Product Quantized Deep Embedding for Semantic Image Clustering

Morarjee Kolla [1], Dr. T. VenuGopal [2]
[1]*Research Scholar, Department of CSE,*
JNTUA, Ananthapuramu, Andhra Pradesh, India.
morarjeek@gmail.com

[2]*Professor, Department of CSE, JNTUH College of Engineering Jagityal,*
Nachupally, Telangana, India.
t_vgopal@rediffmail.com

***Abstract*—** With the increase in the availability of large data involving image data spikes the need of efficient retrieval of relevant content at a low computational cost which imposes a challenge to serve the user with relevant requested information. Semantic Image Clustering counters this problem through unsupervised categorization of meaningful images. The available deep clustering approaches using autoencoders have failed to capture the desired clustering in memory restricted environments. To tackle this, we proposed Deep Embedded Clustering methodology by quantizing deep features using Product Quantized codes. The proposed paradigm enables to enhance the computational speed in contrast to available competitive methods. Our experimental outcomes on MNIST-Full, MNIST-Test, STL-10 datasets prove the efficiency and scalability of the proposed methodology.

Keywords*—Semantic Image Clustering; Deep Embedded Clustering, Product Quantized Codes.*

I. INTRODUCTION

Content Based Image Retrieval (CBIR) fails to fetch relevant content requested by the user from a large volume of image databases. The deviation between low level depiction and high-level understanding describes the semantic gap. Reduction of the semantic gap functions as the prime objective to collide with user perceptions [1]. Current search techniques based on key word tagging are impotent to meet human desirability. Since keyword tagging methodology involves search relying on text input processed as a query. Existing research on CBIR attempts to project consequential images and formulating semantic clusters. Efficient storage and rapid content retrieval ventures as the secondary objective of the CBIR. It attempts to grab homogeneous images by analyzing the image content. Therefore, the image representation and uniform measures are preliminary for these tasks. In CBIR, image representation and computational cost take the driver seat. With the surge in accessibility of large volumes of data, efficient searching methodologies for retrieval of relevant content demands core attention.

Clustering is an Unsupervised content analysis and conceptualization methodology for categorizing identical images. A major slice of the existing clustering methodologies pivot on the distance metric to compare the uniformity of the points. A few methods also feature with embedded space. Analyzing similarity with low level representations utilizing various classification of distance techniques procure in capabilities like suitable distance metric prediction, efficient cluster formation and validation. These approaches fail to justify human expectation due to lack of sensible clusters. Mapping raw pixel values to meaningful concepts is a field of primary research in reducing the semantic gap. Various methods are addressed in literature to minimize this understanding gap [1].

Semantic Image Clustering (SIC) is the concept of categorizing unlabeled images based on their high-level domain definitions. It shortens the search space and the semantic gap to return relevant results from the user query. Searching for homogeneous images rely on the subjective matter and is task dependent. SIC require to embed artificial intelligence to nullify the semantic gap. Relevance feedback is the most prominent technique to minimize the relevance of the result as per the requested query [1]. Despite the approach RF suffers a significant loss in high level semantic extraction and real time processing. Other media of retrieval utilize Ontology to extract semantics which process on the label of the image [2]. Ontology declines its productivity with miscellaneous terms and inaccurate label of images.

The recent surge in the employing of deep learning techniques primarily exercises clustering in two ways. The first way, extracts deep features and perform orthodox clustering on it [3,4]. The second way depicting user data in the form of cluster compatible latent embedded space using autoencoders [5]. Deep architectures were proposed for hash learning in an unsupervised manner by using autoencoders to learn the representations. Compact binary codes are quickly compared using hashing or hamming distance. Deep unsupervised hash methods utilize quadratic

constraints to minimize the difference between compact representation and target binary codes. In the deep quantized auto encoder, encoder for learning compact representation and decoder is for manifold preservation.

