Image Deduplication Using Efficient Visual Indexing and Retrieval: Optimizing Storage, Time and Energy for Deep Neural Network Training

1st M M Mahabubur Rahman
Texas State University
San Marcos, TX-78666, USA
toufik@txstate.edu

2nd Debojyoti Biswas
Texas State University
San Marcos, TX-78666, USA
debojyoti_biswas@txstate.edu

3rd Xiao Chen
Texas State University
San Marcos, TX-78666, USA
xc10@txstate.edu

4th Jelena Tešić
Texas State University
San Marcos, TX-78666, USA
jtesic@txstate.edu

***Abstract*—**

***Index Terms*—Cloud storage, Data redundancy, Mobile crowdsensing, Image deduplication.**

I. Introduction

Leveraging vast amounts of data, deep neural networks have made significant strides in natural language processing, computer vision, and reinforcement learning. Specifically, Convolutional Neural Networks (CNNs) have demonstrated near-human performance across various vision tasks, mainly owing to the abundance of image data. Popular datasets like CIFAR100 [], ImageNet [], and COCO [] encompass thousands to millions of images.

Typically, larger datasets result in higher accuracy and greater robustness in trained networks. However, the quality of these datasets remains unassessed. Among million-scale datasets, redundancy is inevitable. Take the SVHN [] dataset, for instance, which comprises cropped digits from street view house number plates. It is reasonable to expect repetitive images featuring similar plate materials and identical printed digits. These redundant images are often referred to as duplicates or near-duplicates. First, we will establish the definition of duplicate or near-duplicate images.

Definition 1: *Duplicate or near-duplicate images.* An image, denoted as I', is considered a duplicate or near duplicate of another image, denoted as I, if I' is obtained from I through a set of tolerated transformations, represented as T_r, such that,

$$I' = T_r\left(I\right) \tag{1}$$

A substantial volume of redundant data presents challenges such as prolonged training times, increased storage requirements, and higher energy consumption [17]. Addressing these challenges necessitates the development of an efficient method capable of identifying and removing redundant data while preserving accuracy to a significant extent. In simpler terms, this involves training a Deep Neural Network (DNN) on a smaller yet robust dataset.

Image features serve as descriptive representations of the images, capturing crucial visual attributes invariant scale, angle, and appearance in high dimensions. Deep features play a vital role in quantifying the similarity between images and facilitating the identification of redundant data. By mapping images into a feature space, the similarity of any image pair can be quantified through their corresponding feature distance. The more similar the feature vectors are, the closer they are in proximity.

Definition 2: *Similarity measure.* $D(X, Y)$ denotes the distance between feature vectors X and Y, where x_i is the i^{th} component in feature vector X.

$$D_p = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p} \tag{2}$$

When $p = 2$, $D_2(X, Y)$ is known as Euclidean distance. The Euclidean distance-based similarity search is a baseline as it is the most robust way to capture the similarity between deep features [18]. Which leads to define our deduplication rule.

Definition 3: *Deduplication rule.* Images which satisfy the following rule, shall be treated as duplicate images.

$$D_2\left(X, Y\right) \leq T \tag{3}$$

T represents the threshold for determining image similarity and can be manually adjusted.

Hence, an efficient method is required to navigate through the feature database of representative images and locate the most analogous items, which can then be identified as duplicates or near duplicates according to a predefined threshold and subsequently discarded. Nonetheless, effectively indexing or retrieving images from a vast image database presents significant challenges [8].

In this paper, we propose a novel approach called **V**isual **I**ndexing and **R**etrieval-based image **D**eduplication (VIRD) to address the issue of redundant image data. VIRD aims to efficiently identify and eliminate duplicate or near duplicate images from the image database, preserving the data integrity. Figure 1 illustrates the dynamic deduplication process. We propose to extract deep features from new images, compare them to the stored features in the database, and decide whether to store or discard the feature and image. The incoming query image's redundancy status is based on the metrics

