An Indexing Approach for Speeding-Up Image Classification

Rahul Jain, Sudha Praveen M., Pramod Sankar K., C. V. Jawahar *
Center for Visual Information Technology
IIIT-Hyderabad, India
jawahar@iiit.ac.in

ABSTRACT

One of the most common computer vision tasks is that of recognizing the category of objects present in a given image. Previous work has mostly focused on building accurate classifiers based on carefully selected features. Classification is often carried on individual test images, while most of the practical situations, such as webscale image indexing, demand the simultaneous classification of a large collection of images. This is especially true for real-world datasets, that already contain numerous un-indexed images and videos. In this paper, we work towards developing a computationally efficient approach towards object recognition, that is inspired by retrieval schemes. We perform an offline indexing of the features from the collection, so that the classifier only needs to work on a small subset of the entire feature set. Over a set of 2 Million features extracted from 7000 images, classification against 5 object categories using a standard SVM would require more than 260 hours. Over the same test case, the classification time using our indexing based approach is reduced to less than 13 hours. The compromise on the accuracy is less than 7% for the 20X speedup achieved.

1. INTRODUCTION

Computer Vision problems are characterized by humongous amounts of visual data that need to be understood. It is an accepted fact that images and videos are being generated at a much faster rate than they could be processed. For example, more than 24 hours of videos are being uploaded on YouTube every minute. But, there are not many computer vision applications that can process videos at that speed while still providing accurate understanding. Search and retrieval from such large collections of visual data is a very challenging problem.

Popular retrieval systems depend on contextual textual information such as filename, tags and surrounding text.

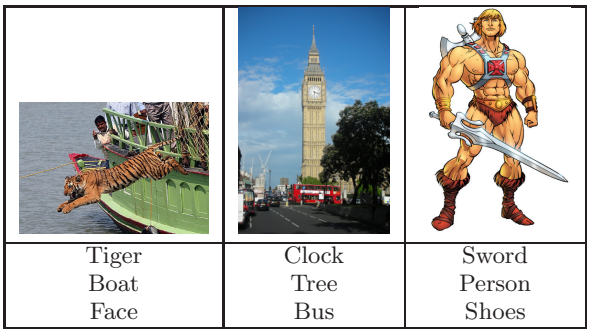

Figure 1: The problem addressed in this paper is to recognize object categories in unconstrained images. The images might contain multiple objects from different categories, all of which should be identified. Such object labels enable building better image retrieval systems, that can search based on image-content (rather than keywords surrounding it).

However, such information is not always available and is expensive to generate manually. A true retrieval system should have a certain understanding of the *content* of an image. Classical content-based image retrieval (CBIR) mostly matched images based on features such as color, texture [26], SIFT [25], etc. Instead, in this paper, we shall look at the problem of retrieval through recognizing object categories within images.

There has been significant progress in object recognition over the last few years [10, 16, 29]. Image recognition is typically posed as a classification problem. Features and classifiers are carefully selected and trained for this purpose. Use of dense SIFT-like descriptors and SVM-like discriminative classifiers have become the widely accepted candidates for visual recognition. Much of the previous work in visual recognition has focused on the *training* phase of the system. However, little work focuses on scaling these classifiers to large test-sets.

There are two common approaches to speeding up object detection: the first is a cascade based approach [30], while the other is by using random forests [3]. Cascaded classifiers were popularized by Viola and Jones [30], which is a modification of the AdaBoost framework, applied for the task of face detection. Each weak classifier of the cascade depends on only one haar-like feature, hence simultaneously acting as a feature selection procedure. Zhu *et al.* [33] integrated classifier-cascades with the Histograms of Oriented Gradi-

*Corresponding author

ICVGIP '10, December 12-15, 2010, Chennai, India

ents (HoG) [7] representation to build a faster pedestrian detector. Similarly, Vedaldi *et al.* [28] uses a cascade of classifiers with progressively increasing complexity, for generic object detection. The first stage involves a linear SVM, which rejects a large number of negative windows; followed by an Additive-SVM and a non-linear Kernel SVM. Each stage of the cascade tests a decreasing number of windows with an increasingly strong classifier. The major drawback of cascade-based approaches is that one cannot recover from the mistakes committed early in the cascade. If the first stage wrongly rejects a window, there is no mechanism to reconsider that window with another classifier. The classifiers using a Kernel-SVM invariably performs better than those using a cascade. In short, cascades with weak classifiers are not good enough for complex concepts, while those with stronger classifiers are computationally prohibitive.

