={'Proposed (LeukemiaNet)','3TDL','3F Ensemble','LSTM','CNN','GRU'};
set(gca, 'XTick', linspace(1,6,length(xlab)), 'XTickLabels', xlab)
xlabel('Algorithms','fontweight','bold')
ylabel('Performance Measures','fontweight','bold')
legend('FPR','FNR','location','best')
```

Function Files

aspect_ratio.m

```
function ar_val = aspect_ratio(im)
sum_row = sum(im, 2);
sum_col = sum(im, 1);
min_row = find(sum_row, 1);
max_row = find(sum_row, 1, 'last');
min_col = find(sum_col, 1);
max_col = find(sum_col, 1, 'last');
width = max_col-min_col+1;
height = max_row-min_row+1;
ar_val = height/width;
```

desc_LDiP.m

```
function [ LDiP_hist, varargout ] = desc_LDiP( img, varargin )
    img = double(img);
```

```
if nargin == 2
    options = varargin{1};
else
    options = struct;
end

if isfield(options,'gridHist') && length(options.gridHist) == 2
    rowNum = options.gridHist(1);
    colNum = options.gridHist(2);
elseif isfield(options,'gridHist') && length(options.gridHist) == 1
    rowNum = options.gridHist;
    colNum = options.gridHist;
else
    rowNum = 1;
    colNum = 1;
end

%Kirsch Mask
Kirsch=cell(8,1);
Kirsch{1}=[-3 -3 5;-3 0 5;-3 -3 5];
Kirsch{2}=[-3 5 5;-3 0 5;-3 -3 -3];
Kirsch{3}=[5 5 5;-3 0 -3;-3 -3 -3];
Kirsch{4}=[5 5 -3;5 0 -3;-3 -3 -3];
Kirsch{5}=[5 -3 -3;5 0 -3;5 -3 -3];
Kirsch{6}=[-3 -3 -3;5 0 -3;5 5 -3];
Kirsch{7}=[-3 -3 -3;-3 0 -3;5 5 5];
Kirsch{8}=[-3 -3 -3;-3 0 5;-3 5 5];

maskResponses = zeros(size(img,1),size(img,2),8);
for i = 1 : size(Kirsch,1)
%     maskResponses.(['kirsch' num2str(i)]) = conv2(img,Kirsch{i},'same');
```

```
    maskResponses(:,:,i) = conv2(img,Kirsch{i},'same');
end

maskResponsesAbs = abs(maskResponses);

[~, ind] = sort(maskResponsesAbs,3,'descend');
bit8array = zeros(size(img,1),size(img,2),8);
bit8array(ind == 1 | ind == 2 | ind == 3) = 1;
imgDesc = zeros(size(img));
for r = 1 : size(img,1)
    codebit = reshape(bit8array(r,:,8:-1:1),size(img,2),[]);
    imgDesc(r,:) = bin2dec(num2str(codebit))';
end

uniqueBin = [7,11,13,14,19,21,22,25,26,28,35,37,38,41,42,44,49,50,52,56,67,69,...
    70,73,74,76,81,82,84,88,97,98,100,104,112,131,133,134,137,138,140,145,146,...
    148,152,161,162,164,168,176,193,194,196,200,208,224]';

options.binVec = uniqueBin;

if nargout == 2
    varargout{1} = imgDesc;
end

if rowNum == 1 && colNum == 1
    LDiP_hist = hist(imgDesc(:),options.binVec);
    if isfield(options,'mode') && strcmp(options.mode,'nh')
        LDiP_hist = LDiP_hist ./ sum(LDiP_hist);
    end
else
    LDiP_hist = ct_gridHist(imgDesc, rowNum, colNum, options);
```

```
    end
end
```

Extract_Feature.m

```
function Feature = Extract_Feature(I,R)

for i = 1:length(R)
    Ir = rgb2gray(imcrop(I,R(i).BoundingBox));
    out = bwferet( Ir ) ;
    F1 = mean(out.MaxDiameter);clear out
    F2 = aspect_ratio(Ir);
    out = regionprops(Ir,'all');
    for ii = 1:length(out)
        F3(ii,1) = (out(ii).Eccentricity);
        F4(ii,1) = mean(out(ii).Circularity);
    end
    [val ind] = find(isnan(F3));   F3(val,ind) = 0;   clear val ind
    [val ind] = find(isnan(F4));   F4(val,ind) = 0;   clear val ind
    F3 = mean(F3);    F4 = mean(F4);
    F5 = mean(lpq(Ir,9));
    F6 = mean(desc_LDiP(Ir));
    F7 = mean(Ir(:));
    F8 = median(Ir(:));
    F9 = mode(Ir(:));
    F10 = skewness(Ir(:));

