Scalable Object Detection by Filter Compression with Regularized Sparse Coding

Ting-Hsuan Chao, Yen-Liang Lin, Yin-Hsi Kuo, and Winston H. Hsu
National Taiwan University, Taipei, Taiwan

Abstract

For practical applications, an object detection system requires huge number of classes to meet real world needs. Many successful object detection systems use part-based model which trains several filters (classifiers) for each class to perform multiclass object detection. However, these methods have linear computational complexity in regard to the number of classes and may lead to huge computing time. To solve the problem, some works learn a codebook for the filters and conduct operations only on the codebook to make computational complexity sublinear in regard to the number of classes. But the past studies missed to consider filter characteristics, e.g., filters are weights trained by Support Vector Machine, and rather they applied method such as sparse coding for visual signals' optimization. This misuse results in huge accuracy loss when a large speedup is required. To remedy this shortcoming, we have developed a new method called Regularized Sparse Coding which is designed to reconstruct filter functionality. That is, it reconstructs the ability of filter to produce accurate score for classification. Our method can reconstruct filters by minimizing score map error, while sparse coding reconstructs filters by minimizing appearance error. This different optimization strategy makes our method be able to have small accuracy loss when a large speedup is achieved. On the ILSVRC 2013 dataset, which has 200 classes, this work represents a 16 times speedup using only 1.25% memory on single CPU with 0.04 mAP drop when compared with the original Deformable Part Model. Moreover, parallel computing on GPUs is also applicable for our work to achieve more speedup.

1. Introduction

In object detection, many works based on part-based model such as the Deformable Part Model (DPM) [8] have achieved high accuracy in recent years. However, long detection time has raised concerns on whether it is suitable for real applications. Most of the computing time of these methods comes from convolution operations between filters and images. Moreover, to overcome the deformation nature of objects, an object class often comprises 20 or even 100 more filters and such a huge number of filters will unacceptably prolong detection time. In general, object detection can be divided into two stages. The first stage collects object proposals, while the second stage uses a bank of filters to decide the class of each proposal. Trying to scale down the number of convolution operations, an efficient method has been proposed [18], which tries to reconstruct all filters with a fixed size codebook. Therefore, convolution operations are only conducted between the codebook and input image to make computational complexity not linear in regard to the number of classes. It proves that sparse coding can be a good method for efficient object detection.

We question that if sparse coding is still suitable for scalable object detection when the number of classes grows in the ILSVRC 2013 detection task. When a large speedup is required, it has to scale down size of codebook and use the codebook to reconstruct huge number of filters. However, sparse coding cannot fully reconstruct all filters with the small codebook and results in accuracy loss. It means that we need a method to reconstruct filters with limit budget (i.e., small memory consumption, small model and small computation) and achieve larger speedup. In object detection, filters are used to convolve with images to get score maps (Figure 2) and then leverage these score maps to produce reliable detection results which consist of several bounding box-label pairs. But as we know, sparse coding is inspired by the function of the human vision system and frequently used to reconstruct visual signals. Using sparse coding to reconstruct filters appearance will be the reason why accuracy drops with limit budget as confirmed by our experiments in Section 4.

To tackle the problem, we proposed Regularized Sparse Coding which is optimized for filter functionality[1]. The intuition is that we should reconstruct filter functionality directly rather than reconstruct filter appearance and ask it to perform like the original filter. In other words, understanding functionality of filter and codebook will be the

[1] In this work we call the ability of filter to produce accurate score map in object detection as *filter functionality*.

