UC3M HIGH LEVEL FEATURE EXTRACTION AT TRECVID 2008

I. González-Díaz, D. García-García, R. Solera-Ureña, J. Madrid-Sánchez,
V. Gómez-Verdejo, M. Martínez-Ramón, F. Díaz-de-María, and J. Arenas-García[*]

Department of Signal Theory and Communications
Universidad Carlos III de Madrid, Leganés, Spain

ABSTRACT

This paper describes experiments carried out by the UC3M team for TRECVID 2008 high-level feature extraction task. Being our first participation in TRECVID, our goal this year has been to develop a modular system to facilitate future developments and incorporation of new functionality (feature extraction and classification modules). We have basically carried out experiments with two different kinds of classification technologies, namely Support Vector Machines (SVMs) and Multi-Task Learning (MTL), resulting in the following runs:

- *SVM baseline*: Scheme with late fusion and SVM classifiers trained on the shot level, keyframe level, and region level low-level descriptors.
- *SVM with bal-SIFT*: Similar to our SVM baseline, but incorporating a different kind of region descriptors using a balanced codeword.
- *Bagging of SVM*: Rather than using individual SVM classifiers, multi-net bagging systems are used.
- *MLP-STL*: Scheme with late fusion and Multi Layer Perceptrons (MLP) trained individually for each high-level concept. This run serves as a baseline for the two following Multi-Task systems.
- *Selective MLP-MTL*: Incorporates Multi-Task Learning, assigning all tasks the same importance.
- *Selective MLP-MTL with priority*: Multi-Task Learning scheme, with focus on the individual performance of each category.

The six submitted runs have achieved very similar performance in terms of average InfAP, with results which are slightly better or slightly worse than the median of all submitted runs.

[*]This work has been supported by Alcatel-Lucent (Spain) and the Spanish Ministry of Science and Innovation under the i3Media CENIT Research Contract.

1. INTRODUCTION

The main goal of our first participation in the TRECVID high-level feature extraction (HFE) was to create a modular system architecture that allows for future developments in the years to come. In this way, in future versions of our system we expect to be able to incorporate new processing units in an easy way and, at the same time, to take advantage of the previous years work.

As Figure 1 shows, the proposed system has four processing steps: (1) a low level feature extraction layer, (2) selection/extraction of the most adequate low-level features (LF), (3) a supervised learning step, and (4) a late fusion stage. The components of the first layer (denoted as FE_1,..., FE_N in the figure) extract groups of low-level descriptors at different levels: Shot Level features, Keyframe Level features and Region Level features. Although we have not considered early fusion for this year system, it could be implemented by adding to this first layer new modules which concatenate low level features of different nature.

Once LF are extracted, there are different algorithmic approaches to reduce the dimensionality of this data, either selecting the most appropriate features (feature selection), or creating new ones (feature extraction). This is the role of the second layer of our architecture, which none of the systems presented to this year competition implemented yet.

For each of the resulting features block and each of the high-level features (HF), we train a system using the labelled data resulting from the collaborative annotation task [1], combining their confidences about the presence or absence of the HF in a final late fusion process.

Following this general framework, our participation in TRECVID consists of six different approaches, where the differences between runs are mainly in the kind of base classifiers employed. In particular, the submitted runs can be classified in two groups according to the kind of base learner employed:

1. Three of the six submitted runs have employed Support Vector Machines (SVM) as base classi-

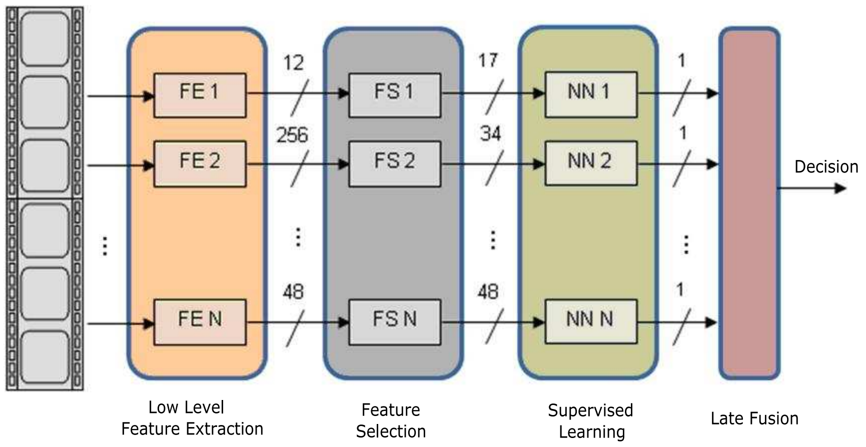


Fig. 1. HFE system modular architecture.

fiers. Namely, these three runs consist of a baseline scheme and two improved schemes that incorporate either novel region level features or a Bagging methodology.

