Vector Quantization Enhancement for Computer Vision Tasks

Remi Trichet
remi.trichet@gmail.com

Noel E. O'Connor
noel.oconnor@dcu.ie

Insight Centre for Data Analytics
Dublin City University, Glasnevin, Ireland

Abstract

This paper augments the Bag-of-Word scheme in several respects: we incorporate a category label into the clustering process, build classifier-tailored codebooks, and weight codewords according to their probability to occur. A size-adaptive feature clustering algorithm is also proposed as an alternative to k-means. Experiments on the PASCAL VOC 2007 challenge validate the approach for classical hard-assignment as well as VLAD encoding.

1 Introduction

Many scientists claim that we are currently experiencing the golden age of computer vision. The introduction of machine learning techniques has played a major role in this ongoing evolution, facilitating the field to constantly break new ground.

Among all the discoveries contributing to this surge in progress, Bag-of Words (BoW) **[15] [14]** is one of the most renowned across computer vision and multimedia applications. The idea is to represent a large feature set with a much smaller *visual codebook* of vector-quantized features, called *codewords*. The analysed visual entities (i.e. image, video, object…) are then described with a distribution of their codewords. This technique has proved to increase classifier robustness due to its capacity to summarize information and has inspired many encoding techniques, either expressing feature descriptors as combinations of visual codewords **[8] [28]** or recording the difference between the features and the visual codeword **[11] [6] [13]**.

However, little attention has been directed to the proper adjustment of this technique to computer vision tasks. Indeed, codebooks are built independently of classifier needs. Thus the resulting codewords do not necessarily discriminate the semantic classes that the user ultimately wants to distinguish. This should not be surprising given that the approach only focuses on feature value information.

We have identified 3 distinct sub-problems to this issue:

1- The construction of a visual codebook is unable to take categories into account. Indeed, the information provided by the codewords might overlap or describe the categories in an imbalanced way. As a result, there is no guarantee that the provided codebook will best suit the training needs of the corresponding classifier.

2- Codewords are assumed to be equiprobable. An erroneous assumption that leads to biased histogram representation.
3- Codebooks are not tailored to classifier purposes. Typically, when faced with an *n*-category classification task, *n* one-against-all classifiers are modelled to address the problem. Quantized histograms fed to these classifiers stem from a unique codebook with equal coverage of these *n* categories (and, eventually, negative examples). For best performance, half of the data fed to the one-against-all classifier should refer to that particular category of interest.

In this work, we intend to overcome the first drawback directly during the clustering process via the integration of prior knowledge. More specifically, during the training phase, we label features with their corresponding visual entity category, and utilize constrained clustering **[3]** to produce clusters including more features belonging to the same category. A modified version of the *k*-means algorithm **[4]**, and a size-adaptive agglomerative clustering **[2]**, harnessing the purity metric to assess a cluster's discriminative power, are introduced. We include the probability of a codeword to occur for an enhanced histogram representation as an answer to the second problem. Finally, we tackle the third problem with category-tailored codebooks that further enhance classifiers' performance.

Our method has the following advantages:

1. It provides, for each category, a specific codebook that enhances classifier performance.
2. It provides, for each codebook, a set of codewords that enhances classifier performance.
3. The only annotations needed are the entity category label.
4. We would like to emphasize the genericity of this work: all these modifications can be applied to any task that uses BoW and learns classifiers.

The remainder of this paper is organized as follows. After reviewing the related literature, section 2 details the aforementioned BoW improvements, namely maximizing intra-cluster category similarity, size-adaptive clustering, and accounting for the probability of a codeword to occur during histogram generation. Section 3 covers experiments; section 4 concludes this paper.

1.1 Related work

Two types of approaches can be associated with codebook generation: encoding techniques and semantically-oriented codebook construction.

