Git: https://github.com/Junx0924/Features-Operators-for-PolSAR-Images

Handcrafted features:

1) **Dependencies**:
   MPEG-7
   T-SNE
   HDF5 (installed by Anaconda)

2) **How-to-use**:

```
main.exe <ratFolder> <labelFolder> <Hdf5File> <featureName> <filterSize> <patchSize>
```

   <filterSize> choose from: 0,5,7,9,11

   <featureName> choose from: mp, decomp, color, texture, polstatistic, ctelements

   mp stands for: morphological profile features

   decomp stands for: target decomposition features

   color stands for: MPEG-7 CSD,DCD features

   texture stands for: GLCM and LBP features

   polstatistic stands for: the statistic of polsar parameters

   ctelements stands for: the 6 upcorner elements of covariance and coherence matrix

3) **Class descriptions**:
   1. Data(const std::string& RATfileFolder, const std::string& labelFolder)
      Description: read PolSAR data from rat files and load masks
      Public members:

```
std::vector<cv::Mat> data;
cv::Mat LabelMap;
std::map<unsigned char, std::string>classNames;
```

      Usage:

```
std::string ratfolder, labelfolder;
ratfolder = "E:\\Oberpfaffenhofen\\sar-data";
labelfolder = "E:\\Oberpfaffenhofen\\label";
Data* ob = new Data(ratfolder, labelfolder);
```

   2. FeatureProcess(const std::string& hdf5_fileName)
      Description: calculate features from PolSAR data, classify and visualize features

Usage:

```
string hdf5_fileName = "E:\\polstatistic.h5";
FeatureProcess* f = new FeatureProcess(hdf5_fileName );
 // calculate features from Data, and store to hdf5;
int filterSize =0, patchSize = 10, batchSize = 5000 ;
string feature_name = “polstatistic”;
f->setParam(feature_name , filterSize , patchSize , batchSize );
f->caculFeatures(ob->data, ob->LabelMap, ob->classNames);

//classify features and store  the class results to hdf5
int K = 10, training_Percent = 80;
f->classifyFeaturesML("opencvKNN", training_Percent, K );

//Generate colormap of class results, calculate the overall accuracy of each class;
f->generateColorMap("opencvKNN");

// Generate feature map for each dimension of feature group
f->generateFeatureMap();

// reduce the dimension of selected batch, dump the data to txt for plotting
int batchID = 1;
f->featureDimReduction(batchID );
```

Modern Features

A. Autoencoder APIs

a. **Autoencoder(int inputDim, int hiddenDim, double learningRate, double momentum)**
Description: This function initialises the values of various variables used
Input 1 : Input dimension of the AE
Input 2 : Hidden/encoder dimension of AE
Input 3 : Learning Rate
Input 4 : Momentum
Usage example:

```
Autoencoder *aeEncoder = new Autoencoder(9, 5, 0.01, 0.9);
```

b. **void feedforward(vector<float>& m_hiddenValues, vector<float>& m_outputValues)**
Description: Feedforward implementation of AE as per the formula (W*x + b)
Output 1 : The hidden/encoder layer values
Output 2 : The output layer value calculated from weight and bias parameters
Usage example:

```
vector<float> m_hiddenValues;
vector<float> m_outputValues;
```

```
feedforward(m_hiddenValues, m_outputValues);
```

c. void backpropagate(vector<float>& m_hiddenValues, vector<float>& m_outputValues)

Description: Feedforward implementation of AE as per the formula (W*x + b)
Input 1 : Hidden value calculated in feedforward step
Input 2 : Output value calculated in feedforward step
Output 1 : Optimised weight parameter
Output 2 : Optimised bias parameter
Usage example:

```
vector<float> m_hiddenValues;
vector<float> m_outputValues;
backpropagate(m_hiddenValues, m_outputValues);
```

d. void train(vector<float>& data, int& cnt, int& epoch)

