Efficient image retrieval using multi neural hash codes and bloom filters

Sourin Chakrabarti
IIIT Allahabad, India
sourin.chakrabarti@gmail.com

Abstract—**This paper aims to deliver an efficient and modified approach for image retrieval using multiple neural hash codes and limiting the number of queries using bloom filters by identifying false positives beforehand. Traditional approaches involving neural networks for image retrieval tasks tend to use higher layers for feature extraction. But it has been seen that the activations of lower layers have proven to be more effective in a number of scenarios. In our approach, we have leveraged the use of local deep convolutional neural networks which combines the powers of both the features of lower and higher layers for creating feature maps which are then compressed using PCA and fed to a bloom filter after binary sequencing using a modified multi k-means approach. The feature maps obtained are further used in the image retrieval process in a hierarchical coarse-to-fine manner by first comparing the images in the higher layers for semantically similar images and then gradually moving towards the lower layers searching for structural similarities. While searching, the neural hashes for the query image are again calculated and queried in the bloom filter which tells us whether the query image is absent in the set or maybe present. If the bloom filter doesn't necessarily rule out the query, then it goes into the image retrieval process. This approach can be particularly helpful in cases where the image store is distributed since the approach supports parallel querying.**

Keywords—Neural hash codes, Bloom filters, Convolutional neural networks.

I. INTRODUCTION

Convolutional neural networks(CNN) [21] have proven to be very useful in image classification and identification tasks. Apart from these, an active research topic in recent times has been image retrieval. Often it is seen that the search space is quite big and it takes lots of time for an exhaustive search through the complete database. Hence, lots of recent studies have also been focused on reducing the number of queries to the database using various data structures that have minimal knowledge of the content of the images but enough for them to be useful for a pre-query optimization along with the image retrieval task.

The approach followed in this paper mainly revolves around the success of CNNs in extracting features from images [22]. The final output of the CNN just before the layer for classification contains the final feature map required for classification. This property has been often leveraged by various image descriptors to produce a compressed representation of the images. But interestingly enough, on comparing the performance of the values of the intermediate layers with the final layer for retrieval, it was seen that often the best results were obtained out of the intermediate layers [2]. Our approach makes use of all the information available from the image by using neural hash codes from both the lower and higher layers.

It is seen that neural codes have performed particularly well when it comes to image retrieval tasks [2], [13]. Using CNN image descriptors along with bloom filters have been tried before by [1] to increase search efficiency. We propose a unique approach of using neural codes from multiple layers simultaneously for image retrieval tasks and pre-query filtering tasks. For the generation of the neural codes, dimensionality reduction was another area that needed consideration. Since PCA compression provides a good degradation to performance trade-off [2], We use it to compress the layer outputs. The final number of dimensions was fixed to 128 since it provides a negligible loss of accuracy. The compressed feature vectors were converted to binary sequences using a modification of the multi k-means approach to accommodate multiple feature vectors and then fed to the bloom filter to limit queries. For image retrieval, the feature vectors of the higher layers helped in similarity matching in the coarse stage whereas lower layers gave more insights about the fine structural details of the image and hence were used for further retrieval

Finally, experiments on the Oxford 5k [18], INRIA holidays [20], and Paris 6k [19] data sets were carried out to report the precision of our proposed method and the time consumed for the queries. We also show the effect on precision caused by varying the number of layers used for generating neural codes for various data sets along with varying the size of the bloom filter.

II. RELATED WORK

Deep learning-based image descriptors have been researched for long. SIFT-based descriptors [17] were widely used for long for image description tasks before the inception of other more robust and efficient descriptors such as VLAD [16] and Fischer descriptors [24]. Moreover, recently triangulation embedding along with democratic aggregation has been shown to outperform Fischer vectors [12]. CNN based image descriptors have been known to outperform the above descriptors on numerous occasions and have proven to act as a baseline in image recognition and retrieval tasks [13]. Often CNNs are fine-tuned to work for image retrieval tasks.[7] proposed a fully automated procedure for fine-tuning a CNN for image retrieval. [2] uses lower layers of CNN for image description tasks and compares it with higher layers showing that often the intermediate layers contain more valuable information required for image description. Capturing local CNN features and compression using VLAD has been experimented by [4].

