Supervised learning and codebook optimization with neural network

Mingyuan Jiu Christian Wolf Christophe Garcia Atilla Baskurt

Université de Lyon, CNRS

INSA-Lyon, LIRIS, UMR CNRS 5205, F-69621, France

{firstname.lastname}@liris.cnrs.fr

Abstract

In this paper, we present a novel approach for supervised codebook learning and optimization with neural networks for bag of words models in visual recognition tasks. We propose a new supervised framework for joint codebook creation and class learning, which learns the codewords in a goal-directed way using the class labels of the training set. As a result, the codebook becomes more discriminative. Two different learning algorithms, one based on error backpropagation and one based on cluster label reassignment, are presented. We evaluate them on the KTH dataset for human action recognition, reporting very promising results. The proposed technique allows to improve the discriminative power of an unsupervised learned codebook, or to keep the discriminative power while decreasing the size of the learned codebook.

Keywords

Bag of words models, supervised learning, neural networks, action recognition.

1 Introduction

Human action recognition from videos has been studied deeply during the past few years in part due to very promising and increasing applications in real world such as video surveillance, human-computer interaction, etc. Many approaches were proposed, among which the *bag-of-words* (BoW) model has been frequently employed in computer vision applications[1, 2]. BoW has proved to be an efficient representation in this context. This popular model was first introduced in natural language processing, in which each document is expressed as a histogram of frequencies of orderless words.

In order to employ the BoW model in computer vision applications, a visual codebook is learned by unsupervised clustering or vector quantization of visual concepts of local primitives in the images or videos. K-means [1] or random forests [3] are usually applied. Motived by scale invariance, and similar to spatial features from local patches in object recognition [4], visual concepts are described as spatial-temporal features extracted from 3 dimensional local patches in videos to capture the characteristics of spatial-temporal evolutions of actions. The BoW for a new video is calculated in a similar way : extraction of descriptors on local primitives, projection of the descriptors on the codebook, and calculation of a histogram of the occurrences of each codeword of the codebook. A video is classified passing the BoW model to any learning machine, for instance a support vector machine (SVM) or a neural network (NN).

In the literature, there are many extensions, for instance, correlograms [5], topic models [6], local grouping and compound features [7], spatial co-occurrences of pairs of features [8, 9] and parts based models [10]. In this paper, we seek to improve the performance of basic BoW models by learning a more discriminative codebook. The traditional codebook creation through unsupervised clustering ignores the class labels of the feature vectors in the training set, which are only used in the phase of classification. Intuitively, the discriminative power of the codewords could be increased by learning using the label information, as we propose in this paper.

The performance of BoW models on different applications largely depends on the discriminative quality of the codebook. Some research has already focused on learning a semantic codebook to improve its quality. In the work of Liu [5], they iteratively obtain an optimal and compact codebook via maximal mutual information. At each iteration they merge two clusters which have minimum loss in mutual information. The iterative procedure continues until a threshold of maximal mutual information or minimum cluster number is achieved. The optimal codebook size is found by unsupervised learning. In [11], Liu uses a diffusion map to embed a mid-level codebook into a semantic codebook. However, it is not appropriate to measure semantic distances using diffusion distances. In recent work, Saghafi [12] proposes a concept space to illustrate the semantic relations between the visual codewords. They apply generative models such as latent semantic analysis (LSA) and probabilistic latent semantic analysis (pLSA) to discover the latent semantic relations between the initial codewords. In contrast to the unsupervised pLSA learning framework in which the number of latent topics is equal to the number of classes [6], the number of topics is variable in this method. L.Ballan et al. build an effective codebook through radius-based clustering and apply soft assignment

to obtain a BoW model for human action [13]. In these methods mentioned above, the codebooks are created by unsupervised methods, and the label information of feature vectors is ignored in the codebook creation. As a consequence, the visual codebook is less discriminative.

In contrast to unsupervised codebook learning, we propose a supervised learning and codebook optimization framework. The whole sequence, codebook creation and class learning, is formulated as an artificial neural network, and the error gradient information is used to update the codebook cluster centers as well as the classical multilayer perceptron (MLP) weights for class recognition. The error gradient is interpreted as two different meanings, pure numerical vector and histogram bin value, leading to two different learning methods.