Description: This function is used to extract feature from the input feature vector from each pixel in the image
Input 1 : Input feature vector
Input 2 : The current epoch count
Input 3 : Total number of epochs
Output 1 : Learned feature vector
Output 2 : The output vector
Usage example:

```
int cnt = 2
aeEncoder->train(coherenceVec[cnt], cnt, 100);
```

e. void InitializeWts()

Description: This function ensures that weight parameters are initialised to same random value for every pixel and intermediate weight variables are reset
Usage example:

```
aeEncoder->InitializeWts();
```

f. void InitializeBias()

Description: This function ensures that bias parameters are initialised to same random value for every pixel and intermediate bias variables are reset
Usage example:

```
eEncoder->InitializeBias();
```

g. vector<float> random(size_t elementSize)

Description: This function generates random values to initialise weight and bias
Input 1 : The dimension of the vector to be initialised randomly
Output : Vector initialized with random values
Usage example:

```
m_hiddenDimension = 5
```

m_hiddenBiasInit = this->random(m_hiddenDimension);

h. **float sigmoid(float value)**
Description: Implementation of activation function - Sigmoid
Input 1 : Input data
Output 1 : Sigmoid value of input value
Usage example:
float result= sigmoid(total);

i. **float sigmoidDerivation(float value)**
Description: Sigmoid derivative used in backpropagation
Input 1 : Input data
Output 1 : Sigmoid derivative of input value
Usage example:
float result = sigmoidDerivation(total);

j. **void AutoencoderUserMenu(vector<vector<float>>& coherenceVec, Data& data)**
Description: User menu for various operations related to autoencoder
Input 1 : Coherence vector calculated for entire image
Input 2 : Image data
Usage example:
Autoencoder ae;
ae.AutoencoderUserMenu(coherenceVec, data);

k. **void CalculateClassification(bool isAE, Data& data, Utils& utils, int k, Autoencoder& aeEncoder, KNN& knn, Performance& perform)**
Description: Function to prepare the calulated coherence matrix of entire image for a test/train
data split that can be used for classification and calculation of accuracy
Input 1 : is the autoencoder vanilla or multilayer?
Input 2 : Input data
Input 3 : Hyperparameter k
Output 1 : Classification result in the form of a csv and image, performance metrics
Usage example:
CalculateClassification(true, data, utils, k, *aeEncoder, knn, perform);

l. **float CalculateDifferenceMatrix(Mat& logInMatrix, Mat& logReconsMatrix)**
Description: Calculate the difference between input and reconstructed values
Input 1 : Log of coherency matrix of input
Input 2 : Log of coherency matrix of reconstructed data
Output 1 : Difference
Usage example:
Mat logInMatrix, logReconsMatrix;
float forbNorm = CalculateDifferenceMatrix(logInMatrix, logReconsMatrix);

m. **void CalculateCoherencyMatrix(Mat& d, int row, Mat& coherencyMat)**
   Description: Calculate the coherency matrix from the feature vector
   Input 1 : Feature vector
   Input 2 : Feature vector number
   Output 1 : Coherency Matrix from feature vector
   Usage example:
   Mat coherencyInMat, inputData;
   int row = 2
   CalculateCoherencyMatrix(inputData, row, coherencyInMat)

n. **void CalculateLogMatrix(Mat& coherencyMat, Mat& logMatrix)**
   Description: Calculate the log of the coherency matrix in order to calculate the difference between input and reconstructed values
   Input 1 : Coherency matrix
   Output 1 : Log of coherency matrix
   Usage example:
   Mat coherencyInMat, logInMatrix;
   CalculateLogMatrix(coherencyInMat, logInMatrix)