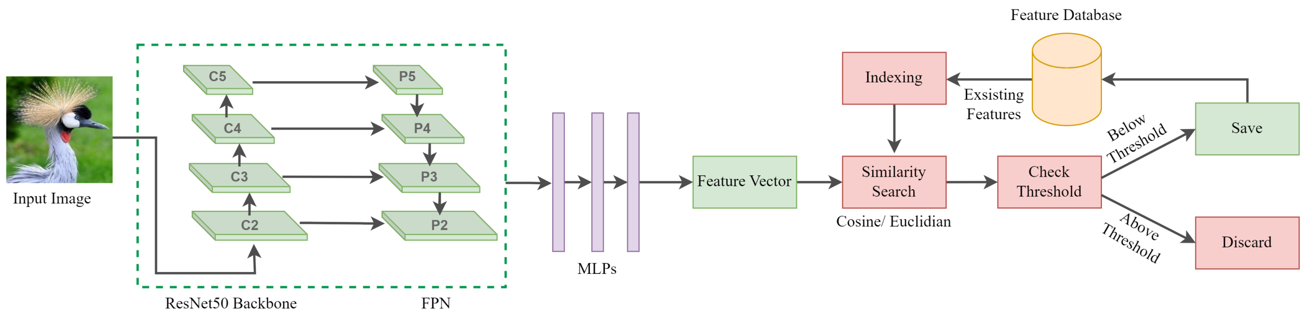

Fig. 1. Pipeline of proposed VIRD architecture for image deduplication.

threshold and can vary based on storage requirements and DNN performance requirements. We propose a Hierarchical Layered Graph (HLG), an Approximate Nearest Neighbor Search (ANNS) to retrieve the most similar item from the feature database. Image features serve as descriptive representations of the images, capturing crucial visual attributes invariant scale, angle, and appearance in high dimensions, and here we propose to index deep descriptors in the image archive and update the index periodically to support efficient and effective approximate nearest-neighbor searches that can scale to billions of items [7]. Furthermore, the suggested approach of integrating ANNS into a deep feature database has minimal impact on the pre-trained network and training data. This is because ANNS proves to be efficient in uncovering unknown classes within the database [19]. This approach enables the efficient detection and elimination of redundant images while preserving the essential visual information necessary for accurate model training. By eliminating redundant data, this method not only enhances computational efficiency but also facilitates better utilization of computational resources, ultimately leading to more practical and sustainable DNN training practices.

Our contributions in this paper are:

1) **V**isual **I**ndexing and **R**etrieval-based image **D**eduplication (VIRD) method. It is the first method that utilizes graph-based approximate nearest neighbor search in deep features to accomplish the image deduplication task.
2) Hierarchical Layered Graph (HLG) is a graph-based approximate nearest neighbor indexing and search method that efficiently stores and retrieves the most similar image descriptors from large data storage.
3) To assess the effectiveness of a deduplication method, we introduce a novel metric termed ”Deduplication efficiency.” This metric is derived from the percentage of data reduction achieved and the percentage of accuracy drop observed.

The rest of this article is organized as follows. Section II summarizes related work, and section III discusses our proposed VIRD method and training pipeline in detail. Next, Section IV describes the experimental data set distribution and characteristics. In Section V, we evaluate the proposed framework and show our experimental results by comparing it with the latest deduplication benchmarks over four consumer and crowd-sensing data sets. Finally, we summarize the findings in Section VI.

II. Related Work

Near-exact or near-duplicate images are images that have undergone modifications like cropping, scaling, or rotation from an original image. Existing research on near-exact image detection varies in the feature vectors used to represent images and the indexing techniques applied. Methods for image deduplication can be broadly categorized into traditional image feature-based and deep feature-based approaches.

In traditional image feature-based methods, various techniques have been proposed. Velmurugan et al. [22] utilized the Speeded-up Robust Features (SURF) algorithm and a k-dimensional (KD) tree for indexing and matching similar image features. Lei et al. [13] introduced a cluster of uniform randomized trees for rapid near-duplicate image detection. Yu et al. [24] presented a SIFT-based geometric transformation fingerprinting technique, while Li et al. [14] proposed an image-matching algorithm using SURF feature points and daisy descriptors. Foo et al. [5] developed a similar image collator (SICO) utilizing PCA-SIFT and LSH for feature indexing. Chen et al. [4] introduced a rapid image retrieval method converting features into binary representations.