Usage of an approximated nearest neighbour or hash-based methods is greater in demand to speed-up retrieval process replacing the linear search. This method depicts the high-dimensional features into low-dimensional space and followed by generating compact binary codes. By using fast binary matching and hamming distance, reduces the computational cost. Deep unsupervised hashing (DUH) methods can learn non-linear transformations for converting images into binary code in an unsupervised manner. A major slice of current DUH methods fail to minimize the quantization error statistically and feature representations are not optimally compatible with binary hash coding.

Some researches on clustering methodologies have emphasized on retaining the similarity of binary codes in the projection of large image repositories into a single machine [6,7]. Using binary codes as image descriptors enable to minimize the storage space and enhance the computational speed for identical search [8]. Understanding the binary codes aims to generate short codes to accommodate a large number of images in memory. High level operations tend to be tedious for computation resulting in increased search time. Since binary operations are hardware friendly they speed up the computation. Our proposed product codes will offer betterment over the binary codes.

II. RELATED WORK

An enormous number of researches were conducted on SIC using Relevance Feedback, Ontology and deep learning. A few notable efforts have been enlisted below.

Some researchers designed SICbased on RF methodologies as a Vector model, Classification problem and learning problem [9] faces challenges to respond to feedback data from the user. A few contributors used Ontology based paradigms[2], which suffer from coin terms and word power discrepancies. Conventional clustering techniques are diverse computationally intensive for huge datasets. K-means is more robust for a large-scale clustering. Although, minimizing the storage of a big amount of data computation cost is the cumbersome activity.

Deep learning paradigms are efficient for clustering through autoencoders [10]. Deep Embedded Clustering (DEC) [5] uses clustering loss to tune the parameters and cluster centres simultaneously. This demands a higher number of parameters and storage space. To counter the same, we are proposing a novel approach in auto encoder through an unsupervised hashing technique. Several hashing functions [11,12,13,14,15,16] are addressed in the literature to reserve the identity of the skeleton structure. Independence and bit balance are not incorporated in the followingparadigms. Their motive is to generate continuous hash functions to map each entity into a binary feature vector. Therefore, similar identical entities are mapped over identical binary codes. Data dependent hash methods of spectral hashing [11], ITQ, K-means hashing are failed to capture the nonlinear manifold structure of samples. ITQ to incorporate independence and bit balance [17]. ITQ stands out from other competitive methods by reducing quantization errors. ITQ process PCA as a data reduction technique for generating continuous codes and then rotates for binary closed codes [18]. ITQ maximizes the variance of each binary bit and minimizes binarization loss. K-Means Hashing [13] minimizes the hamming distance between quantized cells and cluster centres.

It utilizes a sub optional method through relaxation of binary constraints at the time of optimization. Binary Autoencoder (BA) [18] produces binary codes, by using step function. Clustering over the generated hash codes will improvise the clustering speed [19]. Product quantization [20] deems to counter vector quantization problems when the exponentially large number of code words is desired to accurately reconstruct the input vectors. The prime objective is to decompose the original vector into a cartesian product of low dimensional sub spaces and quantize each sub space into code words. Deepquan[21] learns binary codes by minimizing quantization error through product quantization. Weighted triplet loss is predicted to avoid trival solutions and poor generalization. It consists of multiple convolutional and pooling layers to capture image representations. Fully connected bottle neck layers for dimensionality reduction, pairwise cosine law similarity preserve learning. PQ laws for controlling hash quality and quantizability of bottleneck representations.

Our previous approach lies in symmetry with the second approach in memory efficient aspect using binary quantization [28]. It features two step dimensionality reduction and binarization namely for optimized computations and efficient storage. We considered Product quantization replacing binary quantization in this paper to spike the performance of the clustering and enhance computational speed and allows efficient memory storage.

The main contributions of our approach are

- Deep auto-encoder with product quantization nurtures demands reduced memory space in comparison to binary quantization and extracts discriminative features in embedded space.
- End to End joint learning approach, parallelly learns image feature representations and cluster friendly product codes efficiently.
- Huge quantities of images can also be stored in a single machine with product quantization.
- Product quantized clustering will improve the speed of the clustering than binary quantization.
- In contrast with available competitive techniques, the performance in terms of memory consumption and clustering speed will be improved.