The second approach of random forests [3, 17], uses multiple decision trees which typically use simple decision functions at each node. Training is performed by maximizing information gain on a random subset of the data, with the posterior probabilities for each class stored in the leaf nodes. During testing, each feature is run through all the trees, the result being an average of the posteriors. Due to simple node tests, the computation required for random forests is much less compared to SVMs, with comparable performance [31]. However, they have the disadvantages of over-fitting and lack of a principled method to tune and improve their classification performance.

The speedup achieved by previous methods, is only through time saved in the classification module, which typically weakens the classifier. If one assumes that a single classification operation takes x seconds, a naive method of recognizing from N images would require $N \cdot x$ seconds. The savings achieved from reducing x would not be sufficient, since the number of images to be processed N is still very large.

In this paper, our objective is to perform the classification in far less than linear time, in number of images. Our interest is in recognising objects, over a large number of images in reasonable time. A general example of the task is shown in Figure 1. Such recognition of objects in an image would aid in annotation of images with meaningful concepts. The annotated images lend themselves to be easily indexed for semantic image retrieval. We apply our framework toward annotating a collection of 7,000 images from the PASCAL VOC 2009 dataset.

We demonstrate that, over a dataset of 2 Million samples, our method performs classification in 12.5 hours, as compared to more than 260 hours when using standard classification approaches. Our approach is applicable to any given classifier design, with no modifications required for the classifier itself. In other words, our method allows designers to not compromise on classifier training in order to speed it up in test conditions. We use a novel approach that is inspired from text retrieval and from its application to object retrieval. Our method is applicable to problems where a large number of images need to be classified against many classifiers, such as in large-scale image/video annotation, indexing and retrieval.

The paper is organized as follows. In the next section, we shall take a brief look at the related work to this paper. We present our framework in Section 3 and an efficient implementation of it in Section 4. The implementation details and results are provided in Section 5. In Section 6, we discuss the properties of our novel framework, and its possible applications to other computer vision problems. We conclude in Section 7.

2. RELATED WORK

There has been significant previous work regarding object detection and recognition [8]. The recognition process consists of two steps: feature extraction and classification. Features can be either sparse representations based on interest points, or a dense description which incorporates all the information from each window. Popular sparse features are a histogram of oriented gradients based descriptor for SIFT [18] or Harris-Affine [20] interest points. The image is represented as a *set* of these sparse features. On the other hand, with dense representations, the features are extracted from the information in a given scanning window. A commonly used dense descriptor is based on Haar wavelets [30], which are a set of basis functions encoding the difference in color pixels between adjacent regions. They are popular due to the efficient computation using Integral images, as well as their power to encode visual concepts.

Dense descriptors are directly used for classification with either a discriminative classifier (such as a Neural Network or SVM), or using AdaBoost. But, matching of sets of features is typically more computationally expensive than matching features in a vector space. There are two approaches that are generally used to speed up matching sparse features. The first approach performs vector quantization of features [25], and each feature is represented as belonging to one of the quantized bins. The image is then represented as a histogram of occurrences of features belonging to each of the bins. Such histograms can be easily matched using standard distance measures. The second approach bins the features into hierarchical histograms [11], which are then matched using weighted histogram intersection. The resulting pyramid match kernel can be directly plugged into a typical SVM framework.

The bottleneck however, is the need to classify features using an SVM, which in itself is computationally expensive. The time required is of the order $O(|Images| \times |Dimensions| \times |SVs|)$. A novel speedup of SVMs proposed by Zhang *et al.* [32] combines the speed of Nearest Neighbor classifier, with an SVM being used only along class boundaries. Maji *et al.* [19] introduced a speed-up for a set of non-linear kernels which makes the classification independent of the number of support vectors. With linear kernels, the complexity becomes independent of the dimensions, making it dependent only on the number of images/windows. Further speed up can be achieved using the branch-and-bound framework of Lampert *et al.* [15]. In this approach, sets of windows are evaluated for the highest score either of the windows could receive. The search continues by splitting the window set along the larger coordinate, and it terminates when a rectangle has a quality score that is atleast as good as the upperbound of all other candidates.

All the above methods have focused on speeding up the classifier, but the best speedup in still linear in the number of images/windows. In this paper, our aim is to perform classification of test data in sub-linear time. The closest work to ours is the work of Lampert [14]. They combine the aspects of image retrieval with object localization by building a two-layer branch-and-bound scheme which splits the search both across the images and within each image.

