LAMPIRAN

LAMPIRAN 1 Kode Pre-processing Citra

```
clc;
clear;

% Lokasi dataset utama
datasetPath                                                    =
'C:\Users\diasp\Favorites\Downloads\Dataset';

% Subset data
subsets = {'Data Training', 'Data Testing'};

% Kategori tulisan
categories = {'Teratur', 'Tidak Teratur'};

% Folder output
outputPath          =          fullfile(datasetPath,
'Preprocessed_HEQ');
if ~exist(outputPath, 'dir')
    mkdir(outputPath);
end

for s = 1:length(subsets)
    subset = subsets{s};

    for c = 1:length(categories)
        category = categories{c};

        inputFolder   = fullfile(datasetPath, subset,
category);
        outputFolder  =  fullfile(outputPath,  subset,
category);

        if ~exist(outputFolder, 'dir')
            mkdir(outputFolder);
        end

        % Ambil file gambar
        imageFiles    =    dir(fullfile(inputFolder,
'*.jpg'));
        imageFiles                =                [imageFiles;
dir(fullfile(inputFolder, '*.png'))];
        imageFiles                =                [imageFiles;
dir(fullfile(inputFolder, '*.bmp'))];

        for i = 1:length(imageFiles)
            % Baca gambar
            imgPath      =      fullfile(inputFolder,
imageFiles(i).name);
            img = imread(imgPath);

            % Grayscale
            if size(img,3) == 3
                img_gray = rgb2gray(img);
            else
                img_gray = img;
            end
```

```
            % MEDIAN FILTER
            img_gray = medfilt2(img_gray,[3 3]);

            % Invers grayscale (tulisan jadi terang)
            img_inv = imcomplement(img_gray);

            % CLAHE
            img_clahe = adapthisteq(img_inv);

            % Hasil akhir
            img_final = img_clahe;

            % Simpan hasil
            outputFile    =    fullfile(outputFolder,
['heq_' imageFiles(i).name]);
            imwrite(img_final, outputFile);

            fprintf('Selesai   preprocessing:   %s\n',
outputFile);
        end
    end
end

disp('Preprocessing selesai');
```

LAMPIRAN 2 Kode Ekstraksi Fitur Uniform Local Binary Pattern (ULBP)

```
clc;
clear;

% Ekstraksi Fitur LBP + Simpan ke Excel per Subset

% Lokasi dataset hasil preprocessing
datasetPath                                          =
'C:\Users\diasp\Favorites\Downloads\Dataset
\Preprocessed_HEQ';

% Folder output khusus untuk fitur LBP
lbpOutputPath    =    fullfile(fileparts(datasetPath),
'LBP_Features');
if ~exist(lbpOutputPath, 'dir')
    mkdir(lbpOutputPath);
end

% Subset data
subsets = {'Data Training', 'Data Testing'};

% Kategori tulisan
categories = {'Teratur', 'Tidak Teratur'};

% Parameter LBP
radius = 1;
numNeighbors = 8;
```

```
for s = 1:length(subsets)
    subset = subsets{s};

    % Inisialisasi cell array untuk hasil subset ini
    subsetData = {};
    rowIdx = 1;

    for c = 1:length(categories)
        category = categories{c};

        % Lokasi folder input
        inputFolder = fullfile(datasetPath, subset,
category);
        if ~isfolder(inputFolder)
            warning('Folder %s tidak ditemukan.
Lewati.', inputFolder);
            continue;
        end

        % Ambil semua file gambar
        imageFiles = dir(fullfile(inputFolder,
'*.jpg'));
        imageFiles = [imageFiles;
dir(fullfile(inputFolder, '*.png'))];
        imageFiles = [imageFiles;
dir(fullfile(inputFolder, '*.bmp'))];

        for i = 1:length(imageFiles)
            % Baca gambar hasil preprocessing
            imgPath = fullfile(inputFolder,
imageFiles(i).name);
            img = imread(imgPath);

            % Pastikan grayscale
            if size(img,3) == 3
                img_gray = rgb2gray(img);
            else
                img_gray = img;
            end

            % Pastikan tipe uint8
            if ~isa(img_gray,'uint8')
                img_gray_uint8 =
im2uint8(mat2gray(img_gray));
            else
                img_gray_uint8 = img_gray;
            end

            % Ekstraksi LBP dengan built-in MATLAB
            feat = extractLBPFeatures(img_gray_uint8,
...