Encoding methods precursors **[9] [10]** typically perform local descriptor hard quantization and describe the media of interest with a distribution of codewords. The soft assignment variant **[8]** represents each feature descriptor as a weighted combination of its *n* nearest codewords. Locally-constrained linear coding **[27]** (LLC) performs sparse coding **[28]** to code each point according to the *k* nearest neighbours. A recently breakthrough records the difference between features and codewords. Given a predefined vocabulary of size *K* in a feature space of *D* dimensions, Vector of Linearly Aggregated Descriptors (VLAD) **[11]** encodes features first order differences to each visual word, called residuals. These residuals are stacked together to form a *KD*-value vector for each descriptor. Many improvements exist such as the use of tensors

(VLAT) **[7]**, intra-normalization **[30]**, or second order statistics and supervised labelling **[34]**. Similarly, Fisher vector (FV) **[5]** captures the average first and second order differences between the descriptors and GMM centres. A combination of VLAD and FV encodings **[12]**, as well as the introduction of non-linear additive kernels and PCA normalization **[6]**, further improve results. Super vector coding (SV) **[13]** represents descriptors with first order differences, and adds a cluster mass parameter. See **[1]** for a review on feature encoding methods.

Approaches building a semantically meaningful codebook often filter unreliable codewords by looking for co-occurring patterns. More specifically, **[25]** learn what they call co-location and co-activation. Context-constrained linear coding **[26]** (CLC) is a variation of LLC **[27]** that incorporates spatio-temporal context within the distance utilized to compare feature points. **[24]** build up the co-occurrence codebook on a video segmentation layout. Finally, **[29]** proposed contextual clustering, using feature co-occurrences to disambiguate the output of multi-view clustering. Despite the success of these approaches, the necessary pairing of codewords reduces their discriminative power and neighbourhood definitions are often arbitrary.

Alternate ways to extract a codebook in a semantic manner encompass diffusion maps **[21]**, semantic aware distance metric **[22]**, meaningful word pair counts **[23]**, spatial pyramid matching kernel **[10]**, or consensus clustering to minimize the disagreement between several clustering outputs **[20]**. These methods typically rely on cluster shape, density, or even repartition and thus are not tailored for category discrimination.

To address this, several semi-supervised approaches make use of the category labels prior knowledge to refine the codebook construction. **[16]** utilize extremely randomized clustering forest, **[17]** mutual information, and **[19] [18]** a unified vocabulary. But all these methods consider weighting or selecting codewords **after** they have been generated, therefore down-bounding the technique efficiency with the codebook discriminative power.

2 Encoding enhancement.

This section covers our various encoding enhancements. After justifying the need for similar clusters, we present two clustering algorithms using feature point purity. Then, we introduce category-tailored codebooks and incorporate the probability of a codeword to occur.

2.1 Maximizing intra-cluster category similarity

A key requirement for any classification task to perform well is that histogram distributions are expected to be significantly different from one category to another. Hence, the idea is to have each bin characterizing as few categories as possible. At the clustering level, this idea translates to maximising the intra-cluster category similarity. In other words, having clusters that encompass feature points extracted from instances of the same category, to the extent possible. Figure 1 illustrates the principle.

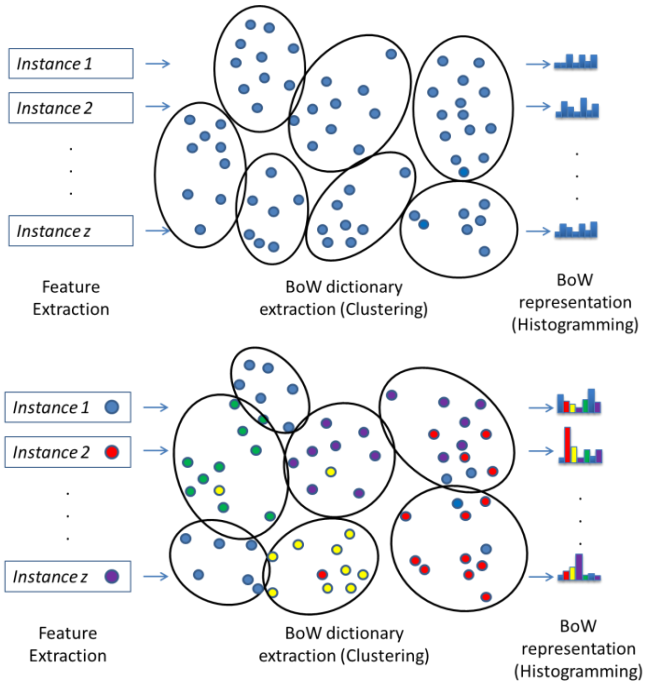

Figure 1: Toy example demonstrating the impact of Maximizing intra-cluster category similarity. Each ellipse depicts one of the 7 clusters. Each colour represents one of the 5 categories. Top: Traditional codebook generation. Bottom: Codebook generation maximizing intra-cluster category similarity. Clusters better cover the categories, therefore leading to increased codeword discriminative power. Best viewed in colour.

