Fast-BoW: Scaling Bag-of-Visual-Words Generation

Dinesh Singh
cs14resch11003@iith.ac.in
Abhijeet Bhure
cs15btech11001@iith.ac.in
Sumit Mamtani
cs15btech11022@iith.ac.in
C. Krishna Mohan
ckm@iith.ac.in

Visual Learning & Intelligence Group
Department of Computer Science and Engineering
Indian Institute of Technology Hyderabad
Kandi, Sangareddy, Telangana, India

Abstract

The bag-of-visual-words (BoW) generation is a widely used unsupervised feature extraction method for the variety of computer vision applications. However, space and computational complexity of bag-of-visual-words generation increase with an increase in the size of the dataset because of computational complexities involved in underlying algorithms. In this paper, we present *Fast-BoW*, a scalable method for BoW generation for both hard and soft vector-quantization with time complexities $O(|\mathbf{h}|\log_2 k)$ and $O(|\mathbf{h}|k)$, respectively[1]. We replace the process of finding the closest cluster center with a softmax classifier which improves the cluster boundaries over k-means and also can be used for both hard and soft BoW encoding. To make the model compact and faster, we quantize the real weights into integer weights which can be represented using few bits $(2-8)$ only. Also, on the quantized weights, we apply the hashing to reduce the number of multiplications which makes the process further faster. We evaluated the proposed approach on several public benchmark datasets. The experimental results outperform the existing hierarchical clustering tree-based approach by ≈ 12 times.

1 Introduction

Due to advances in the content generation and sharing techniques, a large amount of visual data is available which can be exploited for the variety of applications such as content-based retrieval, classification, action/activity recognition, etc. The bag-of-visual-words (BoW) is an important task for unsupervised representation of visual data based on the local feature descriptors [5, 6, 9, 10, 17, 21, 22]. Recently, there has been substantial research on BoW representation [1, 2, 3, 12, 20]. The BoW generation process involves two phases, 1) *vector quantization for vocabulary generation* that is typically performed by k-means clustering algorithm, and 2) *frequency histogram generation* using nearest neighbour search.

During the the vector quantization, the goal is to construct a vocabulary $\mathcal{V}$ with a small average quantization error. For a given $\mathcal{L} = \{\mathcal{F}_1, \cdots, \mathcal{F}_n\}$, the set of local feature descriptors $\mathcal{F}_i = \{\mathbf{f}_1, ..., \mathbf{f}_{m_i}\}, \mathbf{f}_i \in \mathbb{R}^d$ where d is the dimension of the local feature descriptor for

[1] Here, $|\mathbf{h}|$ is the size of the hash table used in the proposed approach and k is the vocabulary size

n training videos. The input to the vector quantization algorithm is a set of m vectors $\mathcal{Q} = \{\mathbf{f}_1, ..., \mathbf{f}_m\}, \mathbf{f}_i \in \mathbb{R}^d$ where $\mathcal{Q} \subseteq \{\mathcal{F}_1 \cup \cdots \cup \mathcal{F}_N\}$. The output of the algorithm is a set of k vectors $\mathcal{V} = \{\mu_1, ..., \mu_k\}, \mu_i \in \mathbb{R}^d$, where $k \ll m$. The set $\mathcal{V}$ is called the vocabulary. We say that $\mathcal{V}$ is a good vocabulary for $\mathcal{Q}$ if for most $f \in \mathcal{Q}$ there is a representative $\mu \in \mathcal{V}$ such that the Euclidean distance between f and μ is small. The average quantization error of $\mathcal{V}$ with respect to $\mathcal{Q}$ is defined as

$$J(\mathcal{V}, \mathcal{Q}) = \mathbb{E}\left[\min_{1 \le j \le k} \|F - \mu_j\|^2\right] = \frac{1}{m}\sum_{i=1}^{n} \min_{1 \le j \le k} \|\mathbf{f}_i - \mu_j\|^2, \tag{1}$$

