DEEP LEARNING VECTOR QUANTIZATION FOR ACOUSTIC INFORMATION RETRIEVAL

Zhen Huang †, *Chao Weng, Kehuang Li, You-Chi Cheng, Chin-Hui Lee*

School of ECE, Georgia Institute of Technology, Atlanta, GA. 30332-0250, USA
† huangzhenee@gatech.edu

ABSTRACT

We propose a novel deep learning vector quantization (DLVQ) algorithm based on deep neural networks (DNNs). Utilizing a strong representation power of this deep learning framework, with any vector quantization (VQ) method as an initializer, the proposed DLVQ technique is capable of learning a code-constrained codebook and thus improves over conventional VQ to be used in classification problems. Tested on an audio information retrieval task, the proposed DLVQ achieves a quite promising performance when it is initialized by the k-means VQ technique. A 10.5% relative gain in mean average precision (MAP) is obtained after fusing the k-means and DLVQ results together.

Index Terms— Deep neural network, learning vector quantization, k-means, information retrieval

1. INTRODUCTION

Deep learning demonstrates a great success recently in the field of automatic speech recognition (ASR) [1, 2] and computer vision [3]. Video, an important part of the *Big Data* initiative, is believed to contain the richest set of audiovisual information. Video data mining has thus become a critical but challenging problem in recent years [4, 5]. This paper addresses issues related to learning a good acoustic codebook extending the learning vector quantization (LVQ) concept [6, 7] to a deep learning structure for representing the features of the sound tracks from videos. Furthermore, the proposed codeword learning method is a general one that can be easily applied to visual features and other multi-modal features in the related fields.

One of the most popular methods to perform acoustic information retrieval is to code an audio clip with proper "words" to convert it into a *text-like* document and employ methods from statistical information retrieval. The most common way is to extract feature vectors from the audio, learn a codebook and vector quantize the feature vectors into codewords with the codewords being treated as *words* in text retrieval [8]. After getting this text representation of an audio clip, the *bag of words* (BoW) based methods [8] with topic model [9] are often employed to find a vector representation [9]. At the last step, various classifier learning schemes, such as supported vector machines (SVMs) [10], and maximal figure of merit (MFoM) [11], are used to derive models for performing the final retrieval. A good codebook is a key to designing a high-quality BoW-based information retrieval system.

To learn a codebook for vector quantization (VQ), k-means [12] or Linde-Buzo-Gray (LBG) [13] algorithms are the most commonly adopted ones. But k-means/LBG based VQs are designed to minimize quantization distortion which usually use mean squared error (MSE) as a criterion. It is beneficial for data compression and reconstruction but might not be the case for getting a good BoW representation. To improve them, learning vector quantization (LVQ) can be utilized which has been shown to help both ASR [7] and text classification [14]. For the learning method for LVQ, the success of deep learning in ASR [1, 2, 15], especially the success in feature representation [16, 17], inspired us that deep neural network (DNN) might be a good representation learner. Utilizing the strong representation power of the deep learning framework, we propose a novel way to perform LVQ. First, with an initial codebook learnt by k-means/LBG, the codeword for each frame is obtained by standard VQ. Then, the codeword is used as the class label for each frame to train a DNN with cross-entropy as the optimization objective. Each element of the output vector (smoothed by a softmax function) represents the posterior probability with which the input frame belongs to a codeword. A BoW representation of an audio clip is then obtained by propagating frames of the clip through the trained DNN and adding up all the output vectors. We refer to this deep structured LVQ as deep learning vector quantization (DLVQ). The proposed DLVQ method is tested on an audio information retrieval task. A 10.5% relative gain in mean average precision (MAP) [18] is obtained after fusing the k-means and DLVQ results together.

2. BASELINE VQ METHODS FOR DLVQ INITIALIZATION

To learn a codebook, the most commonly adopted algorithms are k-means [12] and LBG [13]. Performing exact k-means or LBG algorithm is not feasible in this task because the two similar methods both suffer from large memory consumption and slow convergence speed when it comes to high vector dimension, large number of samples and large cluster (codeword) size in our audio information retrieval task.

