BBM465: Information Security Laboratory

Programming Assignment 4

Topic: Creating A Threat Intelligence For Anti Phishing

Teachers:
- Ahmet Selman Bozkır
- Ali Baran Taşdemir

Students:
- Mert Tazeoğlu (21946606)
- Mehmet Emin Yıldız (21527602)

Table of Contents.....Page

Problem Definition

In this assignment, we implemented a threat intelligence for anti phishing with using image processing and machine learning algorithms.

Solution Scheme

Step-1: Check whether missing csv files exists.

Step-2: If any csv file is missing, with using phish dataset make required image processing operations (my app supports FCTH, CEDD, SCD, SIFT and HOG) calculate vectors and prepare csv files.

Step-3: If mode is trainval, then for each of them apply all of implemented machine learning algorithms. (my app supports 3 different machine learning algorithms: Random Forest, SVM, KNN)

Important Testing & Execution Details

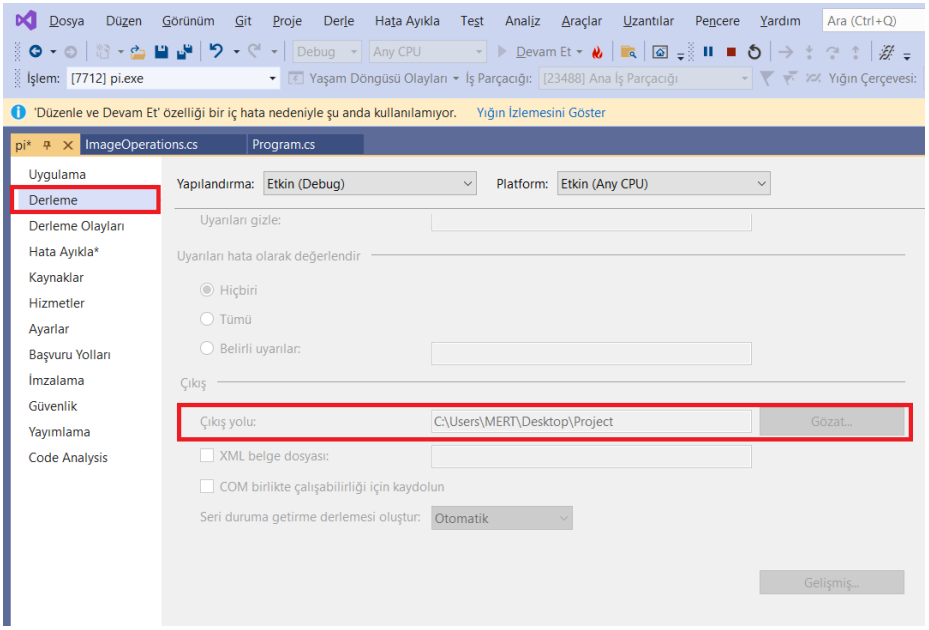
System Requirements:
- .Net Framework v4.6.1
- Visual Studio 2019 Community Edition

File Hierarchy Requirements:
The folder hierachy must be like that:

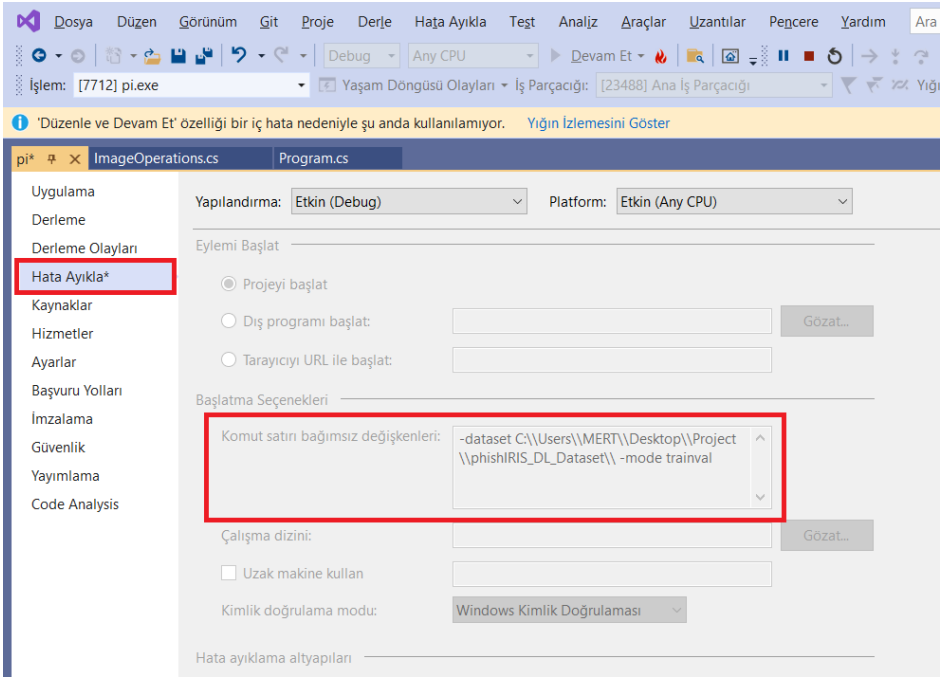
/Project (Main folder, name doesn't matter. In that folder .ll files, .exe files and dataset should be deployed.)
/Project/ phishIRIS_DL_Dataset/ (This folder should contain 2 folders, 'phishIRIS_DL_Dataset' which contains train and val folders; and 'pre-computed' folder which contains .csv files.)