The learning and optimization framework we present in this work is well suited for any application for which bag of words models can be successfully used. The improvements we propose make the codebook more discriminative, and therefore are likely to improve many of the existing extensions of the basic BoW model. To the best of our knowledge, this is the first attempt to combine codebook learning with action classification in a unified framework.

The paper is organized as follows. Section 2 gives an overview of our framework. Section 3 formulates the integrated and joint codebook learning algorithm. Two different learning algorithms, a classical backpropagation algorithm as well as cluster reassignment algorithm specific to the BoW model, are specified. Section 4 presents experimental results on the public KTH human action dataset [15]. Section 5 gives a conclusion.

2 The neural model

Figure 1 illustrates our framework. As can be seen, it consists of two parts : an initial part at the left and a classical MLP part at the right. The left part processes feature vectors and projects them to a codebook, which is stored in the "weights" of this part. While passing through the left part of the network, the feature vector is translated into a binary vector indicating which cluster center it activates. When several feature vectors are presented, this information is integrated into a BoW model internally, and then is transferred into the right part, which takes a decision on the action class. Each layer is explained in detail as follows :

The input layer consists of a set $\boldsymbol{a}$ of M input nodes $\boldsymbol{a} = [a_1, \dots a_M]^T$ corresponding to the feature values assigned to a single local primitive, i.e. an interest point.

The N nodes of the second layer $\boldsymbol{b} = [b_1, \dots b_N]^T$ correspond to the distances of the input feature vectors to each of N cluster centers. To each node i is thus assigned a cluster center $\boldsymbol{w}_i^{cc}$, i.e. a vector of dimension M, which is involved in the distance computation :

$$b_i = ||\boldsymbol{a} - \boldsymbol{w}_i^{cc}|| \tag{1}$$

The N nodes of the third layer compute an indicator of the nearest cluster center. The nearest corresponding node

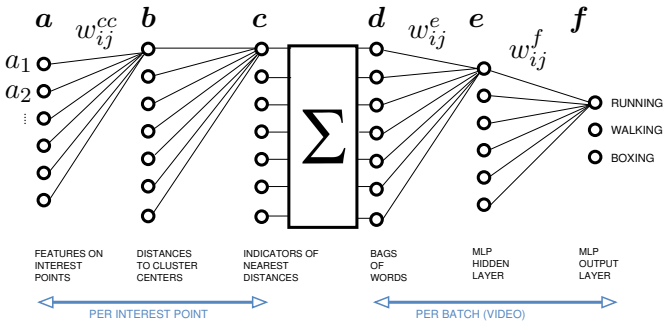

Figure 1 – *A scheme of the different layers in our framework.*

will be assigned 1, the other nodes 0. This is approximated through a softmin function by choosing T of enough value :

$$c_i = g^b(b_i) = \frac{\exp(-b_i/T)}{\sum_j \exp(-b_j/T)} \tag{2}$$

The layers described above stimulate a single feature vector corresponding to a single local primitive. When multiple feature vectors of the same entity are presented iteratively, different values c_i^p are obtained for different feature vector p. The nodes of the next layer integrate the responses for a single video over all P points :

$$d_i = \sum_{p=1}^{P} c_i^p \tag{3}$$

The next two layers, $\boldsymbol{e}$ and $\boldsymbol{f}$, are a classical MLP with weights $\boldsymbol{w}^e$ and $\boldsymbol{w}^f$ and activation function $g(x)$:

$$\bar{e_i} = \sum_j w_{ij}^e d_j, \quad e_i = g(\bar{e_i}) \tag{4}$$

$$\bar{f_i} = \sum_j w_{ij}^f e_j, \quad f_i = g(\bar{f_i}) \tag{5}$$

The last layer is thus the output layer with the set of nodes $\boldsymbol{f} = [f_1, \dots f_C]^T$, one for each of the C actions.