3. THE SOLUTION

3.1 Problem Description

Given a set of images, our goal is to label the images with the names of objects that it contains (from a given set of object categories). Such a labeling would be very useful to annotate images for subsequent retrieval. Similar tasks were posed by the Caltech-256 [12] challenge. In the Caltech dataset, images contained only one object, belonging to one of the 256 categories.

A more difficult dataset is the real-world images of PASCAL VOC [9] challenge, which we shall work with in this paper. In the VOC dataset, images are obtained from Flickr, which contain a large variety in the objects present in them and their spatial configurations. Since the locations of the objects are unknown, the image needs to be sampled at various spatial locations across different aspect ratios. This is typically performed using the sliding-window techniques, by evaluating a large number of candidate windows that are distributed over scale and space.

However, unlike the problem of object detection, where the object needs to be localized, in our case we are only interested in the presence or absence of the object. Instead, we randomly sample windows from the test image collection, typically about 300 windows per image. The assumption is that if sufficient features fall into the window, they could provide enough evidence for the presence of that object. Given such randomly sampled windows from the image collection, the problem is to label the windows if they contain a particular object.

3.2 Recognizing in Image Collections

Typical image classification problems treat each image as an individual test element. The statistics of the test datasets are generally ignored. It is this *isolated test-case* perspective that causes the large computational expense required by these techniques. Our solution is built on two important observations. The first observation is that a large number of windows within an image are very similar, especially since they overlap in scale and space. This is also true across a collection of images; there will be similar windows from different images whenever they contain similar objects. For example, consider an image that contains a region depicting *grass*. The sliding windows from this region would be very similar. In case other images in the test set also contain a region of grass, the windows from that would visually overlap across the images.

The second observation is that, in presence of visual similarity, such windows would be very close in any reasonable feature space they are represented in. Consequently, the classifier scores of these windows would also be very close. This is true for most classifiers which output a real-valued score, the classifier score is assumed to change smoothly across the feature space. This can be proved by disproving the contrary: any classifier which gives highly varied scores for windows that are close in the feature space, is very sensitive to noise in feature extraction or classifier training. Since standard features and classifiers have sufficient noise-tolerance, we believe that similar windows will most probably have similar scores.

This fact is reflected in the statistical evidence of SVM-score variation with respect to feature distances, as shown in Figure 2. A Million random feature pairs were evaluated against an SVM trained for aeroplane (see Section 5 for details). The distances between the SVM scores are plotted against the distances between the features. The plot shows that SVM scores are highly correlated with the feature distances; features that are close to each other typically have very similar classifier scores. It can be seen that SVM scores for features lying at a distance of 0.2 are very close, corresponding to the same classification result for most practical purposes.

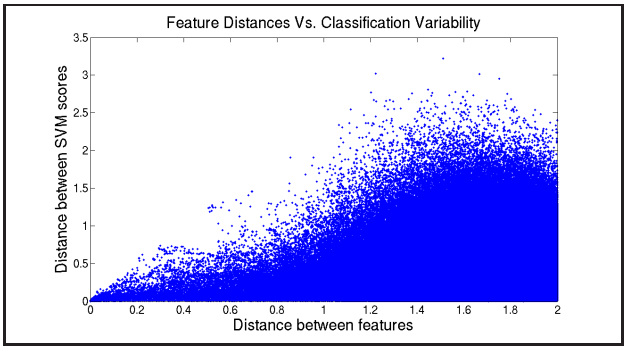

Figure 2: Statistical variation of SVM scores against distances between corresponding features. It is clear that when features are close the SVM scores are similar as well. With increasing feature distances the SVM scores vary widely.

Putting these two observations together, given a set of windows that are similar to each other, it suffices to classify only a few of them; the classifier score can then be propagated to the rest of the windows. This premise will be the basis for our proposed indexing-based image classifier.

3.3 Finding Similar Windows

In our approach, before a classifier is applied to detect an object, we need to identify the windows which are similar across the collection. This is a computationally expensive task because of the number of windows that are typically generated from an image collection is quite large. The VOC 2009 test set, for example, consists of 7,000 images resulting in more than 1.5 Billion windows. Finding sets of similar windows in such a huge collection is a significant task.

This challenge is very similar to the image/object retrieval from videos. In that problem, each image is represented as a set of local feature descriptors. Given an image region, similar regions across the video needed to be retrieved quickly. This was successfully addressed by the Video Google [25] approach. Local feature descriptors are vector quantized using K-Means clustering. Each feature is represented by the cluster number it falls in. The quantized features were then indexed which allowed for super-fast object retrieval.

