TRECVid 2008 High Level Feature Extraction Experiments at JOANNEUM RESEARCH

Roland Mörzinger and Georg Thallinger
JOANNEUM RESEARCH, Institute of Information Systems & Information Management
Steyrergasse 17, 8010 Graz, Austria
roland.moerzinger@joanneum.at

ABSTRACT

This paper describes our experiments for the high level feature extraction task in TRECVid 2008. We submitted the following six runs:

- A_jrs1_1: Baseline, early fusion, tuned SVMs.
- A_jrs2_2: Early fusion, SVMs without parameter tuning.
- A_jrs3_3: Late fusion, tuned SVMs.
- A_jrs4_4: Early fusion, transductive SVM (TSVM) learning using UniverSVM with unlabeled samples on the scale of 20% of test set size.
- A_jrs5_5: Early fusion, transductive SVM learning using SVM^{light} TSVM with unlabeled samples on the scale of the training set size.
- A_jrs6_6: Early fusion, transductive SVM learning using SVM^{light} TSVM with unlabeled samples on the scale of 20% of test set size.

The experiments were designed to study both the performance of various content-based features in connection with classic early and late feature fusion as well as the influence of SVM parameter tuning and learning from unlabeled test data with different implementations of transductive support vector machines (TSVMs). The results show that the time-consuming parameter tuning improves precision only marginally. Compared to standard SVMs, TSVMs did not provide overall improvement and only slight benefits for concepts with a small number of training samples.

1. INTRODUCTION

In TRECVid 2008 our group participated in the BBC Rushes Summarization[1] and in the High Level Feature Extraction task. This notebook paper describes the submission to the High Level Feature Extraction task. Similar to our participation in 2007[6], we submitted a set of 6 runs with the goal to investigate:

- the suitability of various content-based features for building a baseline run,
- performance of early fusion as opposed to late feature fusion,
- the influence of parameter SVM parameter tuning,
- the influence of different transductive learning schemes and
- time complexity.

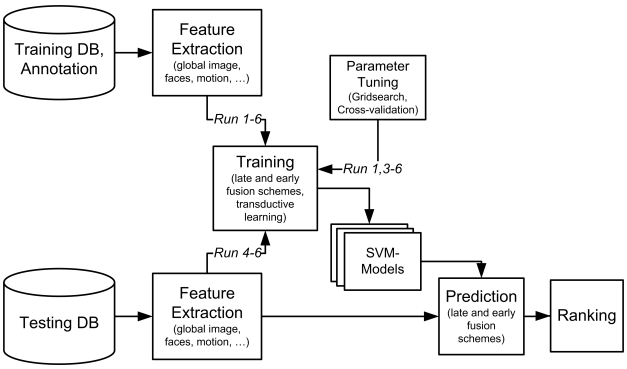

Figure 1: Basic system architecture.

Figure 1 shows the basic system architecture. The annotations for the training data were provided by the MCG-ICT-CAS team[9]. For the implementation of the training and prediction components we used the SVM software packages LIBSVM[2] for building inductive SVMs and SVM^{light} [4] and UniverSVM[8] for transductive SVMs.

The rest of this paper is organized as follows: Section 2 describes content-based features in detail, Section 3 the training and prediction process. Results are presented in Section 4, conclusions in 5.

2. CONTENT-BASED FEATURES

The content-based features described here are the basis for the detection methods for high-level semantic concepts. The following MPEG-7[7] image features were extracted globally:

Color Layout describes the spatial distribution of colors. This feature is computed by clustering the image into 8x8 blocks and deriving the average value for each block. After computation of DCT and encoding, a set of low frequency DCT components is selected (6 for the Y, 3 for the Cb and Cr plane).

Dominant Color consists of a small number of representative colors, the fraction of the image represented by each color cluster and its variance. We use three dominant colors extracted by mean shift color clustering[3].

Color Structure captures both, color content and information about the spatial arrangement of the colors. Specif-

ically, we compute a 32-bin histogram that counts the number of times a color is present in an 8x8 windowed neighbourhood, as this window progresses over the image rows and columns.

EdgeHistogram represents the spatial distribution of five types of edges, namely four directional edges and one non-directional edge. We use a global histogram generated directly from the local edge histograms of 4x4 sub-images.

Gabor Energy is computed by filtering the image with a bank of orientation and scale sensitive filters and calculating the mean and standard deviation of the filtered outputs in the frequency space. We applied a fast recursive gabor filtering for 4 scales and 6 orientations.