Typically, a category label is associated to each training instance. We further extend this labelling process to the features extracted from a particular instance.

Consequently, each feature bears the category label of the instance to which it belongs. Let $C = \{C_0 \ldots C_k\}$ the set of k clusters, and $L = \{l_0, \ldots, l_n\}$ the set of n categories. Intra-cluster category similarity of a cluster p_i is assessed thanks to the purity evaluation measure:

$$purity(C_i, L) = max_j(|C_i \cap l_j|)/|C_i| \quad (1)$$

, with |.| depicting the cardinality. Due to noise, extreme data variability, and outliers, optimal purity is rarely achievable. However, we will show through experiments that a codebook discriminative power is correlated to its global purity.

2.2 Purity *k*-means.

This subsection extends the *k*-means algorithm **[4]** to incorporate feature point purity. The method is inspired from constrained clustering **[3]**. Constrained clustering is a type of semi-supervised clustering incorporating prior knowledge into algorithm by imposing grouping constraints.

Our algorithm is dubbed purity-*k*-means. It takes as input a set of feature points $X = \{x_j \mid j=1,\ldots,N\}$ and a desired numbered of clusters *k*. *X* should equally represent the *n*

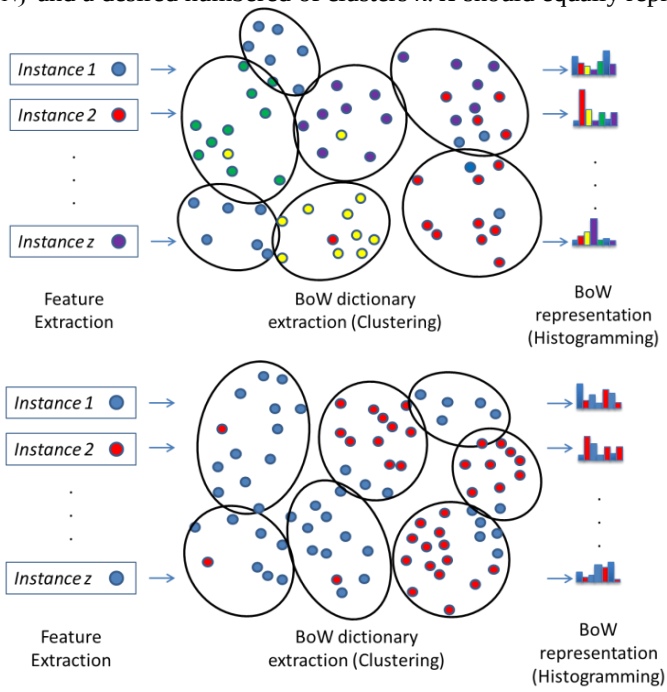

Figure 2: Toy example demonstrating the impact of category-tailored codebooks. Each ellipse depicts one the 7 clusters. Each colour represents one of the 5 categories. Top: single codebook generation. Only one codeword/cluster represents the red category, leading to a discrepancy in the case of a one-versus-all classifier. Bottom: Codebook tailored for the red category (feature points are different). More codewords/clusters are dedicated to the red category, leading to increased discriminative power. Best viewed in colour.

categories for best results. The only difference with k-means concerns the way cluster centres are updated. In order to maximize cluster purities, only features from the majority category are considered during this step. The algorithm is the following:

1. Randomly initialize the cluster centres C_i, $i = 1,\ldots,k$
2. Assign feature vectors the same way as for typical *k*-means:

$$C_i = \{x_j \mid d(x_j, \mu(C_i)) \leq d(x_j, \mu(C_u)) \; \forall \, u \neq i, j = 1, \ldots, N\} \quad (2)$$