Following this line of thought, image-windows could also be vector quantized. Such quantization is also referred to as building a *vocabulary*, due to its analogy with text retrieval. The criterion for building this vocabulary should be to cluster windows from the same object together. It means that the precision of the clusters needs to be high, at the expense of having multiple clusters corresponding to the same object. Only in such a clustering, will the propagation of classifier score across windows becomes acceptable. To achieve high precision clustering the number of clusters needs to be quite

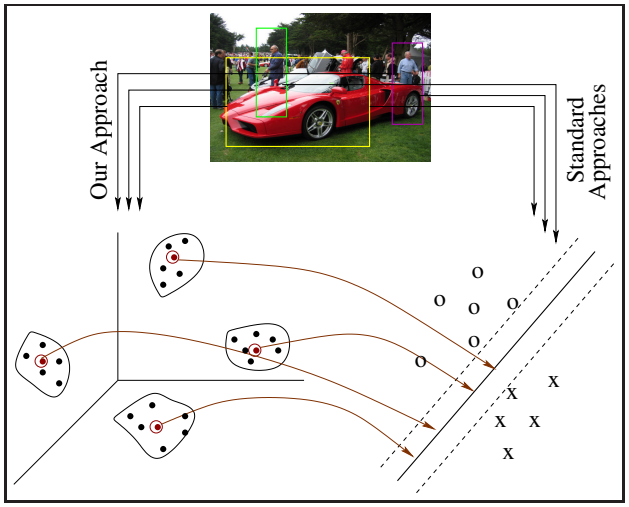

Figure 3: A simplified depiction of our framework. Standard approaches directly classify the features from image windows against a classifier. In our approach, we add an intermediate step of indexing (based on feature clustering). As long as all the features in a cluster belong to the same *concept*, it suffices to classify only the centroid of the cluster, resulting in significant speedup.

high.

Building large number of clusters from even larger feature sets is computationally expensive in itself. There are two directions to make large scale clustering computationally efficient: i) Hierarchical K-Means (HKM) [22] and ii) KDTrees [23]. In this paper we shall explore the HKM direction, leaving KDTrees for future work.

3.4 Hierarchical K-Means

We shall briefly explain HKM here for completeness. In the first stage, the windows are clustered into a small number, say B clusters. Each of these clusters is in-turn clustered into B clusters, resulting in B^2 clusters at the second stage. This process is repeated up to a certain depth D so that there are B^D clusters are formed. Since the clustering process can be depicted as a tree, it is typically referred to as a *vocabulary tree*, with a branching factor B whose leaf nodes are the clusters we seek. The depth D of the vocabulary tree is atleast $D = log_B(N)$. To build the entire HKM tree for a dataset of size N requires $O(N \cdot B \cdot D)$ time while standard K-Means would require $O(N \cdot B^D)$. For $N =$ a million features, on a modern desktop processor the difference is 31 yrs for K-Means vs 13 hrs for HKM.

This process significantly speeds up not only the clustering of data, but also the lookup of windows into the appropriate clusters. Given a new window, to find out which cluster it belongs to, it takes $B \cdot D$ comparisons unlike traditional K-Means which would take B^D comparisons. For example, if $B = 10$, $D = 6$ then there are a million leaf nodes. K-Means requires 10^6 (a million) comparisons while HKM only needs 60 comparisons.

3.5 Indexed Object Recognition

The indexing scheme can now be used to significantly speed up the object recognition process. A schematic of our framework is presented in Figure 3. Randomly sampled windows from the image collection are clustered using the HKM. The architecture of the indexing scheme is generally limited by the visual variety in the data. Given a fixed indexing schema, the number of features remains a *constant* regardless of the number of test data. Once clustered the centroids of the leaf nodes are classified using a set of discriminative classifiers such as SVMs. The label of the centroid is then propagated to the rest of the cluster; which is in-turn used to label the image itself.

4. SPEEDING UP WITH GPU

The process of indexed object recognition can be further speeded-up by exploiting the inherent data parallelism. This can be achieved by utilizing the Graphics Processor Units(GPU), that have recently found profound use in large-scale general purpose computation. New-generation GPUs have a many-core architecture (often hundreds of cores), and support running thousands of threads in parallel. Though motivated primarily to support real-time shading and local illumination computations, the programmable components of the GPU allow the large number of computation units to accelerate general purpose applications. Many algorithms in pattern recognition and machine learning have been speeded-up using GPUs in the recent past, such as Neural networks [1], SVMs [6], Decision trees [24], etc. GPUs have allowed the classification on large data sets to become feasible.