3 Joint codebook and class label learning

Our framework benefits from the error backpropagation strategy of neural network to sequentially and iteratively learn two different types of weights (i.e. the weights $\boldsymbol{w}^{e,f}$ of the MLP as well as the cluster centers $\boldsymbol{w}^{cc}$), leading to joint codebook and class label learning. Assuming a set of videos with interest points and their corresponding feature vectors, as well as a groundtruth action label per video, backpropagation propagates the error between the stimulated response and the ground truth back to the input layers, adjusting weights during the process. The weights $\boldsymbol{w}^{e,f}$ of

Algorithm 1: The iterative codebook learning framework

Input: F_{tr} (training features), F_{val} (validation features)
Output: $\boldsymbol{w}^{cc}$ (optimal codebook)

$\boldsymbol{w}^{cc} \leftarrow$ k-means ;
repeat
  $\boldsymbol{B}_{tr} \leftarrow$ Bags-of-words computation $(\boldsymbol{w}^{cc}, F_{tr})$;
  $\boldsymbol{B}_{val} \leftarrow$ Bags-of-words computation $(\boldsymbol{w}^{cc}, F_{val})$;
  $\boldsymbol{w}^{e,f} \leftarrow$ random ;
  $\boldsymbol{w}^{e,f} \leftarrow$ MLP learning $(\boldsymbol{w}^{e,f}, \boldsymbol{w}^{cc}, \boldsymbol{B}_{tr}, L_{tr})$;
  $\boldsymbol{w}^{cc} \leftarrow$ Cluster center learning $(\boldsymbol{w}^{e,f}, F_{tr}, L_{tr})$ { *section 3.2 or 3.3* } ;
  $E \leftarrow$ Validation error $(\boldsymbol{w}^{e,f}, \boldsymbol{w}^{cc}, \boldsymbol{B}_{val}, L_{val})$;
until *convergence(E)* ;

the MLP are updated with a classical error backpropagation scheme, which is recalled in section 3.1. The cluster centers can be updated by two algorithms, which are respectively addressed in section 3.2 and 3.3. They make use of the backward errors of the learned MLP. The pseudo code of the proposed framework is shown in Algorithm 1.

3.1 MLP learning

Error backpropagation is a common method to learn neural networks due to its simplicity and efficiency. A MLP network with multiple output units is adopted. Sigmoid and softmax activation functions are respectively employed in the hidden layer and output layer. An 1-of-*c* coding scheme is used to describe the target output. Here we briefly present the classical way :

1. The entity $\boldsymbol{d}$ forward propagates through the network, and the activations of all hidden and output layer units are computed.
2. Given a desired output t_j from the groundtruth, compute the error in the output layer :
$$\delta_j^f = (f_j - t_j)$$
3. Backpropagate the error into the hidden layer and the input layer :
$$\delta_j^e = g'(\bar{e_j}) \sum_k w_{jk}^f \delta_k^f \quad \delta_j^d = \sum_k w_{jk}^e \delta_k^e$$
4. Compute the increments for all the weights :
$$\Delta w_{ij}^f = \eta \delta_j^f \bar{e}_i \quad \Delta w_{ij}^e = \eta \delta_j^e \bar{d}_i,$$
where η is a learning parameter.

3.2 Supervised codebook learning with error backpropagation

The classical error backpropagation algorithm can be adapted to our novel formulation. In particular, the errors in the input layer $\boldsymbol{d}$ (the errors on the BoW histograms) are continued backpropagating into layer $\boldsymbol{b}$, i.e. the layer which is directly related to the cluster centers. Due to the integrator, the errors in layer $\boldsymbol{c}$ need to be approximated by taking the error of layer $\boldsymbol{d}$ and equally distributing it over the corresponding feature points :

$$\delta_i^c = \frac{\delta_i^d}{d_i}. \tag{6}$$

Subsequently, the error can be further backpropagated to the previous layer and the cluster centers $\boldsymbol{w}^{cc}$ can be updated in the same manner as the weights of the MLP. In the above, the distance of each node i of layer $\boldsymbol{b}$ can be computed as follows :

$$b_i = ||\boldsymbol{a} - \boldsymbol{w}_i^{cc}||^2 = \sum_j (a_j - w_{ij}^{cc})^2. \tag{7}$$