Accumulation of these deep features has been discussed in [6]. On the other hand, [11] proposed extraction of image features through capturing them from various parts of the image through the construction of windows(more like an R-CNN) and finally compressing using Fischer vectors or PCA compression. These techniques also find heavy usage inframe retrieval from large videos as depicted by [9].

Output produced by CNNs or any image descriptor is generally of larger dimensions than required. These larger dimensions need to be reduced in order for the image retrieval systems to be efficient. This reduction can be achieved by adding hidden layers before the final classification layers in case of CNNs. Some of the most prominent techniques employed in this field are shown in [25]. [26] suggested using the image features and the ground truth texts to devise a similarity graph and hence learn relations among the elements of the data set. In [27], neural codes from a CNN were generated by first pre-training on an Image-net following which another layer was added to the network which generated the required hashes, and finally retrieval was carried out using hierarchical search using both the hash codes and the features of certain layers. [10] used multi-label images to devise a new hashing approach. They used CNNs for finding features and then used a fully connected layer along with a loss function based on the multi-label supervision to generate the hashes and learn the features at the same time. [1] also used CNN descriptors with hashing and indexing along with bloom filters on sharded databases to reduce the number of queries for image retrieval.

Image matching algorithms are quite important in an image retrieval process. Traditionally, image matching problems were reframed as graph optimization problems. Several graph based matching algorithms based on Markov Random fields were suggested by [28], [29]. Using a coarse to fine approach to establish dense pixel level correspondences using randomized search was suggested by [31]. WarpNet [30] proposed architecture which could match objects in various images.[3] suggested using a deep learning based hierarchical image matching architecture which proved to be quite suitable for our cause.

The features obtained were typically hashed and stored in inverted file systems. [35] introduced the inverted multi-index which replaces standard quantization with product quantization. [5] further improved upon this approach by random initialization a k-means dictionary and storing the results entries.

Bloom filters since their inception [15] have proved to be quite effective in identifying true negatives in searching operations. But their application in fields of image processing has been limited. [14] proposed Bloomier filters for storing features of an image data set which made them more memory efficient than storing them in hashes. Image descriptors have found themselves together with bloom filters on numerous occasions such as in [1] where bloom filters were used to construct a feature descriptor which used hash functions that differentiate between inputs of different categories. Appli-cation such as image retrieval from videos which include memory intensive tasks have employed the use of bloom filters along with novel image descriptors to improve efficiency [9]. Bloom filters have also performed well as image descriptors in the past [32]. Recent advances in fields of bloom filters have suggested that bloom filters can be modified to work with heavy throughput of data with acceptable accuracy like the case with neural bloom filters [8].

III. PROPOSED METHODOLOGY

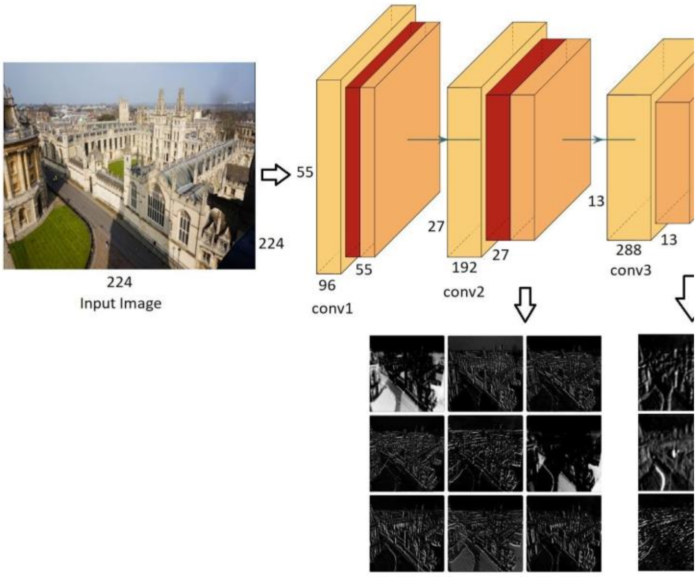
Our proposed methodology for efficient image retrieval primarily revolves around training the model used, identifying the lower layers to be used for feature extraction, feature compression, binary sequencing, insertion in the bloom filter and database and then finally image retrieval.