Standard k-means is performed as follows: for a data point, its distance to a cluster is defined as its distance to the centroid of the cluster, where the centroid of a cluster is defined as the mean position of all the data points contained in the cluster. k-means algorithm uses an iterative approach. Normally, the initial positions of k clusters centroids are randomly chosen. In a standard k-means iteration, each data point is labeled to the nearest cluster. After all the data points are labeled, the centroid of each cluster is then updated according to its data points. The labeling step and updating step are iterated until the labels do not change any more. After having found cluster centroids by performing k-means, a codeword label can be assigned to any new data point by finding the nearest centroid to it, and this is called VQ. The problem is NP-hard in Euclidean space for a general number of clusters k. Much research has been done to improve the performance of k-means including utilizing graphics processor units (GPU), and considerable speedups were reported [19]. But the huge memory consumption problem is still serious with high vector dimension, large number of samples and large cluster number which is almost exactly the situation we are facing when building an acoustic codebook. The LBG algorithm that performs in

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

a similar way as k-means suffers the memory problem as well.

In our effort to build a baseline system, to alleviate the problem of memory consumption, we use a level-structured k-means [20] based VQ system as in Fig. 1. At the first level, data points were clustered by k-means into a small number of clusters (n_1) which is much smaller than the desire cluster number. Then at the second level, k-means was performed within each cluster to get a fixed number (n_2) of sub-clusters. By repeating this up to m levels, we can get $\prod_{i=1}^{m} n_i$ clusters. The cluster centroids are then used to do VQ. This method alleviates the memory issue, because we perform k-means on subsets of data and with smaller cluster numbers, but due to the large amount of data in our task (one hour audio recording will generate around 360,000 feature frames in our common experiment setting), we still need to randomly select subsets of training data to perform level structured k-means and we cannot use high dimensional feature vectors.

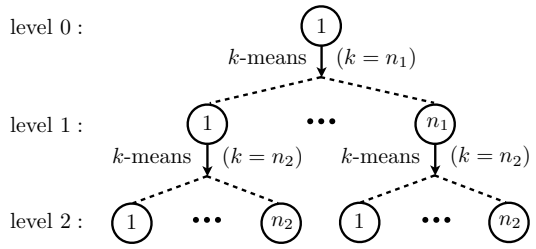

Fig. 1. Performing k-means on a m-level structure with $m = 2$

3. DEEP LEARNING VECTOR QUANTIZER

With the level structured k-means based VQ method in Section 2, an initial codebook and frame level codeword sequence can now be obtained. DLVQ can then be performed based on it. DLVQ follows the concept of LVQ which has been found useful in many fields such as ASR [7] and text classification [14], and at the same time utilizes the strength of deep learning.

3.1. Structure of DLVQ System

The proposed DLVQ in this paper utilizes DNN as a codebook learner and vector quantizer. With the frame level label information obtained from the initial quantizer, a DNN can be trained in a similar way as in DNN based ASR [1]. The overall training structure was shown in Fig. 2. First, an initial codebook is learned by k-means on training frames (no context frames are used). Then the codeword for each frame is obtained by standard VQ. Finally, the codeword is used as the class label for each frame to train a DNN with cross-entropy as the optimization objective.

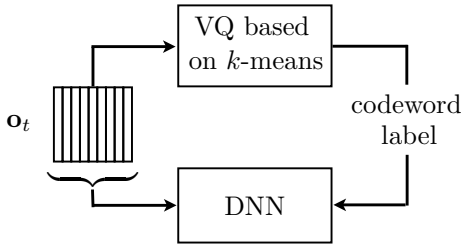

Fig. 2. Structure of Deep Learning Vector Quantizer

3.2. DNN Training

The input of the DNN was a splice of a central frame (whose label is the label for the splice) and its n context frames on both left and right sides, *e.g.,* $n = 8$ or $n = 10$. The hidden layers were constructed by sigmoid units and output layer is a softmax layer which has the same number of nodes as the codeword number of the VQ initializer. The basic structure of a deep neural network is shown in Fig. 3. Specifically, the values of the nodes can be expressed as,