2. The other three runs consist of the combination of Multi Layer Perceptrons (MLP). As in the above case, one of the runs is a baseline approach and the other two are extensions based on Multi Task Learning (MTL).

The rest of the paper is organized as follows: The next section presents the Low Level Features that are extracted from the video data. In Section 3, we describe in detail the kind of base classifiers employed in each run as well as their training process. Next, experimental results are analyzed in Section 4. Finally, conclusions and future work are included in Section 5.

2. LOW-LEVEL FEATURE EXTRACTION

High level feature extraction task relies on low-level features generated from the audiovisual content. In our system, low-level descriptors have been computed at different levels or granularities, namely (a) *Shot Level features*, that capture content variation within a shot, (b) *Keyframe Level features*, that describe image content on each keyframe, and (c) *Region Level features*, that apply spatial regionalization to detect and describe specially discriminant areas in images.

2.1. Shot level features

Two shot level descriptors are extracted to model dynamic content within a shot. In particular, the proposed system uses:

- *Motion Activity (MA)*: this MPEG-7 descriptor [2] provides a general categorization of motion within a shot, modelling parameters such as motion intensity, dominant orientation, and including spatial regionalization to some extent.

- *Mel-Frequency Cepstrum Coefficients (MFCC)*: This set of features gives a description of the inband espectral energy present in the audio track of a video shot. In our particular implementation, 39 components are extracted which include 12 MFCCs, the log-energy, 13 first-order derivatives and 13 second-order derivatives. Information is extracted every 10 ms using an overlapping temporal window of 30 ms.

2.2. Keyframe level features

Each keyframe extracted from the video content is described by means of several image descriptors, namely:

- *MPEG-7 visual descriptors*: Color Layout (CL), Color Structure (CS), Dominant Color (DC), Scalable Color (SC), Homogeneous Texture (HT), Edge Histogram (EH) and Texture Browsing (TB). The interested reader is referred to [2] for a complete review of these descriptors.

- *Color Auto-correlogram (AC)*: Color correlogram, proposed in [3], extends the histogram by incorporating information about the spatial distribution of pixels. Auto-correlogram is a particularization of the correlogram that computes distributions of pairs of pixels with the same color. In our implementation, the AC is extracted at several scales (1, 3, 5 and 7 pixels) using a 64 bins color quantization.

- *Gray Level Co-ocurrence Matrix (GLCM)*: proposed in [4], the GLCM captures the spatial relations that give place to textures at several scales and orientations. Due to the complexity and length of the GLCM, several measures have been suggested in [4] to represent the matrix in a compact form. In particular, our implementation uses the variance computed at each scale and orientation to index the information.

- *Gabor Wavelet (GW)*: In this descriptor a bank of Gabor filters with various scales and rotations is created using the Wavelet transform. Following the implementation given in [5], the output of each filter is employed to compute two measures, mean and variance, that are then used for indexing.

2.3. Region level features

Some degree of regionalization has been also included in our system. Using two affine covariant region detectors, Hessian Affine Detector and Maximally Stable Extremal Regions Detector (MSER) described in [6], a set of elliptical regions is extracted for each keyframe. Then, each region (also known as keypoint) is described using a 128-dimensional SIFT descriptor [7], which has been found to be very discriminative in several computer vision tasks.

Once the SIFT-based features have been extracted, a Bag-of-words model is built which generates a codebook of visual words. A simple clustering technique like K-means has been used to compute those codewords that seem to consistently appear in the video corpus. Then, each image is vector-quantized so each region descriptor is assigned to its closest codeword and a normalized histogram of words is computed. This model allows the system to work with fixed-length input vectors of size N ($N = 1500$ in our implementation), which corresponds to the size of the vocabulary.

Furthermore, we have experimented with the generation of codebooks, trying to incorporate spatial localization of regions to some extent . The original bags-of-words model, since it comes from text retrieval area, does not explicity represent any spatial information while visual documents actually tend to show a well-defined spatial layout.