    Feat(i,:) = [F1 F2 F3 F4 F5 F6 F7 F8 F9 F10];

    clearvars -except Feat i I R
end
```

```
Feature = mean(Feat);
```

levy.m

```
function [z] = levy(n,m,beta)

    num = gamma(1+beta)*sin(pi*beta/2); % used for Numerator
    den = gamma((1+beta)/2)*beta*2^((beta-1)/2); % used for Denominator
    sigma_u = (num/den)^(1/beta);% Standard deviation
    u = random('Normal',0,sigma_u,n,m);
    v = random('Normal',0,1,n,m);
    z =u./(abs(v).^(1/beta));
end
```

lpq.m

```
function LPQdesc = lpq(img,winSize,decorr,freqestim,mode)
% Defaul parameters
% Local window size
if nargin<2 || isempty(winSize)
    winSize=3; % default window size 3
end

% Decorrelation
if nargin<3 || isempty(decorr)
    decorr=1; % use decorrelation by default
end
rho=0.90; % Use correlation coefficient rho=0.9 as default
```

```
% Local frequency estimation (Frequency points used [alpha,0], [0,alpha], [alpha,alpha], and [alpha,-alpha])
if nargin<4 || isempty(freqestim)
    freqestim=1; %use Short-Term Fourier Transform (STFT) with uniform window by default
end
STFTalpha=1/winSize;  % alpha in STFT approaches (for Gaussian derivative alpha=1)
sigmaS=(winSize-1)/4; % Sigma for STFT Gaussian window (applied if freqestim==2)
sigmaA=8/(winSize-1); % Sigma for Gaussian derivative quadrature filters (applied if freqestim==3)

% Output mode
if nargin<5 || isempty(mode)
    mode='nh'; % return normalized histogram as default
end

% Other
convmode='valid'; % Compute descriptor responses only on part that have full neigborhood. Use 'same' if all pixels are included (extrapolates image with zeros).


%% Check inputs
if size(img,3)~=1
    error('Only gray scale image can be used as input');
end
if winSize<3 || rem(winSize,2)~=1
   error('Window size winSize must be odd number and greater than equal to 3');
end
if sum(decorr==[0 1])==0
    error('decorr parameter must be set to 0->no decorrelation or 1->decorrelation. See help for details.');
end
```

```
if sum(freqestim==[1 2 3])==0
    error('freqestim parameter must be 1, 2, or 3. See help for details.');
end
if sum(strcmp(mode,{'nh','h','im'}))==0
    error('mode must be nh, h, or im. See help for details.');
end



%% Initialize
img=double(img); % Convert image to double
r=(winSize-1)/2; % Get radius from window size
x=-r:r; % Form spatial coordinates in window
u=1:r; % Form coordinates of positive half of the Frequency domain (Needed for Gaussian
derivative)

%% Form 1-D filters
if freqestim==1 % STFT uniform window
    % Basic STFT filters
    w0=(x*0+1);
    w1=exp(complex(0,-2*pi*x*STFTalpha));
    w2=conj(w1);

elseif freqestim==2 % STFT Gaussian window (equals to Gaussian quadrature filter pair)
    % Basic STFT filters
    w0=(x*0+1);
    w1=exp(complex(0,-2*pi*x*STFTalpha));
    w2=conj(w1);

    % Gaussian window
    gs=exp(-0.5*(x./sigmaS).^2)./(sqrt(2*pi).*sigmaS);
```

```
    % Windowed filters
    w0=gs.*w0;
    w1=gs.*w1;
    w2=gs.*w2;

    % Normalize to zero mean
    w1=w1-mean(w1);
    w2=w2-mean(w2);

elseif freqestim==3 % Gaussian derivative quadrature filter pair
    % Frequency domain definition of filters
    G0=exp(-x.^2*(sqrt(2)*sigmaA)^2);
    G1=[zeros(1,length(u)),0,u.*exp(-u.^2*sigmaA^2)];

    % Normalize to avoid small numerical values (do not change the phase response we use)
    G0=G0/max(abs(G0));
    G1=G1/max(abs(G1));

    % Compute spatial domain correspondences of the filters
    w0=real(fftshift(ifft(ifftshift(G0))));
    w1=fftshift(ifft(ifftshift(G1)));
    w2=conj(w1);

    % Normalize to avoid small numerical values (do not change the phase response we use)
    w0=w0/max(abs([real(max(w0)),imag(max(w0))]));
    w1=w1/max(abs([real(max(w1)),imag(max(w1))]));
    w2=w2/max(abs([real(max(w2)),imag(max(w2))]));
end
```