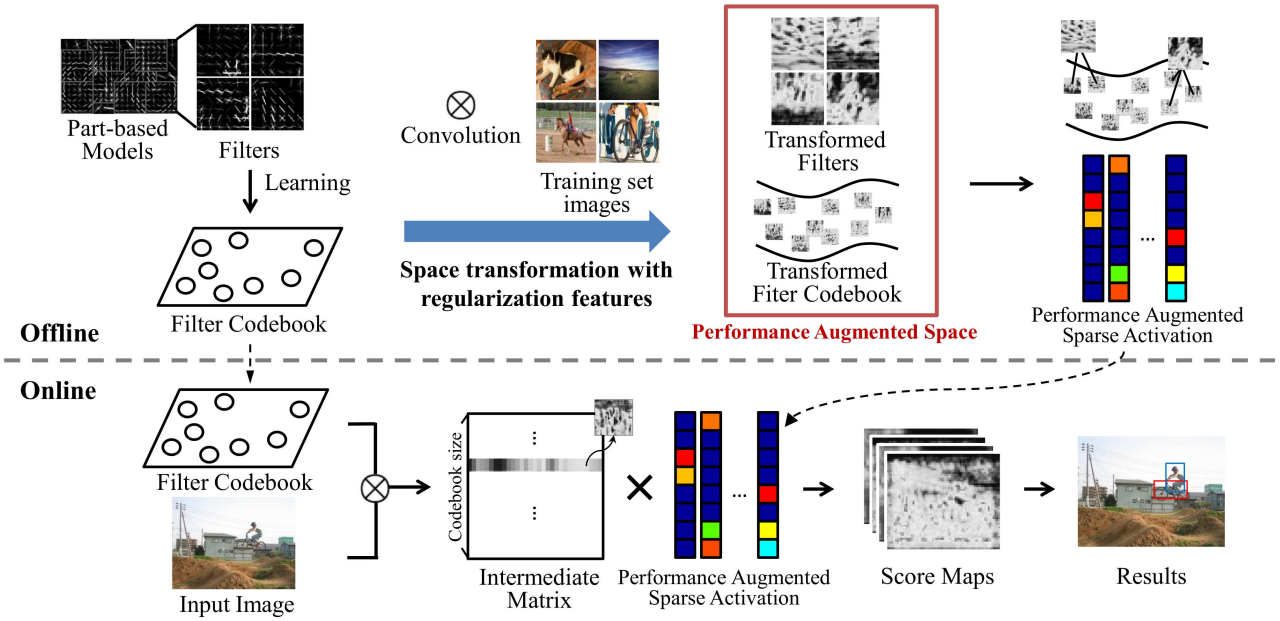

Figure 1. System architecture. In the offline stage, a bank of filters (classifiers) are collected from part-based models and used to learn a codebook. By conducting convolution operations with training set images, Regularized Sparse Coding transforms filters and the codebook into performance augmented space, where filter can match codewords which have similar responses, i.e., classification results. In performance augmented space, meaning of reconstruction is the same as minimizing score map error and we can learn performance augmented sparse activation which is optimized to reconstruct filter functionality and improve detection performance. In the online stage, convolution operations are only conducted between codebook and input image to produce the intermediate matrix. With the intermediate matrix and the performance augmented sparse activation, we can efficiently calculate score maps and produce the final result without conducting all convolution operations between input image and filters to get the score maps.

key to reconstruct filters with less budget and achieve a larger speedup. Our method first introduces regularization process. As shown in Figure 1, it transforms filters and codebook into performance augmented space by conducting convolution operations with regularization features. In the performance augmented space, a transformed filter is a set of classification results of the regularization features. Their positions in performance augmented space clearly represent their functionality, that is, the ability to produce score for classification. In other words, these regularization features regularize filters and codebook to show their functionality. The physical meaning of performance augmented space enforces filters to reconstruct functionality in it and improve detection accuracy. With this different optimization strategy, our method can fully reconstruct filter functionality and achieve a large speedup with negligible accuracy loss. To illustrate the effect of Regularized Sparse Coding, we show some score maps produced by Regularized Sparse Coding (details can be seen in Section 3.2) and sparse coding [18] in Figure 2. To prove our work's scalability regarding the number of classes and to ensure that we have the same or even better performance on a larger dataset with negligible mAP loss, we conduct experiments on the ILSVRC 2013 dataset which is thought to be the largest object detection dataset in recent years. In summary the main contributions of this work are:

- We propose Regularized Sparse Coding to reconstruct filter functionality which sparse coding cannot reconstruct successfully.
- We conduct experiments on several large datasets with up to 200 classes to prove scalability of our method.
- We achieve 16 times speedup using only 1.25% memory with less than 0.04 mAP drop compared to the original Deformable Part Model.

2. Related Work

In general, object detection can be reduced to proposal extraction and object classification. So, the computational complexity of object detection could be $\mathcal{O}(LC)$, where L is the number of proposals and C is the number of classes. These two stages and the impacts on scalability are discussed separately in this section.

2.1. Proposal Extraction

The computational complexity of object detection is linear in regard to the number of proposals. However, in the