A. CNN architecture

In our setup, we leverage the model used by [2] to successfully extract neural codes from lower layers. We use our own modification of the system to include neural codes from multiple layers in one feed forward cycle. The model consists of 5 primary units. A unit consists of a convolutional layer followed by a pooling layer which employs max pooling and then finally an activation layer using the non-linear ReLu transform. These 5 units are followed by 3 fully connected layers at the end of which the output is received in one hot encoding. The architecture is shown in Figure 1. While Training the model softmax loss function was used. The model was pretrained on the Image-Net [34] classes and fine-tuned by training on the relevant datasets for our use case.

For benchmarking, images from the following datasets are used for training and testing purposes:

1) Oxford 5k: The Oxford buildings dataset consists of a collection of 5062 images spanning over the colleges of Oxford. The ground truth text for each one refers to one of the 11 landmark buildings. For each image, labels were associated with them which gave information about the quality of the image. About 55 query images are present with each one belonging to either of the 11 landmarks.
2) Paris 6k: The Paris dataset consists of a collection of 6412 images spanning over the landmarks of Paris. The ground truth text for each one refers to one of the 12 landmark buildings.
3) INRIA holidays: This dataset includes images from holiday destinations containing a wide array of images and corresponding classes. It consists of 991 training and 500 query images spanning across 500 classes. Certain images were rotated but our model being rotation invariant, it didn't make a difference.



1.

2. Figure 1. The CNN architecture used for training. Yellow slabs correspond to convolution layers, red slabs correspond to the max pooling, and the orange slabs correspond to the ReLu layers. Layers fc6, fc7, and fc8 are fully connected to the preceding layers. The layers used for neural codes are L1, L2 and L3. The activations of intermediate layers shown are randomly chosen from all the available filters for the layer.

3.

The images were pre processed before feeding them to the neural network. All the images were resized to 224x224 to be provided as input. For training, the strides used were 4 for all the layers except the first and 1 for the first layer. The model was pre trained on Image-Net classes but was again trained using the datasets mentioned above to suit our use-case. Once the model was trained, randomly selected training images were passed through the neural network as features from layers L1, L2 and L3 were collected to be introduced into the bloom filter. Once the feature maps were collected, PCA compression was applied on the maps to compress the feature vector to a size of 128. This length was decided upon due to the size of our test dataset.

B. Maintaining the Integrity of the Specifications

Our approach uses a modified version of the proposed by[5]. They randomly initialized a k-means dictionary with some feature vectors from the training dataset. Now, whenever an image is introduced, its similarity is compared with each of the current centroids. This computation is done by considering the L2 distances of all the pairs of feature vectors in our case.Once the L2 distances are computed, they are sorted and distances with centroids below a certain threshold are considered. Once an image is included in a cluster, the corresponding bitis marked as 1 in the generated binary sequence for that image and the rest are marked 0. A feature can be assigned to more than one centroids.

C. Hashing and Bloom filters

The binary sequenced codes obtained from the three layers were hashed via a hash function. The choice of hash function for a bloom filter should confirm uniformity and should be logically independent. Over the years various hash functions have found applications to bloom filters, but the primary ones used along with binary signatures include the Murmur3 [33], cryptographic hashes like SHA-256, LCGs etc.

A bloom filter essentially is a data structure that probabilistically determines the presence of an element in a set. False positive outputs are deterministically given by a bloom filter. In traditional bloom filters, k hash functions are used to compute k hashes of an item to be inserted into the structure. Each of the k hashes return a position to be marked in the filter. During retrieval, the query element is again hashed against all the hash functions and the returned positions are matched from the filter. If any one of the positions is unset, it is inferred that the element is definitely not present in the original data set. Whereas if all the positions are marked, then we can't conclude anything deterministically.