B. KNN Classifier and Performance APIs

a. **double Euclidean(Mat& testVal, Mat& trainVal)**
   Description: This function is used to calculate the Euclidean distance between the training and test samples
   Input 1 : test points
   Input 2 : training points
   Output : Calculated euclidean distance
   Usage example:
   int i = 2;
   float dist = Euclidean(testVal[i], trainVal[j])

b. **void KNNTest(vector<Mat>& trainVal, vector<unsigned char>& trainLabels, vector<Mat>& testVal, vector<unsigned char>& testLabels, int k, vector<unsigned char>& classResult)**
   Description: Classify test points using KNN Classifier
   Input 1 : Training values
   Input 2 : Training labels
   Input 3 : Test values
   Input 4 : Test labels
   Input 5 : Hyperparameter k
   Output : Classification result
   Usage example:

```
vector<Mat> trainVal, testVal;
vector<unsigned char> trainLabel, testLabel, classResult;
int k = 1;
KNNTest(trainVal, trainLabel, testVal, testLabel, k, classResult)
```

c. **void OpenCVKNNTest(vector<Mat>& trainVal, vector<unsigned char>& trainLabels, vector<Mat>& testVal, int k, vector<unsigned char>& classResult)**

Description: Classify test points using KNN Classifier by OpenCV

Input 1 : Training values

Input 2 : Training labels

Input 3 : Test values

Input 4 : Test labels

Input 5 : Hyperparameter k

Output : Classification result

Usage example:

```
vector<Mat> trainVal, testVal;
vector<unsigned char> trainLabel, classResult;
int k = 1;
OpenCVKNNTest(trainVal, trainLabel, testVal, k, classResult)
```

d. **unsigned char Classify(vector<pair<float, unsigned char>>& distVec, int k)**

Description : This function counts the number of classes in k neighborhood
Based on which class has the highest count, appropriate class is returned

Input 1 : The sorted distance between test and training points

Input 2 : Hyperparameter k

Output : Classified point value

Usage example:

```
vector<pair<float, unsigned char>> distVec;
int  k = 1;
unsigned char classVal = Classify(distVec, k);
```

e. **double calculatePredictionAccuracy(vector<unsigned char>& classResult, vector<unsigned char>& testLabels)**

Description: This function is used to calculate the OA for each class as well as for the whole image

Input 1 : Classification result/labels

Input 2 : Original test point labels

Output : Overall accuracy of the image. Classwise accuracy is calculated as well

Usage example:

```
vector<unsigned char> testLabel, classResult;
float intAccuracy = calculatePredictionAccuracy(classResult, testLabel);
```

C. Feature Calculation APIs

a. **void getLexiBasis(const Mat& hh, const Mat& vv, const Mat& hv, vector<Mat>& lexi)**
Description: Calculate Lexi decomposition
Input 1 : hh matrix for all pixels in the entire image
Input 2 : vv matrix for all pixels in the entire image
Input 3 : hv matrix for all pixels in the entire image
Output : Lexi decomposition vector for all pixels in the entire image
Usage example:

```
Mat hh, vv, hv;
vector<Mat> lexi;
getLexiBasis(hh, vv, hv, lexi);
```

b. **void getPauliBasis(const Mat& hh, const Mat& vv, const Mat& hv, vector<Mat>& pauli)**
Description: Calculate Pauli decomposition
Input 1 : hh matrix for all pixels in the entire image
Input 2 : vv matrix for all pixels in the entire image
Input 3 : hv matrix for all pixels in the entire image
Output : Pauli decomposition vector for all pixels in the entire image
Usage example:

```
Mat hh, vv, hv;
vector<Mat> pauli;
getLexiBasis(hh, vv, hv, pauli);
```

c. **vector<float> logTransform(vector<float>& in)**
Description: Calculate log transform of input matrix
Input 1 : Input matrix
Output : log of the input matrix
Usage example:

```
vector<float> temp, e;
temp = logTransform(e)
```

d. **Mat getComplexAmpl(const Mat& in)**
Description: Calculate amplitude of complex matrix
Input 1 : Input complex matrix
Output : Amplitude of complex matrix calculated as sqrt(re*re + im*im)
Usage example:

```
Mat out, in;
out = getComplexAmpl(in);
```

e. **void vec2mat(const vector<Mat>& basis, int winSize, vector<Mat>& mat)**
Description: Calculate coherence matrix from pauli decomposition
Input 1 : Pauli decomposition vector