We also resort to gradient descent to adjust the cluster centers by solving a second-order polynomial.
Let us first note that sum-of-error is used at layer $\boldsymbol{c}$:

$$E = \sum_m E^m = \sum_m \sum_i \frac{1}{2}(c_i^m - \widetilde{c_i^m})^2 \tag{8}$$

where m corresponds to the index of the input feature vector and c_i^m is the forward response of the network for feature m, $\widetilde{c_i^m}$ is the optimal value for the best cluster center according to the groundtruth, which can be approximated using equation (6). According to gradient descent :

$$\frac{\partial E}{\partial w_{ij}^{cc}} = \sum_m \frac{\partial E^m}{\partial w_{ij}^{cc}} = \sum_m \frac{\partial E^m}{\partial b_i^m} \frac{\partial b_i^m}{\partial w_{ij}^{cc}} \tag{9}$$

In the following, only the values for a single feature m will be considered. For clarity we remove the superscript m from the notation, in particular E^m will be noted as E. Given the derivative of the approximated softmin function and the derivative of b_i with respect to w_{ij}^{cc}, we have :

$$\frac{\partial E}{\partial w_{ij}^{cc}} = \frac{2}{T}(a_j - w_{ij}^{cc}) \sum_{i'} \frac{\delta_{i'}^c}{T}(c_{i'}\delta_{ii'} - c_{i'}c_i). \tag{10}$$

So far we have obtained the gradient information of backward errors with respect to the weights (cluster centers) in the network. The gradient descent algorithm thus is applied as follows :

$$\Delta w_{ij}^{cc}(t) = \alpha \Delta w_{ij}^{cc}(t-1) - \eta_b \sum_m \frac{\partial E^m}{\partial w_{ij}^{cc}} \tag{11}$$

where α and η_b are learning parameters and the feature vector index m has been used again to distinguish the batch entries.

3.3 Supervised codebook learning through cluster reassignment

In this section we propose another algorithm which uses our prior knowledge that the information stored in layer $\boldsymbol{d}$

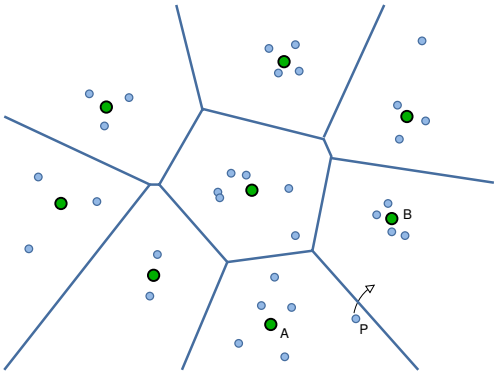

Figure 2 – *Codebook learning through cluster reassignment : a Voronoi diagram of the feature space, for simplicity in 2D. Cluster centers are green and large, training points are blue and small.*

is a BoW, i.e. a histogram. Instead of simply backprojecting an error of this layer through the softmin function, we change it by moving input feature vectors from one histogram bin to another one.

This strategy is illustrated in Figure 2. Supposing the error for cluster A is positive (too many feature vectors) and the error for cluster center B is negative (not enough feature vectors), at each weight update a single feature vector is moved from the Voronoi cell of A into the Voronoi cell of B, followed by an update of the cluster centers as a calculation of the mean of the assigned training points.

For generality, the errors of the BoW layer $\boldsymbol{d}$ are denoted as $\delta^d = \{\delta_0^d, \delta_1^d, \cdots, \delta_N^d\}$. A positive error $\delta_i^d > 0$, indicates that at least one feature vector being assigned to the cluster center corresponding to this bin should been assigned to a different cluster according to the ground truth. In the same sense, a negative error $\delta_j^d < 0$ indicates that at least one feature vector should be added to this histogram bin of the BoW. In the following we suppose that the solution to this problem is specified as a multi set $\boldsymbol{x} = \{x_1, x_2, \ldots, x_D\}$ of indices indicating from where a vector is moved, and a multi set $\boldsymbol{y} = \{y_1, y_2, \ldots, y_D\}$ of indices indicating to which cluster a vector is moved. For instance, $x_1 = 5$ and $y_1 = 7$ indicate that the first move will go from cluster 5 to cluster 7.