I send my testing folder (which is project folder) as an example. Program works without any problems under these conditions.

In that hierarchy, ‘Project’ folder is execution point. You can set execution point in that menu in VS-2019:



Also (i suggest that) without using CMD; you can easily set command line arguments in VS-19 like that:



Execution of SVM and KNN takes only a few seconds but random forest takes minutes. On the other hand, .csv file creation with global image descriptors take only a few minutes but local descriptors (especially SIFT) takes approximately 2 hours. In ML side, execution of HOG takes hours.

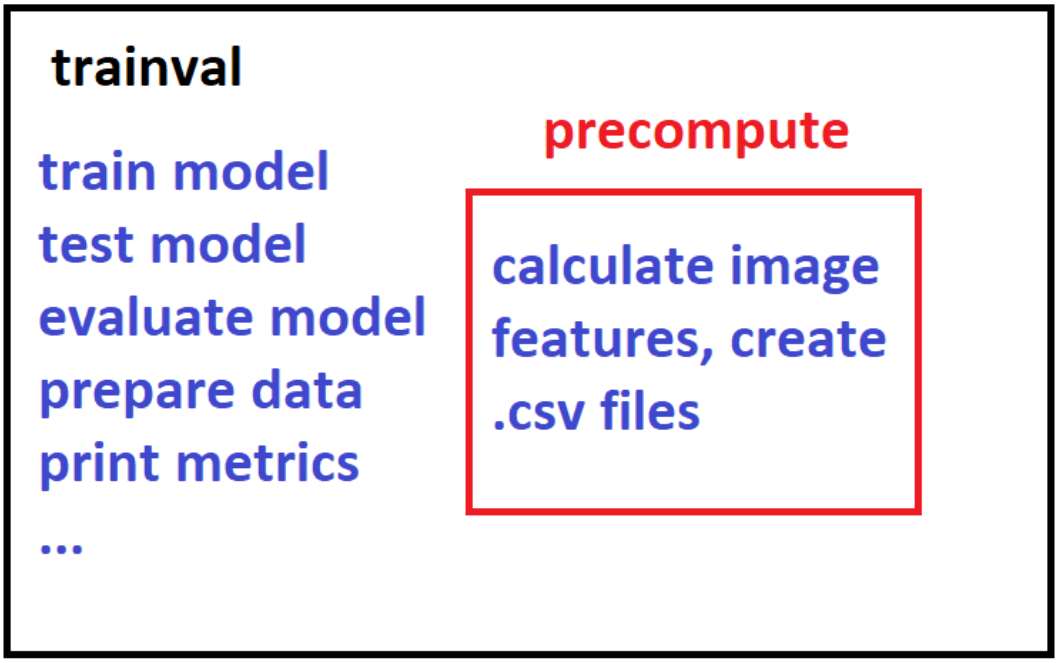
Important Implementation Details

1. Command Line Arguments

Program takes 4 arguments. Only 2 of them are important. 2nd one should be a file path (it can be relative or absolute) which meets conditions in 'Important Testing & Execution Details' part of this report. That path is used for all operations. Other important argument is 4th one, which is about execution mode.

2. Execution Modes

Program has 2 different execution modes, one of them is precompute other one is trainval. Actually precompute is a subset of trainval.



In both of modes at the first program checks whether a .csv file is missing, prints missing files and creates missing .csv files. (for saving time) Then if mode is trainval, program starts to make machine learning processes.