III. PRODUCT QUANTIZED DEEP EMBEDDED CLUSTERING (PQDEC)

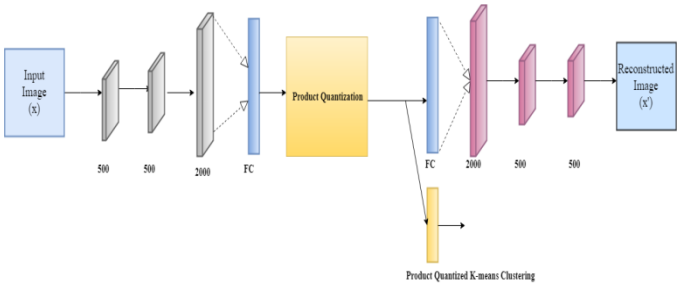


Fig.1. The structure of proposed Product Quantized Deep Embedded Clustering

We modified the DEC by introducing Binary Quantization to the last Fully connected flatten layer of encoder part of Stacked auto-encoder as shown in Fig. 1. In DEC, more number of parameters are reshaped to embedded layer, which will occupy a more amount of memory space. To avoid this, we apply memory efficient binary quantization on these features before reshaping them into embedded layer. This will improve the scaling factor of clustering. Consider a dataset x (MNIST-Full or MNIST-Test or STL-10) with fully connected stacked auto-encoder inspired by van der Maaten [23]. Embedded layer with only 10-dimensional feature space as a number of clusters. We keep the decoder as it is, like in DEC to learn up sampling.

A. Deep Embedded Clustering

In DEC [5], cluster friendly parameters are initialized and then optimized using clustering. Consider the dataset x of n points into k number of clusters with the centroid $\boldsymbol{\mu_j}$, where j=1to k. we transform a high dimensional data space X into a low dimensional non-linear feature space Z [24]. This feature space is called an embedded space with k number of dimensions. Clustering is performed on embedded space. Encoder output is passed to binary quantization layer as shown in Fig. 1. We inspired by DEC [5] for effective clustering and uses a similar framework for our proposed system.

B. Product Quantized Deep Autoencoder

We proposed an efficient and memory restricted embedded space in our autoencoder to speed up the clustering by using product quantization(PQ) [22].

Consider high dimensional vector of x with D dimension of an original space $\mathbf{x} \in \mathrm{R}^{\mathrm{D}}$is mapped to M disjoint sub vectors with L code words in the product space $\mathbf{z} \in \mathrm{R}^{\mathrm{M}}$using hash function h is represented by

$$z = h(x) \in \{1 \dots . L\}^M \quad (1)$$

Where $\boldsymbol{L}$ is the pretrained code word, M is sub space and $\boldsymbol{h}(.)$is hash function

Encoder $\boldsymbol{f_W}(.)$ maps real data points into binary code vector with c bits and Decoder $\boldsymbol{g_U}(.)$ maps binary codes back to real data points is represented by

$$\boldsymbol{f_W(x) = \sigma(Wx) \equiv z}$$
$$g_U(z) = \sigma(Uz) \quad (2)$$

Where W, U are weights of encoder, decoder, and $\boldsymbol{\sigma}$ is an element wise step function.

Our PQ based final hash function of encoder to map continuous inputs into product codes with Product Quantized encoder ($\boldsymbol{P}$) [22] is defined as

$$P = h(x) \in \{1 \dots . L\}^M \quad (3)$$

Locality structure of the data will be preserved by our optimization function to minimize the quantization loss ($\boldsymbol{L_q}$) [21] which is defined as

$$L_q = \sum_{i=1}^{N} \| Z_i^l \dot{-} Ch_i \|_2^2 \quad (4)$$

Where $Z_i^l$are the bottleneck representations, C is Code book and h_i is hash code.

Our desired hash function has to minimize the Re-construction loss (L_r) is defined as

$$L_r = \sum_{i=1}^{n} \| x_i - g_U(f_W(x)) \|_2^2 \quad (5)$$

Where x_i input image and *n* is the number of images in the dataset.