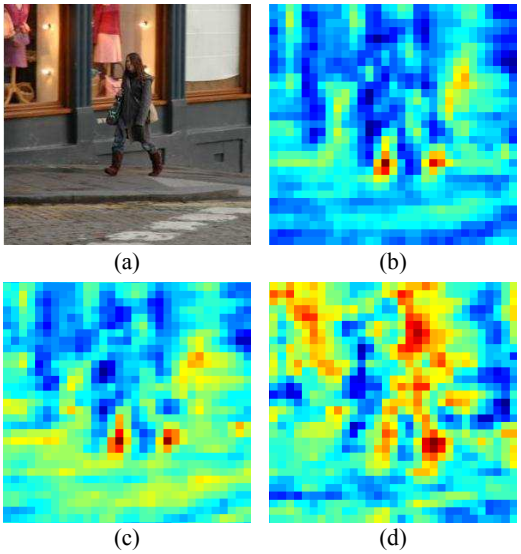

Figure 2. Comparison between Regularized Sparse Coding and sparse coding. A foot filter from person model is used to illustrate difference between the two methods. (a) Input image. (b) Score map produced by the original filter. (c) Score map produced by Regularized Sparse Coding. (d) Score map produced by sparse coding. Apparently, Regularized Sparse Coding can reconstruct filter functionality better, that is, it produces score map which is able to provide more accurate detection result than sparse coding. It shows that our method successfully makes reconstruction perform better on filters.

traditional method, proposals are extracted by sliding windows approach and the huge number of proposals results in very long computing time. From the famous cascade method [23], many works have focused on reducing the number of object proposals by utilizing weak classifiers. In [6, 13] the authors use the part hierarchies and spatial information to implement cascade object detection to reduce computation. Some works have discovered that most of the object classes have clear boundary and possess common attributes of objects. Using this information, they developed fast algorithms to extract class-generic object proposals and eliminate background proposals. Methods such as exploiting segmentation [21, 20], interest points [22] and salience information [1, 15] efficiently produce a small number of object proposals with high recall rate, losing only negligible accuracy. Our method works on the other stage, object classification, and has complementary advantages[2] with these methods to reduce computation even further for object detection.

2.2. Object Classification

With the exception of few object detection frameworks [2, 12], each object proposal represents a candidate bounding box for all classes, which means that each proposal should pass through all classifiers to determine its class. Several methods provide efficient ways to avoid this heavy computing process which has linear computational complexity in regard to the number of classes. [3] introduced a hash based method. They try to transform convolution operations into queries in several hash tables and reach almost class-independent computational complexity. However, memory usage is still linear in terms of the number of filters, which is a weak point in developing a scalable object detection system. [19] jointly trains object models by finding common features that can be shared across the classes. But this method needs to train all the classes again when a new class joins. Other methods such as [14, 18, 17] state that learning a sharable codebook for filters can be a solution to reduce computation without building hash tables and consuming lots of memory resources. However, these methods are still unable to solve the problem when the number of classes grows. Our method tries to fix this problem by introducing regularization process into sparse coding and reduces computational complexity from linear to sublinear in regard to the number of classes. In addition, our method can complement other methods which focus on reducing computation in the proposal extraction stage and parallel computation on GPUs to achieve more speedup.

[2]In [3], complementary advantages are shown helpful with acceptable accuracy loss.

3. Technical Details

In this section we describe a framework based on the Deformable Part Model (DPM) which uses Histogram of Oriented Gradients (HOG) as a feature type. Some other feature types will be able to adapt the framework as well because the optimization subject of our method is the weight trained by SVM. We use part filters in DPM as basic elements (in the following simply called filters). Since the number of root filters in DPM is only about 4% the number of part filters, the root filters will not be considered as target of reducing computation in order to simplify problems.

3.1. Sparse Coding

In order to decrease the number of involved filters when convolution operations take part in object detection, we can represent each filter as a linear combination of codewords of a predefined codebook. [18] has shown that learning based codebook construction methods perform better than matrix factorization methods, such as Singular Value Decomposition (SVD) method, for the filters. Here we quickly go through the process. First, we have filters $\mathbf{X} = \{X_1, ..., X_N\}$ which are collected from a set of part-based models trained on a chosen dataset. Next, we use these filters to learn a codebook $\mathbf{D} = \{D_1, ..., D_K\}$. Codewords of the codebook D can be thought as a set of basic elements consisting of edges and shapes. Redundant information of filters can be eliminated, reserving only informative

elements to reconstruct filters. The optimization method of sparse coding can be formulated as follows.

$$\begin{aligned} \min_{\alpha_{ij}, D_j} \sum_{i=1}^{N} \quad & \left\| X_i - \textstyle\sum_{j=1}^{K} \alpha_{ij} D_j \right\|_2^2 \\ \text{subject to} \quad & \|\alpha_i\|_0 \leq \epsilon, \forall i = 1, ..., N \\ & \|D_j\|_2 = 1, \forall j = 1, ..., K \end{aligned} \tag{1}$$