A good solution should minimize two criteria. First, the error should be low, i.e. we should minimize

$$\min_{\boldsymbol{x},\boldsymbol{y}} \left[\sum_k \delta_k^d - |\{x_s : x_s = k\}| + |\{y_s : y_s = k\}| \right] \quad (12)$$

where $|\{x_s : x_s = k\}|$ is the number of source indices equal to k, and the second expression can be understood in a similar way. Secondly, the feature vector movements performed by the solution should be minimal, i.e. we should minimize

$$\min_{\boldsymbol{x},\boldsymbol{y}} \left[\sum_{s=1}^{D} |b_{x_s} - b_{y_s}| \right] \quad (13)$$

One possibility would be to minimize an energy function consisting of a weighted linear combination of (12) and (13). Instead, we opted for an iterative greedy solution, where cluster pairs (i, j) are chosen decreasing (12) and then for each pair of clusters a feature vector is chosen such that its move from cluster i to cluster j minimizes (13). We added an additional constraint requiring that the chosen feature vector to move — which is (naturally) closest to cluster center A — also be second closest to cluster center B. The details of the update algorithm are given as follows :

1. Randomly choose a pair (i, j) of histogram bins (thus of cluster centers), where the error of one bin is positive and the other is negative, i.e. $\delta_i^d > 0 \ \wedge \ \delta_j^d < 0$.
2. Calculate the Voronoi diagram of the cluster centers in feature space and determine all the feature vectors of the training set falling into the sets of $\boldsymbol{w}_i^{cc}$ and $\boldsymbol{w}_j^{cc}$, respectively.
3. Pick a single feature vector f such that :
   - it falls into the Voronoi cell i.
   - its distance to cluster center $\boldsymbol{w}_j^{cc}$ is second nearest.
   - if several vectors satisfy the above two criteria, choose the one minimizing the distance to the border of the two Voronoi cells, i.e. the one minimizing $|b_i - b_j|$.
4. The chosen feature vector f is reassigned from histogram bin i to histogram bin j with the following consequences :
   - the two centers $\boldsymbol{w}_i^{cc}$ and $\boldsymbol{w}_j^{cc}$ are recalculated as the means of the feature vectors of their respective Voronoi cells.
   - the errors of the BoW layer of the corresponding bins are updated :

$$\delta_i^{d\ [t+1]} = \delta_i^{d\ [t]} - 1, \quad \delta_j^{d\ [t+1]} = \delta_j^{d\ [t]} + 1$$

5. The reassignments are continued (back to step 1) until the error of layer $\boldsymbol{d}$ is zero or none of the feature vectors satisfy the above conditions.

4 Experimental Results

The proposed model and learning algorithms have been evaluated on the publicly available KTH action dataset [15]. It is one of the largest available published datasets and contains 6 actions – boxing, hand clapping, hand waving, jogging, running and walking, performed by 25 subjects in four different scenarios – indoors, outdoors, outdoors with scale variation and clothes changing. It contains 2391 video sequences and each sequence lasts four seconds in average. For video representation, space-time interest points were detected by the 3D Harris-corner detector proposed by Laptev and features of histogram of gradient (HOG) and histogram of oriented flow (HOF) descriptors were employed [14].

As usual, a cross-validation scheme and early stopping strategy are employed to control the MPL learning phase.

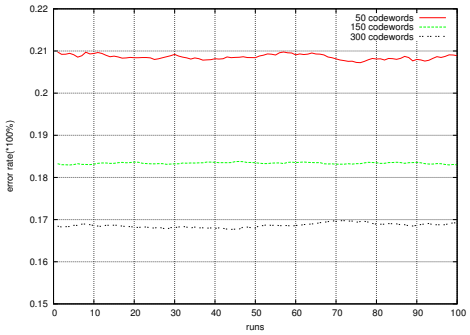

Figure 3 – *Classical unsupervised learning (clustering) : errors on the test set over different MLP learning runs. Each shown error is measured after convergence of a whole MLP learning run, i.e. several epochs.*

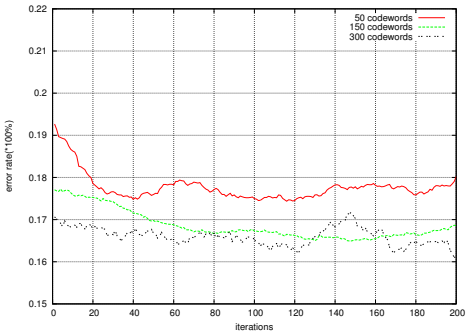

Figure 4 – *Supervised learning with error backpropagation (section 3.2) : errors on the test set over different iterations.*