The probability of presence of a false positive element in a bloom filter is given by the equation:

$$P = (1 - (1 - 1/m)^{nk})^{k} \quad (1)$$

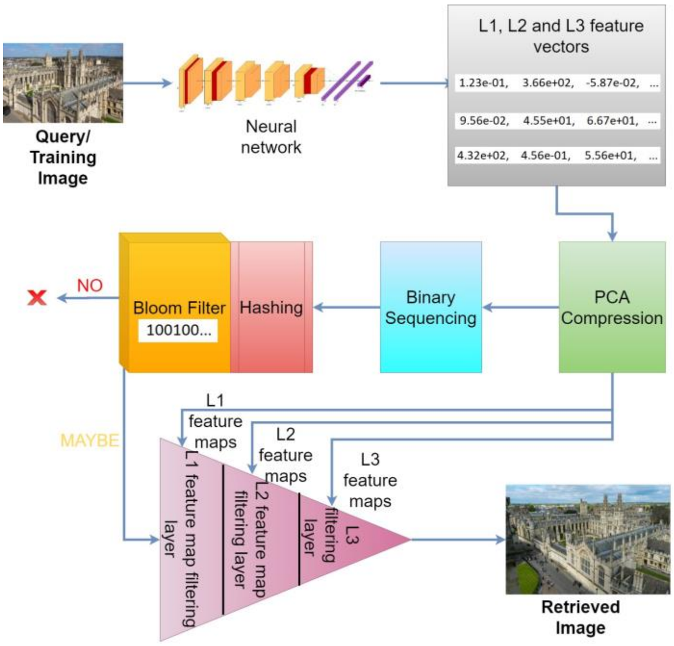
where n is the size of the data set, m is the size of the bloom filter and k is the number of hash functions used [15]. We can obtain optimal values of m for a fixed k and n by the equation:

$$m = kn\,ln(2) \qquad (2)$$

In our case, we have k ranging from 1 to 3. Hence, the corresponding bloom filter sizes would be ideally from 1.5n to 4.5n approximately. To facilitate our bloom filter, we have used Murmur3 hash function to compute hashes from the binary sequences from the layers L1, L2 and L3. Once the hashes are computed, they are inserted into the bloom filters of the fixed size. Once this process is completed, the query images are used for testing.

D. *Storage*

The compressed feature vectors were stored in a hierarchical manner with the L3 layer at the top followed by the L2 and the L1 layer. This model allowed more layers to be used in the matching process in later developments. The average cosine distance between images of the same class were computed during training to be used as a threshold during the retrieval process.



4. *Figure 2: Complete architecture: The query/training image is fed to the neural network for neural code generation which are PCA compressed. These compressed vectors are sent to the respective storage layers for retrieval. The compressed vectors are then binary sequenced and hashed before feeding them to the bloom filter while training or searching them in the bloom filter while querying. Depending on the output of the bloom filter, the query proceeds. For retrieval, PCA compressed vector from the query is first compared with the L1 feature maps followed by L2 and L3 before providing a final output.*

E. *Retrieval process*

For the limiting process, the compressed feature vectors are again extracted from the CNN layers by passing the query image through the neural network. The compressed feature vectors are quantized and passed to the bloom filter post hashing. If atleast one bit is unset, then the image retrieval process must stop. In case the filter couldn't give certainty, then the L1 layer features are compared initially. The comparison is carried out by finding the nearest neighbours based on the cosine distances of two feature vectors and a threshold as calculated previously. Once, we get a refined search space, the process is repeated with finer thresholds for the L2 layer search vectors and finally the best structural matches are found by comparing the L3 layer feature vectors.

The complete algorithm is briefly represented by Figure 2.

5.

IV. EXPERIMENTATION AND RESULTS

The proposed methodology was tested for mean average precision using the Oxford 5k, Paris 6k and the Holidays Dataset. Also, distractor images from the Flickr 100 dataset were used to test the robustness of the system with garbage data. Final feature maps were of the size 128 since they proved to be a good agreement between efficiency and accuracy [2]. The final reported accuracy was based on the mean average precision(mAP) for the system.