3. Update cluster centres.

$$A_i = C_i \cap l_{\,argmax_j(|C_i \cap l_j|)} \qquad \mu(C_i) = \frac{1}{|C_i|} \sum_{x_j \in A_i} x_j \quad (3)$$

4. Repeat step 2 and 3 until convergence.

2.3 Category-tailored codebooks generation.

Most applications that aim to discriminate *n* categories actually build *n* one-versus-all classifiers and eventually combine the classifier confidence values. For improved accuracy, it is then possible to further extend purity-based clustering by creating one codebook per category. Indeed, while constructing a one-versus-all classifier, feature points can be labelled as previously. In this case however, possible labels are "the category of interest" or "all other categories". Similarly, the set of feature points should equally represent the two categories. Therefore, the method builds up a range of *n* codebooks, each one of them specifically tailored for the classification of one particular type of instance. As a consequence, approximately the same number of codewords characterizes the category of interest and the remaining categories, leading to increased discriminative power. Figure 2 illustrates the principle.

2.4 Size-adaptive clustering

However, even with these extensions, *k*-means clustering **[4]**, despite its advantages, still has one major drawback: Each cluster approximately covers the same portion of the feature space. Consequently, too many clusters are associated to regions with homogeneous feature labels and these arbitrary shaped clusters do not necessarily fit the configuration of feature groups in mixed label areas.

For better fit to the data, the cluster size should be driven by the data. More clusters would be dedicated to conflicting areas of the feature space, while fewer and bigger clusters would be associated to areas of high purity.

Hence, we present a clustering algorithm that frees clustering from the arbitrary cluster size, and better fits the data. This algorithm was designed bearing in mind 3 criteria:

1-The algorithm should maximize cluster purity.

2-Cluster size should adjust to the data.

3-New features falling into the cluster area should be easily associated to it.

This last criterion is paramount to keep further codeword encoding tractable. Either clusters should be compact enough to be approximated to *k*-means clusters, either an existing distance metric should be able to test the feature point association to arbitrarily shaped clusters. Figure 3 depicts the idea.

We employed agglomerative clustering **[2]** for this purpose, as it naturally fits the first two criteria. Agglomerative clustering is a hierarchical clustering algorithm that first initializes every data point as a cluster. The two closest clusters are then iteratively fused until a stopping criterion is met. The fusion distance typically considers the two closest points of each cluster for better fit to the data configuration. As this algorithm is renowned for creating clusters of complex shape, we modified the cluster distance assessment to enforce cluster compactness. More specifically, we use centroid-linkage, the centroid being defined by:

$$\mu(C_i) = \frac{1}{|C_i|} \sum_{x_j \in C_i} x_j \tag{4}$$

Also, to discourage extreme discrepancy between cluster sizes, we weight the distance between clusters according to the size of the smallest one:

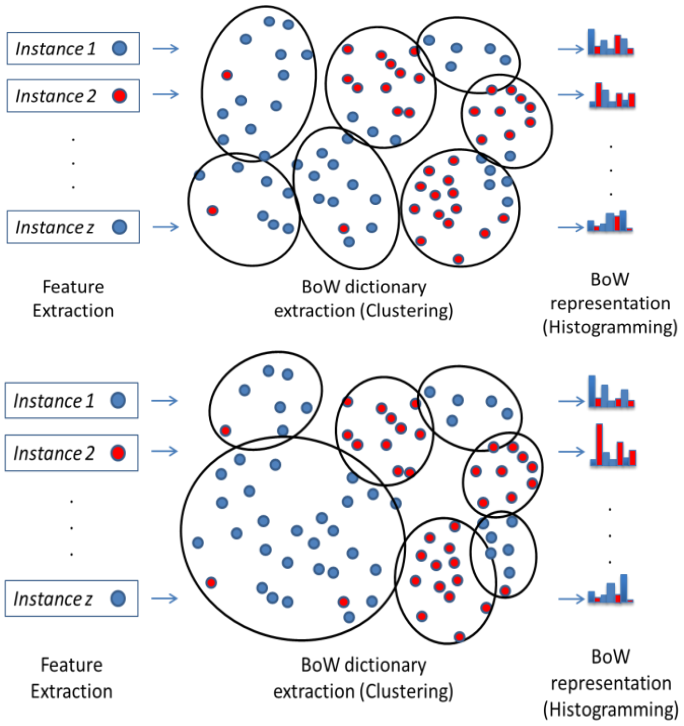

Figure 3: Toy example showing the need of size-adaptive clusters. Each ellipse depicts one the 7 clusters. Each colour represents one of the 2 categories. Top: *k*-means clustering creating clusters of similar radius. Bottom: Size-adaptive clustering. More clusters are dedicated to conflicting areas of the feature space, while fewer and bigger clusters are associated to high purity areas. Best viewed in colour.