C. Fast clustering using Product quantization

Product quantized codes of the dataset images are clustered using Yusuke Matsui [22] approach Minimizing the Euclidean distance between product code points and product quantized cluster center is the objective function of our approach for all dataset images using K-means clustering is represented by

$$\min_{\mu_j} \sum_{j=1}^{k} \|x_i - \mu_j\|_2^2 \quad (6)$$

Where $\mathbf{x_i}$ is the PQ code data points, and μ_j is the PQ code center of j^{th} cluster.

Nearest Product centers of all Product quantized data points will predict using multi-index hash table in constant time [25]. Fast hashing operations on both PQ data points and centers will improve the speed of the clustering compared to the traditional K-means method. The main difference between traditional K-means and Product K-means with PQ is addition of Product code constraint to the cluster center.

In the assignment step, data points are assigned to their nearest cluster centers. We are optimizing the cluster centers with respect to product constraints for each cluster. In the update step, cluster centers and weights of autoencoders are updated.

The entire PQDEC algorithm is summarized below

Algorithm: PQDEC algorithm

Input:

Input data: X; Number of dataset samples: n; Number of clusters: K; Learning Rate λ= 0.01; Mini batch m; Stopping threshold δ=0.1%;

Output:

Autoencoder weights: W, U; Cluster centres: μ_j; Product Quantized embedded space: Z; Quantization Loss: L_q; Reconstruction Loss:L_r;

Algorithm:

Initialize μ_j, W, U

Compute Encoder and Decoder functions of Autoencoder using (2)

Compute embedded Product Quantized embedded space (P) using (3)

Compute all embedded binary points $Z_i = \{ f_W (x_i)\}_{i=1}^n$

Compute Loss functions L_q,L_r using (4), (5)

Objective Loss function minimizes $L = L_r + \gamma L_q$

Where $\gamma > 0$ is a coefficient that controls degree of distortion.

Compute Clustering Objective function using (6)

while not Converged **do**

Update μ_j using

$$\mu_j = \mu_j - \frac{\lambda}{m}\sum_{i=1}^{m}\frac{\partial L_q}{\partial \mu_j}$$

Update W,U using

$$W = W - \frac{\lambda}{m}\sum_{i=1}^{m}\frac{\partial L_r}{\partial W} + \gamma \frac{\partial L_q}{\partial W}$$

$$U = U - \frac{\lambda}{m}\sum_{i=1}^{m}\frac{\partial L_r}{\partial U}$$

Update $\{Z_i\}_{i=1}^n$

Return μ_j, W, U

IV. EXPERIMENT *RESULTS*

A. *Datasets*

We conduct experiments on three standard datasets

- **MNIST-Full:** The MNIST dataset [26] consists of70,000 hand written digit images with digit centred and size normalized.
- **MNIST-Test:** The test set of MNIST consist 10,000 images with normalized size and digit centred.
- **STL-10:** The STL-10 [27] consists of 13000 unlabelled color images with size of 96 X 96 of 10 categories.

B. *Experimental setup*

Experiments are conducted in GPU based system with NVIDIA GPU memory of 16GB. Our implementation is based on Python, TensorFlow and Keras.

1) Comparison Methods

We showed the efficacy of our system by comparing our algorithm with four competitive algorithms K-means, K-medoids, KDK-means and DEC.

2) Parameter Setting

DEC algorithm dimensions for encoder is d-500-500-2000-10 and decoder 10-2000-500-500-d for all data sets. We fixed a mini batch size of 256, learning rate 0.01, dropout 20%, convergence threshold 0.1% and an update interval of 140 for DEC. BQDEC is also a similar parameter setting as DEC. Only embedded layer is binary quantized. We take 256-bit code length for our binary code vector. Similar parameter setting for our PQDEC with product code length of 32 bits of 4 parts with each 8 bits. GitHub codes are collected for K-means, K-medoids, KDK-means, DEC for our experiments.