The popular software architecture to exploit GPUs is the NVIDIA CUDA [2] SDK, which is essentially a wrapper for C/C++. CUDA allows C functions to be executed multiple times, by multiple threads, on multiple GPUs. These functions are called kernels, which are easy to instantiate. A thread can execute a single kernel at any given time. Multiple threads are grouped in blocks and multiple blocks are grouped in grids. The number of blocks and threads that can run simultaneously is limited by the number of streaming processors in the GPU used. We use the NVIDIA Tesla computer which contains four GPUs, each allowing upto 128 threads to execute in parallel.

The clustering algorithm consists of two steps: i) finding the mean of a given cluster initializations, ii) reassigning the points to their closest cluster. Both these steps can be parallelised, the first step across the clusters and the second across the points. Further, in a hierarchical setting, the points being clustered to expand one node in the tree are independent of those in another node. This allows us to easily map the HKM building process onto the GPU architecture.

However, the number of points that can be clustered in each step is limited by the number of features that can be stored in the global memory of the GPU. Over our hardware, this limit was 4GB, accommodating upto 200K features in one instance. The speed-up is also limited by the time required to transfer the data between the CPU and the GPU, which is a non-trivial portion of the compute time. The time to build a HKM using different methods, over a set of 200K features is given in Table 1. HKM on GPU gives us about 24X speed up as compared to that on CPU. This efficiency from GPU implementation comes at no loss of performance.

5. IMPLEMENTATION DETAILS

Our dataset comes from the VOC2009 training set, which

K-Means (CPU)	30 Hours
Hierarchical K-Means (CPU)	10 Hours
Hierarchical K-Means (GPU)	25 Minutes

Table 1: Computational costs for clustering a set of 200K features

contains about 7000 images. The categories of interest is the set: Aeroplane, Bus, Car, Cat, Person. Features are extracted from about 2 Million randomly sampled windows of this image collection.

5.1 Feature Extraction

We extract dense SIFT descriptors from a grid of points with spacing of 5 pixels. These SIFT features were vector quantized in keeping with the Video-Google approach. A large set of randomly chosen SIFT features are quantized by using a GPU version of K-means, into 300 clusters. After the clusters are obtained, each feature from the dense SIFT computation is assigned to the closest cluster or visual word.

A given image or window of the image is now represented as a histogram of the number of occurrences of the visual words in an image. Since the spatial arrangement of the features is ignored, and only their occurrence counted, this representation is called a *bag-of-words*, similar to the terminology in text retrieval. A certain amount of spatial information can be preserved using the Pyramid Histogram Of Words (PHOW) features. PHOW is computed at multiple levels for each given window. At the first level, the standard BoW of the window is computed. At the next level the window is divided into 4 equal parts in length and breadth. BoW features of these 4 parts is computed separately and concatenated to the BoW computed at previous level to form a feature vector of length 1500(5 times that of BoW).

The computation of the PHOW features from an image is speeded up using the *Integral Image* approach [30]. 300 integral images are computed corresponding to each of the words in the visual vocabulary. The count of the features occurring in a given window is obtained by reading only four numbers per visual word. Given the coordinates of the image window, each integral images quickly gives the sum of the corresponding visual word in that bounding box and thus the PHOW histogram is computed easily. The PHOW features thus obtained is normalized using L1 norm.

5.2 Indexing the Features

The features extracted from the dataset are indexed using a HKM. The quality of clustering may be quantified using the cluster purity measure. Cluster purity measures whether a cluster is uncontaminated by windows belonging to different categories. However, this measure is biased towards small clusters. In the extreme case, if each word were its own cluster then the cluster purity would be 100%. This is undesirable and hence we measure the clustering quality using the F-*score* measure, which is popular in the document retrieval community. The F-score is given as

$$F_\beta = \frac{(\beta^2+1) * precision * recall}{\beta^2 * recall + precision}$$

To compute the precision and recall, we evaluate a set of $N \cdot (N-1)/2$ pairwise relationships for each of the windows. If a pair of windows belonging to the same category are clustered together, such a pair contributes to the true-positives(TP). If such a pair is assigned to different clusters, it would be counted as a false-negative(FN). When two windows are clustered together, but belong to different words it is considered as a false-positive(FP). Based on these computed values, the precision and recall are obtained as

$$precision = \frac{TP}{TP+FP}$$

$$recall = \frac{TP}{TP+FN}$$

In our F-score computation, we would like to penalize false positives more than false negatives. This is achieved by using a $\beta > 1$, typically 5. This gives more weight to precision or cluster purity. The optimal setting is identified by evaluating the clustering over a subset of the collection. The F-score of our clustering scheme, with a branching factor of 10, was found to be *0.54*, which is quite reasonable given our large dataset.