$$W_{size} = \sqrt{\frac{k}{N} min\left(|C_i|, |C_j|\right)} \tag{5}$$

, where k is the desired number of clusters and N the feature points crdinality. Finally, we approximate the future purity of clusters to be fused C_i and C_j, named *purity*(C_i, C_j, L), as the respective proportion of their majority label within the other cluster. More formally:

$$purity\left(C_i, C_j, L\right) = \frac{\left|C_i \cap l_{\,argmax_u(|C_j \cap l_u|)}\right|}{2|C_i|} + \frac{\left|C_j \cap l_{\,argmax_v(|C_i \cap l_v|)}\right|}{2|C_j|} \tag{6}$$

Therefore, the similarity between 2 clusters C_i and C_j is a balance between their expected purity and their size-weighted distance:

$$S(C_i, C_j) = \left(1 + W_p\left(1 - purity(C_i, C_j, L)\right)\right) \times d\left(\mu(C_i), \mu(C_j)\right) \times W_{size} \quad (7)$$

, with *d(.)* a metric and W_p the weight assigned to purity. This parameter controls the purity of the clusters over their compactness. A high value will favour purity at the cost of eccentric cluster forms, and vice versa. We empirically set W_p to 0.5 for all our experiments, leading to a trade-off between these two factors.

As agglomerative clustering can be computationally expensive, we used dynamic programing to speed up its execution. More specifically, candidates for fusion are based on the (pre-computed) *nn* nearest neighbours of each feature point, the fusion list is only re-sorted if one of the updated similarities is lower than the current lowest one, and pruned from identical consecutive instances.

We deal with the variance in cluster sizes by computing the space coverage $\sigma(C_i)$ of each cluster C_i with a centroid $\mu(C_i)$ as follows:

$$\sigma^2(C_i) = max_{x_j \in C_i}\left(d\left(x_j, \mu(C_i)\right)\right) \quad (8)$$

This differs from FV **[5]** priors or SV **[13]** posteriors as we aim to emphasise the coverage of the cluster. It is further utilized during feature encoding to weight the distance from a point to a cluster centroid:

$$\mathrm{C}_i = \left\{ \mathrm{x}_j \;\middle|\; \frac{\mathrm{d}\left(\mathrm{x}_j, \mu(\mathrm{C}_i)\right)}{\sigma(\mathrm{C}_i)} \leq \frac{\mathrm{d}\left(\mathrm{x}_j, \mu(\mathrm{C}_u)\right)}{\sigma(\mathrm{C}_u)} \quad \forall\, \mathrm{u} \neq \mathrm{i}; \mathrm{j} = 1, \ldots, \mathrm{N} \right\} \quad (9)$$

The average purity loss caused by this approximation, with the aforementioned parameter setting, is 11% for a single codebook, 6% for category tailored ones.

2.5 Probability of a codeword to occur

Histogram construction is based on the underlying assumption that every single codeword is equiprobable. Consequently, each codeword occurrence increments the corresponding histogram bin by the same value. However, a simple experiment over 4000 *k*-means clusters determined from 1 million dense SIFT features on the PASCAL VOC 2007 dataset **[31]** invalidates this assumption: feature cardinality can vary by 113% from cluster to cluster. This factor increases to 800% when the same experiment is performed with the size-adaptive clustering described in section 2.4. This biased cluster cardinality skews codeword occurrences, and therefore the entire histogram. For a fair distribution, the probability $p(C_i)$ of a codeword to occur within the database should be accounted while building up the histogram. We denote $p(C_i)$ of a codeword C_i, $i = 1,\ldots,n$, as:

$$p(C_i) = |C_i|/N \quad (10)$$

, with |.| the cardinality of a cluster, *k* the codebook size, and *N* the total number of feature points used for clustering. Assuming all clusters to be equiprobable, $p(C_i) = 1/\mathrm{k}\ \forall \mathrm{i}$. Thus we simply increment a histogram codeword bin as follows:

$$h_i = h_i + \sqrt{k/p(C_i)} \quad (11)$$

Note that in the case of equiprobable clusters, this comes down to the traditional increment of h_i by 1.