By leveraging the Orthogonal Matching Pursuit algorithm (OMP) [11], we can efficiently compute an approximate solution to avoid this NP-hard optimization problem. We also test a variety of sparse coding methods (e.g., [10, 4]) which do not have a notable difference from the OMP method. The original convolution process in object detection can be transformed into the following form:

$$X_i \approx \sum_{j=1}^{K} \alpha_{ij} D_j = \alpha_{\mathbf{i}} \mathbf{D} \tag{2}$$

$$\begin{bmatrix} \Psi * X_1 \\ \vdots \\ \vdots \\ \Psi * X_N \end{bmatrix} \approx \begin{bmatrix} \alpha_{\mathbf{1}} \\ \vdots \\ \vdots \\ \alpha_{\mathbf{N}} \end{bmatrix} \begin{bmatrix} \Psi * D_1 \\ \vdots \\ \Psi * D_K \end{bmatrix} = AM \tag{3}$$

In Equation 3 we denote the feature pyramid of an image as Ψ, $*$ the convolution operation, α_i the i^{th} filter's sparse activation, D_i the codeword of the learned codebook and X_i the filter. We can get the brief representation AM as the last term in Equation 3. A is a matrix of sparse activation and M is a matrix of intermediate representation. We illustrate these representation in Figure 1. Here we use DPM as our experiment target. In DPM scores of a single proposal can be calculated as

$$\begin{aligned} score(\phi) &= score_{root}(\phi) + \sum_{i=1}^{P} score_{part}(\phi + \delta_i) + cost(\delta) \\ &= score_{root}(\phi) + \sum_{i=1}^{P} \sum_{j=1}^{K} \alpha_{ij} (\Psi * D_j) + cost(\delta). \end{aligned} \tag{4}$$

In Equation 4, ϕ is the placement in a feature pyramid and δ represents displacements of the parts. Notice that we can calculate the matrix of intermediate representation M instead of calculating the score of each class separately for multiclass object detection. In other words, we can amortize the computation cost of the intermediate matrix. The original DPM requires Nd operations to compute N filters where d corresponds to the dimension of the filters, while our method requires only $Kd + N \|\alpha\|_0$. We can yield speedup

$$speedup = \frac{Nd}{Kd + N \|\alpha\|_0} \tag{5}$$

In Equation 5, N and d are fixed for a given dataset such that we should reduce K and $\|\alpha\|_0$ to yield more speedup.

3.2. Regularized Sparse Coding

The previous section has shown the capability of sparse coding, but here we provide some observations to help revise sparse coding into our method, which is called Regularized Sparse Coding.

First, filters in object detection system are the weights trained by SVM and we have known that image and SVM weight are very different in statistical characteristics. The sparse coding method simply takes filters as images, learns a codebook and use codewords of the codebook to reconstruct filters by minimizing L2 distance between original and reconstructed filters; that is, it minimizes difference of filter appearance. However, we should directly reconstruct filter functionality instead of filter appearance. Another problem is that due to the difficulty of some object classes in the given dataset and limitations of the learning algorithm, some of the classes with low mAP consist of very noisy filters and these noisy filters become a main source of error in reconstruction. To illustrate the idea more clearly, we describe how sparse coding and our method perform on a d-dimension filter: Sparse coding reconstructs each dimension equally, minimizing L2 distance between the original and the reconstructed filter. Our method reconstructs different dimensions with different weights and the weights are learned by introducing regularization features to best reconstruct filter functionality (Equation 7). That is, we use limit budget (codewords of a codebook) to reconstruct part of filter which is important for filter functionality rather than reconstruct filter appearance. Therefore, our method can find that noisy part of filter has no specific functionality and reconstruct informative part with higher weight.

Here we propose to solve this problem by introducing regularization features from the training set of a given dataset to regularize the whole process. It makes functionality receive higher weights than appearance in reconstruction. In detail, we can divide our method into two stages, in the first stage we train a general codebook for filters collected from part-based models. In the second stage we use regularization features to transform filters and the codebook into performance augmented space. Then we reconstruct the transformed filters with the transformed codebook in performance augmented space as shown in Figure 1. As mention before, the physical meaning of performance augmented space will enforce filters to reconstruct functionality in it. The whole process of Regularized Sparse Coding can be formulated as an optimization problem:

$$D = \min_{D \in C} \sum_{i=1}^{N} \min_{\alpha_i} \left(\frac{1}{2} \left\| X_i - \textstyle\sum_{j=1}^{K} \alpha_{ij} D_j \right\|_2^2 + \lambda_1 \|\alpha_i\|_0 \right) \tag{6}$$

$$\min_{\alpha_i}(\frac{1}{2}\left\|F_{reg}\cdot(X_i-\sum_{j=1}^{K}\alpha_{ij}D_j)\right\|_2^2+\lambda_2\left\|\alpha_i\right\|_0) \quad \forall i=1,...,N \tag{7}$$