In contrast, deep feature-based methods leverage Convolutional Neural Networks (CNNs) for image deduplication. For instance, Kaur et al. [8] proposed a CNN-based online image deduplication technique, showing superior performance in detecting exact and near-exact images. Kordopatis-Zilos et al. [9] introduced a method for near-duplicate video retrieval using unsupervised and supervised approaches based on Deep Metric Learning (DML). Liang et al. [16] presented a hierarchical detection method utilizing CNN models and semantic features for near-duplicate video identification.

III. Methodology

Exact feature matching becomes progressively slow when dealing with high-dimensional features and large databases [10]. To address the camera angle and orientation problem, VIRD trains a deep neural network (DNN) model with various images in different orientations, lighting conditions, and angles. The trained DNN model captures near similarities during the feature extraction step. After retrieving the most similar feature using a Hierarchical Layered Graph (HLG) search, VIRD employs Euclidean distance in the feature space to determine the similarity of an incoming image. If the incoming image is considered sufficiently similar to the retrieved feature based on a predetermined threshold, it is classified as redundant and can be discarded. Conversely, if the image is distinctive enough, it is considered unique. The feature is then added to the existing feature database, and the incoming image is stored in the image database.VIRD effectively reduces storage requirements and processing overhead associated with redundant data. By eliminating duplicate or near-duplicate images, the dataset becomes more compact and efficient, improving visual applications' overall performance.

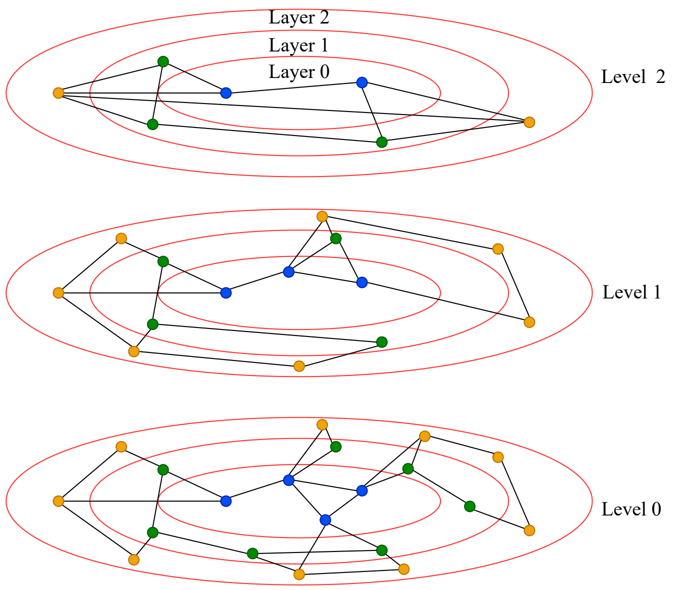

Fig. 2. Hierarchical Layered Graph (HLG) indexing: each feature vector connects to its M-Nearest Neighbor within the same layer and 1-Nearest Neighbor in the next layers.

The proposed pipeline of the VIRD method consists of two main phases. The first phase involves feature extraction in deep descriptor space, which is discussed in detail later. This process is done incrementally for one image at a time, passing each image to the DNN for feature extraction and generating a feature vector for similarity search. Existing features from the feature database, indexed using the HLG for faster search processing, are then loaded. The second phase employs an approximate search in deep descriptor space searching method to find the most similar image features concerning the query feature. During the approximate search using HLG, the most similar image feature is retrieved from the feature database for an incoming query image feature.