Furthermore, annotations in TRECVID training data are given at shot/subshot level, so keypoint level labels are not available. Since object-oriented concepts usually take place in small areas of the images, the basic procedure that randomly samples keypoints from positive/negative keyframes will lead to unbalanced region sets that contain a greater number of negative than positive samples. Unbalanced codebooks will then become less discriminant when detecting specific concepts, since many of the words will point to background areas rather than to the concepts.

To avoid this issue, we have used an unsupervised technique that generates labels at region/keypoint level so that balanced sets can be employed to compute the codebook. This technique is based on the well-known Probabilistic Latent Semantic Analysis (PLSA) [8], but it incorporates spatial localization of regions. Particularly, this approach classifies keypoints into two topics (foreground and background) and separately models spatial location of foreground and background as gaussian and uniform distributions, respectively. Once the regions are labeled, a balanced set with the same number of positive and negative samples is generated to compute the codebook. A complete description of the unsupervised labeling approach can be found in [9].

To distinguish between the two SIFT implementations, in the following we will refer to the former (i.e., the one with unbalanced codebook) as un-SIFT features, and to the latter (balanced codebook) as bal-SIFT features.

3. HIGH-LEVEL FEATURE EXTRACTION

For the HFE process we have relied on supervised classification techniques based on Support Vector Machines (SVMs) and Multi Layer Perceptrons (MLPs). Basically, the differences between runs reside on the specific kind of base classifiers used, altough of all them follow the same training process. Therefore, we start this section by describing the settings shared by all runs, explaining later the particularities of the runs based on SVMs and on MLPs.

3.1. General settings

According to the general system description (Figure 1), and taking into account that feature selection/extraction procedures have not been implemented in this year system, a different base classifier (either SVM or MLP) has to be trained for each kind of LFs, and each high-level label. However, some LFs have not been considered for all categories, specifically:

- SIFT variables have only be extracted for categories: Bridge, Emergency Vehicle, Dog, Airplane flying, Bus, Harbor, Telephone, Hand, Mountain, Boat ship, and Flower. The reason for this is that the codebooks necessary to generate the SIFTs have only been created for the categories with spatial localization.

- The audio data has been removed from the categories where the validation error was close to 50%. Namely, the categories in which no audio information is used are: Classroom, Two People, Hand, and Flower.

- Motion activity (MA) features have only been used by the SVM classifiers, given that the performance of MLP learners generally degraded when considering them.

Before training any classifier, we have normalized input variables to have zero mean and unitary standard deviation. Furthermore, since the TRECVID data set has many more negative patterns than positive ones, we tried to compensate both classes before training the classifiers. For this purpose, we have first taken all positive instances and a random subset of the negative ones. Secondly, the influence of positive and negative patterns on the cost function used by the classifiers has been adjusted, so that both classes are assigned the same importance during the training.

To adjust the free parameters of our models, we have used a five fold cross validation process. To give the same relevance to positive and negative classes, we separately compute their classification error and, next, we average them to obtain the average validation accuracy of the classifier.

The late fusion process is implemented using a weighted linear combination of the outputs of all classifiers, using an hyperbolic tangent activation function at the output of the combination. The combination weights of such network are adjusted using a gradient descent algorithm that minimizes the average overall accuracy, assigning again the same importance to false positives and false negatives.

3.2. SVM base learners

Three of the six runs use SVM with Gaussian kernel as base classifiers. The LIBSVM toolbox available at [10] has been employed to train these SVM classifiers. Their free parameters (kernel dispersion and factor penalty) have been selected following the cross-validation process described above, allowing a maximum of 10 negative patterns for each posive one in the training set. As previously explained, the effect of this unbalanced classes representations was compensated by using different penalizations for both classes in the SVM functional.

Concretely, the submitted runs are:

- *SVM baseline ("A_UC3M_1_5")*: This is our SVM baseline scheme, in which each SVM is directly trained with one of the sets of LFs described in Section 2. For the region descriptors, we have used un-SIFT features for our baseline.

- *SVM with bal-SIFT ("A_UC3M_2_3")*: In contrast to the previous run, this system used the balanced codebook for the region descriptors, i.e., bal-SIFT features are used instead of un-SIFT. Therefore, it only differs from the baseline system in the SVMs associated to the SIFT features (the remaining SVM classifiers have not been retrained).