Equation 6 suggests that the codebook is trained in the original sparse coding way. Based on our experiments, there exist little difference of performance by using the different optimization algorithms to learn a codebook such as [10], [4] and [11]. Equation 7 describes how we involve regularization features in filter reconstruction and generate performance augmented sparse activations. F_{reg} is a matrix of regularization features which is the key to our method. To construct this matrix, we collected lots of images from the training set of the dataset to avoid unfair situation between other methods and our method. Next, we extract feature pyramids from the collected images and randomly pick some feature patches which have the same size of filters. Finally, we use the feature patches as the rows of F_{reg} and use this matrix to regularize sparse coding. Multiply F_{reg} into clause of Equation 7, $\|\cdot\|_2^2$ term in Equation 7 becomes

$$F_{reg}\cdot X_i-F_{reg}\cdot(\sum_{j=1}^{K}\alpha_{ij}D_j)=Score_i-Score_i'. \tag{8}$$

We can consider it as norm2 distance between the original and the reconstructed score, that is, the classification error between two filters. As we proposed earlier, we should reconstruct filter functionality, i.e., the ability of filter to produce accurate score map in object detection, instead of filter appearance. In this work, we use the algorithm provided by [9] to optimize Equations 6 and 7.

4. Experiment Results

4.1. Datasets and Implementation

Pascal Visual Object Classes Challenge 2012 (VOC2012) [5]. This famous object detection benchmark consists of 20 classes. We use this dataset to show our work's capability in reducing computing time. Training set and validation set are used to train models while the test set is used to evaluate performance. Notice that this dataset only has a relative small number of classes but the experiment result proves that we can perform well on it.

ImageNet Large Scale Visual Recognition Challenge 2013 (ILSVRC2013) [16]. To show scalability of our work we choose ILSVRC2013 as our evaluation dataset. ILSVRC2013 is the largest object detection dataset in recent years, containing 200 classes. Due to the ILSVRC2014 competition, the test set evaluation server was shut down temporarily. Instead, we use the training set to train models and the validation set to evaluate performance.

Deformable Part Model release 5 (DPM) [8] [7]. In this work, we use DPM as our detection model and the default configuration as in [8]. In this configuration, each object model consists of 3 components and each component consist of 8 parts. Hence a N object classes dataset indicates that we have $24N$ filters. Although we can use root filters to learn another codebook and speedup whole object detection process, we do not consider it in our work to keep simplicity.

4.2. Performance Analysis

In Equations 6 and 7, there are three parameters, K, λ_1 and λ_2, to be tuned. However, there are so many combinations of these three parameters that it would cost too much time to try all of them to retrieve performances such as mAP. Therefore, we use a faster way to test how a parameter set performs in few seconds. As we have proposed in Section 3.2, filter functionality should be reconstructed first rather than filter appearance. In the same way as we do in Regularized Sparse Coding, we collect some validation features from the training set. Notice that these validation features are different from regularization features to prevent overfitting. Comparing convolution responses between the original and the reconstructed filter, we can average the differences of the convolution responses from several validation features to view as reconstruction error. Here we call it *score error* and it can be formulated as

$$\delta_{score}=\frac{1}{|F_{val}|\cdot N}\sum_{m=1}^{|F_{val}|}\sum_{i=1}^{N}(F_{val}^{(m)}\cdot X_i-F_{val}^{(m)}\cdot\sum_{j=1}^{K}\alpha_{ij}D_j). \tag{9}$$

Using this method as a performance estimator, we only choose few representative parameter sets to go through the standard object detection evaluation process and then produce the final results. Practically, we choose parameter sets with lowest score error for each fixed K (i.e., codebook size).

In the beginning, we examine how the size of regularization features influences Regularized Sparse Coding. In Figure 3, as expected, the more regularization features involved in Regularized Sparse Coding, the lower δ_{score} we can get. In addition, the marginal utility of regularization features diminishes at about 1000 samples. So we only use 1000 regularization features in future experiments.