                'Radius', radius, 'NumNeighbors',
numNeighbors, ...
                'Upright', true);

            % Simpan ke cell array
            [~, name, ext] =
fileparts(imageFiles(i).name);
            subsetData{rowIdx,1} = [name ext];      %
file name

            % simpan label sebagai nama kategori
            subsetData{rowIdx,2} = category;        %
```

```
'Teratur' atau 'Tidak Teratur'
            subsetData(rowIdx,3:2+length(feat))     =
num2cell(feat(:)');

            rowIdx = rowIdx + 1;

            fprintf('Ekstraksi  LBP  selesai:  %s\n',
imgPath);
        end
    end

    % Buat header
    if isempty(subsetData)
        warning('Tidak  ada  data  pada  subset  %s',
subset);
        continue;
    end

    nFeat = size(subsetData,2)-2;
    header  =  [{'FileName','Label'},  arrayfun(@(x)
sprintf('F%d',x), 1:nFeat,'UniformOutput',false)];

    % Gabungkan header + data
    outputExcel = [header; subsetData];

    % Simpan Excel sesuai subset
    safeSubsetName = strrep(subset,' ','_');
    outputFile      =      fullfile(lbpOutputPath,
['LBP_Features_' safeSubsetName '.xlsx']);

    writecell(outputExcel, outputFile);

    disp(['===  Fitur  LBP  untuk  ' subset  ' berhasil
disimpan ke: ' outputFile ' ===']);
end

disp('===  Semua  fitur  LBP  selesai  diekstraksi  dan
disimpan ===');
```

LAMPIRAN 3 Kode Klasifikasi Support Vector Machine (SVM)

```
 clc;
clear;

% 1. BACA DATA TRAINING DAN TESTING
fileTraining                                          =
'C:\Users\diasp\Favorites\Downloads\Dataset\LBP_Featu
res\LBP_Features_Data_Training.xlsx';
fileTesting                                           =
'C:\Users\diasp\Favorites\Downloads\Dataset\LBP_Featu
res\LBP_Features_Data_Testing.xlsx';

opts = detectImportOptions(fileTraining);
opts.VariableNamingRule = 'preserve';
dataTrain = readtable(fileTraining, opts);

opts = detectImportOptions(fileTesting);
opts.VariableNamingRule = 'preserve';
dataTest = readtable(fileTesting, opts);

% 2. PISAHKAN FITUR DAN LABEL
```

```
% Kolom 1 = FileName, Kolom 2 = Label
fiturTrain = table2array(dataTrain(:, 3:end));
labelTrain = categorical(string(dataTrain{:,2}));

fiturTest = table2array(dataTest(:, 3:end));
labelTest = categorical(string(dataTest{:,2}));

% Pastikan urutan kelas konsisten
labelTrain          =          reordercats(labelTrain,
{'Teratur','Tidak Teratur'});
labelTest  = reordercats(labelTest,  {'Teratur','Tidak
Teratur'});

% 3. FOLDER OUTPUT HASIL KLASIFIKASI
outputResultPath                                      =
'C:\Users\diasp\Favorites\Downloads\Dataset\SVM_Resul
ts';
if ~exist(outputResultPath,'dir')
    mkdir(outputResultPath);
end

% 4. DEFINISI KERNEL SVM
kernelList = {'linear', 'rbf', 'polynomial'};
polynomialOrder = 3;

for i = 1:length(kernelList)
    kernel = kernelList{i};
    fprintf('\n==========  KERNEL:  %s  ==========\n',
upper(kernel));

    % 5. TEMPLATE SVM
    if strcmp(kernel, 'polynomial')
        template = templateSVM( ...
            'KernelFunction', kernel, ...
            'PolynomialOrder', polynomialOrder, ...
            'Standardize', true);
    else
        template = templateSVM( ...
            'KernelFunction', kernel, ...