3. Image Descriptors

- Fuzzy Color and Texture Histogram (FCTH) -> This decsriptor extracts 192 features for each image.

- Color and Edge Directivity Descriptor (CEDD) -> This descriptor extracts 144 features for each image.

- Scalable Color Descriptor (SCD) -> This descriptor extracts 256 features for each image.

- Scale Invariant Feature Transform (SIFT) -> This descriptor creates a feature matrix for each image. Then i take each of 128 columns average and extract a feature with that way.

- Histogram of Oriented Gradients (HOG) -> This descriptor extracts 3760 features for each 64x128 image. :)

Important Notes:

a- .NET framework that we used doesn't support SURF extractors, but their implementation is %90 same with SIFT extractor.

b- In SIFT extractor, i take average of each feature in matrix that created by SIFT descriptor. Unfortunately i didn't managed to use a histogram. Therefore machine learning accuracies of SIFT is a bit bad.

4. Machine Learning Algorithms

In my program, 3 machine learning algorithms are supported:

4.1 - Random Forest (From Accord.ML)
- NumberOfTrees parameter is 10, since execution takes long and my computation power is limited. In real world, suggested range is 64-128.
- SampleRatio parameter is 1 (in other word %100), since we have different datasets for training and validation.

4.2 - Support Vector Machine (From Emgu.CV)
- Gamma is 0.005
- C is 100
- Mode is Linear SVM

4.3 - K-Nearest Neighbour (From Emgu.CV)
- K is 5 (it gave best results for me, and it is suggested that selecting a k odd)

In order to provide more reliable results, we train with images in train folder and test with images in val folder.

For testing stuff, i implemented a evaluator model which takes actual and predicted (label, class) values of each image in test set. In that model, with using these inputs i create a 15x15 confusion matrix. And with using it i calculate (and print) accuracy, true positive rate (recall), f1 score and false positive rate.

Also i implemented a .csv normalizer (which scales features between 0 and 1 in order to make each of features has similar effects on results) but somehow it reduced accuracies. Therefore i didn’t use it but you can find source code of it.

Outputs for Trainval Mode

Checking whether missing .csv file exist...
Luckily no missing .csv file found.

Training with precomputed_FCTH_train.csv with 1314 samples.
Done in 162,086 seconds.
Testing with precomputed_FCTH_val.csv with 1540 samples.
Random Forest | Accuracy: 0,587 | TPR: 0,408 | FPR: 0,153 | F1-Score: 0,492
SVM | Accuracy: 0,622 | TPR: 0,381 | FPR: 0,145 | F1-Score: 0,446
KNN | Accuracy: 0,523 | TPR: 0,365 | FPR: 0,155 | F1-Score: 0,447

Training with precomputed_CEDD_train.csv with 1314 samples.
Done in 147,28 seconds.
Testing with precomputed_CEDD_val.csv with 1540 samples.
Random Forest | Accuracy: 0,611 | TPR: 0,439 | FPR: 0,146 | F1-Score: 0,521
SVM | Accuracy: 0,617 | TPR: 0,409 | FPR: 0,152 | F1-Score: 0,482
KNN | Accuracy: 0,542 | TPR: 0,369 | FPR: 0,155 | F1-Score: 0,456

Training with precomputed_SCD_train.csv with 1314 samples.
Done in 573,025 seconds.
Testing with precomputed_SCD_val.csv with 1540 samples.
Random Forest | Accuracy: 0,671 | TPR: 0,494 | FPR: 0,119 | F1-Score: 0,587
SVM | Accuracy: 0,75 | TPR: 0,591 | FPR: 0,092 | F1-Score: 0,681
KNN | Accuracy: 0,651 | TPR: 0,485 | FPR: 0,124 | F1-Score: 0,568

Training with precomputed_SIFT_train.csv with 1314 samples.
Done in 605,384 seconds.

Testing with precomputed_SIFT_val.csv with 1540 samples.
Random Forest | Accuracy: 0,453 | TPR: 0,29 | FPR: 0,207 | F1-Score: 0,364
SVM | Accuracy: 0,453 | TPR: 0,275 | FPR: 0,248 | F1-Score: 0,324
KNN | Accuracy: 0,418 | TPR: 0,256 | FPR: 0,219 | F1-Score: 0,315

Training with precomputed_HOG_train.csv with 1314 samples.
(it took so long and i must submit :( )