Input 2 : Blurring filter window size
Output : Coherency Matrix for entire image
Usage example:

```
vector<Mat> pauli, coherencyMat;
int winSize  = 3;
vec2mat(pauli, winSize, coherencyMat)
```

f. **void GetCoherencyFeatures(Data data, vector<vector<float>>& result, vector<unsigned char>& classValue)**
Description: This function calculates coherency matrix for entire image
Input 1   : Input Data
Output 1 : Coherency vector for entire image
Output 2 : Corresponding label vector for entire image
Usage example:

```
Data data;
vector<vector<float>> result;
vector<unsigned char> labelMap;
GetCoherencyFeatures(data, result, labelMap);
```

g. **void GetCoherencyMat(vector<Mat>& pauli, int winSize, vector<Mat>& coherencyMat)**
Description: Calculate coherence matrix from pauli decomposition
Input 1 : Pauli decomposition vector
Input 2 : Blurring filter window size
Output : Coherency Matrix
Usage example:

```
vector<Mat> pauli, coherencyMat;
int winSize  = 3;
GetCoherencyMat(pauli, winSize, coherencyMat)
```

D. Visualisation

a. **void ContrastCorrection(vector<vector<float>>& featureVector, int cnt, Mat& outPut)**
Description: This functions converts vector to matrix and performs contrast correction by applying gray scale stretching and histogram equalisation
Input 1 : The calculated feature vector
Input 2 : The count number of the current feature vector
Output : Contast corrected matrix
Usage example:

```
vector<vector<float>>& m_featureVector;
int cnt = 5;
Mat featureMat;
ContrastCorrection(m_featureVector, cnt, featureMat);
```

b. **void GenerateFeatureMap(vector<vector<float>>& m_featureVector, string& imagePrefix)**
Description: This function generates the visualisation of the calculated feature map after contrast correction
Input 1 : The calculated feature vector
Input 2 : Generated colormap image name
Usage example:
vector<vector<float>>& m_featureVector;
string imagePrefix = "FeatureVisualisation";
GenerateFeatureMap(m_featureVector, imagePrefix);

E. Helper/Utility APIs

a. **map<string, Vec3f> loadLabelsMetadata()**
Description: Using this function creates a map of the colors and the labels they correspond to. To be used with visualization
Usage example:
map<string, Vec3f> colors = loadLabelsMetadata();

b. **Mat_<Vec3f> visualiseLabels(Mat &image, string& imageName)**
Description: Using this function to assign colors to maps (label and classified)
Input 1 : Image matrix
Input 2 : Name of the generated image
Output : Generated image stored as png
Usage example:
Mat classifiedImage;
string fileName = "Generated.png";
visualiseLabels(classifiedImage, fileName)

c. **void generateLabelMap(vector<Mat>& label, Mat& labelMap)**
Description: This function creates a single label map from a list of various label classes. This map serves as points of reference when trying to classify patches
Input 1 : label images
Output : generated label map stored in a distance csv
Usage example:
Mat labelMap;
generateLabelMap(data.labelImages, labelMap);

d. **vector<pair<vector<Point2i>, uint>> GetPatchPoints(int patchIdx, Data& data)**
Description: This function splits entire image into 5 patches and stores the data in these patches which will be used as training and testing sets
Input 1 : Patch index