3) Evaluation Metric

We take clustering speed with respect to 10 number of epochs in seconds and memory consumption in Megabytes to prove the scaling factor of our proposed memory efficient clustering.

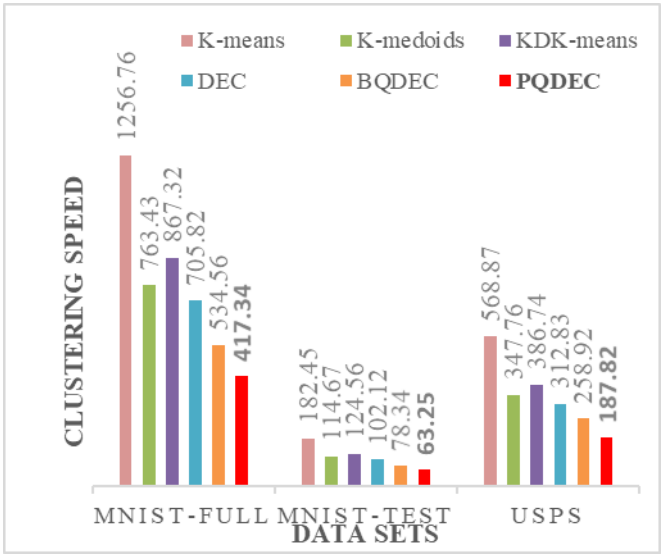
C. *Results*

Clustering speed and memory consumption of three datasets with comparative methods given in Table 1. Fig.2**.** shows Top 10 results of MNIST-Full dataset. Fig. 3. Shows the Clustering Speed Vs Clustering Methods graphs. Fig. 4. Shows Memory Consumption Vs Clustering Methods graphs. These results shows efficacy of our proposed PQDEC method in terms of computational speed and memory consumption.

Table 1. Performance Comparison of clustering algorithms on three databases.

Datasets	**MNIST-Full**		**MNIST-Test**		**STL-10**	
Criterion for 10 Epochs	Clustering speed (in secs)	Memory Consumption (in MB)	Clustering speed (in secs)	Memory Consumption (in MB)	Clustering speed (in sec)	Memory Consumption (in MB)
K-means	1256.76	101.3	182.45	14.3	568.87	43.6
K-medoids	763.43	67.2	114.67	9.5	347.76	29.7
KDK-means	867.32	69.3	124.56	9.8	386.74	29.3
DEC	705.82	62.6	102.12	8.7	312.83	27.3
BQDEC	534.56	52.5	78.34	7.4	258.92	23.4
PQDEC	**417.34**	**42.6**	**63.25**	**6.5**	**187.82**	**21.6**

Fig.2. Top 10 results of MNIST-Full dataset.