where $\|\cdot\|$ denotes Euclidean norm and the expectation is over F drawn uniformly at random from $\mathcal{Q}$. The k-optimal set of visual words is defined to be the vocabulary $\mathcal{V}$ of size k for which $J(\mathcal{V}, \mathcal{Q})$ is minimized. Typically, k-means algorithm is used for this whose per epoch computational complexity is $O(mdk)$. The average quantization error for a general sets of unit diameter in $\mathbb{R}^d$, is roughly $k^{-2/d}$ for large k [4]. For high dimensional local feature vectors where d is high, it becomes too time consuming. For instance, if $d = 100$, and A is the average quantization error for k_1 visual words, then to guarantee a quantization error of $A/2$, one needs a vocabulary of size $k_2 \approx 2^{d/2}k_1$: that is, 2^{50} times as many visual worlds just to halve the error. In other words, vector quantization is susceptible to the same curse of dimensionality that has been the bane of other non-parametric statistical methods.

However, increasing the vocabulary size k increases the time to generate the frequency histograms which requires $O(dk)$ real values vector-vector multiplications per local feature descriptor in a video. As in real-world application, this computation is performed on the fly and thus it effects their real-time performance. The typical solution to this problem is to maintain tree hierarchy of the clusters. The use of a tree leads to $O(d \log_2 k)$ but invariably leads to a significant fall in the effectiveness of the generated BoW features. Dasgupta *et al.* [2] proposed a set of hierarchical random projection trees for vector quantization and subsequent search of the right subspace by traversing each tree and finally making a consensus. This approach reduces the time complexity in tree construction in comparison to other tree methods. But the use of several trees increases the time of BoW generation. Also, due to hard splitting criteria, the projection tree-based algorithms suffer from the high loss in classification accuracy. The proposed *Fast-BoW* addresses both issues, namely, loss in accuracy and computational time by learning probability distribution of the clusters at each level of the tree and then applying quantization and hashing to reduce the vector operations. The contributions of the proposed approach are:

- We develop *Fast-BoW*, which is a scalable algorithm for the generation of the BoW that is essential in many computer vision approaches [5, 6, 9, 10, 15, 21, 22].
- We reduce the number of *integer-real* multiplication upper bounded with $|\mathbf{h}|$ the size of hash table in the worst case while completely avoiding *real-real* multiplications.
- *Fast-BoW* is orders of magnitudes faster than sequential BoW and tree based BoW approaches to obtain a desired time performance. As a specific example, on the *KTH* data sets with $\bar{m} = 849$, the average local feature descriptors in a video and $k = 65536$, *Fast-BoW* runs $4000\times$ and $12\times$ faster than the sequenial and tree approaches, respectively, while reduces the loss in classification to less than 0.5%, while the tree approach drops upto 5%.

- Also, the Fast-BoW brings the ability to control the trade-off between computation efficiency and classification quality in the prediction phase.

The remainder of this paper is organized as follows. Section 2 presents the proposed *Fast-BoW* for fast generation of the BoW. Section 3 discusses the experimental setup, dataset, and performance of *Fast-BoW*. Finally, we conclude in section 4.

2 Proposed Fast-BoW

As shown in the Fig. 1, the proposed approach first uses the clustering algorithm (like k-means) for vector quantization of the training vector space into the pre-defined number of clusters. Then to learn the probability distribution of each cluster, we train a soft-max classifier. The class labels for the soft-max are the cluster index of each point received from the clustering algorithm. Also, the large weight matrices have a significant redundancy that can be avoided by applying model compression techniques. We exploit this fact to reduce the memory and computation time by applying weight quantization and hashing. The weight quantization reduces the number of levels (i.e. unique floating point numbers) in a weight matrix and the hashing reduces the number of floating point multiplications needed for a *matrix-vector* or *vector-vector* multiplication.

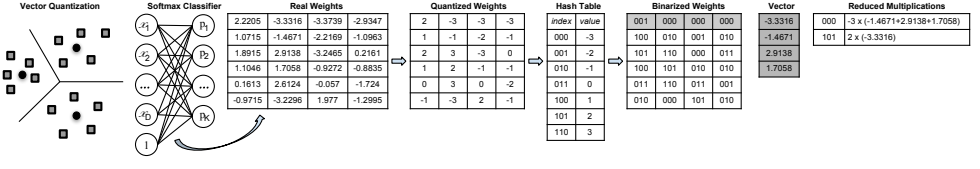

Figure 1: Scaling the process of BoW generation.