The *visual activity* feature is computed by temporally subsampling the video and computing the mean absolute frame differences (MAFD). The description contains statistics about minimum, maximum, mean and median MAFD per shot.

For each shot the *number of faces* is detected on the temporally subsampled video, by using the face detection method implemented in OpenCV[1]. The mode (most frequent value) of the number of detected faces in the frames is described for each shot.

In contrast to our participation in 2007, we did not use the camera motion feature because of a lack of computational power. It has to be noted that this fact influences the chances of comparability between these submissions, which is already impaired by the changed definition of high level features.

2.1 Feature Preprocessing

The used content-based features capture both, global image properties (color and texture) and shot properties (faces, visual activity). To overcome the limitations of having only one keyframe representing a shot's visual content, we extracted every second I-frame from each shot from the test set. Some input features were preprocessed before transferring them to the training or prediction system. Specifically, the *number of faces* value was quantized to 0 (no face), 1 (one face), 2 (two faces) and 3 (three faces or more). This seems valid and necessary considering high level features related to people, e.g. 'Two_People', 'Driver', 'Demonstration_Or_Protest' or 'Singing', where this differentiation may be significant for the concept detector. As feature for *visual activity* the mean MAFD value was adopted. Finally, all feature vectors were statistically normalized, by converting into a normal distribution with zero mean and unit variance.

3. TRAINING

Our approach to high level feature extraction is based on training support vector machines (SVMs) since they had achieved quite satisfactory performance in concept detection over the past few years. This type of classifier also obtained good results for our participation to this task in the last year. The classification of each high level feature (concept) was regarded as a two-class problem, where the positive and negative examples were extracted from the annotations. Since there were more negative examples than positive examples for most of the concepts, the SVM training data was composed of all positive annotations with a comparable number of randomly selected negative annotations. For better comparability we assured that the random selection produced the same annotations across different runs. We adopted the Gaussian RBF kernel function. In all runs except A_jrs2_2 we tuned the SVMs by grid search with cross-validation in order to select the best choice of the parameters C and γ. Depending on the setup of our runs, SVMs were built using a single modality of low-level features (for late feature fusion) or using an input feature vector composed of multiple features. Finally, we ranked the shots according to the prediction outputs from the two-class SVMs for the runs with early fusion. The run A_jrs3_3 performs late fusion which required the combination of the output of various base classifiers, for which we chose the product of probabilities.

[1]http://sourceforge.net/projects/opencvlibrary

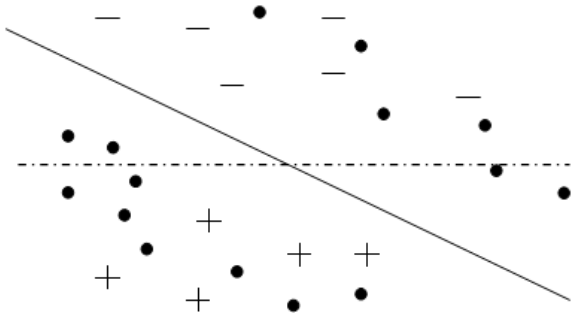

Figure 2: Labeled samples are marked as +/-, unlabeled test samples as dots. The dashed line shows the classification from the inductive SVM using only labeled data. The solid line is the maximum margin hyperplane of the transductive SVM using all samples.

3.1 Transductive SVMs

In the high level feature extraction task, labeled training data is available for development and an equal amount of unlabeled data is used for testing. Semi-supervised learning methods can incorporate both labeled and unlabeled data. In order to compensate for the scarcity of training data in this task, we apply one of such methods, the Transductive SVM (TSVM). The key idea, as introduced by Joachims[5] and illustrated in Figure 2, is to find the labeling of the test examples that optimizes the decision boundary (i.e. maximizing margin) which separates the positive and negative examples of both the training and the test data. Finding a solution to the resulting optimization problem is hard, hence many different approximations exist. Unfortunately, these implementations often do not scale well when dealing with a large number of unlabeled examples.