Density correction	NO	NO	YES	YES	YES	NO	YES	NO	NO	NO	NO
Size (numbers) of codebooks	*4000 (×1)*	*4000 (×1)*	*4000 (×1)*	*4000 (×20)*	*4000 (×20)*	*4000 (×1)*	*4000 (×20)*	*1024 (×1)*	*256 (×1)*	*256 (×1)*	*256 (×20)*
Clustering	KM	PKM	PKM	PKM	PAC	KM	PAC	KM	GMM	KM	PAC
Encoding	BoW-hard **[1]**	BoW-hard	BoW-hard	BoW-hard	BoW-hard	BoW-soft **[1]**	BoW-soft	SV **[1]**	FV **[1]**	VLAD **[34]**	VLAD
Aeroplane	68.65%	74.59%	75.78%	76.96%	76.13%	69.82%	75.96%	74.32%	78.97%	75.90%	79.97%
Bicycle	57.04%	61.51%	60.08%	61.45%	63.32%	59.20%	63.24%	63.79%	67.43%	65.80%	67.53%
Bird	39.86%	46.04%	44.82%	46.41%	47.61%	41.97%	47.94%	47.02%	51.94%	51.10%	50.01%
Boat	64.59%	66.50%	68.08%	67.45%	67.21%	64.85%	67.31%	69.44%	70.92%	73.60%	74.95%
Bottle	21.96%	22.61%	25.00%	24.23%	24.40%	23.90%	25.48%	29.06%	30.79%	28.90%	27.64%
Bus	58.79%	54.46%	53.66%	57.74%	55.97%	59.02%	55.92%	66.46%	72.18%	63.60%	65.74%
Car	73.89%	76.61%	77.66%	77.81%	79.69%	74.98%	80.12%	77.31%	79.94%	80.30%	81.27%
Cat	53.77%	52.18%	53.24%	55.19%	53.02%	54.63%	53.05%	60.18%	61.35%	59.30%	60.57%
Chair	52.40%	45.61%	44.97%	46.57%	47.54%	52.49%	47.63%	50.19%	55.98%	52.10%	52.41%
Cow	38.57%	35.31%	36.45%	36.79%	37.30%	40.48%	36.74%	46.46%	49.61%	44.00%	45.24%
DiningTable	49.20%	51.31%	52.45%	54.06%	51.04%	50.53%	51.81%	51.86%	58.40%	50.50%	54.43%
Dog	36.85%	41.13%	42.72%	42.52%	43.92%	37.98%	43.68%	44.07%	44.77%	42.90%	42.33%
Horse	75.59%	77.23%	77.77%	76.65%	78.16%	76.03%	78.44%	77.85%	78.84%	78.80%	77.31%
Motorbike	61.59%	60.41%	61.65%	61.79%	64.50%	63.73%	64.46%	67.12%	70.81%	62.40%	65.29%
Person	81.63%	83.62%	83.91%	83.79%	84.31%	82.47%	84.60%	83.07%	84.96%	85.00%	85.94%
Potted Plant	20.47%	21.16%	21.34%	24.66%	24.05%	22.31%	24.52%	27.56%	31.72%	26.10%	28.35%
Sheep	40.05%	41.95%	40.09%	40.61%	42.93%	43.08%	42.61%	48.50%	51.00%	46.10%	47.56%
Sofa	50.92%	42.17%	46.03%	45.43%	49.32%	50.96%	50.58%	51.10%	56.41%	49.60%	49.17%
Train	73.39%	75.75%	76.03%	76.26%	76.66%	73.97%	76.36%	75.50%	80.24%	76.30%	79.02%
TV Monitor	49.21%	49.04%	49.31%	48.38%	52.26%	49.68%	51.22%	52.26%	57.46%	52.80%	55.02%
mAP	**53.42%**	**53.96%**	**54.55%**	**55.24%**	**55.97%**	**54.60%**	**56.08%**	**58.16%**	**61.69%**	**58.26%**	**59.49%**
Avg Purity	*30.90%*	*35.20%*	*35.20%*	*65.20%*	*71.90%*	*30.90%*	*71.90%*	*N. A.*	*N. A.*	*N. A.*	*64.30%*