Starting with the experiment on the popular VOC2012 object detection dataset, we compare our method with the sparse coding method [18] in Figure 4. Although it is a relatively small dataset, we can still get a 6 times speedup with merely 0.08 mAP drop while sparse coding has 0.15 mAP drop. We achieve this speedup using a CPU-only implementation and we can easily achieve a larger speedup by utilizing parallel computing. This experiment shows that our method can solve the mAP drop problem which happens when sparse coding shrinks the codebook size to gain

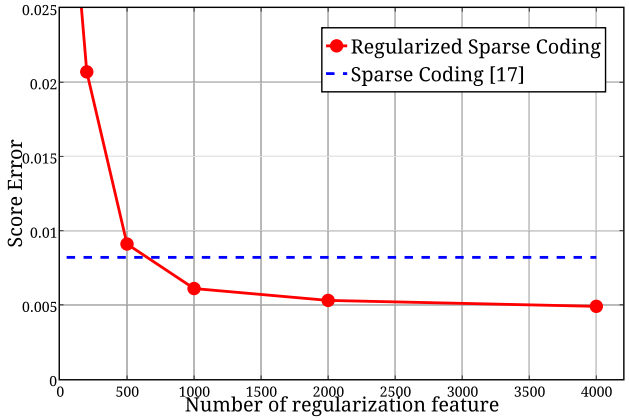

Figure 3. Effect of the number of regularization features used in Regularized Sparse Coding[3]. As the solid line illustrates, the score error drops with more features involved. The dotted line is the original sparse coding method. Although sparse coding also has low score error, small difference of score error between the two method will result in a huge performance difference (mAP) in later experiments.

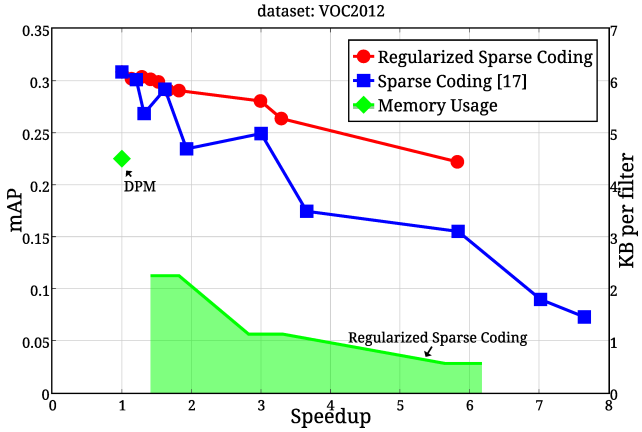

Figure 4. Comparison of sparse coding and Regularized Sparse Coding on VOC2012. The two lines illustrate that Regularized Sparse Coding outperforms sparse coding when a higher speedup is required. Although VOC2012 has only 20 object classes, our work can still achieve a 6 times speedup with merely 0.08 mAP loss while sparse coding has 0.15 mAP loss with the same speedup. Shaded area shows memory usage of our method and only 50% to 12.5% memory is needed compared to the original Deformable Part Model (diamond symbol).

a speedup. We utilize regularization process to avoid such a situation, reconstructing part of the filter which is important for its functionality with higher weights.

[3]Regularization features are also used in training stage of part-based model and involving this kind of feature in our method would not make an unfair situation.

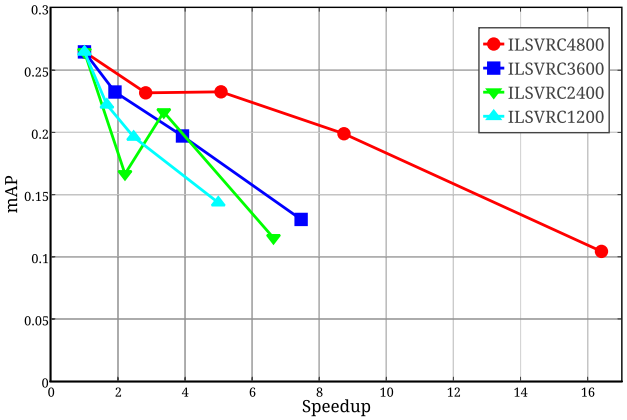

Figure 5. Effect of the filter size on speedup. This experiment shows that how our method performs in object detection systems with different filter size. Results show that our method can perform better, i.e., get higher accuracy in a fixed speedup requirement, in object detection system with larger filter size. For real world applications, object detection system will have extremely huge number of filters and our method can perform better, while other method may fail in it.

4.3. Scalability

In the scalability experiment we use ILSVRC2013 as our dataset. To understand how our method performs with different filter size, we create several subsets of the ILSVRC2013 dataset for evaluation. **ILSVRC4800**, **ILSVRC3600**, **ILSVRC2400** and **ILSVRC1200** are the new subsets of ILSVRC2013 and each contains 4800, 3600, 2400 and 1200 filters respectively, i.e., 200, 150, 100 and 50 classes. In order to compare these 4 datasets, we only evaluate the performance of their common classes, which actually are filters of **ILSVRC1200**. Performance is shown in Figure 5. This experiment shows that our method can perform better on object detection system with more filters. Explanation is that more filters in an object detection system means there are more redundant information among them. Our method works well on recognizing redundant information through regularization process and therefore performs better with larger filter size.