$$\mathbf{x}^i = \begin{cases} W_1\mathbf{o}^t + \mathbf{b}_1, & i = 1 \\ W_i\mathbf{y}^i + \mathbf{b}_i, & i > 1 \end{cases}, \tag{1}$$

$$\mathbf{y}^i = \begin{cases} \mathrm{sigmoid}(\mathbf{x}^i), & i < n \\ \mathrm{softmax}(\mathbf{x}^i), & i = n \end{cases}, \tag{2}$$

where W_1, W_i are the weight matrices and $\mathbf{b_1}, \mathbf{b_i}$ are the bias vectors; n is the total number of the hidden layers and both the sigmoid and softmax functions are element-wise operations. The vector $\mathbf{x}^i$ corresponds to pre-nonlinearity activations and $\mathbf{y}^i$ is the neuron vector at the i^{th} hidden layer. The softmax outputs were considered as the estimated codeword posteriors as in (3),

$$P(C_j|\mathbf{o}_t) = \mathbf{y}_t^n(j) = \frac{\exp(\mathbf{x}_t^n(j))}{\sum_i \exp(\mathbf{x}_t^n(i))}, \tag{3}$$

where C_j represents the j^{th} codeword and $\mathbf{y}^n(j)$ is the j^{th} element of $\mathbf{y}^n$ in Fig. 3.

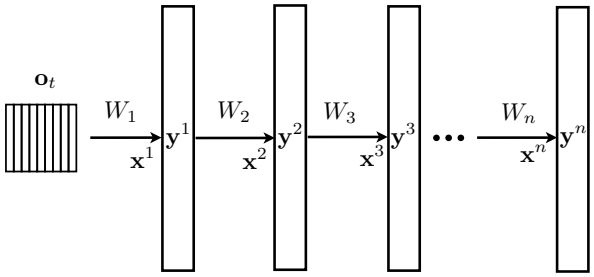

Fig. 3. Basic Structure of a Deep Neural Network:W_i is weight matrix at i^{th} hidden layer, note that the bias terms are omitted for simplicity.

DNN was trained by maximizing the log posterior probability over the training frames. This is equivalent to minimizing the negative cross-entropy loss function. Let $\mathcal{X}$ be the whole training set which contains N frames, *i.e.* $\mathbf{x}_{1:N}^0 \in \mathcal{X}$, then the loss w.r.t. $\mathcal{X}$ is given by,

$$\mathcal{L}_{1:N} = -\sum_{t=1}^{N}\sum_{j=1}^{J} \mathbf{d}_t(j) \log P(C_j|\mathbf{o}_t), \tag{4}$$

where $P(C_j|\mathbf{o}_t)$ is defined in (3); $\mathbf{d}_t$ is the label vector at frame t, which is the "pseudo" one obtained from the k-means VQ initializer. The loss objective function is minimized by using error back propagation which is a gradient-descent based optimization method developed for the neural networks. Specifically, taking partial derivatives of the loss objective function with respect to the pre-nonlinearity activations of output layer $\mathbf{x}^n$ will give us the error vector to be back-propagated to the previous hidden layers,

$$\boldsymbol{\epsilon}_t^n = \frac{\partial \mathcal{L}_{1:N}}{\partial \mathbf{x}^n} = \mathbf{y}_t^n - \mathbf{d}_t, \tag{5}$$

the backpropagated error vectors at previous hidden layer are thus,

$$\boldsymbol{\epsilon}_t^i = W_{i+1}^T \boldsymbol{\epsilon}_t^{i+1} * \mathbf{y}^i * \left(\mathbf{1} - \mathbf{y}^i\right), i < n \tag{6}$$

where $*$ denotes element-wise multiplication. With the error vectors at certain hidden layers, the gradient over the whole training set with respect to the weight matrix W_i is given by,