A. Feature extraction with Deep Neural Network (DNN)

The first step toward eliminating data redundancy is efficiently representing the data. Many techniques have been devised to describe the image data in vector format; using DNN to extract feature vectors is the most popular method. The success of DNNs for feature extraction is mainly due to the availability of the data and the computational power. Based on the previous success of DNNs [1], [2] for object detection in consumer and aerial images, we have chosen to use ResNet50 architecture for generating feature vectors for the image data. Figure 1 shows that the ResNet50 model is built upon many Convolution(Conv) blocks stacked one after one. The first seven blocks in the ResNet50 network are Convolutional (Conv) blocks with 64 approximate searches in profound descriptor space el outputs and only one stride at the beginning. Then, the next block starts with a Conv block with a stride of 2 and an approximate output search in a deep descriptor space of 128. This CNN fashion follows onward with 256 and 512 output approximate searches in deep descriptor space. Next, we perform average pooling on the last Conv layer output. Finally, we feed the output from the average pool into Multi-layered perceptions (MLPs) and save the output from this layer as a feature in our database in 512 lengths of a vector.

```
Algorithm 1: BUILD$(HLG, X, M, and, f)$
  Input: hierarchical layered graph HLG, data vector X,
         number of established connections M, size of
         dynamic candidate list cand, outlier filtering
         factor f
  Output: Update HLG inserting all elements
1 $graph \leftarrow \phi$
2 foreach $x$ of $X$ do
3 |   $graph \leftarrow$ ADD$(x, M, cand)$
4 end
5 $layeredElem \leftarrow$ LAYERING $(X, M, f)$ //Algorithm
    2
6 foreach $layer$ of $layeredElem$ do
7 |   $clg \leftarrow$ get the graph for $layer$
8 |   $nlg \leftarrow$ get the graph for $(layer + 1)$
9 |   foreach $elem$ of $layer$ do
10|   |   $n \leftarrow$ SEARCH$(nlg, elem, k = 1, cand = 1)$
11|   |   update $graph$ inserting $n$ to neighbor list of
  |   |     $elem$
12|   end
13 end
```

B. Hierarchical Layered Graph (HLG)

HLG index building Figure 2 illustrates the index structure of our proposed HLG approach. The HLG first arranges all feature vectors on a hierarchical level where the higher level contains fewer feature vectors and the lower level contains more feature vectors. A probability function, $P(L_v) = F(L_v, l_m)$, is used to determine the level of insertion of an element. The value L_v denotes the level at which an element will be inserted. The probability function normalized by the "level multiplier" l_m, where l_m=0 indicates that vectors are only inserted in level 0, gives the probability of a vector insertion in a given level. We achieve the highest performance when we reduce the overlap of shared neighbors between levels. We can reduce overlap by decreasing l_m. However, doing so increases the average number of search traversals as more vectors are moved to the level 0. After generating the levels, the HLG arranges the feature vectors in layers based on their distances from the centroid, where layer 0 contains the feature vectors that appear to be closer to the centroid and layer L contains the feature vectors that are the farthest from the centroid. The bidirectional graph is constructed by connecting each feature vector to its M-Nearest Neighbor within the same layer and 1-Nearest Neighbor in the next layers. Therefore, the feature vectors of layer $0, 1, 2, ..., L$ will have $M + L - 1, M + L - 2, ..., M$ neighboring nodes in the final graph. The value of M is responsible for the index size and recall. Typically, the optimal value of M ranges from 5 to 48, whereas a larger value of M leads to a larger index size and higher recall.

Indexing complexity The HLG index is constructed in two steps. In the first step, each element is added one at a time by iterative insertions, simply a series of approximate searches in deep descriptor space searches at different levels. Thus, the first phase has a complexity of $O\big(N.log(N)\big)$. The second phase of HLG index building is also a series of approximate searches in deep descriptor space searches at different layers. Thus, similar to the first phase, the second phase has a complexity of $O\big(N.log(N)\big)$. Therefore, the overall complexity of the index building of the HLG scales as $O\big(N.log(N)\big)$. Algorithm 1 outlines the two steps of the index construction,