An implementation that is practical for a few thousand examples in the non-linear case is the widely used $\text{SVM}^{\text{light}}$ TSVM software[4]. In our experiments we use $\text{SVM}^{\text{light}}$ and UniverSVM[8]. UniverSVM is a SVM implementation that uses the Concave-Convex Procedure (CCCP) to optimize transductive SVMs. The authors report that optimizing TSVMs with CCCP improves accuracy and decreases training time when compared to other heuristic methods. In run A_jrs4_4 we used UniverSVM for learning classifiers from both the labeled data (the same as in run A_jrs1_1) and random 20% of the unlabeled test data. The TSVM from $\text{SVM}^{\text{light}}$ has been applied in a similar fashion in run A_jrs6_6. In order to see the influence of changing the number of unlabeled data, this number was put on a level with the number of labeled training samples in run A_jrs5_5 .

4. RESULTS AND DISCUSSION

The results are shown in Table 1. The first observation is that the overall performance is worse than in 2007. This may be due to a more complicated selection of concepts on the one hand and the missing camera motion feature on the other hand. The baseline run was our best performing run with a mean inferred average precision of 0.039, closely followed by runs using transductive SVMs and untuned SVMs.

The performance of the untuned SVMs in run jrs_2_2 is slightly down as compared to the SVMs subject to parameter tuning. Parameter tuning using grid search and cross validation aims at improving generalization and thus is generally beneficial. Interestingly, the detection of the 'Dog' concept was significantly impaired by the tuning. The reason is that the majority of training and test data for this concept consists of shots that show exactly the same scene (dog walking on beach, e.g. shot38_2). Only in this case it is preferable to have a less generalized classifier that *memorizes* the specific appearance of the training data instead of *learning* a general pattern from it. Early fusion outperformed late fusion significantly, which is partly contradictory to our observation in the last year where early fusion went only slightly ahead.

Those features where our system achieved good results generally correlate with a good overall TRECVid median and large number of positive training samples; the opposite holds true for bad scores. The imbalanced training sets, i.e. only between 0.1% and 8% positives samples, also raise a problem for transductive SVMs. A common drawback of these methods is that they require the ratio between positive and negative examples in the test data to be close to the ratio in the training data. For example, SVM$^{\text{light}}$ implements a balancing constraint to make sure that the average prediction of unlabeled points from the test set equals the average label of labeled points from the training set. This requirement does not hold in this task, particularly when the training dataset is small. In our setup the initial SVMs were trained with an equal number of positive and negative samples based on previous empirical experiments with unbalanced datasets. This explains why the TSVMs in general did not outperform the traditional classifiers which were only trained on labeled data. It also advocates the need for techniques improving the training of TSVMs in the presence of imbalanced datasets.

A comparison of the runs using transduction indicates that the SVM$^{\text{light}}$ implementations performed better than UniverSVM. Furthermore it is beneficial to use a larger number of unlabeled samples (run jrs_6_6) which results in an improved classification of some concepts with very small number of training samples. However, TSVMs are generally limited by the quality of the initial classifier trained on the labeled source.

4.1 Run-time Complexity

A summary of the computational complexity for SVM training, parameter selection and prediction and can be found in Table 2. The times were measured on a workstation with an Intel Core 2 CPU (2.1GHz) and 2GB RAM. Generally, the computational time for training and for parameter tuning using grid search with n-fold cross validation increased with the number of training samples. Similarly, the training needs more time when incorporating more unlabeled data (A_jrs4_4 to A_jrs6_6). For the concept 'Two_people' which is trained on about 3000 samples, the tuning process took 2 hours. The difference between the run-time performance of early and late fusion is insignificant.

Naturally, the training time for transductive SVMs is considerably longer than the one needed for building inductive SVMs. This also holds true for the prediction time. The run A_jrs4_4 using transduction with UniverSVM shows a shorter average training and prediction time as compared to A_jrs6_6 which uses the same input but SVM$^{\text{light}}$.

5. CONCLUSIONS

We have presented experiments on high level feature extraction using a series of SVM classifiers based on a variety of low-level features combining global image information, face detection and visual activity. We assume that our baseline outperforms the other runs due to the effective grounding of a variety of low-level visual features and the generalization capability of the SVM framework with high-dimensional feature spaces.

Transduction was applied for incorporating unlabeled data in the training process, early and late fusion methods were used for prediction. The usage of TSVMs was not profitable because of issues with training on imbalanced datasets.

6. ACKNOWLEDGMENTS

The authors would like to thank Werner Bailer and Werner Haas for their support and feedback. The research leading to this paper was partially supported by the European Commission under the contracts IST-045032 (SEMEDIA) and FP6-027122 (SALERO).

7. REFERENCES

[1] W. Bailer and G. Thallinger. Comparison of content selection methods for skimming rushes video. In *Proc. of the TRECVID BBC Rushes Summarization Workshop at ACM MM*, Vancouver, CA, Oct. 2008.