2.1 Learning Probability Distribution of the Clusters

First, we learn k clusters using k-means on the local feature descriptors. As the cluster learning from a large number of local feature descriptors is time-consuming, thus we first use a small sample to learn the initial clusters $\{\mu_1, \cdots, \mu_k\}$ and later refinement the centers using the remaining points only once as suggested by Raghunathan *et al.* [13]. Let, for i^{th} local feature descriptor $\mathbf{f}_i$, μ_j^i be the closest center then it updates the μ_j^i as

$$\mu_j^i = (1-\eta)\mu_j^i - 1 + \eta \mathbf{f}_i, \tag{2}$$

where, $\eta = \frac{3k \log_3 m}{m}$ gives the best results. After successfully learning the means of the k clusters, the next task is to learn the probability distribution for soft/hard cluster assignment for that Gaussian mixture model (GMM) can be used. However, GMM's training and prediction are time-consuming and also speed-up of them is also not so easy. Thus, we learn a softmax classifier on the given data where the labels of each vector are the cluster index previously given by the k-means. Doing so helps in learning the soft boundaries for the clusters that can be applied to both types of hard and soft BoW generations. The probability of a

local feature descriptor $\mathbf{f}_i$ belongs to j^{th} cluster is

$$p(y_i = j|\mathbf{f}_i;\theta) = \frac{e^{\theta_j^T \mathbf{f}_i}}{\sum_{j=1}^{k} e^{\theta_j^T \mathbf{f}_i}} \tag{3}$$

The θ is learned by maximization of the class cross entropy. The object function for the softmax classifier along with regularization is

$$J(\theta) = -\frac{1}{n}\left[\sum_{i=1}^{n}\sum_{j=1}^{k} I(j = y_i)\log\left(\frac{e^{\theta_j^T \mathbf{f}_i}}{\sum_{j=1}^{k} e^{\theta_j^T \mathbf{f}_i}}\right)\right] + \frac{\lambda}{2}\sum_{i=1}^{k}\sum_{j=1}^{d}\theta(i,j)^2 \tag{4}$$

The loss function in Equation (4) is solved using scaled conjugate gradient[11].

2.2 Weight Quantization and Hashing

The parameter θ contains $k.d$ real numbers and the computation of $p(y_i = j|\mathbf{f}_i;\theta)$ requires computation of k inner product of d-dimensional real-valued vectors. Although, its cheaper than the k-means and GMM it still poses significant challenges. Also, a known fact is that the θ contains significant redundancy and the operations on real numbers (generally represented using 32-64 bits) are much more complex than the integer-real numbers. In order to exploit these facts, first we apply quantization on the values of the parameter θ and then maintain a hash table $\mathbf{h}$ where the quantization and hash functions are defined as follows:

Definition 1. *The quantization function $q : \mathbb{R} \to \mathbb{Z}$ on a real scalar parameter $\theta \in \mathbb{R}$ is defined as*

$$z = q(\theta) = \left\lfloor \frac{\theta}{\max(abs(\theta))} \times L \right\rceil, \tag{5}$$

Definition 2. *The hash function $h : \mathbb{Z} \to \mathbb{Z}^*$ on a integer key $z \in \mathbb{Z}$ is defined as*

$$h(z) = z + L, \tag{6}$$

where, $\lfloor\cdot\rceil$ denotes the nearest integer and L is a free parameter such that $|\mathbf{h}| = 2L+1$ is the size of the hash table. The Algorithm 1 provides the pseudocode of the procedure of the learning softmax parameters, quantization, and generation of hash table.

Algorithm 1 Fast-BoW: Gen_Vocabulary

Input: $\mathcal{Q}$: Set of local feature descriptors for vocabulary generation, m : Cardinality of the $\mathcal{Q}$, i.e., $|\mathcal{Q}|$, k : Size of the vocabulary i.e., $|\mathcal{V}|$
Output: θ^* : Softmax parameters, $\mathbf{h}$: Hash table
gen_vocabulary
$[\mathcal{V}, idx] = k\text{-}means(\mathcal{Q}, k)$;
$model = softmax(X = \mathcal{Q}, Y = idx)$;
$\theta = model.\theta$;
$H =$ Unique values in the range of $q(\theta)$;
$\theta^* = h(q(\theta))$; {Replace the values of the θ with its hash value (i.e. indexes in $\mathbf{h}$)}