5.3 Classifier Training

We build a discriminative classifier to recognize the object categories of interest. A Kernel-SVM is learnt for each category to perform a one-vs-all classification. The most suitable kernel for the histogram features is the exponential Chi-Square kernel, which is given as

$$K(p,q) = e^{-\alpha \chi^2(p,q)}$$

$$where, \quad \chi^2(p,q) = \sum_{i=1}^{N} \frac{(p_i - q_i)^2}{p_i + q_i}$$

These classifiers were trained using the labeled positive examples from our selected dataset using the SVMLight package [13]. To improve the number of positive examples, and to obtain certain amount of robustness to pose variations, additional positive windows were obtained by flipping them about their vertical axis. Negative examples for the classifiers were comprised of the positive examples of other categories and 2000 windows with no object present in them.

The classifiers were tested for performance over our dataset of 2 Million windows. Performance analysis of the individual classifiers is given in Table 2 (top row). The best performing category was found to be that of *Person* and the poorest performing category was *Car*. These results are consistent with the work of Vedaldi *et al.* [28], which is understandable since we use very similar features to them. The scores from each classifier were normalized to lie between $[0, 1]$. The category with the highest score is assigned to each window, if the highest score is greater than a predefined threshold. Confusions across the different categories can be seen in Table 3. One would ideally want a block diagonal confusion matrix, but it is apparent that our performance is quite satisfactory.

5.4 Object Recognition from Indexed Windows

The final step of the implementation involves classifying the centroids of the HKM clusters against the classifiers. Following this classification, all the windows indexed to each cluster are assigned the label of the centroid. While the time required to classify each feature against the 5 SVMs is more than 260 hours, the time required to classify only the centroid of each cluster is merely 13 hours.

Category	Aeroplane	Bus	Car	Cat	Person
SVM	78	84	75	83	92
Indexing + SVM	71	78	66	76	87
Introduced Error	7	6	9	7	5

Table 2: Classifier performance without and with using feature indexing. The loss in performance is acceptable, especially for an image retrieval application. The speedup achieved compensates for the average loss in performance of less than 7%.

Category	Aeroplane	Bus	Car	Cat	Person
Aeroplane	78	8	10	2	2
Bus	5	84	6	1	4
Car	3	9	75	5	8
Cat	3	3	1	83	10
Person	1	1	1	5	92

Table 3: Confusion Matrix between the various object classes from the learnt SVMs.

The performance of this recognition propagation is shown in Table 2. Compared to a standard SVM, the indexing scheme results in an average loss of accuracy of about 7%. It is clear that the loss of accuracy by using the indexing scheme is quite acceptable for the speedup we obtain. The loss is least for the category of Person, which means that our approach results in a faster pedestrian detector with little loss of performance.

Each image is scored across the category by considering the vote from each window selected from the image. The votes are normalized by the number of windows picked from the given image. For a given category query, images are ranked based on the normalized scores. Example images retrieved for different object categories are given in Figure 4. Some examples where errors are introduced by the indexing process are shown in the middle column of Figure 4. Most of these errors might be avoided by splitting the nodes further along the HKM tree. Further, as we see in Figure 4 (right column) there are many windows which were misclassified by the regular SVM (as well as our indexing based SVM). A large percentage of these errors seem to occur because the sampled window does not capture sufficient detail/features of the particular object. Using a denser sampling of windows (still less than full sliding window), one could easily obtain better inferences over these objects.

6. DISCUSSION

One of the key characteristic of our algorithm is that it isolates the indexing schema from the classification itself. It also means that multiple classifiers can simultaneously process the same test data, with all classifiers benefiting from the time saving of applying the classifier over the indexed data. The clusters that the classifier makes a mistake with, provides many hard-examples to retrain the classifier with. Our proposed scheme also performs mutual exclusion inherently, which is a post-processing step of typical object detectors. By assigning only one label to each cluster we can enforce the constraint that a given window would not be a candidate for another visual category.

Our approach easily lends itself to be applied further in many ways:

1. The indexing based classification scheme is quite suitable to sliding window techniques for object localisation. With a large number of features being extracted from a small image region, such features easily cluster together, hence requiring lesser classifications per image.

2. Recognising objects from large collections of images such as PASCAL VOC can be made very efficient. These datasets typically contain many similar objects across the collection, which can be easily exploiting by the clustering step.