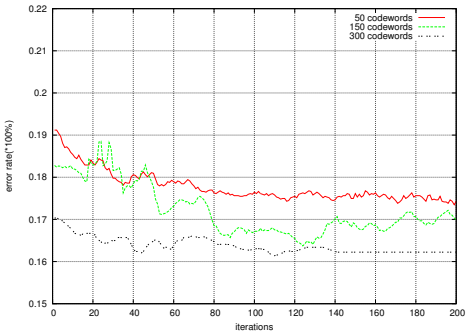

Figure 5 – *Supervised learning with cluster reassignment (section 3.3) : errors on the test set over different iterations.*

The dataset is divided into three independent sets : training (12 people), validation (4 people) and test (9 people), as in [15]. The MLP is trained on the training set and evaluated on the validation set for stopping to avoid over-fitting. Unless said otherwise, all errors are reported on the test set. Different MLP architectures and learning hyper-parameters were tested, and the best ones were chosen from the performances on the test set. Values are given below. Executions times are given for a C++ implementation running on an Intel Core i7 640 PC under Linux.

Classical unsupervised codebook learning — To demonstrate the discriminative power of classical codebook and compare with our methods, we created a baseline with classical unsupervised *k*-means clustering and MLP learning of the BoW descriptors. The class labels of training set were not taking into considering in codebook creation. To cope for random initialization of the MLP weights, we repeated our baseline experiments in order to obtain statistically sound results : for each run, the cluster centers were kept fixed and the MLP weights were randomly initialized between -0.5 and 0.5 and learned. We ran 100 runs for each codebook in our experiments. The numbers of hidden units are set to 25, 75, 100 for 50, 150, 300 codewords.

Figure 3 shows error rates (on the test set) of the learned MLP with different codebooks. A local running mean filter was applied to the results. We can see that the MLP learning is robust.

Supervised codebook learning with error backpropagation — Results with supervised learning through the backpropagation method (section 3.2) are shown in Figure 4. We repeated the above experiments with the same architecture, except that the cluster centers were adjusted by using a gradient descent algorithm according to the back-propagated errors of the optimal MLP in each iteration and the BoW entities of the videos were recomputed. We tried several values and selected the best parameters for gradient descent. α was set to 0.00001 for all the codebooks and η_b varied for different codebooks. The error rates are depicted in Figure 4, again after applying a local running mean to smooth the data. It can be seen that the error decreases at the beginning and converges for 50 codewords and for 150 codewords. However, for 300 codewords the error oscillates after 120 iterations due to the non-adaptive characteristics of gradient descent. Comparative results are presented in Table 1. We can see that supervised learning codebook through error backpropagation approximately gains 2.1% for 50 codewords and 0.8% for 300 codewords with respect to the baseline codebooks obtained with *k*-means clustering.

Supervised codebook learning with cluster reassignment — Results with supervised learning through the cluster reassignment method (section 3.3) are shown in Figure 5. We again repeated the above experiments with the same neural architecture. At each iteration, the cluster centers were adjusted using the Voronoi cell updates, and then the MLP is retrained. Figure 5 shows the results, which are obtained by applying a local running mean. As we can see the classification accuracy on the test set increases as the cluster centers are adjusted. The learned codebooks through cluster reassignment therefore improve by 1.7% for 50 codewords and 0.8% for 300 codewords with respect to the baseline codebooks obtained by *k*-means clustering.

We also did additional experiments using other classifiers to verify the discriminative quality of our learned codebooks. We trained a SVM with a radial basis function kernel based on each codebook on the training set and vali-

Table 1 – *Error in (%) on the test with different methods, different classifiers and different codebook sizes : mean and standard deviation over 3 independent runs.*

	Codebook	50	150	300
MLP	baseline	19.31(±0.26)	17.44(±0.31)	16.98(±0.64)
	method 1	**17.19**(±0.51)	16.56(±**0.07**)	**16.13**(±0.12)
	method 2	17.57(±**0.11**)	**16.29**(±0.53)	16.18(±**0.07**)
	recognition time per video(ms)	1.679	4.696	9.294
SVM	baseline	17.16(±0.66)	15.47(±1.06)	15.7(±0.24)
	method 1	**15.94**(±0.25)	15.30(±0.33)	**14.54**(±0.41)
	method 2	16.80(±**0.16**)	**14.89**(±**0.24**)	14.89(±**0.24**)
	recognition time per video(ms)	1.797	4.905	10.488

dation set, which were the same with the ones used in the above experiments. The errors are shown in the lower block of Table 1. We can see that the classification performance of our two joint supervised methods is maintained after retraining with a different classifier, indicating a real gain in discriminative power of the codebooks, and both methods clearly improve the discriminative quality of the codebook when the codebook size is small.