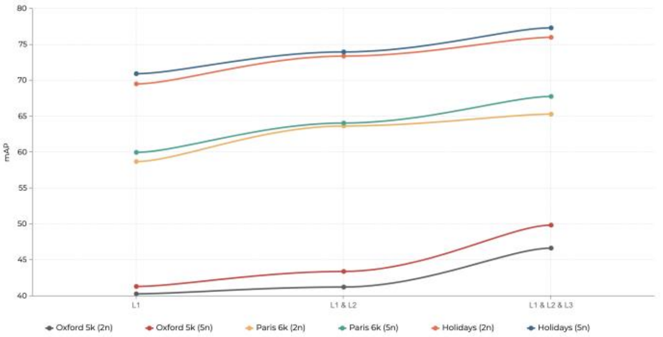
Dataset	**Oxford 5k**		**Paris 6k**		**Holidays**	
Filters	**2n**	**5n**	**2n**	**5n**	**2n**	**5n**
L1	40.31 1.48	41.34 1.21	58.73 1.97	60.01 1.55	69.54 30.60	70.96 27.04
L1 & L2	41.26 1.26	43.43 0.98	63.68 1.37	64.09 1.26	73.41 28.45	73.99 23.67
L1 & L2 & L3	46.68 1.34	49.88 1.15	65.33 1.19	67.79 1.13	76.02 23.26	77.34 20.89

6.

7. *Table 1: This table contains information about the mAP and the average time(in s) for the experiments carried out on the Oxford 5k, Paris 6k and the Holidays dataset without any*

distractor image. n is the number of images stored in the database. L1, L2 and L3 are the layers of feature vectors. The threshold for binary sequencing was fixed at 10 for a 64 bit sequence.

The first setup consisted of training the network using only the dataset images. Tests on every dataset were carried out first using only layer L1 as the feature vector followed byL1 and L2 and finally all the three. The size of the bloom filter was set to be 2n and 5n, n being the number of data items in the bloom filter. The threshold for binary sequencing was fixed to be 10 and the number of centroids were fixed to be 64. It was seen that as the number of feature vectors used for input increased, the accuracy increased generally. Moreover This increase in accuracy was also seen as the size of the filter was increased. We see that the performance was particularly good as compared to the current state-of-the-art with a very minimal compute time. The time required for computation generally decreased with the increase in size of the filters and also with the increase in the number of hashes used.



8. *Figure 3: This graph presents a visual representation of the data presented in Table 1. The horizontal axis represents the number of feature vectors whereas the vertical axis corresponds to the mAP.*

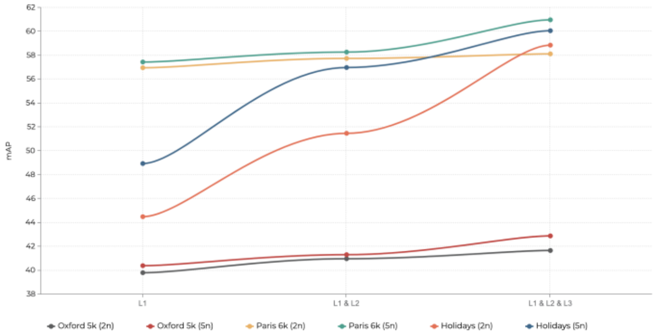
Dataset	**Oxford 5k**		**Paris 6k**		**Holidays**	
Filters	**2n**	**5n**	**2n**	**5n**	**2n**	**5n**
L1	39.81 2.89	40.40 1.58	56.97 4.36	57.45 2.98	44.50 42.60	48.95 40.76
L1 & L2	40.98 1.67	41.32 1.19	57.75 3.74	58.28 2.12	51.48 39.87	56.99 34.49
L1 & L2 & L3	41.67 1.42	42.89 1.27	58.13 2.96	60.98 1.92	58.86 35.21	60.07 33.47

9.

10. *Table 2: This table contains information about the mAP and the average time(in s) for the experiments carried out on the Oxford 5k, Paris 6k and the Holidays datasets mixed with the distractor images. n is the number of images stored in the database. L1, L2 and L3 are the layers of feature vectors. The threshold for binary sequencing was fixed at 10 for a 64 bit sequence.*

The second setup included mixing the datasets along with the distractor images. All the three datasets were mixed andthe model was trained on all the training classes. Tests were carried out simultaneously using bloom filter size twice and five times the sizes of the input set.