Once, we get $\mathbf{h}$ and θ^*, then the BoW histogram is generated using Algorithm 2 where for each local feature descriptor, it first groups and then do summation of its values in a hash bucket $\mathbf{s}$ according to similar parameter indexes in respective θ^*_j's then do the multiplication of the real values vector $\mathbf{s}$ and the quantized integer-valued vector $\mathbf{h}$ which can be represented using less number of bits. Doing so reduces the number of multiplications to less than $|\mathbf{h}|$ which also contains one operand as short integer thus speeds up the entire process and the complexity reduces to $O(|\mathbf{h}|\,k)$ from the $O(d\,k)$.

Algorithm 2 Fast-BoW: Gen_Histogram

Input: $\mathcal{F} = \{\mathbf{f}_1, \cdots, \mathbf{f}_m\}$:Set of m local feature descriptors of a video, $\mathbf{h}$:Hash table, θ^* :Hash values of parameters, k :#Clusters, d :#Dimensions
Output: $\mathbf{x}$:Global BoW feature descriptor
gen_BoW

```
for i = 1 : m do
    max_idx = −1;   max_sum ← −∞;
    for j = 1 : k do
        s ← 0;   sum ← 0;
        for l = 1 : d do
            S_{θ*_jl} += f_il;
        end for
        for l = 1 : |h| do
            sum += s_l × h_l;
        end for
        if sum > max_sum then
            max_sum ← sum;   max_idx ← j;
        end if
    end for
    x_{max_idx} += 1;
end for
```

2.3 Hierarchical Tree for Hard BoW generation

In the hard BoW generation, we need to find most probable cluster only unlike soft BoW where we need to compute the probabilities with all the clusters. Thus to further speed up and sustain the classification accuracy of the simple hierarchical trees, we apply similar techniques as discussed above. At each internal node, we learn and maintain a hash table of softmax classifier's parameters with $k = 2$. Thus, the final complexity of finding a cluster reduces to $O(|\mathbf{h}|\log_2 k)$ from $O(|\mathbf{h}|\,k)$ for soft clustering and $O(d\log_2 k)$ for simple hierarchical trees.

3 Empirical Evaluation

All the methods are implemented in C++. The experiments are conducted on a machine with Intel(R) Xeon(R) CPU E5620@2.40GHz processor and 32GB RAM. We conducted experiments on various large-scale and challenging video datasets for human action recognition, namely, *KTH* [14], *HMDB51* [7] and *UCF101* [19]. The space-time interest points

(STIP) [8] are used as the local features descriptors for the generation of the BoW per video clip. Table 1 provide the details of the datasets used in the experiments.

Table 1: Details of the datasets used

Dataset	Classes	Train Videos	Test Videos	Avg.Length (Desc.)
KTH	6	383	216	4 sec. (849)
HMDB51	51	3,567	1,530	5 sec. (1456)
UCF101	101	9,535	3,782	7.21 sec. (1574)

The sequential BoW generation (Seq-BoW) and tree-based BoW generation (Tree-BoW) approached are used as the baseline for both the effectiveness (classification accuracy) and the computational time. To measure the effectiveness of the generated BoW features, we use the linear SVM with default hyper-parameters for all the existing and proposed approaches because the objective of this research work is to scale the process of BoW generation while preserving the effectiveness of the generated features instead of improving the state-of-the-art on these datasets. However, one can use a scalable kernel SVMs [16, 18] with a suitable kernel to further improve the classification accuracy. Also, we keep the train and test split as described in the respective dataset. In case of multiple train-test splits, the performance is the averaged for the splits. The performance of both the existing and the proposed approach are compared for various sizes of the vocabularies (i.e. $k = \{128, 1024, 8192, 65536\}$). Fig. 2 illustrate the challenge of the rapid increase in the computation time for the sequential BoW with an increase in the vocabulary size on the *KTH* action dataset. It shows that large vocabulary based BoW increase the classification performance but also increase the time taken in the BoW generation rapidly and after certain vocabulary size it becomes impractical to be used in real-time. For example for the vocabulary sizes $k = 8192$ and $k = 65536$ the time taken by Seq-BoW is 3.295 and 31.076 seconds, respectively for the video of 4 second duration.