- *Bagging of SVM ("A_UC3M_3_1")*: Due to each SVM is trained with a subset of all the available negative data, we have exploited the advantages of Bagging methods [11] to increase the representation of negative samples in the training of the classifiers. Then, rather than employing only one SVM for each group of LFs, we have trained a set of 10 SVMs (each one trained with a different subsampled data set), averaging their outputs to obtain the final classifier decision. In this way, and according to the principles of bagging, we should be able to reduce the variance of the error and improve the overall system performance. Note that all SVM networks of this run use the same kernel dispersion and penalization factors cross-validated for the baseline system. Therefore, no additional cross-validation was required. For future systems, however, we will consider and independent cross-validation of these parameters when using bagging, since the settings that provide optimal performance for an individual SVM do not necessarily imply the best possible results of the multi-net system.

  For this third run, un-SIFT descriptors were used, so that results could be directly compared to those of the baseline system.

3.3. Multi-Task learning with MLP base learners

The sharing of training data among the different HF concepts makes Multi-Task Learning (MTL) an interesting approach to improve generalization in the case of lack of training samples [12]. Since the number of positive instances is highly unbalanced among the different high-level concepts, we can use MTL to better exploit the available data using inter-category relationships.

We have built our MTL base classifiers upon the one-hidden layer MLP architecture proposed by Caruana [13]. Under such an approach, tasks interact by sharing all the hidden neurons and adding as many output neurons as tasks to be solved. However, since the internal representation (hidden nodes) is the same for all tasks, this approach becomes suboptimal when the tasks are not closely related [14]. Therefore, as an attempt to overcome a negative knowledge transfer among the high level features, we add to the MLP-MTL network a set of task-private hidden neurons, as illustrated in Figure 2, so that the specific structure of each concept (unrelated information) can be captured [15].

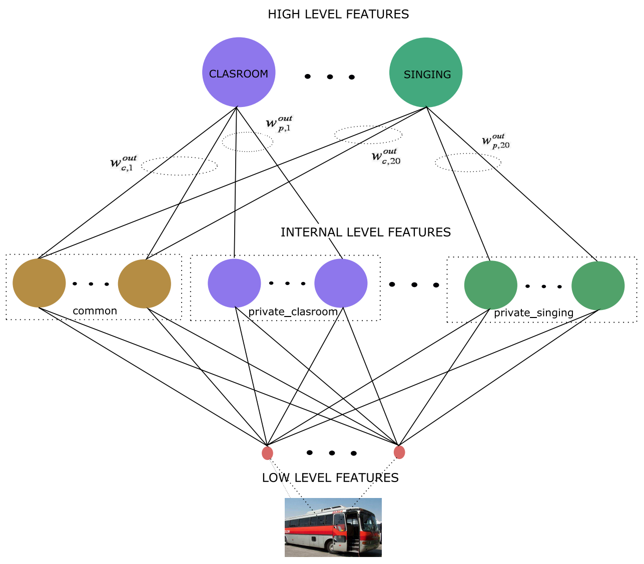


Fig. 2. Structure of MLP networks used for Multi-Task Learning of high-level concepts.

Moreover, we try to discover the relation between the multiple categories by adding a regularization term to the standard MLP-MTL cost function. Thus, considering that we have T different categories and the labeled training data set $\{(\mathbf{x}_1, y_{1t}), ..., (\mathbf{x}_M, y_{Mt})\}_{t=1}^{T}$, M being the number of training instances, we can train the task functions $\{h_t\}_{t=1}^{T}$ minimizing the following regularized cost function:

$$\sum_{t=1}^{T}\sum_{i=1}^{M}[h_t(\mathbf{x}_i) - y_{it}]^2 + \sum_{t=1}^{T}\rho_{\mathrm{c}}\left\|\mathbf{w}_{\mathrm{c},t}^{out}\right\|^2 + \rho_{\mathrm{p}}\left\|\mathbf{w}_{\mathrm{p},t}^{out}\right\|^2 \quad (1)$$

where ρ_{c} and ρ_{p} are coupling parameters weighting the output parameters $\mathbf{w}_{\mathrm{c},t}^{out}$ and $\mathbf{w}_{\mathrm{p},t}^{out}$, respectively. In other words, they weight the importance of the common and private internal representations. For instance, when the ratio $\rho_{\mathrm{c}}/\rho_{\mathrm{p}}$ is low, cost function (1) tends to be the standard MLP-MTL (i.e., only shared hidden nodes are active), whereas a large quotient strengthens the contribution of private representations (in other words, categories tend to be independently solved). This approximation selectively transfers the information among tasks (according to the coupling parameters); therefore, we will call this framework *Selective Multi-Task learning* [16].