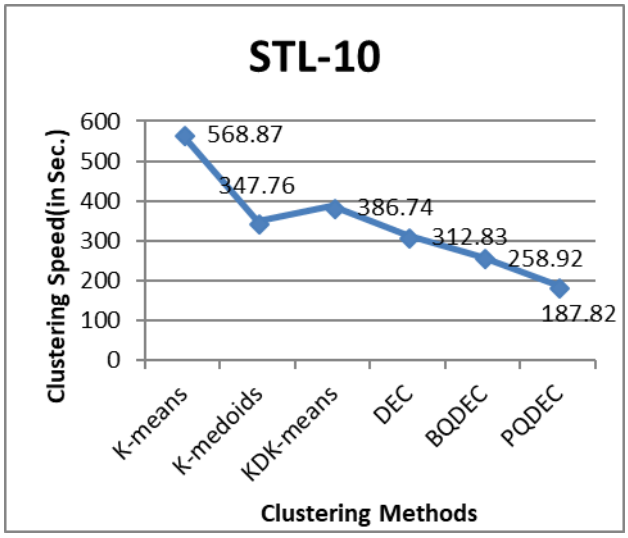
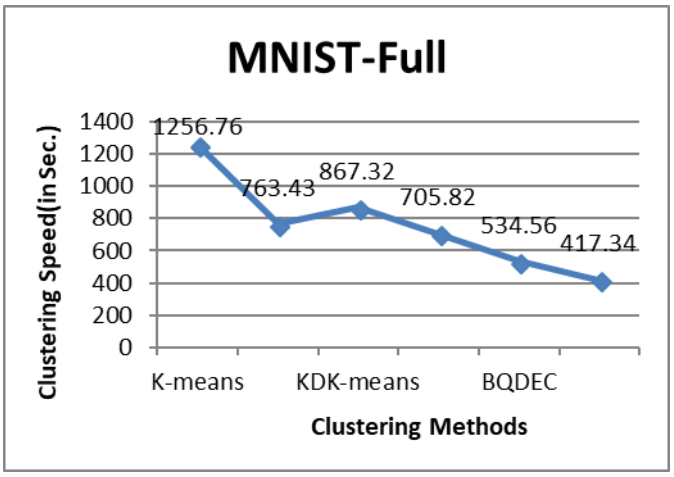
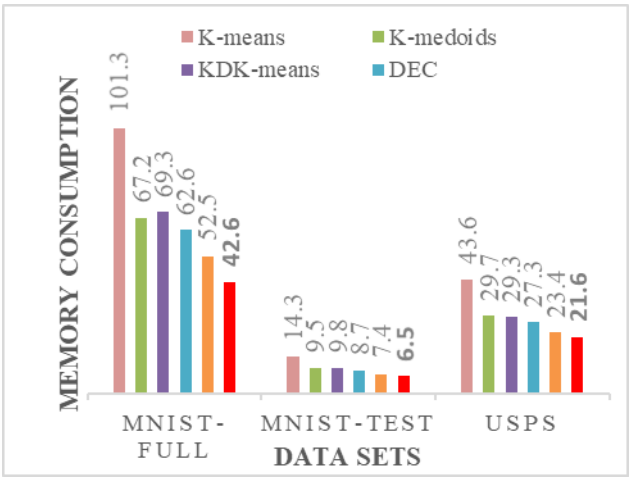
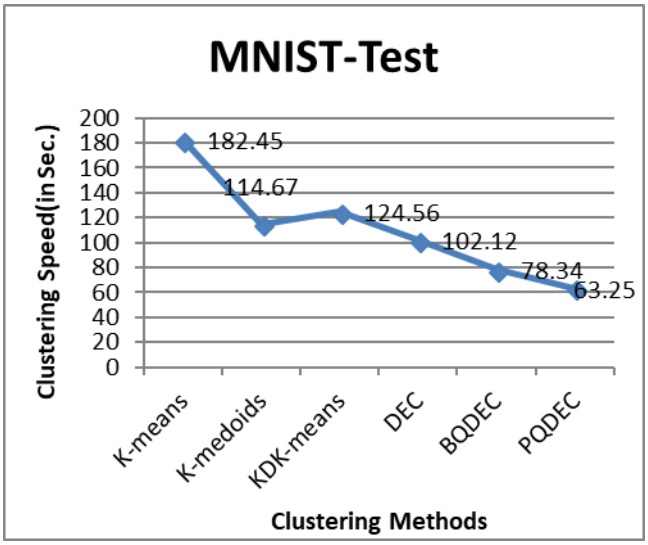
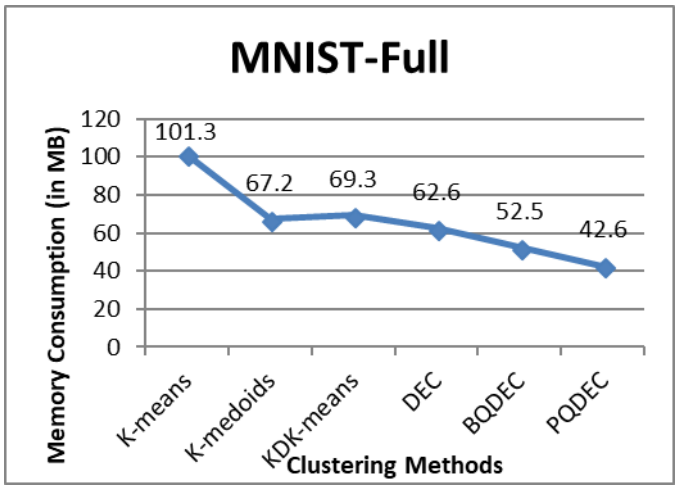