To compare sparse coding with our method, we conduct experiments on ILSVRC2013 to see if our method can perform well on a large dataset. In Figure 6, the result shows that we can get a huge speedup with only negligible mAP loss. In the ILSVRC2013 dataset, we only have limit budget (codewords) to reconstruct huge number of filters. However, we get different results when the two optimization methods try to use a very small number of codewords of the codebook to reach a large speedup. Sparse coding tries to reconstruct filter appearance by looking through codewords to pick best fit one. Such method cannot succeed in this strict situation, while Regularized Sparse Coding can

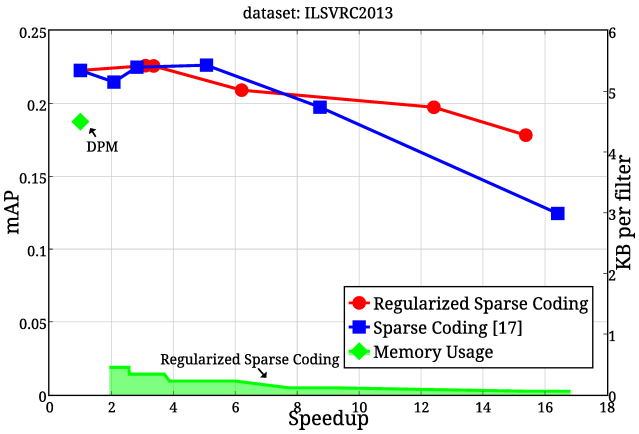

Figure 6. Comparison of sparse coding and Regularized Sparse Coding on ILSVRC2013. When a small speedup is required, size of codebook is large enough for sparse coding and Regularized Sparse Coding to reconstruct all the filters. We can observe that two methods have no difference when a small speedup is required, i.e., 1 to 10 times speedup. When a large speedup is required, Regularized Sparse Coding gradually outperforms Sparse Coding as we expected. For example, Regularized Sparse Coding can get a 16 times speedup with merely 0.04 mAP drop (Some detection results are shown in Figure 7) while sparse coding has 0.10 mAP drop. Shaded area shows memory usage of our method and only 10% to 1.25% memory is needed compared to the original Deformable Part Model (diamond symbol).

easily reconstruct filter functionality by carefully choosing codewords which help most. Results are shown in Figure 6.

5. Conclusion

In this work we proposed a method called Regularized Sparse Coding to speedup large scale object detection. Our method learns a codebook for the filters from part-based models and conduct operations only on the codebook to reduce computation. The original sparse coding method has accuracy loss problem when the number of classes grows larger, especially when a large speedup is required. Our work first introduces regularization process, which transforms filters and codebook into performance augmented space. Reconstruction in performance augmented space can minimize score map error of filters and achieve large speedup with negligible accuracy loss. To evaluate our method, we conduct experiments and compare our method with the sparse coding method [18]. On VOC2012 we prove that our method can perform better than sparse coding especially when a large speedup is required. On ILSVRC2013 our method reaches a 16 times speedup using only 1.25% memory with only 0.04 mAP loss compared to the original Deformable Part Model and outperforms the sparse coding method as well. To evaluate the effect of the number of filters on the speedup, we conduct experiments with different filter size and prove that deploying our method on larger filter size can have better accuracy-speedup trade off. Our method can also complement other methods which focus on reducing computation in the proposal extraction stage and parallel computation on GPUs to achieve more speedup. In future, we plan to apply our method on other applications, such as classification and segmentation, to provide speedup without accuracy loss and huge memory usage.

References

[1] B. Alexe, T. Deselaers, and V. Ferrari. What is an object? In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 73–80. IEEE, 2010.

[2] Q. Chen, Z. Song, R. Feris, A. Datta, L. Cao, Z. Huang, and S. Yan. Efficient maximum appearance search for large-scale object detection. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3190–3197. IEEE, 2013.

[3] T. Dean, M. A. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, and J. Yagnik. Fast, accurate detection of 100,000 object classes on a single machine. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1814–1821. IEEE, 2013.

[4] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

[5] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[6] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*, pages 2241–2248. IEEE, 2010.

[7] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[8] R. B. Girshick, P. F. Felzenszwalb, and D. McAllester. Discriminatively trained deformable part models, release 5. http://people.cs.uchicago.edu/ rbg/latent-release5/.

[9] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 689–696. ACM, 2009.

[10] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *The Journal of Machine Learning Research*, 11:19–60, 2010.

[11] S. G. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *Signal Processing, IEEE Transactions on*, 41(12):3397–3415, 1993.

[12] K. Murphy, A. Torralba, D. Eaton, and W. Freeman. Object detection and localization using local and global features. In *Toward Category-Level Object Recognition*, pages 382–400. Springer, 2006.