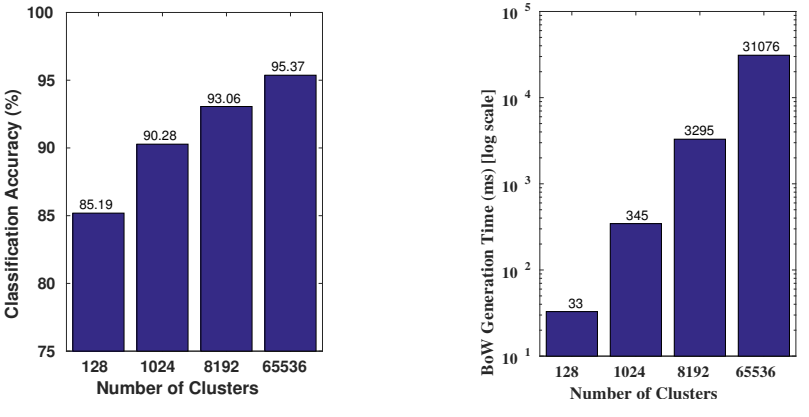

Figure 2: Effect of the vocabulary size on classification accuracy and BoW generation time.

Table 2 shows the performance of the classification for existing and proposed methods on various sizes of the vocabularies. The results demonstrate that the proposed approach *Fast-BoW* is significantly reduced the loss in the effectiveness of the generated BoW in comparison to the hierarchical clustering based tree approach. The results in Table 3 illustrate

Table 2: Comparison of the classification performance (%) of the proposed Fast-BoW approach with the existing Seq-BoW and Tree-BoW approaches

	Number of Clusters (k)			
Dataset+Method	**128**	**1024**	**8192**	**65536**
KTH+Seq-BoW	85.19	90.28	93.06	95.37
KTH+Tree-BoW	79.17	88.89	86.11	90.74
KTH+Fast-BoW	**82.41**	**89.81**	**89.96**	90.74
HMDB51+Seq-BoW	08.17	13.99	15.16	15.37
HMDB51+Tree-BoW	06.54	11.11	13.40	11.50
HMDB51+Fast-BoW	**07.19**	**13.59**	**13.79**	**14.38**
UCF101+Seq-BoW	16.13	30.88	36.20	39.66
UCF101+Tree-BoW	12.14	27.10	32.84	32.18
UCF101+Fast-BoW	**14.99**	**27.47**	**35.69**	**37.82**

that the *Fast-BoW* outperform all the existing approaches with $\approx 4000\times$ and $\approx 12\times$, respectively.

Table 3: Comparison of time (milliseconds) taken in BoW generation per test video.

	Number of Clusters (k)			
Dataset+Method	**128**	**1024**	**8192**	**65536**
KTH+SeqBoW	33	345	3295	31076
KTH+TreeBoW	3	4	5	7
KTH+FastBoW	**0.24**	**0.32**	**0.39**	**0.57**
HMDB51+SeqBoW	40	585	5152	35805
HMDB51+TreeBoW	5	6	9	13
HMDB51+FastBoW	**0.37**	**0.49**	**0.71**	**1**
UCF101+SeqBoW	129	219	9265	18470
UCF101+TreeBoW	14	15	20	26
UCF101+FastBoW	**1**	**1**	**2**	**2**

Values greater than 1 ms are rounded to nearest integers.

Figure 3 illustrate the comparison of the loss in the classification performance of the existing Tree-Bow and the proposed Fast-BoW with respect to the sequential approach Seq-BoW. The figure clearly shows that the loss in the proposed Fast-BoW is significantly lower than the loss in the Tree-BoW for various vocabulary size across all three datasets. Thus, the proposed Fast-BoW preserves the effectiveness of the BoW features in comparison to the existing Tree-BoW approach.