Algorithm 2: LAYERING (X, M, f)

```
Input: data vector X, number of established
       connections M, outlier filtering factor f
Output: Dictionary of elements with their designated
        layer
1  $numLayer \leftarrow \lfloor \log_2 M \rfloor$
2  $cen \leftarrow$ mean of $X$
3  $dist \leftarrow$ distances from the centroid to all data vectors
4  $avg \leftarrow$ mean of all distances
5  $\sigma \leftarrow$ standard deviation of all distances
6  $u_b \leftarrow avg + f \times \sigma$
7  $l_b \leftarrow$ smallest of $dist$
8  $r \leftarrow \frac{u_b - l_b}{numLayer}$
9  $layeredElem \leftarrow \phi$
10 foreach $(d, x)$ $of$ $(dist, X)$ do
11     $l \leftarrow \lfloor \frac{d}{r} \rfloor$
12     add element $x$ to layer $l$ in $layeredElem$
13 end
14 return $layeredElem$
```

as it builds a hierarchical level of proximity graphs within the same layer and then adds the connections to the next layer. The maximum number of levels in the hierarchical graph can be controlled by the maximum established connection parameter M. Then, the exponential decaying probability distribution $\big(\lfloor -log_2\big(unif(0, ..., 1) \times m_l\big)\rfloor\big)$ determines the maximum level for each element, where m_l is $\frac{1}{log_2(M)}$. The insertion process begins at the top level and traverses the graph greedily to locate the closest $cand$ neighbors of the inserted element x. The process is then repeated, utilizing the closest neighbors obtained at the previous level as entry points for the algorithm to carry on with the search from the subsequent level. Algorithm 2 determines the layers of each element based on their distances from the centroid. The control parameter f is used as an outlier filtering factor during the layer determination process. Elements that are f standard deviations from the mean distance do not participate in the layer determination process. The outlier filtering factor f ensures the outliers do not drag the layer boundaries toward them. Next, we extract the graphs for each layer from the previously constructed network.

Algorithm 3: SEARCH$(g, q, k, cand)$

```
Input: graph index g, query element q, number of
       nearest neighbors k, size of dynamic candidate
       list cand
Output: k closest neighbors to q
1 $ep \leftarrow$ get entry point of $g$
2 $L \leftarrow$ get highest level of $g$
3 for $l \in L, L-1, \ldots, 2$ $of$ $g$ do
4     $p \leftarrow$ extract nearest neighbor to $q$ starting with $ep$
5     $ep \leftarrow p$
6 end
7 $C \leftarrow$ extract $cand$ neighbors to $p$ at bottom level of $g$
8 $neighbors \leftarrow$ top $k$ closest from $C$ to $q$
9 return neighbors
```

HLG searching Figure 3 shows the k-NN retrieval approach of our proposed Hierarchical Layered Graph (HLG) approach. The search within an index starts with a random point at the upper level where the edges are the longest, and then a greedy search is used within that level until it reaches a local optimum (Figure 3). The search then switches to the lower level, with shorter edges. This time, the starting point is the previous local optimum, and this process continues until the query is reached and the top k-NN to the top k is returned. Layering helps the HLG avoid visiting all neighboring nodes within the same layer if the query is in a different layer in the feature space. Moreover, the HLG search skips visiting nodes in layers as well if the current node and query are some layers apart from each other in the feature space (Algorithm 3 Line 4). Algorithm 3 identifies the closest nearest neighbors for each inner layer in the subsequent layers using the greedy search algorithm, and we update the previous network by adding the closest Neighbor found in the subsequent layers. Algorithm 3 starts at the top level with the entry point ep of the input network and extracts the closest neighboring point p to the incoming query q at that level. Then, the search switches to the next lower level and starts with the previous local optimum p, and this process continues until the second lowest level. At