Each MTL base classifier simultaneously predicts the 20 TRECVID categories; therefore, its training provides 20 outputs in the second step of the system and the same number of late fusions will be necessary to combine MTL outputs for each category (see Figure 3). It is important to point out that a previous task-level clustering of similarity, e.g. according to the co-ocurrences of high level features, could result in better performance; however,

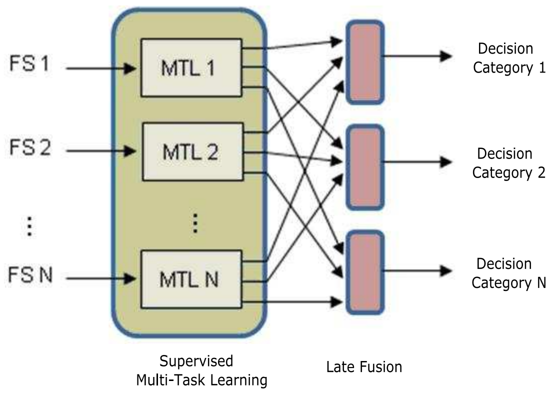


Fig. 3. System architecture for the supervised learning and late fusion layers when Multi-Task Learning is used.

time limitations have prevented us from trying it this year.

We consider two different runs based on the MTL scheme we have just described:

- *Selective MLP-MTL ("A_UC3M_16_2")*: For this run all tasks are considered equally significant. Parameters ρ_{c} and ρ_{p} have the same values for all the categories, and are cross-validated according to the average error over the 20 tasks.

- *Selective MLP-MTL with priority ("A_UC3M_20_4")*: We focus on the individual performance of each category; i.e., ρ_{c} and ρ_{p} have different values for each category.

Additionally, with the aim of analyzing the potential advantages of both MLP-MTL approaches, we have developed the following baseline system using Single-Task Learning (STL):

- *MLP-STL ("A_UC3M_12_6")*: Each category is individually solved by a different MLP.

Unlike the SVM based systems, when using MTL we have used a unique training dataset for all LFs and concepts. To balance the positive and negative classes, these three runs only take for the negative class the negative instances that are positive, at least, in one of the other categories. In this way, we have tried to keep in the training set the most discriminative instances.

Model parameters to select by cross validation in the MTL approaches are the number of hidden nodes in the task-private and common parts (where, for simplicity, we assume that both parts have the same size), and coupling parameters ρ_{c} and ρ_{p}. In the STL model we only need to validate the number of hidden nodes. Moreover, we use in all the MLP approaches the same linearly decreasing input and output learning rates, and an appropriate

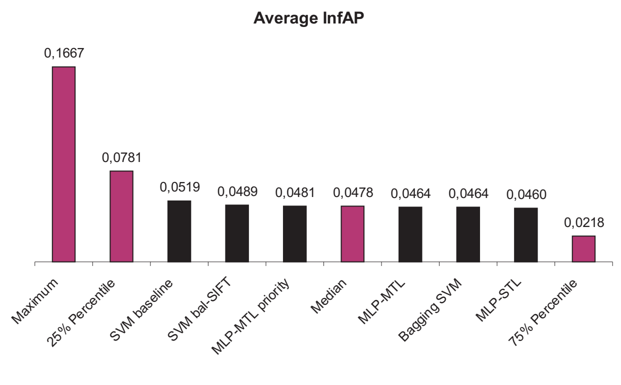


Fig. 4. Average InfAP of all our submitted runs. Results are shown in comparison with the best performing run, median, and 25th and 75th percentiles.

number of epochs according to a previous convergence analysis.

4. PERFORMANCE EVALUATION

In this section we evaluate the performance of all six runs in terms of InfAP. To start with, Figure 4 illustrates the achieved InfAP averaged over the 20 high-level concepts. The best result, median and the 25th and 75th percentiles are also shown in the figure as a reference for comparison. We can see that all runs achieve very similar results, performing in all cases very close to the median of all submitted runs.

