Classifier creation framework for diverse classification tasks

Ivo Řezníček, David Bařina
Department of Computer Graphics and Multimedia
Faculty of Information Technology
Brno University of Technology
Brno, Božetěchova 2, Czech Republic, 612 66
`[ireznice, ibarina]@fit.vutbr.cz`

Abstract *— This paper introduces a novel framework for classifier design process. The well known procedure of the creation is used. The operations of the procedure are represented using executable programs that are capable of processing various types of input data. These executable programs are invoked using the framework in multiprocessor environment and as the result they design the desired classifier using the machine-learning process. The framework is now being tested for design of image recognition classifiers and action recognition tasks.*

1 INTRODUCTION

Today's technology makes it possible to acquire huge amount of the audio-visual data and also the needs for analyzing such data. In these audiovisual data the amount of critical situations can be viewed and detection or analysis of these situations may be wanted to either prevent these situations or be able to deal with it more effectively and save lives, resources etc. Such interesting situation feasible for analysis/detection can be for example:

- Detection of left baggage,
- weapon carrying,
- person's unusual movement detection, for example the moving the other direction than expected,
- traffic violations, etc.

The basic procedure is the feature extraction from the input video or image and the following classification. The feature vectors are used to describe the unique or interesting properties of the input object, where is assumed, that for the alike input object the feature vectors are similar and conversely. These feature vectors are passed to the machine learning technique and model is created. The accuracy of describes procedure is mainly dependent on the feature vectors ability to distinguish between the input classes and the properties of the machine learning technique, such as resistance to noise, etc., for the on-line system the complexity and speed of the feature extraction is very important

When the classifier (model) is computed, it is assumed that classifier is able to recognize the unknown input objects, when the same feature vectors are computed from it.

This paper introduces a framework for classifier creation, which consists of three principal parts:

- The input data processor
- The classifier creator/tester
- The Database

The input data processor handles the feature extraction part of processing and the database is used for communication between all system's parts.

The purpose of the presented framework is to create the classifier which can be next used in the on-line image or video processing system, the only requirements for such systems is the realtime processing capabilities, thus the feature extraction method and following classification technique have to be fast enough to suit this requirement.

In the previous paragraphs, the images and video sequences were used as the input objects, the framework is designed to work with various types of input object, only the procedure of classifiers creation is changeless.

The paper is organized as follows: in the chapter 2 the framework is presented, in the chapter 3 the standard usage of the framework is discussed and in the chapter 4 the experimental results are showed.

2 FRAMEWORK DESIGN

The system is based on a database containing mainly the image and video datasets with annotations; they are imported using especially formatted files and sets of output files previously processed using the input data processor and the classifiers' responses to the certain datasets.

Annotation files fully specifies the dataset content, the number of lines defines the number of objects in the dataset, the first string value of the line specifies the full path to the object (video, image, etc.) and the rest values specify the annotation of the current object, the number of classes is not restricted and the annotations can fulfill the 3 predefined values: 0 - neutral annotation, 1 – positive annotation, -1 negative annotation. The number of fields must be equal in all lines of the dataset.

The feature vectors and classifiers responses are stored in special binary data files, the manipulation utilities for converting from/to other formats are available, for example libsvm [1] format is supported. The format of the classifier model files is depended on the classification algorithm, in most cases all of it are the readable text files.

In the next sections the two basic framework parts are described:

2.1 The input data processor

The purpose of the input data processor is to transform input list of object into feature vectors. The input list of objects is stored in the database as a dataset. The procedure consists the following four parts:

- **Local feature extraction**

  The feature extractor executable is applied to the all objects in the input list and all of the extracted local features are stored into files.

- **Codebook generation**

  Random subset of local feature vectors is selected for codebook creation, the codebook generation process is executed, and the codebook is saved into a file.

- **Bag-of-words like feature vectors creation**

  The local features file for each of the input objects and the codebook is used for translation into the output feature vector representation.

- **Merging the output feature vector file**

  All of the translated feature vectors are merged into a single descriptor file in predefined order, this file location is stored into database and at this point it is ready to be used in the next processing.

The local feature extraction part is based on several feature extractors. The higher number of feature extractors or the higher number of settings of one feature extractors can be specified and the framework will evaluate all of them.

Similar situation occurs in the codebook generation where multiple settings may be specified and then for each local feature vector definition the codebooks are created, all possible combinations are performed, all possible bag-of-words like feature vectors are translated.

The number of output feature vector files is equal to N_f X N_c, where the N_f is the number of feature extractors applied to the input dataset, and N_c is the number of generated codebooks.

All the computation in this subsystem can be computed on local computer or can be submitted to the Sun Grid engine [3] parallel jobs scheduling system and can be computed on a cluster. The system runs all the computation in stages, the stages are the same as in the previous bullets list.