In this paper we proposed two different joint supervised learning algorithms. The reformulated backpropagation algorithm adjusts the cluster centers directly through gradient descent. Two more parameters besides the learning rate η in MLP learning need to be set : the momentum coefficient α and the learning rate η_b. It is difficult to learn a set of optimal parameters, mostly converging to local minima and sometimes even diverging. In comparison, the cluster reassignment algorithm adjusts the cluster centers indirectly by rearranging the cluster labels for all the feature vectors. It does not need any more learning parameters except η, and is easier to control. In Figure 4, although the error with 300 codewords began to converge after 60 iterations, it begin to diverge from 120 iterations. However it converges after 140 iterations in Figure 5.

5 Conclusion

In this paper we proposed a joint supervised codebook learning and optimization framework with neural network, which integrates the codebook learning and the recognition phase together. The codebook therefore is created in a goal-directed way and is more discriminative than classical ones. We have presented two algorithms to update the cluster centers (codewords) through the back-propagated errors : error backpropagation and cluster label reassignments. Our framework has been tested on the public KTH action dataset, and the experimental results have confirmed that our framework is able to optimize the codebooks and that it makes the codebook more discriminative for both methods. At the same time, they demonstrated that error backpropagation learned the optimal codebook faster than cluster reassignment. However it need tune more hyperparameters, while cluster reassignment is easier to control.

Références

[1] Csurka G, Dance C, Fan LX, Willamowski J, Bray C. Visual categorization with bags of keypoints. Proc. of *ECCV International Workshop on Statistical Learning in Computer Vision*, pages. 1–22, 2004.

[2] Dollar P, Rabaud V, Cottrell G, Belongie S. Behavior recognition via sparse spatio-temporal features. In *ICCV Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pages. 65–72, 2005.

[3] Moosmann F, Triggs B, Jurie F. Fast discriminative visual codebooks using randomized clustering forests In *NIPS*, pages. 985–992, 2007.

[4] Lowe DG. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60(2) :91–110, 2006.

[5] Liu J, Shah M. Learning human actions via information maximization. In *CVPR*, pages. 1–8, 2008.

[6] Niebles JC, Wang H, Fei-Fei L. Unsupervised learning of human action categories using spatial-temporal words. *International Journal of Computer Vision (IJCV)*, 79(3) :299–318, 2008.

[7] Gilbert A, Illingworth J, Bowden R. Action recognition using mined hierarchical compound features. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 33(5) :883–897, 2011.

[8] Ryoo MS, Aggarwal JK. Spatio-temporal relationship match : Video structure comparison for recognition of complex human activities. In *ICCV*, pages. 1593–1600, 2009.

[9] Ta A-P, Wolf C, Lavoué G, Baskurt A, Jolion JM. Pairwise features for human action recognition. In *ICPR*, pages. 3224–3227, 2009.

[10] Mikolajczyk K, Uemura H. Action recognition with appearance–motion features and fast search trees. *Computer Vision and Image Understanding (CVIU)*, 115(3) :426–438, 2011.

[11] Liu J, Yang Y, Shah M. Learning semantic visual vocabularies using diffusion distance. In *CVPR*, pages. 461–468, 2009.

[12] Saghafi B, Farahzadeh E, Rajan D, Sluzek A. Embedding visual words into concept space for action and scene recognition. In BMVC, pages. 1–11, 2011.

[13] Ballan L, Bertini M, and Bimbo A.D. Effective codebooks for human action categorization. In *ICCV* Workshop on *VOEC*, 2009.

[14] Laptev I, Marszalek M, Schmid C, Rozenfeld B. Learning realistic human actions from movies. In *CVPR*, pages. 1–8, 2008.

[15] Schuldt C, Laptev I, Caputo B. Recognizing human actions : a local svm approach. In *ICPR*, pages. 32–36, 2004.