            'Standardize', true);
    end

    % 6. LATIH MODEL SVM
    SVMModel = fitcecoc( ...
        fiturTrain, labelTrain, ...
        'Learners', template, ...
        'ClassNames', categories(labelTrain));

    % 7. PREDIKSI DATA TESTING
    labelPrediksi = predict(SVMModel, fiturTest);
    labelPrediksi = categorical(labelPrediksi);

    % 8. HITUNG AKURASI
    akurasi  =  sum(labelPrediksi  ==  labelTest)  /
numel(labelTest) * 100;
    fprintf('Akurasi Klasifikasi SVM (%s): %.2f%%\n',
kernel, akurasi);

    % 9. CONFUSION MATRIX
    confMat  =  confusionmat(labelTest,  labelPrediksi,
...
        'Order', {'Teratur','Tidak Teratur'});
```

```
    figure('Name', ['Confusion Matrix - ', kernel]);
    confusionchart(confMat,         {'Teratur','Tidak
Teratur'});
    title(['Confusion Matrix - ', kernel]);

    % 10. METRIK EVALUASI
    precision = diag(confMat) ./ sum(confMat,1)';
    precision(isnan(precision)) = 0;

    recall = diag(confMat) ./ sum(confMat,2);
    recall(isnan(recall)) = 0;

    f1 = 2 * (precision .* recall) ./ (precision +
recall);
    f1(isnan(f1)) = 0;

    kelas = {'Teratur'; 'Tidak Teratur'};
    T = table(kelas, precision, recall, f1, ...
        'VariableNames',
{'Kelas','Precision','Recall','F1_Score'});
    disp('=== Classification Report ===');
    disp(T);

    % 11. SIMPAN HASIL KLASIFIKASI KE EXCEL
    hasilKlasifikasi = table( ...
        dataTest{:,1}, ...       % FileName
        labelTest, ...           % Label aktual
        labelPrediksi, ...       % Label prediksi
        'VariableNames',
{'FileName','Label_Aktual','Label_Prediksi'});

    outputFile = fullfile(outputResultPath, ...
        ['Hasil_Klasifikasi_SVM_' kernel '.xlsx']);

    writetable(hasilKlasifikasi, outputFile);

    fprintf('Hasil  klasifikasi  kernel  %s  disimpan
ke:\n%s\n', kernel, outputFile);
end

disp('=== Proses klasifikasi SVM selesai ===');
```

LAMPIRAN 4 Kode Klasifikasi Support Vector Machine (SVM) dengan tuning parameter (Grid Search)

```
clc;
clear;


%% ================================
% 1. BACA DATA
% ================================
fileTraining =
'C:\Users\diasp\Favorites\Downloads\Dataset\LBP_Featu
res\LBP_Features_Data_Training.xlsx';
fileTesting  =
```

```
'C:\Users\diasp\Favorites\Downloads\Dataset\LBP_Featu
res\LBP_Features_Data_Testing.xlsx';

opts = detectImportOptions(fileTraining);
opts.VariableNamingRule = 'preserve';
dataTrain = readtable(fileTraining, opts);

opts = detectImportOptions(fileTesting);
opts.VariableNamingRule = 'preserve';
dataTest = readtable(fileTesting, opts);

fiturTrain = table2array(dataTrain(:,3:end));
labelTrain = categorical(string(dataTrain{:,2}));

fiturTest  = table2array(dataTest(:,3:end));
labelTest  = categorical(string(dataTest{:,2}));

labelTrain = reordercats(labelTrain,
{'Teratur','Tidak Teratur'});
labelTest  = reordercats(labelTest,
{'Teratur','Tidak Teratur'});

%% ================================
% 2. GRID SEARCH PARAMETER SVM
% ================================
C_list      = [0.1 1 10];
gamma_list  = [0.01 0.1 1];
degree_list = [2 3 4];

bestAcc = struct('linear',0,'rbf',0,'poly',0);
bestParam = struct;

fprintf('\n=== GRID SEARCH SVM ===\n');

%% ----- LINEAR -----
for C = C_list
    template = templateSVM('KernelFunction','linear',
...
