BBM465 Information Security Laboratory
Experiment 4

Subject: Creating A Threat Intelligence - Case Study: Anti-Phishing
Due Date: 3/1/2023 - 23:59

1 Abstract

This assignment will give you the opportunity to create a threat-intelligence module regarding anti-phishing via visual screenshot analysis. It will sharpen your image representing and machine learning skills along with production of a real-world cyber-security mechanism.

2 Aim and Scope

Phishing, a continuously growing cyber threat, aims to obtain innocent users' credentials by deceiving them via presenting fake web pages which mimic their legitimate targets. To date, various attempts have been carried out in order to detect phishing pages. In this assignment, you will treat the problem of phishing web page identification as an image classification task and propose a machine learning augmented vision-based approach which extracts and classifies compact visual features from web page screenshots.

For this purpose, you will test and evaluate several visual descriptors in the problem domain. A visual descriptor is a vector-based discriminative representation of an image or an image patch. If the patch is the entire image itself, the descriptor functions as *global* image features. On the other hand, extraction of features (i.e. vectors) around some key points makes them referred to *local* image features. A global image descriptor (e.g. Scalable Color Descriptor, Color Layout Descriptor, HOG) produces one n-dimensional vector per image whereas local image features (e.g. SIFT, SURF, Daisy) are a set of feature vectors produced per single image.

Global image descriptors present an easy-to-use setup because a single image is represented with just one vector. Note that, in the local feature extraction

regime, due to the *one image-to-n vector nature*, an image is represented via *Bag-of-Visual-Words* approach involving the following stages: (a) accumulating of all local feature vectors that were collected from a set of images; (b) performing clustering (e.g. K-means algorithm) - *quantization* - to obtain cluster centroids (i.e. visual words); (c) representing the image via mapping all its local feature vectors to the nearest cluster centroid - *pooling* - which finally leads to (d) building a visual word based histogram as the final image representation (See Fig 1.) Thus, local image feature-based analysis involves quantization and pooling stages to produce the final image representation while there is no such need for global image descriptors. Moreover, being a matter of research, the selection of the parameter k has crucial importance in obtaining better classification accuracy. Likewise, some of the visual features require a varying number of parameters such as *patch size* in HOG (Histogram of Oriented Gradients) image features. Thus, the image resolution should be the same for all images. Nonetheless, most of the global image descriptors such as SCD (Scalable Color Descriptor), CLD (Color Layout Descriptor), CEDD (Color Edge Directivity Descriptor), FCTH (Fuzzy Color Texture Histogram), and some local image descriptors such as SIFT (Scale Invariant Feature Transform) and SURF (Speeded-up Robust Features) are agnostic to image resolution.

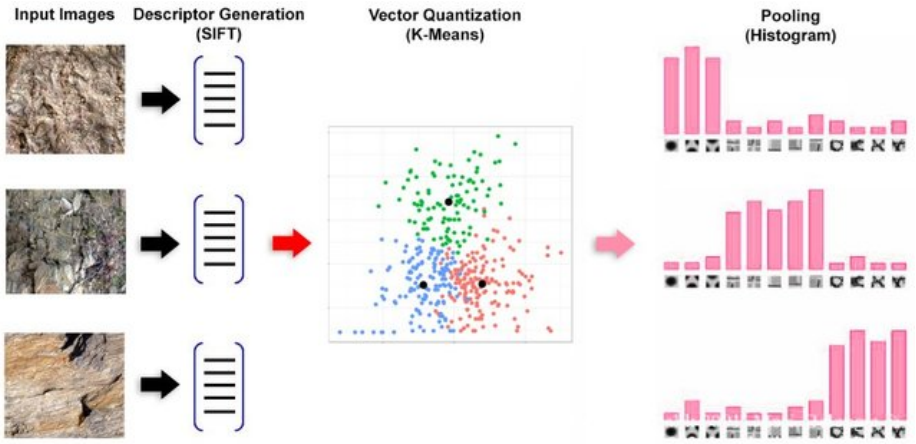

Figure 1: The flowchart of Bag-of-Visual-Words based representation

In this context, your task will be **building a phishing web page brand classifier** by using **three** feature descriptors of your choice among the allowed ones along with **leveraging two machine learning methods** (Linear SVM, Random Forest). **Note that, the ML algorithms must be used with their default parameters** and you can add one more ML method of your choice if you desire, but it is not mandatory!