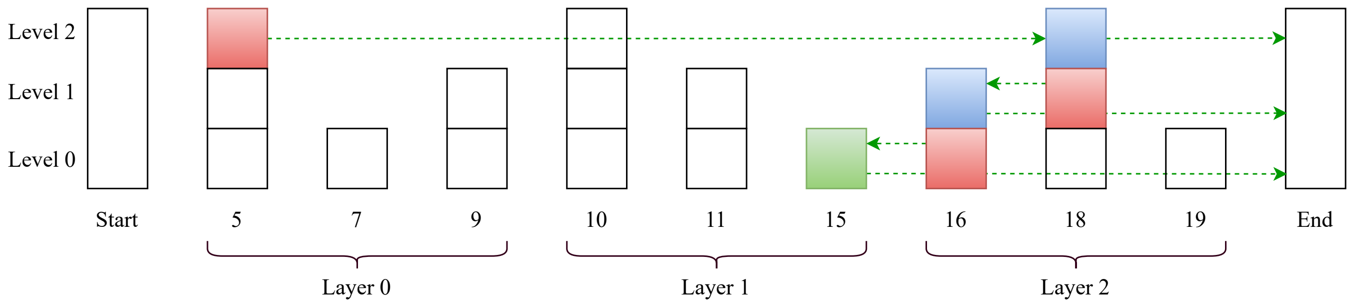

Fig. 3. Illustration of a HLG retrieval. Red denotes the starting point in each level, blue denotes the local optimum in each level, and green arrows show the direction of the greedy algorithm to the query(shown green).

the bottom level of the network, g, the algorithm extracts the list of neighbors $cand$ from p and returns the closest neighbors k to q based on their distances.

Search complexity Each HLG index level is built as a navigable small-world graph, allowing the greedy search path's hop count to scale logarithmically. HLG indexing builds the graph with a set maximum number of links for each element, ensuring a consistent average degree for each element at a certain level. The number of hops and the average degree of the items on the greedy path are multiplied to get the overall amount of distance calculations. As a result, each level of the HLG has logarithmic search complexity. At any given level l with N_l elements, the search complexity is $O(log(N_l))$, where N_l increases from the top to the bottom. The maximum number of elements allowed at the bottom level is N. Therefore, the general search complexity of the HLG is determined by $O(log(N))$.

IV. Setup

A. Data

VIRD method is evaluated on three benchmark datasets: ImageNet [20], cifar10 and cifar100 [11], and one crowd-sensing iNaturalist-Birds dataset [6]. Each data set comes with various classes showing the data set's diversity, as summarized in Table I.

TABLE I
Experimental data set for image deduplication.

Data	# images	DB_{orig}(MB)	Number of classes
cifar10 [12]	50,000	47.4	10
cifar100 [12]	50,000	114.2	100
Birds	70,626	1600	450
ImageNet	386,291	45158	1000

B. Experimental Setup

C. Evaluation Measures

To assess the effectiveness of our deduplication method, we utilize three evaluation criteria: Deduplication Efficiency (DE), Training time, and Total energy consumption.

Deduplication efficiency Given the absence of ground truth for actual duplicates or near-duplicates in our experimental

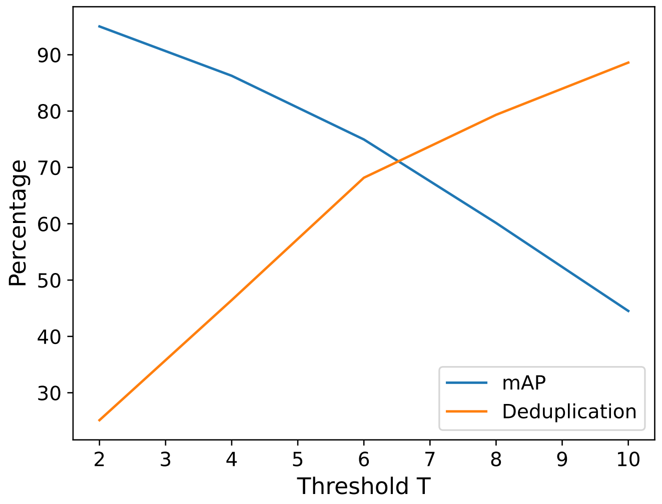

Fig. 4. Effect of Threshold distance T on image deduplication and mAP for Cifar10 dataset.

image datasets, to measure the effectiveness of comparing methods, we introduce a new metric termed *Deduplication efficiency* (DE). Which can be obtained by

$$DE = (1-\alpha) \times \textit{Dedup} + \alpha \times (1 - mAP_{drop}) \qquad (4)$$

Here, *Dedup* represents the percent of duplicate data elimination, where a higher value indicates greater reduction in redundant data. The mAP_{Drop} is the percentage decrease in mean average precision (mAP) resulting from the DNN training on the deduplicated data. α is a weighting factor between 0 and 1 that determines the relative importance of mAP_{Drop} compared to deduplication percentage. It allows us to adjust the trade-off between deduplication and mAP. Higher α values penalize mAP loss more heavily. The *Deduplication efficiency* metric is designed such that higher values indicate better performance, as it rewards higher levels of data reduction while penalizing mAP_{Drop}.