Figure 4 illustrate the speed-up gain by the proposed Fast-BoW with respect to the existing Seq-BoW and Tree-BoW approaches. It can be observed from the figure that the proposed Fast-BoW gains several orders of speed-up over the existing approaches. For sequential approach, it is $\approx 100\times$ faster for the vocabulary size $k = 128$ and this scale increase with the increase in the vocabulary size. Also, it achieves $\approx 12\times$ speed up over the Tree-BoW approach when keeping the $|\mathbf{h}| = \sqrt{d}$, where $d = 162$ is the dimensions of the STIP descriptors. As the reducing the $|\mathbf{h}|$ (for example $|\mathbf{h}| = \log_2 d$) results into further speed-up but also increases the loss in the classification performance. Also, the proposed approach took ≤ 2 milliseconds only for each video of duration $4 - 7.21$ seconds. Thus making the BoW approach practical to be used in real-time with increased vocabulary sizes and in-

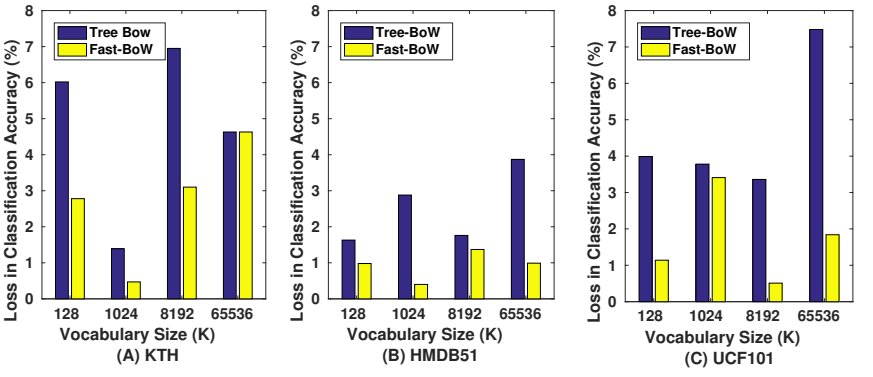

Figure 3: Comparison of the classification loss in existing tree based approach (Tree-BoW) and the proposed Fast-BoW approach with respect to sequential BoW (baseline).

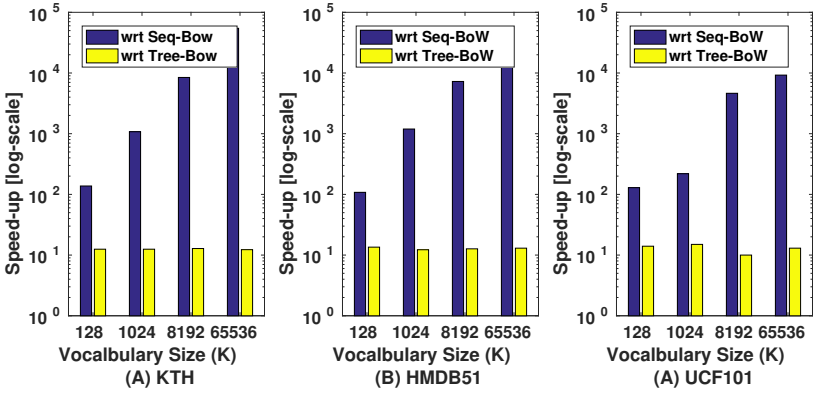

Figure 4: Scaling factor of the proposed approach (Fast-BoW) with respect to sequential approach (Seq-BoW) and the existing tree based approach (Tree-BoW).

creased effectiveness.

4 Conclusion

The proposed *Fast-BoW* scale the computational complexity of hard and soft bag-of-words generation to $O(|\mathbf{h}|\log_2 k)$ and $O(|\mathbf{h}|k)$ from $O(d\log_2 k)$ and $O(dk)$, respectively. While it better preserves the effectiveness of the generated features than the existing hierarchical clustering tree-based approach. The use of softmax for predicting the cluster probabilities for a local feature vector not only improves the clustering performance but also provides the flexibility to further scaling by applying quantization and hashing. The experimental results show that the choosing $|\mathbf{h}| = \sqrt{d}$ gives almost no loss to the final classification accuracy while choosing $|\mathbf{h}| = \log_2 d$ resulted into a mild loss. Thus the proposed method *Fast-BoW*

shows the efficacy to be used in real-time applications with large vocabularies.