For the dataset requirement, the Phish-Iris Dataset which is publicly avail-

able served in https://web.cs.hacettepe.edu.tr/~selman/phish-iris-dataset/ and was proposed in the study entitled "Phish-IRIS: A New Approach for Vision Based Brand Prediction of Phishing Web Pages via Compact Visual Descriptors" will be used. The dataset involves 1313 training and 1539 testing samples distributed among 14 (13 brands such as DHL, Facebook, Yahoo) + 1 legitimate group named "other") classes.

3 Design Considerations

- The application you are asked to code will be written in C#.NET 4.5 framework by using Visual Studio .NET Community Edition which is freely available.
- The allowed feature descriptors are SCD, CEDD, FCTH, SIFT, SURF, and HOG. If you prefer to use HOG, please use a patch size of 128 pixels. For local image descriptors, K should be 400. It is important to resize images to 1024x1024 if you prefer HOG.
- The ML algorithms you can use are Linear Support Vector Machine and Random Forest. You can add one more algorithm if you wish (+5 points) Your classification scheme must be multi-class classification since you have more than 2 classes (i.e. **do not confuse with a binary classification which is for bi-class problems**)
- Neither the image feature extraction nor the ML algorithms will be implemented by yours. You can freely use "Compact Composite Descriptors" presented in https://chatzichristofis.info/?page_id=15#ccd2 Similarly, there exist various DLL-based machine learning .NET assemblies in the .NET realm such as Accord.NET urlhttp://accord-framework.net/ or EmguCV https://www.emgu.com/wiki/index.php/Main_Page. You can add your assemblies by "Add Reference" in your project menu and add your DLL files. In this way, you can leverage pre-built assemblies in your project.
- In order to have computation equivalence, **no** CUDA-based implementation is allowed. Because there are various on-the-shelf CUDA packages. Thus, please use CPU-based algorithms.
- Your application must be a console application
- Your console application will take some start-up arguments and process the data according to the given parameters.
- Pay attention to using relative paths. Never apply absolute paths!

4 Arguments

The application arguments are described below:

- Your application ("pi.exe") first needs to take the **relative** folder path of the Phish-Iris dataset containing "train" and "val" folders. The argument name will be -dataset

```
-dataset ..\phishIRIS_DL_Dataset
```

- There exist two different modes in pi.exe which are (*precompute, trainval*). The *precompute* here refers to producing feature vectors for all train and val images and saving the n-dim feature vectors in a csv.file named as image feature name + ".csv". Such as "precomputed_HOG_train.csv" or "precomputed_CEDD_val.csv". "train" folder images are used to train an ML model whereas the "val" folder images are used to evaluate the performance of the trained model on unseen "val" images. The .CSV files are used to store comma-separated values row-by-row for each data sample. Here, in this assignment, they will contain n-dim feature vectors as comma-separated float or integer numbers + class label (either in integer (i.e. 0,1,2,3..) or a nominal string value (i.e. "DHL" or "Yahoo") depending on your ML usage).

  Trainval mode refers to making training based on pre-computed training .CSV files and evaluating the validation images. The classification report must be given by providing TPR, FPR, and F-1 score.

```
-mode precompute|trainval
```

- Note that, if the pre-computed .CSV files exist in the sub-folder ("precomputed") then trainval mode must benefit from them to avoid the recomputation of the feature representation of each image. If one or more precomputed CSV files are missing, then your application must re-generate related .CSV file(s) to significantly speed up the training/validation stage
- All training and evaluation processes **must** use precomputed feature files by prompting on console. If the file is missing or renamed then pi.exe must first generate and store the related file.
- At the end of the training process, the final output must list all TPR,FPR and F1-Score values for each image feature type and ML algorithm pairs.
- As you might guess, there is no feature type parameter. It is because you are assumed that you already picked your algorithms hard-coded and there is no need to specify them in the arguments. Thus, **be ensured that you provide your *precomputed* files in the folder "pre-computed"** and

let the code train and evaluates much faster. Otherwise, the computation of pre-processing phase will take a significant amount of time causing -20 points in the evaluation.