$$\frac{\partial \mathcal{L}_{1:N}}{\partial W_i} = \mathbf{y}_{1:N}^{i-1} (\boldsymbol{\epsilon}_{1:N}^i)^T, \tag{7}$$

note that in above equation, both $\mathbf{y}_{1:N}^{i-1}$ and $\boldsymbol{\epsilon}_{1:N}^i$ are matrices, which is formed by concatenating vectors corresponding to all the training frames from frame 1 to N, *i.e.* $\boldsymbol{\epsilon}_{1:N}^i = [\boldsymbol{\epsilon}_1^i, \ldots, \boldsymbol{\epsilon}_t^i, \ldots, \boldsymbol{\epsilon}_N^i]$. The batch gradient descent updates the parameters with the gradient in (7) only once after each sweep through the whole training set and in this way parallelization can be easily conducted to speedup the learning process. However, Stochastic gradient descent (SGD) usually works better in practice where the true gradient is approximated by the gradient at a single frame t, *i.e.* $\mathbf{y}_t^{i-1}(\boldsymbol{\epsilon}_t^i)^T$, and the parameters are updated right after seeing each frame. The compromise between the two, the minibatch SGD, is more widely used, as the reasonable size of minibatches makes all the matrices fit into GPU memory, which leads to a more computationally efficient learning process. In this work, we use minibatch SGD to update the parameters.

To train the DNN by minimizing the cross-entropy objective function will make DNN tend to retain the "labels" by its VQ initializer, that is, an "ideal" training cycle will let the DNN get exactly the same VQ results with its initializer. But in the realistic training procedure, it was observed that the frame accuracy is not high (below 50%) for the training and development set. This demonstrates that DNN is not learning what exactly its initializer does but capturing new information in the data.

3.3. Generative Pretraining of DNN

Training a neural network directly from the randomly initialized parameters usually results in a poor local optimum when performing error back propagation, especially when the neural network is deep. To cope with this, pre-training methods have been proposed for a better initialization of the parameters [21]. Pre-training grows the neural network layer by layer without using the label information. Treating each pair of layers in the network as a restricted Boltzmann machine (RBM), layers of the neural network can then be trained using an objective criterion called contrastive divergence [21].

3.4. Getting "Bag of Words" Representation

After pre-training, the DNN can be trained by the error back propagation method. The BoW representation of an audio clip is then obtained by propagating frames of the clip through the trained DNN and adding up all the output vectors as in Fig. 4. It is believed that the deep learning framework can provide more abstract and useful data representations among various learning methods [22]. Moreover, in DLVQ, the context information of each central frame is utilized by splicing context frames which will result in high dimensional feature vector that k-means has a difficulty in handling. All the training data can be used here, unlike for k-means we can only use a small portion. The codeword posteriors output may also be more reasonable than hard decision of a codeword. All these will further help DLVQ get a good representation power.

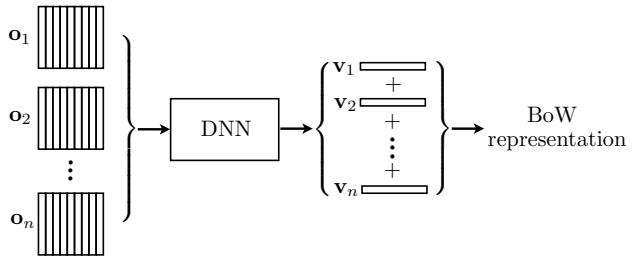

Fig. 4. Getting "Bag of Words" Representation by DLVQ

4. SUPPORT VECTOR MACHINES

In this paper, SVMs [10] with histogram intersection kernel (HIK) [23] was used as a classifier in both the baseline and the proposed system after having obtained the BoW representation of each audio clip. SVMs are widely adopted in field of information retrieval. Its dual formulation of soft margin version is shown in (8):

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} \lambda_j - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \\ \text{subject to} \quad & 0 \leqslant \lambda_i \leqslant C \, and \sum \lambda_i y_i = 0 \end{aligned} \tag{8}$$