Training time metric for DNN training refers to the duration it takes to train a deep neural network model on a given dataset. It measures the total elapsed time from the start of the training process until the model has been fully trained.

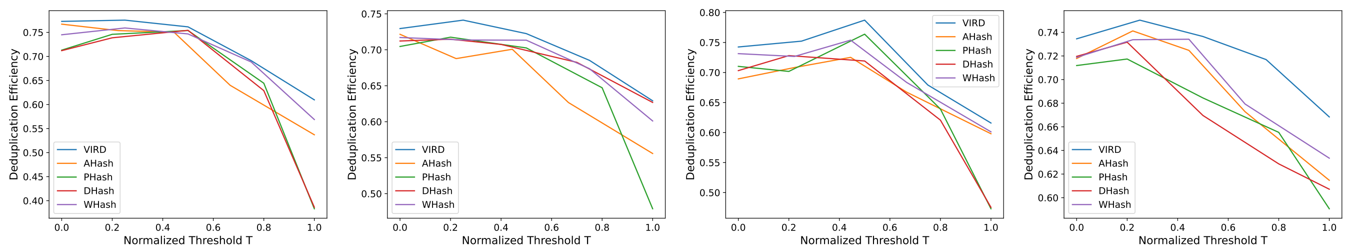

Fig. 5. Deduplication efficiency for comparing methods on Cifar10, Cifar100, Birds and Imagenet datasets (left to right) at different threshold values.

Total energy consumption metric for DNN training quantifies the amount of energy consumed during the entire process of training a deep neural network model on a given dataset. It includes energy usage by computational hardware (such as CPUs or GPUs), memory storage devices, and associated cooling systems.

V. Experiments

Our work is closely aligned with CEDedup [15], which utilizes hash-based feature extraction techniques for efficient image deduplication. Our work is inspired by CEDedup, however, we criticize the hash-based feature extraction techniques for image deduplication. Unlike CEDedup, our approach focuses on deep feature extraction and retrieval, using graph-based indexing methods for improved efficiency in deep neural network training. CEDedup relies on the efficiency of hash codes generated by different hashing algorithms to identify and remove duplicates. We argue that deep features offer a superior representation of an image compared to hash codes. To prove our claim, we compare the proposed VIRD approach with four different state-of-the-art hashing algorithms namely WHash [21], PHash [25], DHash [23] and AHash [3] in terms of Deduplication efficiency.

VI. Conclusion

References

[1] Biswas, D., Rahman, M.M., Zong, Z., Tešić, J.: Improving the energy efficiency of real-time dnn object detection via compression, transfer learning, and scale prediction. In: 2022 IEEE International Conference on Networking, Architecture and Storage (NAS). pp. 1–8 (2022). https://doi.org/10.1109/NAS55553.2022.9925528

[2] Biswas, D., Tešić, J.: Small object difficulty (sod) modeling for objects detection in satellite images. In: 2022 14th International Conference on Computational Intelligence and Communication Networks (CICN). pp. 125–130. IEEE (2022)

[3] Chamoso, P., Rivas, A., Martín-Limorti, J.J., Rodríguez, S.: A hash-based image matching algorithm for social networks. In: Trends in Cyber-Physical Multi-Agent Systems, Proceedings of the 15th Int; Conf, PAAMS 2017 15. pp. 183–190. Springer (2018)

[4] Chen, C.C., Hsieh, S.L.: Using binarization and hashing for efficient sift matching. Journal of Visual Communication and Image Representation **30**, 86–93 (2015)