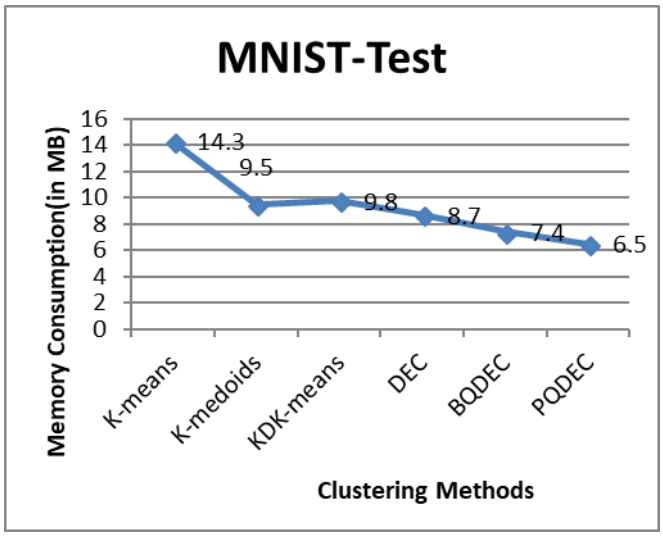
Fig.3. Clustering Speed Vs Clustering Methods











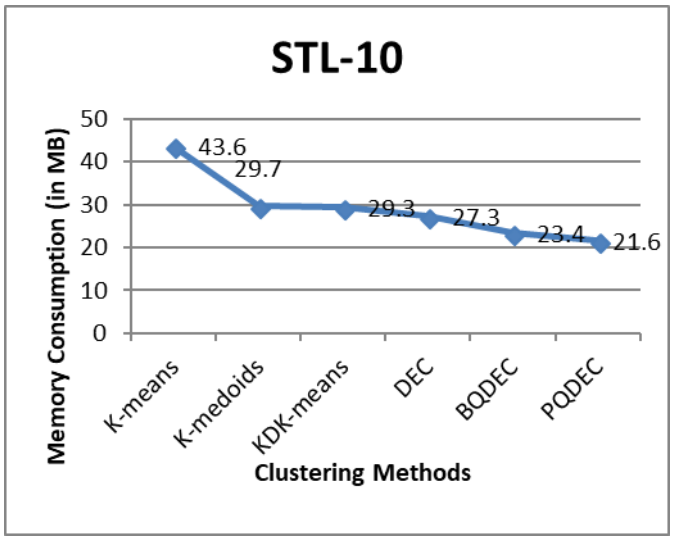


Fig.4. Memory Consumption Vs Clustering Methods

V. CONCLUSION

This paper proposes Product Quantized Semantic Deep Embedded Clustering (PQDEC), which performs memory efficient clustering in reduced running time. This framework minimizes the memory space by quantizing the deep features and speed up the execution process using product hash codes. Our method excels in memory restricted environments with massive images. Experiments conducted on MNIST-Full, MNIST-Test, STL-10 datasets proved the improvement in the efficiency of our method without effecting the cluster accuracy. Compare to our previous method of BQDEC proposed PQDEC method improves clustering speed and reduced memory consumption. In future, experiments will be conducted on billion scale datasets for various real time application with advanced memory efficient fast clustering approaches.

REFERENCES

[1]Mohammed LamineKherfi ,DjemelZiou, "Relevance Feedback for CBIR: A New Approach Based on Probabilistic Feature Weighting with Positive and Negative Examples", IEEE Transactions on Image Processing, Vol.. 15, No. 4, pp.1017-1030, April 2006.

[2]MorarjeeKolla, T .Venu Gopal, Region based Semantic Image Retrieval using Ontology,pp.421-428,Vol. 5, LNNS Springer, 2017.