The decision function is $\text{sign}(h(x))$, with $h(x)$ defined in (9),

$$h(\mathbf{x}) = \sum_{l=1}^{m} \lambda_l y_l k(\mathbf{x}_{new}, \mathbf{x}_l), \tag{9}$$

where $\mathbf{x}_l$ is the support vector and $\mathbf{x}_{new}$ is the vector to be classified. Various kernels $k(\mathbf{x}_{new}, \mathbf{x}_l)$ can be used in SVM, such as radial basis function (RBF), and polynomial kernels [10]. In this paper the HIK is employed. Given feature vector $\mathbf{x}_{new}$ and support vector $\mathbf{x}_l$, the kernel is defined in (10),

$$k(\mathbf{x}_{new}, \mathbf{x}_l) = \sum_{i=1}^{n} \min(\mathbf{x}_{new}^{(i)}, \mathbf{x}_l^{(i)}), \tag{10}$$

where n is the dimension of feature vector and $\mathbf{x}^{(i)}$ means the i^{th} element of vector $\mathbf{x}$. The HIK kernel shows a good performance in object detection and video retrieval [23, 24].

5. EXPERIMENTS

5.1. Data Set and Evaluation Metric

We evaluate the proposed codebook learning method with a collection of 1873 videos from Columbia University [9]. The data were all consumer videos from Youtube concerning 25 concepts, such as $dancing, wedding$ and so on. The training, development and evaluation sets contain 745, 378, and 750 clips, respectively. The standard 39-dimensional MFCC feature vectors with a 25ms window and 10ms shift were extracted from the sound tracks of these videos.

We use MAP [18] as our evaluation metric which is commonly used in the information retrieval community. For retrieval systems that return a ranked sequence of documents, it is desirable to consider the order in which the returned documents are presented. AP is defined as in (11).

$$AP = \frac{\sum_{r=1}^{R} P(r) \times \text{rel}(r)}{\text{number of relevant documents}}, \tag{11}$$

where r is the rank of the retrieved documents, R is the total number of retrieved documents, $P(r)$ is the precision at cut-off r in the list, and $\mathrm{rel}(r)$ is an indicator function equals to 1 if the item at rank r is a relevant document, and 0 otherwise. MAP for a set of queries (each concept is a query in our task) is the mean of the AP scores for each query.

$$MAP = \frac{\sum_{q=1}^{Q} AP(q)}{Q} \tag{12}$$

There are 25 concepts in our experiment data set. Every one of them is a query when we compute the MAP.

5.2. Experiment Setup and Results

To construct the baselines, we built two level-structrued k-means VQ systems with 1024 (3 levels with 32, 8 and 4 clusters in each level) and 4096 (4 levels with 32, 16, 4 and 2 clusters in each level) codewords denoted by 1024 k-means and 4096 k-means in Table 1, respectively. The BoW vector representation of an audio clip is obtained by counting each codeword's occurrence in that clip.

DLVQ systems were constructed based on the pseudo codeword labels generated by the baseline k-means systems. All DNNs use 7 hidden layers with 2048 nodes in each layer and the input is a splice of the central frame and its 8 context frames. The output softmax layer of each system has the same dimension as the codebook vocabulary of its initializer VQ system, that is, 1024 and 4096, respectively. We built the DNN systems based on Kaldi speech recognition toolkit [25].

The following scheme is used for training the DNN: the parameter initialization is done by using layer by layer generative pretraining [21]. Then the network is discriminatively trained with cross-entropy objective function using backpropagation. The mini-batch size is set to 256 and the initial learning rate is set to 0.008. After each training epoch, we validate the frame accuracy on the development set, if the improvements is less than 0.5%, we shrink the learning rate by the factor of 0.5%. The training process is stopped after the frame accuracy improvement is less than 0.1%.