Regarding the SVM based runs, it was the baseline approach which got the highest InfAP, although differences in performance among these three runs are almost negligible. Although we originally expected an improved performance when using the bal-SIFT region descriptors, our validation results (in terms of classification accuracy) already suggested that a very small difference existed between classifiers using un-SIFT and bal-SIFT. This is also obvious when looking at the upper panel of Figure 5, which shows per class InfAP. We can see that bal-SIFT features usually lead to a sligthly worse performance, class *Boat ship* being the only exception. We are currently thinking about some other ways of creating balanced codewords that can improve the performance of the classifiers.

Bagging of SVM was also unsuccessful to improve the performance of the baseline. Although improvements are ocassionally encountered in some categories, it is the very poor performance of this approach in the category *Dog* which is actually degrading somehow significantly the performance of the run with bagging when compared with the SVM baseline.

Regarding the MLP based approaches, the two approaches with MTL were able to improve the average performance of the Single-Task run, although differences in performance are too small to consider these results conclusive. Selective MTL with priority generally performed better than MTL with equally significant tasks, both in average, but also when looking at per class results (lower panel of Figure 5).

As previously explained, we think that MTL performance could be improved by a previous task-level clustering of similarities, based on co-ocurrences of high-level features. We will consider this possibility for next years, as a way to prevent negative transfer of knowledge among classes which are not related and, therefore, should not be learnt together.

To conclude the section, we have compared in Figure 6 the results of the best performing SVM and MLP based runs. SVM improvements are clear in classes *Dog, Two People, Bus, Driver, Telephone* and *Mountain*, and are responsible for the higher average InfAP. In contrast to this, MLP-MTL with priority is only clearly superior to the SVM baseline in two classes: *Boat ship* and *Airplane*. It would be interesting to analyze the performance of Multi-Task schemes using other component networks which are more powerful and easier to train than MLPs, and we consider this as an interesting line for future research.

5. CONCLUSIONS AND FURTHER WORK

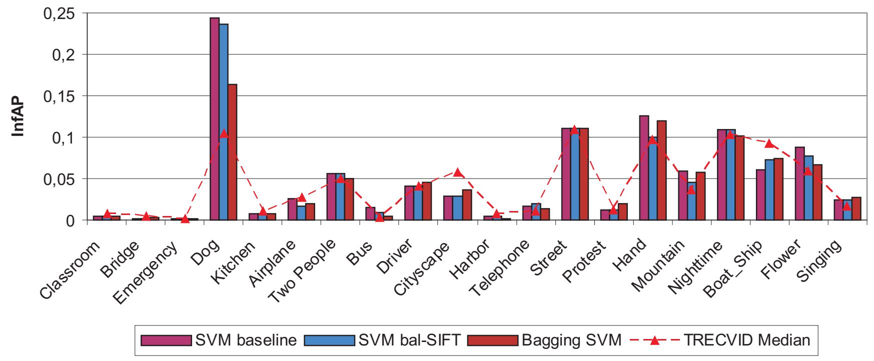
In this document we have presented the architecture of our system for high-level feature extraction, explaining which are the modules which have been implemented so far. Experiments this year have been focused on SVM networks and Multi-Task learning. We have submitted 6 runs based on these technologies, achieving in all cases average performances which are around the median of all submitted runs.

For next years, we will consider the extension of these schemes, further exploring the possibilities of Multi-Net systems and Multi-Task learning. Feature extraction and selection will also be explored as a way to reduce the complexity of the system and the training of the classifiers. Finally, we will also work on the development of new low-level descriptors with the goal of reducing the semantic gap with the high-level concepts.

6. REFERENCES

[1] S. Ayache and G. Quénot, "Video Corpus Annotation Using Active Learning," in *Proc. 30th European Conference on Information Retrieval (ECIR'08)*, Glasgow, UK, 2008, pp. 187–198.