5 Example Input and Outputs

Please perform verbose operations and feed the user with friendly messages to inform how the process is happening.

```
Inp:
pi.exe -dataset ..\phishIRIS_DL_Dataset -mode precompute

Output
Reading phishIRIS_DL_Dataset...
1313 images were found in train folder
1539 images were found in val folder
14 classes exist
FCTH features are being extracted for train...
Done. precomputed_FCTH_train.csv is regenerated in XX seconds
FCTH features are being extracted for val...
Done. precomputed_FCTH_val.csv is regenerated in XX seconds
CEDD features are being extracted for train..
Done precomputed_CEDD_train.csv is regenerated in XX seconds
CEDD features are being extracted for val...
Done precomputed_CEDD_val.csv is regenerated in XX seconds
SIFT features are being extracted for train...
Done precomputed_SIFT_train.csv is regenerated in XX seconds
SIFT features are being extracted for val...
Done precomputed_SIFT_val.csv is regenerated in XX seconds

Inp:
pi.exe -dataset ..\phishIRIS_DL_Dataset -mode trainval

Output

Training with  precomputed_FCTH_train.csv
Done in YY seconds
Testing with precomputed_FCTH_val.csv 1539 samples
Random Forest | TPR 0.829 | FPR 0.118 | F1 0.826
SVM           | TPR 0.694 | FPR 0.084 | F1 0.715
-----------------------------------------------
Training with precomputed_CEDD_train.csv
Done in ZZ seconds
Testing with precomputed_CEDD_val.csv 1539 samples
Random Forest | TPR 0.845 | FPR 0.109 | F1 0.844
SVM           | TPR 0.739 | FPR 0.078 | F1 0.755
```

```
-----------------------------------------------
Training with precomputed_SIFT_train.csv
Done in TT seconds
Testing with precomputed_SIFT_val.csv 1539 samples
Random Forest | TPR 0.887 | FPR 0.008 | F1 0.890
SVM           | TPR 0.875 | FPR 0.008 | F1 0.880
-----------------------------------------------
```

6 Notes

1. It is of utmost importance that your submission must work on the instructor's PC. Therefore, be ensured that you included and shipped all the DLL files (assemblies) in the submission package. Be sure that your code can be compiled without any errors.

2. If you have confusions about how the quantization-based local image feature extraction works, please refer to the paper `https://www.researchgate.net/publication/336829974_Local_Image_Descriptor_Based_Phishing_Web_Page_Recognition_as_an_Open-Set_Problem` for more details. Keep in mind that, local features are based on automatic key point finding and obtaining a vector-based description (e.g. 128-dim vectors) around each key point. The idea is that accumulation of many vectors and clustering them with K-means (you can also use K-medoid) algorithm lets you obtain characteristic visual words within each cluster centroid. The only remaining stage is to construct a 1-D histogram denoting how many times these visual words occur for each image. Thus, it would be an integer-based vector. Here chose of the parameter K explicitly means how rich your *vocabulary* is.

Figure 2: SIFT based key-points on a web page screenshot

3. (For only local image feature analysis) Pay attention to, not including validation samples in the clustering phase. Otherwise, the ML algo-

rithm would also know validation data prior and you obtain incredibly high scores which are not valid since you had cheated in ML terminology. Therefore, it is essential to apply the transformation of validation samples after the clustering-pooling (i.e. quantization) stage.

4. It is not mandatory to use a local image feature extraction based method. However, inclusion of one (SIFT or SURF) will yield +20 points.

5. You need to submit a detailed lab report (i.e report.pdf) to describe what you have done and throughout the homework. It is also expected you to show your understanding of the image descriptors you have used throughout the assignment. Therefore, please explain the image features briefly. Moreover, please include a system flow chart

6. You can ask questions about the experiment via Piazza group (`piazza.com/hacettepe.edu.tr/fall2022/bbm465`).

7. Late submission policy will be accepted as being conducted for other assignments.

8. You are going to submit your experiment to online submission system: `www.submit.cs.hacettepe.edu.tr`

   The submission format is given below:
   <bxxxx id>.zip
   −app/
   Everything excluding the dataset
   −report/
   report.pdf

7 Policy

All work on assignments must be done with your own group unless stated otherwise. You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudocode) will not be tolerated. In short, turning in someone else's work(from internet), in whole or in part, as your own will be considered as a violation of academic integrity. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else.
The content of this assignment can be partially changed by the instructor. However, all modifications will be reported into the Piazza system. Therefore, do not forget to keep an eye on Piazza for potential changes.