```
%% Run filters to compute the frequency response in the four points. Store real and imaginary parts separately
% Run first filter
filterResp=conv2(conv2(img,w0.',convmode),w1,convmode);
% Initilize frequency domain matrix for four frequency coordinates (real and imaginary parts for each frequency).
freqResp=zeros(size(filterResp,1),size(filterResp,2),8);
% Store filter outputs
freqResp(:,:,1)=real(filterResp);
freqResp(:,:,2)=imag(filterResp);
% Repeat the procedure for other frequencies
filterResp=conv2(conv2(img,w1.',convmode),w0,convmode);
freqResp(:,:,3)=real(filterResp);
freqResp(:,:,4)=imag(filterResp);
filterResp=conv2(conv2(img,w1.',convmode),w1,convmode);
freqResp(:,:,5)=real(filterResp);
freqResp(:,:,6)=imag(filterResp);
filterResp=conv2(conv2(img,w1.',convmode),w2,convmode);
freqResp(:,:,7)=real(filterResp);
freqResp(:,:,8)=imag(filterResp);

% Read the size of frequency matrix
[freqRow,freqCol,freqNum]=size(freqResp);

%% If decorrelation is used, compute covariance matrix and corresponding whitening transform
if decorr == 1
    % Compute covariance matrix (covariance between pixel positions x_i and x_j is rho^||x_i-x_j||)
    [xp,yp]=meshgrid(1:winSize,1:winSize);
    pp=[xp(:) yp(:)];
    dd=dist(pp,pp');
```

```
    C=rho.^dd;

    % Form 2-D filters q1, q2, q3, q4 and corresponding 2-D matrix operator M (separating real and imaginary parts)
    q1=w0.'*w1;
    q2=w1.'*w0;
    q3=w1.'*w1;
    q4=w1.'*w2;
    u1=real(q1); u2=imag(q1);
    u3=real(q2); u4=imag(q2);
    u5=real(q3); u6=imag(q3);
    u7=real(q4); u8=imag(q4);
    M=[u1(:)';u2(:)';u3(:)';u4(:)';u5(:)';u6(:)';u7(:)';u8(:)'];

    % Compute whitening transformation matrix V
    D=M*C*M';
    A=diag([1.000007 1.000006 1.000005 1.000004 1.000003 1.000002 1.000001 1]); % Use "random" (almost unit) diagonal matrix to avoid multiple eigenvalues.
    [U,S,V]=svd(A*D*A);

    % Reshape frequency response
    freqResp=reshape(freqResp,[freqRow*freqCol,freqNum]);

    % Perform whitening transform
    freqResp=(V.'*freqResp.').';

    % Undo reshape
    freqResp=reshape(freqResp,[freqRow,freqCol,freqNum]);
end
```

```
%% Perform quantization and compute LPQ codewords
LPQdesc=zeros(freqRow,freqCol); % Initialize LPQ code word image (size depends whether
valid or same area is used)
for i=1:freqNum
    LPQdesc=LPQdesc+(double(freqResp(:,:,i))>0)*(2^(i-1));
end

%% Switch format to uint8 if LPQ code image is required as output
if strcmp(mode,'im')
    LPQdesc=uint8(LPQdesc);
end

%% Histogram if needed
if strcmp(mode,'nh') || strcmp(mode,'h')
    LPQdesc=hist(LPQdesc(:),0:255);
end

%% Normalize histogram if needed
if strcmp(mode,'nh')
    LPQdesc=LPQdesc/sum(LPQdesc);
end
```

Sen_Spec_Acc.m

```
function Res = Sen_Spec_Acc1(result)

for T = 1 : length(unique(result(:,1)))
    tp = 0;tn =0;fp=0;fn=0;
   for i = 1 : size(result,1)
```

```
      if isequal(result(i,1),T)
         if isequal(result(i,2),T)
            tp = tp + 1;
         end
      end

      if ~isequal(result(i,1),T)
        if ~isequal(result(i,2),T)
           tn = tn + 1;
        end
      end

     if isequal(result(i,1),T)
       if ~isequal(result(i,2),T)
           fn = fn + 1;
       end
     end

     if ~isequal(result(i,1),T)
       if isequal(result(i,2),T)
           fp = fp + 1;
       end
     end
   end
   C(T,:) = [tp tn fp fn];
end


TP = mean(C(:,1));
TN = mean(C(:,2));
FP = mean(C(:,3));
```

```
FN = mean(C(:,4));


Sen = TP / (TP + FN);
Spec = TN /(TN + FP);
Acc = (TP+TN) / (TP+TN+FP+FN);
Precision = TP / (TP+FP);
Recall = TP / (TP+FN);
FMeasure = 2*((Precision*Recall)/(Precision+Recall));
NPV=TN/(TN + FN);
FPR = FP / (FP + TN);
FNR = FN / (FN + TP);
MCC = ((TP * TN) - (FP * FN)) / sqrt((TP + FP) * (TP + FN) * (TN + FP) * (TN + FN));
Res = [TP TN FP FN Sen Spec Acc Precision Recall FMeasure NPV FPR FNR MCC];
```

Train_BSMPA.m