[2] J. M. Martínez, "Overview of the MPEG-7 Stan-



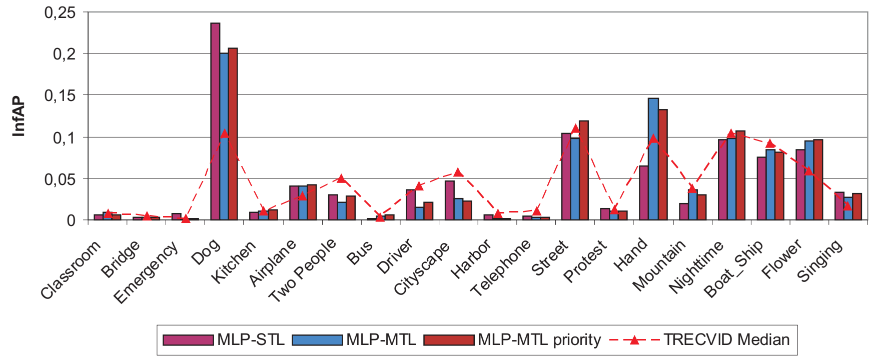


Fig. 5. InfAP per class of all submitted runs. TRECVID's median is also included as a reference for comparison.

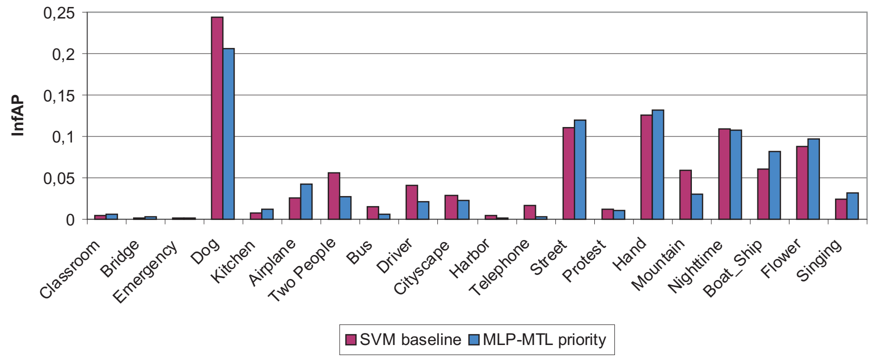


Fig. 6. InfAP per class of best performing SVM and MLP based runs.

dard v. 10.0" ISO/IEC JTC1/SC29/WG11 N4674, 2002.

[3] J. Huang, S. Ravi Kumar, M. Mitra, W.-J. Zhu, and R. Zabih, "Image Indexing Using Color Correlograms," in *Proc. CVPR 1997*, Washington, DC, 2007.

[4] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural Features for Image Classification," *IEEE Trans. Systems, Man and Cybernetics*, vol. 3, pp. 610–621, 1973.

[5] B. S. Manjunath and W.Y. Ma, "Texture features for browsing and retrieval of image data," *IEEE Trans. Pattern Analysis and Mach. Intell.*, vol. 18, pp. 837–842, 1996.

[6] K. Mikolajczyk, et. al., "A Comparison of Affine Region Detectors," *International Journal of Computer Vision*, vol. 65, no. 1/2, pp. 43–72, 2005.

[7] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *Intl. Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[8] T. Hofmann, "Probabilistic latent semantic analysis," in *Proc. Uncertainty in Artificial Intelligence UAI'99*, 1999, pp. 289–296.

[9] D. Liu and T. Chen, "Semantic-Shift for Unsupervised Object Detection," in *Proc. 2006 Conf. on Computer Vision and Patt. Recognition Workshop'*, *CVPR'06*, Washington, DC, 2006.

[10] C. Chang, C. Lin. LIBSVM: a library for support vector machines (2001). Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[11] Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2):123–140.

[12] S. Thrun and L. Pratt, *Learning to Learn*. Norwell, MA: Kluwer, 1998.

[13] R. Caruana, "Multitask learning", Ph.D. dissertation, School of Computer Science, Carnegie Mellon University, Pittsburg, PA, 1997.

[14] R. Caruana, "Multitask learning", *J. Machine Learning Res.*, vol. 28, pp. 41-75, 1997.

[15] P. García-Laencina, A. R. Figueiras-Vidal, J. Serrano-García, J. L. Sancho-Gómez, "Exploiting Multitask Learning Schemes Using Private Subnetworks". In: Proc. IWANN (2005) 233-240

[16] D. Silver and R. Mercer, "Selective functional transfer: Inductive bias from related tasks", in *Proc. IASTED Int. Conf. Artificial Intelligence and Soft Computing*, Cancun, Mexico, 2001, pp. 182-189.