In actual training procedure, the DNN that based on 1024-codeword k-means achieved frame accuracy 47.94%, 33.10% in training and development set, respectively; the one based on 4096-codeword k-means achieved 39.17% and 24.53%. It can be observed in Fig. 5 that the changing tends of the frame accuracies in training and development set are similar and are mostly increasing. This shows that cross-entropy training indeed makes the DNN mimic its VQ initializer (retaining the "labels" by k-means VQ); but from the final frame accuracies achieved (all below 50%), it could be concluded that the DNN is not learning what exactly its initializer does but capturing new information. After DNN is trained, the BoW representation of an audio clip is then obtained by propagating frames of the clip through the trained DNN and adding up all the output vectors. For both the baseline and proposed systems, the BoW vector representation for each clip was normalized to make the attributes of the vector sum to 1 as a histogram. SVMs with HIK kernel were used as the classifiers.

The experimental results are listed in Table 1. It can be seen that DLVQ gets about 4.5% relative gain over the k-means baseline in MAP. By a simple late fusion of the two results from the baseline and proposed systems, an about 10.5% relative gain can be observed in Table 1 which shows that DLVQ learns some complementary information not fully captured by the k-means system. The simple late fusion scheme is just a weighted sum of the classifier scores of the two systems based on their AP scores on the development set. This promising performance gain shows that DLVQ does help enhance the representative power of VQ based BoW vectors.

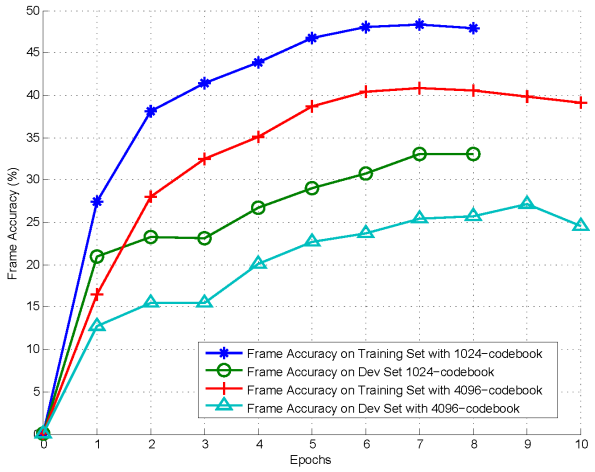

Fig. 5. DNN Training: Frame accuracies on both training and development set with 1024-codebook and 4096-codebook as the function of training epochs

	1024 k-means	4096 k-means
Baseline	0.3851	0.3868
DLVQ	0.4031	0.4039
Late fusion	0.4255	0.4281

Table 1. MAPs of baseline system built using VQ, the proposed system with DLVQ and the fused system

6. CONCLUSION AND FUTURE WORK

In this paper we introduce a discriminative way to perform LVQ using a deep learning framework to learn a good VQ representation from the baseline initializer VQ systems. It can be seen as an good example of the hybrid deep learning architecture [26]. We have demonstrated that the proposed DLVQ system captures new information and gets a quite promising relative performance improvement of 10.5% when fused with its initializer k-means VQ system. This gain, we believe, is benefited from the deep structure of the system which can provide more abstract and useful data representations.

In our efforts to build the baseline VQ systems, inspired by [27], we find that deep autoencoder [21, 28] with very narrow middle bottleneck layer can potentially be a good vector quantizer. We would like to further explore this interesting point and try to integrate it into the DLVQ framework. For DNN training, there are still many points to be improved. We would also like to test DLVQ's performance in other fields, such as computer vision, and investigate the theoretical relationship between DLVQ and its initializers.

7. ACKNOWLEDGMENTS

The authors would like to thank Professor Ji Wu of Tsinghua University for fruitful discussions and Professor Bo Hong of Georgia Institute of Technology and his PhD student Jiadong Wu for helping us set up and utilize GPU in DNN computing.

8. REFERENCES

[1] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 30–42, 2012.

[2] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al., "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1106–1114.

[4] N. Dimitrova, H. J. Zhang, B. Shahraray, I. Sezan, T. Huang, and A. Zakhor, "Applications of video-content analysis and retrieval," *MultiMedia IEEE*, vol. 9, no. 3, pp. 42–55, 2002.