3. Better classifiers such as those based on Multiple Kernel Learning [27] could be built, without compromising on the accuracy for the sake of efficiency. Multiple features such as those based on PHOG [5] and self-similarity [28] could be combined to improve classification accuracy.

4. Further, many other indexing schemes could be explored for reducing the computational complexity of classification. One could possibly look at Locality Sensitive Hashing [4], KD-Trees [21] (and other tree structures), etc.

In the problem of detection, at first glance, it could appear that clustering a large set of features would be more expensive than classifying them directly. However, our framework allows the indexing to be performed incrementally. In incremental indexing, we begin with a sizable set of data and build a HKM on it. In the next step, unseen data is looked-up against this HKM and the features from new images are assigned to their closest clusters. This is an $O(B \times D)$ operation. Now, only those nodes of the HKM need to be expanded, that have a large variety of windows in them. Once the clusters are built, the classification can be quickly performed.

7. CONCLUSIONS

In this paper we have presented a novel approach to perform classification in large scale datasets. Instead of being burdened by the scale of realistic datasets, our approach actually benefits from that very aspect. By deriving inspiration from text and image retrieval, we index features in the offline phase for quick online classification. The efficiency is further boosted with the use of GPU computing wherever possible. We demonstrate the technique over the problem of object recognition in real-world image sets. The results show very little loss of performance for large gains in terms of computational expense. In future work, we shall apply the framework on a much larger dataset, possibly for the task of object localization.

8. ACKNOWLEDGEMENTS

Pramod Sankar would like to thank Andrea Vedaldi for an inspiring talk on object-detection at IIIT-Hyderabad. Rahul Jain acknowledges P. J. Narayanan for helpful discussions regarding GPU implementations of vision algorithms.

9. REFERENCES

[1] Neural Networks on the GPU: http://leenissen.dk/fann/html_latest/files2/gpu-txt.html.

[2] nVIDIA CUDA at: http://www.nvidia.com/object/cuda_home_new.html.

[3] Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *Neural Computation*, 9(7):1545–1588, 1997.

[4] A. Andoni, M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. *Nearest Neighbor Methods in Learning and Vision: Theory and Practice*, 2006.

[5] A. Bosch, A. Zisserman, and X. Munoz. Image classification using random forests and ferns. In *Proc. ICCV*, 2007.

[6] B. Catanzaro, N. Sundaram, and K. Keutzer. Fast support vector machine training and classification on graphics processors. In *Proc. ICML*, pages 104–111, 2008.

[7] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. CVPR*, pages 886–893, 2005.

[8] M. Enzweiler and D. M. Gavrila. Monocular pedestrian detection: Survey and experiments. *IEEE PAMI*, 31(12):2179–2195, 2009.

[9] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2009 (VOC2009) Results. http://www.pascal-network.org/challenges/VOC/voc2009/workshop/index.html.

[10] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. In *Workshop on Generative-Model Based Vision*, 2004.

[11] K. Grauman and T. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *Proc. ICCV*, 2005.

[12] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology, 2007.

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

[14] C. H. Lampert. Detecting objects in large image collections and videos by efficient subimage retrieval. In *Proc. ICCV*, pages 987–994, 2009.

[15] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *Proc. CVPR*, 2008.

[16] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proc. CVPR*, pages 2169–2178, 2006.

[17] V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *IEEE PAMI*, 28(9):1465–1479, 2006.

[18] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[19] S. Maji, A. C. Berg, and J. Malik. Classification using intersection kernel support vector machines is efficient. In *Proc. CVPR*, 2008.

[20] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool. A comparison of affine region detectors. *IJCV*, 65(1/2):43–72, 2005.

[21] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *Proc. VISAPP*, 2009.

[22] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Proc. CVPR*, pages 2161–2168, 2006.

[23] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proc. CVPR*, 2007.

[24] T. Sharp. Implementing decision trees and forests on a gpu. In *Proc. ECCV*, pages 595–608, 2008.

[25] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proc. ICCV*, pages 1470–1477, 2003.

[26] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE PAMI*, 22(12):1349–1380, 2000.

[27] M. Varma. Learning the discriminative powerinvariance trade-off. In *Proc. ICCV*, 2007.

[28] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *Proc. ICCV*, 2009.

[29] A. Vedaldi and S. Soatto. Relaxed matching kernels for object recognition. In *Proc. CVPR*, 2008.

[30] P. A. Viola and M. J. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. CVPR*, pages 511–518, 2001.

[31] J. Winn and J. Shotton. The layout consistent random field for recognizing and segmenting partially occluded objects. In *Proc. CVPR*, pages 37–44, 2006.