The test results were almost consistent with before. The Accuracy was seen to dip significantly on using a single descriptor but it performed considerably well when all the descriptors were used. The overall speed of execution was reasonably faster due to a large number of false positives which got identified during early iterations.



11. *Figure 4: This graph presents a visual representation of the data presented in Table 2. The horizontal axis represents the number of feature vectors whereas the vertical axis corresponds to the mAP.*

V. CONCLUSION

In this paper, we present a fresh approach for efficient image retrieval by using multiple neural hash codes which are PCA compressed and binary sequenced before feeding them to a bloom filter to reduce the average number of queries. The feature vectors are also used in the retrieval process which follows a hierarchical coarse-to-fine approach by using the higher layers for coarse search and the lower layers for a finer search. The followed approach shows precisions matching state-of-the-art while reducing the number of queries and hence mean query times greatly.

Further improvements made to the algorithm can be towards trying out other state-of-the-art networks with a similar approach or experimenting with different feature compression processes. Modern image matching processes suited with multiple feature vectors can also be experimented with to increase accuracy.

REFERENCES

[1] alvi, A., Ercoli, S., Bertini, M. and Del Bimbo, A., 2016, December. Bloom filters and compact hash codes for efficient and distributed image retrieval. In 2016 IEEE International Symposium on Multimedia (ISM) (pp. 515-520). IEEE.

[2] Babenko, A., Slesarev, A., Chigorin, A. and Lempitsky, V., 2014, September. Neural codes for image retrieval. In European conference on computer vision (pp. 584-599). Springer, Cham.

[3] Yu, W., Sun, X., Yang, K., Rui, Y. and Yao, H., 2018. Hierarchical semantic image matching using CNN feature pyramid. Computer Vision and Image Understanding, 169, pp.40-51.

[4] Yue-Hei Ng, J., Yang, F. and Davis, L.S., 2015. Exploiting local features from deep networks for image retrieval. In Proceedings of the IEEE conference on computer vision and pattern recognition workshops (pp.53-61).

[5] Ercoli, S., Bertini, M. and Del Bimbo, A., 2015, June. Compact hash codes and data structures for efficient mobile visual search. In 2015 IEEE International Conference on Multimedia & Expo Workshops (ICMEW) (pp. 1-6). IEEE.

[6] Babenko, A. and Lempitsky, V., 2015. Aggregating deep convolutional features for image retrieval. arXiv preprint arXiv:1510.07493.

[7] Radenovi´c, F., Tolias, G. and Chum, O., 2018. Fine-tuning CNN image retrieval with no human annotation. IEEE transactions on pattern analysis and machine intelligence, 41(7), pp.1655-1668.

[8] Rae, J.W., Bartunov, S. and Lillicrap, T.P., 2019. Meta-learning neural Bloom filters. arXiv preprint arXiv:1906.04304.

[9] Araujo, A., Chaves, J., Lakshman, H., Angst, R. and Girod, B., 2016. Large-scale query-by-image video retrieval using bloom filters. arXivpreprint arXiv:1604.07939.

[10] Xia, Z., Feng, X., Lin, J. and Hadid, A., 2017. Deep convolutional hashing using pairwise multi-label supervision for large-scale visualsearch. Signal Processing: Image Communication, 59, pp.109-116.

[11] Uricchio, T., Bertini, M., Seidenari, L. and Bimbo, A., 2015. Fisher encoded convolutional bag-of-windows for efficient image retrieval and social image tagging. In Proceedings of the IEEE International Conference on Computer Vision Workshops (pp. 9-15).

[12] J´egou, H. and Zisserman, A., 2014. Triangulation embedding and democratic aggregation for image search. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3310-3317).

[13] Sharif Razavian, A., Azizpour, H., Sullivan, J. and Carlsson, S., 2014. CNN features off-the-shelf: an astounding baseline for recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition workshops (pp. 806-813).

[14] Inoue, K. and Kise, K., 2009, September. Compressed representation of feature vectors using a Bloomier filter and its application to specific object recognition. In 2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops (pp. 2133-2140).