[5] M. S. Lew, N. Sebe, C. Djeraba, and R. Jain, "Content-based multimedia information retrieval: State of the art and challenges," *ACM Trans. Multimedia Computing, Communications, and Applications (TOMCCAP)*, vol. 2, no. 1, pp. 1–19, 2006.

[6] T. Kohonen, "The self-organizing map," *Proc. IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[7] S. Katagiri and C. H. Lee, "A new hybrid algorithm for speech recognition based on HMM segmentation and learning vector quantization," *IEEE Trans. Speech and Audio Processing*, vol. 1, no. 4, pp. 421–430, 1993.

[8] J. Sivic and A. Zisserman, "Efficient visual search of videos cast as text retrieval," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 31, no. 4, pp. 591–606, 2009.

[9] K. Lee and D. P. W. Ellis, "Audio-based semantic concept classification for consumer video," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 18, no. 6, pp. 1406–1416, 2010.

[10] C. C. Chang and C. J. Lin, "Libsvm: a library for support vector machines," *ACM Trans. Intelligent Systems and Technology*, vol. 2, no. 3, pp. 27, 2011.

[11] S. Gao, W. Wu, C. H. Lee, and T. S. Chua, "A MFoM learning approach to robust multiclass multi-label text categorization," in *Proc. International Conference on Machine Learning (ICML)*, 2004, p. 42.

[12] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. the fifth Berkeley symposium on mathematical statistics and probability*. California, USA, 1967, vol. 1, p. 14.

[13] Y. Linde, A. Buzo, and R. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Communications*, vol. 28, no. 1, pp. 84–95, 1980.

[14] C. Zhan, X. Lu, M. Hou, and X. Zhou, "A LVQ-based neural network anti-spam email approach," *ACM SIGOPS Operating Systems Review*, vol. 39, no. 1, pp. 34–39, 2005.

[15] M. Seltzer, D. Yu, and Y. Wang, "An investigation of deep neural networks for noise robust speech recognition," in *Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2013, pp. 7398–7402.

[16] H. Hermansky, D. P. W. Ellis Ellis, and S. Sharma, "Tandem connectionist feature extraction for conventional HMM systems," in *Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2000, vol. 3, pp. 1635–1638.

[17] D. Yu and M. L. Seltzer, "Improved bottleneck features using pretrained deep neural networks.," in *Proc. Interspeech*, 2011, pp. 237–240.

[18] A. Turpin and F. Scholer, "User performance versus precision measures for simple search tasks," in *Proc. Special Interest Group on Information Retrieval (SIGIR)*, 2006, pp. 11–18.

[19] J. Wu and B. Hong, "An efficient k-means algorithm on CUDA," in *IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW)*, 2011, pp. 1740–1749.

[20] Jr. R. M. Gray A. Buzo, A. H. Gray and J. D. Markel, "Speech coding based upon vector quantization," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 28, no. 5, pp. 562–574, 1980.

[21] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[22] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[23] M. J. Swain and D. H. Ballard, "Color indexing," *International journal of computer vision*, vol. 7, no. 1, pp. 11–32, 1991.

[24] B. Byun, I. Kim, S.M. Siniscalchi, and C. H. Lee, "Consumer-level multimedia event detection through unsupervised audio signal modeling," in *Proc. Interspeech*, 2012.

[25] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, et al., "The kaldi speech recognition toolkit," in *Proc. Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2011.

[26] L. Deng and X. Li, "Machine learning paradigms for speech recognition: An overview," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 21, no. 5, pp. 1060–1089, 2013.

[27] D. Yu A. Acero L. Deng, M. Seltzer, A. Mohamed, and G. E. Hinton, "Binary coding of speech spectrograms using a deep auto-encoder.," in *Interspeech*. Citeseer, 2010, pp. 1692–1695.

[28] A. Krizhevsky and G. E. Hinton, "Using very deep autoencoders for content-based image retrieval," in *Proc. European Symposium on Artificial Neural Networks (ESANN)*, 2011.