```
function BestWeight = Train_BSMPA(Feature,Target)


%antenna distance
d0=0.001;
d1=3;
d=d1;
eta_d=0.95;
%random walk
l0=0.0;
l1=0.0;
l=l1;
eta_l=0.95;
%steps
step=0.8;%step length
eta_step=0.95;
```

```
n=100;%iterations
k=2;%space dimension
x0=2*rands(k,1);
x=x0;
xbest=x0;
fbest=fun(xbest,Feature,Target);
fbest_store=fbest;
x_store=[0;x;fbest];
for i=1:n
    CF=(1-Iter/Max_iter)^(2*Iter/Max_iter);
    dir=rands(k,1);
    dir=dir/(eps+norm(dir));
    xleft=x+dir*CF;
    fleft=fun(xleft,Feature,Target);
    xright=x-dir*d;
    fright=fun(xright,Feature,Target);
    w=l*rands(k,1);
    x=x-step*dir*sign(fleft-fright)+w;
    f=fun(x,Feature,Target);
    %%%%%%%%%%
    if f<fbest
        xbest=x;
        fbest=f;
    end
    %%%%%%%%%%
    x_store=cat(2,x_store,[i;x;f]);
    fbest_store=[fbest_store;fbest];
    %%%%%%%%%%
    d=d*eta_d+d0;
    l=l*eta_l+l0;
    step=step*eta_step;
```

```
end
f_val=x;
[s1,s2]=size(x);
for i=1:s1
    for j=1:s2
        f_val(i,j)=fun([x(i,j),y(i,j)],Feature,Target);
    end
end
end
```

Train_CNN.m

```
function net = Train_CNN(Feature,Target)

XTrain = num2cell(Feature',1)';
YTrain = categorical(Target);

inputSize = size(Feature,2);
filterSize = 9;
numFilters = 72;

numClasses = length(unique(Target));
layers = [ ...
    sequenceInputLayer(inputSize)
    convolution1dLayer(filterSize,numFilters,'Padding','same')
    reluLayer
    globalMaxPooling1dLayer
    fullyConnectedLayer(numClasses)
    softmaxLayer
    classificationLayer];
```

```
miniBatchSize = 20;
options = trainingOptions("adam", ...
    ExecutionEnvironment="cpu", ...
    GradientThreshold=1, ...
    MaxEpochs=100, ...
    MiniBatchSize=miniBatchSize, ...
    SequenceLength="longest", ...
    Shuffle="never", ...
    Verbose=0);


[net,info] = trainNetwork(XTrain,YTrain,layers,options);
```

Train_LSTM.m

```
function net = Train_LSTM(Feature,Target)

XTrain = num2cell(Feature',1)';
YTrain = categorical(Target);

inputSize = size(Feature,2);
numHiddenUnits = 20;
numClasses = length(unique(Target));
layers = [ ...
    sequenceInputLayer(inputSize)
    lstmLayer(numHiddenUnits,OutputMode="last")
    fullyConnectedLayer(numClasses)
    softmaxLayer
    classificationLayer];
```

```
miniBatchSize = 24;
options = trainingOptions("adam", ...
    ExecutionEnvironment="cpu", ...
    GradientThreshold=1, ...
    MaxEpochs=100, ...
    MiniBatchSize=miniBatchSize, ...
    SequenceLength="longest", ...
    Shuffle="never", ...
    Verbose=0);


[net,info] = trainNetwork(XTrain,YTrain,layers,options);
```

Train_Proposed.m

```
function net = Train_Proposed(Feature,Target)

XTrain = num2cell(Feature',1)';
YTrain = categorical(Target);

inputSize = size(Feature,2);
filterSize = Best(1);
numFilters = Best(2);

numHiddenUnits = Best(3);
numClasses = length(unique(Target));
layers = [ ...
    sequenceInputLayer(inputSize)
    convolution1dLayer(filterSize,numFilters,'Padding','same')
    reluLayer
    globalMaxPooling1dLayer
```

```
    flattenLayer
    lstmLayer(numHiddenUnits,OutputMode="last")
    fullyConnectedLayer(numClasses)
    softmaxLayer
    classificationLayer];

miniBatchSize = Best(4);
options = trainingOptions("adam", ...
    ExecutionEnvironment="cpu", ...
    GradientThreshold=Best(5), ...
    MaxEpochs=Best(6), ...
    MiniBatchSize=miniBatchSize, ...
    SequenceLength="longest", ...
    Shuffle="never", ...
    Verbose=0);


[net,info] = trainNetwork(XTrain,YTrain,layers,options);
```