[13] M. Pedersoli, A. Vedaldi, and J. Gonzalez. A coarse-to-fine approach for fast deformable object detection. In *Computer*

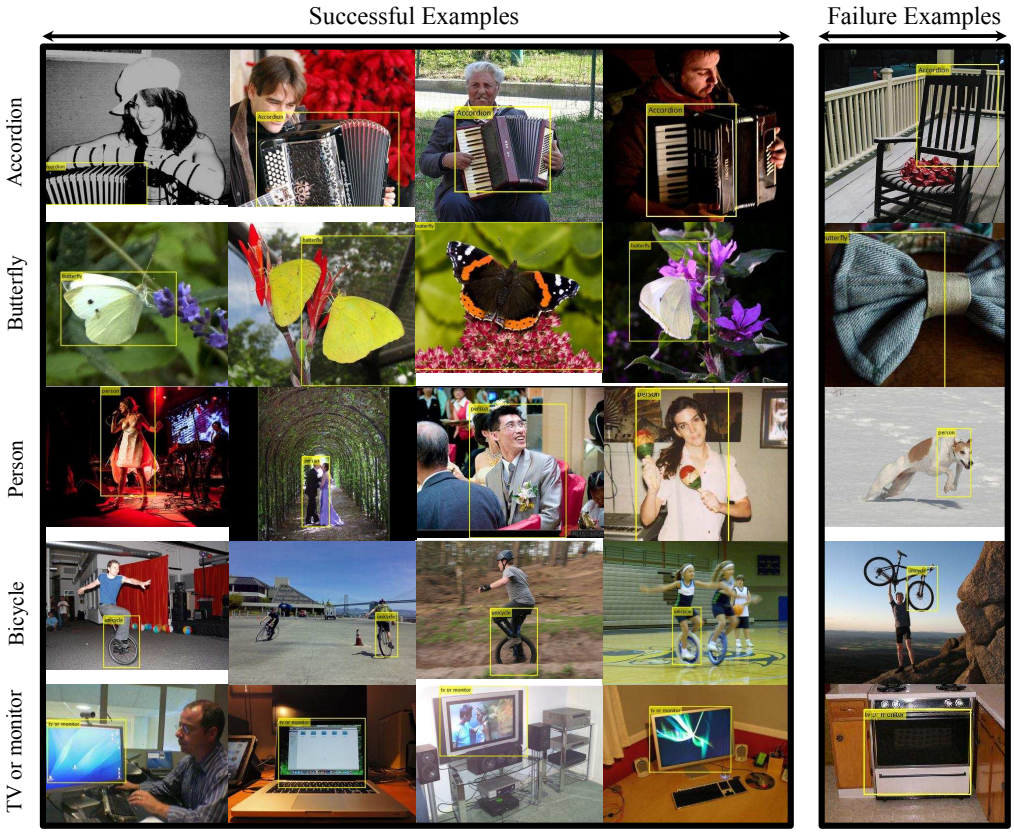

Figure 7. Detection results of our method which achieves 16 times speedup on the ILSVRC 2013 dataset. Each row shows detection results of accordion, butterfly, person, unicycle and TV or monitor from top to bottom and the rightmost column shows some failure examples. Failure examples show that many object classes share very similar parts and this situation mislead classifier to make false detection. However, this problem has not been solved by the Deformable Part Model and part of effort from our method cannot appear in mAP. Our method may be able to get better performance when applying other framework which carefully deals with this problem.

Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, pages 1353–1360. IEEE, 2011.

[14] H. Pirsiavash and D. Ramanan. Steerable part models. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3226–3233. IEEE, 2012.

[15] E. Rahtu, J. Kannala, M. Salo, and J. Heikkilä. Segmenting salient objects from images and videos. In *Computer Vision–ECCV 2010*, pages 366–379. Springer, 2010.

[16] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge, 2014.

[17] H. O. Song, T. Darrell, and R. B. Girshick. Discriminatively activated sparselets. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 196–204, 2013.

[18] H. O. Song, S. Zickler, T. Althoff, R. Girshick, M. Fritz, C. Geyer, P. Felzenszwalb, and T. Darrell. Sparselet models for efficient multiclass object detection. In *Computer Vision–ECCV 2012*, pages 802–815. Springer, 2012.

[19] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–762. IEEE, 2004.

[20] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[21] K. E. Van de Sande, J. R. Uijlings, T. Gevers, and A. W. Smeulders. Segmentation as selective search for object recognition. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1879–1886. IEEE, 2011.

[22] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 606–613. IEEE, 2009.

[23] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001.