2.2 The classifier creator / tester

Input to this subsystem is the feature vector file, the dataset identification, and the annotations. All this information is stored into the database; the purpose is to create the classifier for all of the input annotated classes. As a machine learning technique, the linear and nonlinear Support vector machines (SVMs) are now supported.

The running of the training and testing process is controlled by two files; first defines the feature vectors and dataset which can be used for classifier creation, the second specifies the configuration of learning algorithms. The learning can be executed on local computer or can be submitted to the Sun Grid Engine [3] scheduling system to run in parallel, in this case every learning process is represented as one job and all of then are run simultaneously.

The output of this subsystem is one classifier for every combination of local features used and the codebook definition from preceding paragraph and the number of classes defined in the dataset. The average precision is computed for every class in the dataset and mean average precision is computed over all classes in the dataset.

The subsystem can be changed and another classification technique can be adapted to, for example, neural nets, Adaboost, Gaussian mixture models, etc.

Each classifier and its results (responses) can be stored back to the database for further processing, for example classifiers fusion.

3 USAGE

The standard way of usage of the presented framework is to create the classifier based on the training dataset and testing the classifier on the testing dataset. The typical procedure is as follows:

- Run the input data processor on the training part of dataset, retrieve the feature vectors, and store them in the database.
- Use the resulting feature vectors and create the classifiers according to the number of annotated classes and store all the information about the process into the database.

- Execute the same (as above) operation on the testing dataset and evaluate the stored classifiers on the testing dataset.

The processing of the testing dataset can be speeded up, because of the codebook, which is used for the translation from local features to global feature descriptor have to be simply copied, the codebook has to be the same in both cases for one type of local feature descriptors. The processing of the testing dataset is speeded up once more, the classifier is in that case only tested with new global feature vectors, the only condition for that is, the information of the classifier must be stored in the database before testing.

The main advantage of the whole framework is the ability to perform the experimental development of all components used in the framework:

- The local feature extractors,
- the codebook generation,
- the methods for translation local features to gobal features,
- the classification algorithms.

The most interesting research can be done in the local feature extractors and the translation of local features into global features and the presented framework is designed for rapid parallel testing of the developed parts used in it.

4 RESULTS

This framework has been successfully used in TRECVID 2010 evaluation [4], Video object challenge 2010[5] and will be used in the TA2 project for creating the classifiers for on-line human behavior recognition.

The results in the TRECVID 2010 evaluation were average, the 135 object classes were trained and the 30 selected classes were evaluated. For the local feature extraction from images were the bigger amount of methods used, the codebook were created using the k-means algorithm with 4000 model vectors and the svm classifier were created for all the local extractors. The output from all svm classifiers were fused using the logistic regression.

The results in the VOC 2010 was very successful, the evaluation consist of classifying the 20 classes. The feature extraction performed from the training examples set were the grayscale SIFT [6] and the color SIFT [6] descriptors, the codebook were constructed using k-means clustering algorithm and the number of output model vectors were 4000. For every type of local feature extractor were trained 20 svm classifiers (for each class) and the output classifiers of the same class were fused using the logistic regression.

5 CONCLUSIONS

The framework has been created to operate on various type of input objects, the computational stages are constructed universally, all of it are replaceable and thus, the framework is suitable for creating comparison of diverse techniques at every stage of the framework's pipeline.

The further usage of the framework will be the preparation of classifiers for on-line video data processing, with usage of space-time low level feature extractors.

Acknowledgments

This work has been partially supported by the EU FP7 project TA2: Together Anywhere, Together Anytime (grant no. 214793) and EU FP7-ARTEMIS SMECY: Smart Multicore Embedded SYstems, (grant no.100230).

References

[1] Chih-Chung Chang and Chih-Jen Lin, *LIBSVM : a library for support vector machines*, 2001. Available at *http://www.csie.ntu.edu.tw/~cjlin/libsvm*

[2] R.-E. Fan, et al. *LIBLINEAR: A Library for Large Linear Classification,* Journal of Machine Learning Research 9(2008), 1871-1874. Software available at *http://www.csie.ntu.edu.tw/~cjlin/liblinear*

[3] SUN MICROSYSTEMS, INC. 2010. *N1 Grid Engine 6 User's Guide*, January 2010.

[4] SMEATON, A. F., OVER, P., AND KRAAIJ, W. 2009. *High-Level Feature Detection from Video in TRECVid: a 5-Year Retrospective of Achievements.* In Multimedia Content Analysis, Theory and Applications, A. Divakaran, Ed. Springer Verlag, Berlin, 151–174.

[5] Everingham, M. et al. *The PASCAL Visual Object Classes Challenge 2010 Results* electronic file available at: *http://www.pascal-network.org/ chalenges/VOC/voc2010/workshop/index.html,* 2010.

[6] Koen E. A. van de Sande et al. Evaluating Color Descriptors for Object and Scene Recognition, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 32 (9), pages 1582-1596, 2010.