Table 1: Results on the PASCAL VOC 2007 challenge. Baselines are in red, our runs in black. **KM**: k-means; **PKM**: Purity-kmeans; **PAC**: purity agglomerative clustering **GMM**: Gaussian mixture model.

3 Experiments

Our experiments are designed to compare results with the reference survey on feature encoding **[1]**. This paper comprehensively reviews existing encoding techniques on the PASCAL VOC 2007 object recognition dataset **[31].** This dataset contains about 10000 images split into train, validation, and test sets, and labelled with 20 object classes. A one-vs-all SVM classifier is learned and evaluated independently for each category. The performance is measured as mean Average Precision (mAP) across all classes.

We compared our encoding methods with typical hard-**[10]**, and soft-coding **[8]**, as well as Fisher coding **[5]**. Each contribution is validated independently on the former; only the best run is provided for the latter.

We reproduced the same experimental conditions as in **[1]**. Dense SIFT (PHOW) features **[33]** are extracted with the publicly available VLFeat toolbox **[32]** for all runs. A pre-processing step reduces the feature dimensionality to 80 for all residual based methods. Moreover, VLAD encoding is also whitened to comply with the setup in **[34]**. The codebook size is 4000 for hard- and soft-coding, 1024 for SV, and 256 for Fisher and VLAD vectors. Hellinger's kernel is employed with Fisher encoding, a linear kernel for VLAD and SV, X^2 distance with all other encoding methods. VLAD histograms undergo power and intra-normalization, all others are L^2-normalized. See **[1]** and **[34]** for details. Results are presented in table 1.

Our first 4 runs (in black font) independently test the various improvements presented throughout this paper, demonstrating their gradual improvement over the traditional BoW scheme. Clearly, cluster purity is also correlated to discriminative power.

Our last 2 runs compare the full system with state-of the art encoding methods. The method boosts hard-coded BoW encoding by 2.55%, 1.48% for its soft-coding version and 1.26% for the VLAD encoding. The soft coding experiment does not provide as much performance gain as its hard coding counterpart. This is sensible as soft-coding can be harmful in the case of neighbouring purity clusters with different dominant labels. As for the VLAD encoding, the reduced codebook cardinality explains its small improvement.

4 Conclusion

In this paper, we have enhanced the BoW scheme in several ways: maximizing intra-cluster category similarity, accounting for the probability of a codeword to occur and proposing a size-adaptive clustering algorithm.

As the cluster purity associated to each codeword offers a natural way to perform feature selection, we envision investigating this direction.

5 Acknowledgement

This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under grant number SFI/12/RC/2289.

6 References

[1] K. Chatfield, V. Lempitsky, A. Vedaldi, A. Zisserman, The devil is in the details: an evaluation of recent feature encoding methods, BMVC, 2011.

[2] L. Kaufman, P.-J. Rousseeuw, Finding Groups in Data: an Introduction to Cluster Analysis, John Wiley & Sons, 1990

[3] K. Wagstaff, C. Cardie, S. Rogers, S. Schrödl, Constrained K-means Clustering with Background Knowledge, ICML, 2001.

[4] J. Hartigan and M. Wang, A K-means clustering algorithm, Applied Statistics, 28:100–108, 1979.