[32] H. Zhang, A. C. Berg, M. Maire, and J. Malik. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In *Proc. CVPR*, pages 2126–2136, 2006.

[33] Q. Zhu, M.-C. Yeh, K.-T. Cheng, and S. Avidan. Fast human detection using a cascade of histograms of oriented gradients. In *Proc. CVPR*, pages 1491–1498, 2006.

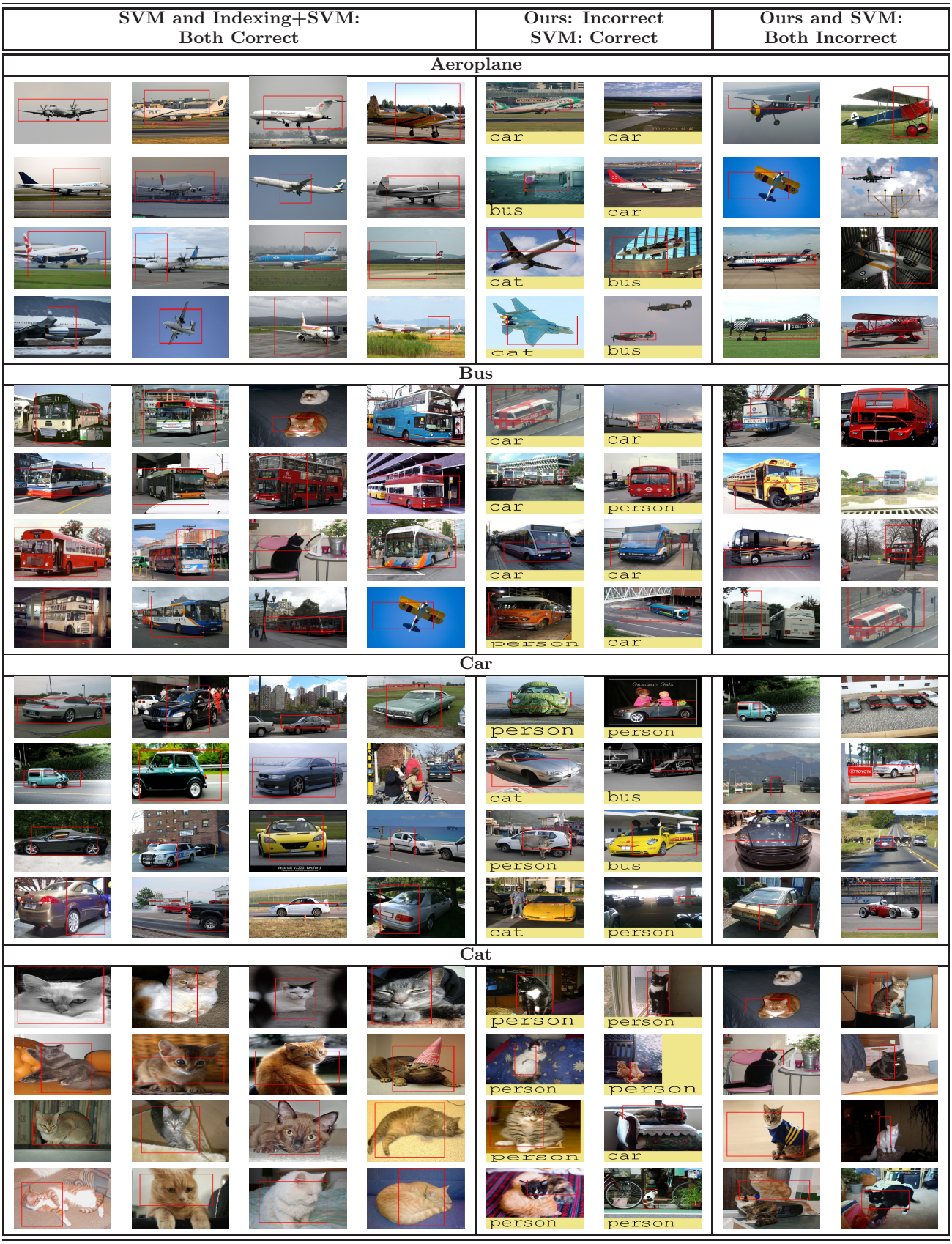

Figure 4: (left) Examples of images correctly recognised for a few categories. The window with the highest score for the given category is outlined in red. (middle) Examples of errors introduced by the Indexing scheme, over those windows where a normal SVM classified correctly. The erroneous label is given with the window that was mis-classified. (right) Windows that were mis-classified by both the regular and indexing based SVMs.