[5] Foo, J.J., Sinha, R., Zobel, J.: Sico: a system for detection of near-duplicate images during search. In: 2007 IEEE International Conference on Multimedia and Expo (ICME). pp. 595–598. IEEE (2007)

[6] iNaturalist: inaturalist is a joint initiative of the california academy of sciences and the national geographic society (December 2022), `https://www.inaturalist.org/observations`

[7] Johnson, J., Douze, M., Jégou, H.: Billion-scale similarity search with gpus. IEEE Transactions on Big Data **7**(3), 535–547 (2019)

[8] Kaur, R., Bhattacharya, J., Chana, I.: Deep cnn based online image deduplication technique for cloud storage system. Multimedia Tools and Applications **81**(28), 40793–40826 (2022)

[9] Kordopatis-Zilos, G., Papadopoulos, S., Patras, I., Kompatsiaris, I.: Finding near-duplicate videos in large-scale collections. Video Verification in the Fake News Era pp. 91–126 (2019)

[10] Kouiroukidis, N., Evangelidis, G.: The effects of dimensionality curse in high dimensional knn search. In: 2011 15th Panhellenic Conference on Informatics. pp. 41–45. IEEE (2011)

[11] Krizhevsky, A.: Learning multiple layers of features from tiny images. Technical Report, University of Toronto, Canada (2009)

[12] Krizhevsky, A., Nair, V., Hinton, G.: Cifar-10 and cifar-100 datasets (2006), `http://www.cs.toronto.edu/ kriz/cifar.html`

[13] Lei, Y., Qiu, G., Zheng, L., Huang, J.: Fast near-duplicate image detection using uniform randomized trees. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) **10**(4), 1–15 (2014)

[14] Li, L., Zic, J.: Image matching algorithm based on feature-point and daisy descriptor. Journal of Multimedia **9**(6), 829–834 (2014)

[15] Li, X., Chang, L., Liu, X.: CEDedup: Cost-effective convolutional neural nets training based on image deduplication. In: 2021 IEEE Intl Conf SPA/BDCloud/SocialCom/SustainCom. pp. 11–18. IEEE (2021)

[16] Liang, S., Wang, P.: An efficient hierarchical near-duplicate video detection algorithm based on deep semantic features. In: 26th International Conference on Multimedia Modeling. pp. 752–763. Springer (2020)

[17] Nbt, Y., Ismail, A., Majid, N.: Deduplication image middleware detection comparison in standalone cloud database. Int J Adv Comput Sci Technol (IJACST) **5**(3), 12–18 (2016)

[18] Rahman, M.M., Tešić, J.: Evaluating hybrid approximate nearest neighbor indexing and search (hannis) for high-dimensional image feature search. In: 2022 IEEE Intl. Conf. on Big Data. pp. 6802–6804 (2022). https://doi.org/10.1109/BigData55660.2022.10021048

[19] Rahman, M.M., Tešić, J.: Hybrid approximate nearest neighbor indexing and search (hannis) for large descriptor databases. In: 2022 IEEE Intl. Conf. on Big Data. pp. 3895–3902 (2022). https://doi.org/10.1109/BigData55660.2022.10020464

[20] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International journal of computer vision **115**, 211–252 (2015)

[21] Singh, S.P., Bhatnagar, G.: A robust image hashing based on discrete wavelet transform. In: 2017 IEEE International Conference on Signal and Image Processing Applications (ICSIPA). pp. 440–444. IEEE (2017)

[22] Velmurugan, K., Baboo, L.: Content-based image retrieval using surf and colour moments. Global Journal of Computer Science and Technology **11**(10), 1–4 (2011)

[23] Wang, J., Fu, X., Xiao, F., Tian, C.: Dhash: Enabling dynamic and efficient hash tables. arXiv preprint arXiv:2006.00819 (2020)

[24] Yu, X., Huang, T.: A sift-based image fingerprinting approach robust to geometric transformations. In: 2009 IEEE International Symposium on Circuits and Systems. pp. 1665–1668. IEEE (2009)

[25] Zauner, C.: Implementation and benchmarking of perceptual image hash functions. Master Thesis, University of Applied Sciences, Hagenberg, Austria (2010)