[5] F. Perronnin, C. Dance, Fisher kenrels on visual vocabularies for image categorizaton. CVPR, 2006.
[6] F. Perronnin, J. Sánchez, T. Mensink, Improving the fisher kernel for large-scale, image classification, ECCV, 2010.
[7] R. Negrel, D. Picard, P.H. Gosselin, Compact tensor based image representation for similarity search, ICIP, 2012.
[8] J. Philbin, O. Chum, M. Isard, J. Sivic, A. Zisserman, Lost in quantization: Improving particular object retrieval in large scale image databases, CVPR, 2008.
[9] G. Csurka, C. Bray, C. Dance, L. Fan, Visual categorization with bags of keypoints, ECCV, 2004.
[10] S. Lazebnik, C. Schmid, J. Ponce, Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories, CVPR, 2006.
[11] H. Jegou, M. Douze, C. Schmid, P. Perez, Aggregating local descriptors into a compact image representation, CVPR, 2010.
[12] J. Delhumeau, P.-H. Gosselin, H. Jégou, P. Pérez, Revisiting the VLAD image representation, ACM Multimedia, 2013.
[13] X. Zhou, K. Yu, T. Zhang, T. S. Huang, Image classification using super-vector coding of local image descriptors. ECCV, 2010.
[14] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, C. Bray, Visual categorization with bags of keypoints, ECCV, 2004.
[15] J. Sivic, A. Zisserman, Video google: A text retrieval approach to object matching in videos, ICCV, 2003.
[16] F. Moosmann, E. Nowak, and F. Jurie, Randomized clustering forests for image classification, PAMI, vol. 30, no. 9, pp. 1632–1646, 2008.
[17] J. Winn, A. Criminisi, and A. Minka, Object categorization by learned universal visual dictionary, ICCV, 2005.
[18] L. Yang, R. Jin, R. Sukthankar, and F. Jurie, Unifying discriminative visual codebook generation with classifier training for object category recognition, CVPR, 2008.
[19] D. Larlus and F. Jurie, Latent mixture vocabularies for object categorization, BMVC, 2006.
[20] R. J. López-Sastre, J. Renes-Olalla, P. Gil-Jiménez, S. Maldonado-Bascón, S. Lafuente-Arroyo, Heterogeneous Visual Codebook Integration via Consensus Clustering for Visual Categorization. TCSVT, 2013.
[21] J. Liu, Y. Yang, and M. Shah, Learning semantic visual vocabularies using diffusion distance, CVPR, 2009.
[22] S. Zhang, Q. Tian, G. Hua, W. Zhou, Q. Huang, H. Li, and W. Gao, Modeling spatial and semantic cues for large-scale near-duplicated image retrieval, CVIU, vol. 115, no. 3, pp. 403–414, 2011.
[23] T. Li, T. Mei, I.-S. Kweon, and X.-S. Hua, Contextual bag-of-words for visual categorization, TCSVT, vol. 21, no. 4, pp. 381–392, 2011.
[24] R. Trichet, R. Nevatia, Video Segmentation and Feature Co-occurrences for Activity Classification, WACV, 2014.
[25] B. Leibe, A. Ettlin, and B. Schiele, Learning semantic object parts for object categorization, Image Vision Computing, vol. 26, no. 1, pp. 15–26, 2008.

[26] Z. Zhang, C. Wang, B. Xiao, W. Zhou, and S. Liu, Action recognition using context-constrained linear coding, IEEE Signal Process. Letter., 19(7), 2012.
[27] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, Locality-constrained linear coding for image classification, CVPR, 2010.
[28] A. Kovashka and K. Grauman, Learning a hierarchy of discriminative space-time neighborhood features for human action recognition, CVPR, 2010.
[29] H. Wang, J. Yuan, and Y.-P. Tan, Combining feature context and spatial context for image pattern discovery, ICDM, 2011.
[30] R. Arandjelovic and A. Zisserman, All about vlad, CVPR, 2013.
[31] M. Everingham, A. Zisserman, C. Williams, and L. Van Gool, The PASCAL visual Obiect classes challenge 2007 (VOC2007) results. Technical report, Pascal Challenge, 2007.
[32] A. Vedaldi and B. Fulkerson, VLFeat - An open and portable library of computer vision algorithms, ACM Multimedia, 2010.
[33] M. Krystian, and C. Schmid, A performance evaluation of local descriptors, PAMI, vol27, no10, p1615-1630, 2005.
[34] X. Peng, L. Wang, Y. Qiao, and Q. Peng, Boosting VLAD with Supervised Dictionary Learning and High-Order Statistics, ECCV, 2014.