[2] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[3] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(5):603–619, 2002.

[4] T. Joachims. Making large-scale support vector machine learning practical. *Advances in kernel methods: support vector learning*, pages 169–184, 1999.

[5] T. Joachims. Transductive inference for text classification using support vector machines. In *Proc. of Int. Conf. on Machine Learning*, 1999.

[6] R. Mörzinger and G. Thallinger. TRECVid 2007 High Level Feature Extraction experiments at JOANNEUM RESEARCH. In *Proceedings of TRECVID Workshop*, 2007.

[7] MPEG-7. Multimedia Content Description Interface - Part 3 Visual. Standard No. ISO/IEC n15938, 2001.

[8] F. Sinz and M. Roffilli. UniverSVM, 2007. `http://mloss.org/software/view/19/`.

[9] S. Tang. TRECVID 2008 High-Level Feature Extraction By MCG-ICT-CAS. In *Proceedings of TRECVID Workshop*, Gaithersburg, MD, USA, 11 2008. NIST.

Feature	#pos. train	#hits test	median	A_jrs1_1	A_jrs2_2	A_jrs3_3	A_jrs4_4	A_jrs5_5	A_jrs6_6
001 Classroom	186	64	0.008	0.003	0.003	0.001	0.004	0.000	0.000
002 Bridge	153	30	0.004	0.011	0.027	0.000	0.010	0.009	0.011
003 Emergency_Veh.	125	22	0.003	0.002	0.001	0.009	0.001	0.003	0.007
004 Dog	99	94	0.101	0.022	0.085	0.004	0.007	0.023	0.006
005 Kitchen	228	124	0.010	0.004	0.005	0.005	0.006	0.002	0.001
006 Airplane_flying	53	64	0.029	0.068	0.042	0.005	0.018	0.040	0.097
007 Two_People	3216	1090	0.050	0.019	0.035	0.016	0.028	0.005	0.024
008 Bus	82	47	0.004	0.002	0.002	0.000	0.001	0.003	0.003
009 Driver	109	364	0.046	0.053	0.035	0.021	0.031	0.033	0.037
010 Cityscape	936	337	0.059	0.078	0.073	0.042	0.057	0.060	0.063
011 Harbor	153	35	0.008	0.004	0.007	0.008	0.005	0.004	0.003
012 Telephone	112	106	0.011	0.005	0.005	0.001	0.002	0.011	0.005
013 Street	1479	458	0.113	0.085	0.076	0.031	0.087	0.060	0.070
014 Demo_Or_Prot.	55	87	0.013	0.006	0.003	0.006	0.001	0.007	0.006
015 Hand	1098	630	0.095	0.113	0.071	0.067	0.084	0.102	0.099
016 Mountain	140	140	0.041	0.035	0.032	0.012	0.018	0.027	0.032
017 Nighttime	336	316	0.102	0.108	0.082	0.079	0.047	0.090	0.102
018 Boat_Ship	372	210	0.093	0.099	0.111	0.090	0.084	0.093	0.092
019 Flower	649	319	0.058	0.063	0.062	0.057	0.002	0.055	0.068
020 Singing	310	133	0.014	0.007	0.007	0.002	0.003	0.007	0.011
mean IAP			0.043	0.039	0.038	0.023	0.025	0.031	0.037

Table 1: High Level Feature Extraction Results. The 20 evaluated features are shown with the number of positive training samples in the training set and number of unique hits in the test set. The TRECVid median and the inferred average precision measures of all our runs are stated.

	Training	Tuning	Prediction
A_jrs1_1	10sec - 2min	2.5min - 2h	30sec - 2min
A_jrs2_2	10sec - 2min	-	30sec - 2min
A_jrs3_3	15sec - 2min	2min - 2h	20sec - 5min
A_jrs4_4	7min - 20min	2.5min - 2h	30sec - 20min
A_jrs5_5	20sec - 3min	2.5min - 2h	20sec - 5min
A_jrs6_6	3min - 1h	2.5min - 2h	3min - 1h

Table 2: Computational complexity for each submitted run. The minimum and maximum needed time is given for training (SVM model creation), prediction on the 2008 test data and parameter tuning per high level feature. The minimum time generally applies to the high level features that come with the fewest training samples, as opposed to the maximum training time which is valid for the more frequent concepts, e.g. Two_people, Street and Hand.