References

[1] Kunal Dahiya, Dinesh Singh, and C. Krishna Mohan. Automatic detection of bike-riders without helmet using surveillance videos in real-time. In *2016 International Joint Conference on Neural Networks, IJCNN 2016, Vancouver, BC, Canada, July 24-29, 2016*, pages 3046–3051, 2016.

[2] S. Dasgupta and Y. Freund. Random projection trees for vector quantization. *IEEE Trans. Information Theory*, 55(7):3229–3242, July 2009.

[3] D. Galvez-LÃşpez and J. D. Tardos. Bags of binary words for fast place recognition in image sequences. *IEEE Trans. Robotics*, 28(5):1188–1197, 2012.

[4] Siegfried Graf and Harald Luschgy. *Foundations of Quantization for Probability Distributions*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2000. ISBN 3540673946.

[5] S. S. Husain and M. Bober. Improving large-scale image retrieval through robust aggregation of local descriptors. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 39(9):1783–1796, 2017.

[6] N. Inoue and K. Shinoda. Fast coding of feature vectors using neighbor-to-neighbor search. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 38(6):1170–1184, 2016.

[7] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. Hmdb: A large video database for human motion recognition. In *IEEE Int. Conf. Computer Vision*, pages 2556–2563, Barcelona, Spain, 6-13 Nov 2011.

[8] Ivan Laptev. On space-time interest points. *Int. J. Computer Vision*, 64(2-3):107–123, 2005.

[9] L. Liu, L. Wang, and C. Shen. A generalized probabilistic framework for compact codebook creation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 38(2): 224–237, 2016.

[10] T. MantecÃşn, C. R. del Blanco, F. Jaureguizar, and N. GarcÃ■a. Visual face recognition using bag of dense derivative depth patterns. *IEEE Signal Processing Letters*, 23 (6):771–775, 2016.

[11] Martin Fodslette Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6(4):525–533, 1993.

[12] R. Okutani, K. Sugimoto, and S. i. Kamata. Efficient keypoint detection and description using filter kernel decomposition in scale space. In *IEEE Int. Conf. Image Processing*, pages 31–35, Phoenix, USA, 25–28 Sep 2016.

[13] Aditi Raghunathan, Prateek Jain, and Ravishankar Krishnaswamy. Learning mixture of gaussians with streaming data. In *Advances in Neural Information Processing Systems*, pages 6608–6617, Long Beach, CA, USA, 4-9 Dec 2017.

[14] Christian Schüldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: A local SVM approach. In *Int. Conf. Pattern Recognition*, pages 32–36, Cambridge, UK, 23-26 Aug 2004.

[15] D. Singh and C. K. Mohan. Deep Spatio-Temporal Representation for Detection of Road Accidents Using Stacked Autoencoder. *IEEE Transactions on Intelligent Transportation Systems*, 2018. doi: 10.1109/TITS.2018.2835308.

[16] Dinesh Singh and C. Krishna Mohan. Distributed quadratic programming solver for kernel SVM using genetic algorithm. In *IEEE Congress on Evolutionary Computation, CEC 2016, Vancouver, BC, Canada, July 24-29, 2016*, pages 152–159, 2016.

[17] Dinesh Singh and C. Krishna Mohan. Graph formulation of video activities for abnormal activity recognition. *Pattern Recognition*, 65:265–272, 2017.

[18] Dinesh Singh, Debaditya Roy, and C. Krishna Mohan. DiP-SVM : Distribution Preserving Kernel Support Vector Machine for Big Data. *IEEE Trans. Big Data*, 3(1): 79–90, 2017.

[19] Khurram Soomro, Amir R. Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *CoRR*, abs/1212.0402, 2012.

[20] F. Zeng, Y. Ji, and M. D. Levine. Contextual bag-of-words for robust visual tracking. *IEEE Trans. Image Processing*, 27(3):1433–1447, 2018.

[21] J. Zhao, L. Wang, R. Cabral, and F. De la Torre. Feature and region selection for visual learning. *IEEE Trans. Image Processing*, 25(3):1084–1094, 2016.

[22] W. L. Zhao, C. W. Ngo, and H. Wang. Fast covariant vlad for image search. *IEEE Trans. Multimedia*, 18(9):1843–1854, 2016.