        'BoxConstraint',C,'Standardize',true);

    model =
fitcecoc(fiturTrain,labelTrain,'Learners',template);
```

```
    pred  = predict(model,fiturTest);
    acc   = mean(pred==labelTest)*100;

    fprintf('[Linear | C=%.2f] Akurasi:
%.2f%%\n',C,acc);

    if acc > bestAcc.linear
        bestAcc.linear = acc;
        bestParam.linear = C;
    end
end

%% ----- RBF -----
for C = C_list
    for g = gamma_list
        template =
templateSVM('KernelFunction','rbf', ...
            'BoxConstraint',C, ...
            'KernelScale',1/sqrt(2*g), ...
            'Standardize',true);

        model =
fitcecoc(fiturTrain,labelTrain,'Learners',template);
        pred  = predict(model,fiturTest);
        acc   = mean(pred==labelTest)*100;

        fprintf('[RBF | C=%.2f | Gamma=%.2f] Akurasi:
%.2f%%\n',C,g,acc);

        if acc > bestAcc.rbf
            bestAcc.rbf = acc;
            bestParam.rbf = [C g];
        end
    end
end

%% ----- POLYNOMIAL -----
for C = C_list
    for d = degree_list
        template =
templateSVM('KernelFunction','polynomial', ...
            'PolynomialOrder',d, ...
```

```
            'BoxConstraint',C,'Standardize',true);

        model =
fitcecoc(fiturTrain,labelTrain,'Learners',template);
        pred  = predict(model,fiturTest);
        acc   = mean(pred==labelTest)*100;

        fprintf('[Poly | C=%.2f | Degree=%d] Akurasi:
%.2f%%\n',C,d,acc);

        if acc > bestAcc.poly
            bestAcc.poly = acc;
            bestParam.poly = [C d];
        end
    end
end

%% ================================
% 3. EVALUASI FINAL (BEST PARAMETER)
% ================================
models = {
    'Linear', templateSVM('KernelFunction','linear',
...

'BoxConstraint',bestParam.linear,'Standardize',true);

    'RBF', templateSVM('KernelFunction','rbf', ...
           'BoxConstraint',bestParam.rbf(1), ...
           'KernelScale',1/sqrt(2*bestParam.rbf(2)),
...
           'Standardize',true);

    'Polynomial',
templateSVM('KernelFunction','polynomial', ...
                'PolynomialOrder',bestParam.poly(2),
...
                'BoxConstraint',bestParam.poly(1),
...
                'Standardize',true);
};

outputResultPath =
```

```
'C:\Users\diasp\Favorites\Downloads\Dataset\SVM_Resul
ts';
if ~exist(outputResultPath,'dir')
    mkdir(outputResultPath);
end

for i=1:size(models,1)
    kernelName = models{i,1};
    template   = models{i,2};

    fprintf('\n===== %s (BEST PARAMETER)
=====\n',kernelName);

    model =
fitcecoc(fiturTrain,labelTrain,'Learners',template);
    pred  = categorical(predict(model,fiturTest));

    acc = mean(pred==labelTest)*100;
    fprintf('Akurasi: %.2f%%\n',acc);

    confMat =
confusionmat(labelTest,pred,'Order',{'Teratur','Tidak
Teratur'});
    figure('Name',['Confusion Matrix -
',kernelName]);
    confusionchart(confMat,{'Teratur','Tidak
Teratur'});

    precision = diag(confMat)./sum(confMat,1)';
    recall    = diag(confMat)./sum(confMat,2);
    f1        =
2*(precision.*recall)./(precision+recall);

    precision(isnan(precision))=0;
    recall(isnan(recall))=0;
    f1(isnan(f1))=0;

    disp(table({'Teratur';'Tidak
Teratur'},precision,recall,f1, ...

'VariableNames',{'Kelas','Precision','Recall','F1_Sco
re'}));
```

```
    hasil = table(dataTest{:,1},labelTest,pred, ...
'VariableNames',{'FileName','Label_Aktual','Label_Pre
diksi'});

    writetable(hasil,fullfile(outputResultPath, ...
['Hasil_Klasifikasi_SVM_',kernelName,'_Best.xlsx']));
end

disp('=== Proses selesai ===');
```