[3]Yunchao Gong, Marcin Pawlowski, Fei Yang, Louis Brandy, LubomirBourdev, and Rob Fergus. 2015. Web Scale Photo Hash Clustering on A Single Machine. In Proc. IEEE CVPR.

[4]James Philbin, Ondřej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman.2007. Object Retrieval with Large Vocabularies and Fast Spatial Matching. In Proc. IEEE CVPR.

[5]Junyuanie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In International Conference on Machine Learning (ICML), 2016.

[6]M. Raginsky and S. Lazebnik. Locality sensitive binary codes from shift-invariant kernels. NIPS, 2009.

[7]A. Torralba, R. Fergus, and Y. Weiss. Small codes and large image databases for recognition. CVPR, 2008.

[8]H. J´egou, M. Douze, C. Schmid, and P. Perez. Aggregating local descriptors into a compact image representation. CVPR, 2010.

[9]Ryota Hinami, Yusuke Matsui, Shin'ichi Satoh, "Region-Based Image Retrieval Revisited", DOI: 10.1145/3123266.3123312,ACM, 2017.

[10]Vincent, Pascal, Larochelle, Hugo, Lajoie, Isabelle, Bengio,Yoshua, and Manzagol, Pierre-Antoine. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. JMLR, 2010.

[11]Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in NIPS, 2008.

[12]Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. 2013. Iterative Quantization: A Procrustean Approach to Learning Binary Codes for Large-scale Image Retrieval. IEEE TPAMI 35, 12 (2013), 2916ś2929.

[13]K. He, F. Wen, and J. Sun, "K-means hashing: An affinity-preserving quantization method for learning binary compact codes," in CVPR, 2013.

[14]J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-e. Yoon, "Spherical hashing," in CVPR, 2012.R.

[15]Salakhutdinov and G. E. Hinton, "Semantic hashing," Int. J. Approx. Reasoning, pp. 969–978, 2009.

[16]J. Lu, V. E. Liong, and J. Zhou, "Deep hashing for scalable image search," TIP, 2017.

[17]Thanh-Toan Do, Dang-Khoa Le Tan, Tuan Hoang, Ngai-Man Cheung: Compact Hash Code Learning with Binary Deep Neural Network,arXiv:1712.02956v2,2018.

[18]M. A. Carreira-Perpinan and R. Raziperchikolaei, "Hashing with binary autoencoders," in CVPR, 2015.

[19]ShengcaiKe, Yongwei Zhao, Bicheng Li1, Zhibing Wu and Xin Liu:Fast Image Clustering Based on Convolutional Neural Network and Binary K-means,Proc. of SPIE Vol. 10033,2016.

[20] Herve Jegou, Matthijis Douze, and Cordelia Schmid. 2011. Product Quantization for Nearest Neighbor Search. IEEE TPAMI 33, 1 (2011), 117ś128.

[21] Yue Cao, Mingsheng Long, JianminWang, Han Zhu and QingfuWen. Deep Quantization Network for Efficient Image Retrieval, AAAI Conference on Artificial Intelligence, 2016.

[22] Yusuke Matsui et. Al. PQk-means: Billion-scale Clustering for Product-quantized Codes.ACM.2017.

[23]Van der Maaten, Laurens. Learning a parametric embedding by preserving local structure. In International Conference on Artificial Intelligence and Statistics, 2009.

[24]Bellman, R. Adaptive Control Processes: A Guided Tour. Princeton University Press, Princeton, New Jersey, 1961.

[25] Yusuke Matsui, Toshihiko Yamasaki, and Kiyoharu Aizawa. 2015. PQTable: FastExact Asymmetric Distance Neighbor Search for Product Quantization using Hash Tables. In Proc. IEEE ICCV

[26]LeCun, Y., Bottou, L., Bengio, Y., Ha_ner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11), 2278{2324 (1998).

[27]Coates, Adam, Ng, Andrew Y, and Lee, Honglak. An analysis of single-layer networks in unsupervised feature learning. In