Input 2 : Input image data points for all classes
Output 1 : Point and label pair for each patch for the entire image
Usage example:

```
Data data;
int patchIdx = 2;
vector<pair<vector<Point2i>, uint>> patchPoint = GetPatchPoints(patchIdx, data);
```

e. **void DivideTrainTestData(Data& data, int fold, int patchIdx)**
Description: This function uses the split data patches and uses 20% points as test samples and 80% points as training samples. The test samples is shuffled depending on the number of the fold and cross validation is achieved. The number of test and training samples are chosen such that the class points are balanced.
Input 1 : Inout image data
Input 2 : Fold number
Input 3 : Patch index
Usage example:

```
Data data;
int fold  = 2;
int patchIdx = 3;
DivideTrainTestData(data, fold, patchIdx);
```

f. **void DivideTrainTestData(Data& data, int fold, vector<pair<vector<Point2i>, uint>> patchPoint)**
Description: This function uses the split data patches and uses 20% points as test samples and 80% points as training samples. The test samples is shuffled depending on the number of the fold and cross validation is achieved. The number of test and training samples are chosen such that the class points are balanced.
Input 1 : Inout image data
Input 2 : Fold number
Input 3 : Patch point determined while splitting data
Usage example:

```
Data data;
int fold = 2;
vector<pair<vector<Point2i>, uint>> patchPoint;
DivideTrainTestData(data, fold, patchPoint)
```

g. **void WriteCoherenceMatValues(vector<pair<vector<float>, unsigned char>>& imgData, string& fileName, bool isApp)**
Description: Use this function to store calculated values in csv files that can be used later for data analysis
Input 1 : input vector
Input 2 : name of the csv file
Input 3 : is data to be appended to an existing file?

Usage example:

```
vector<pair<vector<float>, unsigned char>> imgData;
fileName = “Feature.csv”;
WriteCoherenceMatValues(imgData, fileName, false);
```

h. **void WriteCoherenceMatValues(vector<vector<float>>& featureVector, string& fileName, bool isApp)**

Description: Use this function to store calculated values in csv files
that can be used later for data analysis
Input 1 : input vector and label pair
Input 2 : name of the csv file
Input 3 : is data to be appended to an existing file?

Usage example:

```
vector<vector<float>> m_featureVector;
outFile = “Feature.csv”;
WriteCoherenceMatValues(m_featureVector, outFile, false);
```

i. **void ConvertToCoherenceVector(vector<vector<float>>& result, vector<vector<float>>& coherenceVec)**

Description: This function generates a 9x1 coherence vector for each pixel
Input 1 : the coherence vector values of entire image which is a vector of 9 dimension and each vector has (6640x1390) values
Output : coherency matrix of each pixel for the entire image. 9x1 vector for entire image consisting of 6640x1390 pixels

Usage example:

```
vector<vector<float>> result, coherenceVec;
ConvertToCoherenceVector(result, coherenceVec);
```

F. Data loader APIs

a. **void loadData(string folder)**

Description : This function loads the PolSAR file from the respective directory
Input 1 : PolSAR data folder path

Usage example:

```
data.loadData(argv[1]);
```

b. **Mat loadImage(string fname)**

Description: Function to load Oberpfaffenhofen PolSAR image file
Input : image file name

Usage example:

```
Mat img =  loadImage(“InputImage.png”);
```

c. **void loadLabels(const string &folderPath, vector<string>& labelNames, vector<Mat>& labelImages, vector<vector<Point2i>>& numOfPoints)**
   Description : The function loads the labels and counts the number of
   points for each label class
   Input 1 : Label folder path
   Output 1 : Label names
   Output 2 : Label images
   Output 3 : Number of non zero points for each label class found in the
   label directory
   Usage example:
   loadLabels(argv[3], data.labelNames, data.labelImages, data.numOfPoints);

d. **void ReadClassLabels(string labelPath, vector<string> &labelNames, vector<Mat> &labelImages)**
   Description : This function reads all the label images from the
   label directory
   Input 1 : Label files directory path
   Input 2 : Names of the label files
   Output : Label images found in the label directory
   Usage example:
   ReadClassLabels(argv[3], data.labelNames, data.labelImages);