[15] Bloom, B.H., 1970. Space/time trade-offs in hash coding with allowable errors. Communications of the ACM, 13(7), pp.422-426.

[16] Arandjelovic, R. and Zisserman, A., 2013. All about VLAD. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (pp. 1578-1585).

[17] Lowe, D.G., 2004. Distinctive image features from scale-invariant key-points. International journal of computer vision, 60(2), pp.91-110.

[18] Philbin, J., Chum, O., Isard, M., Sivic, J. and Zisserman,A., 2007, June. Object retrieval with large vocabularies and fast spatial matching. In 2007 IEEE conference on computer vision and pattern recognition (pp. 1-8). IEEE.

[19] Philbin, J., Chum, O., Isard, M., Sivic, J. and Zisserman,A., 2008, June. Lost in quantization: Improving particular object retrieval in large scale image databases. In 2008 IEEE conference on computer vision and pattern recognition (pp. 1-8). IEEE.

[20] Jegou, H., Douze, M. and Schmid, C., 2008, October. Hammingembedding and weak geometric consistency for large scale image search. In European conference on computer vision (pp. 304-317). Springer, Berlin, Heidelberg.

[21] LeCun, Y., Boser, B.E., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W.E. and Jackel, L.D., 1990. Handwritten digit recognition with a back-propagation network. In Advances in neural information processing systems (pp. 396-404).

[22] Jogin, M., Madhulika, M.S., Divya, G.D., Meghana, R.K.and Apoorva, S., 2018, May. Feature extraction using Convolution Neural Networks (CNN) and Deep Learning. In 2018 3rd IEEE International Conferenceon Recent Trends in Electronics, Information & Communication Technology (RTEICT) (pp. 2319-2323). IEEE.

[23] Ng, S.C., 2017. Principal component analysis to reduce dimension on digital image. Procedia computer science, 111, pp.113-119.

[24] S´anchez, J., Perronnin, F., Mensink, T. and Verbeek, J., 2013. Image Classification with the fisher vector: Theory and practice. International Journal of computer vision, 105(3), pp.222-245.

[25] Yang, L. and Jin, R., 2006. Distance metric learning: A comprehensive survey. Michigan State University, 2(2), p.4.

[26] Gao, L., Song, J., Zou, F., Zhang, D. and Shao, J., 2015, October. Scalable multimedia retrieval by deep learning hashing with relative similarity learning. In Proceedings of the 23rd ACM international conference on Multimedia (pp. 903-906).

[27] Lin, K., Yang, H.F., Hsiao, J.H. and Chen, C.S., 2015. Deep learning binary hash codes for fast image retrieval. In Proceedings of the IEEE conference on computer vision and pattern recognition workshops (pp.27-35).

[28] Sun, J., Shum, H.Y. and Zheng, N.N., 2002, May. Stereo matching using belief propagation. In European Conference on Computer Vision (pp.510-524). Springer, Berlin, Heidelberg.

[29] Boykov, Y., Veksler, O. and Zabih, R., 2001. Fast approximate energy minimization via graph cuts. IEEE Transactions on pattern analysis and machine intelligence, 23(11), pp.1222-1239.

[30] Kanazawa, A., Jacobs, D.W. and Chandraker, M., 2016. Warpnet: Weakly supervised matching for single-view reconstruction. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 3253-3261).

[31] Barnes, C., Shechtman, E., Goldman, D.B. and Finkelstein, A., 2010, September. The generalized patchmatch correspondence algorithm. In European Conference on Computer Vision (pp. 29-43). Springer, Berlin, Heidelberg.

[32] Danielsson, O., 2015, June. Category-sensitive hashing and Bloom filter based descriptors for online keypoint recognition. In Scandinavian Conference on Image Analysis (pp. 329-340). Springer, Cham.

[33] Appleby, A., 2011. Murmur3 hash function.

[34] Berg, A. and Deng, J., 2010. and L Fei-Fei. Large scale visual recognition challenge(ILSVRC).

[35] Babenko, A. and Lempitsky, V., 2014. The inverted multi-index. IEEE transactions on pattern analysis and machine intelligence, 37(6), pp.1247-1260